The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network

Thomas J. Hacker
hacker@umich.edu
Center for Parallel Computing

Brian D. Athey
bleu@umich.edu
Cell & Developmental Biology

University of Michigan
Ann Arbor, MI USA 48109

August 1, 2001

1 Introduction

There are considerable efforts within the Grid and high performance computing communities to improve end-to-end network performance for applications that require substantial amounts of network bandwidth. The Atlas project [19] for example, must be able to reliably transfer over 2 Petabytes of data per year over transatlantic networks from Europe to the United States.

Recent experience [1, 2] has demonstrated that actual aggregate TCP throughput realized by high performance applications is persistently much less than the end-to-end structural and load characteristics of the network would indicate is available. One source of poor TCP throughput that has been identified is a packet loss rate that is much greater than what would be reasonably expected [20]. Packet loss is interpreted by TCP as an indication of network congestion between a sender and receiver. Packet loss, however, may be due to random factors other than network congestion, such as intermittent hardware faults [4].

Current efforts to improve end-to-end performance are taking advantage of the empirically discovered mechanism of striping data transfers across a set of parallel TCP connections between a sender and receiver to substantially increase TCP throughput. With the increasing popularity of this method, a sound understanding of the underlying mechanisms that explain how parallel TCP connections improves aggregate throughput and the effects of using parallel TCP connections on the network is crucial for application developers who need to increase the aggregate bandwidth available to their applications, and for network engineers who need to ensure fair and efficient use of the network.

In this paper, we will address several open questions concerning the use of parallel TCP connections. The first question that will be addressed is how the use of parallel TCP connections increases aggregate throughput. The second open question is how to determine the number of TCP connections that are necessary to maximize throughput while avoiding network congestion. Finally, understanding the impact of parallel TCP connections on the network when they are used, and the problem of understanding under what circumstances they should not be used. The purpose of this paper is to address these questions, and to suggest some practical guidelines for the use of parallel sockets to maximize end-to-end performance for applications while simultaneously minimizing their effects on the network.

The remainder of this paper is organized into five sections. Section two will discuss current work. Section three will present a parallel socket TCP bandwidth estimation model and the results of a series of experiments that validates the model. Section four will discuss the behavior of packet loss on the Internet and its effect on TCP throughput. Section five will present some specific conclusions and guidelines for using parallel sockets based on the results presented in this paper, and discuss some possible avenues for future work.

2.0 Current Work

Applications generally take two approaches to improve end-to-end network throughput that effectively defeats the congestion avoidance behavior of TCP. First, an application can utilize UDP, which puts responsibility for both error recovery and congestion control completely in the hands of the application. The second approach used is to open parallel TCP network connections and to "stripe" the data (in a manner similar to RAID) across a parallel set of sockets. These two approaches, however, are considered to be aggressive and do now provide for the fair sharing of network bandwidth available to applications [5].

Recent work [2, 1, 6] has demonstrated that the parallel socket approach greatly increases aggregate network throughput available to an application, but some experiences [6] have demonstrated that the speedup is not consistent.

Many projects are working to address the problems of poor network performance and the unpredictability of end-to-end network bandwidth available to an application. To address unpredictability, the Network Weather Service project [21] is working to predict the network bandwidth available between two sites on the Internet based upon statistical forecasting. Efforts to address poor network performance include Diffserv [22], Quality of Service (QoS) Reservation [23], Bandwidth Brokering [24], and network and application tuning efforts [3, 6].

The work that has been done to date on the use of parallel TCP connections is essentially empirical in nature and is generally from the perspective of an application. Long [8,9] describes some early work done to increase the transfer rate of medical images over the Internet. Allman [10] described work done to increase the throughput over satellite links to increase TCP throughput. Sivakumar [2] developed a library (Psockets) to stripe data transmissions over multiple TCP network connections to deliver dramatically increased performance on a poorly tuned host compared to the performance of a single TCP stream. Measurements using the Psockets library for striping network I/O demonstrated that the use of 12 TCP connections increased TCP performance from 10 Mb/sec to approximately 75 MB/sec. Eggert [17] and Balakrishnan [18] have developed modifications to TCP that take advantage of the positive effects of parallel TCP sockets. Lee [1] provided an argument that explains why network performance is improved over multiple TCP streams compared with a single TCP stream. The Stanford Linear Accelerator (SLAC) network research group [16] has created an exhaustive measurement infrastructure to measure the effect of multiple TCP connections between key sites on the Internet for the Atlas project.

Several applications are using or are planning to use parallel TCP connections to increase aggregate TCP throughput for their application. The ubiquitous example of this is the Netscape Web browser, which uses an empirically determined value of four for the number of parallel TCP connections use by the client [25]. The GridFTP project [26] allows the user to select the number of parallel TCP connections to use for FTP data transfer. Storage Resource Broker (SRB) [27] has provisions to use multiple TCP sockets to improve SRB data transfer throughput. The Internet-2 Distributed Storage Initiative (I2-DSI) [28] is investigating the use of parallel TCP connections to improve the performance of distributed data caches.

All of the current work has focused on investigating the effects of parallel TCP connections from an empirical perspective. Researchers have found that the optimal number of parallel TCP connections range from 4 (Netscape) to 12 (Psockets) to a number from 4 to 20 depending on the window size (SLAC group).

Concerns about the effects of using multiple network sockets on the overall fairness and efficiency of the network have been raised in the networking community [5, 28, 17]. Mechanisms such as traffic shaping and rate limiting [29, 31] have been proposed and implemented to attempt to prevent aggressive users from taking more than their fair share of the network from other users.

Despite the demonstrated effectiveness of using parallel sockets to improve aggregate TCP throughput, little work has been done to develop a theoretical model to validate the use of these optimal values, to understand the underlying mechanisms that allow parallel TCP connections to deliver tremendously increased performance, or to understand the effects of using parallel sockets on the fairness and efficiency of the network and in what conditions and circumstances they should be used.

The next section of this paper will develop a theoretical model of parallel TCP connections that will explain how they take advantage of random packet loss to improve aggregate throughput, and present experimental results that validates the theoretical model.

3.0 TCP Bandwidth Estimation Models

There are several studies that have derived theoretical expressions to calculate single stream TCP bandwidth as a function of packet loss, round trip time, maximum segment size, along with a handful of other miscellaneous parameters. Bolliger [11] performed a detailed analysis of three common techniques and assessed their ability to accurately estimate TCP bandwidth across a wide range of packet losses. The most accurate model is described in [12] as an approximation of the form[1]

$$TCP_{BW}(p) \approx \min\left(\frac{W_{\max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p\left(1+32p^2\right)}\right) MSS \qquad (1)$$

In this equation, $TCP_{BW}(p)$ represents bytes transmitted per second, MSS is the maximum segment size, W_{max} is the maximum congestion window size, RTT is the round trip time, b is the number of packets of transmitted data that is acknowledged by one acknowledgement (ACK) from the receiver (usually b=2), T_0 is the timeout value, and p is the packet loss ratio, which is the number of retransmitted packets divided by the total number of packets transmitted.

