The Dynamics of Responsiveness and Smoothness in Heterogeneous Networks

Vassilis Tsaoussidis, *Senior Member, IEEE,* and Chi Zhang, *Member, IEEE*

***Abstract*—Additive increase/multiplicative decrease-based protocols, including transmission control protocol (TCP), TCP-friendly, and a new generation of rate-based protocols, attempt to control the tradeoff of responsiveness and smoothness. Traditionally, smoothness has not been a main concern since it does not impact the performance of regular Internet applications such as the Web, FTP, or e-mail. However, multimedia-driven protocols attempt to favor smoothness at the cost of responsiveness. In general, smoothness and responsiveness constitute a tradeoff; however, we uncover undesirable dynamics of the protocols in the context of wireless/mobile networks with high-error rate or frequent handoffs: low responsiveness is not counterbalanced by gains in smoothness, but instead, produces a conservative behavior that degrades protocol performance with both delay-tolerant and -sensitive applications. Based on our observations, as well as on further analysis of the impact of the bottleneck queue on channel utilization, we seek an alternative strategy for smooth window adjustments. We introduce a new parameter γ, which implements a congestion avoidance tactic and reaches better smoothness without damaging responsiveness.**

***Index Terms*—Congestion control, fairness, transmission control protocol (TCP)-friendly protocols.**

I. Introduction

THE CONGESTION control algorithms of Internet transport protocols (e.g., [1]) are based on a somewhat "blind" increase/decrease window adjustment, which exploits dynamically the availability of bandwidth, avoids persistent congestion, and achieves fair utilization. For example, the window adjustment is modeled on the additive increase/multiplicative decrease (AIMD) algorithm, which allows for fair resource allocation and efficient resource utilization [3]. AIMD is also becoming the core algorithm of all transport protocols that support congestion control functions [4]. According to AIMD, all senders keep increasing their transmission rate additively (i.e., the congestion window W increases by α packets per round-trip time (RTT) if there is no packet loss), until the network becomes congested. At this stage, a multiplicative decrease ratio is used to avoid a congestive collapse (i.e., the congestion window W decreases to βW upon congestion). The standard TCP itself uses an increase rate 1 packet per window of acknowledgments ($\alpha = 1$) and a multiplicative decrease ratio of 1/2 of the current window ($\beta = 0.5$).

Manuscript received January 29, 2004; revised December 8, 2004.

V. Tsaoussidis is with the Department of Electrical and Computer Engineering, Demokritos University, Xanthi 67100, Greece (e-mail: vtsaousi@ee.duth.gr).

C. Zhang is with the School of Computer Science, Florida International University, Miami, FL 33199 USA (e-mail: czhang@cs.fiu.edu).

Digital Object Identifier 10.1109/JSAC.2005.845627

While TCP congestion control is appropriate for bulk data transfer (i.e., traditional Internet applications such as the Web, FTP, or e-mail.), some real-time applications such as media-streaming applications suffer severe consequences (e.g., data-rate oscillations and even transmission gaps) due to the standard multiplicative window decrease by a factor of 2 upon congestion [14]. This is due to the inherent characteristic of increase/decrease algorithms to produce sawtooth transmission behavior. Therefore, transmission smoothness becomes an important concern. Departing from the results of Chiu and Jain [3], researchers have proposed some new variations of the traditional increase/decrease schema to control congestion with small oscillations, driven by the requirements of multimedia applications for smooth patterns of data transmission. To achieve rate adjustments, window decrease ratio β is increased during congestion. Therefore, the magnitude of sawtooth oscillation is reduced. The analysis of [3] and later the proofs of [8] made clear however, that favoring *smoothness* inevitably will damage a system's convergence time to equilibrium (i.e., lesser *responsiveness*).

The dominance of TCP in the Internet gave ground to another limitation, more artificial rather than natural: a protocol that utilizes more bandwidth than TCP is not considered a TCP-friendly protocol. Practically, the TCP-friendly approach not only increases the multiplicative decrease ratio β during congestion, but also reduces the additive increase rate α to pay back the credit. Based on the aforementioned interrelation of throughput, smoothness, and responsiveness, an equation was invented in [14] to allow a throughput-oriented TCP-friendly modulation of the additive increase factor α and the multiplicative decrease factor β. TCP-friendly $\mathrm{TCP}(\alpha, \beta)$ protocols parameterize the congestion window increase value α and decrease ratio β, and tradeoff responsiveness (low α) for smoothness (high β). They also 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 protocol behavior under various network and traffic conditions.

The aforementioned approach, to restrict protocol behavior within the confines of TCPs resource utilization, and meanwhile, to keep the flexibility to adjust smoothness and responsiveness according to application requirements, sounds like a straightforward and clear procedure. However, the calculation of TCPs throughput entails a number of assumptions, which may not necessarily hold in general, but certainly do not hold specifically in heterogeneous (wired/wireless) networks or under dynamic traffic loads. That makes matters much more complicated than they appear. We attempt to: 1) explore this

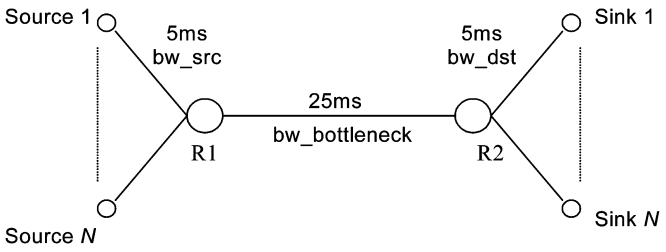

Fig. 1. Network topology.

complexity; 2) present some anomalies through experiments; and 3) provide a justified proposal for overcoming the present anomalies. In summary, we investigate whether the damage in responsiveness is indeed counterbalanced by gains in smoothness also when the prescribed patterns of network traffic and the error characteristics of traditional networks do not apply. Our study initially involves the following:

- heterogeneous networks with both wired and wireless components;
- three classes of TCP-friendly $\mathrm{TCP}(\alpha,\beta)$ protocols: 1) standard TCP(1, 1/2); 3) responsive TCP is $\mathrm{TCP}(\alpha,\beta)$ with *relatively* low β value and high α value; and 3) smooth TCP is $\mathrm{TCP}(\alpha,\beta)$ with *relatively* high β value and low α value;
- static and dynamic[1] environments;
- delay-sensitive and -tolerant applications.

Based on the analysis of the assumptions of equation-based congestion control and on our experimental data, we arrive at the conclusion that TCP-friendly protocols, which 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 analyze further the potential of a new parameter γ, in the role of congestion avoidance, prior to congestion control.

We combine an analytical and experimental study. First, we describe the experimental methodology and the variety of the performance metrics used in the paper. In Section III, we present a series of observations, which we justify with selected results. In Section IV, we present a proposal for better handling of the responsiveness/smoothness tradeoff. In Section V, we conclude our findings.

II. Experimental Methodology

We have implemented our testing plan on the ns-2 network simulator [9]. The network topology used as a testbed is shown in Fig. 1. The number of flows (or the number of source-sink pairs) N, varied from experiment to experiment. 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 reconfigured for the different scenarios. In most cases, however, $bw\_bottleneck = bw\_src = bw\_dst$ unless it is pointed out explicitly otherwise. The buffer sizes at routers are set to the delay-bandwidth product. We have also simulated competing flows with highly diverse RTTs, and the results are given in Section III. 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). We have also simulated random error rates with a two-state Markov model, the results and the observations are similar. Furthermore, to test protocol behaviors with mobile handoffs, we used a two-state continuous Markov chain. Each state has exponentially-distributed sojourn time. The "Off" state has a long average sojourn time with a zero drop rate, while the "Off" state has a short average sojourn time, during which all transmitted packets were lost on the wireless access links.

