On Individual and Aggregate TCP Performance

Lili Qiu, Yin Zhang, and Srinivasan Keshav
{lqiu, yzhang, skeshav}@cs.cornell.edu
Department of Computer Science
Cornell University, Ithaca, NY 14853

Abstract

As the most widely used reliable transport in today's Internet, TCP has been extensively studied in the past. However, previous research usually only considers a small or medium number of concurrent TCP flows. The TCP behavior under many competing TCP flows has not been sufficiently explored.

In this paper we use extensive simulations to investigate the individual and aggregate TCP performance for a large number of concurrent TCP flows. First, we develop a simple yet realistic network model to abstract an Internet connection. Based on the model, we study the performance of a single TCP flow with many competing TCP flows by evaluating the best-known analytical model proposed in the literature. Finally, we examine the aggregate TCP behavior and derive general conclusions about overall throughput, goodput, and loss probability.

1. Introduction

TCP is the most widely used reliable transport in today's Internet. It is used to carry a significant amount of Internet traffic, including WWW (HTTP), file transfer (FTP), email (SMTP) and remote access (Telnet) traffic. Due to its importance, TCP has been extensively studied in the past. However, previous research usually only considers a small or medium number of concurrent TCP flows. The behavior of many competing TCP flows has not been sufficiently explored.

In this paper, we use extensive simulations to explore the performance of TCP-Reno, one of the most common TCP flavors in today's Internet. We first develop a generic model for Internet connections by exploring the Internet hierarchical routing structure. Based on the abstract model, we study the behavior of a single TCP flow under many competing TCP flows by evaluating the TCP analytical model proposed in [12]. We also investigate the aggregate behavior of many concurrent TCP flows, and derive general conclusions about overall throughput, goodput, and loss probability.

The rest of the paper [1] is organized as follows: Section 2 presents our abstract network model for wide-area Internet connections. Section 3 studies the individual TCP performance under many competing TCP flows. Section 4 further simplifies our network model so that we can better explore the aggregate TCP behavior. Section 5 presents our simulation results and analysis for the aggregate TCP performance. Section 6 summarizes related work. We end with concluding remarks and future work in Section 7.

2. Network Abstraction

In this section, we develop a simple yet realistic model for wide-area Internet connections based on the Internet hierarchical routing structure.

2.1. Connections from a single domain

There are three levels of routing in today's Internet. The top level is at the Internet backbone, which interconnects multiple autonomous systems (AS's). The middle level is within a single AS, which is from the routers in an enterprise domain to the gateway. At the bottom level, we have routing within a single broadcast LAN, such as Ethernet or FDDI [6]. The upper portion of Figure 1 shows an abstract view of a cross-domain network connection.

The link between an enterprise domain and the transient backbone is typically dedicated to the enterprise, and usually has enough bandwidth to carry its own traffic. Internet backbones generally also have large enough bandwidth, though sometimes can get congested. In contrast, the access link usually has very limited bandwidth, and is shared among multiple domains. Therefore, in many cases, it is reasonable to assume that **the access link is the bottleneck (Assumption 1)** for wide-area Internet connections. Under

[1] This paper has been abbreviated to conform to the ICNP page count limit. The complete version is available as technical report [13].

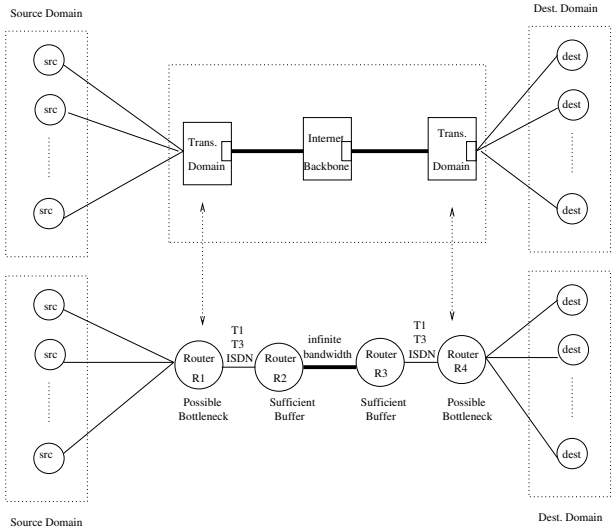

Figure 1. Abstract network topology for connections from a single domain

such an assumption, the topology can be further abstracted as shown in the lower portion of Figure 1, where the interconnection between transient backbones and internet backbones is abstracted as routers connected by access links and one high capacity link. The router at either access link can become the bottleneck depending on the traffic condition.

2.2. Connections from multiple domains

When access links are shared by multiple domains, the scenario looks much more complicated, as illustrated in Figure 2:

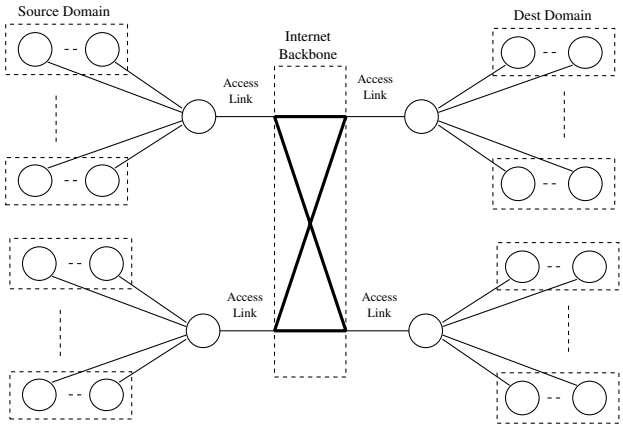

Figure 2. Abstract network topology for connections from multiple domains

At the first glance, it seems that all connections are intermixed with one another. However, according to the assumption that only access links can be bottlenecks, two connections not sharing any access links are thus independent. Based on such observation, we can partition all the connections into independent groups as follows:

1. Map the network connection into an undirected graph. Represent each access link as a node in the graph. Connect two nodes with an edge if and only if there is at least one connection going through both access links corresponding to the two nodes;
2. Find all connected components in the graph. The connections in different connected components are clearly independent of one another.

