

# Enhancements for RTT-Fair HighSpeed TCP

Damien Phillips, Jiankun Hu

School of Computer Science and Information Technology, RMIT University

GPO Box 2476V, Melbourne 3001, Australia

Email: {damphil, jiankun}@cs.rmit.edu.au

**Abstract**— TCP is known to have bandwidth fairness problems for flows of different RTT. HighSpeed TCP, which adapts TCP’s congestion avoidance to grow faster, and thus be able to scale to high bandwidth environments, aggravates the RTT fairness problem since the congestion window increases at a rate proportional to the RTT and the current window size.

We propose a method to eliminate the RTT fairness problem for competing high bandwidth flows. We decouple the rate of window growth in HighSpeed TCP from the current window size, such that competing flows adjust their window size, instead, according to their perceived byte sending rate. We derive conditions and a normalisation scheme to ensure that high bandwidth competing flows converge to fair bandwidth allocation regardless of the RTT or segment size used, while maintaining HighSpeed TCP’s ability to scale to very high bandwidth environments. We introduce features to preserve TCP friendliness in low bandwidth environments such that, the RTT fairness is no worse than that of standard TCP in scenarios of moderate packet loss. We have conducted simulations to validate the abilities of our proposal to eliminate the RTT bias from HighSpeed TCP.

## I. INTRODUCTION

Using TCP on high bandwidth paths with significant round trip times (RTT) has highlighted the intrinsic problems that TCP faces when required to scale to very large congestion window sizes, while maintaining acceptable throughput and utilisation. These paths can arise in the field of Grid Computing whereby computation of often large scale data sets are assigned to dedicated high performance machines in other parts of the country or internationally. TCP’s congestion avoidance algorithm adds one segment to the congestion window every RTT, which requires too long a time to reach window sizes required for high bandwidth paths [1]. This can, in turn, discourage Grid Computing users to utilise non-local resources as data transfer begins to add too much to the application’s completion time. In response to this, several new protocols have been proposed. One method adopted by Scalable TCP (STCP) [2] and HighSpeed TCP (HSTCP) [1] is to change the congestion avoidance algorithm to base its rate of increase on the current window size. In this way, as the congestion window grows, its rate of growth accelerates to achieve large congestion windows in a shorter period of time compared to standard TCP. However, TCP poses a bandwidth fairness problem for competing flows of different RTT, where lower RTT flows grow their window faster and gain more bandwidth[3][4]. STCP and HSTCP not only inherit this RTT unfairness, but amplify the problem since the rate of window growth is proportional to both RTT and the current congestion window [5]. This also poses a problem to Grid Computing

users, such that if equal sized data sets are being shipped out to several destinations, the intrinsic behaviour of HSTCP will lead to a bias towards local resources. This bias is unnecessary if the connections traverse paths of similar bandwidth.

This paper introduces our contributions of extensions to HighSpeed TCP to allow flows of disproportionate RTT to compete fairly for bandwidth resources. Conditions are derived to ensure convergence to fairness, and a normalisation scheme is proposed to eliminate the RTT bias from HighSpeed TCP connections once each flow achieves a high sending rate. The problems of a RTT fairness proposal for standard TCP (Constant Rate TCP [6][7]) are analysed, and it is shown that for our targeted region of fairness, the discovered problems are overcome or do not apply. This leads to identical operation to TCP for low sending rates and fair throughput gains for high sending rates, which are not biased towards low RTT.

The simulations conducted are extended from the topology and bandwidth/delay scenarios of the HighSpeed TCP test suite in NS-2. These scenarios are used with competing connections of a range of RTT to show that our proposal does indeed remove the bias of RTT from HighSpeed TCP.

This paper is organised as follows, Section II introduces HighSpeed TCP and illustrates the problems of unfairness with flows of different RTT. In Section III we detail our Normalised-Rate HSTCP proposal. Following is an analysis of the proposed policy in Section IV and finally simulation validation and conclusion in Sections V and VI.

## II. HIGHSPEED TCP RTT FAIRNESS ISSUE FOR SYNCHRONISED LOSSES

HighSpeed TCP (HSTCP) has been proposed to allow TCP to make better use of high bandwidth-delay-product paths. This is achieved by modifying TCP’s response curve (the window size achievable given a particular connection loss probability) to cater for higher window sizes at realistic line loss rates.

