A Study on High Speed TCP Protocols

Lori A. Dalton and Ciji Isen
Department of Electrical Engineering and Department of Computer Science
Texas A&M University, College Station, Texas 77843–3128
ldalton@ee.tamu.edu, cisen@tamu.edu

***Abstract*— Standard TCP is successful at low speeds but it is unfit for high speed communication due to its slow ramp-up of congestion window size. Many modern fields of study require high bandwidth connectivity. Various solutions to this, including server side modifications of TCP like HighSpeed TCP and Fast TCP, are being advocated. In this work we study the effectiveness of these two protocols by comparing and contrasting both of them with standard TCP.**

I. Introduction

In recent times, new and exciting fields of study, such as High Energy physics, nuclear physics, and Bioinformatics, have emerged and call for high speed communication network links. These applications require communications between groups that are geographically spread and involve sharing of large volumes of data. For example, the SLAC (Stanford Liner Accelerator Center) and Fermilab organizations have data accumulated to the scale of petabyes [1]. It is estimated that for effective sharing of data, a throughput of 10Gbps would have to be sustained on the main links of such systems [1].

Having said that, consider why one needs variations for TCP in the first place. Network congestion, or times when the available bandwidth in a link is below that expected by users, is a common occurrence in networks. Congestion is a phenomenon that must be carefully considered in all network and protocol designs. TCP congestion control [2] is the basic method used in present TCP to combat congestion. For flow control, TCP relies on end-to-end acknowledgements and maintains a congestion window, which is an upper bound on the number of segments that can be sent while still waiting for an acknowledgment. TCP uses duplicate ACKs and ACK time out to infer packet loss. In simple terms, for each acknowledgment received, Reno TCP (which we call standard TCP) modifies the congestion window size as follows [2], [3]:

$$w \leftarrow w + \frac{a}{w}, \tag{1}$$

where the unit of w is in packets and a is the increase parameter. Upon detecting congestion in a RTT,

$$w \leftarrow w - bw, \tag{2}$$

where b is the decrease parameter. For a slow start ACK,

$$w \leftarrow w + c, \tag{3}$$

where c is the slow start parameter. In standard TCP, $a = 1$, $b = 0.5$, and $c = 1$ by default.

While this algorithm in standard TCP is successful at low speeds, it is unfit for high speed communication. Consider the following popular example. Suppose there is a connection with a RTT of 200ms and a packet size of 1500 bytes. For a link with bandwidth 1Gbps, one would have a congestion window of 17000 packets. When congestion is detected, w reduces to 8500, for a rate of only 500Mbps. For standard TCP, it will take 8500 RTTs to ramp back up to 1Gbps from 500Mbps. That translates to 28 minutes, which is totally unacceptable in present day networks since the duration of typical transfers are often much smaller. Therefore, TCP is not efficient at high speeds. The key issue in this problem is that congestion is a temporary phenomenon and that a faster recovery algorithm is needed in high speed links. We seek the fastest fair method to reach the optimal transfer rate which maximizes utilization.

II. Literature Review

The problem of high speed communication with TCP has been approached in a number of ways. The methods that we have looked at try to be compatible with standard TCP for a peaceful coexistence, and require changes only at the server end for effective use. There are several works on such modifications of TCP for high speed, including [4], HighSpeed TCP [5], [6], Fast TCP [1], [7], Westwood TCP [8], and Scalable TCP (which builds on HighSpeed TCP) [3]. Some of these protocols make use of TCP Vegas [9], [10] and variants [11] to use buffer delay as a sign of congestion, as opposed to drop inference. Reference [4] emulates an aggregate of N virtual channels opened, but deciding on the value of N itself can be tricky, making scalability questionable.

A. HighSpeed TCP

In this report, we focus on HighSpeed TCP and Fast TCP. HighSpeed TCP is a variation of standard TCP and thus very compatible. Like standard TCP, it uses packet drop inference to detect congestion (ACK-clocking). Standard TCP has a sending rate of [12],

$$T = \sqrt{\frac{a(2-b)}{2b}}\frac{1}{\sqrt{p}} \text{ packets per RTT,} \tag{4}$$

where p is the packet loss rate. To ensure compatibility with standard TCP, HighSpeed TCP uses standard TCP's default congestion window response parameters in (1)–(3) when the drop rate, p, is above a fixed threshold value P. Typically, P is set to 0.0015, corresponding to a threshold current window size of W = 31 [5]. Because HighSpeed TCP is the same as standard TCP for small congestion window sizes, it does not affect network behavior in heavy congestion environments.