Bolliger found that the Mathis equation [13] is essentially as accurate for packet loss rates less than 1/100 as equation (1), but has a much simpler form:

$$BW \leq \frac{MSS}{RTT} \frac{C}{\sqrt{p}} \qquad (2)$$

In equation (2), p, MSS, and RTT are the same variables used in equation (1), C is a constant factor, and BW represents the number of bytes transmitted per second.

To understand the underlying mechanisms of TCP throughput, it is useful to consider the dynamic behavior of MSS, RTT and p and the degree of effect that each of these three parameters have on overall TCP bandwidth.

Of the three factors, MSS is the most static. If both sides of the TCP session have MTU discovery enabled [30] within the host operating system, both sides will attempt to negotiate the largest possible maximum transmission unit (and thus MSS) possible for the session. The MSS setting selected depends heavily on the structural characteristics of the network, host adapters, and operating system. Most often, the "standard" maximum MTU supported by networks and network adapters is 1500 bytes. In some cases, however, the data link layers of routers and switches that make up the end-to-end network will support larger frame sizes. If the MTU of a TCP connection

[1] Equation (1) is rescaled from the original form in [12] to match the scale of equation (2) by adding the MSS term.

can be increased from 1500 bytes to the "jumbo frame" size of 9000 bytes, the right hand side of equation (2) increases by a factor of 6, thus increasing actual maximum TCP bandwidth by a factor of 6 as well.

The value of RTT during a session is more dynamic than MSS, but less dynamic than p. The lower bound on the value of RTT is the transmission speed of a signal from host to host across the network, which is essentially limited by the speed of light. As the path length of the end-to-end network increases, the introduction of routers and framing protocols on the physical links between the two hosts adds latency to the RTT factor, and other factors involved with queuing and congestion can increase RTT as well. From an end host perspective, however, there is little that can be done to substantially improve RTT.

The final factor, packet loss rate p, is the most dynamic parameter of the triplet MSS, RTT, and p. The TCP congestion avoidance algorithm [32] interprets packet loss as an indication that the network is congested and that the sender should decrease its rate of transmission. In the operational Internet, the packet loss rate p spans many orders of magnitude and represents a significant contribution to variability in end-to-end TCP performance. It is important to note at this point that the packet loss rate has been observed to fall into two regimes: packet loss due to network congestion, and traffic insensitive packet loss. These two packet loss regimes will be explored in more detail in section 3.2.

The next section of this paper will present the derivation of an expression for aggregate TCP bandwidth, describe some of the characteristics of packet loss on the Internet, and describe how these characteristics affects the performance of single and multi stream TCP sessions.

3.1 Multi-stream TCP Bandwidth

If an application uses *n* multiple TCP streams between two hosts, the aggregate bandwidth of all *n* TCP connections can be derived from equation (2), in which MSS_i, RTT_i, and p_i represent the relevant parameters for each TCP connection *i*:

$$BW_{agg} \le C\left[\frac{MSS_1}{RTT_1\sqrt{p_1}} + \frac{MSS_2}{RTT_2\sqrt{p_2}} + \cdots + \frac{MSS_n}{RTT_n\sqrt{p_n}}\right] \quad (3)$$

Since the value for *MSS* is determined on a system wide level by a combination of network architecture and MTU discovery, it is reasonable to assume that each MSS_i value is identical and constant across all simultaneous TCP connections between the two hosts.

We may reasonably assume that *RTT* will be equivalent across all TCP connections, since the entire network path traversed by packets for each TCP connection will likely take the same network path and converge to equilibrium. At this point, note that since the TCP congestion avoidance algorithm is an equilibrium process that seeks to balance all TCP streams to fairly share network bottleneck bandwidth [15], that each stream must either respond to changes in the packet loss rate, RTT, or a combination of both to converge to equilibrium. Since all of the streams on a set of parallel TCP connections are between two end hosts, all of the streams should converge to equivalent RTT values, as long as the network between the two hosts remains uncongested. For purposes of this discussion, *C* can be set aside.

With these assumptions, equation (3) can be modified to the form:

$$BW_{agg} \le \frac{1}{RTT}\left[\frac{MSS}{\sqrt{p_1}} + \frac{MSS}{\sqrt{p_2}} + \cdots + \frac{MSS}{\sqrt{p_n}}\right] \quad (4)$$

Upon examination of equation (4), some features of parallel TCP connections become apparent. First, an application opening *n* multiple TCP connections is in essence creating a large "virtual MSS" on the aggregate connection that is *n* times the MSS of a single connection. If we then factor *MSS* out of equation (4):

$$BW_{agg} \le \frac{MSS}{RTT}\left[\frac{1}{\sqrt{p_1}}+\frac{1}{\sqrt{p_2}}+\cdots+\frac{1}{\sqrt{p_n}}\right] \quad (5)$$

It becomes apparent that given the relatively static nature of the values of *MSS* and *RTT* compared with the dynamic nature of *p*, the packet loss rate *p* is a primary factor in determining aggregate TCP throughput of a parallel TCP connection session.

3.2 The Behavior of Packet Loss and its Effect on TCP Bandwidth

It is apparent from equation (4) that the increased virtual MSS of parallel TCP connections is directly affected by the packet loss rate *p* and *RTT* of each connection. *RTT* has hard lower bounds that are structural and difficult to address. Packet loss *p*, on the other hand, is the parameter that is most sensitive to load on the network and is affected by a number of factors.

It has been observed that packet loss falls into two characteristic regimes: random losses not due to congestion, and congestion related losses. Paxson [14] found that packet losses tend to occur at random intervals in bursts of multiple packets, rather than single packet drops. Borella[33] found bursty packet loss behavior as well. Additionally, the probability of a packet loss event increases when packets are queued in intermediate hops as the network becomes more loaded. Bolot[20] found that packet loss demonstrates random characteristics when the stream uses a fraction of the available network bandwidth.

As the number of multiple TCP connections increases, the behavior of each packet loss factor p_i is unaffected as long as few packets are queued in routers or switches at each hop in the network path between the sender and receiver. In this context, in the absence of congestion, it is appropriate to assume that the proportion of packet loss will be fairly distributed across all connections. However, when the throughput of the aggregate packet stream begins to create congestion in the network, any router or switch in the network path may begin to drop packets. The packet loss attributable to each TCP stream will depend on the particular queuing discipline used in the individual network switches and routers, and on any phase effects that may be present from the interaction of TCP senders sharing a network bottleneck [39].

