The Interaction between Window Adjustment Strategies and Queue Management Schemes

Chi Zhang[1] and Lefteris Mamatas[2]

[1] School of Computer Science, Florida International University,
Miami, FL 33139, USA
czhang@cs.fiu.edu
[2] Department Of Electrical and Computer Engineering, Demokritos University,
Xanthi 67100, Greece
emamatas@ee.duth.gr

Abstract. In this paper, we investigate extensively the joint network dynamics with different AIMD window-adjustment parameters on end-hosts, and different queue management schemes (i.e. DropTail vs. RED) in routers. We reveal that with DropTail buffer, although smooth TCPs causes less queuing-delay jitter, its average queuing delay is significantly higher than that of responsive TCPs. The direct implication of this discovery is that when mobile users of media-streaming and short messages share a bottleneck link, the energy consumption for sending short messages can increase severely if media-streaming users adopt smooth TCPs. With RED, on the other hand, smooth TCPs not only lead to smaller queue oscillation, the average/max queue length is smaller as well.

1 Introduction

In computer networks, commodity routers and switches often use FIFO buffers to multiplex packets from different flows. Computer networks thus rely on the congestion control algorithms on end-hosts to 'probe' available bandwidth, avoid persistent congestion, and achieve system fairness. The congestion control algorithm of TCP [1] is based on the Additive Increase / Multiplicative Decrease (AIMD) window adjustment strategy [2], to reach satisfactory system equilibrium in a distributed fashion.

While TCP congestion control is appropriate for bulk data transfers, media-streaming applications find the standard multiplicative decrease by a factor of 2 upon congestion to be unnecessarily severe, as it can cause serious throughput oscillations [3]. Authors in [6] investigated the impact of transport protocols on real-time application QoS. Since throughput smoothness is crucial to the subjective performance of multimedia applications, TCP-friendly protocols [3,10] have been proposed with two fundamental goals: (i) to achieve smooth downward adjustments; this is done by increasing the window decrease ratio during congestion, and (ii) to compete fairly with TCP flows; this is approached by

reducing the window increase step according to a steady-state TCP throughput equation. TCP friendly protocols favor *smoothness* by using a gentle backward adjustment upon congestion, at the cost of lesser *responsiveness* - through moderated upward adjustments. In this research, we will study one family of TCP-friendly protocols: TCP(α,β) protocols [10], which parameterize the additive increase value α and multiplicative decrease ratio β of AIMD. The authors in [10] incorporated α and β into a TCP throughput equation, and derived a rough guide for appropriately selecting α and β to achieve TCP friendliness:

$$\alpha = 4(1-\beta^2)/3 \tag{1}$$

Based on experiments, they propose a $\beta = 7/8$ as the appropriate ratio for downward window adjustments upon congestion (i.e. smoother than standard TCP). With $\beta = 7/8$, equation (1) gives an increase value α=0.31 (i.e. less responsive than TCP). We are interested in the family of *TCP-friendly* TCP(α,β) protocols that follow equation (1), because they make tradeoffs between responsiveness and smoothness, and provide a good opportunity to acquire interesting and useful insights into the strategy of window adjustments: By tuning the protocol parameters α and β, we can watch the trends of protocol behaviors under various network and traffic conditions. We categorize three classes of TCP-friendly TCP(α, β) protocols: (i) Standard TCP(1, ½); (ii) Responsive TCP is TCP(α, β) with *relatively* low β value and high α value; (iii) Smooth TCP is TCP(α, β) with *relatively* high β value and low α value.

At the router side, Active Queue Management (AQM) has been recommended for overcoming the two important drawbacks of the straightforward "Drop-Tail" buffer [4]: (i) Lock-Out: In some situations drop-tail buffer allows a single connection or a few flows to monopolize queue space, preventing other connections from getting resources. (ii) Full Queues: Drop-tail strategy tends to keep the queue in a (almost) full status for a long period of time. A persistent large queue increases the end-to-end delay. Random Early Detection (RED) [4] is an AQM scheme dropping packets from among various flows randomly before the gateway's queue overflows, when the average queue length starts to build up.

