Improving Throughput and Maintaining Fairness using Parallel TCP

Thomas J. Hacker
Center for Advanced Computing
University of Michigan
Ann Arbor, Michigan 48109
Email: hacker@umich.edu

Brian D. Noble
Electrical Engineering and
Computer Science
University of Michigan
Ann Arbor, Michigan 48109
Email: bnoble@eecs.umich.edu

Brian D. Athey
Michigan Center for
Biological Information
University of Michigan
Ann Arbor, Michigan 48109
Email: bleu@umich.edu

***Abstract*— Applications that require good network performance often use parallel TCP streams and TCP modifications to improve the effectiveness of TCP. If the network bottleneck is fully utilized, this approach boosts throughput by unfairly stealing bandwidth from competing TCP streams. Improving the effectiveness of TCP is easy, but improving effectiveness while maintaining fairness is difficult. In this paper, we describe an approach we implemented that uses a long *virtual round trip time* in combination with parallel TCP streams to improve effectiveness on underutilized networks. Our approach prioritizes fairness at the expense of effectiveness when the network is fully utilized. We compared our approach with standard parallel TCP over a wide-area network, and found that our approach preserves effectiveness and is fairer to competing traffic than standard parallel TCP.**

***Index Terms*— System design, Network measurements, Experimentation with real networks/Testbeds**

I. Introduction

The ability to move large amounts of data quickly over a shared wide area network is a necessity for many applications today. The Atlas project [1] must be able to move many petabytes of data per year between Europe and the United States. Multiuser collaborative environments combine visualization, video conferencing, and remote application steering into a distributed application with low latency and high bandwidth demands [2], [3]. The Optiputer project [4] aims to build a distributed computer using a wide area network path for a backplane. Other tools such as GridFTP [5], bbcp [6], DPSS [7], and PSockets [8] are used by applications that need to move large amounts of data over wide area networks.

Many of these applications use the Transmission Control Protocol (TCP) for accurate and reliable in-order transmission of data. TCP relies on the congestion avoidance algorithm (CAA) [9] to measure the capacity of a network path, fairly share bandwidth between competing TCP streams, and to maximize the effective use of the network.

Unfortunately, the CAA prevents the effective and efficient use of wide area networks for these applications. The CAA relies on packet loss to indicate that the network is overloaded, and responds to packet loss by decreasing the transmission rate of the TCP stream by half. Van Jacobson [9] assumed three things about packet loss when he designed the CAA – the fraction of non-congestion packet loss ("damaged packets") is $\ll 1\%$; network speeds (circa 1988) are low enough to prevent measurable sensitivity to low ($< 5\%$) loss rates; and an implicit assumption that packet reordering would not affect the congestion avoidance algorithm. Jacobson found that the CAA is sensitive to packet loss when the packet loss rate is the same order of magnitude as the square of the congestion window in packets. This has become a problem for high-speed networks, since the maximum frame size is usually fixed at 1500 bytes [10].

There is a substantial body of evidence that effects such as packet reordering and non-congestion packet loss can degrade TCP performance over wide area networks. The non-congestion losses perceived by TCP are due to many factors unrelated to simple network congestion that include end-host [11]–[15] and network infrastructure effects [16]–[19]. These effects on wide-area high-speed networks can lead to situations in which it can require over 4 hours for a TCP stream to completely recover from one loss event [20]. Non-congestion loss also has significant effects on satellite links [21]. There are efforts within the networking community to design modifications to TCP congestion avoidance to overcome these effects. Most of these efforts rely on mechanisms that aggressively compete for network bandwidth, which can lead to unfair behaviors at best, and congestion collapse at worst.

In this paper, we describe an approach that modifies the aggressive tendencies of parallel TCP to allow an application to consume unused bandwidth when the network is underutilized. When the network is fully utilized, our approach is substantially fairer to competing traffic than an equivalent number of parallel TCP streams.

II. Background

In previous work [22], we performed a series of network measurements between the University of Michigan in Ann Arbor and NASA AMES in Moffett Field, CA over a high speed path to determine if parallel TCP streams were effective at improving throughput, and to gain insight on the relationship between the number of parallel TCP streams and performance. We found that as the number of TCP sockets used in the

parallel stream increased, aggregate throughput increased linearly until the $bandwidth * delay$ product of the network was achieved.

We were convinced that parallel TCP was effective, but were concerned about the effects of parallel TCP on other network streams. To gain deeper insight into the effects of parallel TCP, we performed a series of simulations to measure the effects of parallel TCP on effectiveness and fairness [23]. The simulation loss model was based on loss characteristics observed from data transfers over a wide area network. We found that the simulation loss rate was high enough to prevent a single TCP stream from fully utilizing available network bandwidth, and that a set of parallel TCP streams could collectively consume unused bandwidth without stealing bandwidth from competing traffic. However, when the network was fully utilized, parallel TCP could improve performance only by unfairly stealing bandwidth from competing traffic.

In simulation, we discovered that a set of parallel TCP streams that coupled one standard TCP stream with a group of TCP streams with reduced aggressiveness could effectively consume unused bandwidth, yet were unable to steal bandwidth from standard TCP streams when the network was busy. We found that we could adopt this method to retain the desirable effectiveness of parallel TCP and substantially reduce unfairness when the network was busy.

Our implementation modulates the aggressiveness of an individual TCP stream by increasing the number of ACKs necessary to increase the congestion window. This has the effect of creating a *virtual* long round trip time (RTT) for the stream. Our approach exploits a behavior of TCP in which long RTT TCP streams are unable to compete with short RTT TCP streams for available network bandwidth [24]. We call these modulated TCP streams *fractional* streams because their aggressiveness is a fixed fraction of the aggressiveness of a single unmodulated TCP stream. We call the combination of the unmodified single TCP stream and the set of fractional streams *Combined Parallel TCP*. This paper describes the implementation and test of this modification on a real network.