As a side note, there are four exceptions to the assumption that packet loss is fairly distributed when congestion occurs. It has been empirically determined [34, 7] that three pathological conditions exist. One condition, lockout, occurs when one stream dominates the queue in a router. The second condition, drop-tailed queues, arises due to queuing algorithms that unfairly targets a number of flows through the queue with excessive packet loss rates for newly arriving packets. The third pathological condition manifests itself in the form of heavy-tailed transmission time distributions for transmitting data over TCP connections due to congestion and high packet loss rates [40]. Finally, Floyd [39] found that multiple TCP streams that converge at the same congested bottleneck over network links with differing round trip times can demonstrate phase effects in which one stream unfairly dominates the queue and thus the outbound link.

It has been determined [31] that the unfair distribution of packet loss is an undesirable condition in congested routers. To provide mechanisms in routers to fairly distribute packet loss, new queuing schemes, such as Random Early Detection (RED) [31] are being designed and deployed to address the issue of fairly distributing packet loss across multiple streams. For the purposes of this analysis, we'll assume that packet loss impacts parallel TCP streams to the same extent.

For an illustration of the impact of multiple TCP streams in an uncongested network, consider the following example.

If we assume that $MSS = 4418 bytes$, $RTT = 70ms$, and $p_i = \frac{1}{10000}$ for all connections, and using

$$K = \frac{MSS(4418 bytes)}{RTT(70m\sec)}\left(\frac{8 bits/byte}{1000000 bits/Mbit}\right)\left(\frac{1000m\sec}{\sec}\right) \cong 0.5$$

The upper bound on aggregate TCP bandwidth can then be calculated using equation (5). Table 1 contains the results of this calculation for a varying number of sockets.

Number of Connections	$\sum_n \frac{1}{\sqrt{p_i}}$	Maximum Aggregate Bandwidth Using Equation (5)
1	100	50 Mb/sec
2	100+100	100 Mb/sec
3	100+100+100	150 Mb/sec
4	4 (100)	200 Mb/sec
5	5 (100)	250 Mb/sec

Table 1. Effect of I.I.D. Packet Loss on Aggregate TCP Bandwidth

Now, as the aggregate utilization of the network increases to the point where queues and buffers in switches and routers on the network path begin to overflow and packets are dropped, the network becomes congested. If the packet loss due to congestion is fairly shared over all of the connections through a switch or router, the negative effects of packet loss on the aggregate TCP bandwidth for a set of *n* simultaneous connections is magnified by a factor of *n*. For example, if the packet loss rate from the previous example doubles, the multiplicative packet loss rate factor in Table 1 is reduced from 100 to 70.71. For five simultaneous streams, the effect of this is to reduce aggregate bandwidth from 250 Mb/sec to 176.78 Mb/sec – a reduction of 30%. Even with this reduction, however, the aggregate bandwidth of 176.78 Mb/sec using five parallel TCP connections is still substantially better than the throughput obtained using only one connection at the desirable packet loss rate.

It is difficult to predict at what point the packet loss will become congestion dependent as the number of parallel TCP connections increase. There is, however, a definite knee in the curve of the graph of packet loss that indicates that adding additional network sockets beyond a certain threshold will not improve aggregate TCP performance. An examination of figures 1, 2, and 3 presented in the next section indicates that for a MTU of 1500 bytes, 10 sockets is the effective maximum number of sockets; for a MTU of 3000 bytes, 5 sockets is the effective maximum; and for a MTU of 4418 bytes, 3 or 4 sockets is the effective maximum. The effective maximum presented in figure 3 (MTU 1500) roughly corresponds to the results of Sivakumar [2], who found that the point of maximum throughput was at 16 sockets or less. Sivakumar did not mention the MTU used in [2], but if the default system settings or MTU discovery were used on the system, the MTU used was probably less than or equal to 1500 bytes.

3.3 Validation of Multistream Model with Data

To validate the theoretical derivation of the expression for parallel TCP connection throughput (equations 3 through 5), a series of experiments were conducted across the Abilene network from University of Michigan to NASA AMES Research Center in California. Each experiment consisted of a set of data transfers for a period of four minutes from U-M to NASA AMES, with the number of parallel TCP connections varied from 1 to 20 for each transfer. Seven of the experiments were run with the maximum transmission unit on the Abilene network (4418 bytes). Two experiments were run with a MTU of 3000 bytes, and two were run with a MTU of 1500 bytes. The host computer at U-M was a dual processor 800 Mhz Intel Pentium III server with a Netgear GA620 gigabit Ethernet adapter, 512 MB of memory running Redhat Linux 6.2 and the Web100 measurement software (without the use of auto tuning) [35]. The host computer at NASA AMES was a Dell PowerEdge 6350 containing a 550Mhz Xeon Intel Pentium III processor with 512MB of memory and a SysKonnect SK-9843 SX gigabit Ethernet adapter card running RedHat Linux. The network settings on the host at U-M were tuned for optimal performance, and the default TCP send and receive socket buffer was set to 16 MB. The host at NASA AMES was also well tuned and configured with a TCP socket buffer size of 4 MB. Each host had SACK [41] and Window Scale (RFC 1323) [42] enabled and the Nagle algorithm disabled [3]. Each data transfer was performed with the Iperf utility [36] with a TCP window size of 2 MB, data block size of 256 KB and the Nagle algorithm disabled. A traceroute was performed at the start and end of each run to assess the stability of the network path between sender and receiver.

The Web100 software (without autotuning) was utilized on the sender to collect the values of all the variables that Web100 measures at 10-second intervals during the 240 second run. The following Web100 parameters were extracted from each experiment: round trip time (SmoothedRTT), total count of the packets transmitted (PktsOut), total count of packets retransmitted (PktsRetrans), total number of bytes transmitted (DataBytesOut), total number of bytes retransmitted (BytesRetrans), and the total number of congestion recovery events, which are controlled by SACK (Recoveries).

The following Iperf measurements were extracted from the data collected from each experiment: bandwidth measured by Iperf for each TCP connection, and the number of TCP sockets used for each experiment. Missing observations in the figures are due to lost or incomplete measurements.

The statistical box plots in the following ten figures are notched box and whisker plots [43]. This method of statistical display is desirable compared with other commonly used statistical displays because it actually gives a complete graphical representation of the entire data set rather than a summary observation that may not reveal the complete character of the observations.

The parameters necessary to validate the theoretical model were extracted from the datasets gathered during the experiments: RTT = SmoothedRTT, p = (Recoveries)/(PktsOut), and MSS was statically configured for each test. Figure 1 shows the relationship between the number of parallel TCP connections and aggregate bandwidth for MSS of 4366 bytes. Figure 2 shows the relationship for an MSS of 2948 bytes, and Figure 3 shows the relationship for an MSS of 1448 bytes.

