AFStart: An Adaptive Fast TCP Slow Start for Wide Area Networks

AFStart: An Adaptive Fast TCP Slow Start for Wide Area Networks

Yan Zhang and Nirwan Ansari
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ 07012
{yz45, nirwan.ansari}@njit.edu

Mingquan Wu and Heather Yu
Media Technologies Research
Huawei Technologies USA
Bridgewater, NJ, 08807
{heatheryu,Mingquan.Wu}@huawei.com

***Abstract*—Transmission Control Protocol (TCP) slow start degrades TCP performance under conditions of long-distance and high end-to-end latency, i.e., inherent characteristics of wide area networks (WANs). In this paper, we propose a new TCP slow start algorithm for WANs, called Adaptive Fast Start (AFStart), which incorporates an inline available bandwidth measurement over TCP technique into TCP slow start to set the slow start threshold adaptively and adjusts the congestion window intelligently. The performance of AFStart is evaluated through simulations using the dumb-bell topology and parking-lot topology by applying AFStart to Fast TCP. The simulation results show that AFStart can ramp up the congestion window from its initial value to the slow start threshold more quickly and smoothly than standard slow start, and AFStart achieves higher network link utilization and TCP throughput during the slow start than Fast TCP.**

***Index Terms*—Wide area network (WAN), Transmission Control Protocol, slow start, inline measurement TCP (ImTCP).**

I. Introduction

Today's IT organizations tend to deploy their infrastructures geographically over a wide area network (WAN) to increase productivity, support global collaboration and minimize costs, thus constituting to today's WAN-centered environments. Traditional local area network (LAN) oriented infrastructures are insufficient to support global collaboration with high application performance and low costs. Deploying applications over WANs inevitably incurs performance degradation owing to the intrinsic nature of WANs such as long-distance, high latency and high packet loss rate. Transmission Control Protocol (TCP) is the de facto standard for Internet-based commercial communication networks. However, TCP is well known to have poor performance under conditions of moderate to high packet loss and end-to-end latency.

Many studies have observed that TCP performance suffers from the TCP slow start mechanism in high-speed long-delay networks. In standard slow start, TCP exponentially increases the congestion window *cwnd*, doubling *cwnd* every round-trip time (RTT), until it reaches the slow start threshold *ssthresh*. In WANs, the round-trip time (RTT) is generally in tens and even hundreds of milliseconds. At the beginning of a connection, TCP is trying to probe the available bandwidth while the application data is waiting to be transmitted. Therefore, TCP slow start increases the number of round trips and delays the entire application, thus resulting in inefficient network capacity utilization. TCP slow start can be enhanced by setting slow start threshold *ssthresh* [1]–[4] intelligently and adjusting the congestion window *cwnd* [5]–[7] wisely.

As the switching point between slow start and congestion avoidance, the slow start threshold *ssthresh* is critical to TCP performance. If *ssthresh* is set too low, TCP switches from slow start to congestion avoidance prematurely that may cause TCP to experience a very long time to reach a proper window size. However, a high *ssthresh* may lead to multiple packet losses and more seriously may cause TCP timeouts. Bandwidth-delay product (BDP) calculated with the available bandwidth is a good estimate for *ssthresh*. So, it would be great that TCP can provide some way to measure the available bandwidth and set *ssthresh* intelligently.

At the beginning of the transmission, the exponential increase of the congestion window size is necessary to increase the bandwidth utilization quickly. However, it is too aggressive as the connection nears its equilibrium, leading to a large number of packet losses within one RTT, and more seriously, potentially causing TCP timeouts. In order to shorten the interval that TCP increases *cwnd* from its initial value to *ssthresh* quickly at the beginning and smoothly when it nears *ssthresh*, TCP slow start needs to provide an efficient method to adjust *cwnd* targeting to *ssthresh* wisely.

Moreover, it has been observed that traffic during slow start can be very bursty and can far exceed the BDP of the network path. This serious overshoot may put a heavy load on router queues and produce higher queuing delays, more packet losses and lower throughput. TCP pacing [8] has been proposed to smooth the behavior of TCP traffic by evenly spacing, or pacing, data transmission across a RTT.

In this paper, we propose a new TCP slow start algorithm, called adaptive fast start (AFStart), for WANs. AFStart incorporates an inline available bandwidth probing over TCP technique, namely inline measurement TCP (ImTCP) [9]–[11], into the TCP slow start algorithm. The slow start threshold *ssthresh* can thus be set adaptively with the measured available bandwidth. AFStart recommends an efficient way to grow the congestion window from its initial value to the slow start threshold *ssthresh* quickly and smoothly. AFStart also incorporates TCP pacing [8] to fill the link pipe smoothly.

