The Interrelation of TCP Responsiveness and Smoothness in Heterogeneous Networks

C. Zhang and V. Tsaoussidis
College of Computer Science, Northeastern University
Boston, MA 02115, USA
{czhang, vassilis}@ccs.neu.edu

Abstract

TCP(α,β) protocols trade the congestion window increase value α for the decrease ratio β, to generate smoother traffic patterns and to maintain a friendly behavior. In this paper, we study the design assumptions of TCP(α,β) protocols and discuss the impact of equation-based modulation of α and β on application efficiency. We confirm experimentally that, in general, smoothness and responsiveness constitute a tradeoff; however, we uncover undesirable dynamics of the protocols when the network or flow characteristics do not follow a prescribed and static behavior. For example, we show that smooth backward adjustments confine the protocol's capability to exploit resources that become available rapidly, and embarrass the fair and efficient growth of incoming flows. Furthermore, we show that in the context of wireless networks with high error rate, a low α dictates a conservative behavior that degrades the protocol performance with both delay-tolerant and -sensitive applications; and in the context of high contention of heterogeneous flows, a low α does not contribute to efficiency and friendliness.

1. Introduction

Transmission control of reliable protocols, as exemplified by TCP [1], is based on somewhat "blind" increase/decrease window mechanism that exploits the bandwidth availability dynamically and, meanwhile, avoids persistent congestion. The adjustments are modeled on the Additive Increase/Multiplicative Decrease algorithm from the perspective of fair resource allocation and efficient resource utilization [2]. AIMD is the core algorithm of standard TCP and is becoming the core algorithm of all transport protocols that support congestion control functions [3].

The problems of standard TCP have been mainly investigated from two different perspectives, namely the application requirements and the characteristics of the underlying networks. The former expounds the impact of the transmission gaps caused by halving the transmission rate during congestion on the quality of delay-sensitive applications; authors in [4, 5, 8, 9] propose TCP-friendly protocols that satisfy two fundamental goals: (i) to achieve smooth window adjustments; this is done by reducing the window decrease ratio during congestion, and (ii) to compete fairly with TCP flows; this is approached by reducing the window increase factor according to a steady-state TCP throughput equation. It has been effectively established that TCP can achieve application-oriented improvements by favoring smoothness using a gentle backward adjustment upon congestion, at the cost of lesser responsiveness (i.e., speed to approach an equilibrium) - through moderated upward adjustments. The latter perspective unfolds the need for error detection and classification that would permit a responsive strategy, oriented by the nature of the error detected (congestion in wired networks versus transient random errors in wireless networks) [7]. Implementation of such strategy requires a more responsive TCP.

In this paper, we investigate the interrelation of TCP smoothness and responsiveness by studying the behaviors of TCP(α,β) protocols [8]. TCP(α,β) protocols parameterize 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. We discuss the impact of the smoothness/responsiveness tradeoff on application performance, assuming initially that it follows strictly the friendliness-oriented α/β tradeoff. A natural question is therefore "under what conditions can we achieve efficiency and friendliness". That is, when the prescribed conditions change, would their impact be symmetrical on both TCP and TCP-friendly protocols? Extending this question in the context of the present work, since friendliness is based on the combined dynamics of smoothness and responsiveness, the real question is whether the conditions affect symmetrically these two components of congestion

control. 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.

We study the protocols' behavior based on a contrasting-pair strategy for different requirements and characteristics of applications and networks, respectively. More precisely, we compare the performance of our TCP(α,β) versions under the following conditions: (i) in wired and heterogeneous (wired and wireless) networks. (ii) in static and dynamic[1] environments (iii) with delay-sensitive and -tolerant applications. (iv) with single- and multi-protocol communication channels. Based on the assumptions of equation-based congestion control and on experimental data, we arrive at the conclusion that TCP-friendly protocols that are based entirely on the α/β tradeoff may be adequate for specific applications, networks and scenarios; however, they are inappropriate for several other occasions.