After decomposition, we only need to study a single connected component. The single connected component looks exactly the same as multiple cross-domain connections shown in Figure 2, except that now all the connections compete with one other, whereas before decomposition two connections can be independent of each other.

3. A Single TCP Flow Under Cross Traffic

The best-known analytical model for the steady state performance of TCP-Reno is proposed in [12]. According to their model, the steady state throughput $B(p)$ for an individual TCP flow can be approximated as follows:

$$min(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)})$$

where p is the loss probability, b is average number of packets acknowledged by an ACK, W_{max} is the maximum congestion window, RTT is the average roundtrip time experienced by the connection, and T_0 is the average duration of a timeout without back-off.

In [12] the model is validated empirically using real traces. We believe that further investigating the model through simulations has its unique value:

- Simulations can accurately determine the value of each parameter used in the model, which is not always possible in real traces. For example, the dropping probability p can only be approximated in real traces by loss indications. RTT estimation can be inaccurate due to coarse-grained timer.
- Simulations using a generic model can cover a wider range of scenarios.
- Simulation offers a well-controlled environment, whereas in real traces, a lot of unknown factors, such as topologies, processing time, different TCP versions etc., can make the results hard to interpret [11].

We use **ns** [10] to run a large number of simulations on different variations of the abstract network topology in Section 2. The topologies are labeled according to Table 1, and are illustrated in Figure 3 and Figure 4.

Table 2 summarizes the accuracy of the model, where the percentage is based on over 10000 data points. In other

Topology	$D(L2)$	$BW(L2)$	$Buff$	$Conn$
1	50 ms	64 kbps	100 - 500	1 - 100
2	50 ms	64 kbps	100 - 500	110 - 600
3	50 ms	1.6 Mbps	100 - 500	1 - 100
4	50 ms	1.6 Mbps	100 - 500	110 - 600
5	as shown in Figure 4		100 - 500	1 - 600

Table 1. Simulation topologies. $D(L)$ and $BW(L)$ denote the propagation delay and bandwidth of link L respectively; $Buff$ is the buffer size (in packets) at all routers; $Conn$ stands for the number of connections.

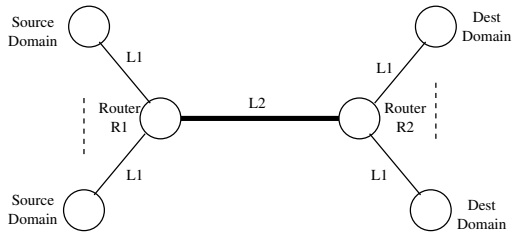

Figure 3. Settings for topology 1 - 4. Link L1 is a 10 Mbps Ethernet link with 0.001 ms delay. Table 1 lists other parameters.

simulation settings not shown in this paper, similar results have been observed.

As shown in Table 2, most of their estimations are accurate within a factor of 2, which is reasonably good. We have also evaluated their more accurate full model and observed similar accuracy (See [13] for details).

A number of factors contribute to the divergence between their model and the simulation results:

- The simplistic assumption that once a packet in a given round is lost, all remaining packets in the same round are lost as well [12].
- Ignoring the packets sent in the slow start phase.
- Several simplifications that can introduce distortion:

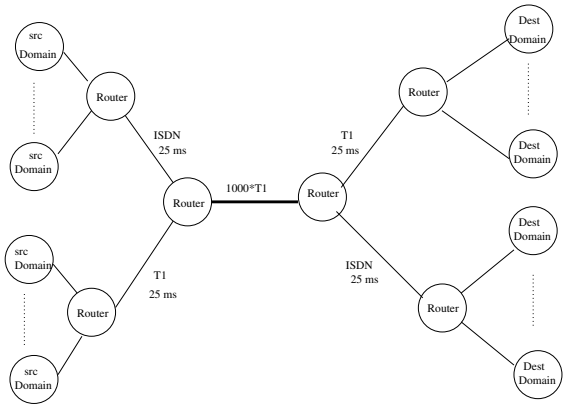

Figure 4. Topology 5. All unlabeled links have 28.8 kbps bandwidth and no propagation delay.

Topology	$P(1.5)$	$P(2)$	$P(3)$	$P(4)$
1	21.04	78.48	98.83	99.31
2	31.65	58.70	87.23	94.29
3	69.02	78.92	88.66	91.77
4	77.01	91.51	97.58	98.61
5	58.60	78.60	90.90	95.42

Table 2. The accuracy of the model proposed in [12]. $P(x)$ denotes the percentage of connections such that $max(\frac{Real\ Throughput}{Estimated\ Throughput}, \frac{Estimated\ Throughput}{Real\ Throughput}) \leq x$.

ignoring the effect of losing ACKs; ignoring timeout could occur before triple duplicate ACKs; ignoring packet reordering, that is, assuming loss is the only cause of duplicate ACK. (See [13] for more details).

4. Aggregate Behavior of Many TCP Flows - Model Simplification

From the perspective of network provisioning, it is very important to understand the aggregate behavior of many TCP flows, such as the overall throughput, goodput, loss rate, and fairness. Due to the same reasons we mentioned in Section 3, we again choose to study the aggregate TCP behavior by simulations.

To make simulation an effective approach, it is necessary to have a small parameter space so that we can identify exactly how the performance varies with different parameters. So we further simplify our network model using the assumption that **the bottleneck will eventually stabilize, and different connections sharing the same access link congest at the same place. (Assumption 2)**.

Under this assumption, our abstract network model can be further simplified as the widely used single-bottleneck model, as shown in Figure 5. The entire system can now be characterized by only four parameters: $propagation\ delay$, the buffer size of the bottleneck router S ($BufferSize_S$), the number of connections ($Conn$), and $Type_{bottleneck\ link}$.