The remainder of this paper is organized as follows. In the next section, we provide some background on TCP Slow Start. The proposed adaptive fast TCP algorithm for WANs is described in Section II. The TCP performance especially on slow start performance is evaluated in Section III. Finally, we end with a discussion on related work and conclusions in Section V.

II. Adaptive Fast Start Algorithm

This section describes how we integrate an inline measurement, ImTCP, into the TCP startup phase to set *ssthresh* intelligently and how to grow the congestion window from its initial value to the slow start threshold quickly and smoothly.

A. AFStart Algorithm

The AFStart alorithm is described in Fig. 1. Initially, the congestion window $cwnd$ is set to 4 packets. The initial 4 packets are sent back-to-back to measure the available bandwidth by the packet train algorithm [12]. The estimated available bandwidth may not be very accurate with 4 probe-packets one time only measure, but it can provide a good enough reference whether to continue the adaptive fast slow start process or not. Based on this rough available bandwidth estimation, if the calculated BDP is larger than 16 packets, the adaptive fast start algorithm will continue with the updated congestion window $cwnd = 16pkts$ and measuring the available bandwidth by the ImTCP method. Otherwise, the slow start process will switch back to standard slow start.

In order to estimate the available bandwidth within some bandwidth searching range, this bandwidth searching range is divided into 2-4 subranges based on the ratio of the whole bandwidth searching range over the previously estimated available bandwidth as suggested in ImTCP [9]. If the congestion window is large enough, 8 packets are used as the probes for each subrange; otherwise, 4 probes are used for each subrange. Thus, a maximum of only 32 packets will be used for the available bandwidth measurement within one RTT. Moreover, as the results reported in [10] that ImTCP can yield acceptable available bandwidth estimate in intervals as short as 1-4 RTTs. Therefore, ImTCP can be integrated in TCP slow start to set the slow start threshold intelligently without degrading TCP data transmission performance.

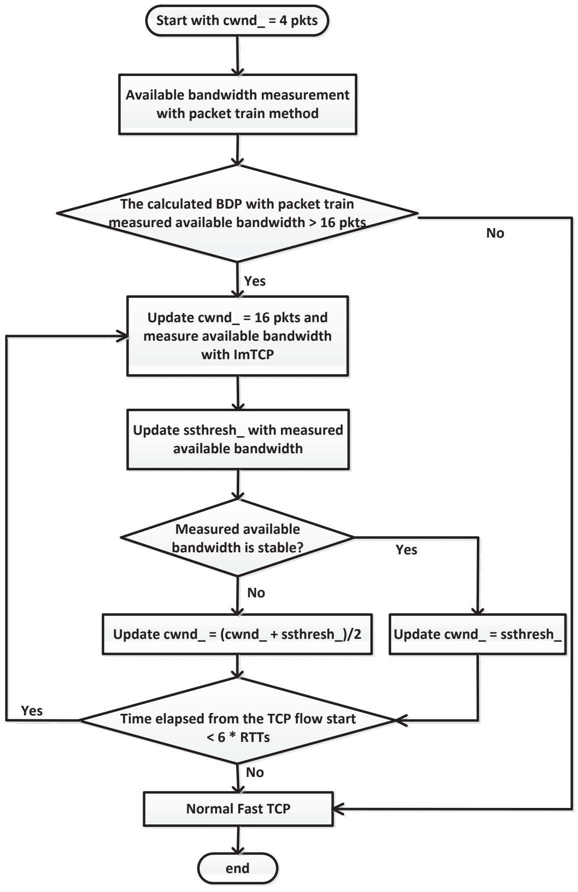
Using ImTCP to measure the available bandwidth, the slow start threshold parameter $ssthresh$ will be updated as:

$$ssthresh = (Avail\_BW * RTT)/(8 * pktsize) \qquad (1)$$

where $Avail\_BW$ denotes the measured available bandwidth, RTT represents the estimated RTT, and $pktsize$ is the packet size in bytes.

