A Spectrum of TCP-friendly Window-based Congestion Control Algorithms*

Shudong Jin Liang Guo Ibrahim Matta Azer Bestavros

Computer Science Department
Boston University
Boston, MA 02215

{jins, guol, matta, best}@cs.bu.edu

December 2001

Abstract

The increasing diversity of Internet application requirements has spurred recent interest in transport protocols with flexible transmission controls. In window-based congestion control schemes, increase rules determine how to probe available bandwidth, whereas decrease rules determine how to back off when losses due to congestion are detected. The control rules are parameterized so as to ensure that the resulting protocol is TCP-friendly in terms of the relationship between throughput and loss rate. This paper presents a comprehensive study of a new spectrum of window-based congestion controls. Contrary to previous memory-less controls, our controls utilize history information in their control rules. They are TCP-friendly as well as TCP-compatible under RED. That is, the steady-state throughput of our controls is roughly equal to that of TCP under the same conditions. Furthermore, our controls solve the problems raised by recently-proposed slowly-responsive congestion controls such as general AIMD and binomial controls. Our controls have much better transient behavior. That is, they can achieve better tradeoffs among smoothness, aggressiveness, and responsiveness, and they can achieve faster convergence. We demonstrate analytically and through extensive *ns* simulations the steady-state and transient behavior of several instances of this new spectrum.

Keywords: Congestion Control, TCP-friendliness, TCP-compatibility, Fairness, Transient Behavior.

1 Introduction

TCP uses additive-increase and multiplicative-decrease (AIMD). It probes available bandwidth by increasing the congestion window size linearly, and responds to increased congestion (indicated by packet losses) by decreasing the window size multiplicatively. Recently proposed congestion control mechanisms include generalizations of TCP-like window-based schemes [1, 2, 3, 4], and equation-based schemes [5, 6, 7]. A common objective of these schemes is to reduce the high variability of TCP's transmission rate. Such high variability may limit network utilization. In addition, it is not desirable for emerging applications such as real-time streaming applications on the Internet.

A new transport protocol should implement congestion control mechanisms that interact well with TCP [8]. That is, it should maintain *TCP-compatibility*, or fairness across connections using different protocols. To provide such fairness, one solution is to satisfy *TCP-friendliness*, which means the (λ, p)

*This work was supported in part by NSF grants CAREER ANI-0096045, ANI-0095988, and ANI-9986397. Shudong Jin was also supported by an IBM PhD Research Fellowship.

relationship $\lambda \sim 1/(R\sqrt{p})$ should hold, where λ is the throughput of a flow, p is the loss rate, and R is the round-trip time.

In addition to TCP friendliness, *smoothness, aggressiveness*, and *responsiveness* [1, 9] are important indices of congestion control performance. Smoothness indicates the variability in transmission rate. Aggressiveness indicates how fast a connection probes extra bandwidth by opening up its window. Responsiveness measures how fast a connection reacts to increased congestion by decreasing its window size. Smoothness characterizes the steady-state behavior of congestion control protocols, whereas both aggressiveness and responsiveness characterize transient behavior. An important observation is that there are tradeoffs among smoothness, aggressiveness, and responsiveness [1, 9]. Comparisons of TCP, general AIMD [1, 3], TFRC [5], and TEAR [2] have shown that typically higher smoothness means less aggressiveness and responsiveness.[1]

Several questions remain unanswered. First, both window-based and equation-based congestion control schemes have been studied recently. Window-based schemes do *not* use history while equation-based schemes do so. Could one explore the design space between these two schemes? Second, can one provide a wider selection of TCP-friendly congestion control schemes by using history information? Third, previous approaches provide smoothness of transmission rate but sacrifice aggressiveness. Can one provide high smoothness in steady state as well as better transient behavior when network conditions change drastically (e.g., when there is a sudden increase in available bandwidth)?

We provide answers to these questions. This paper presents a thorough study of TCP-like window-based congestion control schemes that utilize history information, in addition to current window size. These schemes are fundamentally different from memory-less AIMD [1, 3] and binomial schemes [4]. The only history used in our schemes is the window size at the time of detecting the last loss. Such a small step allows a much broader exploration of TCP-friendly congestion controls than memory-less AIMD and binomial schemes. To this end, we propose a spectrum of window-based congestion controls possessing high smoothness in steady state, while reacting promptly to sudden changes in network conditions. We analyze the smoothness, transient behavior, and performance tradeoffs of this new spectrum of controls, of which our recently proposed SIMD [10] is an instance.

Our work is the first step toward exploring a new design space between memory-less window-based congestion control schemes and equation-based schemes which use more history information. Compared to memory-less window-based schemes, our controls improve the transient behavior by using history. Compared to equation-based schemes, our controls have several unique properties: the self-clocking nature of window-based schemes, and simple modifications to TCP's implementation.

The remainder of this paper is organized as follows. We propose our controls in Section 2, and define our TCP-friendly controls in Section 3. We analyze the tradeoffs among smoothness, aggressiveness, and responsiveness in Section 4. The convergence properties of our SIMD instance is studied in Section 5. Our results from extensive simulations using the *ns* simulator [11] are presented in Section 6. We revisit related work in Section 7 and finally conclude the paper.

2 Window-based Congestion Control Using History

A TCP-like window-based congestion control scheme increases the congestion window as a result of the successful transmission of a window of packets, and decreases the congestion window upon the detection of a packet loss event. We call such a sequence of window increments followed by one window decrement a *congestion epoch*. A congestion control scheme defines one control rule for window increase, and another

[1]In feedback control systems, of which congestion control is an example, there is inevitable tension between stability and responsiveness. In our context, we use smoothness as a quality measure of stability, and both aggressiveness and responsiveness as measures of responsiveness. Note, in the control-theory literature, responsiveness usually means how fast the system reaches a target state (rise time), whereas we use aggressiveness and responsiveness to distinguish between how fast the window is increased and decreased, respectively, to reach a target window size.

rule for window decrease. AIMD uses the following control rules:

$$\begin{aligned} \textit{Increase}: &\quad w_{t+1} \leftarrow w_t + \alpha, \quad \alpha > 0, \\ \textit{Decrease}: &\quad w_t \leftarrow w_t - \beta w_t, \quad 0 < \beta < 1. \end{aligned}$$

where w_t is the window size at time t (in RTTs). That is, for AIMD, the window size is increased by a constant when a window of packets are transmitted successfully, and it is decreased by a constant factor instantaneously when a packet loss event is detected. [2] Binomial controls [4] generalize AIMD and use the following control rules:

$$\begin{aligned} \textit{Increase}: &\quad w_{t+1} \leftarrow w_t + \alpha / w_t^k, \quad \alpha > 0, \\ \textit{Decrease}: &\quad w_t \leftarrow w_t - \beta w_t^l, \quad 0 < \beta < 1. \end{aligned}$$

That is, binomial controls generalize additive-increase by increasing inversely proportional to a power k of the current window, and generalize multiplicative-decrease by decreasing proportional to a power l of the current window.

We say that AIMD and binomial controls are memory-less since the increase and decrease rules use only the current window size w_t and constants (α, β, k, and l). Neither of them utilizes history information. We argue that the window size at the end of the last congestion epoch is useful, not only as an indicator of the current congestion level of the network, but also as a good predictor of the congestion state for the next epoch. Thus, our proposed scheme maintains such a state variable $w_{\max}$, which is updated at the end of *each* congestion epoch. In addition, let w_0 denote the window size after the decrease. Given a decrease rule, w_0 can be obtained from $w_{\max}$, and vice versa. For example, for AIMD, $w_0 = (1-\beta)w_{\max}$. Henceforth, for clarity, we use both $w_{\max}$ and w_0.[3]

Such history information can then be used to improve the transient behavior of the control. We propose to adopt the following window increase function:

$$w(t) = w_0 + ct^u, \qquad u, c > 0, \tag{1}$$

where $w(t)$ is the continuous approximation of the window size at time t (in RTTs) elapsed since the window started to increase. By definition, $w_0 = w(0)$. This window increase function is equivalent to the following window increase rule: [4]

$$w_{t+1} \leftarrow w_t + \alpha/(w_t - w_0)^k, \quad \alpha > 0, \tag{2}$$

[2] We use AIMD(α, β) to refer to the general AIMD with additive constant α and multiplicative decrease parameter β. The term *TCP AIMD* refers to AIMD(1, 0.5) or standard TCP. For simplicity, we also use *AIMD* for the general case.

[3] When the slow-start phase of TCP ends and the congestion avoidance phase starts, we have the first value of w_0, i.e., the current window size. Then the first value of $w_{\max}$ is obtained.

[4] **Equivalence of window increase function (1) and window increase rule (2):**
Using linear interpolation and continuous approximation, from (2), we have

$$\frac{dw(t)}{dt} = \frac{\alpha}{(w(t) - w_0)^k}.$$

This gives us

$$(w(t) - w_0)^k dw(t) = \alpha dt,$$

and then by integrating both sides, we have

$$\frac{(w(t) - w_0)^{k+1}}{k+1} = \alpha t + C,$$

Notice that the constant $C = 0$ since when $t = 0$, $w(t) = w_0$. We then rewrite it as (1):

$$w(t) = w_0 + ((k+1)\alpha t)^{1/(k+1)}.$$

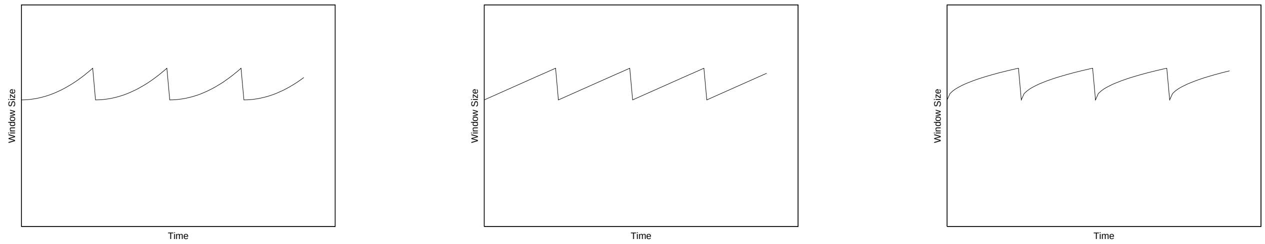


Figure 1: Different increase patterns of congestion window.