HSTCP defines three parameters, *Low\_Window*, *High\_Window* and *High\_P*, while *Low\_P* is obtained from the intersecting point of HSTCP’s response curve with standard TCP’s curve at the point of *Low\_Window* = 38. Given HSTCP’s suggested values of *Low\_Window* = 38, *High\_Window* = 83000 and *High\_P* =  $10^{-7}$ , the following window increase and decrease functions are obtained [1]:

$$b(w) = (H_D - 0.5) \frac{\log(w) - \log(W)}{\log(W_1) - \log(W)} + 0.5 \quad (1)$$

$$a(w) = w^2 * p(w) * 2 * b(w) / (2 - b(w)) \quad (2)$$

$$p(w) = 0.078 / w^{1.2} \quad (3)$$

where  $H\_D = High\_Decrease = b(High\_Window) = 0.1$ ,  $W = Low\_Window$  and  $W_1 = High\_Window$ .

To examine the RTT fairness of HSTCP under synchronised loss, we propose a simple method whereby we consider the length of time needed in congestion avoidance to recover back to the flow's window size after a loss. This translates into recovering  $w \cdot b(w)$ . In a scenario of synchronised loss, each competing connection suffers a loss at the same absolute time. Furthermore, in steady-state for synchronised losses, each connection takes the same length of time to reach their previous congestion window. Given an example time interval, we can determine whether a connection will be able to gain further bandwidth (increasing beyond their previous window before loss), or whether a connection will decrease their bandwidth share.

Figure 1(a) shows the time taken for several HSTCP (light grey in figure) flows of increasing RTT to recover after loss, given the sending rate at the time of loss<sup>1</sup>. Notice the two distinct slopes present for each curve in Figure 1(a). The sending rates at the points where the slopes change (between 2Mbps and 20Mbps) indicate the points where the HSTCP flow's window size exceeds the *Low\_Window* threshold. When the sending rate results in a congestion window  $w < Low\_Window$ , HSTCP is operating under standard TCP AIMD parameters. Notice also that when  $w > Low\_Window$  (sending rates greater than 20Mbps in Figure 1(a)), the lines do not converge, indicating that having a greater RTT will always equate to taking a longer time to recover to the previous congestion window size after a loss. This is the same when using either TCP or HSTCP.

We can use Figure 1(b) to infer what the bandwidth share will be if we know the period of loss. For example, if synchronised losses are known to occur every 4 seconds, for  $MSS=1500$ Bytes, then for connections of  $RTT=\{87.5\text{ms}, 100\text{ms}, 112.5\text{ms}, 125\text{ms}\}$  their bandwidth share would be approximately  $\{96\text{Mbps}, 43\text{Mbps}, 22\text{Mbps}, 13\text{Mbps}\}$  respectively. In this scenario, having a RTT a little over half that of a competing connection ( $87.5/125 = 0.7$ ) can result in having more than four times smaller peak sending rate ( $96/13 = 7.38$ ). By using Active Queue Management techniques such as RED [8], we can reduce the likelihood of synchronised loss. However, these techniques cannot fully remove this bias, which worsens as sending rates increase.

### III. PROPOSED NORMALISED-RATE HSTCP

The formulation of *High\_Window* in the HSTCP window is based on a 10Gbps flow, at a RTT of 100ms, with a MSS of 1500Bytes, resulting in  $High\_Window = 83000$ . However, since the increase and decrease parameters are based on window alone ( $a(w), b(w)$ ), flows with different RTT or different MSS can have increase rates (change in sending rate

<sup>1</sup>A flow has to recover  $w \cdot b(w)$ , where  $w$  is calculated from the sending rate and RTT.

per second) orders of magnitude in difference. This is the core of the RTT fairness problem.

#### A. Conditions for Convergence to RTT Fairness

To ensure convergence to fairness among flows of different RTT, we should engineer mechanisms such that flows that have less than their fair share can increase their share, while other flows with more than their fair share decrease their share. Referring back to the synchronised loss case discussed in Section II, if flows with less than a fair share can recover more than  $w \cdot b(w)$  before the next loss, while flows with more than a fair share cannot recover  $w \cdot b(w)$  before the next loss, then the flows can begin to converge to RTT fairness. Specifically, in a case of synchronised loss among flows  $A$  and  $B$ , with the time to recover  $t_{wb(w)}$ , sending rates  $X_A$  and  $X_B$ , and RTTs of  $T_A$  and  $T_B$ , the following condition will ensure fairness;

$$t_{wb(w)}(A, T_A) < t_{wb(w)}(B, T_B) \iff X_A < X_B \quad (4)$$

Unfortunately, for standard TCP and HSTCP, this condition cannot be met since we can easily find a flow with smaller RTT that can recover a greater sending rate in a shorter time period.