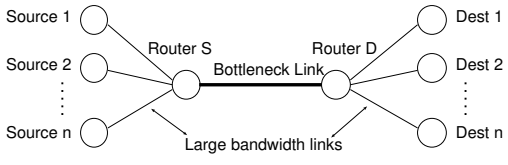

Figure 5. Simplified abstract network model

5. Aggregate Behavior of Many TCP Flows - Simulation and Analysis

In this section, we study the aggregate TCP performance through extensive simulations using **ns** [10]. Our simulation topology is based on our abstract model simplified in Section 4. In Figure 5, the large bandwidth link has 10 Mbps Ethernet bandwidth and 0.001 ms delay. We vary each of the four parameters in the model: $propagation\ delay$, $BufferSize_S$, $Conn$, and $Type_{bottleneck\ link}$ to see how each of them affects TCP performance. More specifically, we consider both ISDN and T1 access links, with delay of either 50 ms (typical for terrestrial WAN links) or 200 ms (typical for geostationary satellite links). We also vary the buffer size and the number of connections in each scenario.

The bottleneck link router uses FIFO scheduling and drop-tail buffer management, which are most commonly used in the Internet. The TCP segment size is set to 500 bytes. As [9] points out, it is very common to have hundreds of concurrent TCP flows competing for the bottleneck resource in today's Internet, so we are particularly interested in investigating the TCP behavior for such a large number of flows.

We use the following notations throughout our discussions:

- Let W_{opt} = *propagation delay*bottleneck bandwidth*, which is the number of packets the link can hold.
- Let $W_c = W_{opt} + B$, where B is the buffer size at the bottleneck link. W_c is the total number of packets that the link and the buffer together can hold.

For the interest of brevity, we omit the simulation results for ISDN whenever they are similar to T1. Interested readers can refer to [13] for more extensive results.

5.1. TCP behavior for flows with the same propagation delay

Our study of TCP flows with the same propagation delay shows TCP exhibits wide range of behaviors depending on the value of $\frac{W_c}{Conn}$, where $Conn$ denotes the number of connections. Based on the capacity of the pipe (measured as $\frac{W_c}{Conn}$), we classify our results into the following three cases: large pipe ($W_c > 3 * Conn$), small pipe ($W_c < Conn$), and medium pipe ($Conn < W_c < 3 * Conn$).

5.1.1. $W_c > 3 * Conn$ (Large pipe case)

Previous studies have shown a small number of TCP connections with the same RTT can get synchronized [14]. Our simulation results demonstrate that synchronization persists even for a large number of connections.

Figure 6 depicts the synchronization behavior. **In all the graphs we sort the connection ID's by the total number of packets each connection has received, which reveals synchronization behavior more clearly.** As shown in the figure, the buffer occupancy periodically fluctuates from half to full, which implies all connections halve their congestion windows in synchrony. The global synchronization behavior can be further illustrated by the periodic white stripes in the scatter plot of ACK arrival time, which imply all the connections start and end loss recovery in a synchronized manner.

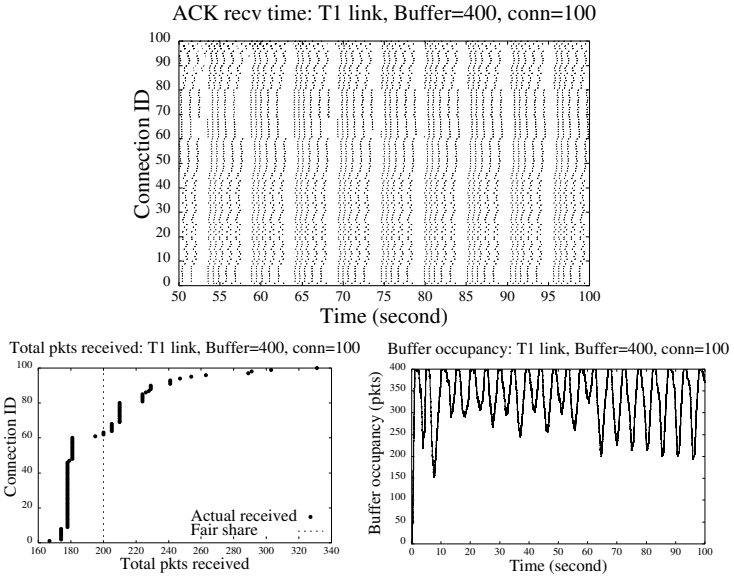

Figure 6. Global synchronization in large pipe case: 100 flows share a T1 link with 50 ms $delay$ and 400 packet $buffer$.

The explanation for the synchronization behavior is similar to the case for small number of connections. At the end of each epoch, the bottleneck buffer becomes full. Each flow will thus incur a loss in the same RTT when it increments its congestion window. Since $W_c > 3 * conn$, most flows have more than 3 outstanding packets before the loss. So they can recover from the loss by fast retransmission, and reduce the window by half, leading to global synchronization.

Due to global synchronization, all the flows share the resource fairly: in the steady state they experience the same number of losses and send the same number of packets. We can aggregate all the connections as one big connection, and accurately predict the aggregate loss probability as follows:

$$Loss\ Probability = \frac{1}{b * (\sum_{x=\lfloor \frac{W+1}{2} \rfloor}^{W} x) + 2 * W + 1}$$

where b is the average number of packets acknowledged by an ACK, and $W = \frac{W_c}{Conn}$. (See [13] for details.)

When $b = 1$, $Loss\ Probability$ can be approximated as

$$Loss\ Probability \approx \frac{8}{3 * W^2 + 21 * W + 8}.$$

Figure 7 shows our prediction matches very well to the actual loss probability.

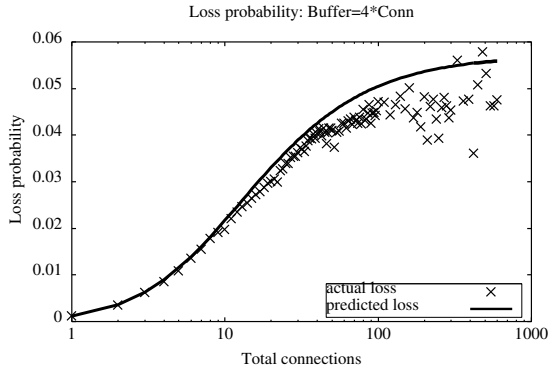

Figure 7. Loss prediction for large pipe case: Varying number of connections share T1 link with 50 ms $delay$ and $buffer = 4 * Conn$.

