

# 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

February 2, 2001

Revised on April 27, 2001

Technical Report BUCS-TR-2001-015

## Abstract

The increased diversity of Internet application requirements has spurred recent interests 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 parameterization of these control rules is done so as to ensure that the resulting protocol is TCP-friendly in terms of the relationship between throughput and loss rate.

In this paper, we define a new spectrum of window-based congestion control algorithms that are TCP-friendly as well as TCP-compatible under RED. Contrary to previous memory-less controls, our algorithms utilize history information in their control rules. Our proposed algorithms have two salient features: (1) They enable a wider region of TCP-friendliness, and thus more flexibility in trading off among smoothness, aggressiveness, and responsiveness; and (2) they ensure a faster convergence to fairness under a wide range of system conditions. We demonstrate analytically and through extensive *ns* simulations the steady-state and transient behaviors of several instances of this new spectrum of algorithms. In particular, SIMD is one instance in which the congestion window is increased super-linearly with time since the detection of the last loss. Compared to recently proposed TCP-friendly AIMD and binomial algorithms, we demonstrate the superiority of SIMD in: (1) adapting to sudden increases in available bandwidth, while maintaining competitive smoothness and responsiveness; and (2) rapidly converging to fairness and efficiency.

**Keywords:** Congestion Control, TCP-friendly, Fairness, Convergence.

---

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

# 1 Introduction

In a shared network, end-hosts must react to network conditions and adapt their transmission rates to avoid severe congestion [1] while still maintaining high bandwidth utilization. The success of the Internet is due in part to the congestion control mechanisms [16] implemented in the dominant transport layer protocol TCP. A TCP connection uses additive-increase/multiplicative-decrease (AIMD), i.e., it probes available bandwidth by increasing its 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 generalization of TCP-like window-based schemes [13, 27, 31], and equation-based schemes [14, 24, 30]. A common objective of these schemes is to reduce the high variation of TCP’s transmission rate. Such high variation may limit network utilization. In addition, it is not desirable for emerging applications such as real-time streaming applications on the Internet.

It is required that new transport protocols implement congestion control mechanisms that interact well with TCP [12]. That is, they should maintain *TCP-compatibility*, or fairness across connections using different protocols. To provide such fairness, *TCP-friendliness* is necessary, which means the  $(\lambda, p)$  relationship  $\lambda = \sqrt{3/2}/(R\sqrt{p})$  should hold, where  $\lambda$  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* [13, 31] are important indices of a congestion control algorithm. Smoothness measures the variability in transmission rate. Aggressiveness means how fast the connection probes extra bandwidth by opening up its window. Responsiveness means how fast the 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 [31]. An important observation is that there are tradeoffs among smoothness, aggressiveness, and responsiveness [13, 31]. Comparisons of TCP AIMD<sup>1</sup>, general AIMD [13, 32], TFRC [14], and TEAR [27] indicated that higher smoothness typically means less aggressiveness and responsiveness.

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 information while equation-based schemes do so. Could one explore the design space between these two? Second, can one provide more choices for 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* high aggressiveness when network conditions change drastically (e.g., when there is a sudden increase in available bandwidth)? Finally, can one improve the convergence behavior to fairness and efficiency by using history in window-based congestion control algorithms?

We provide answers to these questions. Specifically, this paper studies TCP-like window-based congestion control algorithms that employ history information, in addition to current window size. The only history used in our schemes is the window size at the time of detecting the last loss. This allows a much broader exploration of TCP-friendly congestion control algorithms than memory-less AIMD and binomial schemes [3]. To this end, we propose a new spectrum of window-based congestion control algorithms. Two salient features of our algorithms are their high smoothness in steady state while reacting to sudden changes in network conditions promptly, and their better convergence behavior. We demonstrate that connections using our algorithms can converge to fairness and efficiency faster than memory-less AIMD and binomial algorithms. We extensively study an instance called SIMD (Square-Increase/Multiplicative-Decrease), which decreases the window size multiplicatively, but increases the window in proportion to

---

<sup>1</sup>We use  $\text{AIMD}(\alpha, \beta)$  to refer to the general AIMD with additive constant  $\alpha$  and multiplicative decrease parameter  $\beta$ . The term *TCP AIMD* refers to  $\text{AIMD}(1, 0.5)$  or standard TCP. For simplicity, we also use *AIMD* for the general case.

the *square* of the time elapsed since the detection of the last loss event.

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 algorithms improve the transient behavior by using history. Compared to equation-based schemes, our algorithms 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 algorithms in Section 2, and define our TCP-friendly algorithms in Section 3. We analyze their convergence behavior in Section 4, and the tradeoffs among smoothness, aggressiveness, and responsiveness in Section 5. Our extensive simulations using *ns* [9] and the results are described in Section 6. We revisit related work in Section 7 and finally conclude the paper. Appendices A through E provide analysis and implementation details.

## 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*. The congestion control scheme defines one control rule for window increase, and another rule for window decrease. For example, AIMD (additive-increase/multiplicative-decrease) uses the following linear control rules:

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

where  $w_t$  is the window size at time  $t$ ,  $R$  is the round-trip time, and  $\delta$  is the time to detect packet loss since the last window update. 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 when a packet loss event is detected. Binomial algorithms [3] generalize AIMD with non-linear controls. They use the following control rules:

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

That is, binomial algorithms generalize additive-increase by increasing inversely proportional to a power  $k$  of the current window (for AIMD,  $k = 0$ ), and generalize multiplicative-decrease by decreasing proportional to a power  $l$  of the current window (for AIMD,  $l = 1$ ).

We say that AIMD and binomial algorithms are memory-less since the increase and decrease rules use only the current window size  $w_t$  and constants ( $\alpha$ ,  $\beta$ ,  $k$ , 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 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 TCP,  $w_0 = (1 - \beta)w_{max}$ . Henceforth, for clarity, we use both  $w_{max}$  and  $w_0$ .<sup>2</sup>

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

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

---

<sup>2</sup>When TCP slow start ends and 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.