While there are extensive research works on the joint dynamics of one congestion control strategy with DropTail/RED, little has been done on the interactions between *different* congestion control strategies with *different* queue management schemes. In this paper, we investigate extensively the joint network dynamics with different (α,β) parameters and different queue management schemes (i.e. DropTail vs. RED). We reveal that (i) With DropTail buffer, although smooth TCP causes less queuing-delay jitter, its average queuing delay is significantly higher than that of responsive TCP. The direct implication of this discovery is that when mobile users of media-streaming and short messages share a bottleneck link, the energy consumption for sending short messages can increase severely if media-streaming applications adopt smooth TCPs. (ii) With RED, on the other hand, smooth TCP not only has smaller queue oscillations, the average or max queue length is smaller as well. (iii) Although RED can control the magnitude of queue oscillations, the frequency of queue oscillations can increase, especially with the fast

additive increase speed of responsive TCP. (iv) With multiple bottlenecks and different levels of capacity aggregations, the system fairness is better with responsive TCP when routers use DropTail buffers, or better with smooth TCP when routers use RED.

The rest of the paper is organized as follows. In section 2, we give an intuitive analysis of the dynamics of AIMD control and queue fluctuations. In section 3, the experiment methodology and metrics are given. Section 4 provides detailed results and analysis. Section 5 concludes the paper.

2 The Dynamics of Congestion Control with a Drop Tail Buffer

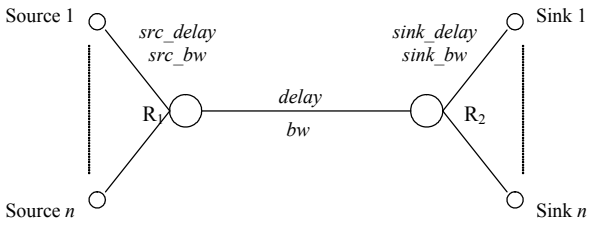

Fig. 1. A simple network topology

Previously (see [11]) we extended the network model of [2]. Consider a simple network topology shown above, in which link bandwidths and propagation delays are labeled. n TCP flows share a bottleneck link with capacity of bw, and the round trip propagation delay is RTT_0. To capture the overall system behavior, we define the aggregated window size at time t as: $cwnd(t) = \sum cwnd_i(t)$, where $cwnd_i(t)$ is the window size of the i^{th} flow. Consequently, the system throughput at time t can be given by the following equation:

$$throughput(t) = \frac{cwnd(t)}{RTT(t)} = \frac{cwnd(t)}{RTT_0 + qdelay(t)} \tag{2}$$

where $qdelay(t)$ is the queuing delay at the bottleneck router R_1.

Consider the time period when all flows are in the additive increase stage. If $cwnd(t)$ is below the point *knee* [2], where $cwnd_{knee} = RTT_0 \cdot bw$, then there is no steady queue build-up in R_1 (i.e. $RTT(t) = RTT_0$), and according to (2), the throughput grows in proportion to $cwnd$. The bottleneck capacity is not fully utilized until $cwnd$ reaches $cwnd_{knee}$.

If $cwnd$ increases further beyond $cwnd_{knee}$, however, the bottleneck queue builds up steadily, with a saturated bottleneck link. If

$$cwnd(t) = cwnd_{knee} + \Delta w(t) \quad (\Delta w(t) > 0) \tag{3}$$

then $\Delta w(t)$ packets will linger in the queue. The TCP flows continue to additively expand their window sizes, until the queue length $\Delta w(t)$ reaches the buffer size, i.e. when *cwnd* touches the point *cliff*, where $cwnd_{cliff} = (RTT_0 + max\ qdelay) \cdot bw$. TCP senders then multiplicatively decrease their congestion windows, after packet losses due to buffer overflow are detected.