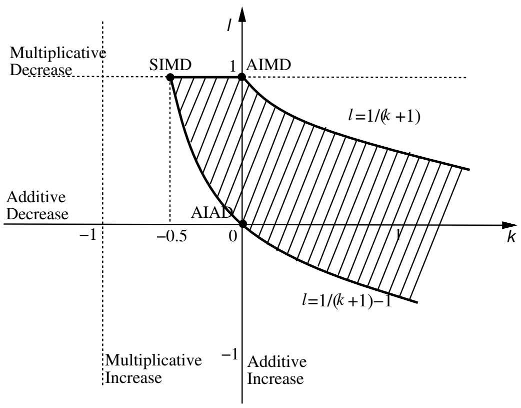


Figure 2: A spectrum of TCP-friendly congestion controls using history.

where $k > -1$ and α is independent of t. In particular, $u = 1/(k+1)$ and $c = ((k+1)\alpha)^u$.

We are interested in congestion control schemes that have various window size increase patterns (different u's, or equivalently, different k's). Consider three cases, as shown in Figure 1. First, if $-1 < k < 0$, the congestion window increases super-linearly. The window is increased cautiously just after the detection of packet loss, and the increase becomes more and more aggressive when no more loss occurs. Second, if $k = 0$, the window increases linearly, i.e., additive increase. The aggressiveness does not change with time. Third, if $k > 0$, the window increases sub-linearly. The connection approaches the previously probed window size fast, but it becomes less aggressive beyond that. These various schemes possess different degrees of aggressiveness, and may satisfy different applications. For example, super-linear increase can support applications that need to quickly acquire bandwidth as it becomes available.

Therefore, we consider the following control rules:

$$\begin{aligned} &\mathit{Increase}: \quad w_{t+1} \leftarrow w_t + \alpha/(w_t - w_0)^k, \quad \alpha > 0, \\ &\mathit{Decrease}: \quad w_t \leftarrow w_t - \beta w_t^l, \qquad 0 < \beta < 1. \end{aligned} \tag{3}$$

Here we use the same decrease rule as binomial controls. It generalizes the multiplicative decrease of AIMD control. For the increase rule, we consider $k > -1$, since otherwise the window size increases exponentially or faster and we consider it unstable. For the decrease rule, we consider $l \le 1$, since otherwise $(w_t - \beta w_t^l)$ can be negative when w_t is large enough.

We illustrate this family of controls as the (k, l) space in Figure 2. In [12], we show that the spectrum inside the shaded area satisfies the convergence-to-fairness property under the synchronized feedback model used by Chiu and Jain [13].

Before further elaboration, we state several main properties of our controls. First, we show that our controls can be TCP-friendly by appropriately defining α as a function of the constant β and the state

(k, l)	Increase rule	Decrease rule	Increase function
$k=0, l=1$, AIMD	$w_{t+1} \leftarrow w_t + \frac{3\beta}{2}$	$w_t \leftarrow w_t - \beta w_t$	$w(t) = w_0 + \frac{3\beta}{2} t$
$k=-\frac{1}{3}, l=1$	$w_{t+1} \leftarrow w_t + 1.89\beta^{\frac{2}{3}} (\frac{w_t - w_0}{w_{\max}})^{1/3}$	$w_t \leftarrow w_t - \beta w_t$	$w(t) = w_0 + \frac{1.4\beta}{\sqrt{w_{\max}}} t^{1.5}$
$k=-\frac{1}{2}, l=1$, SIMD	$w_{t+1} \leftarrow w_t + \frac{3\sqrt{\beta}}{\sqrt{2}} \sqrt{\frac{w_t - w_0}{w_{\max}}}$	$w_t \leftarrow w_t - \beta w_t$	$w(t) = w_0 + \frac{9\beta}{8w_{\max}} t^2$
$k=0, l=\frac{1}{2}$	$w_{t+1} \leftarrow w_t + \frac{3\beta}{2\sqrt{w_{\max}}}$	$w_t \leftarrow w_t - \beta\sqrt{w_t}$	$w(t) = w_0 + \frac{3\beta}{2\sqrt{w_{\max}}} t$
$k=0, l=0$, AIAD	$w_{t+1} \leftarrow w_t + \frac{3\beta}{2w_{\max}}$	$w_t \leftarrow w_t - \beta$	$w(t) = w_0 + \frac{3\beta}{2w_{\max}} t$

Table 1: Several special cases of our TCP-friendly congestion controls using history.

variable $w_{\max}$. We elaborate on this in Section 3. Second, our controls enable different tradeoffs among smoothness, aggressiveness, and responsiveness. We elaborate on this in Section 4. Third, our controls can have better convergence behavior as we show in Section 5 using SIMD [10] as an instance. For SIMD, $k = -0.5$ and $l = 1$.

We need to point out that our controls are radically different from binomial controls [4]. Binomial controls generalize AIMD, but they are still in the memory-less space. Therefore, binomial controls cannot be simply situated on the spectrum in Figure 2.

3 TCP-Friendliness

We show that our control scheme using the control rules in (3) can be TCP-friendly. The notion of TCP-friendliness refers to the relationship between throughput and packet loss rate. We consider a random loss model, where the losses are Bernoulli trials; packets are dropped uniformly with a fixed probability.

In Appendix A, assuming such a random loss model, and without considering the effect of TCP's timeout mechanisms, we explain the use of the following definition of α to make our congestion control scheme TCP-friendly:

$$\alpha = \frac{3}{2(k+1)(1-\frac{1}{k+2}\beta w_{\max}^{l-1})} (\frac{\beta}{\Gamma(\frac{1}{k+1}+1)})^{k+1} w_{\max}^{kl+l-1}, \tag{4}$$

where the Gamma function $\Gamma(.)$ is a constant. According to Section 2, c in Equation (1) is defined as a function of α and we have:

$$c = (\frac{3}{2(1-\frac{1}{k+2}\beta w_{\max}^{l-1})})^{\frac{1}{k+1}} \frac{\beta}{\Gamma(\frac{1}{k+1}+1)} w_{\max}^{l-\frac{1}{k+1}}. \tag{5}$$

When the window size variation is small, i.e., the window decrease is small, $\beta w_{\max}^{l} \ll w_{\max}$, we can simplify α and c as:

$$\alpha \approx \frac{3}{2(k+1)} (\frac{\beta}{\Gamma(\frac{1}{k+1}+1)})^{k+1} w_{\max}^{kl+l-1}. \tag{6}$$

$$c \approx (\frac{3}{2})^{\frac{1}{k+1}} \frac{\beta}{\Gamma(\frac{1}{k+1}+1)} w_{\max}^{l-\frac{1}{k+1}}. \tag{7}$$

That is, α is a constant factor of $w_{\max}^{kl+l-1}$, and c is a constant factor of $w_{\max}^{l-\frac{1}{k+1}}$.

Table 1 gives several special cases. We give their control rules and the window increase functions using, for simplicity, the definition of α in Equation (6) and the definition of c in Equation (7). When $k = 0$ and

$l = 1$, from (4) we have $\alpha_{\text{AIMD}} = 3\beta/(2-\beta)$. If $\beta \ll 1$, $\alpha_{\text{AIMD}} \approx 3\beta/2$. It degenerates to the memory-less TCP-friendly AIMD control [1, 3]. When $k = -0.5$ and $l = 1$,

$$\alpha_{\text{SIMD}} = \frac{3\sqrt{\beta}}{(1-\frac{2\beta}{3})\sqrt{2w_{\text{max}}}}. \tag{8}$$

If $\beta \ll 1$, $\alpha_{\text{SIMD}} \approx \frac{3\sqrt{\beta}}{\sqrt{2w_{\text{max}}}}$. In this case, the window size decreases multiplicatively upon the detection of packet loss, but increases in proportion to the *square* of the time elapsed since the detection of the last loss event (cf. Table 1). We call this control SIMD (Square-Increase/Multiplicative-Decrease).

Another way of illustrating TCP-friendliness is to compare our controls with binomial controls. In [4], the authors show that binomial controls are TCP-friendly. We observe that for every instance of the binomial controls, there is a corresponding point along the line where $k = 0$ and $0 \leq l \leq 1$ in Figure 2 which roughly gives the same control rules. For example, the point $k = l = 0$ (marked as "AIAD" in Figure 2) corresponds to the special case IIAD (Inverse-Increase/Additive-Decrease) of binomial controls. IIAD has the following control rules:

$$\begin{array}{rc} \textit{Increase}: & w_{t+1} \leftarrow w_t + \frac{3\beta}{2w_t}, \\ \textit{Decrease}: & w_t \leftarrow w_t - \beta. \end{array}$$

The only difference between IIAD and our AIAD is in the window increase factor: in IIAD, the factor is inversely proportional to the *current* window size w_t, while in AIAD, the factor is a constant whose value is inversely proportional to w_{max}.[5] Notice that w_{max} records the maximum window size in the previous congestion epoch, thus its value is proportional to the time average of w_t if the TCP congestion window has reached steady state. In other words, IIAD and AIAD controls are equivalent in steady state. However, when there is a sudden increase in network bandwidth, AIAD's linear increase rule is more aggressive than the IIAD's sub-linear increase rule.

The above observation applies to all instances of binomial controls, with only one exception at $k = 0, l = 1$, i.e. AIMD control, where our control algorithm degenerates precisely to general AIMD. However, as shown earlier, for the whole shaded area in Figure 2, our controls can be *adjusted* to be TCP-friendly (cf. equation (4)). This gives the needed flexibility to *control the transient behavior*. For example, as shown in the next section, by exploiting the history information w_{max}, SIMD control is able to increase the window super-linearly (more aggressively than AIMD) and shows much better transient behavior, without affecting TCP-friendliness.

In this paper, due to space limitation, we only present results for SIMD, AIMD, and AIAD as instances in the spectrum of Figure 2.

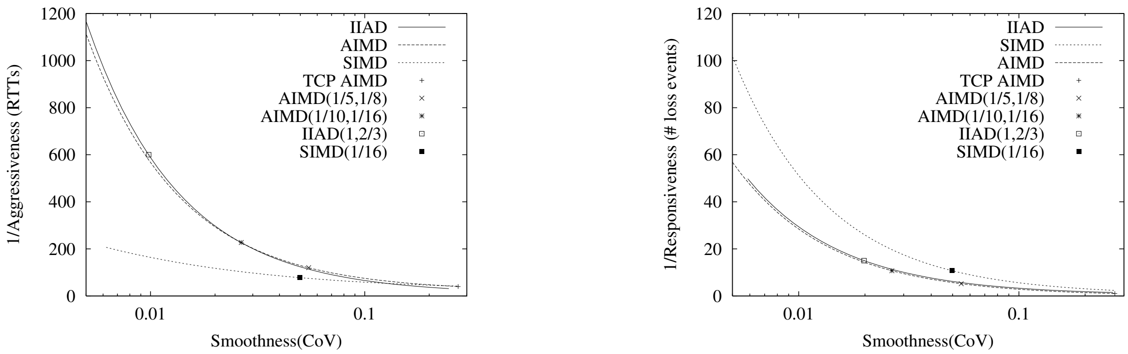
4 Tradeoffs among Smoothness, Aggressiveness, and Responsiveness

In this section, we consider important properties of congestion controls other than TCP-friendliness. These are smoothness, aggressiveness, and responsiveness. Smoothness measures the variability in a connection's window size over time. High variability is not desirable. Aggressiveness measures how fast a connection probes bandwidth as it becomes available by opening up its window. Higher aggressiveness implies potentially higher utilization. Responsiveness measures how fast a connection decreases its window size in response to increased congestion. Both aggressiveness and responsiveness are measures of transient behavior.