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 span across a spectrum of smoothness and responsiveness and satisfy the TCP-friendly (3) shown in Section III. 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 made use 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 also evaluated the protocols' performance with delay-sensitive applications, by configuring the ns-2 constant-bit rate (CBR) agent above the $\mathrm{TCP}(\alpha,\beta)$ protocols; we simulate a playback-enabled application with data rate of 1 Mb/s. The bottleneck link bandwidth satisfies the condition

$$1\ \text{Mb/s}^{*}N = bw\_bottleneck$$

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

A. Performance Metrics

In a 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. The *Goodput* for each flow is defined as

$$\text{Goodput} = \frac{\text{Original\_Data}}{\text{Connection\_Time}}$$

where Original_Data is the number of bytes delivered to the high-level protocol at the receiver (i.e., excluding retransmitted packets) and Connection_Time is the amount of time required for the data delivery. The *allotted system goodput* (ASG), is defined as the system goodput within a short sampling period (usually at a time scale of several RTTs), and is used to capture the particularity of protocol behavior over time, in dynamic environments.

The protocol efficiency can be studied from another perspective. Overhead is used as a metric to realize the protocol transmission effort to complete reliable data delivery

$$\text{Overhead} = \frac{(\text{Bytes\_Sent-Original\_Bytes})}{\text{Bytes\_Sent}}.$$

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

Bytes_Sent is the total bytes transmitted by TCP senders, while Original_Bytes is the number of bytes delivered to the higher level protocol by all receivers, excluding retransmitted packets and TCP header bytes. This metric captures the portion of consumed bandwidth, or the percentage of the transmission energy (a scarce resource in mobile computing), that is wasted on packet retransmissions and protocol header overhead.

Long-term fairness is measured by the *fairness index*, defined by [3]

$$\text{Fairness Index} = \frac{\left(\sum_{i=1}^{n} \text{throughput}_i\right)^2}{n\sum_{i=1}^{n} \text{throughput}_i^2}$$

where throughput_i is the throughput of the ith flow, measured at a time scale of connection time. Similarly, the *allotted fairness* is defined as the corresponding fairness within a short sampling period, and is used to capture the system fairness over time under dynamic traffic loads. Allotted throughput is also used to compute the short-term fairness of a single run, derived from the traditional fairness index

$$\text{Short Term Fairness} = E_t\left\{\frac{\left(\sum_{i=1}^{n} \text{throughput}(t)_i\right)^2}{n\sum_{i=1}^{n} \text{throughput}(t)_i^2}\right\}$$

where $E_t\{\cdot\}$ denotes the computation of the mean along the time, and $\text{throughput}_i(t)$ is allotted throughput of the ith flow around time t.

To investigate the performance smoothness observed by end users, allotted throughput of individual flow $\text{throughput}_i(t)$ is used to observe the performance fluctuations. Following the metric in [13], we use *coefficient of variation* (CoV) to gauge the throughput smoothness experienced by flow i

$$\text{CoV}_i = \frac{\sqrt{E_t\left\{\text{throughput}_i^2(t)\right\} - E_t\{\text{throughput}_i(t)\}^2}}{E_t\{\text{throughput}_i(t)\}}.$$

For a system with multiple flows, the system *CoV* is the average of CoVs of all flows. As argued in [13], short-term fairness is closely related to CoV and smoothness.

Allotted throughput and, hence, CoV and short-term fairness, are often measured 15 s after the simulation starts, in order to just capture the system performance after it enters equilibrium state.

In experiments with real-time CBR applications with 1 Mb/s data rate, the application attempts to read and consume up to 125 kB every second, (assuming the playback buffer is exactly 125 kB). Because of the sending window fluctuation and the transmission gaps of $\text{TCP}(\alpha,\beta)$, there are instances when the data is unavailable to the application. The percentage of application's successful attempts to read $x\%$ of 125 kB data from the playback buffer, namely, $x\%$ *application success percentage*, is used to measure the protocol's smoothness and real-time performance

$$\text{Application Success Percentage} = 100^* \left[\sum_{j=1}^{T}\sum_{i=1}^{n} \frac{\text{Success}(i,j)}{(nT)}\right]\%$$

where n is the total number of flows; T is the connection time (in seconds); $\text{Success}(i,j)$ is defined as

$$\text{Success} = \begin{cases} 1, & \text{if } \dfrac{\text{Alloted Goodput}(i,j)}{\text{Targeted Receiving Rate}} > x\% \\ 0, & \text{otherwise} \end{cases}$$

where Allotted $\text{Goodput}(i,j)$ is the goodput of the ith flow within the jth second. In our experimental configuration, targeted receiving rate is 1 Mb/s. 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 Mb/s data receiving rate.

III. Observations on The Dynamics of Responsiveness and Smoothness

A throughput equation for standard TCP is first introduced in [10]. GAIMD [14] extends the equation to include parameters α and β, as shown in (1) at the bottom of the page 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 $\text{TCP}(\alpha=1,\beta=0.5)$, which means that (2), which is derived from (1) (see [14]) could provide a rough guide to achieve friendliness

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

Authors of [14] derive from (1) and (2) a simple relationship for α and β

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

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 $\alpha = 0.31$.

We present next some observations on the dynamics of TCP responsiveness and smoothness, based on theoretical analyses. We further use experimental results to verify our observations and analyses.

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

The observations of the window dynamics and event losses are frequently assumed within a time period of a *congestion*

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

epoch [5], which reflects the *uninterrupted growing lifetime of congestion window*. More precisely, a congestion epoch begins with $\beta \mathtt{W}$ packets, increased by α packets per RTT and reaching a congestion window of $\mathtt{W}$ packets, when a packet is dropped. The congestion window is then decreased to $\beta \mathtt{W}$. Hence, a congestion epoch involves

$$\mathtt{n} = (1-\beta) * \frac{\mathtt{W}}{\alpha + 1 \ \mathrm{RTT}}. \tag{4}$$

Assuming that the capacity of the bottleneck link is $\mathtt{B}$ packets/s and the number of active flows going through the bottleneck router is $\mathtt{N}$, and assuming a control system as in [3], we further calculate that:

$$\mathtt{W} = \mathtt{B} * \frac{\mathrm{RTT}}{\mathtt{N}}. \tag{5}$$

Equation (1) is modeled by calculating the average throughput over a congestion epoch, which is associated with several RTTs. Since (1) gives the *steady-state* TCP throughput, in a dynamic network where conditions change 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. Finally, the propagation delay has a direct impact on the time required for $\mathrm{TCP}(\alpha, \beta)$ to reach a full-window size. Practically (and deterministically) this means that for a window of 64 kB and an RTT of 100 ms TCP(1, 1/2) needs at least 3.2 s to reach the maximum window size.

Observation 1 reveals inherent characteristics of equation-based TCP-friendly protocols, and is the ultimate reason for some phenomena and observations described below. For example, since TCP-friendly protocols are based on an equation that gives the steady-state TCP throughput, in a dynamic environment where traffic load changes rapidly, friendliness might not be accomplished (Observation 3).

Observation 2: Throughput is not a direct function of the sending window. Smooth protocols may gain throughput when delay-bandwidth product is large, and reduce packet drops when contention is high.

Equation (2) indicates that the protocols will always achieve about the same throughput as the standard TCP. However, the assumption of (1)–(3) that the system throughput increases in proportion to the sending window, is inaccurate. We now give an in-depth analysis.