If the variance of the measured available bandwidth is smaller than some pre-defined threshold, the measured available bandwidth is stable and the congestion window $cwnd$ is set as $ssthresh$; otherwise, the measured available bandwidth is unstable and the congestion window size is updated with half of the difference between $ssthresh$ and $cwnd$.

$$\begin{aligned} cwnd &= cwnd + (ssthresh - cwnd)/2 \\ &= (ssthresh + cwnd)/2 \end{aligned} \qquad (2)$$



Start with cwnd_ = 4 pkts
Available bandwidth measurement with packet train method
The calculated BDP with packet train measured available bandwidth > 16 pkts
No
Yes
Update cwnd_ = 16 pkts and measure available bandwidth with ImTCP
Update ssthresh_ with measured available bandwidth
Measured available bandwidth is stable?
Yes
No
Update cwnd_ = (cwnd_ + ssthresh_)/2
Update cwnd_ = ssthresh_
Yes
Time elapsed from the TCP flow start < 6 * RTTs
No
Normal Fast TCP
end

Fig. 1: Adaptive fast start algorithm.

Fig. 2 shows the congestion window update iteration by using the AFStart algorithm where $cwnd$ is increased by half of the difference between $ssthresh$ and $cwnd$. From this figure, we can see that with 6 rounds of RTT time, the congestion window can ramp up from its initial value to more than 90% of $ssthresh$. Therefore, after 6 rounds of RTT time, TCP sets the congestion window to $ssthresh$ and switches to congestion avoidance. As a comparison, the congestion window update with standard slow start is also shown in Fig. 2 with $ssthresh$ set to 1000 packets. From this comparison, it is obvious that the standard slow start algorithm increases the congestion window much more aggressive than AFStart, while AFStart ramps up the congestion window faster than exponential growth as the standard slow start does at the beginning of the TCP connection. Thus, AFStart can ramp up the congestion window very quickly from the very beginning of the connection to improve TCP throughput performance during slow start, and avoid the congestion window overshooting problem when the congestion window nears its equilibrium point by decreasing the congestion window increase increment.

Further, in order to smooth TCP behavior in slow start, TCP pacing is applied to all packets except the probe packets used for ImTCP. Within each RTT interval, ImTCP probe packets are transmitted first, and then the other packets are transmitted evenly spacing across the left time interval if more packets can be transmitted.

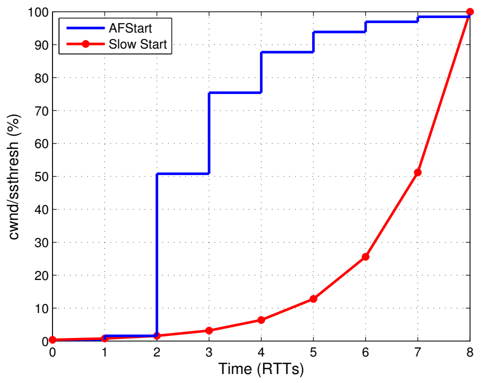


Fig. 2: Congestion window ramps up with AFStart.

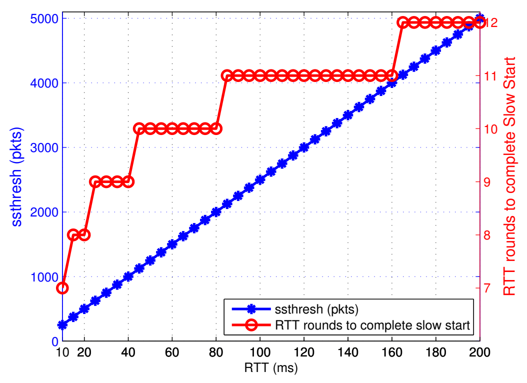


Fig. 3: RTT rounds to complete slow start.

B. Slow Start Improvement Analysis

With standard slow start, the slow start threshold parameter $ssthresh$ is set up initially. This initial $ssthresh$ may be larger than BDP, thus causing TCP flows suffer from temporary queue overflow and multiple packet losses, or smaller than BDP, thus causing TCP flows switch to the congestion avoidance phase prematurely. With standard slow start, TCP flows take $N_{ss} = \lfloor log_2(ssthresh) \rfloor$ rounds of RTT time to complete the slow start stage.

Keeping a consistent available bandwidth 200 $Mbps$ and assuming that the initial $ssthresh$ is set to BDP, Fig. 3 shows the relationships between the RTT time and the slow start threshold parameter $ssthresh$, and the relationship between the RTT time and the required rounds of RTT time to complete the slow start stage with the standard slow start algorithm (the initial $cwnd$ is set to 4). With the increase of RTT from 10 ms to 200 ms, the slow start threshold parameter $ssthresh$ increases linearly and the required rounds of RTT time to complete the slow start stage increases almost log-linearly. However, with the proposed adaptive fast slow start algorithm, a constant $N_{AF} = 6$ rounds of RTT time is required to increase the congestion window to more than 90% $ssthresh$.