We organized the paper as follows: We give an overview of TCP(α,β) protocols in section 2. In Section 3 we discuss the assumptions and conditional applications of equation-based congestion control. Here, we also justify our expectations in the context of networks with distinctive error characteristics (e.g. congestion versus wireless errors) and we highlight some observations on the dynamics of responsiveness and smoothness. In section 4 we justify our testing methodology and we select performance metrics. In section 5 we present the results of our experiments and in section 6 we highlight our conclusions.

2. Trading α For β

A throughput equation for standard TCP is first introduced in [6]. GAIMD [8] extends the equation 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 $(\alpha = 1, \beta = 0.5)$, which means that equation (2), which is derived from (1) (see [8]) could provide a rough guide to achieve friendliness.

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

[1] From the perspective of the participating flows with criterion whether their number is fixed or not.

Authors of [8] 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 $\beta = 7/8$ as the appropriate value for the reduced the window (i.e. less rapidly than TCP does). For $\beta = 7/8$, (3) gives an increase value $\alpha = 0.31$.

The observations of the window dynamics and event losses are frequently assumed within a time period of a *congestion epoch* [4], which reflects the *uninterrupted growing lifetime of congestion window*. More precisely, a congestion epoch begins with βW packets, increased by α packets per RTT and reaching a congestion window of W packets, when a packet is dropped. The congestion window is then decreased to βW. Hence, a congestion epoch involves

$$n = (1-\beta) * W / \alpha + 1 \ RTTs \quad (4)$$

Assuming that the capacity of the bottleneck link is B packets per second and the number of active flows going through the bottleneck router is N, and assuming a control system as in [2], we further calculate that:

$$W = B * RTT / N \quad (5)$$

3. Observations on the Dynamics of Responsiveness and Smoothness

We present below some observations on the dynamics of TCP responsiveness and smoothness. We use the observations to justify our experimental results and also to present further assumptions in the form of hypotheses. We use the results to verify the correctness of the hypotheses.

Observation 1: *It takes several RTTs for a small α to pay back the bandwidth credit of a high β.*

Equation (1) is modeled by calculating the average throughput over a congestion epoch, which is associated with several RTTs. Since equation (1) gives the *steady state* TCP throughput, in a dynamic network where conditions changing rapidly, friendliness might not be attained. More precisely, based on (4) we conclude that (1) and (2) can be achieved at a time *n RTTs* or later since multiple drops will extend further the time of convergence. Based on (4) and (5) we further conclude that the time period required for (1) and (2) to hold is in reverse proportion to the number of flows within a fixed bandwidth channel; the smaller the number, the larger the window and therefore the longer the convergence time. This is confirmed by our results shown in section 5.2. Finally, the propagation delay has a direct impact on the time required for TCP(α, β) to reach a full-window size. Practically (and deterministically) this means that for a window of 64KB and an RTT of 100ms TCP(1, ½) needs at least 3.2 seconds to reach the maximum window size.

Observation 2: *In the case of multiple packet drops, the aggressive / conservative behavior of TCP(α, β) is dominated by α. Hence, a smooth TCP may not balance low responsiveness with smoothness in heterogeneous networks.*

A hidden assumption behind (3) is that when packet drops occur at the end of the congestion epoch, the window decreasing by a factor of (1-β) is applied only once. However, multiple packet drops could cause the window size to be decreased multiple times, or they could also cause the retransmission timer to expire. At the end, it is possible that the window size and the *ssthresh* could be decreased down to 2 segments, even with smooth backward adjustments. Under such scenarios, the performance of applications (including real-time applications) is not affected by how slowly the sender reduces its sending rate, but rather by how fast it can recover from the error and restore its sending rate. Note that our scenario is not unrealistic. For example, in mobile networks, burst correlated errors and handoffs generate this kind of error pattern. The aggressiveness of responsive TCP is the desirable behavior, because in this scenario the bandwidth is still available though the packet dropping rate is high. This is confirmed by the results shown in sections 5.1 and 5.3.

Observation 3: *When additional Bandwidth becomes available, a responsive TCP approaches its fair share faster than a smooth TCP. Hence, in a dynamic system of multiple, smooth TCP flows, if a number of flows leaves the system earlier than others, the remaining flows cannot exploit the bandwidth well.*

It can be seen from observation 1 that a smooth TCP extends the duration of the congestion epoch. Due to a smaller α, the protocol requires more steps to approach its fair-share, when some flows leave the system and bandwidth becomes available. Obviously, responsiveness is here too the dominant parameter of efficiency since it reflects the protocol's capability to exploit the available bandwidth. See our results in section 5.2.