We believe that the work described in this paper contributes to existing work in several ways. First, our *Combined Parallel TCP* approach retains the TCP-friendly characteristics of each TCP stream. We exploit a behavior of TCP that does not fundamentally change the AIMD characteristics of TCP. Second, our work substantially retains effectiveness *on underutilized networks* and *automatically* prioritizes fairness *over* effectiveness when the network is busy. Finally, our approach provides a mechanism to *dynamically tune* the aggressiveness of parallel TCP using the *virtual RTT* factor.

The next section of this paper describes other approaches used to improve TCP performance over wide area networks.

III. Related Work

Many papers have been written on the topic of improving TCP performance for long-lived TCP streams on high-speed networks. Scalable TCP [20] improves TCP performance by using a fixed additive increase (AI) value and a multiplicative decrease (MD) factor that is less than the *1/2cwnd* factor used by the standard TCP congestion avoidance algorithm. These AIMD factors make Scalable TCP more aggressive than standard TCP by seizing more bandwidth while probing network capacity and releasing less bandwidth on packet loss. Using these AIMD modifications, a Scalable TCP stream competing with a standard TCP stream on a shared network path with similar round trip times will not seek the optimal fairness and efficiency point in the phase diagram of Chiu and Jain [25]. Thus, the use of competing Scalable TCP flows on a busy network will lead to a higher packet loss rates and reduce the amount of effective work (”goodput”) performed by the network.

PSockets [8] is an application level library that opens concurrent TCP streams to increase aggregate TCP throughput. PSocket exploits the fact that a set of *N* TCP streams is functionally equivalent to a *virtual Maximum Segment Size (MSS)* of *N * MSS* of the MSS used by a single TCP stream. Parallel TCP streams are aggressive on a shared network and can steal bandwidth from competing TCP streams. Previous work [22] showed that a single application using *N* parallel TCP streams competing with *k* other streams will receive *N/(N+k)* of the network bandwidth, rather than the *1/(N+k)* portion the other streams receive.

High Speed TCP (HSTCP) [16] is designed to ameliorate the effects of media bit error rates on long-lived TCP streams over wide-area networks. HSTCP modifies the AIMD parameters of the standard TCP congestion avoidance control system as a function of the current *cwnd* and observed packet loss rate. This parameter modification is designed to gently probe network capacity during the early startup phase of a HSTCP stream (when *cwnd* is small). After the gentle startup phase, when *cwnd* is large, HSTCP releases substantially less network bandwidth than *1/2cwnd* on packet loss, and recovers more aggressively than standard TCP during the AI phase. When the modified AIMD parameters are is use, HSTCP is unfair to competing unmodified TCP streams. Moreover, due to its aggressive tendencies, HSTCP is unsuitable for use on networks slower than 10 Gb/sec, limiting its applicability.

FAST TCP [26] uses both packet loss and increased packet round trip time as indications of network congestion. FAST TCP implicitly assumes that the minimum round trip time value observed during a TCP session is the propagation delay (transmission delay with no queueing) on the end-to-end network path. However, if FAST TCP is used on networks with persistently populated queues, it will incorrectly infer the propagation delay of the network and behave more aggressively than competing unmodified TCP streams. Biaz [27] found that RTT measurements do not adequately represent queue length, and should not be used as a predictor of congestion. Measurements [26] show that FAST TCP is unfair to competing unmodified TCP streams using network measurements from CERN to the United States over a 2.4 Gb/sec network.

Tsunami [28] is a new reliable transport protocol that uses UDP as an unreliable datagram delivery service to improve

end-to-end application performance. Tsunami is an example of a class of protocols that improves throughput by using UDP to bypass congestion avoidance. Applications that implement or utilize UDP based protocols such as Tsunami without TCP-friendly congestion avoidance put the Internet at risk of suffering excessive congestion induced packet loss events and congestion collapse.

Congestion Manager [29] is a software framework that can be used to implement alternative congestion avoidance mechanisms. The binomial congestion control algorithm [30] is a nonlinear control system that generalizes the AIMD class of congestion control that can be used with Congestion Manager. Binomial congestion control parameters that are TCP-compatible can be selected to improve TCP performance over lossy wide-area networks. Unfortunately, binomial congestion control is unfair to competing unmodified TCP streams across networks with droptail queue routers. Since RED [31] has not been widely deployed, binomial congestion control is not suitable for use on public production networks.

The important lesson that can be learned from the approaches described in this section is that increasing throughput (effectiveness) is easy, but improving effectiveness while maintaining fairness is difficult. Finding the balance between effectiveness and fairness is the primary motivation for the work described in this paper.

IV. Combined Parallel TCP Streams

Our goals in designing and implementing the combined parallel TCP approach is to improve effectiveness on an underutilized network and maintain fairness at the expense of effectiveness if the network is busy. The existing aggressive approaches described in the previous section improve effectiveness at the expense of fairness when the network is busy.

Our combined approach exploits the fact that short round trip time TCP streams dominate long round trip time TCP streams when both are competing for network bandwidth [24], [32]. When the network link is uncongested, either due to underutilization or systemic, non-congestion packet loss, the combined set of parallel TCP streams will consume the unused bandwidth without appropriating bandwidth from existing single TCP streams. When the network is congested, the single unmodified TCP stream of the set of combined parallel TCP streams will effectively compete with other TCP streams to secure at least one stream's fair portion of network bandwidth. The fractional components of the combined parallel TCP streams will not be able to compete effectively with other unmodified TCP streams, and will each secure much less than a single stream's fair share of the network bandwidth.

Simulation results from previous work [23] shows that combined parallel TCP streams are more effective than a single unmodified TCP stream when the network is underutilized, and are substantially fairer than an equivalent number of unmodified parallel TCP streams.

