A Study of Deploying Smooth- and Responsive- TCPs with Different Queue Management Schemes

Chi Zhang
Juniper Networks, USA
chizhang@juniper.net

Lefteris Mamatas
Department of Electronic & Electrical Engineering
University College London, UK
lmamatas@ee.ucl.ac.uk

Vassilis Tsaoussidis
Department of Electrical and Computer Engineering
Demokritos University of Thrace, Greece
vtsaousi@ee.duth.gr

ABSTRACT

While there exist extensive research works on congestion control and active queue management, or the joint dynamics of a congestion control strategy with the Random Early Detection (RED) algorithm, little has been done on the interactions between different window adjustment strategies and different queue management schemes such as DropTail and RED. In this paper, we consider a spectrum of TCP-friendly Additive Increase and Multiplicative Decrease (AIMD) parameters. At the one end of this spectrum, smooth TCP enhances smoothness for multimedia applications by reducing the window decrease ratio upon congestion, at the cost of the additive increase speed and the responsiveness to available bandwidth. At the other end, responsive TCP enhances the responsiveness by increasing the additive increase speed, at the cost of smoothness. We investigate the network dynamics with various combinations of AIMD parameters and queue management schemes, under different metrics. The investigation is conducted from the deployment (especially incremental deployment) point of view. We discussed the impact of the interactions on the goodput, fairness, end-to-end delay, and its implications to energy-consumption on mobile hosts.

1. Introduction

In packet-switched computer networks, routers and switches usually do not maintain per-flow states, and nor do they provide bandwidth reservation mechanisms for competing flows. In

most routers, FIFO buffers are used for multiplexing packets from different flows. Since the available bandwidth and the number of competing flows are unknown to end-hosts, protocols need to 'probe' to detect the available bandwidth and congestion in the network. Congestion control of standard TCP [ASP99] is based on the Additive Increase / Multiplicative Decrease (AIMD) window adjustment strategy [CJ89] that exploits available bandwidth, avoids persistent congestion, and achieves system fairness. Traditional AIMD is a somewhat "blind" mechanism, in the sense that the congestion window increases steadily until the occurrence of congestion, which triggers the subsequent error recovery. That is, congestion control mechanism itself is the natural cause of congestion, and congestion can be detected only by packet drops.

In addition, on routers and switches, a complementary mechanism, Active Queue Management (AQM), has been proposed to alleviate the congested bottleneck queue with a straightforward "Drop-Tail" buffer [BRA98]. Random Early Detection (RED) [FJ93] is an active queue management scheme highly recommended for deployment in routers [BRA98]. RED attempts to provide negative feedback to sending hosts by dropping packets. To prevent high delays and bursty drops, prior to the queue overflow, RED drops packets from among various flows probabilistically, if contention increases (i.e. the queue steadily builds up). Each drop is expected to trigger a congestion-oriented response from the corresponding TCP sender. Notably, high queuing delay hurts the subjective performance of interactive applications such as Telnet or short Web transfers.

Although TCP congestion control is basically appropriate for bulk data transfers, some real-time applications such as media-streaming find the standard multiplicative decrease by a factor of 2 upon congestion to be unnecessarily severe, as it can cause data-rate oscillations and even transmission gaps [TZ99]. TCP-friendly protocols [FHPW00, YL00] therefore have been proposed with two fundamental goals: (i) to achieve smooth backward 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 [PFTK98]. That is, TCP-friendly protocols favor smoothness by using a gentle backward adjustment upon congestion, at the cost of lesser responsiveness - through moderated upward adjustments.

To be more specific, GAIMD [YL00] is a TCP-friendly protocol that generalizes AIMD congestion control by parameterizing the congestion window increase value α and decrease ratio β, where the sender's window size is increased by α if there is no packet loss in a round-trip time, and the window is decreased to β times the current value if there is a loss indication. A

throughput equation for standard TCP (α=1, β=½) is first introduced in [PFTK98], which is extended by [YL00] to include parameters α and β:

$$T_{\alpha,\beta}(p, RTT, T_0, b) = \frac{1}{RTT\sqrt{\frac{2b(1-\beta)}{\alpha(1+\beta)}}p + T_0 \min\left(1, 3\sqrt{\frac{(1-\beta^2)b}{2\alpha}}p\right)p(1+32p^2)} \quad (1)$$

where p is the loss rate; T_0 is the retransmission timeout value; b is the number of packets acknowledged by each ACK. The overall throughput of TCP-Friendly (α, β) protocols is bounded by the average throughput of standard TCP(α=1, β=0.5), which means that equation (2) derived from (1) (see [YL00]) could provide a rough guide to achieve TCP-friendliness.

$$T_{\alpha,\beta}(p, RTT, T_0, b) = T_{1,0.5}(p, RTT, T_0, b) \quad (2)$$

Authors of [YL00] derive from (1) and (2) a simple relationship for α and β:

$$\alpha = 4(1-\beta^2)/3 \quad (3)$$

Based on experiments, they propose a β = ⅞ as the appropriate value for smooth multiplicative decrease (i.e. less rapidly than TCP does) for media-streaming applications. For β = ⅞, (3) gives an increase value α=0.31, which sacrifices the protocol's responsiveness to rapidly-available bandwidth.

We are particularly interested in TCP(α, β) protocols, as they provide a good opportunity to obtain interesting and useful insights into the strategy of window adjustments: by tuning the protocol parameters α and β, we can observe the protocol behavior under various network and traffic conditions. In this paper, we consider a spectrum of TCP-friendly AIMD parameters and their interactions with queue management schemes such as Drop-Tail and RED. In our discussion below, we refer to three classes of 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.

Basically, smooth TCP enhances smoothness for multimedia applications by reducing the window decrease ratio upon congestion, at the cost of the additive increase speed and the responsiveness to available bandwidth. Responsive TCP enhances the responsiveness by increasing the multiplicative decrease ratio, at the cost of smoothness.