Hypothesis 1: *In a dynamic system where contention gradually increases, if flows are homogeneous[2] and β is high, the smooth window adjustments of existing flows may not guarantee friendliness to incoming flows.*

A conclusion of the control system presented in [2] is that the flows approach fairness faster when the window oscillations are larger. Since improved smoothness (hence degraded responsiveness) implies more steps to reach the desired level of fairness, convergence to fairness can be extended. Essentially, when new flows enter a system of multiple smooth TCP flows at equilibrium, the smooth backward adjustment is expected to extend the time to converge to fairness. In such case, it is desirable that existing flows drop their sending rate quickly to make available bandwidth for new flows. See our results in section 5.2.

Hypothesis 2: *In a system where flows are heterogeneous[3] and contention is high, responsiveness is the dominant factor of bandwidth utilization. Hence, when smooth and responsive protocols co-exist and contention is high, the responsive protocols may be favored.*

Due to high contention, the initial resources of each flow are rather modest and the multiplicative decrease is rarely activated; the increasing rate of smooth protocols has also a minor effect, in contrast to the responsive protocols that manage occasionally to increase their windows. Here responsiveness is expected to dominate the consumption of bandwidth. If multiple protocols co-exist in this context, smooth TCP may be too conservative, allowing the standard TCP to consume bandwidth more aggressively. This is shown by the results in section 5.4.

Hypothesis 3. *Since throughput is not a direct function of the sending rate, throughput of smooth TCP may be greater than the throughput of standard and responsive TCP.*

Equation (2) indicates that the protocols will always achieve about the same throughput as the standard TCP. However, the assumption of equations (1) – (3) that the system throughput increases in proportion to the sending rate, is inaccurate. After the load reaches the network capacity, throughput stops increasing. If the load is increased beyond this point, called *knee* in [2], the queue length builds up. Throughput may suddenly drop when the load increases beyond a point *cliff*, where packets start experiencing significant delay, or may be dropped. An efficient congestion control mechanism should keep the network operating in the zone between the knee and the cliff, where throughput is maximal. In a homogeneous environment with high bandwidth and large fair share, a responsive TCP is likely to operate outside that zone at the beginning of the congestion epoch. This degrades the protocol's capability to utilize the available bandwidth throughout the connection, *pace* equation (2)'s projection. The experimental results are shown in section 5.1.

4. Experimental Methodology

4.1 Testing Plan

We have implemented our testing plan on the ns-2 network simulator. The network topology used as a test-

[2] (α, β) is the same for all flows.

[3] (α, β) is not the same for all flows.

bed is the typical single-bottleneck *dumbbell*, as shown in Figure. 1. 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 were inserted into the access links at the sink nodes. The Bernoulli model was used to simulate link-level errors with configurable packet error rate (PER). The number of flows (or the number of source-sink pairs) *N*, varied from experiment to experiment. The connection time was fixed at 100 seconds.

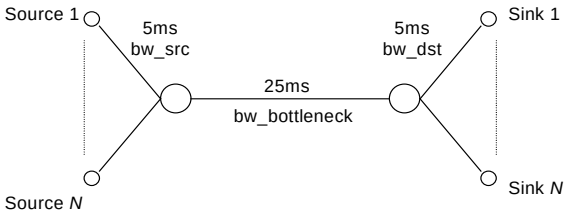

Figure 1. Network topology

In order to validate our statements about the behavior of equation-based protocols with parameters α and β, we selected and evaluated four protocols that that span across a spectrum of smoothness and responsiveness and satisfy the TCP-friendly equation (3). Our four versions 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.

We begin our testing with simulations of ftp applications, over wired and heterogeneous (wired and wireless) networks in a static environment. To evaluate how efficiently and fairly the protocols can exploit the bandwidth that becomes available, or can share the existing bandwidth with new incoming flows, we considered dynamic scenarios where the number of active flows gradually falls off or picks up, respectively, during the experimentation time.