Ideally, we could make use of a mathematical relationship between the round trip time of an unmodified TCP stream and the virtual round trip time multiplier used for the fractional streams in a combined parallel TCP stream to predict the optimal trade-off between effectiveness and fairness. Although the literature has shown the effects of long round trip times on fairness, there is not a relationship that we can use to tightly bound the minimum and maximum bandwidth for each fractional stream. Lakshman [32] empirically discovered that the effect of round trip time on throughput when two TCP streams are competing for bandwidth is proportional to $RTT^{-\alpha}$, where $1 \leq \alpha \leq 2$. Unfortunately, the lower and upper bounds of this factor are too far apart to determine a tight bound on the expected throughput for a fractional multiplier.

To understand the effects of combined parallel TCP streams and the virtual round trip time multiplier on fairness, there are two scenarios to consider. The first scenario is an underutilized network, in which the majority of packet loss is due to latent systemic non-congestion sources (P_L). In this scenario, the TCP throughput of an individual stream is limited by P_L when the structural capacity of the bottleneck is greater than

$$\frac{C * MSS}{RTT\sqrt{P_L}} \tag{1}$$

. This condition currently exists on intercontinental 10 Gb/sec networks because the inherent fiber bit error rate (P_L) is large enough to limit the throughput of a single TCP stream [16]. In this situation, if we assume that P_L is independent of load, the latent loss P_L independently affects the throughput of each TCP stream. Parallel TCP streams can take advantage of this situation without significantly affecting fairness because each TCP stream does not affect the latent packet loss rate of the other streams, as long as the bandwidth capacity of the network bottleneck is not reached. This allows both the unmodified and combined parallel TCP approaches to utilize bandwidth without adversely affecting other TCP streams.

The second scenario to consider is when packet loss occurs due to network congestion (P_c) and total packet loss $P_T = P_L + P_C$. When several TCP streams with equivalent round trip times are competing for bandwidth on a congested network, they "negotiate" with each other by probing the network to determine how P_C will be distributed to each TCP stream at equilibrium. If the RTT for each stream is identical, a fair distribution of the bottleneck bandwidth will be realized by a "fair" distribution of P_C (each stream of *n* streams will have $P_i = P_T/n$). However, if one of the streams in the negotiation has a longer RTT value than the other TCP streams, it will converge to a smaller "unfair" bandwidth share by being forced to accept a P_T value larger (worse bandwidth) than would be required to achieve a fair bandwidth share with a higher RTT value. From examining the relationship

$$TCP_{BW} \leq \frac{C * MSS}{RTT\sqrt{p}} \tag{2}$$

[33], it is clear that as RTT increases, a proportionate decrease in p^2 is required to maintain the same bandwidth. Thus, the sensitivity to loss for a long RTT stream is quadratic in P_T.

For our combined approach, the increased sensitivity to P_T provides the mechanism by which the long virtual RTT

streams can reduce their bandwidth when competing with other streams. It is important to note that as the number of fractional TCP streams is increased, attempts to reduce the aggregate aggressiveness of the combined streams by increasing the virtual RTT multiplier will greatly increase the sensitivity to P_T.

For this paper, we selected a fractional multiplier of 10 for all of our experiments, which is functionally equivalent to a virtual round trip time of 10 times the RTT and 1/10 the aggressiveness of an unmodified single TCP stream. In practice, selection of the fractional multiplier is a tradeoff between aggressiveness and effectiveness (large multiplier) and fairness (small multiplier). To prevent excessively aggressive behavior, the product of the fractional multiplier and the number of fractional streams in the set of combined parallel TCP streams should be less than 1. This fractional multiplier value will ensure that the set of fractional streams will be less aggressive than a single unmodified TCP stream. Since our measurements were made on a wide area network path from University of Michigan to California Institute of Technology, we were confident that our fractional TCP streams with a virtual RTT multiplier of 10 would likely have the longest RTT of any of the TCP streams on the network path. [1]

The next section of this paper describes the modifications made to the Linux kernel to support combined parallel TCP. The sections following discuss the methodology and evaluation that demonstrates the effects of combined parallel TCP on effectiveness and fairness.

V. Implementation

To mimic the effects of a long round trip time on a TCP connection, the function tcp_cong_avoid() in net/ipv4/tcp_input.c was modified. Normally, tcp_cong_avoid() increments the congestion window size *cwnd* by one for every new data packet ACKed when $cwnd \leq ssthresh$, and by $1/cwnd$ if $cwnd > ssthresh$. In practice, floating point calculations are not supported in the kernel (for portability), so an integer variable is maintained that counts up the number of ACKed packets for the case in which $cwnd > ssthresh$. To supplement this algorithm for a long virtual RTT, we placed another integer counter (*float_snd_cwnd*) within the integer ACK counter *snd_cwnd_cnt* to accumulate a fractional factor *frac_cwnd* that is set by a call to setsockopt() for the TCP socket. The fractional factor *frac_cwnd* is an integer $\in [1, 1000]$, where a value of 1000 is equivalent to 1 * RTT, a value of 100 is equivalent to 1/10 * RTT, and so on. When the accumulated $frac\_cwnd \geq 1000$, *cwnd* is incremented. So, if *frac_cwnd* is 500, the number of ACKs required to increase *cwnd* by one is doubled, which in terms of the rate of increase of *cwnd* is functionally equivalent to a TCP stream in which the ACKs take twice as long to arrive at the sender, or 2 * RTT. This modification (less than 20 lines of code) was made for both the exponential and linear increase phases of the congestion avoidance algorithm. A new getsockopt() and setsockopt() option was added to allow a user level application to query and set the *frac_cwnd* value. A default value of 1000 is used in the kernel if the value is not explicitly set.

We did not modify congestion avoidance behavior when a packet loss occurs ($cwnd \leftarrow 1/2cwnd$). Thus, only the aggressiveness of the TCP congestion avoidance AI phase was modified to allow a user level application to reduce the rate of growth of *cwnd* for a fractional stream, but the multiplicative decrease factor of 1/2 was not changed to ensure that a modified TCP stream would respond identically to an unmodified TCP stream on packet loss.

VI. Methodology

We assessed the congestion avoidance algorithm modifications described in the previous section to compare the effectiveness and fairness of our combined parallel TCP method with unmodified parallel TCP streams.