In this paper, we investigate network dynamics with various combinations of additive increase and multiplicative decrease parameters and queue management schemes. This is important from the deployment (especially incremental deployment) point of view. More specifically,

(a) With DropTail buffer, although smooth TCP causes less queuing delay jitter, its average queuing delay is much higher than that of responsive TCP, since the bottleneck buffer is always close full. The direct implication of this discovery is that when mobile users of media-streaming and short messages share some bottleneck links, the energy consumption for sending short messages can increase dramatically if media-streaming users adopt smooth TCP.

(b) With RED, smooth TCP not only has smaller queue oscillations, the average/max queue length is smaller as well. But, when the number of competing flows is high, the interactions of TCP(α, β) protocols with RED display dynamics similar to that with DropTail.

(c) Although RED can control the magnitude of queue oscillations, the frequency of queue oscillations can be significantly affected by the fast additive increase speed of responsive TCP.

(d) With multiple bottlenecks and different levels of capacity aggregations, the system fairness can be better with smooth TCP in case of RED gateways, or better with responsive TCPs in case of DropTail gateways.

(e) With partial / incremental deployment of smooth or responsive TCP, the system behaviors with goodput and fairness depend on the buffer management schemes at routers, the network topology, as well as the *location* where the new protocol is deployed.

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, experiment methodology and metrics are given. Section 4 provides detailed results and analysis. Section 5 concludes the paper.

2 Dynamics of Congestion Control with DropTail Buffer

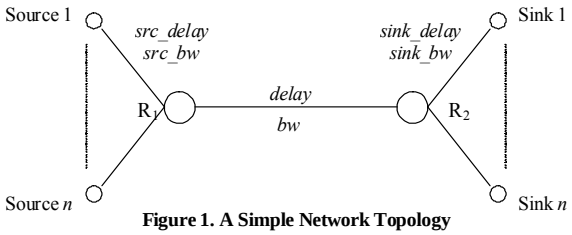


Figure 1. A Simple Network Topology

First we extend the analysis model of AIMD in [CJ89] by taking into account the role of bottleneck queue. Consider a simple network topology shown in Figure 1, in which the link bandwidth and propagation delay are labeled. In this scenario, *n* TCP flows share a bottleneck

link with capacity of *bw*, and the round trip propagation delay is RTT_0 = *2 * (src_delay + delay + sink_delay)*. We define the aggregated congestion window size at time t as:

$$cwnd(t) = \sum_{i=1}^{n} cwnd_i(t) \quad (4)$$

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:

$$throughput(t) = \frac{cwnd(t)}{RTT(t)} = \frac{cwnd(t)}{RTT_0 + qdelay(t)} \quad (5)$$

where *qdelay(t)* is the queuing delay at the bottleneck router R1. As can be seen from (5), the throughput is not only a function of the congestion window, but also a function of the queuing delay.

Assume all flows are in the additive increase stage. First consider the case where *cwnd(t)* is below the point knee [CJ89]:

$$cwnd_{knee} = RTT_0 \cdot bw \quad (6)$$

Then there is no steady queue build-up in R1 (i.e. RTT(t) = RTT0), and according to (5), the throughput grows in proportion to *cwnd*. The bottleneck capacity is not fully utilized until cwnd increases to $cwnd_{knee}$.

If *cwnd(t)* increases further beyond $cwnd_{knee}$, the system displays different dynamics. The bottleneck queue starts to build up, after the bottleneck capacity is saturated. Rewrite *cwnd(t)* as:

$$cwnd(t) = cwnd_{knee} + \Delta w(t) \quad (\Delta w(t) > 0) \quad (7)$$

Since the bottleneck link can transmit at most $cwnd_{knee}$ packets in one RTT_0 (see (6)), $\Delta w(t)$ packets will linger in the queue. Hence the steady queuing delay at the bottleneck will be:

$$qdelay(t) = \Delta w(t) / bw \quad (8)$$

Intuitively, the system throughput is bounded by the physical capacity *bw*, in spite of the increase of *cwnd(t)* beyond the knee, because *qdelay(t)* in the denominator of (5) grows as well. This is confirmed by the following:

$$throughput(t) = \frac{cwnd_{knee} + \Delta w(t)}{RTT_0 + qdelay(t)} = \frac{RTT_0 \cdot bw + qdelay(t) \cdot bw}{RTT_0 + qdelay(t)} = bw \quad (9)$$

The system dynamics can be described by equations (7) – (9), until the queue length $\Delta w(t)$ reaches the maximum buffer size, i.e. when *cwnd* touches the point cliff.

$$cwnd_{cliff} = (RTT_0 + \max qdelay) \cdot bw \quad (10)$$

TCP senders then multiplicatively decrease their congestion window, after packet losses due to buffer overflow are detected.

Equation (9) demonstrates that increasing *cwnd* beyond the knee does not enhance further the system throughput, but only results in increasing queuing delay. However, the analysis also indicates that some queue build-up is inevitable, in order to provide to the fairness-oriented AIMD algorithm an operating scope where the system throughput fully exploits the bottleneck bandwidth. More precisely, although multiplicative decrease is necessary to accomplish fairness dynamically [ZT02], it does not necessarily mean that the throughput will be sacrificed, as long as the system operates between the knee and the cliff.

Furthermore, the analysis of [CJ89] assumed a synchronized model, meaning that all flows synchronously adjust backward upon congestion indication. However, results in [TZ05] confirm the early findings [DIM01] that global synchronization rarely happens even with DropTail buffer. That is, in a real system, packet losses do not occur to all flows when the bottleneck buffer overflows. Some flows experiencing early packet drops reduce their sending windows quickly, which might bring about partial queue draining. This could leave sufficient space for additive increase afterwards, and hence the remaining flows keep growing. Due to this partial backward adjustment upon congestion, from the system perspective, the multiplicative decrease ratio of the aggregated window is higher than the ratio β an individual flow adopts. The selection of which flows to drop is random by nature. With active queue management, such as RED, random congestion indications are explicitly performed. Obviously, unsynchronized multiplicative decrease degrades the short-term fairness, due to random congestive drops that permit some flows to grow beyond their fairshares, at the cost of the other flows forced to decrease, in a short period of time. However, an adequate level of long-term fairness can still be achieved, due to the inherent characteristic of randomness in selecting which flows to drop.