[5] Unlike our history-based AIAD control, memory-less AIAD increases its window by an amount that is constant over all congestion epochs. Memory-less AIAD controls have been shown not to converge [13].

	Smoothness	*1/Aggressiveness*	*1/Responsiveness*
AIMD	$\frac{0.41\beta}{1-\beta/2}$	$\frac{m-1}{\beta}\frac{2W}{3}$	$\log_{(1-\beta)}\frac{1}{m}$
IIAD	$\frac{0.41\beta}{W-\beta/2}$	$\frac{(m^2-1)W^2}{3\beta}$	$\frac{W(1-1/m)}{\beta}$
SIMD	$\frac{0.73\beta}{(1-2\beta/3)^2}$	$\sqrt{\frac{m-1}{\beta}\frac{2\sqrt{2}W}{3}}$	$\log_{(1-\beta)}\frac{1}{m}$
AIAD	$\frac{0.41\beta}{W-\beta/2}$	$\frac{2(m-1)W^2}{3\beta}$	$\frac{W(1-1/m)}{\beta}$

Table 2: Smoothness, Aggressiveness, and Responsiveness comparisons of AIMD, IIAD, SIMD, and AIAD.



(a) Aggressiveness vs Smoothness.

(b) Responsiveness vs Smoothness.

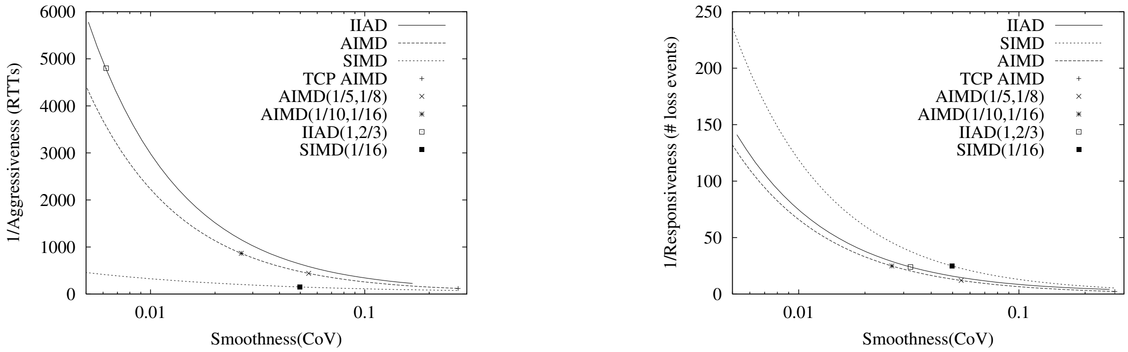
Figure 3: Tradeoffs of smoothness, aggressiveness, and responsiveness. For (a), we assume available bandwidth is doubled. For (b) we assume the window is reduced to half, i.e., $m = 2$. The initial average window size, W, before bandwidth changes is 20.

Smoothness can be observed at different time scales [1]. We consider short time scales since long-term smoothness can be affected by other dynamics in the system. We define smoothness as the variation of the window size of a connection during one congestion epoch. In particular, we use the coefficient of variation of window size in one congestion epoch as a measure of short-term smoothness. Note that the coefficient of variation is not necessarily an accurate measure of smoothness, but it is adequate to give insight into the tradeoffs. We define aggressiveness as the inverse of the time needed for the connection to increase the window size, in response to a step increase of available bandwidth [9]. That is, the available bandwidth is increased by a factor of m. We define responsiveness as the inverse of the number of loss events required for the connection to decrease its window by a substantial amount, in response to a step increase of congestion [9]. That is, a decrease of available bandwidth by a factor of m.

Table 2 gives the approximate expressions of smoothness, aggressiveness, and responsiveness for AIMD, IIAD, SIMD, and AIAD controls. More details are given in [12]. Intuitively, the smoothness index is proportional to the window decrease divided by the average window size. Aggressiveness is determined by the window size increase function. Responsiveness is determined by the decrease rule.

Numerical results in Figure 3 show the tradeoffs among smoothness, aggressiveness, and responsiveness. Results for AIAD are not shown here since they are similar to those of IIAD except that AIAD has higher aggressiveness. Figure 3(a) shows the inverse of aggressiveness of AIMD, SIMD, and IIAD as the coefficient of variation varies. Their special cases TCP, AIMD(1/5,1/8), AIMD(1/10,1/16), IIAD(1,2/3), and SIMD(1/16) are also shown by points. The inverse of aggressiveness is computed as the number of RTTs necessary to double the window size, i.e., $m = 2$. Figure 3(b) shows the inverse of responsiveness of AIMD, IIAD, and SIMD as the coefficient of variation varies. The inverse of responsiveness is computed assuming the target window size is half of the current window size, i.e., $m = 2$.

From this figure, we can see that SIMD has much higher aggressiveness (fewer RTTs) than the others,



(a) Aggressiveness vs Smoothness.

(b) Responsiveness vs Smoothness.

Figure 4: Tradeoffs of smoothness, aggressiveness, and responsiveness. The configurations are the same as those of Figure 3 except that here the bandwidth decrease or increase factor $m = 5$.

especially when high smoothness (low coefficient of variation) is needed. Meanwhile, SIMD has comparable responsiveness index. In particular, SIMD shows up to order of magnitude better aggressiveness at less than about 1.7 times lower responsiveness for about the same smoothness value. For example, we can predict that AIMD(1/20,1/30), SIMD(1/30), and IIAD(1,2/3) have comparable smoothness when the average window size is 20. However, SIMD(1/30) can react to a substantial increase of available bandwidth much faster. The smoothness-aggressiveness relationship can also be inferred from Table 2. For both AIMD and IIAD, aggressiveness varies in proportion to the coefficient of variation. For SIMD, aggressiveness varies as the square root of the coefficient of variation. Thus, when the transmission rate is very smooth, SIMD has much higher aggressiveness than AIMD and IIAD.

Figure 4 shows the same tradeoffs, except that we use a larger factor $m = 5$ for the sudden increase and decrease of available bandwidth. It shows that the advantage of SIMD's aggressiveness is more pronounced. We can also observe from Table 2 that, for SIMD, aggressiveness is inversely proportional to the square root of m, and for AIMD and IIAD, aggressiveness is inversely proportional to m or even m^2, respectively. Therefore, larger m makes SIMD more favorable.

Remark: In the spectrum of controls in Figure 2, SIMD is the one whose aggressiveness grows the fastest. SIMD has the best tradeoff between smoothness in steady state and aggressiveness during transient periods. As k increases, the spectrum of controls have worse tradeoffs.

5 Convergence to Fairness and Efficiency

In this section, we first show the convergence of our SIMD instance. Then we show that SIMD has better convergence behavior than AIMD.

We adopt the ideal synchronized feedback assumption [13]. To show that multiple users with synchronized feedbacks using our control scheme converge to fairness, we use the vector space used by Chiu and Jain [13] to view the system state transitions as a trajectory. For ease of presentation, we show a two-user case. It is straightforward to apply the same technique to the multiple-user case to reach the same conclusion.

As shown in Figure 5, any two-user resource allocation can be represented by a point $X(x_1, x_2)$, where x_i is the resource allocation (normalized by total capacity) for the i^{th} user, $i = 1, 2$. We define the fairness index as $\max(\frac{x_1}{x_2}, \frac{x_2}{x_1})$. If the fairness index is closer to unity, the resource allocation is more fair. The line $x_1 = x_2$ is the "fairness line". The line $x_1 + x_2 = 1$ is the "efficiency line". The goal of control schemes is to bring the system to the intersection of the fairness line and the efficiency line. When the system is

(a) AIMD trajectory

(b) SIMD trajectory

Figure 5: Convergence of AIMD and SIMD

(a) Metrics Definition

(b) AIMD vs SIMD

Figure 6: Comparison of convergence speed

under-utilized, assuming $x_1 \leq x_2$ without loss of generality, AIMD increases the resource allocation of both users by a constant. Figure 5(a) shows the trajectory to X' parallel to the fairness line. This movement improves fairness, i.e., reduces the fairness index. Then both users use multiplicative decrease, which does not change fairness. Hence, as the system evolves, AIMD brings the resource allocation point toward the fairness line, finally oscillating around the efficiency line.

For SIMD control, we first observe Table 1. We can see that the window size of a connection increases in proportion to $\frac{1}{w_{\max}}$ or $1/x_i$ here for $i = 1, 2$. Thus, as shown in Figure 5(b), the increase trajectory emanates from $X(x_1, x_2)$ with slope $\frac{x_1}{x_2}$. Indeed, at any point between the two lines emanating from the origin with slopes $\frac{x_1}{x_2}$ and $\frac{x_2}{x_1}$, the resource allocation X' is more fair than X as it reduces the value of the fairness index. Therefore, the increase phase of SIMD improves fairness. Since like AIMD, SIMD uses multiplicative decrease, the decrease phase of SIMD does not change fairness. Hence, SIMD converges to fairness and efficiency.

We also analytically compare the convergence time of SIMD, AIMD, and binomial control schemes. We still assume synchronized feedback and use Figure 6(a) to illustrate the process of convergence. For ease of analysis, we choose the variables to be the actual window sizes (w_1,w_2). We also divide the convergence time into two parts: T_1, the time it takes the control mechanism to bring an arbitrary initial point (W_1, W_2), where $W_1 \leq W_2$ and $W_1 + W_2 < W$, close to the efficiency line $w_1 + w_2 = W$, and T_2, the time until the difference between the two user windows stays within a certain small bound, i.e., $|w_1 - w_2| < \epsilon$. T_1 and T_2 are measured in round-trip times. We also denote the difference between the two user windows after T_1 as Δ. Due to space limitation, we only present the main results here in Table 3. The detailed analysis can be found in [12].