Consider a simple dumbbell network topology, as shown in Fig. 1, where n TCP flows share a bottleneck link with capacity of bw, and the round-trip propagation delay is RTT_0. We define the *aggregated* congestion window size at time t as

$$\mathrm{cwnd}(t) = \sum_{i=1}^{n} \mathrm{cwnd}_i(t) \tag{6}$$

where $\mathrm{cwnd}_i(t)$ is the window size of the ith flow. Consequently, the system throughput at time t can be given by the following equation:

$$\begin{aligned} \text{throughput}(t) &= \frac{\mathrm{cwnd}(t)}{\mathrm{RTT}(t)} \\ &= \frac{\mathrm{cwnd}(t)}{\mathrm{RTT}_0 + \mathrm{qdelay}(t)} \end{aligned} \tag{7}$$

where $\mathrm{qdelay}(t)$ is the queueing delay at the bottleneck router R_1. As can be seen from (7), the throughput is not only a function of the congestion window, but also a function of the queueing delay, which was not incorporated into the analyses in [3], [10], and [14].

Assume all flows are in the additive increase stage. First, consider the case where $\mathrm{cwnd}(t)$ is below the point knee [3]

$$\mathrm{cwnd}_{\mathrm{knee}} = \mathrm{RTT}_0 \cdot bw. \tag{8}$$

Then, there is no *steady* queue buildup[2] in R_1 (i.e., $\mathrm{RTT}(t) = \mathrm{RTT}_0$), and according to (7), the throughput grows in proportion to cwnd. The bottleneck capacity is not fully utilized until cwnd increases to $\mathrm{cwnd}_{\mathrm{knee}}$.

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

$$\mathrm{cwnd}(t) = \mathrm{cwnd}_{\mathrm{knee}} + \Delta w(t) (\Delta w(t) > 0). \tag{9}$$

Since the bottleneck link can transmit at most $\mathrm{cwnd}_{\mathrm{knee}}$ packets in one RTT_0 [(see (8)], $\Delta w(t)$ packets will linger in the queue. Hence, the steady queueing delay at the bottleneck will be

$$\mathrm{qdelay}(t) = \frac{\Delta w(t)}{bw}. \tag{10}$$

Intuitively, the system throughput is bounded by the physical capacity bw, in spite of the increase of $\mathrm{cwnd}(t)$ beyond the knee, because $\mathrm{qdelay}(t)$ in the denominator of (7) grows as well. This is confirmed by the following computation:

$$\begin{aligned} \text{throughput}(t) &= \frac{\mathrm{cwnd}_{\mathrm{knee}} + \Delta w(t)}{\mathrm{RTT}_0 + \mathrm{qdelay}(t)} \\ &= \frac{\mathrm{RTT}_0 \cdot bw + \mathrm{qdelay}(t) \cdot bw}{\mathrm{RTT}_0 + \mathrm{qdelay}(t)} \\ &= bw. \end{aligned} \tag{11}$$

The system dynamics can be continuously described by (9)–(11), until the queue length $\Delta w(t)$ reaches the maximum buffer size, i.e., when cwnd touches the point cliff[3]

$$\mathrm{cwnd}_{\mathrm{cliff}} = (\mathrm{RTT}_0 + \max \mathrm{qdelay}) \cdot bw. \tag{12}$$

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

[2]There could be *temporary* queue buildup in this scenario, due to the traffic burstiness. This is neglected to simplify our analysis.

[3]The intuitive concept of knee and cliff was first introduced in [3]. Here, we give an analytical expression.

It is important to note that the system throughput does not always increases in proportion to the sending window. The network capacity can be fully utilized when the window size reaches the knee. The computation of (11) demonstrates that increasing cwnd beyond the knee does not enhance further the system throughput, but only results in increasing queueing delay. Throughput may suddenly drop when the window size increases beyond the point $cliff$, where the queue overflows and packets may be dropped. However, our analysis also indicates that some queue buildup is inevitable, in order to provide 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 [3], it does not necessarily mean that the throughput will be sacrificed, as long as the system operates in the zone between the knee and the cliff, where throughput is maximal… In order to prevent the system from operating below the knee where bandwidth is underutilized, and meanwhile maintain adequate AIMD oscillation (which affects the speed to converge to fairness [8]), an efficient window decreasing ratio could be

$$\begin{aligned} \beta &= \frac{\mathrm{cwnd}_{\mathrm{knee}}}{\mathrm{cwnd}_{\mathrm{cliff}}} = \frac{1}{1+k} \\ \text{where } k &= \frac{\max \mathrm{qdelay}}{\mathrm{RTT}_0} = \frac{\max \mathrm{qdelay} \cdot bw}{\mathrm{RTT}_0 \cdot bw} \\ &= \frac{\text{Buffer Size}}{\mathrm{RTT}_0 \cdot bw}. \end{aligned} \tag{13}$$

When the bottleneck buffer size equals to delay-bandwidth product, $k = 1$ and $\beta = 0.5$.

We observe that in a homogeneous environment with high bandwidth and small buffer size, a responsive TCP is more 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* (2)'s projection. Although this may sound as a negative property in a static environment where steady-state behaviors dominate performances, it is indeed positive when the number of competing flows frequently changes and responsiveness becomes crucial. We will exploit this situation later on; at this point, we conclude that throughput of smooth protocols is not, as it is claimed, equal to the throughput of standard TCP.

Our simulations without wireless packet drops (i.e., with wired networks or wireless networks with little interferences) confirm the aforementioned statement. The number of flows ranges from 10 to 100 in the experiments, and the system goodput is measured on 10 and 100 Mb/s bottleneck, as shown in Figs. 2 and 3, respectively. Although the results on 10 Mb/s links comply with the projection of TCP-friendly equations, the result with 100 Mb/s bottleneck are supportive to our analysis: smooth TCPs outperform responsive TCPs with static and large share of bandwidth. Note that the system goodput jumps sharply after the number of flows increased from 10 to 20. That is because the maximum window size allowed by standard TCP header is 64 kB and, hence, ten flows cannot consume all of the 100 Mb/s capacity.

The system overhead of the above two simulations are shown in Figs. 4 and 5. Although goodput performance of

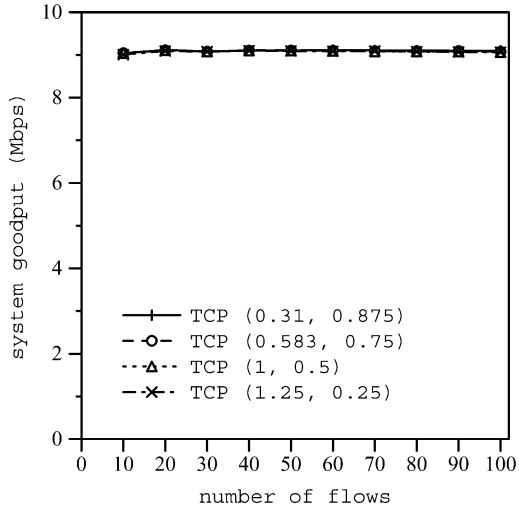

Fig. 2. Goodput over wired network (100 Mb/s bottleneck link).

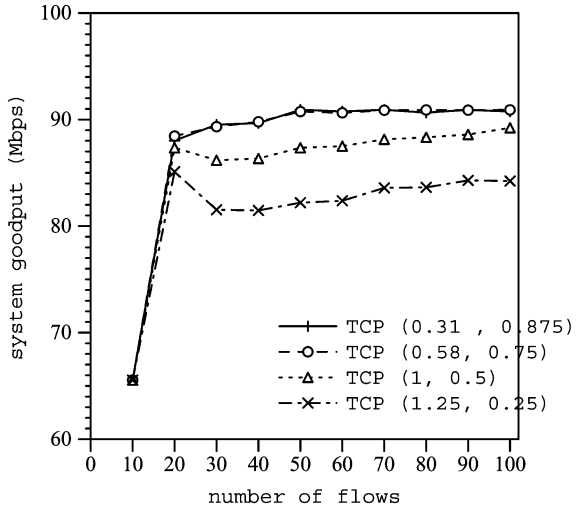

Fig. 3. Goodput over wired network (100 Mb/s bottleneck link).