However, when $p < P$ (or equivalently, $w > W$), HighSpeed TCP allows for variability in the parameters a, b and c [2]. In this approach, the values of a and b are now viewed as functions of the current window size, w. For a fixed congestion window size W_1 and target high speed drop rate P_1, we choose $a(w)$ and $b(w)$ to satisfy [5],

$$W_1 = \sqrt{\frac{a(w)(2-b(w))}{2b(w)}}\frac{1}{\sqrt{P_1}}. \qquad (5)$$

For example, the above relation is satisfied by a = 72 and b = 0.1 for W_1 = 83000 and P_1 = 1e-7, i.e., a congestion window decrease of 10% for a drop and an increase of 0.1% for a transmission without drop [5].

B. Fast TCP

The Fast TCP protocol is based on Vegas TCP instead of Reno TCP. Thus, it differs fundamentally from HighSpeed TCP in that it primarily makes use of queuing delay, rather than packet loss, to estimate congestion. This may be advantageous in high speed networks where queuing delay can be measured more reliably. Furthermore, by monitoring both queuing delay and packet loss, Fast TCP has the potential to give a more complete picture of congestion in the network.

Fast TCP uses an equation based algorithm that paces the sender, rather than alternating between “testing” the network with more packets and backing-off dramatically in the event of a packet loss, thereby reducing oscillatory behavior [1]. This effect appears in simulations of Fast TCP for large buffers. Furthermore, it is claimed that Fast TCP can maintain stability and equilibrium by only modifying the sending hosts [1], and that it is highly scalable with link capacity [1], [13].

One draw back of Fast TCP is that detailed information is not as accessible as in the other protocols, primarily because the work is commercial in nature. In our understanding, Fast TCP is less friendly to standard TCP than HighSpeed TCP.

III. Implementation

In this work we study HighSpeed TCP and Fast TCP and try to analyze their effectiveness. To compare and contrast these protocols, we simulated them over various network parameters using ns-2.26 [1] on cygwin 1.5.5 [2].

We used the HighSpeed TCP implementation build in the latest version of ns2. Since an ns2 implementation of Fast TCP has not been released, we used a Fast TCP code written by Xiaoqiao Meng of UCLA (not available in public domain). The code adheres to the published Fast TCP standards [1].

The standard TCP, HighSpeed TCP, and Fast TCP protocols were simulated under various network conditions in the “dumbbell” network topology illustrated in Fig. 1. To compare each of these protocols, we vary the number of nodes, bottleneck bandwidth, error probability in the channel, queue buffer size, and queue type (DropTail or RED). Our default values for these parameters are 1 flow, 100 Mbps, 0 error probability, 50 packet buffer size, and DropTail queuing, respectively. We also tested these protocols with reverse traffic.

[1]Available at http://www.sims.berkeley.edu/ christin/ns-cygwin.shtml.
[2]Available at http://www.cygwin.com/.

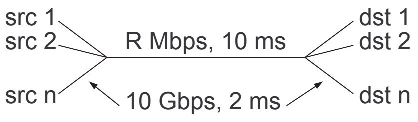

Fig. 1. Topology of all simulations. Default is R = 100 Mbps.

TABLE I
PERFORMANCE UNDER VARIOUS TOPOLOGY CONFIGURATIONS.

Flows	Protocol	Average BW (Mbps)	Average cwnd
1	Standard TCP	63.8687	227.944
1	HighSpeed TCP	84.3331	305.051
1	Fast TCP	73.4996	264.939
2	Standard TCP	76.1065	137.59
2	HighSpeed TCP	82.9526	156.471
2	Fast TCP	75.421	140.412
4	Standard TCP	74.5453	69.0467
4	HighSpeed TCP	81.182	79.9547
4	Fast TCP	75.4806	70.8203

IV. Simulation Results

A. Effect of Multiple Nodes

Simulation results for the standard, HighSpeed, and Fast TCP protocols are shown in Fig. 2 and 3 for n = 1 and 4 flows, respectively. A summary of the average bandwidth and congestion window size is presented in Table I. Fig. 2 is considered the standard network condition and will be used for comparison in later sections.