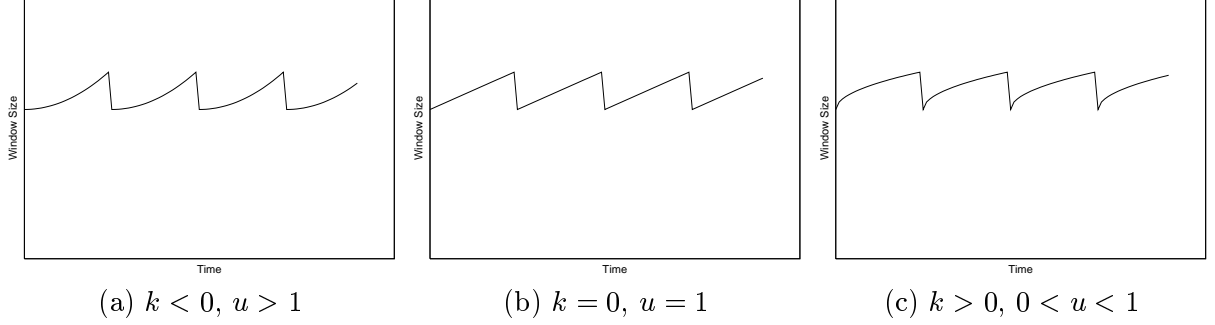


Figure 1: Different increase patterns of congestion window.

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,  $w_0 = w(0)$ .  $c$  depends on  $w_0$  but is independent of  $t$ . In other words, how aggressive the congestion window is increased also depends on recent congestion state. The above increase function is equivalent to the following window increase rule:

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

where  $k > -1$  and  $\alpha$  is independent of  $t$ . In particular<sup>3</sup>,  $u = 1/(k+1)$  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 congestion control algorithm 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:

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

---

<sup>3</sup>**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):

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

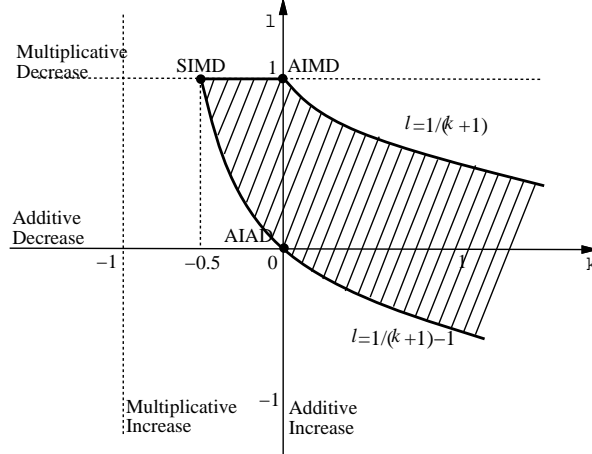


Figure 2: A spectrum of TCP-friendly congestion control algorithms using history.

Here we use the same decrease rule as binomial algorithms. 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  $l \leq 1$ , since otherwise  $(w_t - \beta w_t^l)$  can be negative when  $w_t$  is large enough.

Before further elaboration, we state several properties of our new algorithms:

- First, we show that our algorithms can be TCP-friendly by appropriately defining  $\alpha$  as a function of the constant  $\beta$  and the state variable  $w_{max}$ . For this purpose, we consider a network in steady state and assume random packet losses.
- Second, under the synchronized feedback model used by Chiu and Jain [6], we derive conditions for our algorithms to possess the convergence-to-fairness property. In particular, in the  $(k, l)$  space shown in Figure 2, the spectrum (the shaded area) satisfies this property.
- Third, the algorithms in this new spectrum possess different tradeoffs among smoothness, aggressiveness, and responsiveness. SIMD is an instance of the spectrum in Figure 2. For SIMD,  $k = -0.5$  and  $l = 1$ . It is the most aggressive instance: its window size increases in proportion to the *square* of the time elapsed since the detection of the last loss. It is also a responsive instance at different window scales: its window size decreases multiplicatively upon the detection of packet losses. It can achieve high smoothness in steady state by using a small decrease factor  $\beta$ .

We need also to point out that our algorithms are radically different from the binomial algorithms [3]. Binomial algorithms generalize AIMD, but they are still in the memory-less space. On the contrary, our algorithms are in another space that uses history. Therefore, binomial algorithms can not 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  $\alpha$  to make our congestion control

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

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

scheme TCP-friendly:

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

where  $\Gamma(\cdot)$  is the Gamma function. According to Section 2,  $c$  in Equation (1) is defined after  $\alpha$  and we have:

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

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

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

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

That is,  $\alpha$  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  $\alpha$  in Equation (6) and the definition of  $c$  in Equation (7). When  $k = 0$  and  $l = 1$ , from (4) we have  $\alpha_{AIMD} = 3\beta/(2 - \beta)$ . If  $\beta \ll 1$ ,  $\alpha_{AIMD} \approx 3\beta/2$ . It degenerates to the memory-less TCP-friendly AIMD control [13]. When  $k = -0.5$  and  $l = 1$ ,

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

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

As we mentioned earlier, recently proposed binomial algorithms do not belong to our spectrum since they are in a memory-less space. However, it is interesting to note that TCP-friendly binomial algorithms

can be *projected* on the line where  $k = 0$  and  $0 \leq l \leq 1$  in Figure 2. For example, a special case of TCP-friendly binomial algorithms IIAD (Inverse-Increase/Additive-Decrease) can be compared to our special case  $k = l = 0$ . IIAD has the following control rules:

$$\begin{aligned} \text{Increase : } & w_{t+R} \leftarrow w_t + \frac{3\beta}{2w_t}, \\ \text{Decrease : } & w_{t+\delta} \leftarrow w_t - \beta. \end{aligned}$$

The only difference between IIAD and our special case  $k = l = 0$  is that we use the history  $w_{max}$  instead of the *current* window  $w_t$  in the increase rule. Thus, IIAD increases the window size sub-linearly even if no losses occur, while our algorithm does not slow down the increase if no losses occur. That is, our algorithm is a special AIAD control, except that the additive increase parameter,  $\frac{3\beta}{2w_{max}}$ , is redefined through the history  $w_{max}$  once losses are detected. AIAD is also studied in our simulations. Another special case of binomial algorithms, SQRT, has the following control rules:

$$\begin{aligned} \text{Increase : } & w_{t+R} \leftarrow w_t + \frac{3\beta}{2\sqrt{w_t}}, \\ \text{Decrease : } & w_{t+\delta} \leftarrow w_t - \beta\sqrt{w_t}. \end{aligned}$$

SQRT can be compared to our special case  $k = 0, l = \frac{1}{2}$ . The only difference is, we use the history  $w_{max}$  instead of  $w_t$  in the increase rule. Similar to IIAD, SQRT increases the window size sub-linearly even if no losses occur.

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

## 4 Convergence to Fairness and Efficiency

In this section, we first show conditions for convergence of our congestion control algorithms. Then we show they can have better convergence behavior than memory-less AIMD control.

### 4.1 Convergence Conditions

We adopt the ideal synchronized feedback assumption [6]. 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 [6] 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 3(a), 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

$$\max\left(\frac{x_1}{x_2}, \frac{x_2}{x_1}\right).$$

If the fairness index is closer to unity, then 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 “maximum utilization line” or “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 under-utilized (assuming  $x_1 \leq x_2$  without loss of generality), AIMD increases the resource allocation of both users at the same rate. In Figure 3(a), the trajectory is shown as the line 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 towards the fairness line, finally oscillating around the efficiency line.