$\mathrm{TCP}(\alpha, \beta)$ protocols are indistinguishable with 10 Mb/s bottleneck (Fig. 4), the metric overhead reveals that smooth TCP(0.31, 0.875) causes significantly less packet drops due to its moderated window adjustments, especially when the number of flows increases and the contention is high. Cautious window adjustments allow more time for flows to respond to packet drops at the congested bottleneck, before the windows grow further. Since the network is already congested, raising additive increase speeds cannot increase bandwidth utilization, but only cause mores packets drops and, hence, more energy consumption. With 100 Mb/s bottleneck (Fig. 5), low contention extends the congestion epoch and, hence, reduces the packet drops. The difference between smooth TCPs and responsive TCPs is negligible.

The protocol performance for CBR applications without any wireless interference involved is shown in Fig. 6. We simulated a scenario in a wired network. The number of flows ranges from 10 to 100 in the experiments, while the bottleneck bandwidth varies from 10 to 100 Mb/s correspondingly, to guarantee exactly 1 Mb/s bandwidth fair share for each flow. Fig. 6 depicts the successful attempts of the application to read at least 90%

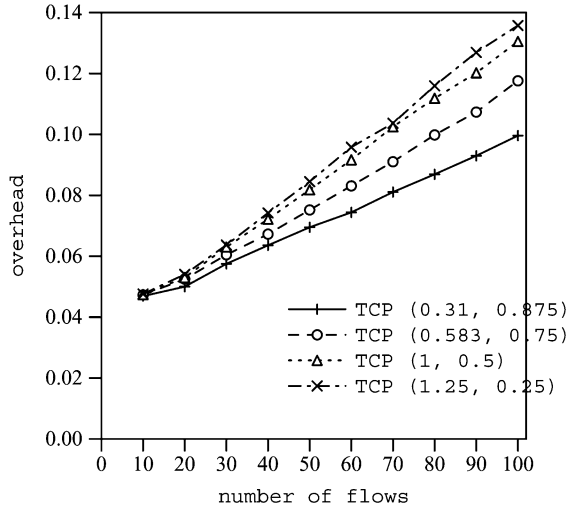

Fig. 4. Overhead over wired network (10 Mb/s bottleneck link).

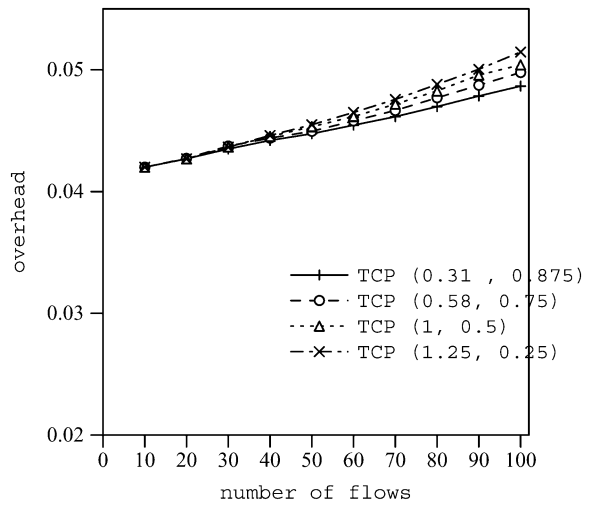

Fig. 5. Overhead over wired network (100 Mb/s bottleneck link).

of the data sent. We can observe that a high β value results in a smooth traffic and, therefore, high application success percentage. As it is anticipated by the design goals of TCP-friendly protocols, the application success percentage of smooth TCPs is significantly higher than responsive TCPs.

Observation 3: With dynamic traffic, smoothness has an negative impact on system fairness when the load increases gradually, while it has a cost on system goodput when the load decreases gradually.

In a dynamic system where contention gradually increases, with smooth TCPs, the gentle window adjustments of existing flows may not guarantee fairness to incoming flows. A conclusion of the control system presented in [3] 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

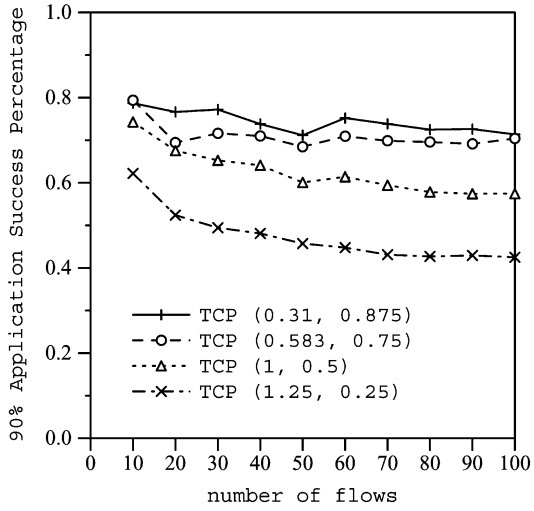

Fig. 6. 90% application success percentage (1 Mb/s fair share at bottleneck and 10 Mb/s access link).

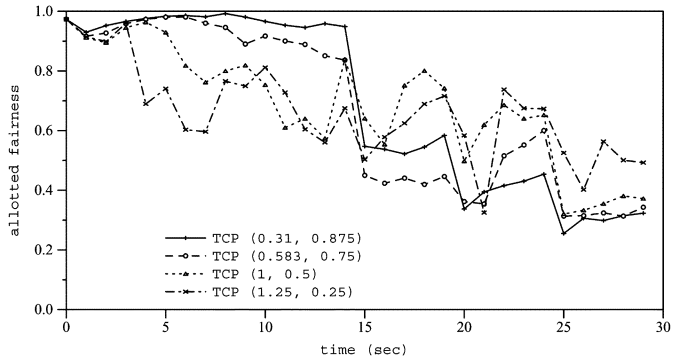

Fig. 7. Allotted fairness with increasing number of flows (10 Mb/s bottleneck).

flows drop their sending rate quickly to make available bandwidth for new flows.

Our first simulation of dynamic traffic is conducted over a wired network with 100-Mb/s bottleneck link. The number of flows N increases with time as follows:

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

That is, N doubles every 5 s after a 15-s period. Note that the metrics now are allotted fairness, as shown in Fig. 7. As it is correctly claimed generally, generating smoother traffic improves allotted fairness; this is confirmed in the time period from $t_0 = 0$ to $t_1 = 15$ in Fig. 7. However, when new flows join after 15 s, fairness displays a dependency on the throughput rate of existing and incoming flows. In this context, the responsive TCP achieves better fairness. When new flows come in, fairness drops for all protocols. However, the fairness of TCP(1.25, 0.25) recovers faster; after a 20-s period the responsive TCP displays the highest fairness, as it was anticipated from our analysis. We can also see in Fig. 7 what we also expected from the analysis in Observation 1. The smaller the number of flows, the longer the congestion epoch is and the longer it takes the smooth TCP to converge to fairness when the fair share decreases.

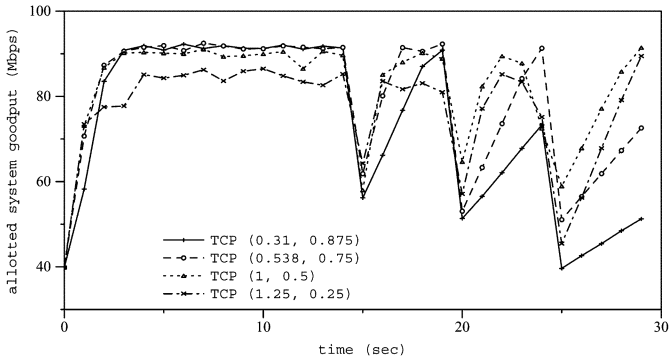

Fig. 8. Allotted goodput with decreasing number of flows (100 Mb/s bottleneck).