In all cases, HighSpeed TCP performs the best in terms of average achieved bandwidth. In the default case of Fig. 2, Fast TCP performs better than standard TCP. However, when multiple nodes are added, Fast TCP and standard TCP achieve similar performance. This observation was also reported in [4]. Thus, the advantage of Fast TCP over HighSpeed TCP is questionable based on our observation in these simulations. We believe that the aggressive windowing technique used in Fast TCP may be too excessive and negatively affecting Fast TCP’s performance.

Notice that in some cases, as the number of flows increases the total bandwidth utilized in the channel also increases. This is because we essentially divide the 100 Mbps channel into several smaller sub-channels, and due to this smaller effective bandwidth, each sub-channel can capture a higher percentage of its total allocated resources.

Finally, note as the number of nodes increases, the congestion window size of the standard and HighSpeed TCP protocols appears to decay slowly to an equilibrium, whereas Fast TCP behaves very erratically for the first 4 seconds and afterwards converges rapidly to equilibrium.

B. Effect of Bottleneck bandwidth Size

Results for the standard, HighSpeed, and Fast TCP protocols are shown in Fig. 4 with bottleneck bandwidth R = 1000 Mbps. For comparison, refer to Fig. 2 for R = 100 Mbps.

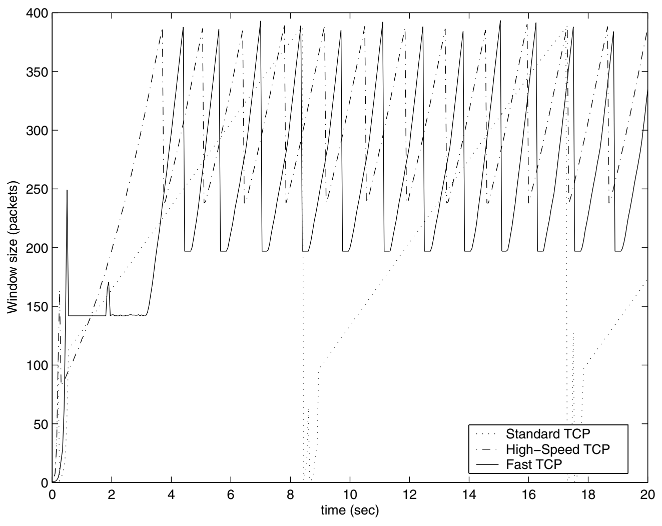

Fig. 2. Flow 0 congestion window size for n = 1 flows.

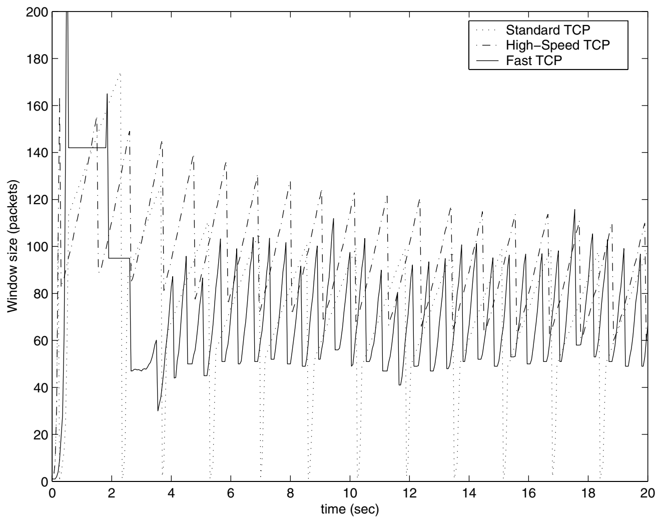

Fig. 3. Flow 0 congestion window size for n = 4 flows.

A summary of the average bandwidth and congestion window size for all relevant cases is presented in Table II.

At low bandwidths, all protocols nearly achieve the capacity of the channel. For extremely high bandwidths, HighSpeed TCP is vastly superior to the other protocols, and curiously, standard TCP performs significantly better than Fast TCP.