5.1.2. $W_c < Conn$ (Small pipe case)

When $W_c < Conn$, we find TCP flows share the path very unfairly: only a subset of flows are active (i.e. with goodput considerably greater than 0), while the other flows are shut-off due to continuous timeout as shown in Figure 8.

The number of active flows is close to W_c, and the exact value depends on both W_c and the number of competing flows. When $Conn$ exceeds the number of active flows the network resource can support, adding more flows only creates more shut-off flows. Almost all the packets sent by the shut-off flows get dropped.

5.1.3. $Conn < W_c < 3 * Conn$ (Medium pipe case)

As shown in Figure 9, TCP behavior in this case falls in between the above two cases. More specifically, in contrast to the large pipe case in Section 5.1.1, when different flows experience loss in the same RTT, since $1 < \frac{W_c}{Conn} < 3$, the flows respond to loss differently: those flows with $cwnd > 3$ before the loss can recover the loss through fast retransmission, while the others have to resort to timeout. Since the set of flows recovering loss using fast retransmission and the set using timeout can change over time, no global synchronization occurs, and the network resources are not shared as fairly as when $W_c > 3 * Conn$. On the other hand, there is still local synchronization, as shown in Figure 9, where some groups of flows are synchronized within the groups. Furthermore, since $\frac{W_c}{Conn} > 1$, all the flows can

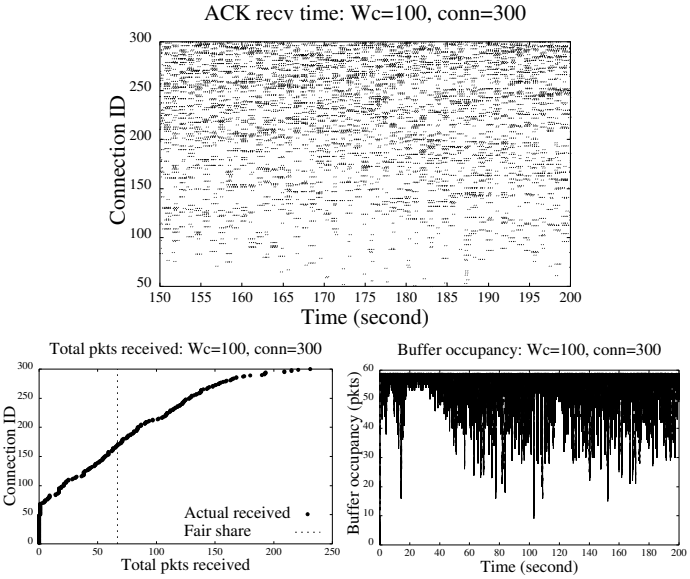

Figure 8. Small pipe case: 300 concurrent flows compete for T1 link with 50 ms $delay$ and 60 packet $buffer$ ($W_c = W_{opt} + Buffer = 100\ packets$). Note that the buffer occupancy fluctuates widely. Moreover part of flows receive little goodput as shown in the bottom left graph.

get reasonable amount of throughput. Therefore, in contrast to the small pipe case, almost no flow gets shut off.

5.1.4. Aggregate Throughput

We define *normalized aggregate TCP throughput* as the number of bits sent by the bottleneck link in unit time normalized by the link capacity. Our results are as follows:

- As shown in Figure 10(a), when the number of flows is small and the buffer size is less than W_{opt} (160 packets in this case), the normalized TCP throughput is less than 1. The degree of under-utilization depends on both the number of flows and the ratio of the buffer size to W_{opt}. The smaller the number of flows and the lower the ratio, the lower the network utilization is.
- As shown in Figure 10(b), when the buffer size exceeds W_{opt} (40 packets in this case), the normalized TCP throughput is close to 1, regardless of the number of flows.
- When the number of flows is large, even if the buffer size is small (smaller than W_{opt}), the normalized TCP throughput is close to 1. This is evident from Figure 10(a), where the throughput is close to 1 with large number of flows for all the buffer sizes considered.

5.1.5. Aggregate Goodput

We define *normalized aggregate goodput* G as the number of *good* bits received by all the receivers (excluding un-

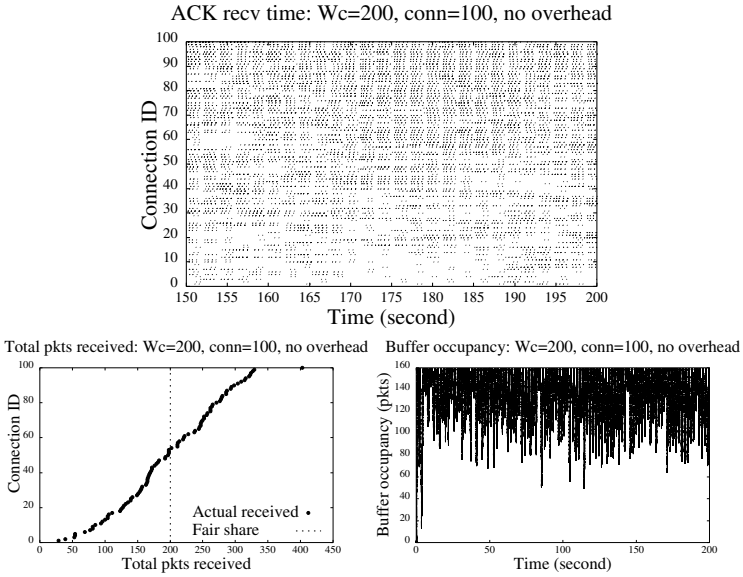

Figure 9. Medium pipe case: 100 concurrent flows compete for T1 link with 50 ms $delay$ and 160 packet $buffer$ ($W_c = 200\ packets$). Buffer occupancy fluctuates widely. There is local synchronization within some groups.

necessary retransmissions) in unit time normalized by the link capacity. As shown in Figure 11, there is a linear decrease in goodput as the number of connections increases. The slope of the decrease depends on the bottleneck link bandwidth: the decrease is more rapid when the bottleneck link is ISDN, and is slower when T1 is used as the bottleneck link.