Algorithm	T_1 (RTT)	Δ	T_2 (RTT)
TCP	$\frac{W-W_1-W_2}{2}$	$W_2 - W_1$	$\frac{W}{4}\log_{1/2}\frac{\epsilon}{\Delta}$
AIMD	$\frac{(W-W_1-W_2)(2-\beta)}{6\beta}$	$W_2 - W_1$	$\frac{(2-\beta)W}{6}\log_{1-\beta}\frac{\epsilon}{\Delta}$
IIAD	$\frac{1}{12\beta}((\frac{(W_2^2-W_1^2)}{W})^2 - 2(W_1^2+W_2^2)+W^2)$	$\frac{W_2^2-W_1^2}{W}$	$\frac{W}{3}\log_{1-2\beta/W}\frac{\epsilon}{\Delta}$
SIMD	$\frac{2}{3}(1-\frac{2\beta}{3})\sqrt{\frac{2}{\beta(1-\beta)}}\sqrt{\frac{W_1W_2(W-W_1-W_2)}{W_1+W_2}}$	$(2-\frac{W}{W_1+W_2})(W_2-W_1)$	$\frac{\sqrt{2}W}{3}\log_{1-2\beta}\frac{\epsilon}{\Delta}$

Table 3: Performance measures on convergence to fairness and efficiency

Description	*Value*
Packet size	1000 bytes
Maximum window	128 packets
TCP version	SACK
TCP timer granularity	0.1 seconds
RED queue limit Q	2.5 $\times$ B/W delay product
DropTail queue limit	1.5 $\times$ B/W delay product
RED parameters	min_{th}: 0.15Q, max_{th}: 0.5Q, w_q:0.002
	max_p:0.1, wait_ on, gentle_ on

Table 4: Network configuration

We numerically solve the above equations for different initial points. Figure 6(b) shows the regions for which SIMD with $\beta = 1/16$ converges faster/slower (i.e., $T_1 + T_2$ is smaller/larger) than TCP-friendly AIMD with $\beta = 1/16$ for $\epsilon = 1$ and $W = 100$. In most cases SIMD converges faster than AIMD. Numerical results also show that IIAD with $\alpha = 1$ and $\beta = 2/3$ is much slower than AIMD and SIMD in all cases.

6 Simulation Results

We use the *ns* simulator [11] to validate that with RED [14] queue management, our proposed controls, most notably SIMD, are TCP-friendly and TCP-compatible. In addition, we compare our controls to standard TCP [15], generalized AIMD [3], and IIAD [4], in terms of smoothness, responsiveness, and aggressiveness. In most simulations, we also include AIAD. In addition, we investigate the way two homogeneous flows converge to their bandwidth fair share and show that our SIMD algorithm outperforms other algorithms. Details about the implementation of SIMD in the *ns* simulator are described in Appendix B.

Unless explicitly specified, in all of the experiments, RED is used as the queue management policy at the bottleneck link. The bottleneck queue configuration and other simulation parameters are listed in Table 4.

The bottleneck queue size and RED queue parameters are tuned as recommended in [16]. The "gentle_" option of RED queue is turned on as recommended in [17]. We choose $\beta = 1/16$ for SIMD and AIMD (and thus $\alpha \approx 1/10$ for AIMD to ensure TCP-friendliness). For IIAD, $\alpha = 1$ and $\beta = 2/3$. For AIAD, $\beta = 2/3$. For ease of presentation, in the rest of this section, we will call these implementations by their family name, e.g., AIMD for AIMD(1/10,1/16) when there is no confusion. We use SACK [18] for congestion detection. We also obtained similar results for other mechanisms such as Reno and newReno. We assume no delayed acknowledgments.

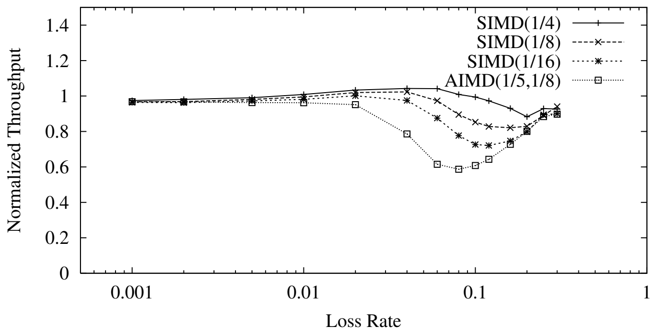


Figure 7: TCP-Friendliness

6.1 TCP-Friendliness and TCP-Compatibility

6.1.1 TCP-Friendliness

We conduct the following experiment to test the TCP-friendliness of our SIMD control: A single flow under investigation is traveling through a single fat link with infinite bandwidth and buffer size. However, the link drops an incoming packet uniformly with probability p. We vary the loss rate p and compare the normalized long-term throughput of SIMD (with respect to standard TCP measured over 3000 RTT) for different β values and plot them in Figure 7. For comparison, we also plot the throughput of AIMD(1/5,1/8).

We notice that all of the curves have a dip when the loss rate is moderate. A close look at the following TCP-friendly equation [19] can reveal one possible explanation of this abnormality.

$$\lambda(p,\alpha,\beta) \approx \min(\frac{W_{\max}}{R}, \frac{1}{R\sqrt{\frac{2\beta}{\alpha(2-\beta)}p} + T_0\min(1, 3\sqrt{\frac{\beta(2-\beta)}{2\alpha}p})p(1+32p^2)})$$

When loss rate is low, TCP mainly stays in the *congestion avoidance* stage, and AIMD control dominates the equation, while when loss rate is very high, TCP spends most of its time retransmitting packets, and the *exponential back-off* control dominates the equation. Since all controls studied in this paper use the same timeout mechanism as standard TCP, and they carefully calibrate the values of their parameters during congestion avoidance to match standard TCP, they can achieve comparable throughput as standard TCP for very high and low loss rates. However, for the loss regime in between, it becomes hard, if not impossible, to obtain α and β values that would approximate well both congestion avoidance and exponential backoff components of the TCP-friendly equation [3].

Nevertheless, in the worst case with loss rate around 15%, SIMD(1/16), which is the worst among all SIMD controls considered, can achieve at least 75% throughput as standard TCP, and performs much closer to standard TCP than AIMD(1/5,1/8) [6]. Given the fact that most parts of the Internet are experiencing less than 5% loss rate [20], our control is TCP-friendly under these conditions.

6.1.2 TCP-Compatibility

We use the method described in [1] to test TCP-compatibility. n SIMD flows and n standard TCP SACK flows compete for bandwidth over a shared bottleneck link. There are also four background TCP flows transmitting packets in the opposite direction to introduce random ACK delays. We consider both RED

[6] The weakness of AIMD(α, β) with small β under intermediate loss conditions is also reported in [1]. The authors try to compensate for the bandwidth loss by increasing the value of α. However, when loss rate is small (e.g. less than 3%), AIMD with large α could achieve significantly higher bandwidth than standard TCP and become less TCP-friendly. Therefore, we maintain the theoretical α values throughout our simulations.

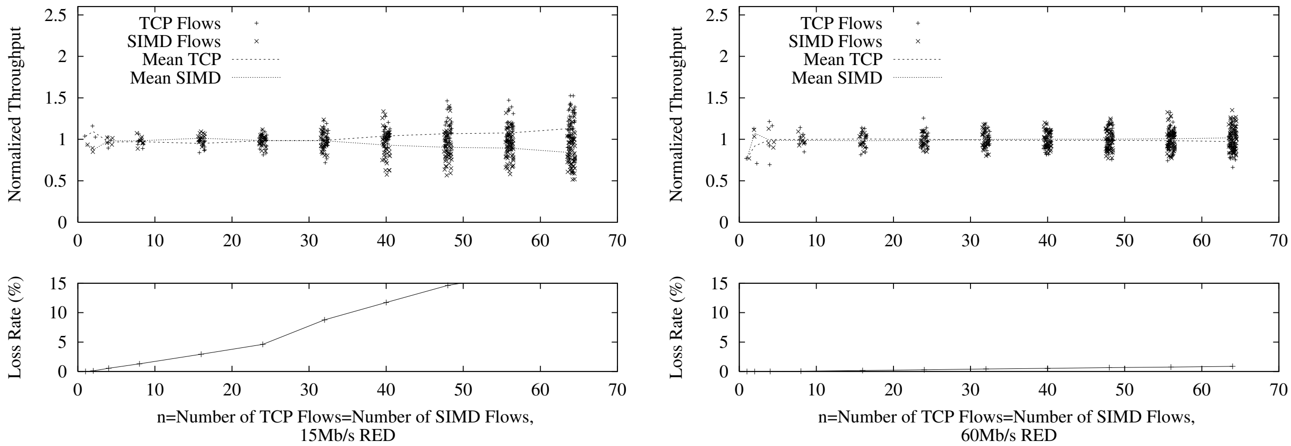


Figure 8: TCP competing with SIMD(1/16), RED with ECN

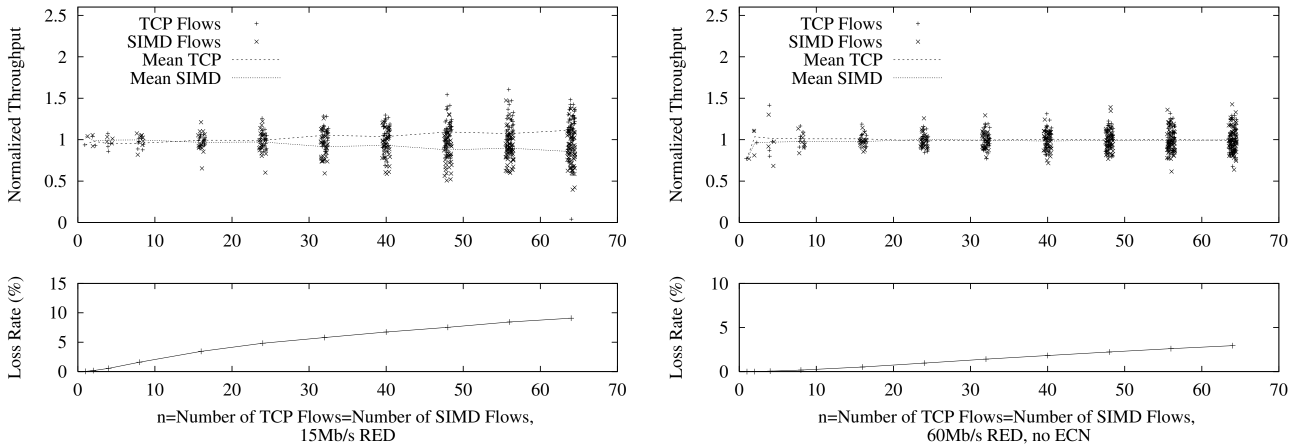


Figure 9: TCP competing with SIMD(1/16), RED without ECN

and DropTail queues. Figure 8 and Figure 9 show the simulation results for RED queues, with and without ECN bit set, respectively. In each case, results are shown for a bottleneck link bandwidth of 15Mbps and 60Mbps. The measured average round-trip delay is around 0.1 second. Each point in the graph represents the throughput of an individual flow in the last 60 seconds, and the dashed lines represent the average throughput of SIMD and standard TCP flows. In the lower graphs, we also plot the packet loss rate for the RED without ECN case, and the rate of ECN early marking plus dropping due to queue overflow for the RED with ECN case.