3. Experimental Methodology

3.1 Simulation Configuration and Evaluation Plan

We have implemented our evaluation plan on the ns-2 network simulator [NS]. The network topology used as a test-bed is shown in Figure 2. The bottleneck link capacity (bw_bottleneck), the access links to source nodes (bw_src) and the access links to sink nodes (bw_dst) were occasionally re-configured for the different scenarios. In most cases however, bw_bottleneck = bw_src = bw_dst unless it is pointed out explicitly otherwise. For simulations

of heterogeneous (wired and wireless) networks, ns-2 error models [NS] were inserted into the access links at the sink nodes. The Bernoulli model with a configurable bit error rate (BER) was used to simulate independent bit errors due to wireless interferences. The number of flows (or the number of source-sink pairs) *N*, varied from experiment to experiment. The simulated connection time was fixed at 100 seconds.

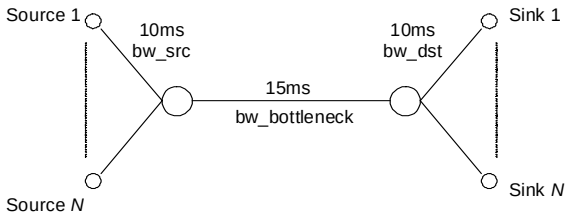


Figure 2. Network topology

In order to validate our statements on the behavior of equation-based protocols with parameters α and β, we selected and evaluated four protocols that span across the spectrum from smoothness to responsiveness and satisfy the TCP-friendly equation (3). Our four protocols are TCP(0.31, 0.875), TCP(0.583, 0.75), TCP(1, 0.5) and TCP(1.25, 0.25). TCP(1, 0.5) is the standard TCP. The size of the bottleneck buffer is 100 packets. The settings for RED are as per [Flo97]. Specifically, we adjusted the RED settings as follows: *max_th = 3 min_th*, *min_th = BufferSize / 6*.

Protocol performance and fairness were also evaluated with multiple bottlenecks and cross traffic, using the scenario of Figure 3. The router R1 is the bottleneck for the main traffic (flows between source nodes to sink nodes), while the router R3 is another bottleneck for the competing main traffic and cross traffic (flows between peripheral source nodes and peripheral sink nodes).

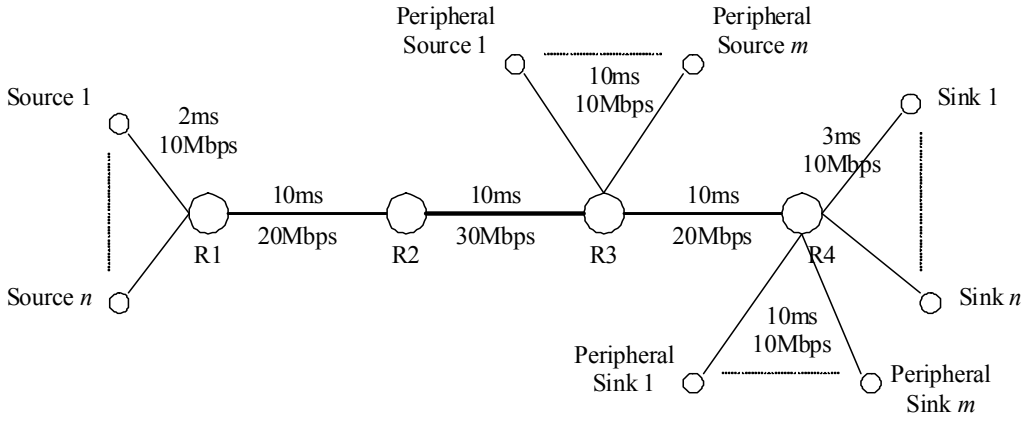


Figure 3. Network Topology with Multiple Bottlenecks and Cross Traffic

3.2 Performance Metrics

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 goodput results in this paper are measured in Bytes/second.

Long-term Fairness is measured by the Fairness Index, defined by [CJ89]:

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

where $throughput_i$ is the throughput of the i^{th} flow, measured at a time scale of connection time. 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 [ZT03] is also used:

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

The range of worst-case fairness is also in [0, 1], with 1 representing the greatest fairness. As an example demonstrating why worst-case fairness is used, consider a scenario of 6 flows, the throughputs of which are 9 Mbps, 9.5 Mbps, 8.5 Mbps, 9 Mbps, 9 Mbps, and 6 Mbps, respectively. The traditional "average-case" fairness index is 0.982, while the worst-case fairness is 0.667. Compare this scenario with a perfectly fair case in which all flows achieve 9.5 Mbps, and both the "average-case" fairness index and worst-case fairness index are 1.0. The difference between the first scenario and the ideal case can't be obviously distinguished by the "average-case" fairness index. In the first scenario, the system is fair in general, but is particularly unfair to the 6th flow. This unfairness to a very small fraction of flows can only be captured by the worst-case fairness.

In addition, the queue size of the bottleneck router, measured in packets, is traced and sampled every 100ms.

4. Results and Analysis

4.1 The Impact on Queue Length and End-to-End Delay

4.1.1 Dynamics with DropTail Gateways

We first evaluated the protocol dynamics over the network topology shown in Figure 2. 10 flows shared a 10Mbps bottleneck link with a DropTail buffer of size 100 packets. The round trip propagation delay is around 70 ms. Bottleneck queue lengths over time are depicted in Figures 4 (a) – (d). As can be expected, the fluctuation of queue lengths reflects the fluctuation of sending rate. The queue length fluctuations of responsive TCP is so dramatic (due to the low β) that at some time instances the 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 delay jitter, the queuing delay remains high throughout the simulation. Due to the high window decrease ratio upon congestion indications, the average queuing length of TCP(0.31, 0.875) is much higher than the other protocols.