The results can be explained as follows. The difference between the throughput and goodput is the number of unnecessary retransmissions. As the number of connections increases, the loss probability increases, which in turn increases the number of unnecessary retransmissions. Therefore the more connections, the lower the goodput is. On the other hand, since

$$loss\ in\ G = \frac{total\ unnecessary\ retransmissions}{link\ capacity}$$

the decrease in the goodput is more substantial with slower bottleneck link (e.g. ISDN), and less significant with faster bottleneck (e.g. T1).

5.1.6. Loss Probability

Our simulation results indicate when the W_c is fixed and the number of connections is small, the loss probability grows quadratically with the increasing number of connections as shown in Figure 12. The quadratic growth in the loss probability can be explained as follows. When $\frac{W_c}{Conn} > 3$, TCP connections can recover loss without timeouts. Every connection loses one packet during each loss

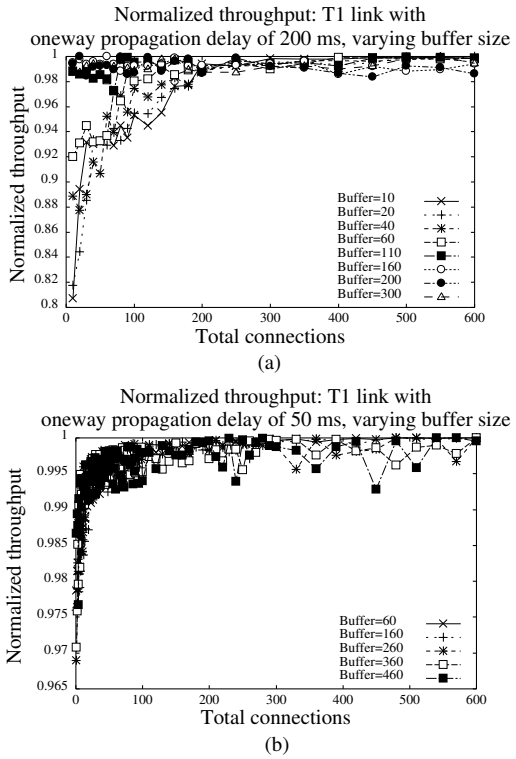

Figure 10. Throughput: varying number of connections compete for the bottleneck link T1 with 200 ms or 50 ms $delay$.

episode. So altogether there are $Conn$ losses every episode. Meanwhile the frequency of loss episode is proportional to $Conn$. Therefore for small number of connections, the loss probability is proportional to $Conn^2$. Such quadratic growth in loss probability is also reported in [9] for routers with RED dropping policy.

When the number of connections is large (larger than $\frac{W_c}{3}$), as shown in Figure 13, the growth of loss probability with respect to the number of connections matches impressively well with the following family of hyperbolic curves represented by $y = \frac{b*x}{x+a}$. Table 3 gives the corresponding parameters a and b for curves in Figure 13.

Wc	100	200	300	400	500
a	144.93	285.71	454.55	526.32	769.23
b	0.3237	0.3429	0.3682	0.3517	0.3948

Table 3. Parameters for the hyperbolic curves used for fitting loss probability in Figure 13

5.2. TCP behavior with random overhead

Our discussions in Section 5.1 focus on the macroscopic behavior of concurrent TCP connections with the *same*

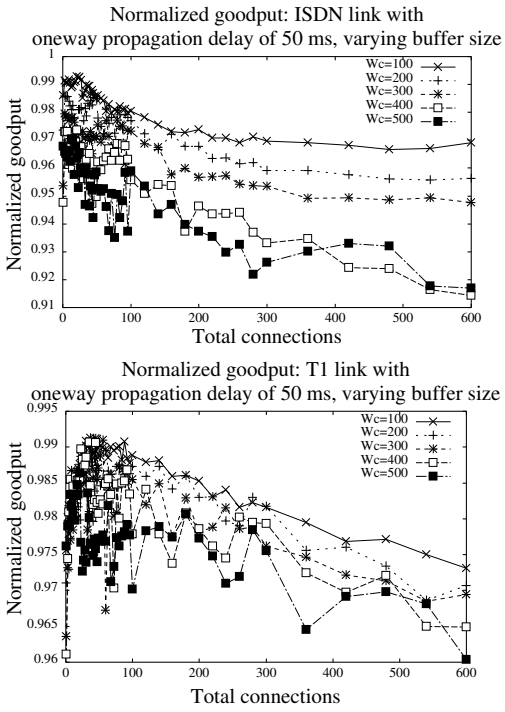

Figure 11. Goodput: varying number of flows compete for the bottleneck link of either ISDN or T1, both with 50 ms $delay$.

propagation delay. In order to explore properties of networks with Drop Tail gateways unmasked by the specific details of traffic phase effects or other deterministic behavior, we add random packet-processing time in the source nodes. This is likely to be more realistic. The technique of adding random processing time was first introduced in [3], though in different context.

5.2.1. $W_c > 3 * Conn$ (Large pipe case)

For the large pipe case, we find that a random processing time ranging from zero to $10\% * RTT$ is required to break down the global synchronization. This is shown in Figure 14, where the global synchronization is muted after adding the random processing time up to $10\% * RTT$.

The performance results in the non-synchronization case also differ from the global synchronization case. As shown in Figure 15, when the number of connections is less than 100, the loss probability in both cases are almost the same; as the number of connections increases further, the gap between the two opens up: the non-synchronization case has higher loss probability than the synchronization case. Nevertheless, using the prediction based on global synchronization gives a reasonable approximation (at least a lower bound) of loss probability for non-synchronized case. However in the non-synchronization case, the connections do not share the bandwidth fairly, and we can no longer predict the bandwidth share for each connection.