III. Performance Evaluation

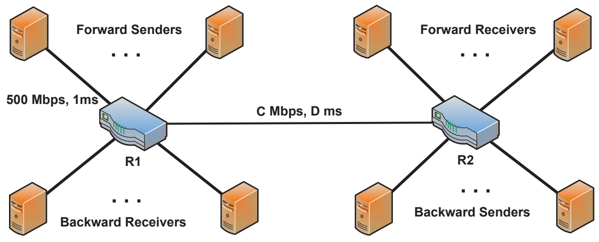
We evaluate the performance of the proposed AFStart algorithm by applying it to Fast TCP [13], and testing it under two different network topologies, namely, the dumb-bell topology and parking-lot topology. Fast TCP is a TCP congestion control algorithm especially targeted at long-distance and high-latency network links. The dumb-bell topology is shown in Fig. 4 (a) with C $Mbps$ link capacity and D ms link delay between routers R_1 and R_2. In the dumb-bell topology simulations, the link capacity is 200 $Mbps$ and the link delay is $D = 20$ ms. We simulate 4 flows totally that are activated and terminated in 20 ms interval, and their activation and termination time are shown in Fig. 4(b). The parking-lot simulation topology is shown in Fig. 5 (a) with (C_1, C_2, C_3) $Mbps$ link capacities and (D_1, D_2, D_3) ms link delays. In the parking-lot topology simulations, the link capacities are all 200 $Mbps$ ($C_1 = C_2 = C_3 = 200$ $Mbps$), and the link delays are all 20 ms ($D_1 = D_2 = D_3 = 20$ ms). We simulate 4 flows in total, and their activation and termination time are shown in Fig. 5 (b). The congestion window trajectories over time under the dumb-bell topology and parking-lot topology are shown in Fig. 6 (a) and Fig. 6 (b), respectively.

We first investigate the congestion window ramp up speed of the proposed AFStart. The slow start improvement analysis for each TCP flows in the dumb-bell topology and the parking-lot topology is summarized in Table I. From this table, we can see that AFStart can ramp up the congestion window from its initial value to the slow start threshold in 6 rounds of RTT time. It is easy to identify from this table that the time spent in the slow start stage with AFStart can save more than 75% than that of Fast TCP.

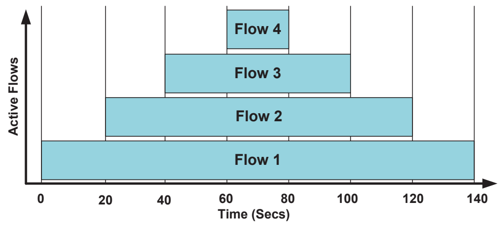
Furthermore, we investigate the averaged TCP throughput during the first several seconds. Fig. 7 (a) and Fig. 7 (b) show the averaged TCP throughput of 4 Fast TCP flows with or without AFStart over the first two seconds under the simulated dumb-bell topology and parking-lot topology, respectively. It is obvious that AFStart improves the TCP throughput during the slow start because AFStart ramps up the congestion window more quickly than the standard slow start.

IV. Related Work

Several works focused on improving the estimate of *ssthresh* with an estimation of BDP. Hoe [1] proposed to enhance TCP slow start performance by setting a better initial value of *ssthresh* to be the estimated value of BDP which is measured by the packet pair method. It has been reported in the literature that other cross traffic may hinder proper estimation of the available bandwidth by the packet pair method. Aron and Druschel [2] proposed to improve the estimate of *ssthresh* iteratively with multiple packet pairs. Paced Start [3] uses packet trains to estimate the available bandwidth. These methods avoid TCP from prematurely switching to the congestion avoidance phase, but they may suffer from temporary queue overflow and multiple packet losses when the bottleneck buffer is not big enough as compared to the BDP. Adaptive Start [4] was proposed to reset *ssthresh* repeatedly to a more appropriate value by using eligible rate estimation. Adaptive

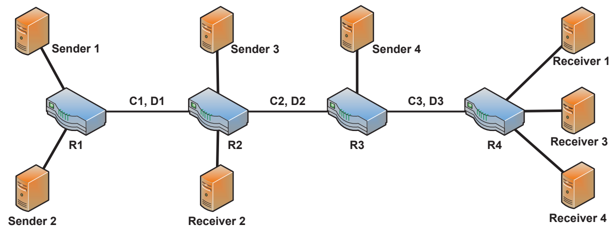


(a) Dumb-Bell topology.