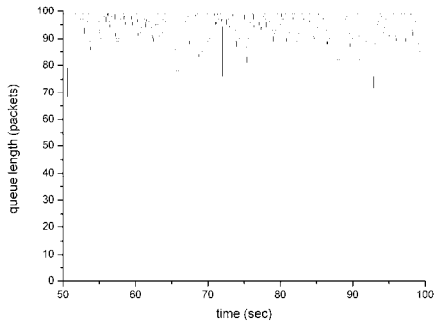


Figure 4(a) Queue Length with TCP (0.31, 0.875) (DropTail gateway)

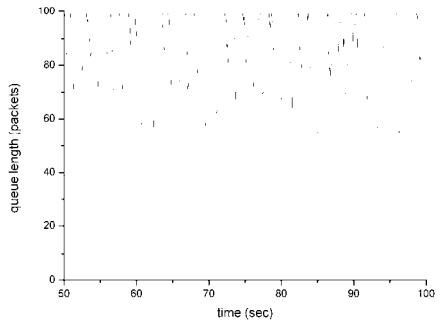


Figure 4(b) Queue Length with TCP (0.583, 0.75) (DropTail gateway)

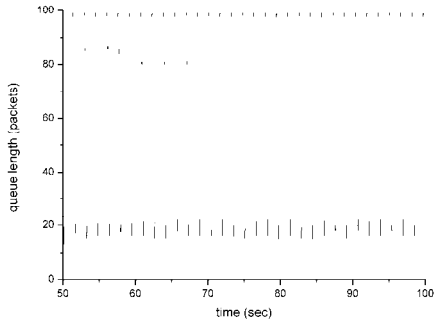


Figure 4(c) Queue Length with TCP (1, 0.5) (DropTail gateway)

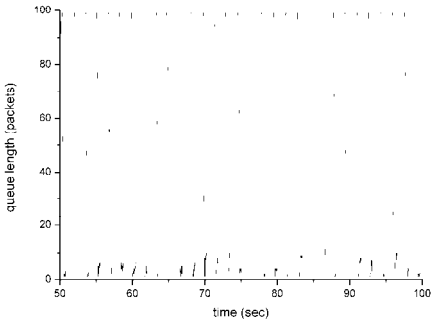


Figure 4(d) Queue Length with TCP (1.25, 0.25) (DropTail gateway)

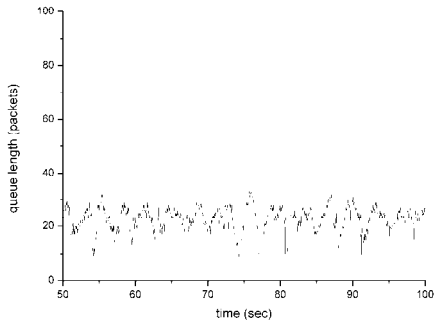
4.1.2 Dynamics with RED Gateways

The queue lengths were also traced with RED gateways, shown in Figures 5 (a) – (d). The buffer size remains 100 packets, while `min_th` and `max_th` are set to be 1/6 and 1/2 of the buffer size, respectively, as per [Flo97]. With smooth TCPs, not only the queue oscillation is smaller, but also the maximum queue size is lower, as a result of the interaction between the end-

to-end AIMD window control and the active queue management. It seems that RED can control the queue growth more effectively with smooth TCP flows, since the moderated additive increase steps allow more time for RED to respond. We analyze why α is the dominant factor in this scenario as follows. The queue length is determined by the aggregated window size *cwnd(t)* (the sum of congestion window sizes of all flows). The average window size of each flow is therefore given by *acwnd(t) = cwnd(t) / n*, assuming there are *n* competing flows. The packet dropping rate p_a of RED is a function of the queue length. The probability that the i^{th} flow multiplicatively decreases is $p_f = 1 - (1 - p_a)^{cwndi}$, where $cwnd_i$ is the window size of the i^{th} flow. The expected size of the aggregated window size in the next RTT will be:

$$
\begin{aligned}
& cwnd(t + RTT) \\
&= n \cdot [p_f \cdot (acwnd(t) \cdot \beta) + (1 - p_f) \cdot (acwnd(t) + \alpha)] \\
&= n \cdot acwnd(t) + \Delta win \\
&= cwnd(t) + \Delta win \\
& where \;\; \Delta win = -p_f \cdot (1 - \beta) \cdot cwnd(t) + (1 - p_f) \cdot n\alpha
\end{aligned}
$$

Δwin , the increase rate of aggregated window size, consists of two terms. The β-related term is negative while the α-related term is positive. Intuitively, with a small queue size and hence a small p_f, the second term (α-related) is the dominant factor for window growth. The influence of the multiplicative decrease ratio β is mitigated by the low dropping rate p_f. More importantly, as the number of flows increases, the α-related term increases with *n,* while the β-related term does not. That is, with the same aggregated window size *cwnd(t)*, the number of flows and the corresponding fair-share of the network can be different. Although the instant queue length mainly depends on the instant *cwnd(t)*, the above equation shows that with a large *n*, the momentum of queue length increase is stronger. In other words, multiplicative decrease ratio does not scale with the number of flows, while additive increase value does. The impact of α can be higher than β, as the number of competing flows increases. Hence, smooth TCPs with a low α are more "responsive" to the early packet drops, and the growing speed of the queue can be more effectively controlled by RED.



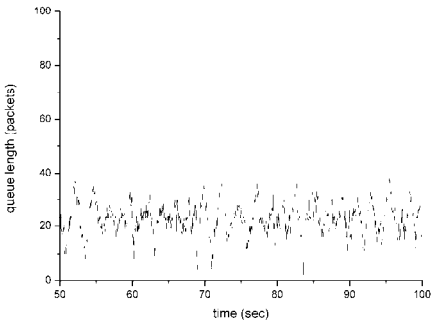


Figure 5(a) Queue Length with TCP (0.31, 0.875) (RED gateway)

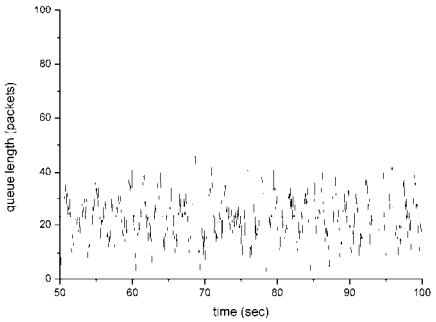


Figure 5(b) Queue Length with TCP (0.583, 0.75) (RED gateway)