A. Measuring Effectiveness

We define the effectiveness of a TCP transmission method to be the ability of the method to move data between two hosts in a time period $t \in [0, \tau]$, given the constraints of congestion avoidance and competition with other streams for network bandwidth. We used two measurement techniques to assess the effectiveness of a single unmodified TCP stream, a set of unmodified parallel TCP streams, and our combined parallel TCP streams method.

The first measurement technique measures the ability of a TCP transmission method to compete with a "stiff" time varying UDP stream. The UDP stream is considered "stiff" because, unlike a TCP-friendly stream, the UDP stream does not modify its transmission rate in response to changes in network RTT or packet loss events. By varying the transmission rate of the UDP stream as a function of time, we can assess the ability of the TCP transmission method to utilize the remaining network capacity. By interleaving a sequence of ICMP packets using fping [35] at 10 msec intervals with the UDP stream and measuring the round trip time of the ICMP packets, we can measure the impact of the TCP transmission method on queueing delays induced by the total network load. The UDP stream will experience packet loss due to queueing loss when the network becomes congested. We measure the UDP packet loss rate and use it in combination with the ICMP round trip time to assess the overall impact of the TCP transmission method on the network.

We modified Iperf [36] to generate a UDP stream with throughput controlled by the equation

$$f(t) = Avg[\cos(t/c)] + Min + Avg \quad (3)$$

where *Max* and *Min* are the maximum and minimum throughput rate of the UDP stream, $Avg = (Max - Min)/2$, and $c = 0.05\pi(period)$, and *period* is the period of the sinusoidal

[1] The IEPM Project [34] at Stanford University Linear Accelerator Center (SLAC) routinely measures the average RTT between continents. For the month of March 2003, the average RTT from North America to Caltech was 43.31 ms, and the largest average RTT from each continent (except Antarctica) to SLAC was 260.07 ms. With a virtual RTT multiplier of 10, our minimum virtual RTT from U-M to Caltech was approximately 680 ms.

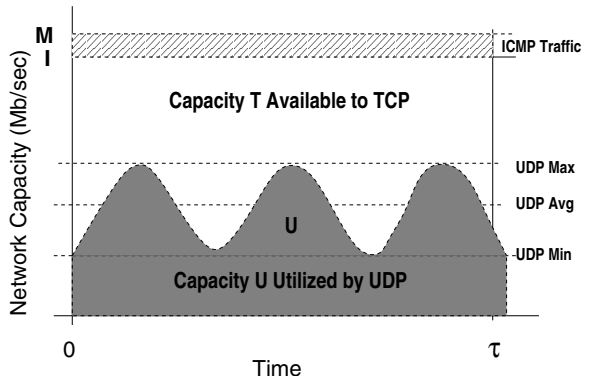

Fig. 1. Relationship between ICMP, TCP, and UDP Traffic

throughput of the UDP stream. Figure 1 shows the relationship between the ICMP, UDP and TCP traffic on the network.

We can calculate the number of bytes available to TCP by subtracting the number of bytes consumed by the UDP stream from time $t \in [0, \tau]$ from the theoretical maximum number of network bytes $M - I$ that can be transferred on the network link of capacity *M* Mb/sec over the same time period. *I* represents the ICMP test traffic generated by fping. Let

$$g(t) = M - I \tag{4}$$

To calculate theoretical TCP capacity *T*, we simply need to calculate the difference in area between g(t) and f(t) over $t \in [0, \tau]$.

$$T = \int_0^\tau g(t)\,dt - \int_0^\tau f(t)\,dt \tag{5}$$

$$T = \int_0^\tau [(M - I) - U]\,dt \tag{6}$$

Substituting $f(t)$ for *U*,

$$T = \int_0^\tau [(M - I) - Avg\{cos(t/c) + Min + Avg\}]\,dt \tag{7}$$

which after integration reduces to

$$T = (M - I)t - Avg[c\,sin(t/c) + t\,(Min/Avg) + t] \tag{8}$$

This equation represents the theoretical maximum number of bytes the network can transmit for a network stream over time $t \in [0, \tau]$. If a TCP stream is perfectly efficient, it should be able to utilize 100% of the transmission capacity available to it. In practice, the TCP congestion avoidance algorithm does not fully utilize the available transmission capacity, since it probes the capacity of the network by changing the instantaneous rate of transmission. To measure the ability of a TCP transmission method to improve TCP throughput (effectiveness), we use this equation to determine an upper bound on TCP efficiency given a sinusoidal UDP stream $f(t)$ with parameters *Max*, *Min*, and *period*.

Selection of the time period of the UDP stream throughput has two constraints. If the period is too small, the TCP stream's RTT ACK clocking will not have enough time to probe available bandwidth. If the period is too long, the available bandwidth as seen by the congestion avoidance algorithm will appear to be static. We selected a time period of 10 seconds for our experiments. This value is large enough to ensure that the long RTT TCP streams will be able to probe available network capacity, but not so large as to make the UDP stream appear to be flat and unchanging to a TCP stream.

The second measurement technique we used to assess the efficiency of a TCP method was to fully load the network with 5 competing single unmodified TCP streams. We used this technique to compare the efficiency of unmodified parallel TCP streams with the same number of combined parallel TCP streams when the network is busy.

B. Measuring Fairness

Our other goal was to compare the fairness of unmodified parallel TCP streams with our combined parallel TCP method. We define fairness as the ability of a TCP transmission method to effectively use at least one unmodified TCP stream's fair share of bandwidth without appropriating bandwidth from other competing TCP streams.

We devised two techniques to assess fairness. The first technique measures changes in the RTT and the square root of the packet loss rate. These values are related to TCP bandwidth by the fundamental relationship

$$BW \leq \frac{C * MSS}{RTT\sqrt{p}} \tag{9}$$

described by Mathis [33]. Looking at this equation, it is clear that if RTT or the square root of the packet loss rate increases, the TCP bandwidth of an individual stream is reduced.