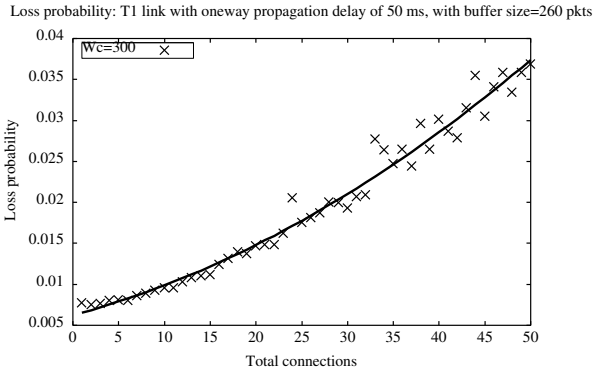

Figure 12. Loss probability for small number of flows: varying number of flows compete for the bottleneck link of T1 with 50 ms $delay$. The loss probability grows quadratically when the flow number is small.

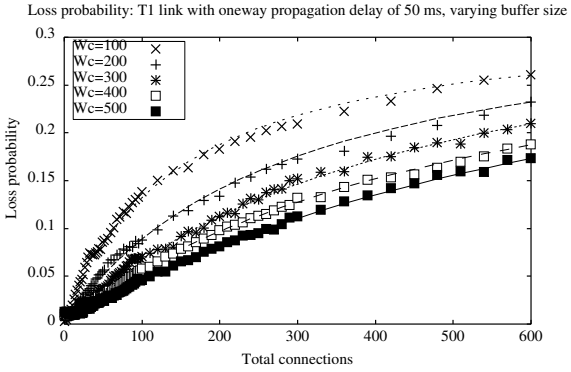

Figure 13. Loss probability for a large number of connections: varying number of connections compete for the bottleneck link of T1 with 50 ms $delay$. The loss probability curves match very well with hyperbolic curves when the number of connections is large, where the parameters of the hyperbolic curves are given in Table 3.

5.2.2. $W_c < Conn$ (Small pipe case)

For the small pipe case, adding random processing time makes systematic discrimination much less severe than before. As shown in Figure 16, the number of shut-off connections is considerably smaller than before. Furthermore, the buffer occupancy is mostly full and stable, whereas without random processing time, the buffer occupancy is quite low, and fluctuates a lot.

5.2.3. $Conn < W_c < 3 * Conn$ (Medium pipe case)

As shown in Figure 17, adding random processing time does not have much impact on TCP behavior in the case of medium size pipe: as before, most connections get reasonable goodput, though not synchronized. On the other hand, the buffer occupancy now becomes mostly full and stable

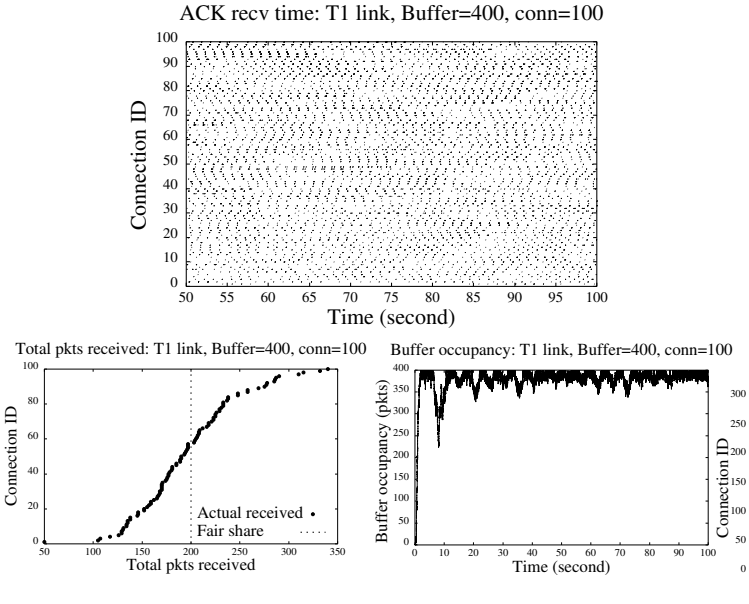

Figure 14. Adding random process time in large pipe case breaks down the global synchronization: 100 connections share the bottleneck link of T1 with 50 ms $delay$ and 400 packet $buffer$. Compared to the case of without random processing time, the buffer occupancy is quite stable. Moreover global synchronization disappears as shown in the scatter plot for ACK arrival.

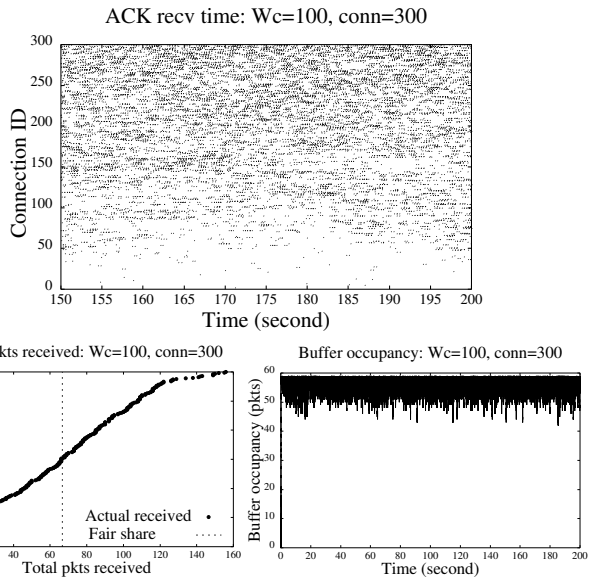

Figure 16. Adding random processing time in small pipe case: 300 concurrent connections compete for T1 link with 50 ms $delay$ and 60 packet $buffer$ ($W_c = 100$ packets). The buffer occupancy is quite stable, and the consistent discrimination is less severe.

in contrast to without random processing time, where the buffer occupancy is low, and fluctuates a lot. In addition, even local synchronization disappears after adding random processing time.