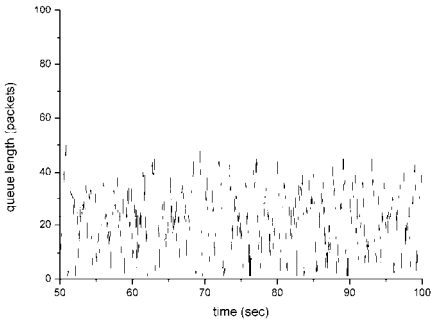


Figure 5(c) Queue Length with TCP (1, 0.5) (RED gateway)

Figure 5(d) Queue Length with TCP (1.25, 0.25) (RED gateway)

After we further increase the number of competing flows to 60, the system behavior changes completely with RED, as showed with Figures 6 (a) – (d). The queue lengths are straightened at the top, since the maximum queue length is bounded by the parameter *max_th* of RED (in our simulation, *max_th* • *buffer_size* = 0.5 ·100 = 50 packets), for both smooth TCP and responsive TCP. Responsive TCP has a smaller average queue size, due to the larger queue oscillation. Based on the analysis above, with a large number of competing flows, the additive increase speed of the entire system $N\alpha$ is higher (even with a small α of smooth TCP), and the random packet drops between *min_th* and *max_th* of RED cannot effectively control the queue length. Eventually, the average queue length reaches *max_th* • *buffer_size*. RED then enforces to drop all packets. This makes the system dynamics similar to a DropTail buffer, except that the maximum queue length is halved.

Figure 6(a) Queue Length with TCP (0.31, 0.875) (RED gateway)

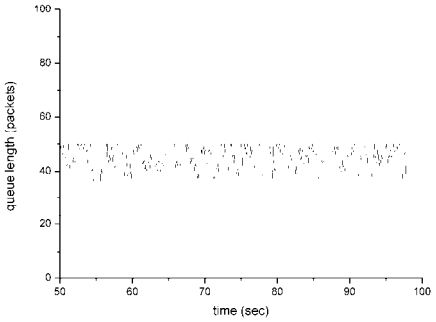
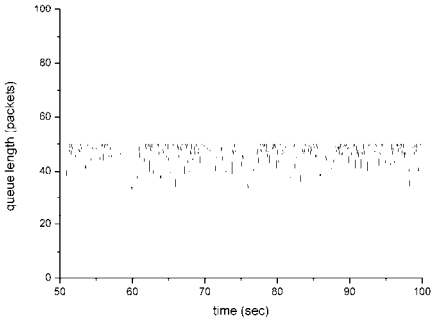


Figure 6(b) Queue Length with TCP (0.583, 0.75) (RED gateway)



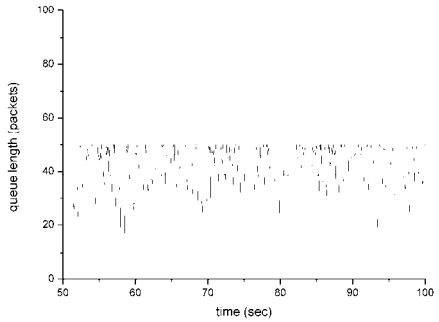


Figure 6(c) Queue Length with TCP (1, 0.5) (RED gateway)

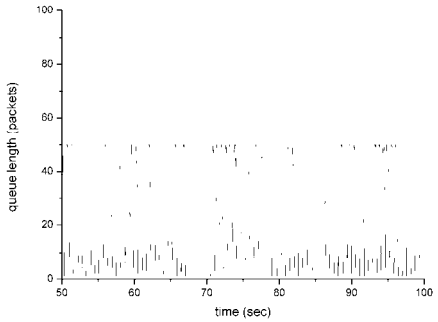


Figure 6(d) Queue Length with TCP (1.25, 0.25) (RED gateway)

4.1.3 Implications to Energy Consumptions of Mobile Users

For users of mobile devices, the energy consumptions depend largely on the transmission time [JSAC01, KB02], which is determined by throughput. With long flows (elephants, such as media-streaming applications) the throughput is determined by the available bandwidth, while for short flows (mice, such as web page retrievals) the throughput is mainly bounded by RTT.

Assume that a number of mobile users in a wireless network access a single media-streaming server. That is, there are a number of smooth TCP flows coming from the server to the wireless users watching a show (maybe on laptops relying less on battery lives). If RED is not deployed, smooth TCP can cause large persistent queues in the bottleneck buffer (see section 4.1.1). Suppose another group of users send short messages (that can fit into one packet) to another server (for relaying short messages based on the Session Initiation Protocol) that shares the same bottleneck buffer (e.g. the buffer on base stations) with the media-streaming server. Since the traffic of media-streaming causes large persistent queues, the TCP ACKs sent from the short-message server to the mobile hosts are delayed. That is, without RED, the deployment of smooth TCP for media streaming applications can significantly increase the connection time of mobile users sending short messages. The extended connection time implies higher energy consumption, since the message senders cannot turn off their devices until they receive acknowledgements.

Even with RED configured, if the number of competing flows is high, this implication still holds (as shown in section 4.1.2), except that the maximum queuing delay is lowered by *max_th* of RED.

4.2 Network Behaviors over Heterogeneous (Wired/Wireless) Networks

We further repeated the simulations over the same network topology as section 4.1, and inserted random wireless bit errors (see section 3.1) into the access links to sink nodes. In the following results, a Bit Error Rate of 1% is used.