For our control scheme, we first observe Equation (1) and the definition of  $c$  in Equation (7). Ignoring the constant factor in (7), we can see that the window size of a connection increases in proportion to  $x_i^{l-1/(k+1)}, i = 1, 2$ . To ensure that the increase trajectory moves towards a more fair point, we can choose

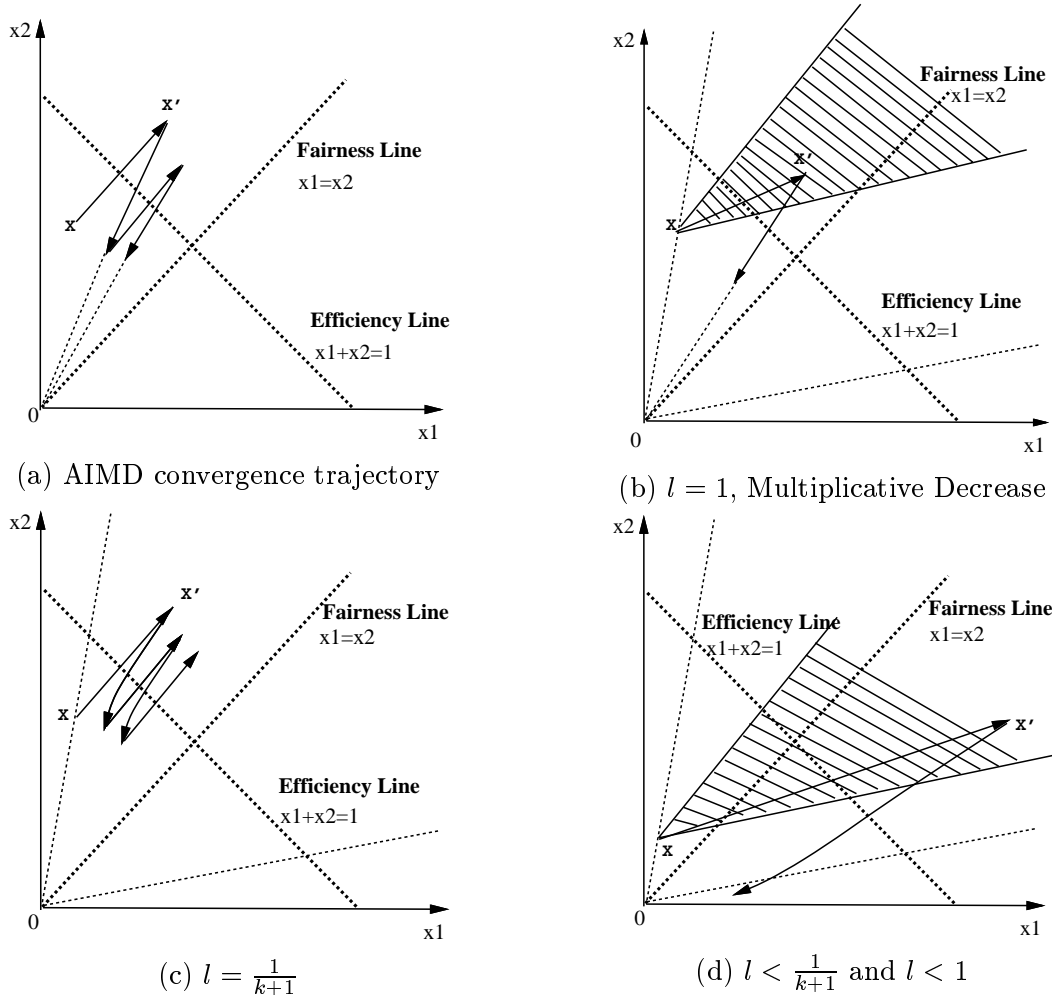


Figure 3: Convergence of our congestion control scheme.

$-1 \leq l - \frac{1}{k+1} \leq 0$ . This condition is shown as the shaded area in Figure 2. Under this condition, the trajectory in Figure 3(b) shows that the system moves inside the shaded area between the  $45^\circ$  additive increase line and the line emanating from  $X(x_1, x_2)$  with slope  $\frac{x_1}{x_2}$ . However, different areas of the spectrum in Figure 2 exhibit different convergence behavior. We consider the following scenarios:

(a)  $l = 1$

This corresponds to multiplicative decrease. Examples include AIMD and SIMD. The decrease trajectory moves towards the origin, as shown in Figure 3(b). Hence the decrease does not change the fairness index. Since the increase has been shown to improve fairness, repeated increases and decreases gradually improve fairness, i.e., reduce the fairness index. Finally, the system stays on the fairness line and oscillates around the efficiency line.

(b)  $l = \frac{1}{k+1}$

In this case, observing Equation (7), we find that  $c$  is *independent* of  $w_{max}$ . This means that the two users increase the resource allocation at the same speed. Although, the increase is not necessarily additive, the increase trajectory follows exactly the additive increase trajectory, as shown in Figure 3(c). Then we consider a decrease after the increase. Since  $0 < l < 1$ , the decrease trajectory is between the last additive increase trajectory and the multiplicative decrease trajectory. Thus, the decrease trajectory intersects the efficiency line at a point that is more fair than the intersection of the increase trajectory

and the efficiency line. Therefore, repeated increases and decreases will move the system towards the fairness line.

**(c) Otherwise**

In this case,  $l < 1$  and the increase trajectory, which depends on  $x_i^{l-1/(k+1)}$ , may cross the fairness line. After the increase trajectory crosses the fairness line, as shown in Figure 3(d), the decrease trajectory may move below the line emanating from the origin with slope  $\frac{x_1}{x_2}$ , hence, to a point more unfair than  $X$ . However, this case occurs only if the increase and decrease trajectories are not bounded. It is reasonable to assume that the overshoot of the increase trajectory and the undershoot of the decrease trajectory are bounded, given the self-clocking nature of window-based controls. Under this assumption, Appendix B gives a condition to ensure the convergence of our scheme when  $l \geq 0$ .

In summary, in both cases (a) and (b), the system converges to fairness. In case (c), the system converges conditionally. However, the condition is easy to satisfy under realistic assumptions.

## 4.2 Convergence Speed

We first intuitively show that our control scheme converges faster than AIMD. Then we analytically show the time for different schemes to bring the difference between two user allocations within a certain bound.

First, to intuitively show that our scheme converges faster than AIMD, we show that the increase trajectory of our control scheme intersects the efficiency line at a point that is usually more fair than that of AIMD.