One such solution to satisfy this condition, is to ensure that  $t_{wb(w)}$  monotonically increases as the sending rate increases, while keeping  $t_{wb(w)}$  equal for all connections with same sending rate regardless of RTT. One such mechanism that achieves this for standard TCP is the "Constant-Rate" window increase algorithm (CR) in [6]. In [7], the fairness improvements for isolated CR flows was clearly demonstrated. However, [7] also showed that CR would be difficult to successfully deploy in an operational network, as the performance of CR is hampered by Droptail queues, and that the proper selection of the required constant is difficult to determine.

We show that the CR goals can be achieved for HighSpeed TCP, such that once flows enter the HSTCP target sending rate range, flows of different RTT can begin to converge to fairness.

#### B. Normalised-Rate Scheme

This section proposes a scheme, Normalised-Rate HSTCP (NR-HSTCP), to equalise the increase rates of competing HSTCP connections. Under this scheme, flows with the same sending rate ( $w \times MSS/RTT$ ), but with different RTT or MSS, should have the same increase rate ( $a(w) \times MSS/RTT$ ). As the HSTCP response function was formulated under the assumption of a MSS of 1500Bytes with an RTT of 100ms, NR-HSTCP uses these values as the basis for the normal HSTCP response function. It is proposed that each connection normalise their congestion window increase parameters to this one standard HSTCP response function, and then calculate increase parameters to achieve a normalised increase. Taking the HSTCP specification as the normal response

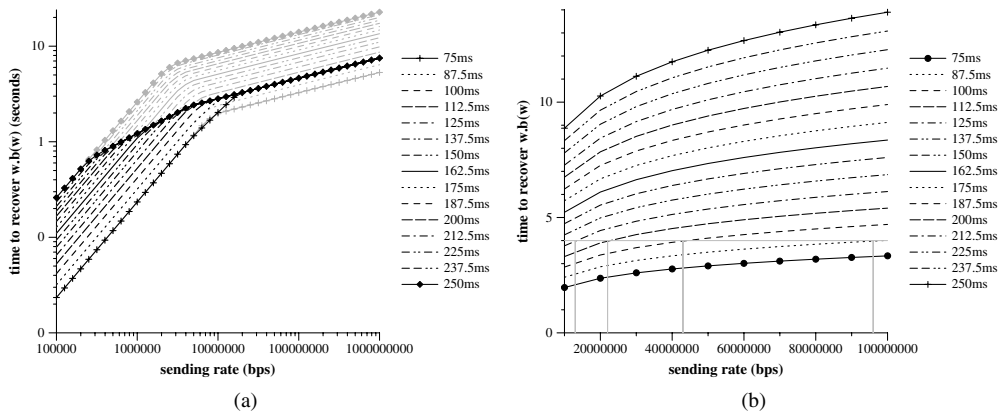


Fig. 1. (a) Time in seconds for connections of different RTT to recover back to their previous congestion window for HSTCP (light grey) and our proposed Normalised Rate HSTCP (black). (b) Zoomed in section of HSTCP curves with the drop lines showing the bandwidth share of connections where losses occur every 4 seconds

function, calculating parameters becomes;

$$normcwnd = \frac{cwnd \cdot MSS}{RTT} \frac{100ms}{1500Bytes} \quad (5)$$

$$decrease = b(normcwnd) \quad (6)$$

$$increase = \frac{a(normcwnd)1500Bytes}{MSS} \frac{RTT^2}{100ms^2} \quad (7)$$

Note that (7) requires square of the RTT ratio. If no square was used, this would simply serve to normalise the window sizes of competing connections, which will not achieve sending rate normalisation.

If all connections followed the above steps when calculating their increase and decrease parameters, then any number of connections with the same sending rate will increase at the same rate. More importantly, (4) will hold true, ensuring connections converge to fairness.