C. Effect of Channel Errors

Simulation results for the standard, HighSpeed, and Fast TCP protocols are shown in Fig. 5 with channel error probability p = 1e-4. Refer to Fig. 2 with channel error probability p = 0 for comparison. A summary of the average bandwidth and congestion window size is presented in Table III.

For moderately high error probabilities (p = 1e-4), which may be used to model congestion, performance is nearly unchanged for HighSpeed and Fast TCP, but begins to degrade significantly for standard TCP. In extremely noisy conditions

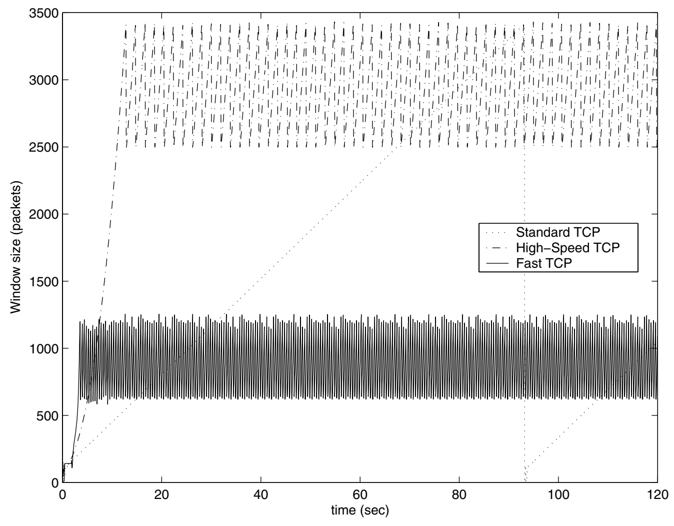

Fig. 4. Congestion window size for R = 1000 Mbps.

TABLE II

PERFORMANCE UNDER VARIOUS BOTTLENECK BANDWIDTHS.

Rate (Mbps)	Protocol	Average BW (Mbps)	Average cwnd
1	Standard TCP	0.962132	39.5112
1	HighSpeed TCP	0.967863	41.5263
1	Fast TCP	0.964731	47.2552
100	Standard TCP	63.8687	227.944
100	HighSpeed TCP	84.3331	305.051
100	Fast TCP	73.4996	264.939
1000	Standard TCP	425.292	1489
1000	HighSpeed TCP	783.266	2748.68
1000	Fast TCP	250.648	858.913

(p = 1e-2), all protocols suffer dramatically, with Fast TCP performing the best, followed by HighSpeed TCP and then standard TCP. HighSpeed TCP exhibited extremely erratic behavior here, and at certain time instants momentarily set the congestion window size to an extremely high level.

D. Effect of Queue Buffer Size with DropTail

Results for the standard, HighSpeed, and Fast TCP protocols with DropTail queuing are shown in Fig. 6 with buffer size B = 500 packets. Refer to Fig. 2 with buffer sizes B = 50 for comparison. A summary of the average bandwidth and congestion window size is presented in Table IV.

In all cases regarding queue buffer size, HighSpeed TCP

TABLE III

PERFORMANCE UNDER VARIOUS CHANNEL ERROR PROBABILITIES.

p	Protocol	Average BW (Mbps)	Average cwnd
0	Standard TCP	63.8687	227.944
0	HighSpeed TCP	84.3331	305.051
0	Fast TCP	73.4996	264.939
1e-4	Standard TCP	32.8905	115.032
1e-4	HighSpeed TCP	84.2535	304.665
1e-4	Fast TCP	71.9742	259.745
1e-2	Standard TCP	2.58479	8.97549
1e-2	HighSpeed TCP	3.69619	-
1e-2	Fast TCP	5.6098	20.2149

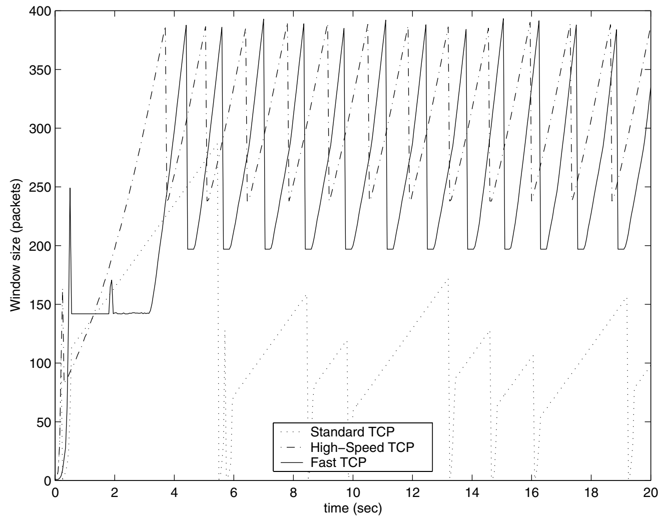

Fig. 5. Congestion window size for p = 1e-4.