To measure RTT, we used the fping utility [35] to inject ICMP packets into the network at a fixed rate of 1 packet every 10 ms. To measure the square root of the packet loss rate we monitored the UDP packet loss rate reported by the "stiff" varying UDP stream generated by our modified Iperf. Since the varying Iperf UDP stream is "stiff" and does not adjust to network load, when the queue in the network bottleneck overflows, UDP packets will be lost, along with ICMP and TCP packets. Thus, an increase in the packet loss rate reported by Iperf indicates that the queue is becoming overloaded. An increasing RTT indicates an increase in the queue length at the network bottleneck. Using this technique, we use the UDP stream as a proxy TCP stream that does not respond to network congestion. We used the relative values of RTT and packet loss to assess the relative fairness of each TCP transmission method.

The second technique we used to assess fairness was to create 5 competing unmodified single TCP streams. We measured the per-stream throughput and the total throughput of the competing TCP streams using Iperf. A decrease in the aggregate throughput of the competing TCP streams indicates that the parallel TCP method improved its efficiency by stealing bandwidth from the 5 competing TCP streams.

C. Experimental Setup

Figure 2 shows the network testbed configuration. The measurement equipment on the sending side (Server Host

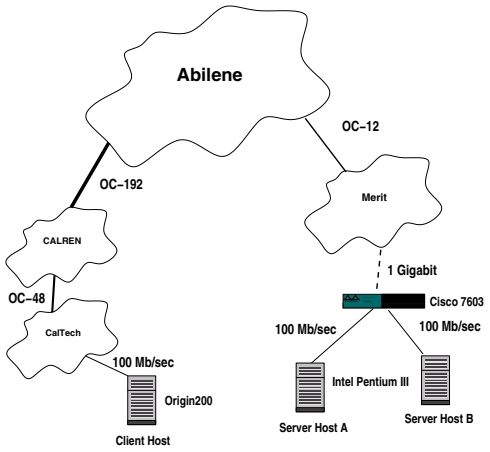

Fig. 2. Testbed Network

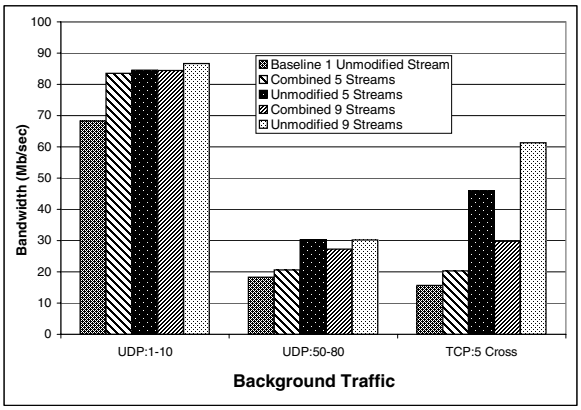

Fig. 3. Effectiveness of TCP Methods

A and Server Host B) consisted of two dual processor 800 MHz Intel Pentium III computers with 512 MB of RAM running on an Intel L440GX+ Lancewood motherboard. The tests were conducted using the onboard 100 Mb/sec Intel LAN controller. Each system used a Linux 2.4.19 kernel with the TCP modifications described earlier. The TCP send and receive socket buffer sizes were set to 4 MB to eliminate buffer size restrictions on TCP throughput, and the cached TCP parameters for each experiment were explicitly flushed before each test. The tests were conducted using a shared Cisco 7603 switch on the sender side with 100 Mb/sec line cards. The switch was connected via a lightly loaded gigabit ethernet connection to a University of Michigan test gigabit ethernet backbone. The backbone was connected to Abilene via a 622 Mb/sec OC-12 network link. The sending hosts were located at the University of Michigan in Ann Arbor, Michigan and the receiving host was a SGI Origin 200 (Client Host) running Irix 6.5 located at California Institute of Technology in Pasadena, California. The Origin 200 was connected via a 100 Mb/sec link to through the California Research and Education Network (CALREN) to the Abilene network. The experiments were conducted between February and April 2003.

Each experiment consisted of 4 minute TCP transfers from one host running concurrently with a competing 4 minute UDP or TCP transmission from the other host. The ICMP traffic was a 1440 byte packet sent every 10 msec from the UDP/TCP host using fping. Each experiment was initiated at 30 minute intervals for several days, and included at least one weekend day to include diurnal and weekly effects of network load on the measurements.

VII. Evaluation

A. Effectiveness Results

Figure 3 shows the effectiveness of the baseline single TCP stream, our combined method, and the unmodified parallel TCP method. The category *UDP:1-10* represents effectiveness for experiments competing with the time-varying UDP stream that ranged from 1 to 10 Mb/sec, *UDP:50-80* represents the range 50 to 80 Mb/sec. *TCP 5 Cross* represents the effectiveness of each TCP method competing with 5 unmodified single TCP streams. The first column in each category is the median throughput for the baseline single unmodified TCP stream, the second is the median throughput for the combined method with 5 streams, the third is the median throughput for the unmodified parallel TCP method with 5 streams, the fourth is the median throughput for the combined method with 9 streams, and the fifth is the median throughput for the unmodified parallel approach with 9 streams.

TABLE I
TCP Effectiveness for UDP Stream ∈ [1,10] Mb/sec and Theoretical Capacity T = 2801.5 MB

TCP Method	Throughput Mb/sec	Measured MB	Achieved %
Baseline 1	64.5	1843.2	65.8%
Combined 5	83.54	2402.5	85.75%
Combined 9	84.46	2441.6	87.15%
Parallel 5	84.5	2430	86.74%
Parallel 9	86.63	2464	87.95%

Table I shows the median number of bytes transferred over the network for each method, the maximum theoretical throughput calculated using Equation 8, and the fraction of theoretical capacity achieved by the TCP method competing with a UDP stream ranging from 1 to 10 Mb/sec. Table II shows the results for each method competing with a UDP stream ranging from 50 to 80 Mb/sec. Table III shows the results for each method competing with 5 TCP cross traffic streams. The M value used in Equation 8 for these experiments was 100 Mb/sec.