### C. Transition Between TCP and NR-HSTCP

HSTCP suggests that once connections reach a window size of  $Low\_Window = 38$ , the connection can begin to ramp up its increase rate to scale with larger capacity links. However, by using a constant congestion window threshold, those connections with lower RTT will reach this window first, and accelerate their growth, even though they have a higher sending rate at  $Low\_Window$ . Therefore, it is proposed that the switching point between TCP and NR-HSTCP be determined by the resulting normalised congestion window increase. It is proposed that NR-HSTCP takes the maximum of one segment per RTT and the normalised congestion window increase when increasing sending rate. After the initial calculation of parameters from (5), (7) and (6), they are limited as such;

$$b(w) = \min[\max[decrease, 0.1], 0.5] \quad (8)$$

$$a(w) = \max[\min[increase, w], 1] \quad (9)$$

Figure 1(a) illustrates the effect on the time taken to recover after a loss when NR-HSTCP (black in figure) algorithms are used. Note that higher RTT connections begin to scale

up first, while the lower RTT connections are still limited to one segment per RTT. As the sending rates increase, all connections with different RTT converge to the one sending rate increase curve. It is in this region that RTT fairness can be met.

## IV. DISCUSSION OF THE NORMALISED-RATE POLICY

To ensure that the NR-HSTCP proposal does not suffer the same problems as the CR proposal, we first need to provide in-depth analysis on the limitations of CR set forth in [7].

### A. NR-HSTCP improvements on Constant Rate TCP

In [7], some issues were raised against the possibility of deploying CR to combat the fairness issues of standard TCP. In this section we address these issues for use in our NR-HSTCP policy.

*Small congestion windows together with large RTT can lead to bursty send patterns and should be limited to counter the resulting increased loss.* For this issue, the Normalised-Rate algorithms will not result in an increase value greater than one segment per ACK once greater than a modest sending rate. For example, considering a RTT of 1 second, using MSS of 576Bytes,  $a(w)$  will be limited to  $w$  until a window size of 16 or approximately 10kB. If a MSS of 1500Bytes is used, the limiting is in effect below a window size of 5 or approximately 8kB. It is possible for this limiting to be in effect for much longer for a combination of small MSS and very large RTT. However, in this environment, we believe that ensuring friendliness with flows that could have RTTs orders of magnitude smaller, is not an achievable target, and as such, some other windowing scheme should be used instead. In this respect, simulations investigate scenarios with RTT in the range of 75ms to 250ms.

For higher sending rates at higher RTT, the window increase per RTT can get fairly large, though this increase is modest in comparison to the window size required to reach such high sending rates. However, should ACK compression occur, such as with congestion on the reverse path, there is potential for a higher than usual number of packets arriving together.

This needs to be further investigated to determine whether  $a(w)$  must be limited. Furthermore, large increases per RTT tend to cause large number of packets dropped per loss indication. Nevertheless, this problem can be alleviated if the path allows use of larger packet sizes, such as “Jumbograms” (9kB frames instead of 1500B frames). Our proposed protocol makes the increase rate independent of both the RTT and MSS. Therefore, the problems associated with increases of a large number of packets or many packets dropped in one window, translates into a fewer number of larger packets added per RTT and fewer number of large packets dropped per loss indication, such that recovery at the protocol level is made easier.

*It is difficult to select a proper constant for successful deployment of CR in a highly heterogeneous network.* For this issue, HSTCP itself has a response function based on such predetermined constants. In NR-HSTCP, we have simply extended the use of the resultant increase rate to apply to connections of different RTT.

*The fairness benefits of CR can be hampered when running alongside standard TCP.* For this final issue, NR-HSTCP is designed to increase no slower than standard TCP for all RTT. For lower sending rates, the increase and decrease parameters are identical to standard TCP giving identical performance.

### B. Benefits of NR-HSTCP

NR-HSTCP provides a mechanism for HighSpeed TCP flows to maintain RTT fairness, while maintaining friendliness with standard TCP for lower sending rates. In the fairness range of operation, regardless of the RTT or MSS, all connections gain bandwidth at the same rate. The normalisation process has the effect of decoupling window size from rate of bandwidth gain. This then allows connections to scale their bandwidth share based on sending rate rather than window. Such decoupling of window and bandwidth increase can accommodate migration to a larger MSS while still maintaining sending rates identical to existing standard MSS flows.