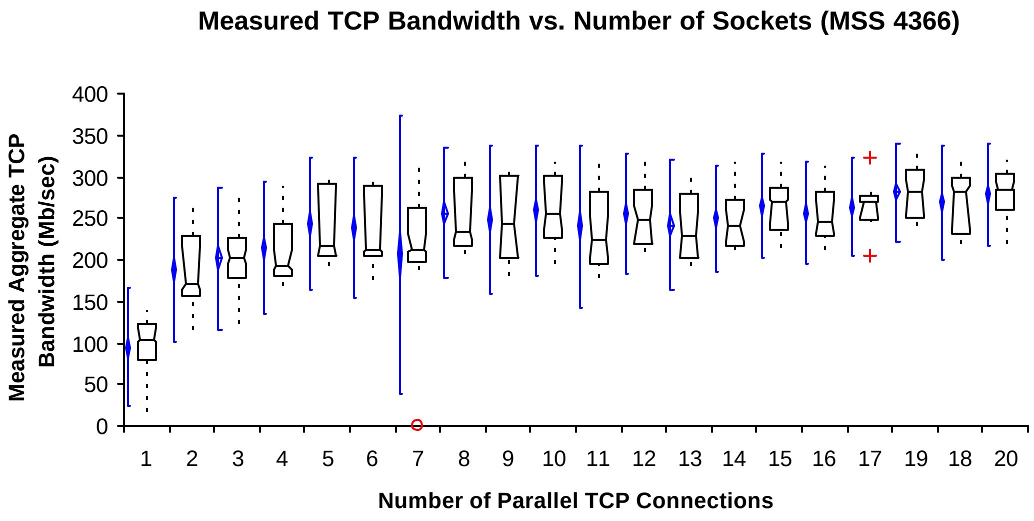

Figure 1. Effects of Parallel TCP Sockets with MSS of 4366 Bytes.

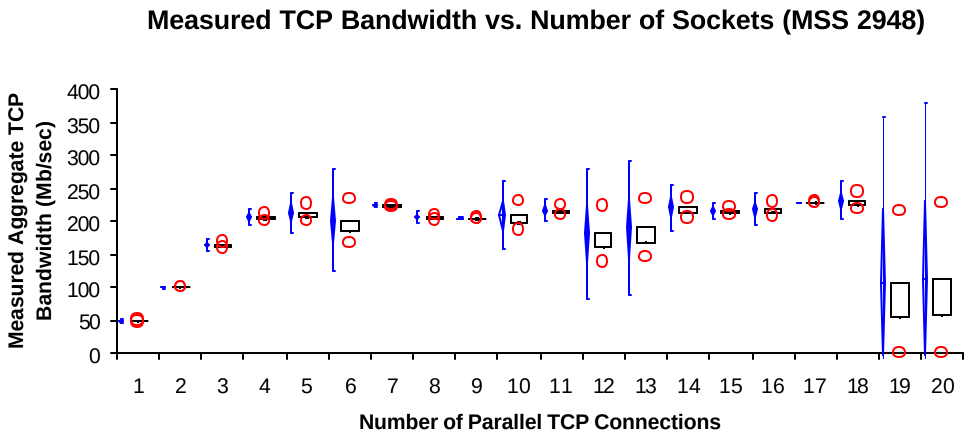

Figure 2. Effects of Parallel TCP Sockets with MSS of 2948 Bytes

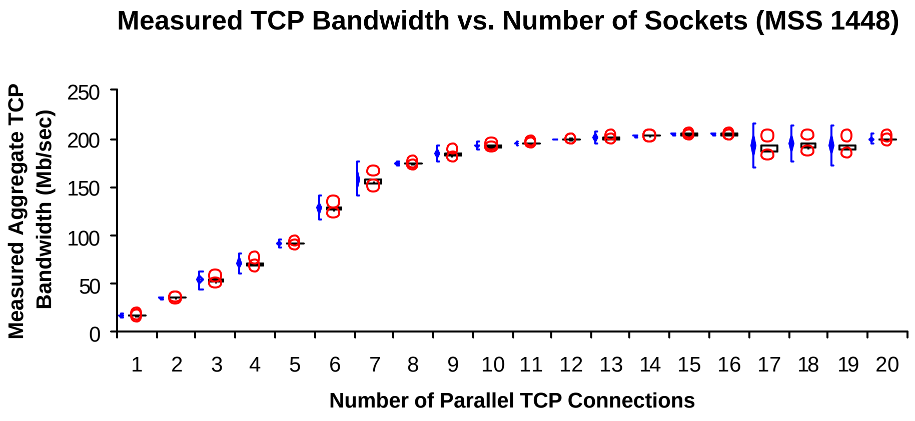

Figure 3. Effects of Parallel TCP Sockets with MSS of 1448 Bytes.

Since MSS is constant, and RTT is relatively static, the packet loss rate *p* is essential for determining the maximum aggregate TCP bandwidth. Figures 4, 5 and 6 show *p* (calculated from the ratio of SACK recoveries to the total number of outbound packets).

In examining these figures, it becomes apparent that there are two characteristic regimes of packet loss. In the first regime, as the number of sockets increases, the packet loss increases only slightly, and (with the exception of figure 6) the variation in packet loss rate is low. At some point, however, there is a knee in each curve where congestion effects begin to significantly affect the packet loss rate. After this point, the packet loss rate increases dramatically, and the variability in packet loss rate becomes much larger. TCP interprets packet loss as an explicit congestion notification from the network that indicates that the sender should decrease its rate of transmission. In the random regime of packet loss however, the TCP sender improperly throttles its data transmission rate. The knee in each one of these curves corresponds to the knee in the estimated and actual aggregate TCP throughput curves in figures 1-3 and 7-9.

When the knee in the packet loss rate and aggregate TCP throughput curves is reached, the benefits of adding additional TCP connections are lost due to two factors. First, the packet loss rate will increase for every additional socket added if the packet loss rate is in the congestion regime. This additional packet loss will offset any aggregate TCP bandwidth gains that might have been realized from additional TCP connections. Second, and most importantly, the bottleneck in the network between the sender and receiver simply has no additional network bandwidth to offer to the application. At this point, the bottleneck in the network is too congested to allow any additional streams.