The above analysis demonstrates that increasing *cwnd* beyond the knee does not enhance further the system throughput, but only results in increasing queuing delay. Moreover, in order to prevent the system from operating below the knee where bandwidth is underutilized, and meanwhile maintain adequate AIMD oscillations (which affects the speed to converge to fairness [9]), an efficient window decreasing ratio should be

$$\beta = \frac{cwnd_{knee}}{cwnd_{cliff}} = \frac{1}{1+k} \quad where\ k = \frac{BufferSize}{RTT_0 \cdot bw} \tag{4}$$

Furthermore, in [11] we confirmed that in a real system, packet losses may not occur to all flows when the bottleneck buffer overflows, even with drop tail buffers. The selection of which flows to drop is random by nature. With RED, random congestion indications are explicitly performed. Therefore, downward window adjustments are not synchronized among competing flows. We revealed that with unsynchronized multiplicative decreases, the convergence speed to fairness is very slow, if measured by the worst-case fairness [11].

3 Experimental Methodology

We implemented our experiment plan on the ns-2 network simulator. The network topology is shown in Figure 1 in section 2. The propagation delay of access links is 10ms, while the delay of the bottleneck link is 15ms. The capacity of the bottleneck link (*bw*), access links to source/sink nodes (*src_bw/sink_bw*) is 10 Mbps. For simulations of heterogeneous (wired and wireless) networks, ns-2 error models were inserted into the access links to the sink nodes. The Bernoulli model was used to simulate link-level errors with configurable bit error rate (BER). The connection time was 100 seconds.

We selected and evaluated four protocols across a spectrum of TCP-friendly TCP(α, β) protocols, from smooth TCP to responsive TCP: TCP(0.31, 0.875), TCP(0.583, 0.75), TCP(1, 0.5) (standard TCP) and TCP(1.25, 0.25). The size of the bottleneck buffer is 100 packets. The settings for RED are as per [5]. Specifically, *max_th = 3·min_th* and *min_th = BufferSiz/6*.

Protocol behaviors were also evaluated with multiple bottlenecks and cross traffic, using the topology in Figure 2. The router R1 is the bottleneck for the main traffic, which includes TCP flows between “source nodes” to “sink nodes”. The router R3 is another bottleneck for the competing main traffic and cross traffic, which includes TCP flows between “peripheral source nodes” and “peripheral sink nodes”.

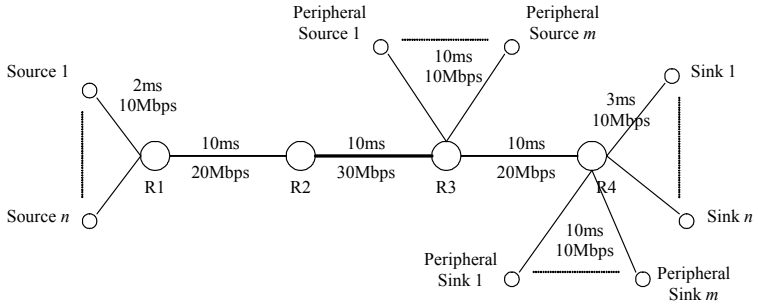

Fig. 2. Network topology with multiple bottlenecks and cross traffic

The system goodput, defined as the sum of the goodput of all flows, is used to measure the overall system efficiency in terms of bandwidth utilization at the receivers. The queue size of the bottleneck router is traced and sampled every 100ms. Long-term Fairness is measured by the Fairness Index, defined in [2]:

$$FairnessIndex = \frac{\left(\sum_{i=1}^{n} throughput_i\right)^2}{n\sum_{i=1}^{n} throughput_i^2}$$

where $throughput_i$ is the throughput of the i^{th} flow. This Fairness Index provides a sort of "average-case" analysis. In order conduct a "worst-case" analysis and provide a tight bound on fairness, the Worst-Case Fairness is defined as:

$$WorstCaseFairness = \frac{\min_{1 \le i \le n} throughput_i}{\max_{1 \le i \le n} throughput_i}$$

When the system is fair on average but particularly unfair to a very small fraction of flows, the unfairness can only be captured by the worst-case fairness (see [11] for details).