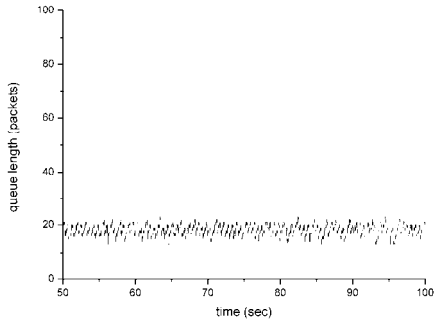


Figure 7(a) Queue Length with TCP (0.31, 0.875) (RED gateway)

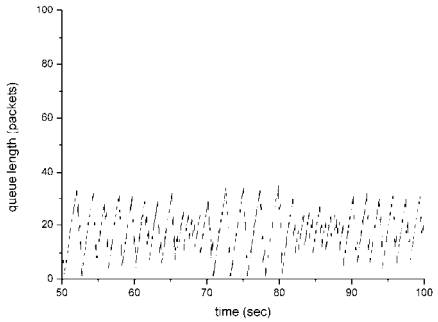


Figure 7(b) Queue Length with TCP (0.583, 0.75) (RED gateway)

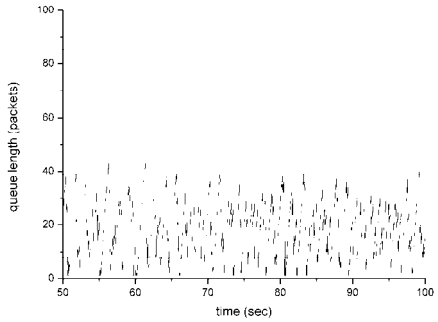


Figure 7(c) Queue Length with TCP (1, 0.5) (RED gateway)

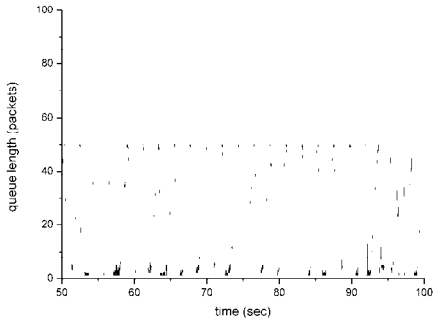


Figure 7(d) Queue Length with TCP (1.25, 0.25) (RED gateway)

It is interesting to observe that although RED can limit the maximum queue size and hence the magnitude of queuing delay fluctuation, it can not fully control the frequency of queue oscillations (Figures 7 (a) – (d)). Note that high frequency indicates that the end-to-end delay is more difficult to predict, given the current instant delay. In fact, RED increases the frequency of queue oscillations by forcing TCP senders to adjust before the system reaches the cliff. Therefore, the congestion epochs[1] are reduced with RED, and the queue approaches its peak quickly and fluctuates more frequently. On the other hand, the frequency of queue oscillations is also significantly affected by the window adjustment strategies on end hosts. With smooth TCPs, congestion epochs are extended. Thus, the growing speed of queue is moderated, and the frequency of queue oscillations is reduced.

[1] the time period that reflects the *uninterrupted growing lifetime* of sending window

4.3 Traffic over Complex Topology with Multiple Bottlenecks

We also tested the system behaviors with different TCP(α, β) protocols over the complex topology with multiple bottlenecks (shown in Figure 3), with both DropTail and RED. 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. The fairness, worst-case fairness, goodput performance are depicted in Figures 8 - 11. The results are analyzed and discussed below.

4.3.1 Homogeneous Flows over Multiple Bottlenecks

We first configured the simulations such that all flows adopt the same TCP(α, β) protocol. With DropTail, responsive TCP achieves slightly lower throughput than smooth TCP, which verifies the early findings in [TZ05].

There is a large oscillation with the worst-case fairness index, reflecting the inherent characteristics of randomized multiplicative decrease discussed in section 2. Notably the lowest goodput of individual flow is less then 10% of the highest one, as demonstrated by worst-case fairness in Figure 8(b). It has already been shown in [TZ02] that with DropTail routers and standard TCP, the main traffic consumed more bandwidth than the cross traffic, due to the fact that packets aggregated into link R1-R2 (see Figure 3) before entering the bottleneck R3 were more uniformly distributed in the time domain, therefore having smaller probability to get dropped, compared to the bursty traffic of non-aggregated (i.e. connecting to R3 directly) cross-traffic flows. That contributes to the low worst-cast fairness. As shown below, with RED gateways, better system fairness can be achieved.

Interestingly, the responsive TCP's fairness (Figure 8(a)), especially the worst-case fairness (Figure 8(b)) is much higher than smooth TCPs. 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 large AIMD window adjustments, the system is less likely to have bias against less aggregated flows.

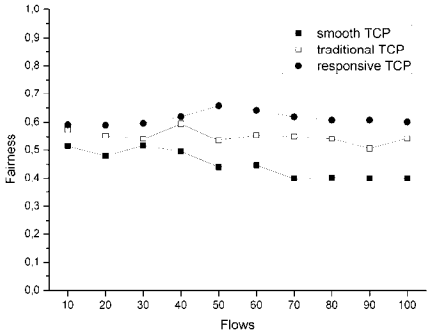


Figure 8(a) Fairness vs Number of Flows

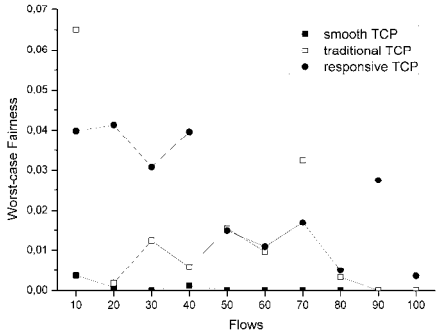


Figure 8(b) Worst-case Fairness vs Number of Flows

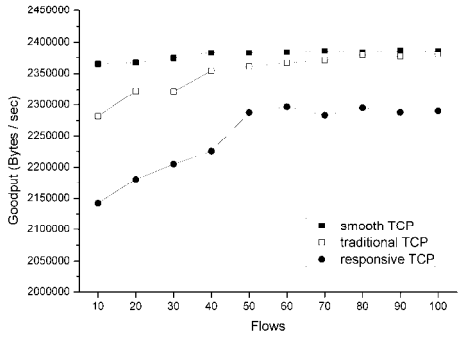


Figure 8(c) Goodput vs Number of Flows

Figure 8. Protocol Behaviors with DropTail buffer

With RED turned on at the router R1, in contrast to the scenario of DropTail buffer, responsive TCP may achieve slightly higher goodput (Figure 9(c)). Overall, the system goodput is slightly higher with RED

Moreover, with RED turned on, the system fairness (Figure 9(a)) is significantly improved. RED's random packet drop prior to buffer overflow will discard more packets from flows that consumes more bandwidth, therefore counterbalance the system unfairness due to different levels of link aggregation. However, with large number of competing flows, fairness, especially the worst-case fairness (Figure 9(b)), is still low. But, in contrast to the case of DropTail gateways, smooth TCP achieves much better fairness than responsive TCP. A detailed trace analysis reveals that with smooth TCP, RED can more effectively control the queue growth and packet losses. With responsive TCP's faster additive increase speed, the senders can easily overshoot, and the average queue tends to touch or even exceeds the point *max_th*, where forced

drops (dropping all packets, instead of dropping randomly) are executed and RED behaves similar to DropTail, in terms of bias against less aggregated flows.