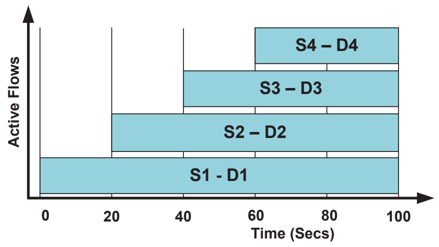


(b) Dumb-Bell topology simulation flows.

Fig. 4: Dumb-Bell topology and simulation flows.

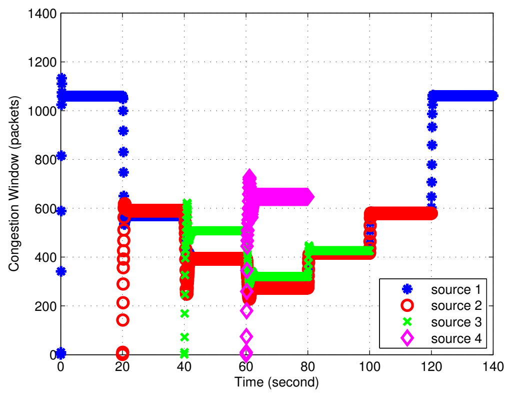


(a) Parking lot topology.

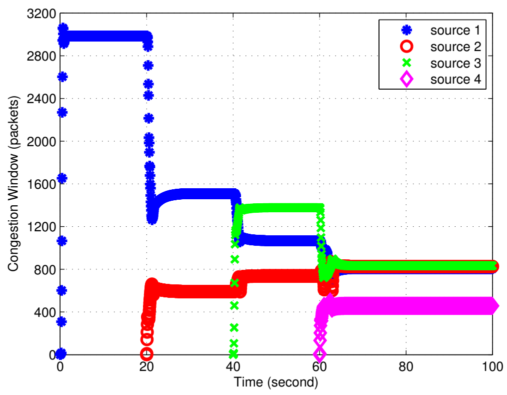


(b) Parking lot topology simulation flows.

Fig. 5: Parking-Lot topology and simulation flows.



(a) Dumb-bell topology.



(b) Parking lot topology.

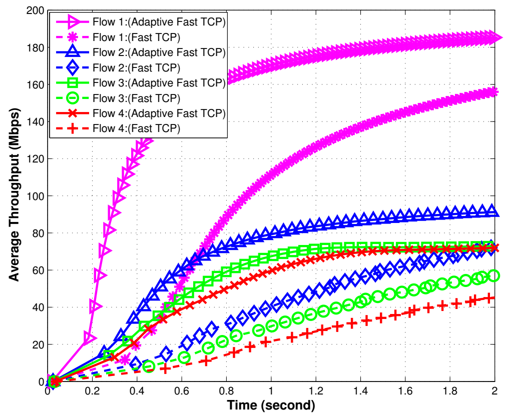
Fig. 6: Congestion window trajectories over time for simulated dumb-bell and parking-lot topologies.

Start interleaves the exponential growth and linear growth of the congestion window to avoid packet overflows. However, Adaptive Start is slower than standard slow start. Moreover, it is designed specially for TCP-Westwood, and not integrated with other TCP variants easily.

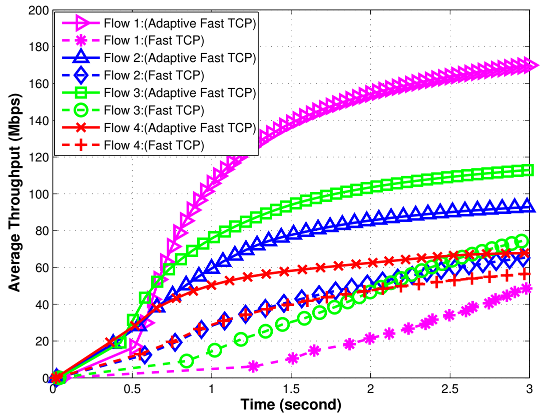
Several TCP slow start enhancements focus on intelligent adjustment of the congestion window. Smooth Start [5] improves the TCP slow start performance as the congestion window approaches the connection equilibrium by splitting the slow-start into two phases, the filling phase and the probing phase. In the filling phase, the congestion window is adjusted in the same manner as standard slow start, while it is increased more slowly in the probing phase. How to distinguish these phases is not addressed in smooth start. An additional threshold *max_ssthresh* is introduced in Limited Slow Start [6]. The congestion window doubles per RTT if the congestion window is smaller than *max_ssthresh*; otherwise, the congestion window is increased by a fixed amount of *max_ssthresh* packets per RTT. Limited Slow Start reduces the number of drops in the TCP slow start phase, but *max_ssthresh* is required to be set statistically prior to starting a TCP connection. Quick-Start [14] can determine the flow's allowed sending rate very quickly, but it requires the cooperations of the routers along the path. In order to address the overshooting problem of slow start, Hybrid Slow Start (HyStart) [7] suggests a slow-start exit point to finish slow start and switch to congestion avoidance, by using two congestion indicators: ACK train length and the increase of RTT delays.