We evaluate the protocols' performance with delay-sensitive applications, by configuring the ns-2 CBR (Constant Bit Rate) agent above the TCP(α, β) protocols; we simulate a playback-enabled application with data rate 1Mbps. The bottleneck link bandwidth satisfies the condition

```
1Mbps * N = bw_bottleneck
```

in order to allow for provisioning *just enough* bandwidth to all flows.

Finally, we compare the TCP-friendliness of TCP(α, β) protocols by flows of protocol *pairs* competing for the channel's bandwidth so that their "friendliness" can be adequately demonstrated. One protocol is the TCP(α, β) under study, where (α, β) $\neq$ (1, 0.5); the other is TCP(1, 0.5), the standard TCP.

4.2 Performance Metrics

In static environment, 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. Similarly, we define *Aggregated Protocol Goodput*, as the goodput sum of all the flows that correspond to a particular protocol. The metric is used in protocol-pair tests to enable comparison of protocol *friendliness*. Fairness is measured by the *Goodput Fairness Index*, derived from the formula given in [2].

The *Allotted System Goodput (ASG)*, is defined as the system goodput within one second, and is used to capture the particularity of protocol behavior in dynamic environments. Similarly, the *Allotted Fairness* is defined as the corresponding fairness within one second.

In experiments with real time applications, the application attempts to read and consume up to 125KB every second, (assuming the playback buffer is exactly 125 KB). Because of the sending window fluctuation and the transmission gaps of TCP(α, β), there are instances when the data is unavailable to the application. The percentage of application's successful attempts to read x% of 125KB data from the playback buffer, namely *x% Application Success Percentage*, is used to measure the protocol's smoothness and real-time performance:

```
Application Success Percentage = 100 *
(Success / Attempts) %
```

where, *Success* is defined as:

```
Success:(Allotted Goodput / Targeted
Receiving Rate) > x%
```

In our experimental configuration, Targeted Receiving Rate is 1Mbps. From another perspective, the metric *x% Application Success Percentage* captures the *number* of discrete time slots when the flow achieves at least x% of 1 Mbps data receiving rate.

5. Results and Discussion

5.1 Static Environments

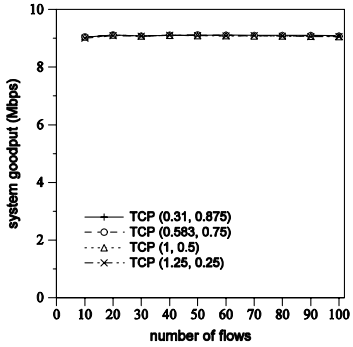

Figure 2. Goodput over wired network (10Mbps bottleneck link)

The experiments conducted on wired networks show that when the fair-share for each flow is large, a smooth

downward adjustment contributes to exploiting the system goodput better. The number of flows ranges from 10 to 100 in the experiments, and the system goodput is measured on 10Mbps and 100Mbps bottleneck, as shown in Figures 2 and 3, respectively. Although the results on 10Mbps links comply with the projection of TCP-friendly equations, the result with 100Mbps bottleneck are supportive to our 3rd hypothesis of section 3 although the arguments made there may not constitute the exclusive justification.

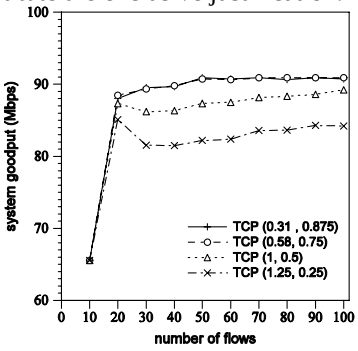

Figure 3. Goodput over wired network (100Mbps bottleneck link)

In contrast to the results with wired networks, the protocols' goodput performance over heterogeneous (wired/wireless) networks highlight the weakness of the high β choice. Results with 100 Mbps bottleneck link are depicted in figures 4. When the error rate is low, the smooth TCP attain higher goodput, and the comparative protocol performance is not far from the previous scenario. With random transient errors increasing from 0 to 0.05 PER on the wireless link, smooth TCP's goodput performance degrades faster and responsive TCP outperforms the smooth one. This is justified by our 2nd observation in section 3. Here the choice of β doesn't make much difference, unlike the high α value of TCP(1.25, 0.25) which permits a more aggressive behavior. Note that the high wireless error rate is different from the high congestion, although in both cases the window size is reduced to a small value due to high packet dropping rate. In the former case, the bandwidth is available; in the latter case, the available bandwidth is low and the aggressiveness due to the high α value doesn't improve the system goodput, as shown in Figure 2.