Let us consider the case when  $l - \frac{1}{k+1} = -1$ . For other cases, the intersection is closer to the intersection of AIMD trajectory, as shown in Figure 4(a), thus it is easier for our scheme to reach a more fair intersection than AIMD.

Let  $X(x_1, x_2)$  be the initial under-utilized allocation,  $x_1 + x_2 < 1$  and assume  $x_1 < x_2$ . Using AIMD,<sup>4</sup> the intersection of the trajectory and the efficiency line is  $(\frac{1+x_1-x_2}{2}, \frac{1-x_1+x_2}{2})$ . Using our control scheme with  $l - \frac{1}{k+1} = -1$ , the intersection is  $(x_1 - x_2 + \frac{x_2}{x_1+x_2}, x_2 - x_1 + \frac{x_1}{x_1+x_2})$ . By comparing the fairness index at these two intersections, we found that our control scheme reaches a more fair intersection if  $x_1 + x_2 > 1/3$ . This condition is shown as area (1) in Figure 4(a). Since the size of area (1) is much larger than that of area (2), we intuitively say our scheme usually converges faster than AIMD.

Then, we analytically compare the convergence time of our control scheme, general AIMD [13, 32], and binomial control scheme [3]. For our control scheme, we choose the case of  $k = -0.5$  and  $l = 1$ , i.e., SIMD as a representative. For binomial control scheme, we choose IIAD (Inverse-Increase/Additive-Decrease) as a representative.<sup>5</sup> The control rules of IIAD are shown in Section 3. We still assume synchronized feedback and use Figure 4(b) to illustrate the process of convergence to fairness. 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  $(W_1,$

<sup>4</sup>We get these two intersections as follows. Let  $\Delta x_i, i = 1, 2$  denote the increase of the  $i^{th}$  user. For AIMD, to get  $\Delta x_1$  and  $\Delta x_2$ , we solve the following:

$$\begin{aligned} (x_1 + \Delta x_1) + (x_2 + \Delta x_2) &= 1, \\ \Delta x_1 &= \Delta x_2. \end{aligned}$$

For SIMD, since the increase  $\Delta x_i$  is inversely proportional to  $x_i$ , we solve the following:

$$\begin{aligned} (x_1 + \Delta x_1) + (x_2 + \Delta x_2) &= 1, \\ \frac{x_1}{x_2} &= \frac{\Delta x_2}{\Delta x_1}. \end{aligned}$$

<sup>5</sup>Another binomial algorithm SQRT with  $k = l = 0.5$  lies between AIMD and IIAD. We expect its behavior to also be between those of AIMD and IIAD.

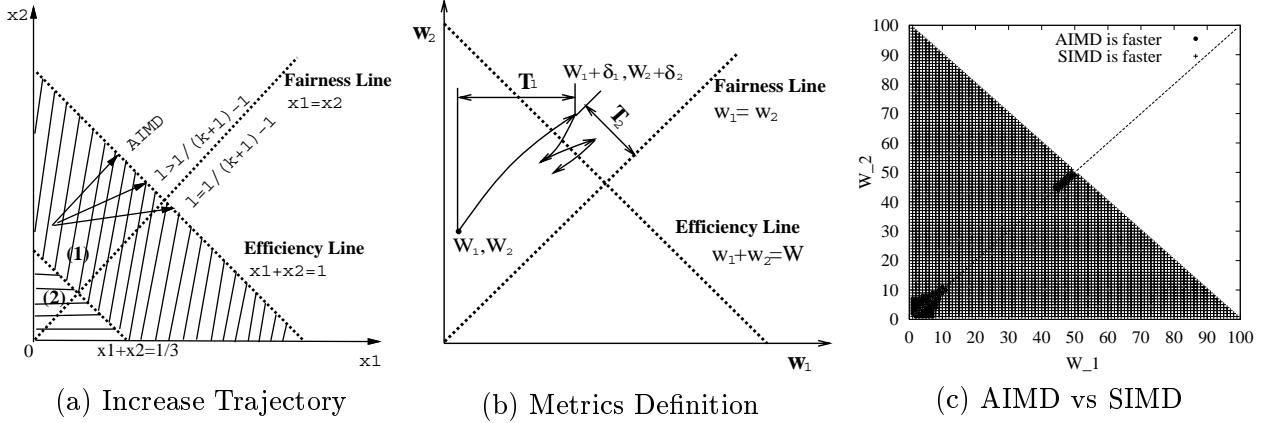


Figure 4: Comparison of convergence speed

$W_2$ ), where  $W_2 \geq W_1$  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 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 (measured in packets) after  $T_1$  as  $\Delta$ . The detailed analysis is given in Appendix C and we only present the main results here in Table 2.

Algorithm	$T_1$ (RTT)	$\Delta$ (pkts)	$T_2$ (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	$\frac{1}{12\beta} \left( \left( \frac{W_2^2 - W_1^2}{W} \right)^2 - 2(W_1^2 + W_2^2) + W^2 \right)$	$\frac{W_2^2 - W_1^2}{W}$	$\frac{W}{3} \log_{1 - 2\beta/W} \frac{\epsilon}{\Delta}$
SIMD	$\frac{2}{3} \left( 1 - \frac{2\beta}{3} \right) \sqrt{\frac{2}{\beta(1 - \beta)}} \sqrt{\frac{W_1 W_2 (W - W_1 - W_2)}{W_1 + W_2}}$	$\left( 2 - \frac{W}{W_1 + W_2} \right) (W_2 - W_1)$	$\frac{\sqrt{2}W}{3} \log_{1 - 2\beta} \frac{\epsilon}{\Delta}$

Table 2: Performance measures on convergence to fairness and efficiency

We also numerically solve the above equations for different initial points. Figure 4(c) 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  $\beta = 2/3$  such that IIAD is TCP-friendly) is much slower than AIMD and SIMD in all cases.

## 5 Tradeoffs among Smoothness, Aggressiveness, and Responsiveness

In this section, we consider important indices of congestion control algorithms other than TCP-friendliness and convergence. These indices are smoothness, aggressiveness, and responsiveness. Smoothness measures the variation of the transmission rate (window size) of a connection. High variation is not desirable. Aggressiveness means how fast a connection probes available bandwidth. Higher aggressiveness implies potentially higher utilization of bandwidth. Responsiveness means how fast a connection decreases its window size in response to increased congestion. Both aggressiveness and responsiveness are measures of transient behavior.

	<b>Smoothness</b>	<b>1/Aggressiveness</b>	<b>1/Responsiveness</b>
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 3: Smoothness, Aggressiveness, and Responsiveness comparisons of AIMD, binomial controls, and our control scheme.