B. Effectiveness Discussion

From this data, it is clear that both unmodified and combined parallel TCP streams both increase aggregate throughput and the number of bytes transferred compared to a single unmodified TCP stream. When competing with a ”stiff” UDP stream ∈ [1,10] Mb/sec, the throughput of our combined method is almost equivalent to the throughput of the same number of unmodified parallel TCP streams. In the case of

TABLE II

TCP EFFECTIVENESS FOR UDP STREAM ∈ [50,80] MB/SEC AND THEORETICAL CAPACITY T = 1014.6 MB

TCP Method	Throughput Mb/sec	Measured MB	Achieved %
Baseline 1	18.25	524	51.65%
Combined 5	20.6	601.8	59.31%
Combined 9	27.3	798	78.65%
Parallel 5	27.6	800	78.85%
Parallel 9	30.2	876.6	86.4%

TABLE III

TCP EFFECTIVENESS FOR 5 CROSS TCP STREAMS

TCP Method	Throughput Mb/sec	Measured MB
Baseline 1	15.65	450.5
Combined 5	20.27	590.3
Combined 9	29.84	868.6
Parallel 5	45.96	1328.5
Parallel 9	61.33	1779

TABLE IV

FAIRNESS FOR UDP STREAM ∈[1,10] MB/SEC

TCP Method	RTT (msec)	$\sqrt{p}$
Baseline 1	68.65	0.261
Combined 5	87.14	0.332
Combined 9	100.51	0.332
Parallel 5	73.01	0.436
Parallel 9	74.94	0.648

TABLE V

FAIRNESS FOR UDP STREAM ∈[50,80] MB/SEC

TCP Method	RTT (msec)	$\sqrt{p}$
Base	67.92	0.158
Combined 5	68.06	0.200
Combined 9	79.42	0.298
Parallel 5	69.57	0.313
Parallel 9	120.27	0.424

UDP stream ∈ [50,80] Mb/sec, 5 combined parallel TCP streams are slightly better than a single unmodified TCP stream, and 9 combined parallel TCP streams are nearly equivalent to 9 unmodified parallel TCP streams. These results demonstrate that our combined method can consume unused network bandwidth if competing network streams are unable to fully utilize the network. The varying UDP stream is similar to a competing TCP stream that is unable to fully probe and utilize the capacity of the network. This situation arises when the TCP socket buffers are too small, or when the systemic non-congestion packet loss rate is sufficiently high enough to limit TCP throughput [23].

The second observation is that a single TCP stream is not effective at using network capacity over a long period of time. For the UDP stream ranging from 1 Mb/sec to 10 Mb/sec shown in Table I, there is approximately 2.8 GB of transmission capacity over a period of 240 seconds available to TCP. The single unmodified stream only utilized 65.8% of the available capacity. In contrast, 9 unmodified parallel TCP streams used 87.95% of the capacity. Our combined approach was not as effective as unmodified TCP parallel streams, but was more effective than a single unmodified TCP stream.

We were surprised by how ineffective the TCP congestion avoidance algorithm was at consuming network capacity. This ineffectiveness should be further investigated to understand the relationship between TCP streams and the aggregate utilization rate of the network over a long period of time.

C. Fairness Results

We have shown that our combined approach is effective. This section considers the question: effective at what cost? There are many methods that improve effectiveness at the cost of fairness. Our goal was to create a method to improve efficiency and maintain fairness during competition for bandwidth.

From the first fairness assessment technique using the time-varying UDP stream, Table IV shows the median RTT and square root of the measured packet loss rate measured using fping and our modified Iperf for each assessed TCP method. This table contains measurements for the UDP stream ranging from 1 to 10 Mb/sec. Table V contains the median RTT and square root of the median packet loss rate for the UDP stream ranging from 50 to 80 Mb/sec. The RTT and packet loss rate are factors in the Mathis [33] and Padhye [37] TCP equations.

The second technique we used to assess fairness was based on measuring the impact of unmodified parallel TCP streams and our combined parallel TCP streams on 5 competing unmodified single TCP streams. To establish a baseline to compare fairness, we used a single unmodified TCP stream to compete with the 5 reference streams. Figure 4 shows the aggregate throughput of the 5 competing unmodified TCP streams for each evaluated TCP method. The central category (Baseline 1 Unmodified Stream) is the median aggregate throughput of the 5 competing TCP streams competing with a single unmodified TCP stream. The categories to the left of baseline are our combined parallel TCP approach with 5 and 9 parallel streams. The categories to the right of baseline are for the unmodified parallel TCP streams approach with 5 and 9 parallel streams. The measured baseline median aggregate throughput was 75.3 Mb/sec.

The use of unmodified parallel TCP streams resulted in an appropriation of 38.7% of the bandwidth for 5 parallel streams, and 58% of the bandwidth for 9 parallel streams. In contrast, our combined parallel TCP approach resulted in 6.2% appropriation for 5 parallel streams, and 17.5% appropriation for 9 parallel streams.

D. Fairness Discussion

Using the varying UDP stream measurement technique, the unmodified parallel TCP streams increased the median RTT and packet loss rate compared with a single unmodified TCP stream. Our combined parallel TCP approach resulted in a median RTT larger than the unmodified single and parallel TCP streams for the UDP stream $\in$[1,10] Mb/sec, and a median RTT smaller than unmodified parallel TCP streams for UDP $\in$[50,80] Mb/sec.