4 Results and Observations

4.1 Queue Length and its Impact on Energy Consumption of Mobile Devices

We first simulated 10 flows over the simple network topology described in section 3, with a drop-tail buffer. The bottleneck queue lengths over time are depicted in Figures 3-6. As can be seen from the analysis in section 2, the fluctuations of queues reflect the oscillations of sending rates, when the system operates above the knee. The queue fluctuation of the responsive TCP(1.25, 0.25) is so dramatic that sometimes its queue

length approaches zero, and the bottleneck link is temporarily underutilized because of the idle queue. On the other hand, although smooth window adjustment leads to smaller jitters in queuing delays, the queuing delay remains high throughout the simulation. Due to the high window-decrease ratio upon congestion, the average queue length of TCP(0.31, 0.875) is much higher than the other protocols. Notably this is also true with another smoothness-oriented TCP-friendly protocol TFRC (see our results in [11]).

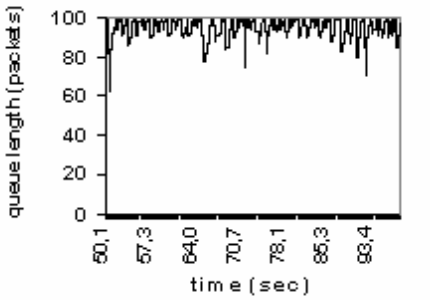

Fig. 3. DropTail queue with TCP (0.31, 0.875)

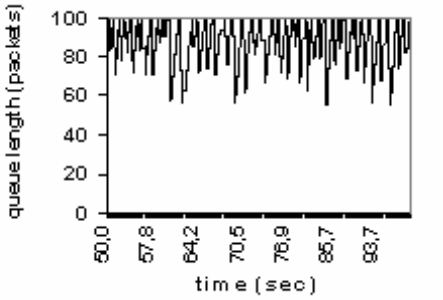

Fig. 4. DropTail queue with TCP (0.583, 0.75)

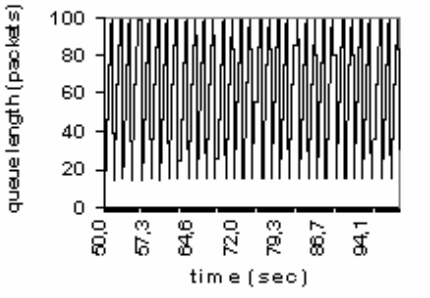

Fig. 5. DropTail queue with TCP (1, 0.5)

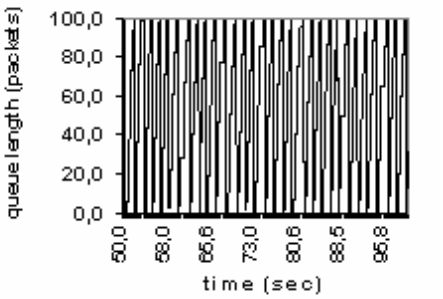

Fig. 6. DropTail queue with TCP (1.25, 0.25)

The queue lengths were also traced with RED configured in routers, shown in Figures 7-10. With smooth TCPs, not only the queue oscillation is smaller, but also the maximum/average queue size is lower. It seems that RED can control the queue growth more effectively with smooth TCP flows. We now give an intuitive analysis why α is the dominant factor in this scenario. Assume the *average* probability that an individual flow experiences packet drops is p_f, which increases with the queue length. The current queue length is determined by the aggregated window size *cwnd(t)*, as shown in section 2. The expected size of the aggregated window in the next RTT will be:

$$
\begin{aligned}
cwnd(t+RTT) &= \sum_{flow\ j \in MD}(cwnd_j(t)\cdot\beta) + \sum_{flow\ i \in AI}(cwnd_i(t)+\alpha) \\
&= p_f \cdot cwnd(t)\cdot\beta + (1-p_f)\cdot cwnd(t) + (1-p_f)\cdot n\cdot\alpha \\
&= cwnd(t) - p_f\cdot(1-\beta)\cdot cwnd(t) + (1-p_f)\cdot n\cdot\alpha
\end{aligned}
\tag{5}
$$