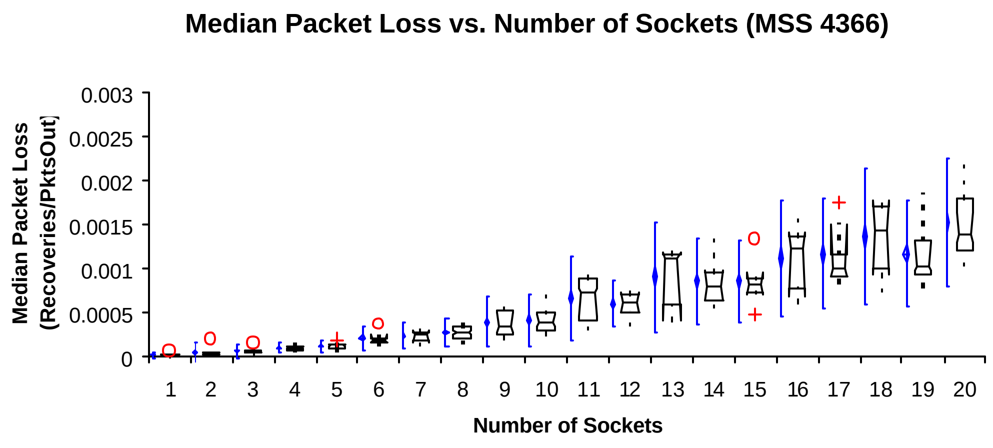

Figure 4. Packet loss rate for MSS 4366

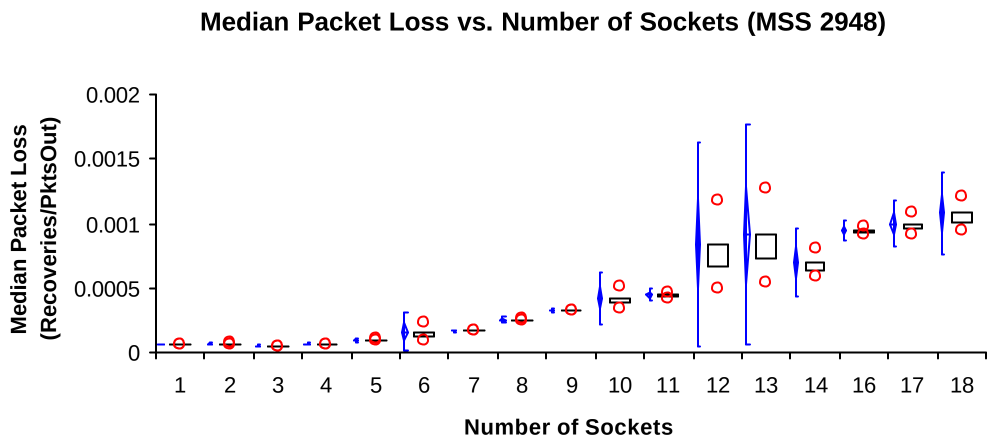

Figure 5. Packet Loss Rate for MSS 2948

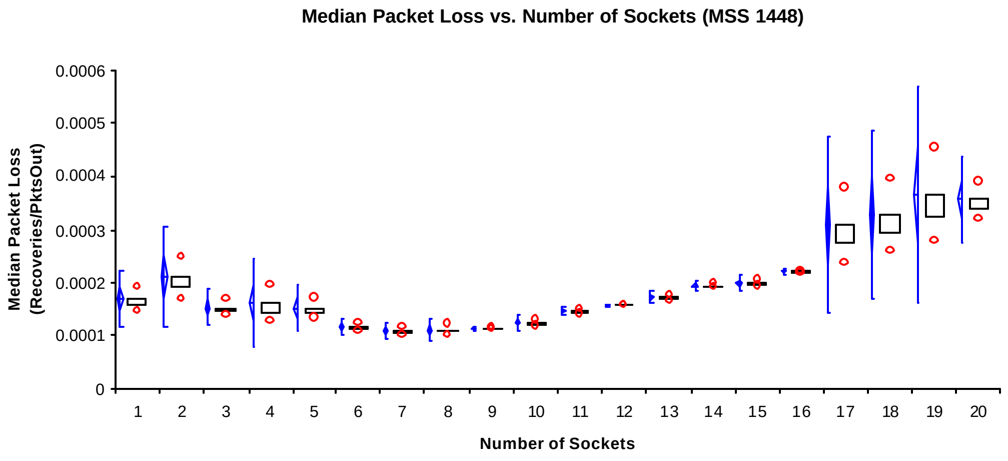

Figure 6. Packet Loss Rate for MSS 1448

Figures 7, 8, and 9 show the estimated aggregate TCP bandwidth as a function of the parameters gathered from the experiments. These parameters were used in equation (5) to generate these graphs.