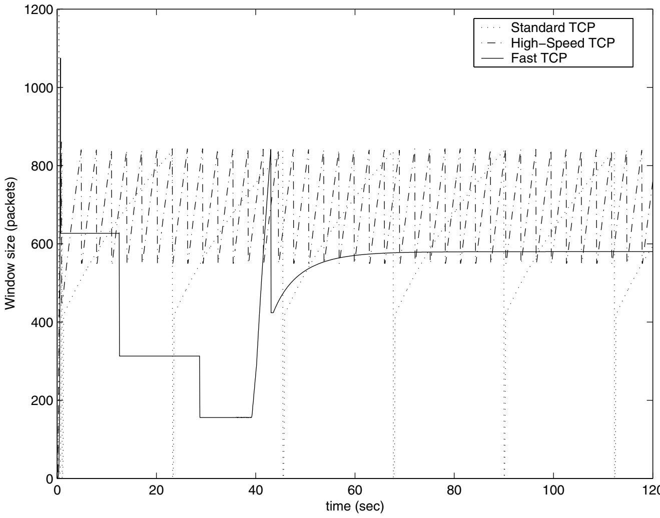

Fig. 6. Congestion window size with DropTail for B = 500.

performs the best. For small buffer sizes (below 50), Fast TCP performs better than standard TCP. However, Fast TCP is the worst protocol for larger buffer sizes (above 100).

E. Effect of Queue Buffer Size with RED

Simulation results for the standard, HighSpeed, and Fast TCP protocols with RED queuing are shown in Fig. 7 with buffer size B = 500. A summary of the average bandwidth and congestion window size is presented in Table IV.

As with DropTail queuing, HighSpeed TCP is superior in all cases with RED queuing, Fast TCP is worst for high buffer sizes (above 100), and standard TCP is the worst for low buffer sizes (below 50). DropTail and RED queuing appear to have nearly identical performance for small window sizes. However, for the largest buffer size of 500, the HighSpeed and Fast TCP protocols behave very differently for RED queueing with an abrupt change around 40 seconds into the simulation.

TABLE IV
PERFORMANCE UNDER VARIOUS QUEUE BUFFER SIZES WITH DROPTAIL AND RED QUEUING.

B	Protocol	Average BW (Mbps)		Average cwnd (pac.)	
		DropTail	RED	DropTail	RED
10	Standard	43.0001	50.336	152.279	176.965
10	HighSpeed	76.0341	76.0968	268.394	268.785
10	Fast	67.8116	67.6661	240.32	240.147
50	Standard	63.8687	64.0865	227.944	229.075
50	HighSpeed	84.3331	85.2966	305.051	308.919
50	Fast	73.4996	73.8282	264.939	267.08
100	Standard	84.3251	84.4709	319.54	320.818
100	HighSpeed	91.3422	91.4583	351.197	352.326
100	Fast	84.3043	81.586	320.565	307.142
500	Standard	94.7585	85.6263	630.356	331.991
500	HighSpeed	95.9727	71.5531	690.53	377.017
500	Fast	69.4632	65.5129	498.665	370.366