Intuitively, with a small queue size and hence a small p_f, the α-related third term is the dominant factor for window adjustments. More importantly, as the number of flows increases, the impact of the α-related term increases with n, while the β-related term does not. With the same queue length, the larger n is, the stronger the momentum of queue/window increase. Hence, smooth TCPs with a low α are more "responsive" to the early packet drops by RED.

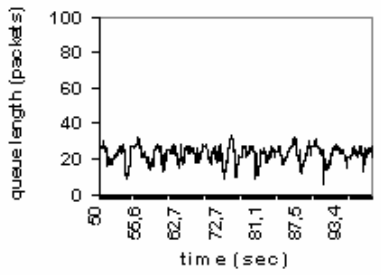

Fig. 7. RED queue with TCP (0.31, 0.875)

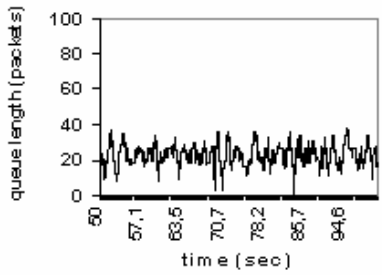

Fig. 8. RED queue with TCP (0.583, 0.75)

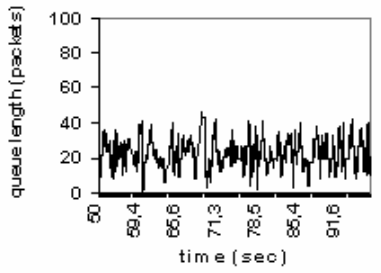

Fig. 9. RED queue length with TCP (1, 0.5)

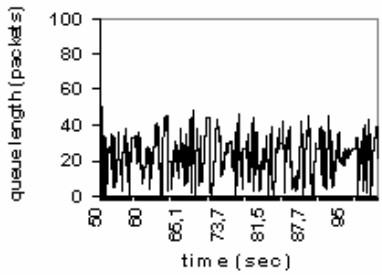

Fig. 10. RED queue with TCP (1.25, 0.25)

After we increase the number of competing flows to 60, the behavior of RED becomes close to DropTail (figures not shown due to the space limit). With a large n, the aggregated increase speed of the system is higher, due to the large $n\alpha$ in equation (5) (even with a small α). The random packet drops by RED between *min_th* and *max_th* cannot effectively reduce the momentum of queue buildup. The average queue frequently touches *max_th*, and RED then drops all packets. The system dynamics is similar to a DropTail buffer, except that the maximum queue is bounded by *max_th = 0.5• buffer_size*.

Implications to Energy Consumptions of Mobile Devices: While the throughput of long flows (e.g. multimedia applications) is determined by the available bandwidth, the connection time of short flows (e.g. short messages of mobile phones) is mainly bounded by RTT. Furthermore, for mobile devices, the extended connection time means higher energy consumptions [7], since they cannot quickly switch back to power-saving mode. Assume that a group of mobile users subscribe to a media-streaming server. Another group of mobile users frequently send/receive short messages (that can fit into one packet) to a short-message server. Media-streaming users use laptops that rely less on batteries, while short messages of mobile phones are more sensitive to energy consumptions. Assume that flows to/from these two servers share a bottleneck link somewhere in the network. Smooth TCP-friendly flows originated from the media-streaming server can cause large persistent queues in the bottleneck buffer, if RED is not deployed, or if RED is deployed but the number of competing flows is large. Data packets or ACKs from the short-message server to the mobile phones can be significantly delayed. That is, the deployment of smooth TCP for media-streaming can adversely affect the connection time and hence the energy consumption of short messages.