As a result, the system becomes increasingly unfair with smooth protocols.

On the other hand, 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, the increasing rate follows the same pattern. Hence, a smooth TCP extends further the time needed to approach the new fair share. Obviously, responsiveness is here too the dominant parameter of efficiency since it reflects the protocol's capability to exploit the available bandwidth.

In our next simulation, the number of flows gradually decreases

$$N = \begin{cases} 100, & (0 \le t \le 15\ \mathrm{s}) \\ 50, & (15 \le t \le 20\ \mathrm{s}) \\ 25, & (20 \le t \le 25\ \mathrm{s}) \\ 12, & (25 \le t \le 30\ \mathrm{s}) \end{cases}.$$

That is, after a 15-s period, half of the flows complete their task and leave the channel every 5 s. The results of allotted system goodput are shown in Fig. 8. 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 s), 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) achieves the lowest performance; this result justifies our projection that smoothness has a negative impact when the load decreases. 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 Fig. 8 are also justified by (4): the smaller the number of flows, the larger the window and, hence, the longer it takes smooth TCP to exploit the available bandwidth when the fair share gradually increases.

Observation 4: In heterogeneous (wired/wireless) networks, low responsiveness is not counterbalanced by gains in smoothness, damaging the overall throughput, and energy efficiency.

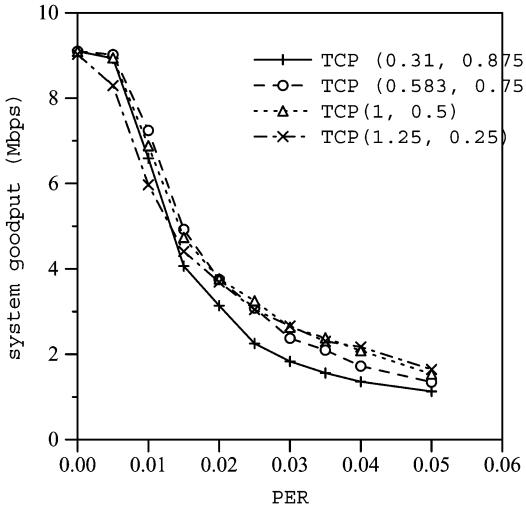

Fig. 9. Goodput over heterogeneous network (ten flows, 10 Mb/s bottleneck).

Likewise, real-time application efficiency degrades when responsiveness is low.

According to (2), TCP(α, β) protocols achieve approximately the same throughput as the standard TCP, since the protocol's aggressiveness (conservativeness) caused by high (low) β is canceled by the conservativeness (aggressiveness) caused by low (high) α. A hidden assumption behind (2) is that when packet drops occur at the end of the congestion epoch, the window decreasing by a factor of $(1 - \beta)$ is applied only once. However, multiple packet drops under high wireless interferences 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 two segments, even with smooth backward adjustments. Assuming that the window starts with two segments at the beginning of the congestion epoch, regardless of β, (4) should be replaced by (14)

$$\mathtt{n} = \frac{(\mathtt{w} - 1)}{\alpha + 1}\ \mathrm{RTTs}. \tag{14}$$

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 case the bandwidth is still available though the packet dropping rate is high.

Our simulations confirmed that the protocols' goodput performance over heterogeneous (wired/wireless) networks highlight the weakness of the high β choice. Results with 10 and 100 Mb/s bottleneck links are depicted in Figs. 9 and 10, respectively. When the error rate is low, smooth TCPs attain higher goodput. 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 ones. Here, the choice of β does not make much difference, while the high α value of TCP(1.25, 0.25) permits a more aggressive behavior.

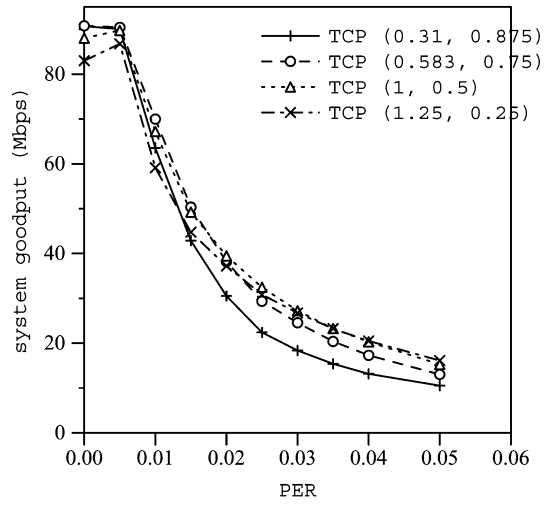

Fig. 10. Goodput over heterogeneous network (100 flows, 100 Mb/s bottleneck, and 100 Mb/s wireless link).

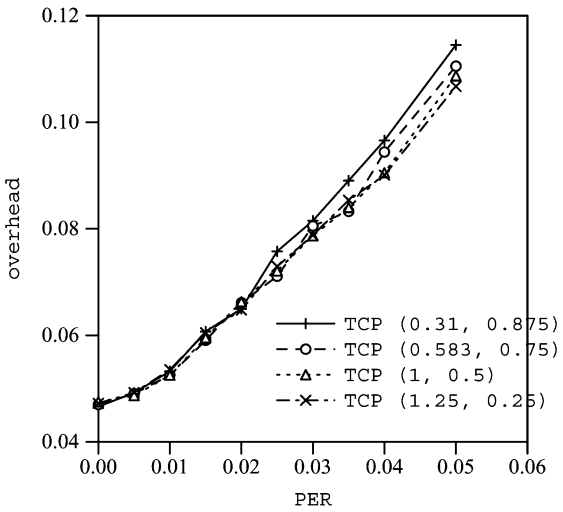

Fig. 11. Goodput over heterogeneous network (ten flows, 10 Mb/s bottleneck).

Note that the high wireless error rate is different from high congestion (see Fig. 2), although in both cases the window size can be reduced to a small value due to high packet dropping rate. In the former case, bandwidth is available; in the latter case, the available bandwidth is low and the aggressiveness due to the high α value does not improve the system goodput.

The system overhead of the above simulations is depicted in Figs. 11 and 12. TCP(0.31, 0.875)'s overhead (hence, the energy consumption) is even slightly higher than other protocols, in contrast to the results in wired scenarios (Figs. 4 and 5). This is consistent with the goodput performance demonstrated in Figs. 9 and 10: Smooth TCP's conservative behavior under high wireless errors diminishes its efficiency in bandwidth utilization; while responsive TCP reduces packet retransmissions by taking the chance to grow, when wireless error is absent.

The next scenario presented here intends to provide a framework for characterizing protocol aggressiveness when bandwidth becomes available rapidly in heterogeneous networks. The 10 Mb/s throughput capacity of the wireless links is interrupted by a handoff every 5 s, while the length of handoff

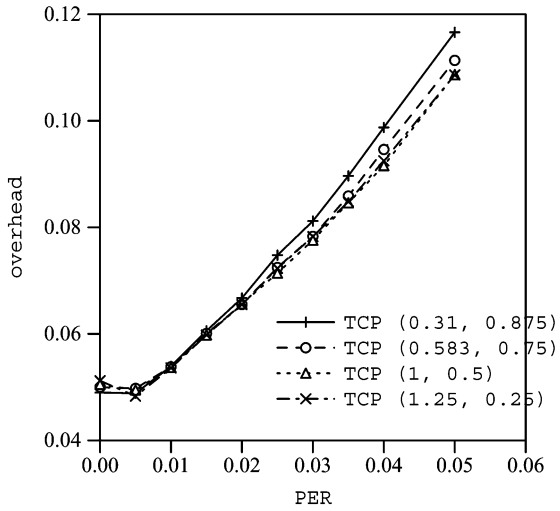

Fig. 12. Overhead over heterogeneous network (100 flows, 100 Mb/s bottleneck, and 10 Mb/s wireless link).