HSTCP designed its decrease rate such that with increased window size, a connection would not drop to half window size, as does standard TCP, so that it would instead take less time to recover between losses and maintain a high utilisation. With NR-HSTCP, it is possible for connections with high RTT to have a large window and lower their sending rate significantly more after each loss when compared to HSTCP. Conversely, for low RTT NR-HSTCP, connections lower their window size significantly less after each loss than an equivalent HSTCP connection with the same large window size. This effect has subtle benefits, since for high RTT, loss signals take longer to propagate back to the sender, such that it is prudent to drop the sending rate enough to alleviate any congestion. Similarly, low RTT connections react quickly to loss signals and can afford to maintain a more steady window size.

## V. SIMULATION RESULTS

Simulation was conducted with the NS-2 simulator<sup>2</sup>. Simulation scripts were adapted from the HighSpeed TCP test suite,

<sup>2</sup><http://www.isi.edu/nsnam/ns/>

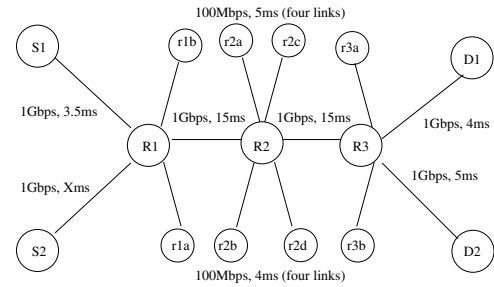


Fig. 2. Simulation topology. S1 connects to D1, S2 to D2, using HSTCP or NR-HSTCP. Xms is part of the variable RTT connection. Four standard TCP connections run as background traffic over the 100Mbps links.

with a second hop included in the main bottle neck link, and several more source-destination pairs added for background traffic. The topology is displayed in Figure 2. The link between S2 and R1 is set to variable delay to set up competing connections with different RTT. The links connecting R1, R2 and R3 use a 500 packet queue size with a DropTail queueing discipline. The background traffic consists of one standard TCP flow connecting between the pairs r1a-r2a, r1b-r2b, r2c-r3b, and r2d-r3a. All receiver maximum window sizes were set large enough for the window size to be limited by the network.

Each simulation was run for 1000 seconds, with the results being obtained from the final 600 seconds. This allows ample time for the competing connections to converge to a near steady state sending rate, while allowing minor fluctuations to average out over the extended time period. The background traffic flows were started at time 0, while both low and high RTT flows were started at time 50. A total of 120 separate simulations were initiated for each protocol, with a fixed lower RTT of 75ms and a random higher RTT between 75ms and 250ms.

### A. RTT Fairness of HSTCP

In the study of [5], the fairness of HighSpeed TCP was investigated under the conditions of varying degrees of synchronised loss, from no synchronisation to moderate synchronisation. It was found that under no synchronisation, HSTCP inherited TCP’s RTT unfairness, while for moderate synchronisation, the fairness problems were amplified. The simulation results shown in Figure 3(a), confirms the existing RTT unfairness of the current HSTCP protocol. Note how there is a clear bias towards the lower connection with RTT ratios in excess of 1.5. This region highlights how the lower RTT HSTCP connection, as well as the standard TCP flows, out-perform HSTCP flows with significantly larger RTT. This is in part due to the level of synchronised loss experienced by all flows competing through the DropTail router.

### B. RTT Fairness of NR-HSTCP

Figure 3(b) illustrates the simulation running on identical topology and bandwidth/delay settings as the previous scenario, however, our proposed NR-HSTCP mechanisms are employed. As can be seen, the obvious bias to the lower RTT