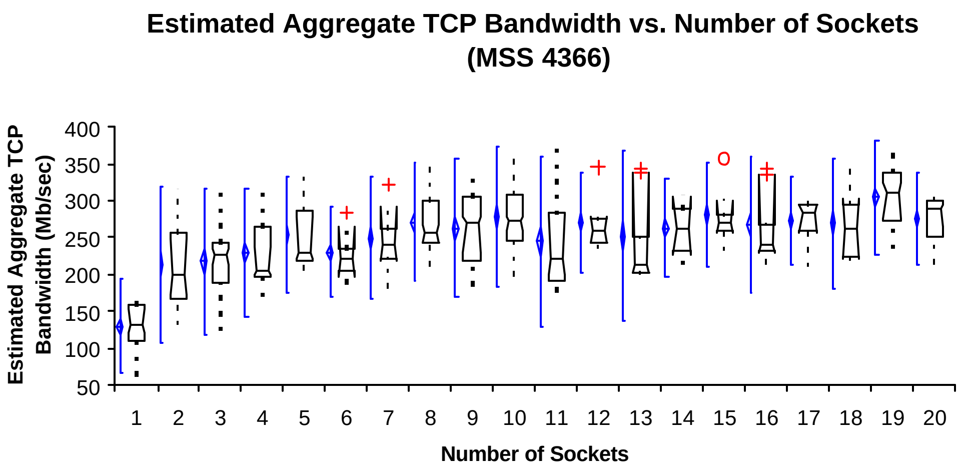

Figure 7. Estimated Aggregate TCP Bandwidth for MSS 4366

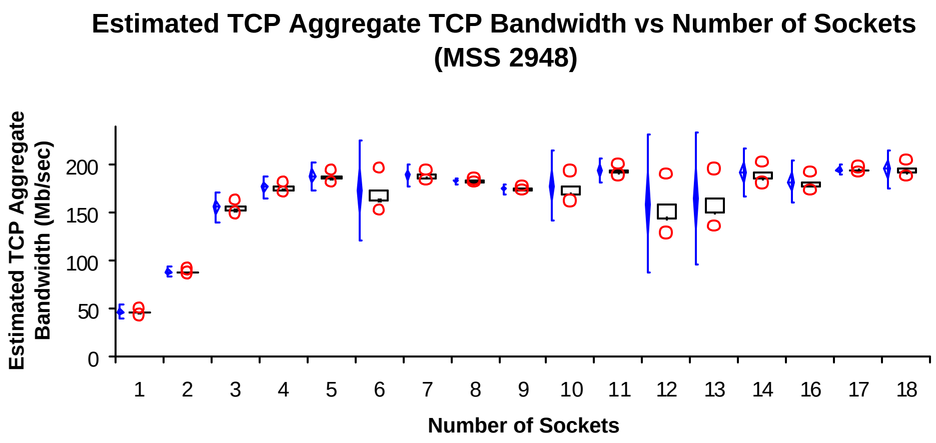

Figure 8. Estimated Aggregate TCP Bandwidth for MSS 2948

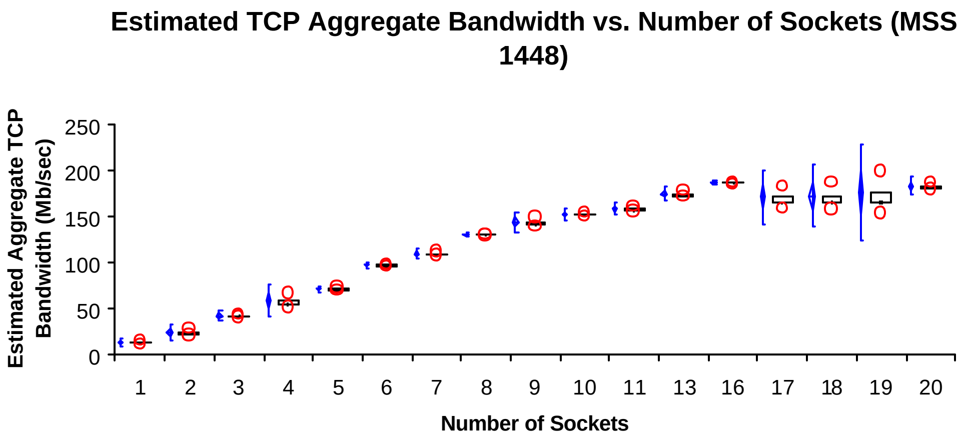

Figure 9. Estimated Aggregate TCP Bandwidth for MSS 1448

The round trip time (RTT) gathered from Web100 measurements demonstrated the expected static properties and remained in the range of 60 to 70 msec.

To determine the statistical difference between the estimated and actual TCP bandwidth as measured by Iperf, the method described by Jain [37] that is used to determine if two paired observations are statistically different with a confidence interval of 90% was employed. Figure 10 shows the differences between the measured and estimated values for each experiment for MSS 4366. The set of estimated values used for this calculation was based on the number of bytes transmitted. The 90% confidence interval for the differences between estimated and actual includes zero if the measurements are statistically similar. It is apparent from figure 10 that the Mathis equation slightly overestimates aggregate TCP bandwidth. This is in agreement with equation (5), which puts an upper bound on aggregate TCP throughput. To more accurately predict aggregate TCP throughput, a precise selection of the multiplicative constant *C* as described in Mathis [13] should be performed.

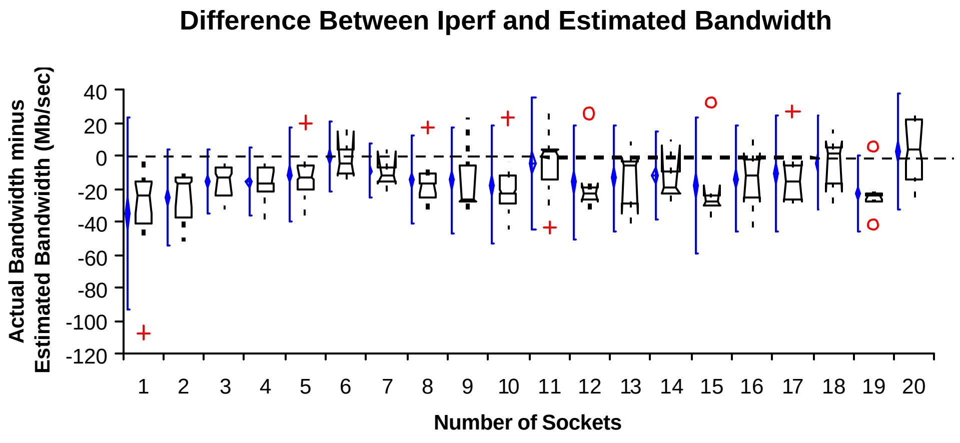

Figure 10. Difference between Actual and Estimated for MSS of 4366 Bytes

The measurements described in this section demonstrated that the theoretical model described in the previous section accurately determines an upper bound on actual TCP throughput as a function of *MSS, RTT* and packet loss rate *p*.

4 Why Parallel Sockets Work: The Effects of Multiple Network Connections in Uncongested Networks.

It seems counterintuitive that using parallel TCP sockets would improve aggregate throughput, since one would hope that the network would make a best effort to maximize throughput on a single stream. There are however, sources of traffic insensitive packet loss in the network that are not due to congestion. In this random packet loss regime, the use of parallel TCP connections allows an application to alleviate the negative effects of the misinterpretation of packet loss by the TCP congestion control algorithm. This section will give an explanation of why using parallel TCP connections increases aggregate throughput.

The derivation of equation 2 in Mathis [13] uses a geometric argument with constant probability packet loss rate $\frac{1}{p} = \left(\frac{W}{2}\right)^2 + \frac{1}{2}\left(\frac{W}{2}\right)^2$, where W is the congestion window size in packets. When a loss event occurs every $\frac{1}{p}$ packets, the slow-start algorithm will decrease the congestion window by half. This leads to the classic "saw tooth" pattern shown in figure 11.

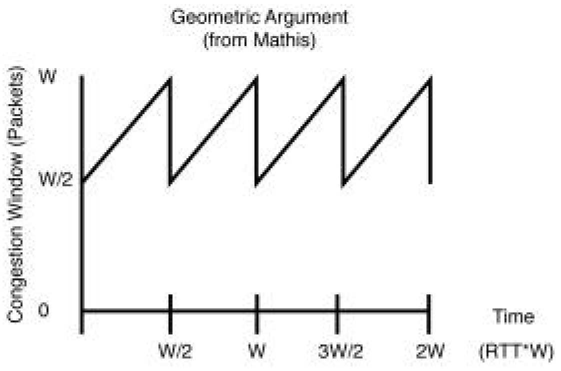

Figure 11. Classic TCP Saw tooth Pattern.