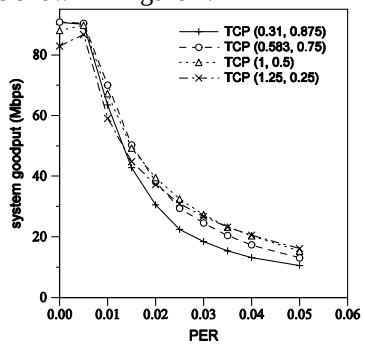

Figure 4. Goodput over heterogeneous network (100 flows, 100Mbps bottleneck, 10Mbps wireless link)

The next scenario presented here intends to provide a framework for characterizing protocol aggressiveness when bandwidth becomes available rapidly in heterogeneous networks. Every 5 seconds, the 10Mbps wireless links are interrupted by a handoff, during which all transmitted packets were lost and the channel's bandwidth was becoming available immediately afterwards. The length of the handoff period is exponentially distributed, with a mean of 1 second. Figure 5 plots the allotted goodput of one flow. Since bandwidth becomes available immediately after the handoff, a high sending rate reflects a desirable behavior. Since the handoff period is long enough, all protocols will reduce their window size and *ssthresh* to 2. After the handoff period is over, a responsive TCP recovers faster and attains smoother rates.

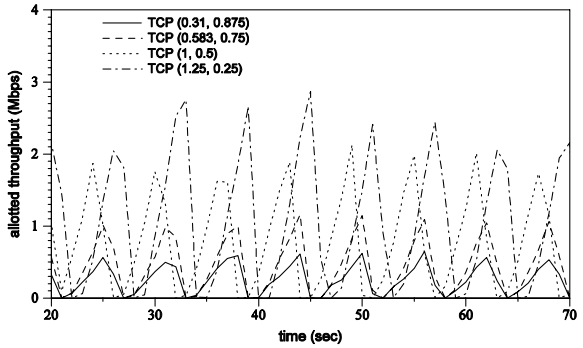

Figure 5. Allotted throughput with 1 sec. handoff (1 flow)

5.2. Dynamic Environments

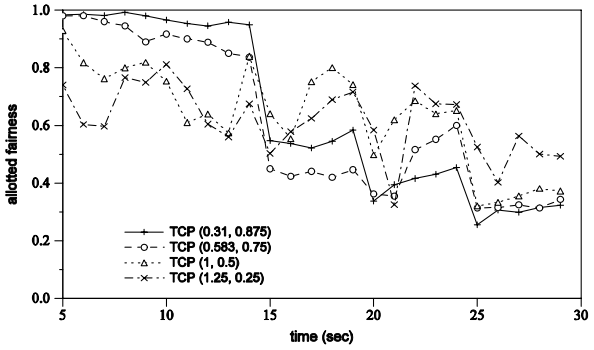

Figure 6. Allotted fairness with increasing number of flows (10Mbps bottleneck link)

Our observations in section 3 call for a comparison of TCP(α,β) protocols, with increasing or decreasing number of flows. Our first experiment is conducted over a wired network with 100Mbps bottleneck link. The number of flows *N* increases with time as follows:

$$N = \begin{cases} 12 & (0 \le t \le 15\text{sec}) \\ 25 & (15 \le t \le 20\text{sec}) \\ 50 & (20 \le t \le 25\text{sec}) \\ 100 & (25 \le t \le 30\text{sec}) \end{cases}$$

That is, *N* doubles every 5 seconds after a 15-second period. Note that the metrics now are allotted fairness, as shown in Figure 6. [9] claims that generating smoother traffic improves allotted fairness, as confirmed in the time period from t_0=0 to t_1=15 in Figure 6. However, when new flows join after 15 seconds, fairness displays a dependency on the sending rate of existing and incoming flows. In this context, the responsive TCP achieves better fairness. This result is in accord to our 1st hypothesis in section 3. When new flows come in, fairness drops for all protocols. However, the fairness of TCP(1.25, 0.25) recovers faster; after a 20-second period the responsive TCP displays the highest fairness, as it was anticipated by our 1st hypothesis.