5.2.4. Aggregate Throughput & Goodput

Adding random processing time has little impact on the overall throughput and goodput. The conclusions drawn in Section 5.1.4 and 5.1.5 still applies. See [13] for more details.

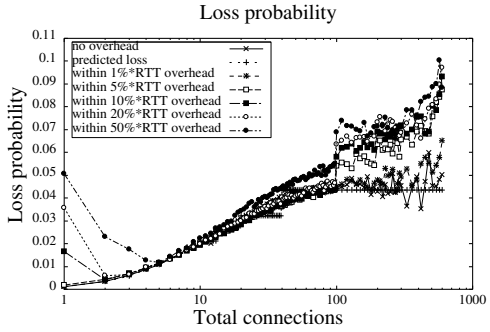

Figure 15. Compare the loss probability by adding different amount of random processing time: varying number of connections compete for T1 link with 50 ms $delay$

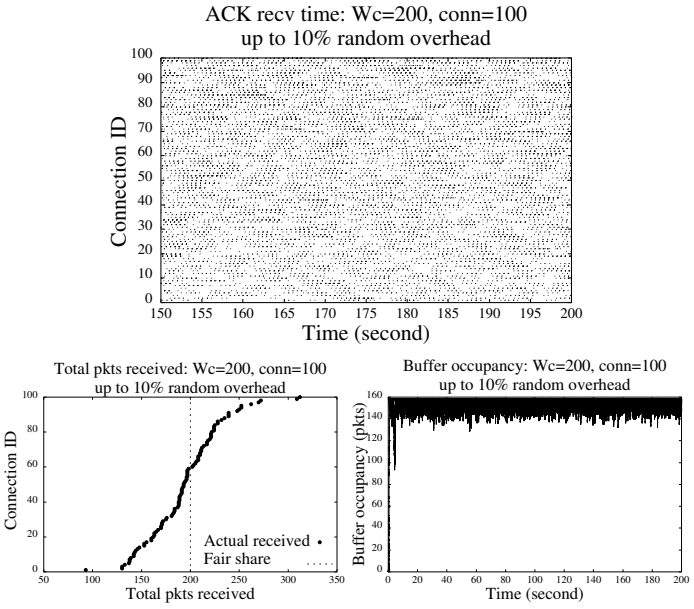

Figure 17. Adding random processing time in medium pipe case: 100 concurrent connections compete for T1 link with 50 ms $delay$ and 160 packet $buffer$ ($W_c = 200$ packets). The buffer occupancy is quite stable. In contrast to without random processing time, there is no local synchronization.

5.2.5. Loss Probability

For a small number of connections, adding random processing time makes the loss probability grow almost linearly as the number of connections increases. This is evident from Figure 18, which compares the loss probability curves before and after adding random processing time.

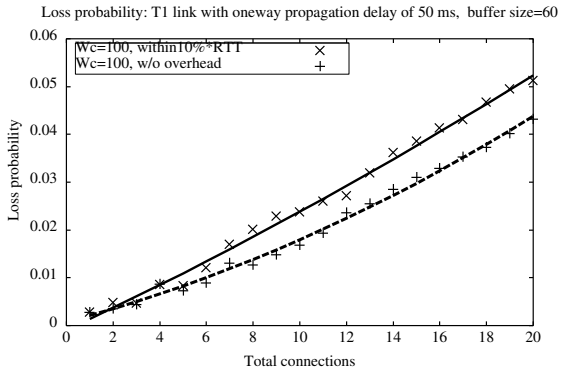

Figure 18. Loss probability for small number of connections after adding random processing time of up to $10\% * RTT$ at TCP source: varying number of connections compete for the bottleneck link of T1 link with 50 ms $delay$. The loss probability grows linearly when the number of connections is small.

For a large number of connections, adding random processing time does not change the general shape of the loss probability curves: as before these curves matches very well with hyperbolic curves, though with different parameters (See [13] for details).

5.3. TCP behavior with different RTT

It is well-known that TCP has bias against long roundtrip time connections. We are interested in quantifying this discrimination through simulations. Our simulation topology is similar to Figure 5 (in Section 4), except that we change the propagation delay of the links. More specifically, we divide all the connections into two equal-size groups, where one group of connections has fixed propagation delay on the large bandwidth links, and the other group of connections has varying propagation delay on the large bandwidth links. As suggested in [3], we add a random packet-processing time in the source nodes that ranges from zero to the bottleneck service time to remove systematic discrimination. Our goal is to study how the throughput ratio of two groups changes with respect to their RTT's.

Our simulation results are summarized in Figure 19, which plots the throughput ratio vs. their RTT ratio both in **log2 scale**. As shown in Figure 19, the throughput ratio is bounded by two curves. More specifically, when $RTT_1 \leq RTT_2$,

$$(\frac{RTT_2}{RTT_1})^2 \leq \frac{Throughput_1}{Throughput_2} \leq 2 * (\frac{RTT_2}{RTT_1})^2 \qquad A(1)$$

where RTT_1 and RTT_2 are the average RTT the connections in group 1 and group 2 experience respectively. Since we can swap the labels for groups 1 and 2, so the throughput ratio is symmetric as shown in Figure 19.

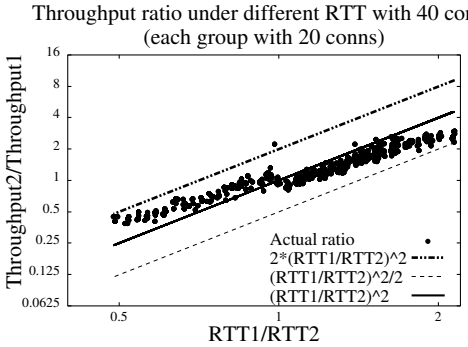

Figure 19. Two groups of TCP connections compete for T1 link

Now let's try to explain the relationship (A1). For ease of discussion, we aggregate all the connections in one group as a big connection. So in the following we just consider two connections compete with each other. Moreover, due to symmetry, we only need to consider the case when $RTT_1 \leq RTT_2$.