As can be observed from the graphs, when the loss rate is low, SIMD achieves very close throughput as standard TCP. When the loss rate exceeds a certain level, SIMD achieves a slightly lower average throughput. This is partly due to the reason we illustrate in Figure 7. Another possible explanation is that when severe congestion happens, SIMD can not compete well against standard TCP since compared to TCP, SIMD opens its congestion window more conservatively at the beginning of each congestion epoch. Therefore, when the time between two consecutive packet losses is short, the more aggressive TCP tends to gain more throughput. However, in a reasonable loss regime with loss rate below 10%, SIMD shows

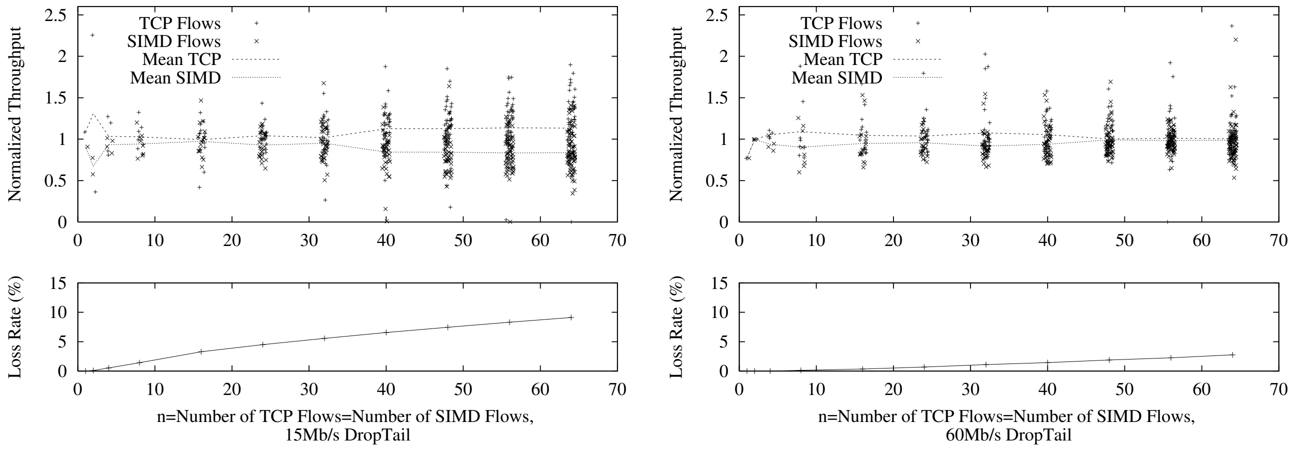


Figure 10: TCP competing with SIMD(1/16), with DropTail

very impressive TCP-compatibility [7].

We also found that with DropTail queue management, as shown in Figure 10, SIMD can still be TCP-friendly and TCP-compatible. The difference, compared to the RED queue experiment, is that the variance becomes larger and SIMD now gets slightly less share of bandwidth. Note that the assumption of randomized packet losses made in our analysis does not apply to DropTail. Under DropTail, packet losses are more correlated. We conjecture that because the round-trip times of connections are randomized in the simulation, the chance of having synchronized packet arrivals is small, and the side effect of a DropTail queue (correlated drops for each flow) is thus not so significant.

We also report corresponding results in Figure 11 for the case of AIAD competing for bandwidth with TCP under the same simulation setup. The conclusion is similar: AIAD shows TCP-compatibility across a wide range of simulation parameters.

6.2 Smoothness, Responsiveness and Aggressiveness

6.2.1 Smoothness

As revealed by the study in [1], the long-term smoothness of traffic is mainly determined by packet loss patterns and it tends to follow the same distribution at large time-scales (more than 100 RTT's), regardless of which congestion control is used. We thus focus our simulation on short-term smoothness and use the simulation code contributed by [1] to study the traffic generated by the congestion controls under investigation. To this end, we let n such flows compete for a bottleneck link (with capacity C) with another n standard TCP flows. There are also some TCP flows traversing in the opposite direction to introduce random ACK delays. In Figure 12 we show the case for $n = 16$ and $C = 60$Mbps, which corresponds to roughly 0.3% packet drop rate. The bottleneck queue strategy is RED with ECN enabled. Figure 13 shows the same setup with ECN turned off. Each graph shows one flow's throughput on the congested link during the time interval between 250 to 270 seconds of a 500-second simulation. The throughput is averaged over 0.2-second intervals, which correspond to twice a typical round-trip time for this simulation. As in [5], we also plot the time at which a packet is marked (or dropped in Figure 13) at the bottom of each curve.

We can observe from the graphs that all four controls AIMD, IIAD, SIMD, and AIAD have roughly

[7] Note that in case of 60Mbps link and less than four flows, the length of the measurement period (60 seconds) is too short compared to the length of each congestion epoch (more than 40 seconds), thus the variance of the results appears to be large.

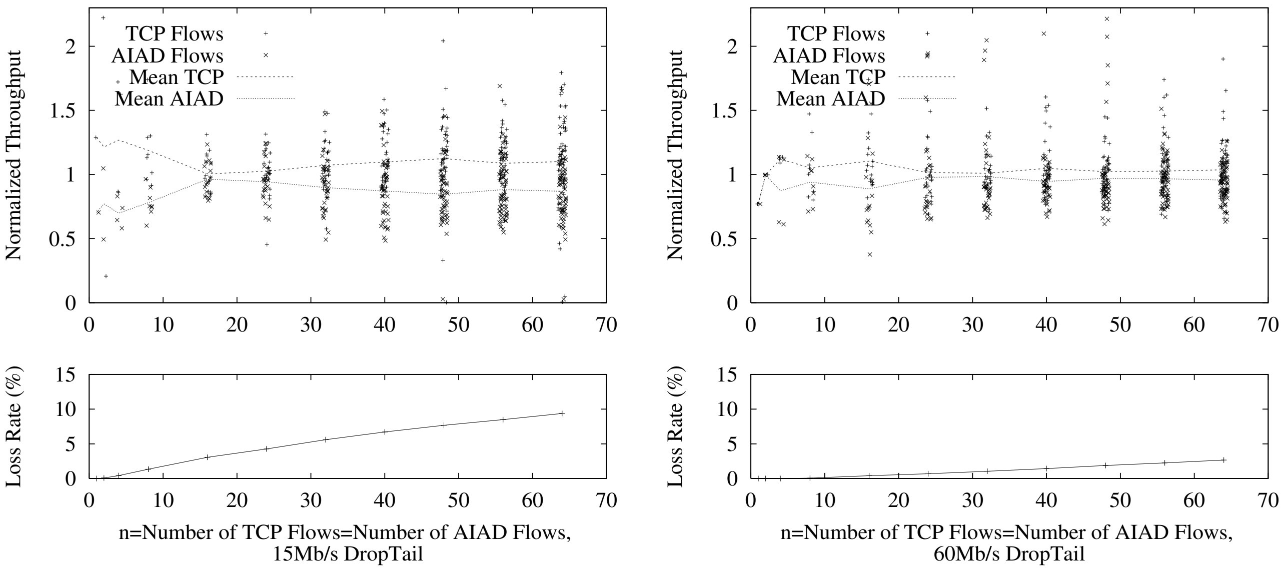
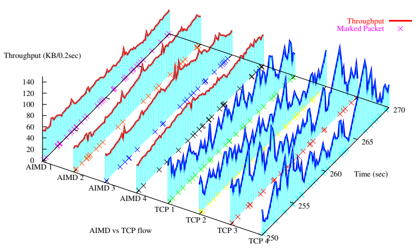
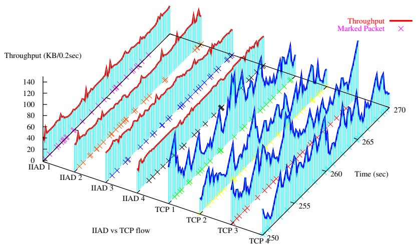


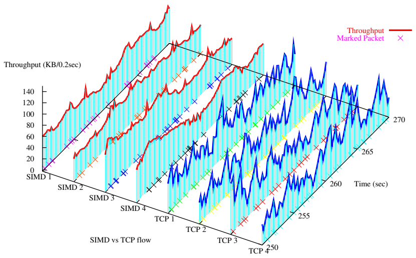
Figure 11: TCP competing with AIAD(2/3), with DropTail



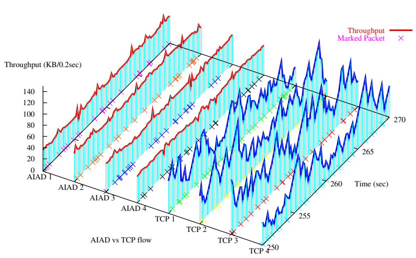
(a) AIMD(1/10,1/16) with TCP



(b) IIAD with TCP

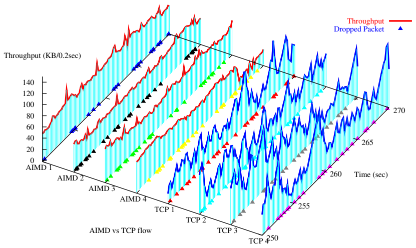


(c) SIMD(1/16) with TCP

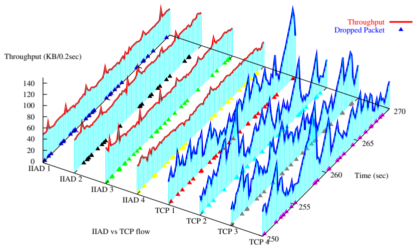


(d) AIAD(2/3) with TCP

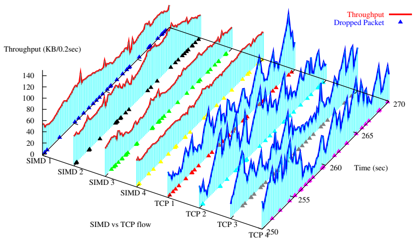
Figure 12: Traffic smoothness, 16 + 16 flows, 60Mbps link, RED with ECN



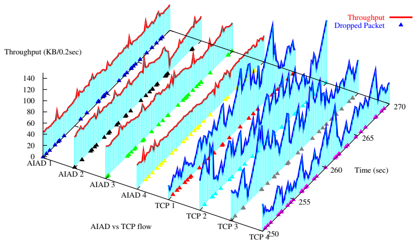
(a) AIMD(1/10,1/16) with TCP



(b) IIAD with TCP



(c) SIMD(1/16) with TCP



(d) AIAD(2/3) with TCP

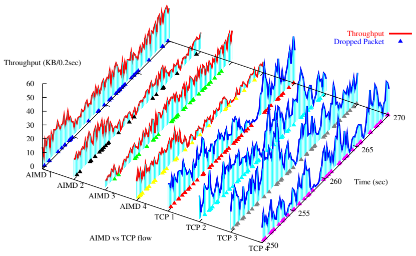
Figure 13: Traffic smoothness, 16 + 16 flows, 60Mbps link, RED without ECN

the same scale of short-term burstiness with SIMD having a little larger variation. This agrees with our analysis (cf. Section 4). In particular, by plugging in equations of Table 2 the values we choose in our simulation of $\beta = 1/16$ for AIMD and SIMD, and $\beta = 2/3$ for IIAD and AIAD, and since the average window size in this simulation is about 23 packets, or $W \approx 23$, we find that the order of the coefficients of variation of these controls (from low to high) is: IIAD (and AIAD), AIMD, and SIMD. Our experiment results show that this is indeed the case.