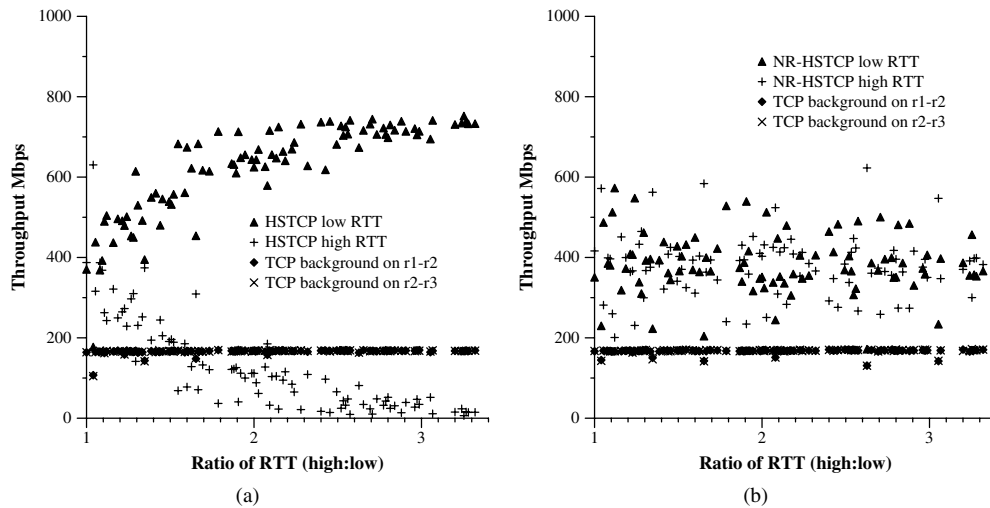


Fig. 3. Simulation results for HSTCP (a) and NR-HSTCP (b): 1Gbps bottle neck, one 75ms (NR-)HSTCP flow, one (NR-)HSTCP flow ranging from 75ms to 250ms, four standard TCP flows as background traffic, with background bandwidth grouped for each bottleneck link.

connection is removed. Also note that a consistent level of throughput is achieved, indicating that no penalty of lowered utilisation exists. The NR-HSTCP proposal exhibits the same throughput levels regardless of the connections' RTT, given that the sending rates lie in the region where fairness can be achieved (see Figure 1(a)).

## VI. CONCLUSION

Section III presents enhancements to HighSpeed TCP to remove the bias towards low RTT flows for connections that reach high bandwidth. Through decoupling the increase rate, not only is the RTT bias from HSTCP removed, but it is ensured that connections with disproportionate RTT and MSS, operating with a sending rate in our region of fairness, will converge to fair bandwidth share.

NR-HSTCP incorporates a mechanism to allow the protocol to operate identically to standard TCP for small sending rates, while converging to RTT fair operation once passing a moderate sending rate. Larger packet sizes can be used for cases of Long Fat Networks. Larger packets will help alleviate large congestion window increases per RTT and simplify recovery at times of loss, while still maintaining the RTT fairness offered by NR-HSTCP.

Under varying levels of loss synchronisation, NR-HSTCP provides RTT fairness better than standard TCP once window sizes pass the equivalent thresholds of  $a(w) > 1$  and  $w > w_T$ .

Simulation has validated the NR-HSTCP scheme's ability to ensure that competing NR-HSTCP connections with disproportionate RTT can compete and achieve fair bandwidth share.

Further work would include evaluating the NR-HSTCP scheme in comparison with other existing TCP congestion control algorithms for high speed and high latency networks, such as BIC-TCP [9] and FAST TCP [10].

## VII. ACKNOWLEDGEMENTS

This work is partially supported by the ARC Linkage project LP0455324.

## REFERENCES

- [1] S. Floyd, "Highspeed TCP for large congestion windows," IETF, Tech. Rep. RFC3649, 2003. [Online]. Available: <http://www.ietf.org/rfc/rfc3649.txt?number=3649>
- [2] T. Kelly, "Scalable tcp: improving performance in highspeed wide area networks," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 83–91, 2003.
- [3] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Netw.*, vol. 5, no. 3, pp. 336–350, 1997.
- [4] E. Altman, C. Barakat, E. Laborde, P. Brown, and D. Collange, "Fairness analysis of TCP/IP," in *Proceedings of the IEEE Conference on Decision and Control (CDC'00)*, 2000.
- [5] D. Phillips and J. Hu, "Analytic models for highspeed TCP fairness analysis," in *IEEE ICON 2004*, 2004, pp. 725–730.
- [6] S. Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic," *Computer Communications Review*, vol. 21, no. 5, pp. 30–47, October 1991. [Online]. Available: [citeseer.ist.psu.edu/floyd91connections.html](http://citeseer.ist.psu.edu/floyd91connections.html)
- [7] T. H. Henderson, E. Sahouria, S. McCanne, and R. H. Katz, "On improving the fairness of TCP congestion avoidance," *IEEE Globecom conference, Sydney*, 1998. [Online]. Available: [citeseer.ist.psu.edu/henderson97improving.html](http://citeseer.ist.psu.edu/henderson97improving.html)
- [8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, 1993.
- [9] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (bic) for fast long-distance network," in *INFOCOM*, 2004.
- [10] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, architecture, algorithms, and performance," in *INFOCOM*, 2004.