4.2 Network Behaviors over Heterogeneous (Wired/Wireless) Networks

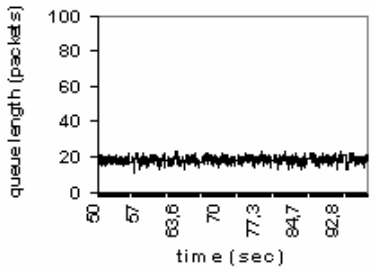

Fig. 11. RED queue Length TCP (0.31, 0.875)

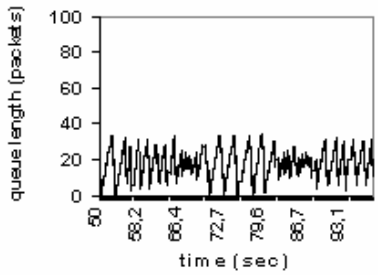

Fig 12. RED queue with TCP (0.583, 0.75)

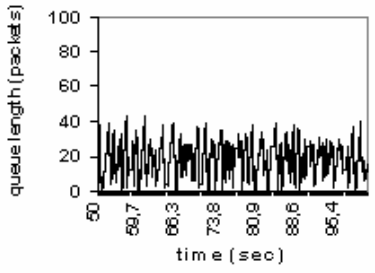

Fig. 13. RED queue with TCP (1, 0.5)

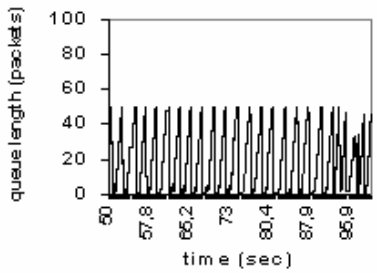

Fig. 14. RED queue with TCP (1.25, 0.25)

We further repeated the simulations in section 4.1 by inserting random wireless bit errors, with packer error rate 1%. It is interesting to observe that although RED can limit the maximum queue and hence the magnitude of queue fluctuation, it cannot fully control the frequency of queue oscillations (Figures 11-14). Since RED forces TCP senders to adjust before the system reaches the *cliff*, the queue fluctuates more frequently. On the other hand, with smooth TCPs, congestion epochs (the time period between two consecutive multiplicative decreases) are extended. Thus, the growing speed of queue is moderated, and the frequency of queue oscillations is reduced.