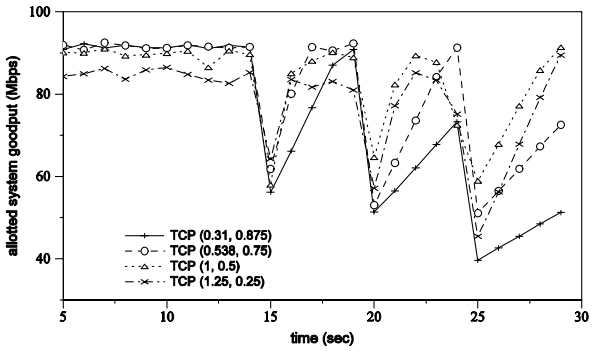

Figure 7. Allotted goodput with decreasing number of flows (100Mbps bottleneck link)

In our next experiment, the number of flows gradually decreases:

$$N = \begin{cases} 100 & (0 \le t \le 15\text{sec}) \\ 50 & (15 \le t \le 20\text{sec}) \\ 25 & (20 \le t \le 25\text{sec}) \\ 12 & (25 \le t \le 30\text{sec}) \end{cases}$$

That is, after a 15-second period, half of the flows complete their task and leave the channel every 5 seconds. The results of allotted system goodput are shown in Figure 7. From t_0=0 to t_1=15, when the number of flows is fixed, the higher the β, the higher the allotted goodput. When flows start leaving the channel (after 15 seconds), the available bandwidth gradually increases; the goodput initially decreases for all protocols since the resource consumption suddenly drops. Obviously aggressiveness is now a desirable behavior and it is not surprising that TCP(0.31, 0.875)'s achieves the lowest performance; this result is justified by our 3rd observation in section 3. Due to the tradeoff of α and β, TCP(1, 0.5)'s goodput is the highest, although TCP(1.25, 0.25)'s recovery speed is the fastest after the step decrease of participating flows. The results in Figure 7 are also justified by our 1st observation: the smaller the number of flows, the longer it takes smooth TCP to exploit the available bandwidth when the fair share step-increases.

5.3 Performance of Real-time Applications

The real-time performance comparison of the protocols is shown in Figure 8. We simulated 100 flows over a 100 Mbps bottleneck, with the wireless error rate varying from 0.0 to 0.4 in experiments. Note that the metrics now are 70% application success percentage. When the error rate is low, smooth TCP outperforms responsive TCP, as anticipated by the design goals of TCP-friendly protocols. However, when the error rate becomes dense the application success percentage of TCP(0.31, 0.875) degrades sharply. The reason behind this behavior can be found in our 2nd observation in section 3: when the error rate is high, α is the dominant factor. Then, the throughput smoothness, which affects the performance of real-time applications, is not determined by how slowly the sender reduces it's sending rate, but rather by how fast it can recover from the loss and restore an appropriate sending rate.

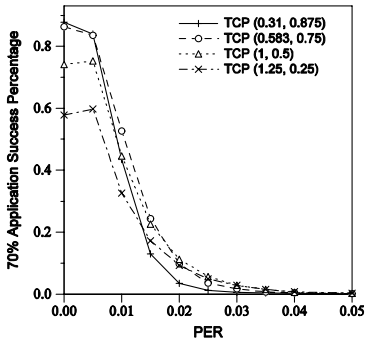

Figure 8. 70% application success percentage (100 flows, 100Mbps bottleneck, 10Mbps wireless link)

5.4 Friendliness

TCP-friendliness tests were conducted with a number of participating flows ranging from 10 to a 100. Flows are divided into two groups per experiment. Half of the flows are instances of one protocol: TCP(0.31, 0.875), TCP(0.583, 0.75) or TCP(1.25, 0.25). The other half are standard TCP flows and serve as the reference for comparison. Ideally, each group of flows should consume exactly half of the bottleneck link capacity. A group that exceeds its fair-share at the expense of the other group's aggregated goodput would mean that the specific protocol is too aggressive. From the results shown in Figures 9, 10 and 11, we observe that with 10 Mbps bottleneck and a large number of flows, smooth TCP appears to be too conservative, allowing the standard TCP to consume bandwidth more aggressively, exceeding its fair share. This result appears initially conflicting with the results of Figure 3 where the smooth TCP achieved higher goodput. However, in this experiment the flows are heterogeneous. It appears that here responsiveness dominates the

consumption of bandwidth, as it is presented in the 2nd hypothesis.