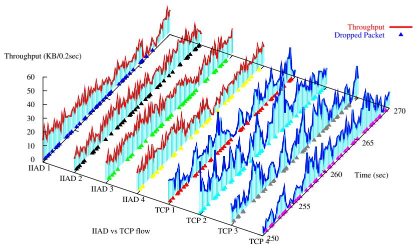
We also decrease C to 15 Mbps (thus increase the congestion level to nearly 5% loss rate) in another experiment set. We show the results in Figure 14 with ECN turned off.

We observe that the smoothness of all four controls becomes worse when the network becomes more congested. This is again due to the self-clocking mechanism of window-based congestion control. With smaller average congestion window, the chance that a retransmission timeout happens becomes higher, so does the chance that the congestion window reduces to 1. We thus can observe abrupt reduction of sending rate more frequently. Although, in general, AIMD, IIAD, AIAD, and SIMD still exhibit smoother transmission than TCP, it is not easy for window-based schemes to achieve high smoothness. This is probably a common weakness of window-based schemes. On the contrary, equation-based schemes [5] can achieve high smoothness even when the loss rate is high.

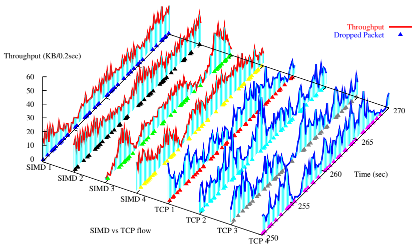
We also observe that the throughput of AIMD degrades significantly. IIAD and AIAD also get less than their fair share. This is in part due to the reason mentioned in Section 6.1.1, that is, AIMD becomes less competitive than standard TCP in this loss regime. The other reason, we conjecture, is that AIMD control does not give any preference to the sender with smaller congestion window (cf. Section 6.3). Thus, when no loss happens, TCP increases its congestion window more aggressively and gets higher throughput than AIMD, which eventually gives up the fair share it deserves. SIMD overcomes this problem and can achieve throughput close to TCP in this scenario.



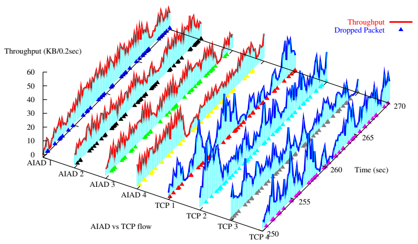
(a) AIMD(1/10,1/16) with TCP



(b) IIAD with TCP



(c) SIMD(1/16) with TCP



(d) AIAD(2/3) with TCP

Figure 14: Traffic smoothness, 16 + 16 flows, 15Mbps link, RED without ECN

6.2.2 Impulse Response

To better illustrate the aggressiveness and responsiveness properties of different controls, we now study the behavior of different controls responding to impulse disturbance from a periodical On/Off constant-bit-rate (CBR) flow. [8] The model is similar to the "square-wave" model used in the simulation study of [21]. In the experiment, we let the CBR flow alternate between On and Off state, each of which lasts for t_{on} and t_{off}, respectively. The sending rate of the CBR flow during the active period is set to γ times C, the capacity of the bottleneck link. We intend to see the effect of such bandwidth oscillation on the transmission of a long TCP flow using the control under study. The results reported here are for $C = 1.5$Mbps, average end-to-end RTT (including queueing delay) = 100ms, $t_{on} = 30$ seconds, $t_{off} = 30$ seconds, and $\gamma = 0.5$. Both flows start around time 0 with some random disturbance. Figure 15 plots the congestion window value of different controls over time period [80:200].

We also prolong our simulation to repeat this impulse disturbance pattern and measure the average aggressiveness and responsiveness according to our definitions in Section 4 and report these data in Table 5. We choose the steady-state error to be one packet within the target window size, and the simulation results are shown in the form of 95% confidence intervals.

As expected, standard TCP is highly variable, IIAD and AIMD are the smoothest since the average window size is larger than 10, at the expense of slow response to bandwidth increases. With similar smoothness, SIMD is much more aggressive than AIMD, IIAD, and AIAD. In addition, AIAD is more

[8]To make the graphs more readable, we use error detection mechanisms of TCP newReno, instead of SACK, so that different controls detect and react to loss at about the same time, in response to duplicate acknowledgments. Using TCP SACK does not qualitatively change the conclusion.

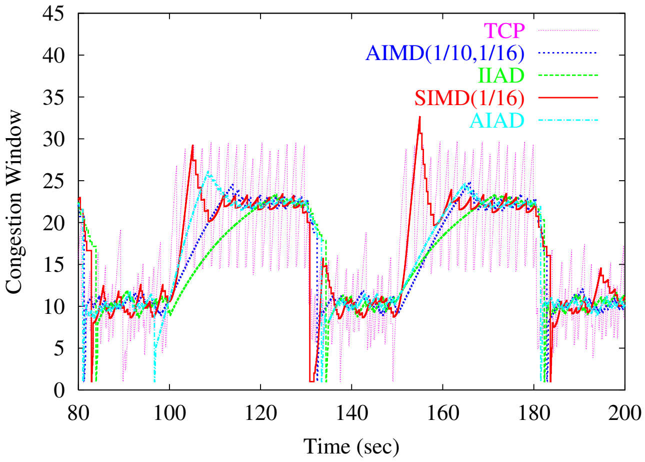


Figure 15: Impulse response to square-wave CBR flow.

Algorithm	*1/Aggressiveness (RTT)*		*1/Responsiveness (losses)*	
	simulation	analysis	simulation	analysis
TCP	(12.8,14.1)	14.7	(1.54,1.63)	1
AIMD	(108.1,110.7)	117.6	(3.85,5.19)	10.7
IIAD	(172.8,176.0)	181.5	(4.20,5.82)	16.5
SIMD	(31.6,34.6)	41.5	(4.21,5.47)	10.7
AIAD	(103.3,107.9)	121.0	(3.73,4.81)	16.5

Table 5: Quantitative Measures

aggressive than IIAD. Notice the close match between the simulated measure of aggressiveness and the analytical results.

Aggressiveness of a congestion control is directly related to how much bandwidth a flow can get when it is competing with other flows. It has been shown in [21] that the set of slowly-responsive congestion controls proposed so far all tend to receive significantly less bandwidth than competing standard TCP flows in a highly dynamic network environment. However, since SIMD maintains good aggressiveness property, the loss of bandwidth is relatively minor (cf. Figure 9).

We also notice that the responsiveness of a control is hard to measure due to the extreme way TCP responds to burst of losses, which will occur when it sees sudden decrease of bandwidth. In this case, all TCP flows reduce their congestion window to one regardless of which congestion avoidance strategy is used. However, we still show the measured responsiveness in Table 5 to provide a qualitative comparison. Generally speaking, the smooth transmission of a slower responsive flow comes at the cost of more packet losses when available bandwidth is suddenly decreased.

6.3 Convergence to Fairness and Efficiency

In this section, we assume a homogeneous protocol environment, i.e., all flows use the same algorithm for congestion control. We then vary the network configuration to study the convergence time of different algorithms.

We use the topology shown in Figure 16 to perform this experiment. In the beginning of the simulation, there are $c_1 + 1$ connections sharing link ($b1$, $b2$), 2 connections sharing link ($b2$, $b3$), $c_2 + 1$ connections between $b3$ and $b4$. Link bandwidths and delays are shown in the figure. At time 400, all background flows terminate and only two flows (s1-r1) and (s2-r2) stay to compete for the bottleneck link ($b2$,$b3$). We use packet size of 500 bytes in these experiments.

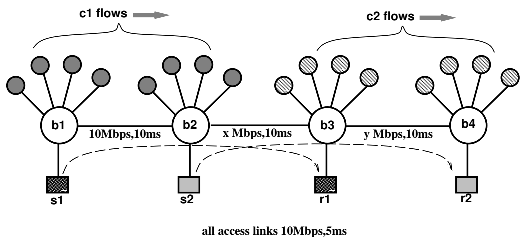


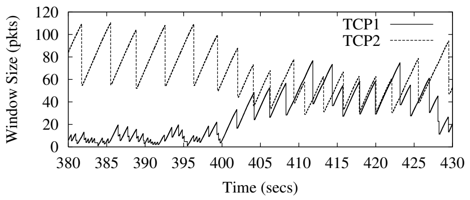
Figure 16: Simulation topology for convergence test

Algorithm	Experiment 1				Experiment 2					
	W_1	W_2	T_2 (RTT)		W_1	W_2	T_1 (RTT)		Δ (pkts)	
			simu	anal			simu	anal	simu	anal
TCP	6.1	99.6	68.0	88.7	8.8	13.8	55	43.7	5.8	6.0
AIMD	7.9	99.2	776	1217	12.7	31.0	349	342	18.6	18.3
IIAD	7.7	99.8	4232	6684	11.8	31.2	1284	1242	8.1	7.6
SIMD	6.6	96.3	**218**	**852**	10.2	33.2	90	85.1	13.6	12.3

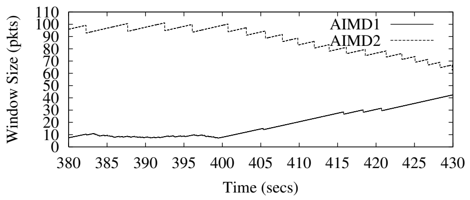
Table 6: Quantitative measures on convergence time

6.3.1 Convergence to Fairness ($W_1 + W_2 = W$,$W_1 < W_2$)

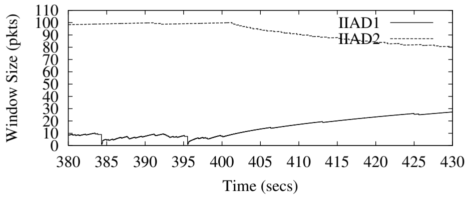
We create this scenario to study the convergence time to fairness given that the initial point (W_1, W_2) is on the efficiency line ($w_1 + w_2 = W$). To create this setup, we let $c_1 = 15$, $c_2 = 0$, $x = 6$Mbps, $y = 6$Mbps. So the bottleneck link for flow (s2,r2) remains link ($b2$,$b3$), but for flow (s1,r1), the bottleneck changes from link ($b3$,$b4$) to ($b2$, $b3$) at time 400. We can also compute that: $W \approx 110$, $W_1 \approx 7$, and $W_2 \approx 100$. Figure 17 plots the transient behavior of the congestion window of different protocols.