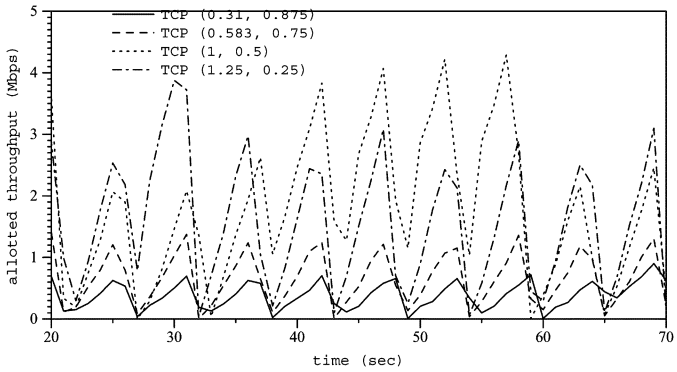

Fig. 13. Allotted throughput with 500 ms handoff (one flow).

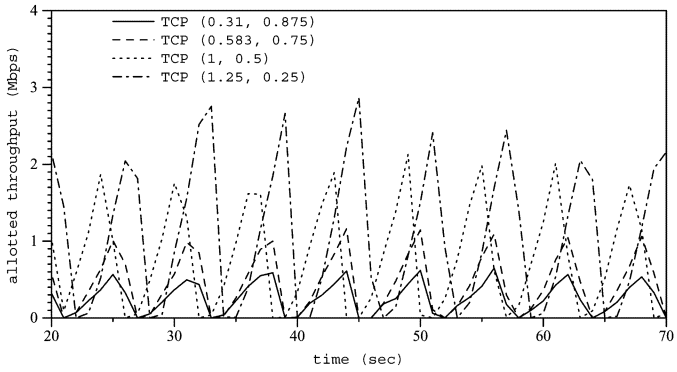

Fig. 14. Allotted throughput with 1 s handoff (one flow).

period is exponentially distributed. Figs. 13 and 14 plot the allotted goodput of a flow with 500 ms and 1 s mean handoff period, respectively. Since bandwidth becomes available immediately after the handoff, a high sending rate reflects a desirable behavior; the protocols need not to adjust the rate due to congestion. If the handoff period is long enough (Fig. 14), 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. However, since the length of the handoff period is exponentially distributed, there are occasions where the duration of the handoff period is short, enabling

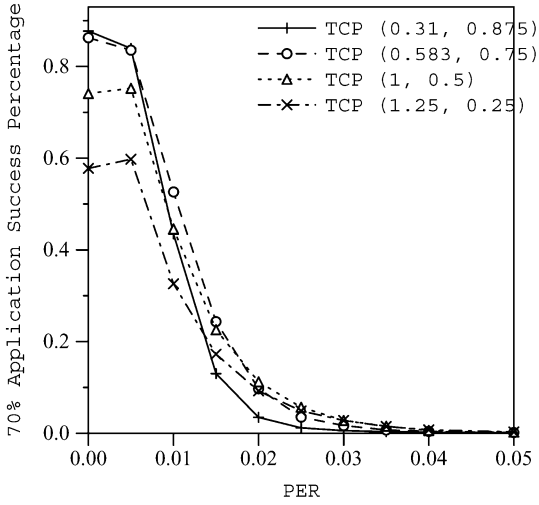

Fig. 15. 70% application success percentage (100 flows, 100 Mb/s bottleneck, and 10 Mb/s wireless link).

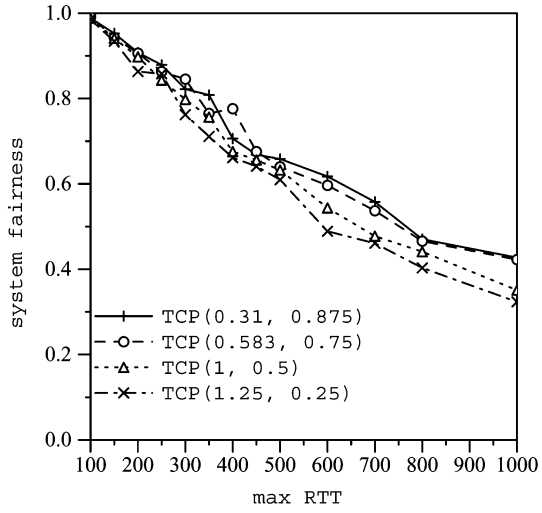

Fig. 16. Fairness with drop-tail gateway (10 Mb/s bottleneck and 20 flows with diverse RFFs).

a fast recovery to be triggered by three duplicate acknowledgments (see the period of time from the 40th s to the 60th s in Fig. 13). In such occasions, TCP (1, 0.5), due to its relatively high β value, attains higher allotted goodput than TCP(1.25, 0.25) does, although the latter's goodput performance, is also relatively high.

The real-time performance comparison of the protocols is shown in Fig. 15. We simulated 100 flows of CBR applications, 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 reasoning behind this behavior is that 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 its sending rate upon packet loss, but rather by how fast it can recover from the loss and restore an appropriate sending rate after wireless interference is over.

Observation 5: In mobile WAN environment with diverse RTTs, smooth TCP alone enhances system fairness, in contrast to the deployment of RED.

TCPs bias against flows with large RTTs is a well-known problem. In the same context, diverse RTTs are inherent characteristics for TCP connections in mobile WAN environment. However, since smooth downward window adjustment enhances the capability of bandwidth consumption for flows experiencing large delay-bandwidth product, as argued by Observation 2, the system unfairness to connections with large RTTs can be slightly canceled by smooth TCP. Therefore, we simulated flows with diverse RTTs sharing a 10 Mb/s bottleneck link. The minimum RTT was fixed at 30 ms, while the maximum flow RTT varied from simulation to simulation. The total number of flows is 20, and the ith flow's RTT is given by the following equation:

$$\mathrm{RTT}_i = \min \mathrm{RTT} + \frac{i^*(\max \mathrm{RTT} - \min \mathrm{RTT})}{(20-1)(\mathrm{ms})}.$$

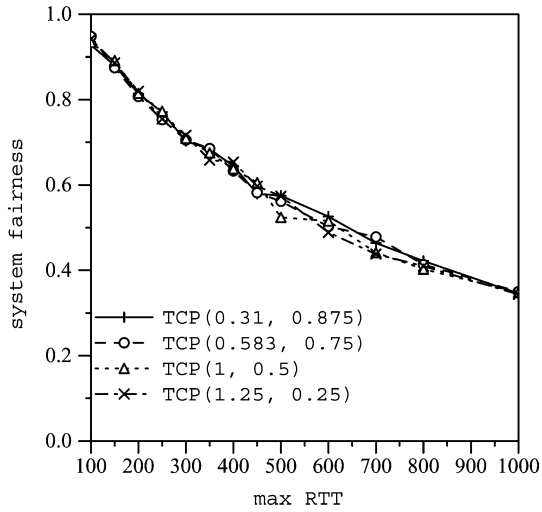

Fig. 17. Fairness with RED gateway (10 Mb/s bottleneck and 20 flows with diverse RTTs).

That is, flows' RTTs are uniformly distributed between min RTT and max RTT. The buffer sizes of routers are set to the product of the min RTT and the bandwidth. The results shown in Fig. 16 corroborate our analysis. As the diversity of RTTs increases, the fairness decreases for all protocols. However, due to the goodput improvement for flows with large RTTs, smooth TCP slightly enhances the system fairness as well.

We repeated the simulation, after turning on RED at [7] the bottleneck router. The results confirm our early findings [15] that RED can reduce the system throughput with large delay-bandwidth product and small buffer sizes. More specifically, RED reduces the throughput of smooth TCPs experiencing long propagation delays. Therefore, the fairness of smooth TCPs decreases down to the level of responsive TCPs (Fig. 17). That is, RED worsens TCP's system unfairness in the context of diverse RTTs.