Smoothness can be observed at different time scales [13]. 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 measurement 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 [31]. That is, the available bandwidth is increased by a factor of  $m$ . We define responsiveness as the inverse of the number of loss events necessary for the connection to decrease its window by a substantial amount, in response to a step increase of congestion [31]. That is, a substantial decrease of available bandwidth by a factor of  $m$ .

Table 3 gives the approximations of these indices for the general AIMD, IIAD, SIMD, and AIAD algorithms. More details are given in Appendix D.

Numerical results in Figure 5 shows the tradeoffs among smoothness, aggressiveness, and responsiveness. Figure 5(a) shows the inverse of aggressiveness of AIMD, SIMD, and IIAD as the coefficient of variation varies. Results for AIAD are not shown here since they are similar to those of IIAD except that AIAD has better aggressiveness. Their special cases TCP AIMD, 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 5(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 better aggressiveness (fewer RTTs) than the others, 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 3. 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 maintains much higher aggressiveness than AIMD and IIAD.

Figure 6 shows the same tradeoffs, except that this time we use a larger factor  $m = 5$  for the sudden decrease and increase of available bandwidth. It shows that the advantage of SIMD's aggressiveness is more pronounced. We can also observe from Table 3 that, for SIMD, aggressiveness varies inversely proportional to the square root of  $m$ , and for AIMD and IIAD, aggressiveness varies inversely proportional to  $m$  or even  $m^2$ . Therefore, even larger  $m$  makes SIMD more favorable.

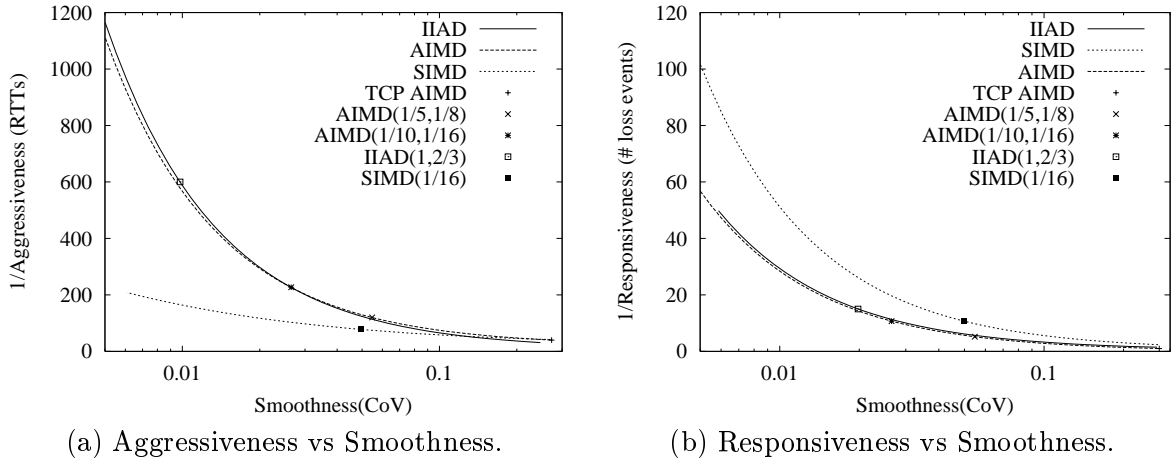


Figure 5: Tradeoffs of smoothness, aggressiveness, and responsiveness. For (a), we assume available bandwidth is doubled. For (b) we assume a step increase of congestion and the window is reduced to half. The initial average window size,  $W$ , before bandwidth changes is 20. The coefficient of variation of IIAD depends on the average window size (the CoVs of AIMD and SIMD do not). We compute its value before transition and its value after transition, and then take the geometric mean.

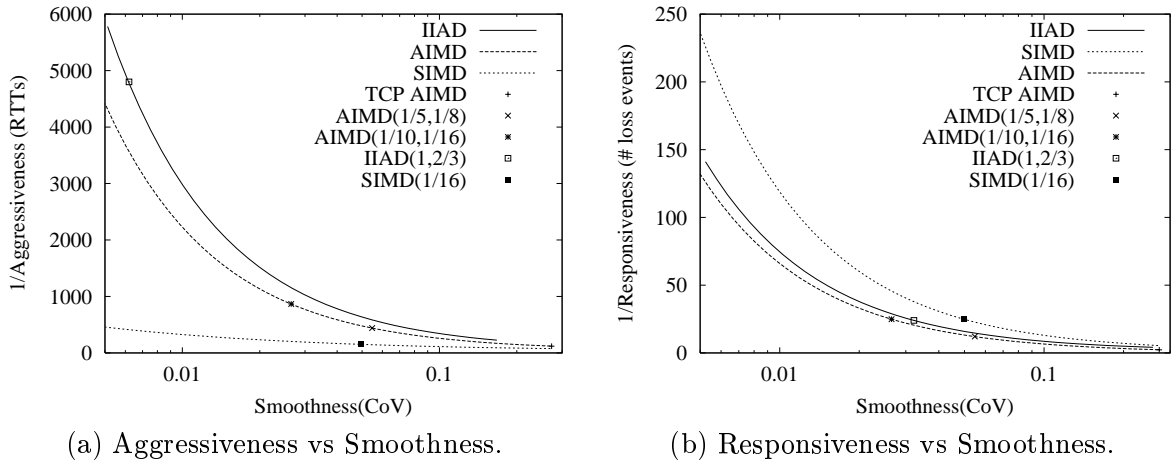


Figure 6: Tradeoffs of smoothness, aggressiveness, and responsiveness. For (a), we assume available bandwidth is increased by a factor of 5. For (b) we assume a step increase of congestion and the average window is reduced to one fifth. The initial average window size,  $W$ , before bandwidth changes is 20. The coefficient of variation of IIAD depends on the average window size. We compute its value before transition and its value after transition, and then take the geometric mean.

## 6 Simulation Results

We use the *ns* simulator [9] to validate that with RED [15] queue management (or randomized dropping), our proposed algorithms, most notably SIMD, are TCP-friendly and TCP-compatible. In addition, we compare SIMD to standard TCP [16], generalized AIMD [32], and IIAD [3], in terms of smoothness, responsiveness, and aggressiveness. We also present results for AIAD.

We also 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 E.

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

Description	Value
Packet size	1000 bytes
Maximum window	128 packets
TCP version	SACK
TCP timer granularity	0.1 seconds
RED queue limit $Q$	$2.5 \times \text{B/W delay product}$
DropTail queue limit	$1.5 \times \text{B/W delay product}$
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

The bottleneck queue size and RED queue parameters are tuned as recommended in [7]. The “gentle\_” option of RED queue is turned on as recommended in [11]. 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 [20] for congestion detection. We also obtained similar results for other mechanisms (e.g. Reno, newReno). We assume no delayed acknowledgments.