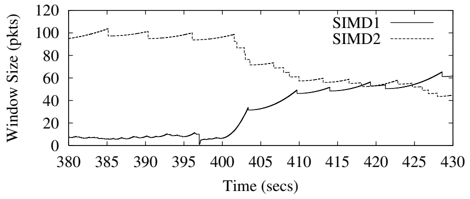
(a) TCP



(b) AIMD(1/10,1/16)



(c) IIAD



(d) SIMD(1/16)

Figure 17: Two flows converge to fair share of bandwidth

We observe that standard TCP has the highest convergence speed, and IIAD generates the smoothest but least responsive traffic. It is worth noticing that in this scenario, where significant bandwidth change happens, our proposed algorithm converges much faster than AIMD to the fair share of the bandwidth.

Table 6 gives the convergence time to fairness (T_2). Here we use $\epsilon = 10$ packets (cf. Section 5). The theoretical value is also given in the table for comparison. The following observations can be made from the table.

First, the simulation results agree with the theoretical analysis in the ranking of various protocols except that all measured convergence times are smaller than the corresponding theoretical values. This is expected since our analysis is based on synchronized feedback assumption, and routers that do not differentiate among flows when dropping packets. In contrast, in the simulation, we use RED, so flows with larger window sizes would see more packet drops. In other words, RED helps the convergence speed to fairness.

Second, SIMD benefits from RED much more than other schemes. The T_2 value from simulations is much smaller than the value obtained from analysis (shown in boldface). This is because RED allows SIMD flows with smaller windows to experience fewer packet losses, which gives them a better chance to become more aggressive[9]. On the contrary, AIMD does not fully capitalize on the random loss property of RED since its window increase rate does not change. As a result, SIMD converges to fairness much faster.

6.3.2 Convergence to Efficiency ($W_1 < W_2 < \frac{W}{2}$)

To create such scenario, we let $c_1 = 11$, $c_2 = 3$, $x = 6$Mbps, $y = 10$Mbps. So initially the bottleneck link for flow (s1,r1) is ($b1$,$b2$), and for flow (s2,r2) the bottleneck is ($b3$,$b4$). But at time 400, both of them switch to link ($b2$, $b3$). Roughly, we have $W \approx 110$, $W_1 \approx 10$, and $W_2 \approx 30$. We can then study T_1, the convergence time to efficiency of different control schemes. Figure 18 plots the transient behavior of different protocols.

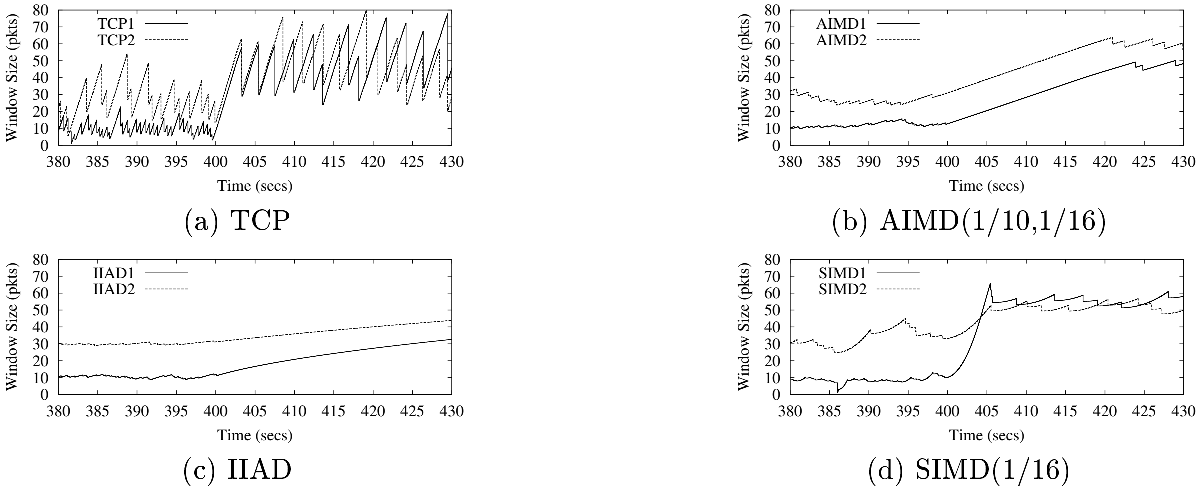


Figure 18: Two flows converge to fair share of bandwidth

The advantage of our SIMD algorithm is more pronounced in this scenario. TCP is still the fastest responding protocol, but still at the expense of high variability in steady state. In addition, general AIMD suffers from the problem of convergence efficiency, i.e, all flows have the same window increments, so before packet loss happens, they increase their congestion windows at the same rate and thus do not efficiently converge to the fair share. On the contrary, our SIMD algorithm allows the two competing flows to quickly transit to the fair steady state, since the flow with smaller window grows more aggressive than the one with larger window. IIAD takes a much longer time to converge due to its inherent weak aggressiveness (sub-linear increase).

[9] Recall that the congestion window size of a SIMD connection increases in proportion to $\frac{1}{w_{\max}}$.

We also give convergence time to efficiency (T_1) in Table 6. Analytical results closely match the simulation results.

7 Related Work

The earliest congestion controls known are Jacobson's TCP Reno [15] and Ramakrishnan and Jain's DECbit scheme [22]. To provide smoother transmission rate than that given by TCP, several TCP-like window-based congestion control mechanisms have been proposed, including the general AIMD [1, 3] and TEAR [2]. These mechanisms use a moderate window decrease parameter to reduce rate variability, meanwhile use a matching window increase parameter to satisfy TCP-friendliness.

Non-linear control was initially considered not robust and not suitable for practical purposes [13]. On the contrary, Bansal and Balakrishnan [4] proposed binomial controls that interact well with TCP. Binomial controls are memory-less in that they use only the current window size in their control rules. Our controls are fundamentally different from memory-less binomial controls. To our knowledge, our proposed scheme presents the first set of window-based TCP-friendly congestion controls that use history information in their control rules. By doing so, our controls improve transient behavior without sacrificing smoothness in steady state.

Another approach to provide smoother transmission rate is equation-based congestion controls [5, 6, 7], first proposed in [23]. In these schemes, the end-systems measure the packet loss rate and round-trip time, and use the TCP-friendly equation [19] to compute the transmission rate. Two comparisons [1, 9] of equation-based and window-based congestion controls have shown that equation-based schemes and window-based AIMD share similar transient behavior but equation-based schemes provide higher smoothness. However, the aggressiveness of equation-based schemes is limited by the nature of rate-based control, which lacks a self-clocking mechanism for overload protection as in window-based control. In [21], Bansal *et al.* integrate self-clocking into the equation-based control to enhance its safety in deployment. They also compared such enhanced control with other slowly-responsive but smooth congestion control schemes such as binomial controls. Their simulation results show that all schemes become less competitive to standard TCP in a highly dynamic environment. They also have the problem of converging slowly to fairness in case of sudden increase/decrease of available bandwidth. Notably, equation-based schemes use more history information up to eight congestion epochs [5]. Therefore, our work is a step toward enhancing transient measures like aggressiveness by exploring the design space between window-based memory-less control schemes and equation-based schemes that make use of longer history.

Much of the literature has focused on the modeling of TCP congestion control [24, 25, 26, 27, 28, 19, 29]. Ott *et al.* showed that if packet losses are independent with small probability p, the average window size and long-term throughput are of the order of $1/\sqrt{p}$. Lakshman *et al.* [26] studied the properties of TCP in a regime where the bandwidth-delay product is high and losses are random. In [27], Mathis *et al.* studied the relationship between TCP throughput and packet loss rate when TCP is in congestion avoidance mode and came up with the well-known TCP-friendly equation. Padhye *et al.* [19] extended this method and used a stochastic model that also captures the effect of TCP's timeout mechanism on throughput. Altman *et al.* [24] analyze TCP throughput under a more general loss process which is assumed to be stationary. The model thus can account for any correlation and inter-loss time distributions. Recently, Low *et al.* [29] presented a duality model of TCP Vegas congestion control mechanism [30].

8 Conclusions

We proposed a spectrum of TCP-like window-based congestion controls. Unlike memory-less controls such as AIMD and binomial controls, our controls utilize history information. They are TCP-friendly and TCP-compatible under RED queue management. They possess different smoothness, aggressiveness, and responsiveness tradeoffs. Thus instances from our spectrum can be chosen as the transport schemes of various applications, for example, streaming applications on the Internet which are required to be TCP-friendly and need smoothness of transmission rates. We conducted extensive simulations using the *ns* simulator. In particular, we presented simulation results of SIMD, AIMD, and AIAD as special instances. Analysis and simulation were used to demonstrate the TCP-friendliness and TCP-compatibility of our controls, the possible tradeoffs among smoothness, aggressiveness, and responsiveness, as well as better convergence behavior of our SIMD instance. The code for our *ns* implementations and the simulation scripts used for this paper are available on-line [31].

To summarize, most encouragingly, in a new design space where control rules use history information, window-based congestion control mechanisms can be TCP-friendly, and still provide smoothness as well as better transient behavior. They can solve the problem raised by slowly-responsive congestion controls. Given that equation-based congestion control schemes use longer history, we believe comparisons between equation-based schemes and our scheme remain an interesting future work.

References

[1] Sally Floyd, Mark Handley, and Jitendra Padhye, "A comparison of equation-based and AIMD congestion control. http://www.aciri.org/floyd/papers.html," May 2000.

[2] Injong Rhee, Volkan Ozdemir, and Yung Yi, "TEAR: TCP Emulation At Receivers – flow control for multimedia streaming," Tech. Rep., Department of Computer Science, North Carolina State University, April 2000.