IV. A Proposal

Observation 6: A new parameter γ, which handles congestion avoidance, can favor smoothness without damaging responsiveness.

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, or under dynamic traffic loads. A supportive mechanism for error classification and adaptive window adjustments may complement the equation-based adjustments and produce positive dynamics.

On the aspect of error classification [11], previously, we have proposed TCP-Real [12], which relies on a novel receiver-oriented and measurement-based congestion control mechanism to differentiate the error nature. If the packet loss is due to random wireless losses, the congestion window is not reduced and the transmission rate is maintained.

On the aspect of AIMD adjustment itself, it is also important to realize that since throughput is not a direct function of the sending window, adjusting the sending window will not necessarily mean that throughput will drop. Hence, a smooth protocol may drop its sending window without really dropping its throughput. A more responsive protocol which adjusts the sending window more rapidly may go well below the point knee. That is, it will drop its throughput and underutilize the network capacity. The real issue, therefore, is whether the increase rate will suffice to keep the queue length at some level that allows for continuous forwarding of packets at the router. Although some of the above experiments show that responsive protocols do not ensure the continuity of packet forwarding due to their low β, it is indeed positive, due to its high α, when contention decreases and bandwidth becomes available rapidly. Equation (13) corroborates that an efficient window decrease ratio depends on the network settings. It calls for a measurement-based congestion control scheme that can adapt the control parameters to the network condition.

Furthermore, a congestion avoidance strategy may be appropriate to reduce the number of congestive drops and, therefore, the amount and scale of oscillations. Whenever this strategy fails, a classic AIMD parameters ($\alpha = 1, \beta = 0.5$) may apply. The question, therefore, is "Can we achieve smoothness through an avoidance-oriented strategy that, in addition, guarantees a fair and efficient channel multiplexing and utilization?" In fact, we attempt to preserve the responsive capabilities of the transport protocols, which are damaged when we apply smooth TCPs to wireless networks or dynamic environments. One can start off with a simple but—perhaps, at this point—powerful idea to complement the α/β with an additional parameter γ, which will handle the congestion avoidance policy. Our question, therefore, becomes more specific: "How shall we set dynamically parameter γ."

The standard congestion control is complemented with the following congestion avoidance mechanism. The sender measures the fine-grained RTT (see [17] for further details). It records the minimum RTT and the maximum RTT perceived. The queueing delay can be derived by deducting the minimum RTT from the current RTT measured. Upon the detection of the following condition:

$$\frac{\text{qdelay}(t)}{\max \text{qdelay}} = \frac{\text{RTT}(t) - \min \text{RTT}}{\max \text{RTT} - \min \text{RTT}} \geq \text{Th}_{\text{upper}} \tag{15}$$

where the threshold Th_{upper} is experimentally set to be 0.5, the congestion window is decreased after one RTT, with window decrease ratio γ set to be:

$$\begin{aligned}\gamma &= \frac{\text{cwnd}_{\text{knee}}}{\text{cwnd}(t)} = \frac{bw \cdot \min \text{RTT}}{bw \cdot \text{RTT}(t)} \\ &= \frac{\min \text{RTT}}{\min \text{RTT} + \text{Th}_{\text{upper}} \cdot \max \text{qdelay}} \\ &= \frac{\min \text{RTT}}{\text{Th}_{\text{upper}} \cdot \max \text{RTT} + (1 - \text{Th}_{\text{upper}}) \min \text{RTT}} \\ &= \frac{1}{1 + \text{Th}_{\text{upper}} \cdot k}\end{aligned} \tag{16}$$

where k is defined in (13). Equation (16) bears some similarities with (13). Both of them assume a synchronized model, and intend to prevent the system from operating below the knee, based on an adaptive congestion control parameter. In contrast to a static multiplicative decrease ratio of standard TCP, both multiplicative decrease parameters in (13) and (16) are adjusted to prevent the system from operating below the knee, according the current network condition. However, from the perspective of congestion control, γ determines the window decreasing ratio when the level of contention exceeds a threshold that indicates an upcoming congestion. It avoids congestion by scheduling backward window adjustments before the occurrence of congestion. Therefore, transmission gaps due congestive packet drops can be reduced. Since γ-based multiplicative decrease is applied when the system is half way between the knee and the cliff, γ is higher than β.

Another optional control parameter δ can be introduced, in order to enhance the additive increase speed when the network is underutilized. More specifically, when the following condition persists:

$$\frac{\text{qdelay}(t)}{\max \text{qdelay}} = \frac{\text{RTT}(t) - \min \text{RTT}}{\max \text{RTT} - \min \text{RTT}} \leq \text{Th}_{\text{lower}} \tag{17}$$

the queue is relatively close to empty and the bandwidth is possibly underutilized. The additive increase adopts a faster speed: $\delta = 2$. The threshold Th_{lower} is experimentally set to be 0.1. Once the threshold is exceeded, δ hands over the control to α.

For evaluation, we selected five protocol configurations: standard TCP Reno ($\alpha = 1, \beta = 0.5$), Reno with RED [7] configured at R_1, Reno with the γ parameter, Reno with γ and $\delta(= 2)$, and smooth TCP ($\alpha = 0.31, \beta = 0.875$) [14], and TFRC [6]. TCP Vegas [2] was not selected because it does not follow AIMD and, hence, the system does not converge

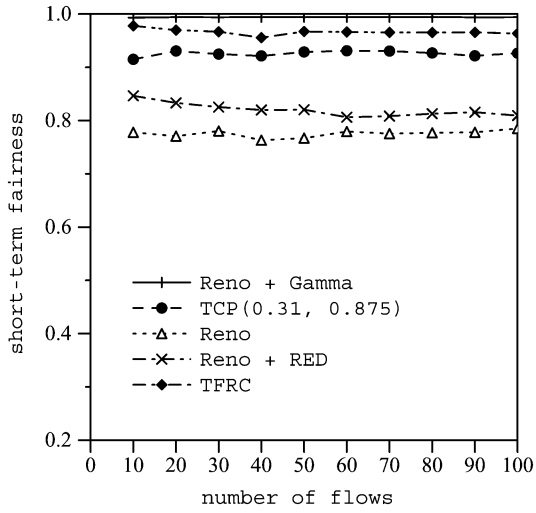

Fig. 18. Short-term fairness.

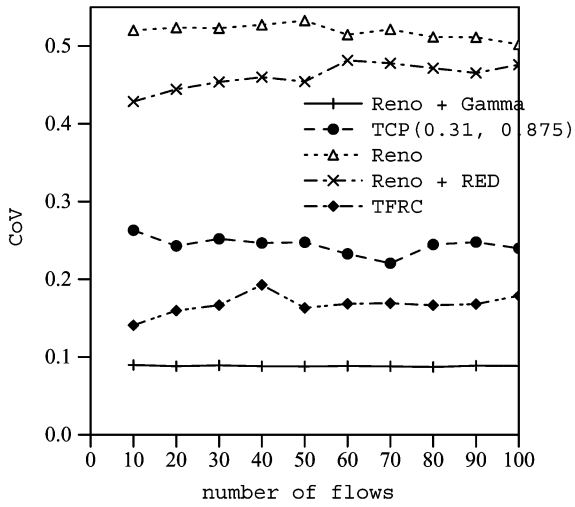

Fig. 19. CoV.

to fairness [8]. We were particularly interested in the comparison between the γ mechanism and TCP(0.31, 0.875), whose smoothness is achieved by increasing the window decrease ratio, at the cost of lesser responsiveness. TFRC is an equation-based rate control mechanism for unicast applications. Since TFRC is not a reliable transport control, it is somewhat unreasonable to compare its performance with reliable TCP protocols. Nonetheless, TFRC was included as a reference of smoothness. Notably, our mechanism can be easily adapted for unreliable media-streaming.