V. Conclusion

In this paper, we have proposed Adaptive Fast Start (AF-Start) for wide area networks that incorporates an inline



(a) Dumb-Bell topology.



(b) Parking-Lot topology.

Fig. 7: Throughput Improvement with adaptive fast start.

TABLE I: **Congestion Window Ramp Up Improvements in Slow Start**

TCP Flows	**Dumb-Bell**			**Parking-Lot**		
	AF Start (RTTs)	**Fast TCP (RTTs)**	**Saving**	**AF Start (RTTs)**	**Fast TCP (RTTs)**	**Saving**
Flow 1	6.25	18.75	66.7%	6.25	25.00	75.0%
Flow 2	6.25	18.75	66.7%	6.25	18.75	66.7%
Flow 3	6.25	37.50	83.3%	6.25	31.25	80.0%
Flow 4	6.25	37.50	83.3%	6.25	37.50	83.3%
Average	6.25	28.13	75.0%	6.25	28.13	76.25%

available bandwidth measurement over TCP technique, called ImTCP, into TCP slow start to measure the available bandwidth efficiently and to set the slow start threshold *ssthresh* adaptively to the bandwidth delay product derived from the measured available bandwidth, and ramp up the congestion window *cwnd* more quickly and smoothly than the standard slow start. We have validated our claims by applying AFStart to Fast TCP, and demonstrated its superiority under two network topologies, namely, dumb-bell topology and parking-lot topology.

References

[1] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme for tcp," in *Proc. of SIGCOMM*, Palo Alto, California, United States, 1996, pp. 270–280.

[2] M. Aron and P. Druschel, "Tcp: Improving start-up dynamics by adaptive timers and congestion control," Rice University, Tech. Rep. TR98-318, 1998.

[3] N. Hu and P. Steenkiste, "Improving tcp startup performance using active measurements: algorithm and evaluation," in *Proc. of IEEE International Conference on Network Protocols*, Nov. 2003, pp. 107 – 118.

[4] R. Wang, G. Pau, K. Yamada, M. Sanadidi, and M. Gerla, "Tcp startup performance in large bandwidth networks," in *Proc. of INFOCOM*, vol. 2, March 2004, pp. 796 – 805.

[5] H. Wang and C. Williamson, "A new scheme for tcp congestion control: smooth-start and dynamic recovery," in *Proc. ot the Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, July 1998, pp. 69 –76.

[6] S. Floyd, "Limited Slow-Start for TCP with Large Congestion Windows," *RFC 3742*, March 2004.

[7] S. Ha and I. Rhee, "Hybrid slow start for high-bandwidth and long-distance networks," in *Proc. of the Sixth International Workshop on Protocols for Fast and Long Distance Networks*, Manchester, UK, 2008.

[8] S. S. Amit Aggarwal and T. Anderson, "Understanding the Performance of TCP Pacing," in *Proc. of the IEEE INFOCOM 2000 Conference on Computer Communications*, March 2000, pp. 1157 – 1165.

[9] C. Le, T. Man, G. Hasegawa, and M. Murata, "A New Available Bandwidth Measurement Technique for Service Overlay Networks," in *Proc. of IFIP/IEEE Management of Multimedia Networks and Services*, Sept. 2003, pp. 436–448.

[10] ——, "Available Bandwidth Measurement via TCP Connection," in *Proc. of IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, 2004, pp. 38–44.

[11] T. Tsugawa, G. Hasegawa, and M. Murata, "Implementation and evaluation of an inline network measurement algorithm and its application to tcp-based service," in *Proc. of 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, April 2006, pp. 34–41.

[12] R. L. Carter and M. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," Boston, MA, USA, Tech. Rep., 1996.

[13] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246 –1259, Dec. 2006.

[14] S. Floyd, M. Allman, A. Jain, and P. Sarolahti, "Quick-Start for TCP and IP," *RFC 4782*, Jan. 2007.