If we slightly modify the assumption made that p is a constant probability with the assumption that the constant probability p for an individual TCP stream is independent of the loss rate of other TCP streams from the same sender on an uncongested network, and that for each stream i, p_i is from a distribution identical to the other distributions for loss rate, the situation described in figure 11 can be used to describe the effects of parallel TCP connections as shown in figure 12.

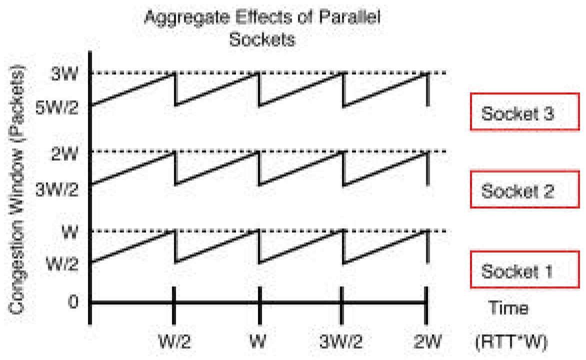

Figure 12. Geometric Construction of the Effects of Multiple Sockets

Given that the packet loss rates for all of these parallel TCP connections are not sensitive to traffic, and that packet losses occur in each channel at the same rate (as long as packet losses are not due to network congestion), an interesting effect occurs. If we combine the three streams in figure 12 into the aggregate representation shown in figure 13, it is clear that the effect of using multiple network sockets is in essence equivalent to increasing the rate of recovery from a loss event from one MSS per successful transmission to three times MSS. Note that this increased rate of recovery is theoretically and functionally equivalent to using a larger MSS on a single channel with the same packet loss rate *p*.

As the number of simultaneous TCP connections increases, the overall rate of recovery will increase until the aggregate network load begins to congest the network. At this point, the packet loss rate no longer remains traffic independent and begins to become dependent on the number of sockets and the amount of congestion in the network. This change in packet loss rate truly indicates that the network is congested, and that the TCP sender should reduce its congestion window.

As the number of parallel TCP connections increases, and the effects of higher packet loss rates decreases the impact of multiple sockets, the aggregate TCP bandwidth will stop increasing, or begin to decrease.

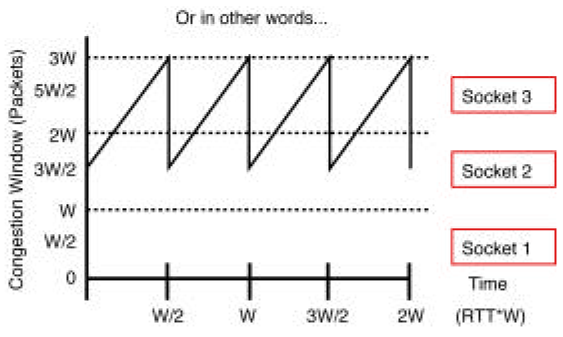

Figure 13. Geometric Construction of the Aggregate Effects of Multiple TCP Connections.

Given that the aggregate rate of congestion recovery across all of the parallel TCP streams is functionally equivalent to an increased recovery rate, there is an interesting observation that can be made. TCP connections over wide area networks suffer from the disadvantage of long round trip times relative to other TCP connections that may have smaller round trip times. This disadvantage allows TCP senders with small RTTs to recover faster from congestion and packet loss events than TCP sessions with longer RTTs. Since the use of parallel TCP sockets provides a higher recovery rate, host with longer RTTs are able to compete on a fairer basis with small RTT TCP connections for bandwidth in the presence of congestion in the network bottleneck.

4.1 Selecting an Appropriate Number of Sockets

When the packet loss rate *p* transitions from the random loss to the congestion loss regime, the benefits from using additional sockets is offset by the additional aggregate packet loss rate across all of the channels. From the previous section, it is apparent that the knee that is present in the TCP bandwidth curve directly corresponds to the knee in the packet loss curve. The challenge in selecting an appropriate number of sockets to maximize throughput is thus the problem of moving up to, but not beyond, the knee in the packet loss curve.

Any application using parallel TCP connections must select the appropriate number of sockets that will maximize throughput while avoiding the creation of congestion. It is imperative that applications avoid congesting the network to prevent congestion collapse of the bottleneck link. As shown by the measurements in the previous section, adding additional TCP connections beyond the knee in the packet loss curve has no additional benefit, and may actually decrease aggregate performance.

Determining the point of congestion in the end-to-end network *a priori* is difficult, if not impossible, given the inherent dynamic nature of the network. However, it may be possible to gather relevant parameters using Web100 from actual data transfers, which then can be used in combination with statistical time-series prediction methods to attempt to predict the end-to-end packet loss rate *p*, *RTT*, and *MSS*, and thus the limit on TCP bandwidth. In addition to using statistical predictions to predict the value of *p*, it may also be possible to use the same techniques to collect and store information on the appropriate number of parallel TCP connections necessary to maximize aggregate performance and avoid congestion. The predicted values of *p* and the effective number of parallel TCP connections can then be used as a starting point for a simple greedy search algorithm that adjusts the number of parallel TCP connections to maximize throughput.

5 Conclusion and Future Work

This paper addressed the question of explaining how parallel TCP connections can improve aggregate TCP bandwidth, the problem of selecting the maximum number of sockets necessary to maximize TCP throughput while simultaneously avoiding congestion. These issues were addressed by developing a theoretical model that was validated by a series of experiments. It was shown in this paper that in the absence of congestion, the use of parallel TCP connections is equivalent to using a large MSS on a single connection, with the added benefit of reducing the negative effects of random packet loss

It is imperative that application developers do not arbitrarily select a value for the number of parallel TCP connections. If the selected value is too large, the aggregate flow may cause network congestion and throughput will not be maximized.

For future work, there are several avenues of research worth pursuing. First, the use of time-series prediction models (such as Network Weather Service [21]) for predicting values of the packet loss rate p and the number of parallel TCP connections (s) would allow application developers to select an appropriate value for s. The ability to predict p would provide a mechanism for Grid computing environments to place an accurate commodity value on available network bandwidth for purposes of trading network bandwidth on an open Grid Computing trading market [44, 45]. Finally, the use of constraint satisfaction algorithms for choosing the optimal value for s by applications should be investigated.