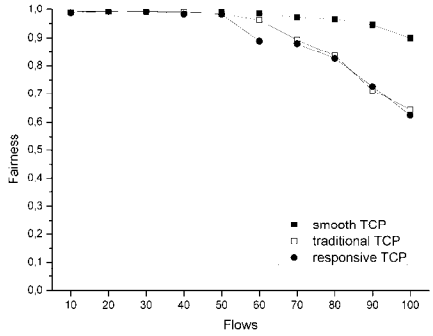


Figure 9(a) Fairness vs Number of Flows

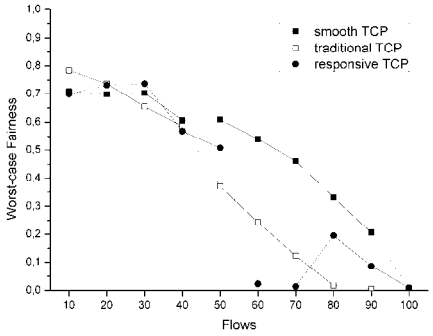


Figure 9(b) Worst-case Fairness vs Number of Flows

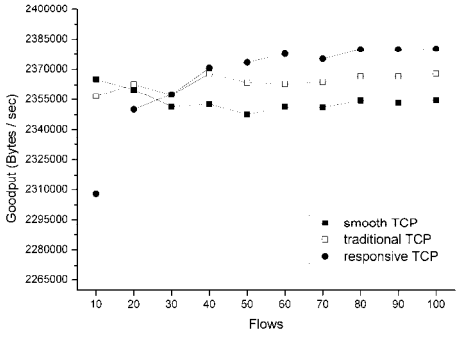


Figure 9(c) Goodput vs Number of Flows

Figure 9. Protocol Behaviors with RED configured at R1

4.3.1 Heterogeneous Flows over Multiple Bottlenecks

We then configured the simulation such that flows adopt different TCP(α, β) protocols over the multi-bottleneck topology. In this scenario with hybrid protocols, the main traffic uses smooth TCP (or responsive TCP), while the cross traffic uses responsive TCP (or smooth TCP). The protocol performance with RED and DropTail are depicted in Figures 10 – 11.

The four curves on each figure represents the test cases where (i) all flows use smooth TCP (ii) all flows use responsive TCP (iii) the flows of main traffic use smooth TCP, while the flows of cross traffic use responsive TCP (iv) the flows of main traffic use responsive TCP, while the flows of cross traffic use smooth TCP. The results show that with heterogeneous flows, when the main traffic uses smooth TCP, the system behavior is closer to the case where all flows

use smooth TCP. Similarly, with heterogeneous flows, when the main traffic uses responsive TCP, the system behavior is closer to the case where all flows use responsive TCP. Therefore, the results verify that the dominant factor is the protocol deployed on the main traffic with aggregated flows. We conclude that deploying smooth/responsive TCP alone cannot control the behavior of fairness or goodput, since the system behavior also depends on the buffer management at routers as well as where smooth/responsive TCP is deployed.

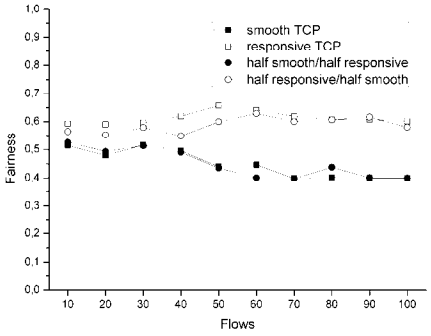


Figure 10(a) Fairness vs Number of Flows

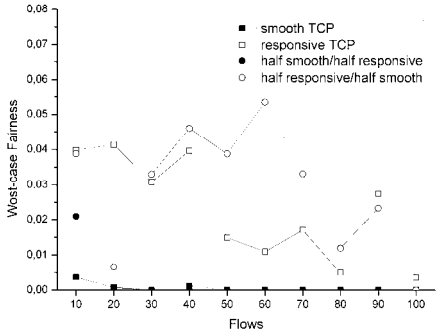


Figure 10(b) Worst-case Fairness vs Number of Flows

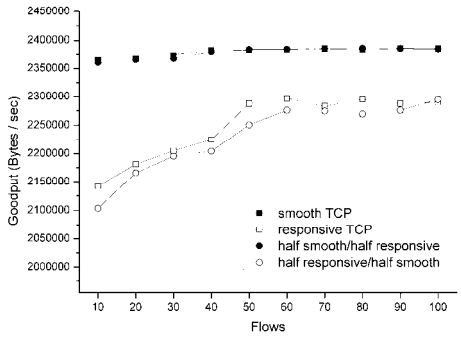
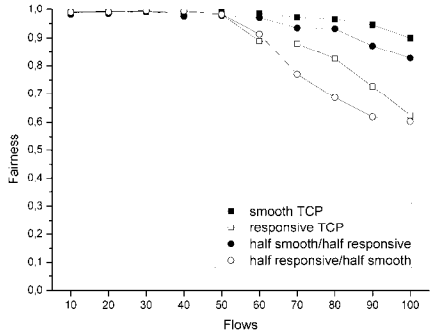
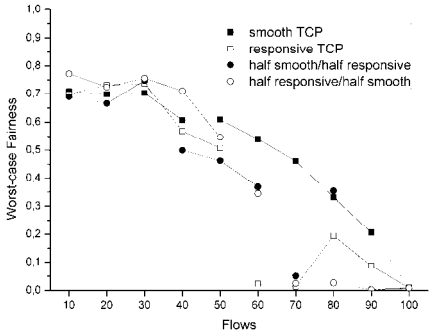