The median measured packet loss rate for our approach was lower than the packet loss rate for the unmodified parallel TCP approach. We believe that this difference is evidence of increased utilization of the queue with fewer queue overflow events. This makes sense, since our combined approach uses substantially less aggressive flows that probe queue capacity less forcefully than unmodified TCP. This difference leads to a "smoother" set of streams with less bursty and self-synchronized traffic behaviors which are more likely to overflow the queue. Yang [38] found that smoothness, defined as small sending rate variations over time for a particular flow, and fairness are positively correlated. We believe that the negative effects of increased RTT are offset by the decreased packet loss rate, which occurs when unmodified parallel TCP "overshoots" the bandwidth-delay product of the network that it is trying to measure.

The TCP congestion avoidance algorithm reacts drastically to a lost packet by halving the congestion window *cwnd*. The effects of increased RTT in the absence of loss on *cwnd* is less pronounced. Since the congestion avoidance algorithm is clocked by returning ACKs from the receiver, an increasing RTT has the effect of slowing the ACK clock. This will decrease the aggressiveness of the TCP stream as it recovers from a packet loss event, but not decrease the transmission rate.

When our combined approach competes with an unmodified TCP stream, an increase in RTT due to queueing delays affects all streams to the same extent, since our approach extends the current ACK clocking rate to create an extended virtual RTT. Thus, when queue utilization increases without an increase in the loss rate, our approach should not steal more bandwidth from competing unmodified TCP streams.

Given this evidence, we believe that our combined parallel TCP approach does a better job of probing network capacity without inducing oscillations in throughput that cause more queueing overflows. Thus, based on using a varying UDP stream that approximates a "stiff" TCP flow to measure RTT and packet loss, our approach is fairer than unmodified parallel TCP streams.

Our combined parallel TCP approach was substantially fairer than unmodified parallel TCP streams when competing for bandwidth with five single TCP streams. However, we were concerned about the large appropriation for 9 combined parallel streams, and wondered if it was possible to reduce the appropriation by increasing the fractional multiplier from 10 to 100 (giving us 1/100 the maximum theoretical bandwidth of an unmodified TCP stream). We performed an experiment with our combined method with a virtual RTT multiplier of 100. The effectiveness was reduced (18.5 Mb/sec vs. 29.8 Mb/sec), but was still greater than the 15.65 Mb/sec throughput of the single unmodified TCP stream. The appropriation was substantially reduced from 17.5% to 2.9%. This experiment clearly demonstrates to us the tradeoff between effectiveness and fairness that must be considered when using aggressive TCP implementations.

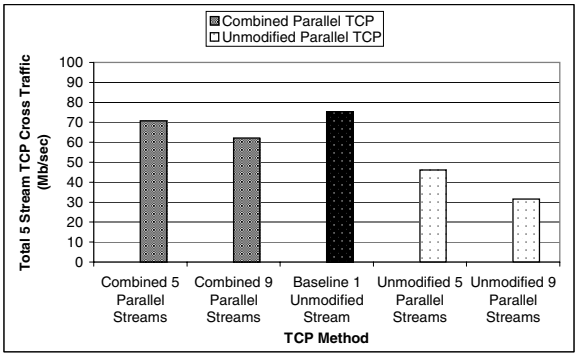

Fig. 4. Fairness of TCP Methods Competing with 5 Single TCP Streams

Thus, using the technique of measuring appropriation from competing single TCP streams, our combined method is substantially more fair than the unmodified parallel TCP approach. With judicious selection of the number of streams and virtual RTT factor, our approach provides a way to balance effectiveness and fairness for an application.

E. Effectiveness and Fairness Conclusions

In this section, we showed that our combined parallel TCP approach is more effective than an unmodified single TCP stream, almost as effective as unmodified parallel TCP stream when there is available network bandwidth, and much less effective than unmodified parallel TCP when the network is congested. We showed that our combined approach is fairer than unmodified parallel TCP streams in both congested and uncongested networks, due to the effects of a long virtual RTT on competition for network bandwidth. We showed that our approach is not perfectly fair under congestion, but the effectiveness and fairness tradeoff is adjustable by varying the virtual RTT multiplier value.

VIII. Conclusion

In this paper, we demonstrated that our combined parallel TCP approach can effectively consume available network bandwidth on an uncongested network. We also showed that our approach is fairer to competing TCP streams than the unmodified parallel TCP method when the network is congested, and that the effectiveness and fairness tradeoff can be adjusted by changing the virtual RTT multiplier.

We showed that our method exploits a feature of the TCP congestion avoidance algorithm in which short RTT streams dominate long RTT streams.

The fundamental characteristics of network technology have changed since the congestion avoidance algorithm was designed in 1988. The goals of fairly sharing bandwidth and efficiently using network resources have not changed. We believe that new approaches to congestion avoidance must consider *fairness* as well as effectiveness to preserve shared public internetworks.

ACKNOWLEDGMENT

The authors would like to thank the National Institutes of Health Visible Human Project (Grant N01-LM-0-3511) for their continued encouragement and support. The work described in this paper utilized resources of the National Partnership for Advanced Computational Infrastructure (NPACI) and the Michigan Center for Biological Information. Finally, many thanks to Matt Mathis for his encouragement and mentioning.

REFERENCES

[1] (2003) Atlas high energy physics project. [Online]. Available: http://atlasexperiment.org

[2] A. E. Johnson, J. Leigh, and T. DeFanti, "Multidisciplinary experiences with CAVERNsoft tele-immersive applications," in *Proc. of Fourth International Conference on Virtual System and Multimedia*, 1998, pp. 498–503.

[3] K. S. Park, Y. J. Cho, N. K. Krishnaprasad, C. Scharver, M. J. Lewis, J. Leigh, and A. E. Johnson, "CAVERNsoft G2: A toolkit for high performance tele-immersive collaboration," in *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST-00)*, K. Y. Wohn, Ed. N. Y.: ACM Press, Oct. 22–25 2000, pp. 8–15.

[4] L. Smarr. (2002, Sept.) The 'OptIPuter': California, Illinois Researchers Fashion New Paradigm for Data-Intensive Computing and Collaboration Over Optical Networks. [Online]. Available: http://www.calit2.net/news/2002/9-25-optiputer.html

[5] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data management and transfer in high-performance computational grid environments," *Parallel Computing*, vol. 28, no. 5, pp. 749–771, May 2002.