### 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 algorithm: 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  $\beta$  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 TCP-friendly equation (9) [23] can reveal one possible explanation of this abnormality.

$$\lambda(p, \alpha, \beta) \approx \min\left(\frac{W_{max}}{R}, \frac{1}{R\sqrt{\frac{2\beta}{\alpha(2-\beta)}}p + T_0 \min(1, 3\sqrt{\frac{\beta(2-\beta)}{2\alpha}}p)p(1 + 32p^2)}\right) \quad (9)$$

When loss rate is low, TCP mainly stays in the *congestion avoidance* stage, and AIMD algorithm dominates the TCP-friendly equation (9), while when loss rate is very high, TCP spends most of its time

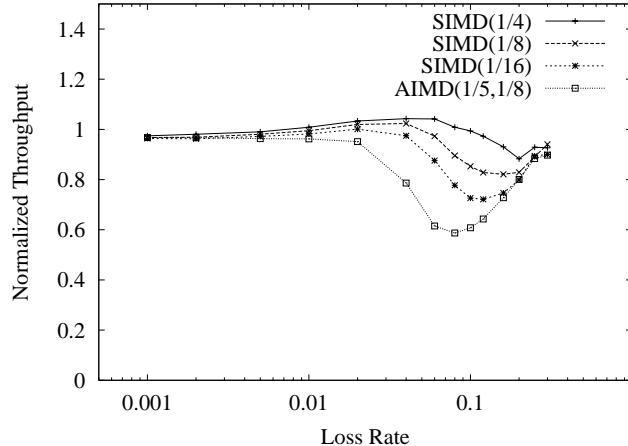


Figure 7: TCP-Friendliness

retransmitting packets, and the *exponential back-off* algorithm dominates equation (9). Since all TCP variants 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  $\alpha$  and  $\beta$  values that would approximate well both congestion avoidance and exponential backoff components of the TCP-friendly equation [32].

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

### 6.1.2 TCP-Compatibility

We use the method described in [13] to test TCP-compatibility.  $n$  SIMD flows and  $n$  standard TCP SACK flows compete for bandwidth over a shared bottleneck link. There are also 4 background TCP flows transmitting packets in the opposite direction to introduce random ACK delays. We consider both RED 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

<sup>6</sup>The weakness of AIMD( $\alpha, \beta$ ) with small  $\beta$  under medium loss conditions is also reported in [13]. The authors try to compensate for the bandwidth loss by increasing the value of  $\alpha$ . However, when loss rate is small (e.g. less than 3%), AIMD with large  $\alpha$  could achieve significantly higher bandwidth than standard TCP and become less TCP-friendly. Therefore, we maintain the theoretical  $\alpha$  values throughout our simulations.