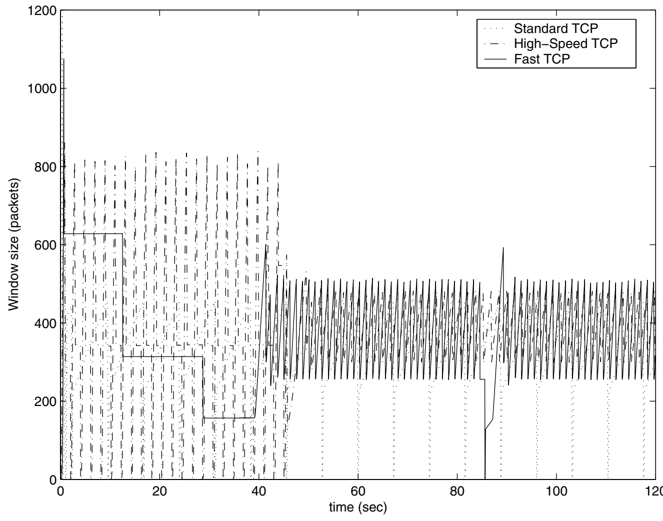

Fig. 7. Congestion window size with RED for B = 500.

F. Effect of Reverse Traffic

To test the robustness of these protocols in the presence of reverse traffic, we used the same topology and reversed the flow direction for half of the links. A summary of the average bandwidth and congestion window size for the standard, HighSpeed, and Fast TCP protocols with reverse traffic is presented in Table V.

In the congestion window plots, we saw that both standard TCP and HighSpeed TCP appear stable, with flows in both directions behaving almost identically. However, Fast TCP, shown in Fig. 8, had a slightly diminished overall bandwidth and seemed to oscillate between favoring flows in the forward and reverse directions, especially when the number of flows was low. Because reverse traffic causes ACK packets to be delayed and returned in bursts, resulting in ACK compression, protocols such as TCP Vegas and Fast TCP that use RTT to measure the state of the network receive a misleading picture of congestion and tend to perform poorly. Fast TCP was also reported to have problems with reverse traffic in [14].

TABLE V
PERFORMANCE UNDER VARIOUS REVERSE FLOW CONFIGURATIONS.

Flows	Reverse	Protocol	Avg. BW (Mbps)	Avg. cwnd
2	1	Standard TCP	85.6852	156.0924
2	1	HighSpeed TCP	93.4708	171.5883
2	1	Fast TCP	83.7556	153.7652
4	2	Standard TCP	83.9058	77.4082
4	2	HighSpeed TCP	91.9685	84.6614
4	2	Fast TCP	84.2136	77.8559

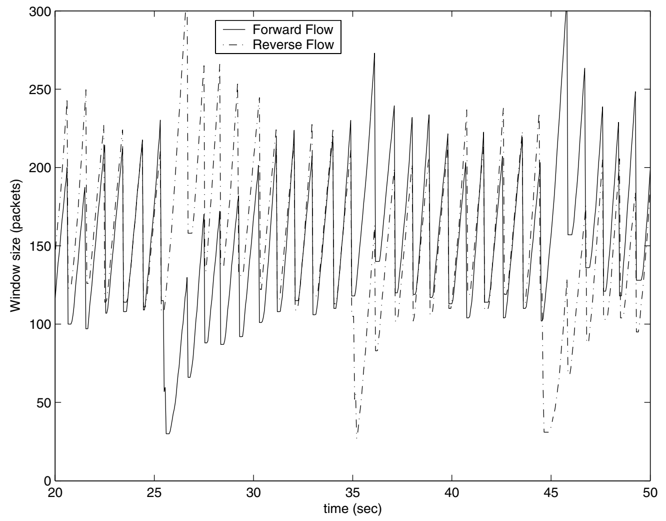

Fig. 8. Congestion window size with 1 forward and 1 reverse flow.

V. CONCLUSION

On the whole, our simulation results seem to match with [14] but differs with [7]. To summarize our results:

- Fast TCP behaves the most erratically.
- In the default case: HighSpeed $>>>$ Fast $>$ standard.
- With multiple nodes: HighSpeed $>>>$ Fast $=$ standard.
- At low BW: HighSpeed $=$ Fast $=$ standard $=$ capacity.
- At high BW: HighSpeed $>>>$ standard $>$ Fast.
- For middle error prob.: HighSpeed $=$ Fast $>$ standard.
- For high error prob.: Fast $>$ HighSpeed $=$ standard.
- For small buffer sizes: HighSpeed $>$ Fast $>$ standard.
- For large buffer sizes: HighSpeed $>$ standard $>$ Fast.
- RED and DropTail perform the same.
- Fast is unstable in the presence of reverse links.