[6] A. Hanushevsky. (2003) bbcp copy. [Online]. Available: http://www.slac.stanford.edu/ abh/bbcp/

[7] B. Tierney, J. Lee, T. Chen, H. Herzog, G. Hoo, G. Jin, and B. Johnston, "Distributed parallel data storage systems: A scalable approach to high speed image servers," in *Proceedings of the Second ACM International Conference on Multimedia (MULTIMEDIA '94)*. New York: ACM Press, Oct. 1994, pp. 399–406.

[8] H. Sivakumar, S. Bailey, and R. L. Grossman, "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," in *SC2000: High Performance Networking and Computing. Dallas Convention Center, Dallas, TX, USA, November 4–10, 2000*, ACM, Ed. ACM Press and IEEE Computer Society Press, 2000, pp. 63–64.

[9] V. Jacobson and M. J. Karels, "Congestion Avoidance and Control," *Proceedings, SIGCOMM '88 Workshop*, pp. 314–329, 1988.

[10] C. Hornig, "A standard for the transmission of IP datagrams over ethernet networks," RFC 894, 1984.

[11] R. Hughes-Jones, S. Dallison, and G. Fairey, "Performance measurements on gigabit ethernet nics and server quality motherboards," Feb. 2003. [Online]. Available: http://datatag.web.cern.ch/datatag/pfldnet2003/program.html

[12] S. Ubik and P. Cimbal, "Debugging end-to-end performance in commodity operating systems," Feb. 2003. [Online]. Available: http://datatag.web.cern.ch/datatag/pfldnet2003/program.html

[13] J. Stone and C. Partridge, "When the CRC and TCP checksum disagree," *Proc. ACM SIGCOMM, Stockholm, Sweden*, pp. 309–319, Sept. 2000.

[14] J. Padhye and S. Floyd, "On inferring TCP behavior," *ACM SIGCOMM*, pp. 287–298, 2001.

[15] P. Gray and A. Betz, "Performance evaluation of copper-based gigabit ethernet interfaces," in *27th Annual IEEE Conference on Local Computer Networks (LCN 2002), 6-8 November 2002, Tampa, FL, USA, Proceedings*. IEEE Computer Society, 2002.

[16] S. Floyd. (2003, Feb.) High Speed TCP for Large Congestion Windows. Internet draft. [Online]. Available: http://www.icir.org/floyd/papers/draft-floyd-tcp-highspeed-02.txt

[17] J. C. R. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789–798, Dec. 1999.

[18] U. Hengartner, S. Moon, R. Mortier, and C. Diot, "Detection and analysis of routing loops in packet traces," Sprint ATL, Tech. Rep. TR02-ATL051001, May 2002.

[19] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *ACM SIGCOMM*, Aug. 2002.

[20] T. Kelly, "Scalable tcp: Improving performance in highspeed wide area networks," in *Proceedings of the First International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet)*, Feb. 2003. [Online]. Available: http://www-lce.eng.cam.ac.uk/ ctk21/scalable

[21] M. Allman, H. Kruse, and S. Ostermann, "An application-level solution to tcp's satellite inefficiencies," 1997. [Online]. Available: citeseer.nj.nec.com/allman97applicationlevel.html

[22] T. Hacker, B. Athey, and B. Noble, "The end-to-end performance effects of parallel tcp sockets on a lossy wide-area network," in *16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium, Ft. Lauderdale, FL.* IEEE-CS/ACM, Apr. 2002.

[23] T. J. Hacker, B. D. Noble, and B. D. Athey, "The effects of systemic packet loss on aggregate TCP flows," in *Proceedings of Supercomputing 2002*. Baltimore, MD: IEEE/ACM SIGARCH, Nov. 2002, pap270.

[24] T. H. Henderson, E. Sahouria, S. McCanne, and R. H. Katz, "On improving the fairness of TCP congestion avoidance," *IEEE Globecomm conference, Sydney*, pp. 539–544, 1998.

[25] D. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, p. 14, 1989.

[26] C. Lin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, and S. Singh, "FAST TCP: from theory to experiments," *Submitted for publication in IEEE Communications Magazine*, 2003.

[27] S. Biaz and N. H. Vaidya, "Is the round-trip time correlated with the number of packets in flight ?" Texas A&M University, Technical Report TR99-006, 1999.

[28] (2002) Indiana University Advanced Network Management Lab Tsunani Project. [Online]. Available: http://www.uanml.iu.edu/anmlresearch.html

[29] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for internet hosts," in *SIGCOMM*, 1999, pp. 175–187.

[30] D. Bansal and H. Balakrishnan, "Binomial congestion control algorithms," in *IEEE Infocom '01: The Conference on Computer Communications, Volume 2: 20th Annual Joint Conference of the IEEE Computer and Communications Societies (IC '01)*. Washington - Brussels - Tokyo: IEEE, Apr. 2001, pp. 631–640.

[31] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[32] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, pp. 336–350, June 1997.

[33] M. Mathis, J. Semke, and J. Mahdavi, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Communications Review*, vol. 27, no. 3, July 1997.

[34] L. Cottrell and W. Mathews. (2003) The Internet End-to-end Performance Monitoring Project, Stanford University. [Online]. Available: http://www-iepm.slac.stanford.edu

[35] Roland Schemers III. (2003, Apr.) 'fping'. [Online]. Available: http://www.fping.com

[36] A. Tirumala. (2001) End-to-end bandwidth measurement using iperf. National Laboratory for Applied Network Research (NLANR). [Online]. Available: http://dast.nlanr.net/Projects/Iperf/

[37] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 303–314, 1998.

[38] Y. R. Yang, M. Kim, and S. Lam, "Transient behaviors of TCP-friendly congestion control protocols," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM-01)*. Los Alamitos, CA: IEEE Computer Society, Apr. 22–26 2001, pp. 1716–1725.