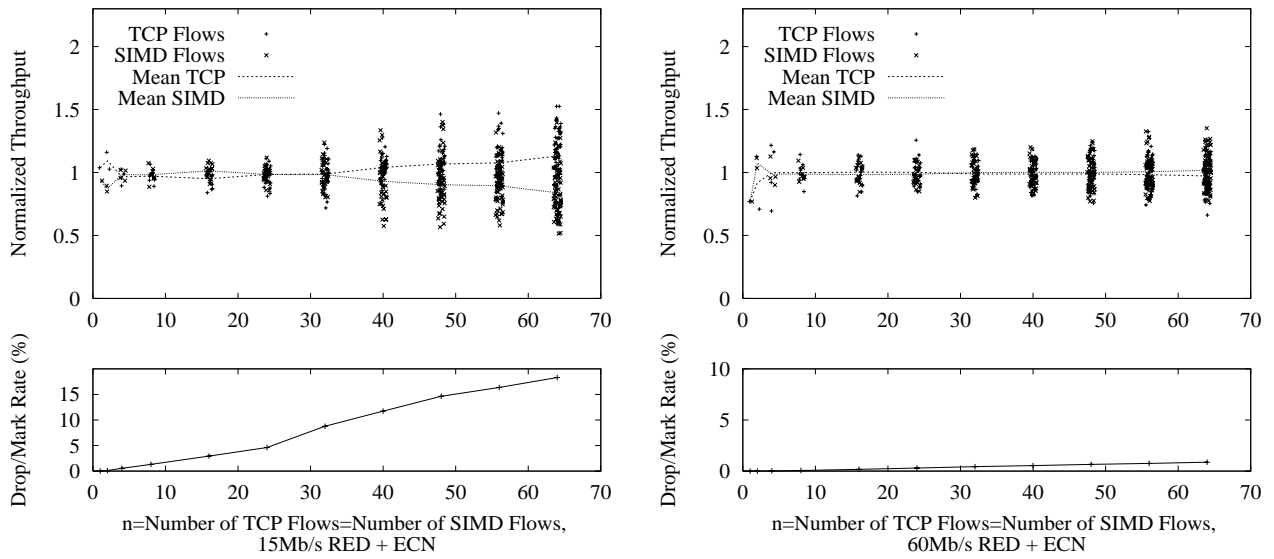


Figure 8: TCP competing with SIMD(1/16), RED with ECN

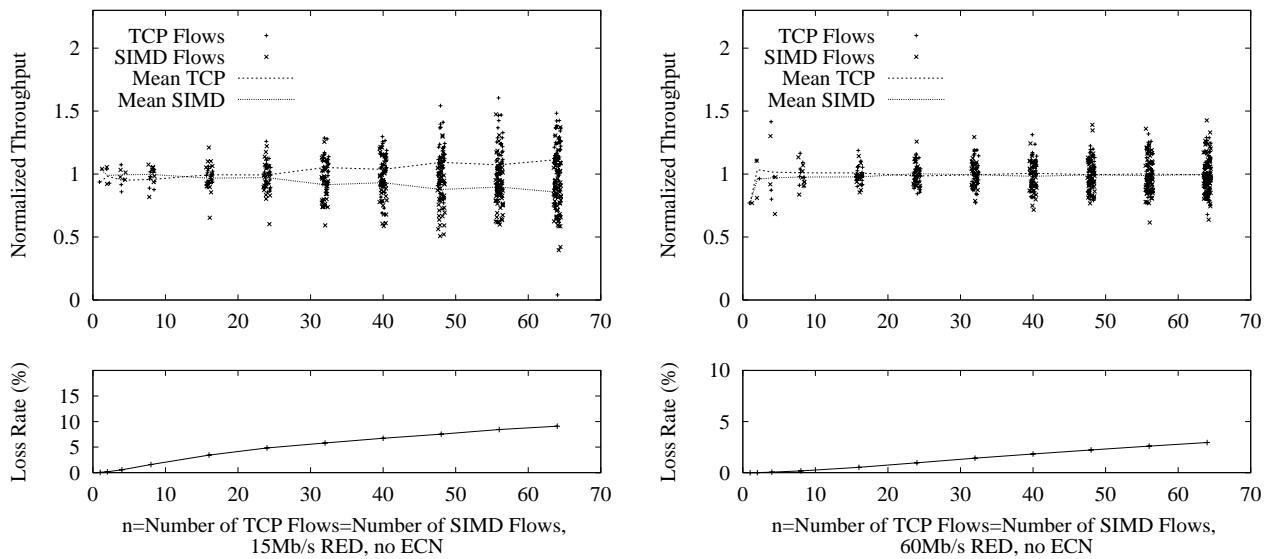


Figure 9: TCP competing with SIMD(1/16), RED without ECN

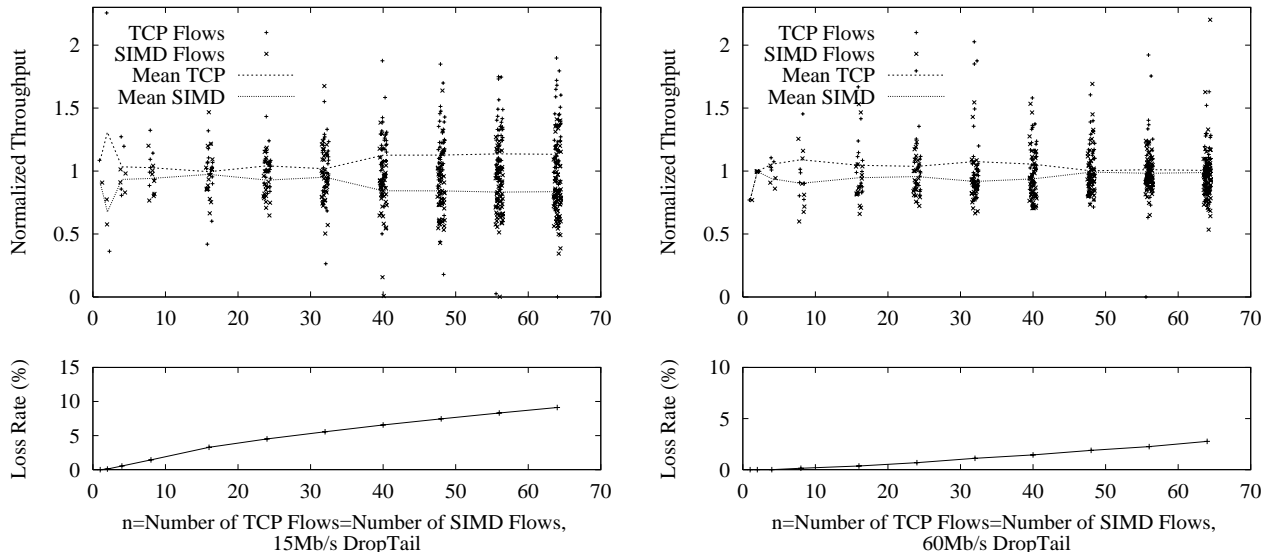


Figure 10: TCP competing with SIMD(1/16), with DropTail

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 (loss rate below 10%), SIMD shows very impressive TCP-compatibility <sup>7</sup>.

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 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 (bursty drops). 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.

For completeness, we also report corresponding results in Figures 11, 12 and 13 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 [13], 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 algorithm is used. We thus focus our simulation on short-term smoothness and use the simulation code contributed by [13] to study the traffic generated by the congestion control algorithms 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 14 we show the case for  $n = 16$  and  $C = 60Mbps$ , which corresponds to roughly 0.3% packet drop rate. The bottleneck queue strategy

<sup>7</sup>Note that in case of 60Mbps link and less than 4 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.