Both new protocols require only changes at the server side. Thus they can be implemented and the gains observed irrespective of what protocols the linking routers are running. HighSpeed TCP only requires a simple change to the congestion window update logic to improve the throughput in high speed WAN.

Except in the case of extremely high channel error probabilities, HighSpeed TCP is the best protocol. In the default network scenario, HighSpeed TCP performs 32% better than standard TCP, achieving 84.3% of the capacity of the link. Fast TCP is best in high noise, and consistently performs better than standard TCP, except in the cases of extremely high BW and large buffer sizes. Since Fast TCP seems to be the most robust at high error rates, it might be worth exploring how it would fare in a wireless setup.

VI. FUTURE WORK

As an extension of this work, we propose to study other promising high-speed protocols such as Scalable TCP [3] and Westwood TCP [8]. Fairness, compatibility, scalability, and stability will also be considered for a more complete comparison. We also hope to simulate more complex networks, subject to dynamic flows and network changes, to better understand the behavior of these protocols under more realistic conditions.

VII. ACKNOWLEDGEMENTS

We thank Xiaoqiao Meng of UCLA, for sharing his ns2 implementation of Fast TCP and for his valuable comments on Fast TCP.

We also thank Narasimha Reddy of Texas A&M, Evandro de Souza of Lawrence Berkley Lab, Tom Kelly of Cambridge, Sally Floyd of ICIR, and Steven Low of Caltech for their helpful discussions.

REFERENCES

[1] C. Jin, 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 to *IEEE Commun. Magazine*, April 2003.

[2] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM Special Interest Group on Data Communications Conf. (SIGCOMM '88)*, Stanford, CA, Aug. 1988, pp. 314–329.

[3] T. Kelly, "Scalable TCP: improving performance in high speed wide area networks," in *Proc. 1st Intern. Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2003)*, Geneva, Switzerland, Feb. 2003.

[4] P. Gevros, F. Risso, and P. Kirstein, "Analysis of a method for differential TCP service," in *Proc. IEEE Global Commun. Conf. (GLOBECOM '99)*, Rio de Janeiro, Brazil, Dec. 1999, vol. 3, pp. 1699–1708.

[5] S. Floyd, S. Ratnasamy, S. Shenker, "Modifying TCP's congestion control for high speeds," Internet draft, May 2002. [Online] Available: http://www.icir.org/floyd/hstcp.html.

[6] S. Floyd, "High speed TCP for large congestion windows," Internet Engineering Task Force (IETF) Internet draft, Dec. 2003. [Online] Available: http://www.icir.org/floyd/hstcp.html.

[7] C. Jin, D. Wei, and S. H. Low, "Fast TCP: Motivation, architecture, algorithms, performance," in *Proc. IEEE Commun. Soc. Conf. (Infocom 2004)*, Honk Kong, March 2004.

[8] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: bandwidth estimation for enhanced transport over wireless links," in *Proc. 7th Intern. Conf. Mobile Computing and Networking (MOBICOM 2001)*, Rome, Italy, July 2001, pp. 287–297.

[9] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global internet," *IEEE J. on Select Areas in Commun.*, vol. 13, pp. 1465–1480, Oct. 1995.

[10] E. Weigle and W. Feng, "A case for TCP Vegas in high-performance computational grids," in *Proc. 10th IEEE Intern. Symposium on High Performance Distributed Computing (HPDC' 01)*, San Francisco, CA, Aug. 2001, pp. 158–167.

[11] H. Choe and S. H. Low, "Stabilized Vegas," in *Proc. 22nd Annual Joint Conf. of the IEEE Computer and Commun. Societies (Infocom 2003)*, San Francisco, CA, April 2003, vol. 3, pp. 2290–2300.

[12] S. Floyd, M. Handley, and J. Padhye, "A comparison of equation-based and AMID congestion control," Internet draft, May 2000. [Online] Available: http://www.aciri.org/tfrc/.

[13] F. Paganini, J. C. Doyle, and S. H. Low, "Scalable laws for stable network congestion control," in *Proc. IEEE Conf. on Decision and Control*, Orlando, FL, Dec. 2001, vol. 1, pp. 185–190.

[14] H. Bullot, R. Les Cottrell, and R. Hughes-Jones, "Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks," in *J. of Grid Computing*, vol. 1, pp. 345–359, Dec. 2003.