Figure 10(c) Goodput vs Number of Flows

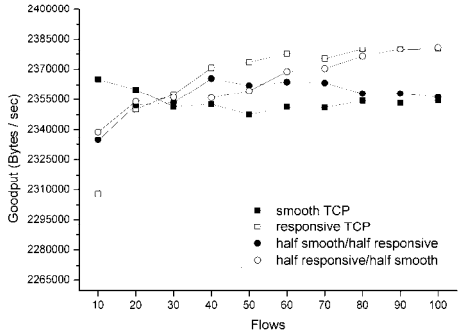
Figure 10. Behaviors of Heterogeneous Flows with DropTail Buffer



11(a) Fairness vs Number of Flows



11(b) Worst-case Fairness vs Number of Flows



11(c) Goodput vs Number of Flows

Figure 11. Behaviors of Heterogeneous Flows with RED Configured at R1

5. Conclusion and Future Work

We investigated the network dynamics with different window adjustment strategies and queue management schemes. From the deployment (especially incremental deployment) point of view, we observed the combined dynamics between different (α, β) parameters on end-hosts and different queue management schemes, their impact on the fairness, end-to-end delay, and their implications to energy-consumption of mobile hosts in wireless networks.

We discover that with DropTail buffer, although smooth TCP causes less queuing delay jitter, its average queuing delay is much higher than that of responsive TCP. This implies that when mobile users of media-streaming and short messages share the same bottleneck link, the energy consumption for sending short messages can increase dramatically if media-streaming users adopt smooth TCP. On the other hand, with RED and relatively small number of competing

flows, smooth TCP not only has smaller queue oscillations, the average/max queue length is smaller as well. However, the frequency of queue oscillations can be significantly affected by the fast additive increase speed of responsive TCP.

We further investigated incremental deployment issues with different AIMD parameters and different queue managements. With multiple bottlenecks and different levels of capacity aggregations, the system fairness with smooth TCP can be much worse than responsive TCP, if RED is not deployed at the gateways. With partial / incremental deployment of smooth or responsive TCP, the system performance with goodput and fairness depends on the buffer management schemes at routers, the network topology, as well as the *location* where the new congestion control mechanism is deployed.

We conclude that incremental deployment of window-adjustment strategies or queue management schemes will not necessary lead to the desired effect, if the behavior of the combined system is not considered.

Our future work will be providing analytical analysis to theoretically support the findings in this paper.

References

[APS99] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control", RFC2581, April 1999.

[BRA98] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[CJ89] 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, June 1989.

[DIM01] C. Diot, G. Iannaccone and M. May, "Aggregate Traffic Performance with Active Queue Management and Drop from Tail", ACM SIGCOMM Computer Communication Review, vol. 31, issue 3, pp. 4 – 13, July 2001.

[FHPW00] S. Floyd, M. Handley, J. Padhye, and J. Widmer, “Equation-Based Congestion Control for Unicast Applications”, In Proceedings of ACM SIGCOMM 2000, pp. 43 – 56, August 2000.

[FJ93] S. Floyd, and V. Jacobson, “Random Early Detection gateways for Congestion Avoidance”, IEEE/ACM Transactions on Networking, 1(4):397-413, August 1993.

[Flo97] S. Floyd, “RED: Discussions of Setting Parameters”, November 1997, available from http://www.icir.org/floyd/REDparameters.txt

[JSAC01] C. Jones, K. Sivalingam, P. Agrawal and J. Chen, “A Survey of Energy Efficient Network Protocols for Wireless Networks”, ACM/Baltzer Journal on Wireless Networks, vol. 7, No. 4, pp. 343 - 358, 2001.

[KB02] R. Krashinsky and H. Balakrishnan, "Minimizing energy for wireless web access with bounded slowdown", Proceedings of ACM MobiCom, pp. 119 - 130, September 2002

[KZT02] M. Khanna, C. Zhang and V. Tsaoussidis, “Experimental Evaluation of RED in Heterogeneous Networks”, The 3rd International Conference on Internet Computing, pp. 217-224, June 2002.

[NS] ns-2 Network Simulator, http://www.isi.edu/nsnam/ns/, 2001.

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", In Proceedings of ACM SIGCOMM ’98, pp. 303 – 314, August 1998.

[TZ99] D. Tan and A. Zakhor, “Real-time Internet Video Using Error Resilient Scalable Compression and TCP-friendly Transport Protocol”, *IEEE Transactions on Multimedia*, vol. 1, no. 2, pp. 172-186, May 1999.

[TZ02] V. Tsaoussidis and C. Zhang, “TCP-Real: Receiver-Oriented Congestion Control”, Computer Networks Journal (Elsevier), Vol. 40, No. 4, pp. 477-497, November 2002.

[TZ05] V. Tsaoussidis and C. Zhang, “The Dynamics of Responsiveness and Smoothness in Heterogeneous Networks”, IEEE Journal on Selected Areas in Communications, vol. 23, no. 6, pp. 1178-1189, March 2005.

[YL00] Y.R. Yang and S.S. Lam, “General AIMD Congestion Control”, In Proceedings of the 8th International Conference on Network Protocols, p. 187, November 2000.

[ZT02] C. Zhang and V. Tsaoussidis, “The Interrelation of TCP Smoothness and Responsiveness in Heterogeneous Networks”, Proceedings of the 7th IEEE Symposium on Computers and Communications (ISCC 2002), pp. 291-297, July 2002.

[ZT03] C. Zhang and V. Tsaoussidis, “Improving TCP Smoothness by Synchronized and Measurement-based Congestion Avoidance”, Proceedings of ACM NOSSDAV 2003, pp. 131-140, June 2003.