The experiments on the 100Mbps bottleneck (not presented here due to the space limitation) demonstrate, however, with large bandwidth and consequently a relatively larger fair share, TCP(0.31, 0.875), TCP(0.583, 0.75) and TCP(1.25, 0.25) compete fairly with standard TCP.

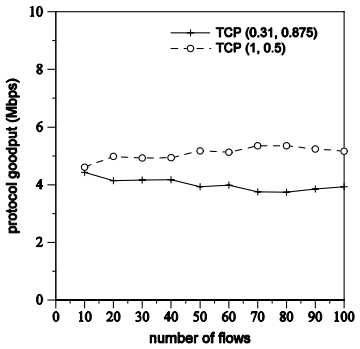

Figure 9. TCP (0.31, 0.875) competing with TCP (1, 0.5) flows (10Mbps bottleneck)

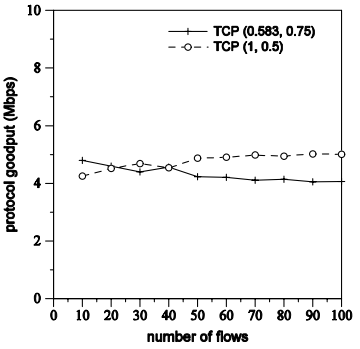

Figure 10. TCP (0.583, 0.75) competing with TCP (1, 0.5) flows (10Mbps bottleneck)

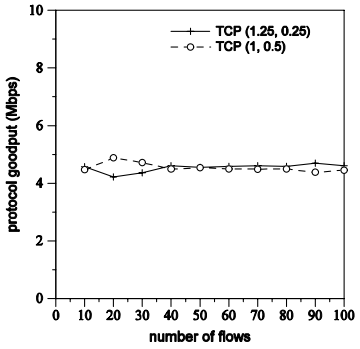

Figure 11. TCP (1.25, 0.25) competing with TCP (1, 0.5) flows (10Mbps bottleneck)

6. Conclusions and future work

We discussed the interrelation of responsiveness and smoothness in the context of the tradeoff of the additive increase rate and the multiplicative decrease ratio of TCP windows. We observed that there are occasions where high responsiveness is not balanced by improved smoothness, and vice versa. Such occasions may arise from the requirements of the applications, the level of network contention, the underlying network characteristics, or the system dynamics.

By and large, targeting responsiveness by using an aggressive additive increase or targeting smoothness by using a modest reduction through multiplicative decrease, is not the appropriate strategy. We conclude that equation-based adjustments are certainly a powerful mechanism for TCP-friendly congestion control, but it can guarantee neither efficiency nor friendliness on its own, in the context of heterogeneous networks. A supportive mechanism for error classification and bandwidth detection may complement the equation-based adjustments and produce positive dynamics. Our future work is scheduled for this target.

7. 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, “Congestion Control Principles”, RFC 2914, September 2000.

[4] S. Floyd, M. Handley and J. Padhye, “A Comparison of Equation-based and AIMD Congestion Control”, May 2000. URL: http://www.aciri.org/tfrc/.

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

[6] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation", ACM SIGCOMM 1998, August 1998.

[7] V. Tsaoussidis and I. Matta, “Open issues on TCP for Mobile Computing”, Journal of Wireless Communications and Mobile Computing, Wiley Academic Publishers, Issue 2, Vol. 2, February 2002.

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

[9] Y.R. Yang, M.S. Kim and S.S. Lam, “Transient Behaviors of TCP-friendly Congestion Control Protocols”, Proceedings of IEEE INFOCOM 2001, April 2001.