REFERENCES

[1] J. Lee, D. Gunter, B. Tierney, W. Allock, J. Bester, J. Bresnahan, S. Tecke Applied Techniques for High Bandwidth Data Transfers across Wide Area Networks Dec 2000, LBNL-46269.
[2] Harimath Sivakumar, Stuart Bailey, Robert L. Grossman. PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks, SC2000: High-Performance Network and Computing Conference, Dallas, TX, 11/00
[3] Pittsburgh Supercomputer Center Networking Group. “Enabling High Performance Data Transfers on Hosts”, http://www.psc.edu/networking/perf_tune.html
[4] T. V. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. IFIP Transactions C-26, High Performance Networking, pages 135--150, 1994.
[5] Floyd, S., and Fall, K., Promoting the Use of End-to-End Congestion Control in the Internet, IEEE/ACM Transactions on Networking, August 1999.
[6] J. Lee, D. Gunter, B. Tierney, W. Allock, J. Bester, J. Bresnahan, S. Tecke Applied Techniques for High Bandwidth Data Transfers across Wide Area Networks. Sept 2001, LBNL-46269, CHEP 01 Beijing China.

[7] Matt Mathis, Personal Communication.
[8] R. Long, L. E. Berman, L. Neve, G. Roy, and G. R. Thoma, "An application-level technique for faster transmission of large images on the Internet", Proceedings of the SPIE: Multimedia Computing and Networking 1995 Vol. 2417 February 6-8, 1995, San Jose, CA.
[9] Long LR.,Berman LE.,Thoma GR."Client/Server Design for Fast Retrieval of Large Images on the Internet." Proceedings of the 8th IEEE Symposium of Computer-Based Medical Systems (CBMS'95), Lubbock TX June 9-10, 1995 pp.284-291.
[10] Mark Allman, Shawn Ostermann, and Hans Kruse. Data Transfer Efficiency Over Satellite Circuits Using a Multi-Socket Extension to the File Transfer Protocol (FTP). In Proceedings of the ACTS Results Conference. NASA Lewis Research Center, September 1995.
[11] Juerg Bolliger, Thomas Gross, and Urs Hengartner. Bandwidth modelling for network-aware applications. In INFOCOM '99, March 1999.
[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: a simple model and its empirical validation. ACMSIGCOMM, September 1998.
[13] M. Mathis, J. Semke, J. Mahdavi, T. Ott. “The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm.” Computer Communication Review, volume 27, number3, July 1997.
[14] V. Paxson, "End-to-end Internet packet dynamics," in Proc. ACM SIGCOMM, pp. 139--152, September 1997.
[15] Dah-Ming Chiu and Raj Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks," Computer Networks and ISDN Systems, vol. 17, pp. 1--14, 1989.

[16] Internet End-to-End Performance Monitoring. http://www-iepm.slac.stanford.edu/
[17] Lars Eggert, John Heidemann, and Joe Touch. Effects of Ensemble -TCP. ACM Computer Communication Review, 30 (1), pp. 15-29, January, 2000.
[18] H. Balakrishnan, H. Rahul, S. Seshan, "An Integrated Congestion Management Architecture for Internet Hosts", Proc. ACM SIGCOMM, September 1999.
[19] ATLAS High Energy Physics Project. http://pdg.lbl.gov/atlas/atlas.html
[20] Jean-Chrysostome Bolot. "Characterizing End-to-End packet delay and loss in the Internet.", Journal of High Speed Networks, 2(3):305--323, 1993.
[21] R. Wolski. "Dynamically Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service." In 6th High-Performance Distributed Computing, Aug. 1997.
[22] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," Internet Draft, IETF Diffserv Working Group, August 1998. ftp://ftp.ietf.org/internet-drafts/draft-ietf-diffservarch-01.txt
[23] L. Georgiadis, R. Guerin, V. Peris, and K. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. IEEE ACM Trans. on Networking,Aug. 1996.
[24] Sander, V. Adamson, W., Foster, I., Alain, R. End-to-End Provision of Policy Information for Network QoS. In 10th High-Performance Distributed Computing, August 2001.
[25] E. Cohen, H. Kaplan and J. Oldham, "Managing TCP Connections under Persistent HTTP", Proceedings of the Eighth International World Wide Web Conference, Toronto, Canada, May 1999
[26] Grid Forum GridFTP Introduction: http://www.sdsc.edu/GridForum/RemoteData/Papers/gridftp_intro_gf5.pdf
[27] C. Baru, R. Moore, A. Rajasekar, and M. Wan. "The SDSC Storage Resource Broker." In Procs. of CASCON'98, Toronto, Canada, 1998
[28] S. Floyd. "Congestion Control Principles", RFC 2914.
[29] Semeria, C. "Internet Processor II ASIC: Rate-limiting and Traffic -policing Features." Juniper Networks White Paper. http://www.juniper.net/techcenter/techpapers/200005.html
[30] J. Mogul and S. Deering, "Path MTU Discovery," Network Information Center RFC 1191, pp. 1--19, Apr. 1990.
[31] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking, 1(4):397--413, August 1993. http://citeseer.nj.nec.com/floyd93random.html
[32] V. Jacobson. Congestion Avoidance and Control. In Proceedings of the ACM SIGCOMM '88 Conference, pages 314--329, 1988.
[33] M. S. Borella, D. Swider, S. Uludag, G. Brewster, "Internet Packet Loss: Measurement and Implications for End-to-End QoS," Proceedings, International Conference on Parallel Processing, Aug. 1998.
[34] W. Feng and P. Tinnakornsrisuphap, "The Failure of TCP in High-Performance Computational Grids", SC2000: High-Performance Network and Computing Conference, Dallas, TX, 11/00
[35] Web100 Project. http://www.web100.org
[36] Mark Gates and Alex Warshavsky, Iperf version 1.1.1, Bandwidth Testing Tool, NLANR Applications, February 2000.
[37] Raj Jain. The Art of Computer Systems Performance Analysis. John Wiley & Sons, Inc., New York, New York, 1991.
[38] S. Floyd. "TCP and Explicit Congestion Notification." ACM Computer Communication Review, 24(5):10-23, Oct. 1994.
[39] Floyd, S., and V. Jacobson. 1992. *On traffic phase effects in packet-switched gateways*. Internetworking: Research and Experience 3:115--156.
[40] L. Guo, M. Crovella, and I. Matta, "*TCP congestion control and heavy tails*," Tech. Rep. BUCSTR -2000-017, Computer Science Dept - Boston University, 2000.
[41] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgement Options. RFC 2018, Proposed Standard, April 1996." URL *ftp://ftp.isi.edu/in-notes/rfc2018.txt*
[42] V. Jacobson, R. Braden, D. Borman, "RFC1323: TCP Extensions for High Performance", May 1992
[43] McGill, Tukey and Larsen, "Variations of Box Plots," Am. Statistician,Feb. 1978, Vol. 32, No. 1, pp. 12-16
[44] Buyya, R., Abramson, D., Giddy, J. "An Economy Grid Architecture for Service-Oriented Grid Computing", 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), In Conjunction with IPDPS 2001, San Francisco, USA, April 2001.
[45] Hacker, T., Thigpen, W. "Distributed Accounting on the Grid", Grid Forum Working Draft.