We first conducted ten simulations, with the number of flows varied from 10 to 100. The bottleneck link capacity scaled correspondingly, such that the fair share for each flow was 1 Mb/s. The goodput performances of all protocols are indistinguishable. If assessed by traditional fairness index, all protocols achieve high level of fairness. The interesting metrics here are short-term fairness and CoV measurements defined in Section II.

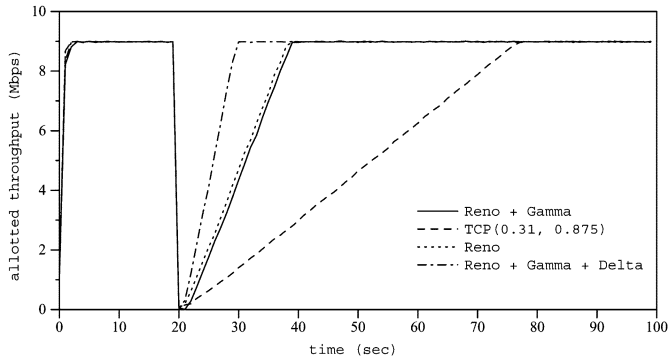

Fig. 20. Throughput with 1.0 s handoff.

The short-term fairness and CoV measurements are shown in Figs. 18 and 19, respectively. Reno $+\gamma$ achieves higher short-term fairness and lower CoV (i.e., higher smoothness) than Reno, Reno + RED, TCP(0.31, 0.875), and even TFRC. Note that the smoothness of Reno $+\gamma$ is achieved not by increasing the window decreasing ratio as with TCP(0.31, 0.875). Since the buffer size is set to the round-trip propagation delay times the bottleneck bandwidth, γ is around 2/3, according to (16). Smoothness is achieved by implementing coordinated downward adjustments, taken before the occurrence of congestion. Furthermore, how much to decrease can be adaptively set according to the measurement on the network condition.

Moreover, the high smoothness of coordinated window adjustments is achieved not at the cost of responsiveness. To evaluate the protocol's capability to exploit the bandwidth that becomes available rapidly (i.e., the responsiveness), we also created a scenario of temporary "blackouts" due to mobile handoffs, during which all transmitted packets were lost. A single flow runs on the 10 Mb/s bottleneck link. At time 20 s, the wireless access link was interrupted by a 1 s handoff period, during which all packets were lost. Since both cwnd and *ssthresh* are reduced to minimum during the handoff, a high sending rate increase is the desired behavior when the handoff is over. The protocols' aggressiveness after the handoff is shown in Fig. 20. Due to the lesser responsiveness ($\alpha = 0.31$), as a result of TCP-friendly α/β tradeoff, it takes 57 s for TCP(0.31, 0.875) to fully recover the transmission speed, while the recovery time for Reno or Reno $+\gamma$ is 19 s, with $\alpha = 1$. If the optional parameter $\delta = 2$ is enabled, the recovery time can be further reduced to 10 s. As the condition of bandwidth underutilization is detected after the handoff is over, a faster additive increase step can be adopted.

V. Conclusion

Departing from our analysis in Section I, we presented a series of observations relevant to the dynamics of responsiveness and smoothness. The gentle backward adjustments entail some serious drawbacks in the presence of high error rates, mobility, and contention decrease. We proposed an additional parameter γ to enhance the α/β congestion control dynamics with a congestion avoidance strategy.

REFERENCES

[1] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC2581, Apr. 1999.

[2] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[3] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comput. Netw. ISDN Syst.*, vol. 17, no. 1, pp. 1–14, 1989.

[4] S. Floyd, "Congestion control principles," RFC 2914, Sep. 2000.

[5] S. Floyd, M. Handley, and J. Padhye. (2000) A comparison of equation-based and AIMD congestion control. [Online]. Available: http://www.aciri.org/tfrc/

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM 2000*, Aug. 2000.

[7] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[8] A. Lahanas and V. Tsaoussidis, "Exploiting the efficiency and fairness potential of AIMD-based congestion avoidance and control," *Comput. Netw.*, vol. 43, no. 2, pp. 227–245, Oct. 2003.

[9] NS-2, The Network Simulator. [Online]. Available: http://www.isi.edu/nsnam/ns/

[10] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM 1998*, Aug. 1998.

[11] V. Tsaoussidis and I. Matta, "Open issues on TCP for mobile computing," *J. Wireless Commun. Mobile Comput.*, vol. 2, no. 1, pp. 3–20, Feb. 2002.

[12] V. Tsaoussidis and C. Zhang, "TCP-Real: Receiver-oriented congestion control," *Comput. Netw.*, vol. 40, no. 4, pp. 477–497, Nov. 2002.

[13] Y. R. Yang, M. S. Kim, and S. S. Lam, "Transient behaviors of TCP-friendly congestion control protocols," in *Proc. IEEE INFOCOM 2001*, vol. 3, Apr. 2001, pp. 1716–1725.

[14] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proc. 8th Int. Conf. Netw. Protocols*, Osaka, Japan, Nov. 2000, pp. 187–198.

[15] C. Zhang, M. Khanna, and V. Tsaoussidis, "Experimental assessment of RED in wired/wireless networks," *Int. J. Commun. Syst.*, vol. 17, no. 4, pp. 287–302, May 2004.

[16] C. Zhang and V. Tsaoussidis, "The interrelation of TCP smoothness and responsiveness in heterogeneous networks," in *Proc. 7th IEEE Symp. Comput. Commun.*, Jul. 2002, pp. 291–297.

[17] ——, "Improving TCP smoothness by synchronized and measurement-based congestion avoidance," presented at the NOSSDAV 2003, Monterey, CA, Jun. 2003.

Vassilis Tsaoussidis (M'97–SM'04) received the B.Sc. degree in applied mathematics and the Diploma in statistics and computer science from the Aristotle University, Greece, and the Hellenic Institute of Statistics, respectively, and the Ph.D. degree in computer networks from Humboldt University of Berlin, Germany, in 1995.

He has held faculty positions at Rutgers University, New Brunswick, NJ, State University of New York (SUNY) Stony Brook, Amherst, NY, and Northeastern University, Boston, MA. In May 2003, he joined the Department of Electrical and Computer Engineering, Demokritos University, Xanthi, Greece. His research interests lie in the area of transport/network protocols, i.e., their design aspects and performance evaluation. He edited three journal special issues on related topics. He Chaired IC 2002, the International Workshop on Wired/Wireless Internet Communications (WWIC 2002), and WWIC 2004. He participates in several Technical Program Committees in his area of expertise, such as INFOCOM, GLOBECOM, ICCN, ISCC, EWCN, WLN, and several others.

Dr. Vassilis is an Associate Editor for IEEE TRANSACTIONS ON MOBILE COMPUTING and an Editor for the Journals *Computer Networks* and *Wireless Communications and Mobile Computing*.

Chi Zhang (M'03) received the B.E. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 1996 and the Ph.D. degree in computer science from Northeastern University, Boston, MA, in 2003.

During the summer of 1999, he was a Research Assistant at the Panasonic Information and Networking Technologies Laboratory, Princeton, NJ. He is an Assistant Professor at Florida International University (FIU), Miami, FL. He has published 16 papers and issued 1 U.S. patent. His research interests lie in the area of network protocols, mobile computing and QoS.

Dr. Zhang received the runner-up award at the IEEE ISCC 2002. He is a member of the Phi Kappa Phi Honor Society. He has Chaired the organizing committee of the First International Workshop on Wired/Wireless Internet Communications (WWIC 2002) and was a TPC member of WWIC 2004/2005 and IC 2003.