[3] Y. Richard Yang and Simon S. Lam, "General AIMD congestion control," in *Proceedings of ICNP*, November 2000.

[4] Deepak Bansal and Hari Balakrishnan, "Binomial congestion control algorithms," in *Proceedings of INFOCOM*, April 2001.

[5] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "Equation-based congestion control for unicast applications," in *Proceedings of SIGCOMM*, August 2000.

[6] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," in *Proceedings of NOSSDAV*, June 1999.

[7] Wai-Tian Tan and Avideh Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Transactions on Multimedia*, vol. 1(2), pp. 172–186, June 1999.

[8] Sally Floyd and Kevin Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Transactions on Networking*, vol. 7(4), pp. 458–472, August 1999.

[9] Y. Richard Yang, Min Sik Kim, and Simon S. Lam, "Transient behavior of TCP-friendly congestion control protocols," in *Proceedings of INFOCOM*, April 2001.

[10] Shudong Jin, Liang Guo, Ibrahim Matta, and Azer Bestavros, "TCP-friendly SIMD congestion control and its convergence behavior," in *Proceedings of ICNP*, November 2001.

[11] E. Amir et al, "UCB/LBNL/VINT Network Simulator - ns (version 2)," Available at http://http://www.isi.edu/nsnam/ns/.

[12] Shudong Jin, Liang Guo, Ibrahim Matta, and Azer Bestavros, "A spectrum of TCP-friendly window-based congestion control algorithms," Tech. Rep. BU-CS-2001-015, Computer Science Department, Boston University, July 2001, Available at http://www.cs.bu.edu/techreports/2001-015-spectrum-tcp-friendly.ps.Z.

[13] Dah-Ming Chiu and Raj Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, 1989.

[14] Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1(4), pp. 393–417, August 1993.

[15] Van Jacobson, "Congestion avoidance and control," in *Proceedings of SIGCOMM*, August 1988.

[16] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith, "Tuning RED for Web Traffic," in *Proceedings SIGCOMM*, Stockholm, Sweden, Aug.-Sep. 2000.

[17] Sally Floyd, "Recommendation on using the "gentle_" variant of RED," http://www.aciri.org/floyd/red/gentle.html, March 2000.

[18] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," Internet RFC 2018, April 1996.

[19] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of SIGCOMM*, 1998.

[20] Cooperative Association for Internet Data Analysis, "The CAIDA Website," http://www.caida.org.

[21] Deepak Bansal, Hari Balakrishnan, and Sally Floyd, "Dynamic behavior of slowly-responsive congestion control algorithms," in *Proceedings of SIGCOMM*, August 2001.

[22] K. Ramakrishnan and R. Jain, "Congestion avoidance in computer networks with a connectionless network layer: Part IV: A selective binary feedback scheme for general topologies," Tech. Rep., DEC, August 1987.

[23] J. Mahdavi and Sally Floyd, "TCP-friendly unicast rate-based flow control. Note sent to end2end-interest mailing list," 1997.

[24] Eitan Altman, Konstantin Avrachenkov, and Chadi Barakat, "A stochastic model of TCP/IP with stationary random losses," in *Proceedings of SIGCOMM*, August 2000.

[25] Sally Floyd, "Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic," *Computer Communication Review*, vol. 21(5), August 1991.

[26] T. V. Lakshman and Upamanyu Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. on Networking*, vol. 5(3), 1997.

[27] M. Mathis, J. Semske, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithms," *Computer Communication Review*, vol. 27(3), July 1997.

[28] Teunis J. Ott, J.H.B. Kemperman, and Matt Mathis, "The stationary behavior of ideal TCP congestion avoidance. http://www.argreenhouse.com/papers/tjo," 1996.

[29] Steven H. Low, Larry Peterson, and Limin Wang, "Understanding TCP Vegas: A duality model," in *Proceedings of SIGMETRICS*, June 2001.

[30] Lawrence S. Brakmo and Larry L.Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13(8), October 1995.

[31] Shudong Jin, Liang Guo, Ibrahim Matta, and Azer Bestavros, "Simulations for Stateful TCP Congestion Control," Available at http://csr.bu.edu/simd/sims.html.

Appendix

A TCP-friendliness of Our Control Scheme

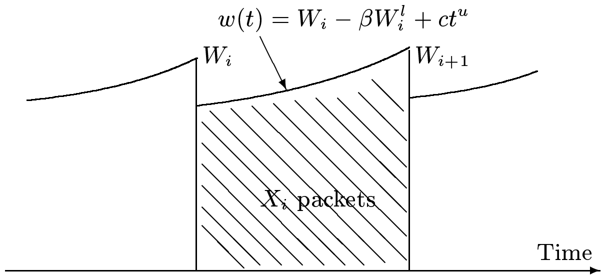


Figure 19: Window increases with time, and decreases on packet losses.

This appendix explains our choice of α in Equation (4), or equivalently, the choice of c in Equation (5) to make our control scheme TCP-friendly. We assume packet losses occur randomly with a fixed probability p, and the window size variation is small. We do not consider the effect of TCP's timeout mechanisms.

Consider many congestion epochs where the window increases and decreases alternately in steady state, as shown in Figure 19. Let W_i be the window size in the beginning of the i^{th} epoch. In this epoch, the window size is decreased to $W_i - \beta W_i^l$, then increased by, say I_i packets, to W_{i+1} before the first packet loss happens. Assume X_i packets are sent successfully in this epoch. Before we consider random losses, it will be helpful to consider periodic losses first.

Periodic Losses

Under a periodic loss model, the window size increase and decrease are deterministic. Both W_i and X_i are constants, denoted as W and X, respectively. I_i is a constant equal to βW^l.

Given the window increase function (1) in Section 2, we can compute the duration (in RTTs) of each congestion epoch:

$$T = (\frac{\beta W^l}{c})^{1/u},$$

and the number of packets in each epoch is given by:

$$\begin{aligned} X &= \int_0^T (W - \beta W^l + ct^u)dt \\ &= (W - \beta W^l)T + \frac{c}{u+1}T^{u+1}. \end{aligned}$$

For the congestion control to be TCP-friendly, the throughput and loss rate relationship must hold. Without considering the effect of TCP's timeout mechanisms, the relationship is $\lambda = \sqrt{3/2}/(R\sqrt{p})$, where λ is the average throughput and R is the round-trip time. We have $\lambda = \frac{X}{TR}$, i.e., average throughput is the number of packets between two consecutive losses divided by the time (in seconds) between the two losses. We also have $p = \frac{1}{X}$. Plugging them into the (λ, p) relationship, we get

$$c = (\frac{3}{2(1 - \frac{1}{k+2}\beta W^{l-1})})^{\frac{1}{k+1}} \beta W^{l - \frac{1}{k+1}}. \tag{9}$$

Notice that here $w_{\max}$ is equal to W, by definition. Therefore, under the periodic loss model, this definition satisfies TCP-friendliness.

Random Losses

Now we consider a random loss model where the losses are Bernoulli trails: packets are dropped uniformly with a fixed probability p. Consider the random process $\{X_i\}$ where X_i is the number of packets sent in the i^{th} epoch up to but not including the first packet lost. Given the random loss model, the probability that j packets are acknowledged successfully before the first loss is

$$\begin{aligned} P[X_i = j] &= (1-p)^j p, \quad j = 0, 1, 2, ... \\ &\approx pe^{-pj}, \quad p \ll 1 \end{aligned}$$

Let T_i denote the number of rounds between two consecutive loss events. T_i can be computed by X_i divided by the average window size in the i^{th} epoch $\overline{w_i}$, i.e., $T_i = X_i/\overline{w_i}$. Using (1), this results in a window increase of size

$$I_i \approx c(\frac{X_i}{\overline{w_i}})^u.$$

Computing $E[I_i]$ is difficult since X_i and $\overline{w_i}$ are correlated. However, when the window size variation is small enough, we ignore such correlation and use the time-average window size $\overline{w}$ to approximate $\overline{w_i}$. Therefore,

$$I_i \approx c(\frac{X_i}{\overline{w}})^u.$$

Then the expected window increase is:

$$\begin{aligned} E[I_i] &= \sum_{j=0}^{\infty} I_i P[X_i = j] \\ &\approx \sum_{j=0}^{\infty} c(\frac{j}{\overline{w}})^u (1-p)^j p \\ &\approx \int_0^{\infty} c(\frac{x}{\overline{w}})^u pe^{-px} dx \\ &= \frac{c\Gamma(u+1)}{(p\overline{w})^u}, \end{aligned} \tag{10}$$

Note that, under the periodic loss model, $X_i = 1/p$, and $T_i = X_i/\overline{w} = \frac{1}{p\overline{w}}$. Therefore,

$$E[I_i] = \frac{c}{(p\overline{w})^u}. \tag{11}$$

For TCP-friendliness, we need to equalize the expected window increases $E[I_i]$ under both loss models. In steady state, the expected increase of the window size is equal to the expected decrease of the window size. Under both loss models, the expected decreases of the window size are roughly equal, given the same loss rate and roughly the same average window size. Therefore, we need only to equalize the expected increases under both loss models. Noticing the only difference between (10) and (11) is a factor of $\Gamma(u+1)$, we only adjust the definition in (9). Thus, we get Equation (5), and equivalently, Equation (4).

Considering that the random loss model is obviously more realistic, we use the definition in Equation (4) and (5) in this paper. In Section 6, we use simulations to validate the TCP-friendliness of SIMD under a wide range of loss rate.

B Implementation

To implement our SIMD algorithm, we only need to change the way the congestion window is updated in standard TCP according to Equation (3). However, since now we need to know the value of the congestion window after the last packet loss, we add a special variable w_0 to record this value. We then divide the increment in each RTT by the current window size to approximate the window increment rule upon each acknowledgment packet. For example, for SIMD(β), we have the following equation:

$$w_{new} = w_{old} + \alpha \frac{\sqrt{w_{old} - w_0}}{w_{old}}, \tag{12}$$

where α is given in Equation (4). Note that $w_0 = w_{max}(1-\beta)$, where w_{max} is the window size right before the loss is detected.

There's one problem with this approximation rule: for the first acknowledgment, we have to use some other equation since the current window size $w_t = w_0$ and that will make the increment to be zero. We solved this problem by noticing that since $w(t) = w_0 + ct^2$, we have $w(1) - w_0 = c$. Thus, upon receiving the first ACK packet, we increment the window as:

$$w_{new} = w_0 + c/w_0 = w_0 + (\frac{\alpha}{2})^2/w_0$$

The value of w_0 is reset to the current congestion window size whenever the congestion window is decreased. And the decrement rule is as follows:

$$w_{new} = w_{old} - \beta w_{old}$$