Figure 20 depicts roughly how the congestion windows evolve during congestion for two connections with different RTT. As shown in the figure, during every epoch the $cwnd$ of connection i grows from W_i to $W_i * 2$. So the average length of epoch, denoted as E_i, is roughly equal to $RTT * W_i$. Therefore

$$Throughput_i = \frac{3 * W_i^2}{2} * \frac{1}{Ei} = \frac{3 * W_i}{2 * RTT_i} \qquad (A2)$$

.

Now let x denote $\frac{E_1}{E_2}$. Using (A2), we have

$$\frac{Throughput_1}{Throughput_2} = (\frac{RTT_2}{RTT_1})^2 * x.$$

Applying the equality of $A(1)$, we obtain $1 \leq x \leq 2$. This means the average epoch length of connection 1 is usually no larger than twice the epoch length of connection 2. That is, for every two losses in connection 2, on average there is usually at least one loss in connection 1. This implies there is no consistent discrimination against any particular connection, which is likely to be the case after adding random processing time [3].

The roundtrip time bias in TCP/IP networks has received lots of attention. [7] gives an analytical explanation for this,

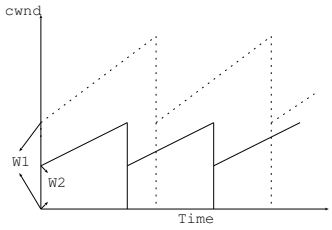

Figure 20. Window evolution graph for two connections with different RTT's

and concludes the ratio of the throughput of two flows using TCP-Tahoe (i.e. $\frac{Throughput_i}{Throughput_j}$) is proportional to $(\frac{RTT_j}{RTT_i})^2$. Their analysis is based on the assumption that both flows get synchronized, which is not always valid in reality. For TCP-Reno, they also show the throughput ratio of two flows is unpredictable: the connection with smaller propagation delay can sometimes get lower throughput due to systematical discrimination. *Our simulation study shows that though the throughput ratio of two connections may fluctuate a lot, the aggregate throughput ratio of two groups of connections is relatively stable and is clustered within a band close to* $(\frac{RTT_1}{RTT_2})^2$ *(the width of the band is usually one unit in log2 scale).*

6. Related Work

Large scale performance analysis has been an active research area recently. Many researches are currently focused on building scalable simulators, such as [1, 5, 15].

Analyzing simulation results to estimate TCP performance, as done in this project, is a very different approach from building a scalable simulator. The strategy taken by [9] is the closest to ours. It studies how TCP throughput, loss rates, and fairness are affected by changing the number of flows. Their work differs from ours in that they only study the impact of varying one parameter - the number of flows. Moreover, they study TCP-Tahoe assuming RED dropping policy at the routers, which is not widely used in today's Internet. RED dropping policy is not sensitive to instantaneous queue occupancy, so it is relatively easy to obtain the steady state performance. Several analytical models have been proposed for studying the steady state TCP throughput when routers use RED dropping policy [8, 16].

7. Conclusion and Future Work

In this paper, we have investigated the individual and aggregate TCP performance when there are many competing TCP flows. We first develop a simple yet realistic network model to abstract an Internet connection. Based on the model, we study the behavior of a single TCP flow by evaluating the best-known TCP analytical model. We also examine the aggregate behavior. Through extensive simulations, we have identified how TCP performance varies with changing parameters in the network model. These results give us valuable insights into how TCP behaves in the diverse Internet.

There are a number of directions for future work. For example, we plan to further explore TCP performance under different RTT's. In particular, we want to consider the following two extensions: (i) when the two different RTT groups are not equal size; and (ii) with different number of RTT groups. We also plan to use Internet experiments to verify some of the results in the paper.

References

[1] J. Ahn and P. B. Danzig. Speedup vs. Simulation Granularity. [unpublished]

[2] S. Floyd. Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic. Computer Communication Review, Vol.21, No.5, October 1991, p. 30-47.

[3] S. Floyd and V. Jacobson. On Traffic Phase Effects in Packet-Switched Gateways. Internetworking: Research and Experience, V.3 N.3, September 1992, p.115-156.

[4] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, V.1 N.4, August 1993, p. 397-413.

[5] P. Huang, D. Estrin, and J. Heidemann. Enabling Large-scale Simulations: Selective Abstraction Approach to the Study of Multicast Protocols. USC-CS Technical Report 98-667, January 1998.

[6] S. Keshav. An Engineering Approach to Computer Networking, Addison-Wesley, 1997.

[7] T. V. Lakshman and U. Madhow. Performance Analysis of Window-based Flow Control using TCP/IP: Effect of High Bandwidth-Delay Products and Random Loss. In *Proc. IFIP TC6/WG6.4 Fifth International Conference on High Performance Networking*, June 1994.

[8] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. Computer Communication Review, July 1997.

[9] R. Morris. TCP Behavior with Many Flows. In *Proc. IEEE International Conference on Network Protocols '97*, October 1997.

[10] UCB/LBNL/VINT Network Simulator - ns (version 2). http://www-mash.cs.berkeley.edu/ns, 1997.

[11] V. Paxson. Automated Packet Trace Analysis of TCP Implementations. In *ACM SIGCOMM'97*, 1997.

[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, Modeling TCP Throughput: A Simple Model and Its Empirical Validation. In *Proc. ACM SIGCOMM '98*, 1998.

[13] L. Qiu, Y. Zhang, and S. Keshav. On Individual and Aggregate TCP Performance. Cornell CS Technical Report, TR99-1744, May 1999.

[14] S. Shenker, L. Zhang, and D. D. Clark. Some Observations on the Dynamics of a Congestion Control Algorithm. ACM Computer Communication Review pp.30-39, 1990.

[15] VINT. http://netweb.usc.edu/vint.

[16] X. Yang. A Model for Window Based Flow Control in Packet-Switched Networks. In *IEEE INFOCOM '99*, 1999.

[17] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A Quantitative Comparison of Graph-Based Models for Internet Topology. IEEE/ACM Transactions on Networking, December 1997.