4.3 Traffic over Complex Topology with Multiple Bottlenecks

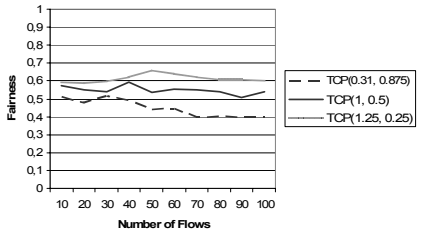

Fig. 15. DroTail Fairness

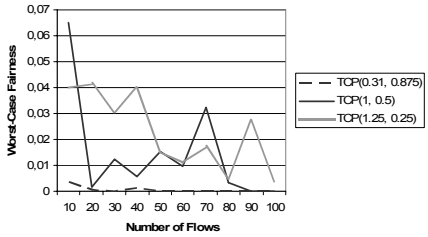

Fig. 16. DroTail worst-case fairness

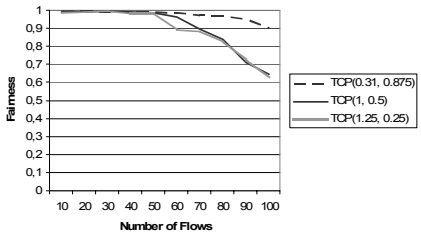

Fig. 17. RED Fairness

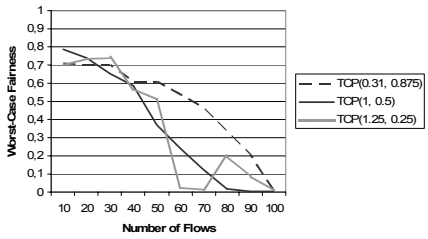

Fig. 18. RED Worst-case Fairness

We also tested the system behaviors with different TCP(α, β) protocols over complex topologies, shown in Figure 2 in section 3. Half of the flows form the main traffic, while the other half form the cross traffic. We choose TCP(0.31, 0.875) to represent smooth TCP, and TCP(1.25, 0.25) to represent responsive TCP. We have already shown in [8] that with drop tail buffers and standard TCP, the main traffic consumes more bandwidth than the cross traffic, due to the fact that packets of main-traffic flows are aggregated before entering R2. They are more uniformly distributed in the time domain, therefore having a smaller probability to get dropped, compared to the burstiness of non-aggregated cross-traffic flows. With RED gateways, better system fairness is achieved. In this paper, we further study the system behaviors with different window adjustment strategies and queue management schemes. With DropTail, responsive TCP's fairness (Figure 15-16),

especially the worst-case fairness, is much higher than smooth TCPs. Notably the lowest goodput (Figure 16) of individual flow is less then 10% of the highest one. Upon packet losses, responsive TCP flows in the main traffic adjust downwards more dramatically, leaving sufficient space for flows in the cross traffic to grow. That is, with responsive AIMD window adjustments, the system is less likely to be biased against less aggregated flows. With RED turned on in the routers, the system fairness (Figures 17-18) is significantly improved, because RED's early random packet drops discard more packets from flows consuming more bandwidth. However, with large number of competing flows, the fairness is still low. Interestingly, with RED, smooth TCP achieves better fairness than responsive TCP. With smooth TCP, RED can more effectively control the queue growth. With the faster increase speed of responsive TCP, the senders can easily overshoot, and the average queue frequently touches the point max_th, where RED drops all packets and behaves similar to DropTail.

5 Conclusion

We investigated the interaction between different window adjustment strategies, and different queue management schemes in routers. We discussed its impact on the end-to-end delay and fairness, and its implications to the energy consumption of mobile phones.

References

1. M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control", RFC2581, April 1999.
2. D.-M. Chiu and R. Jain, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", Computer Networks and ISDN Systems, 17(1):1-14, 1989.
3. S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", In Proceedings of ACM SIGCOMM 2000, August 2000.
4. S. Floyd, and V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, 1(4):397-413, August 1993.
5. S. Floyd, "RED: Discussions of Setting Parameters", November 1997, available from http://www.icir.org/floyd/REDparameters.txt
6. P. Papadimitriou and V. Tsaoussidis, "On Transport Layer Mechanisms for Real-Time QoS", TR-DUTH-EE-2005-10, Feb. 2005
7. C. Jones, K. Sivalingam, P. Agrawal and J. Chen, "A Survey of Energy Efficient Network Protocols for Wireless Networks", ACM Journal on Wireless Networks, vol. 7, No. 4, 2001.
8. V. Tsaoussidis and C. Zhang, "TCP-Real: Receiver-Oriented Congestion Control", Computer Networks Journal (Elsevier), Vol. 40, No. 4, November 2002.
9. V. Tsaoussidis and C. Zhang, "The Dynamics of Responsiveness and Smoothness in Heterogeneous Networks", IEEE Journal on Selected Areas in Communications, March 2005.
10. Y.R. Yang and S.S. Lam, "General AIMD Congestion Control", IEEE ICNP '00, Nov 2000.
11. C. Zhang and V. Tsaoussidis, "Improving TCP Smoothness by Synchronized and Measurement-based Congestion Avoidance", In Proceedings of ACM NOSSDAV 2003, June 2003.