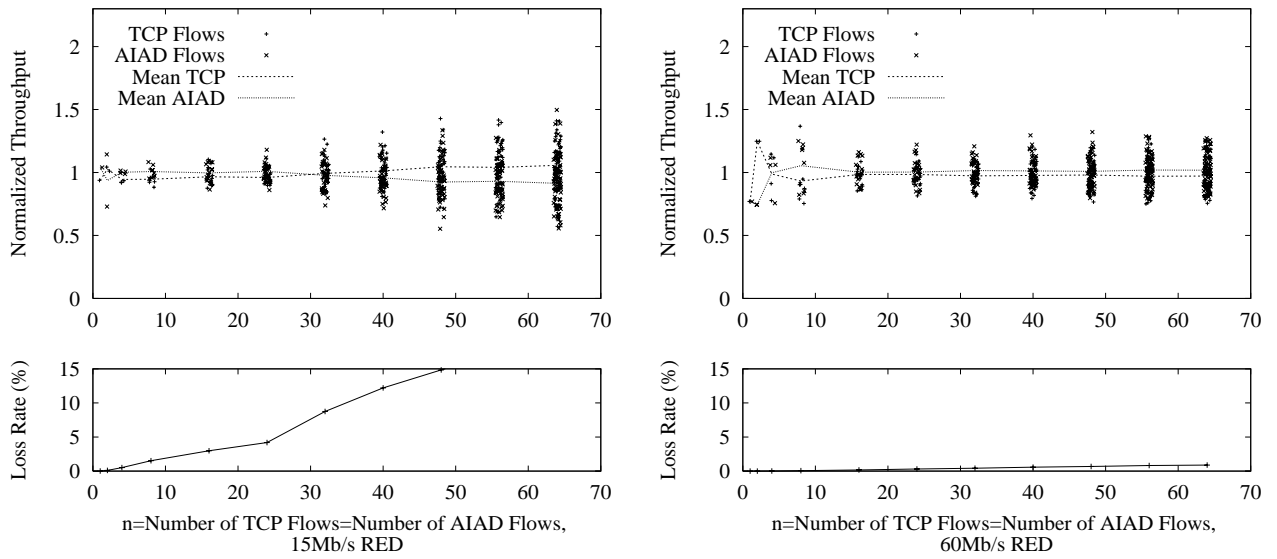


Figure 11: TCP competing with AIAD(2/3), RED with ECN

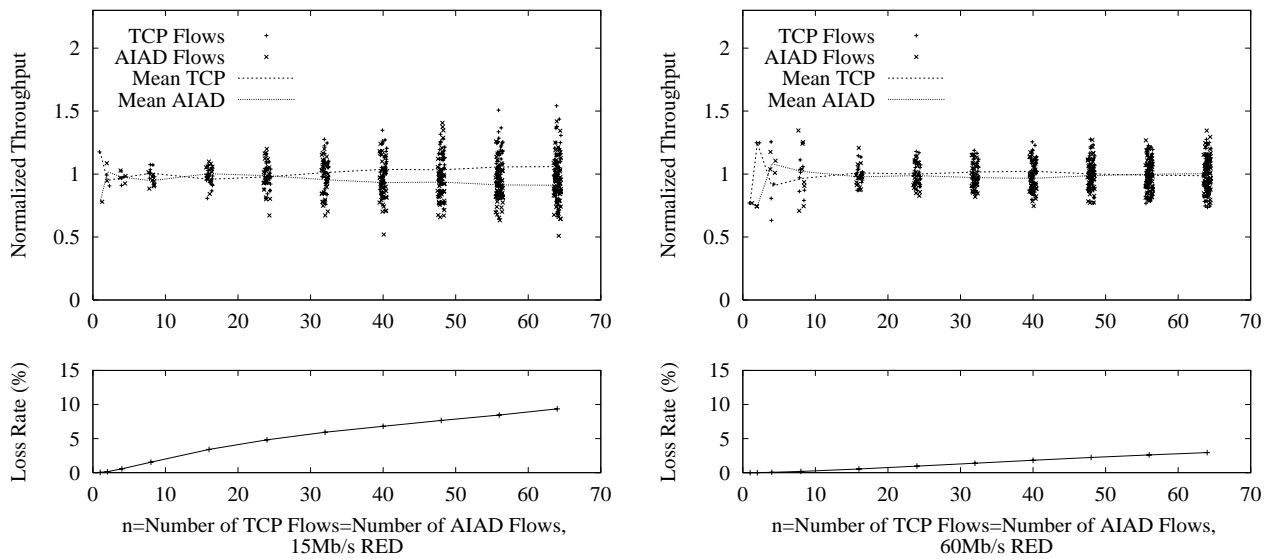


Figure 12: TCP competing with AIAD(2/3), RED without ECN

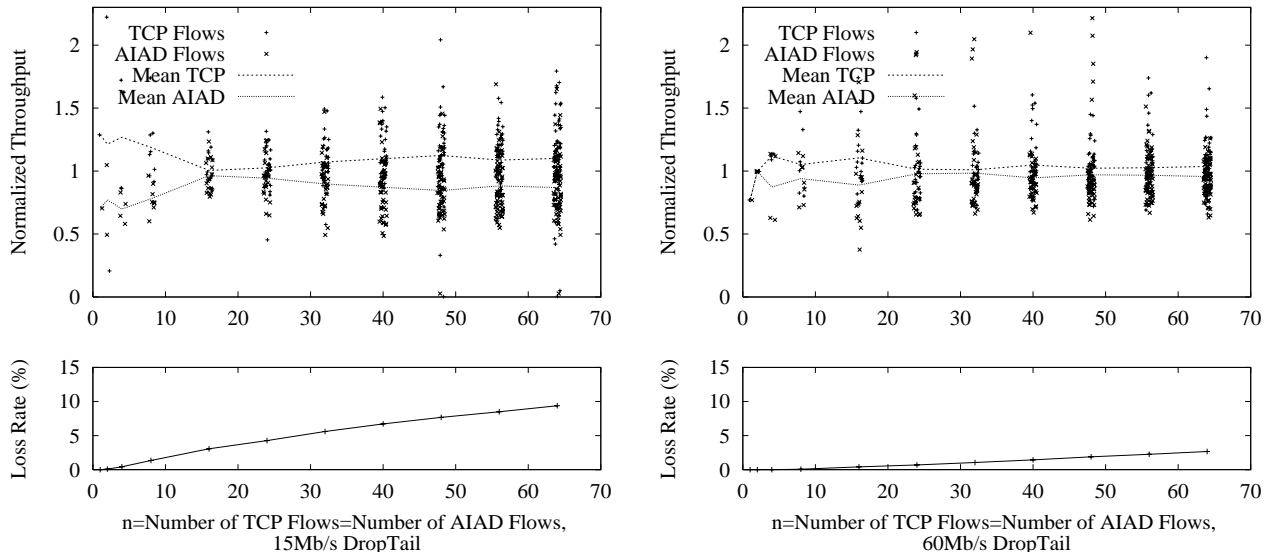


Figure 13: TCP competing with AIAD(2/3), with DropTail

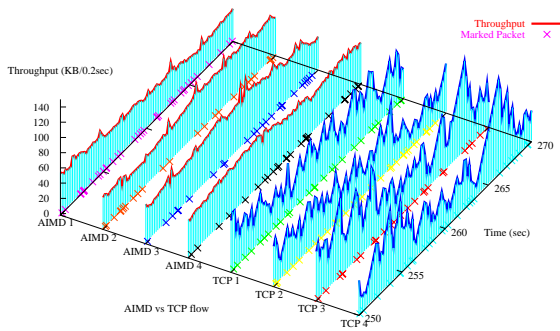
is RED with ECN enabled. Figure 15 shows the same setup with ECN turned off. Each graph shows one flow's throughput on the congested link during the time interval between 250 to 270 seconds of a 500-second simulation. The throughput is averaged over 0.2-second intervals, which correspond to twice a typical round-trip time for this simulation. As in [14], we also plot the time at which a packet is marked (or dropped in Figure 15) at the bottom of each curve.

We can observe from the graphs that all four algorithms AIMD, IIAD, SIMD, and AIAD have roughly the same scale of short-term burstiness with SIMD having a little larger variation. This agrees with our analysis (cf. Section 5). In particular, by plugging in equations of Table 3 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 algorithms (from low to high) is: IIAD (and AIAD), AIMD, and SIMD. Our experiment results show that this is indeed the case.

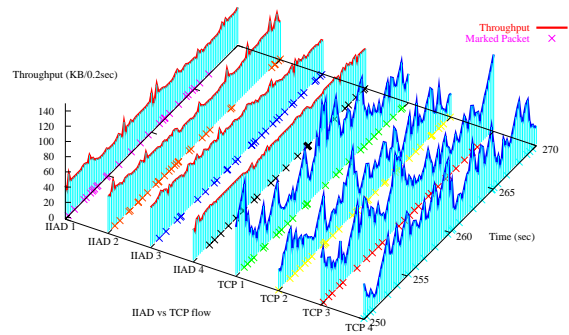
We also decrease  $C$  to 15 Mbps (thus increase the congestion level to nearly 5% loss rate) in another experiment set. The results are shown in Figures 16 and 17.

We observe that the smoothness of all four algorithms 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 [14] can achieve high smoothness even when the loss rate is high.

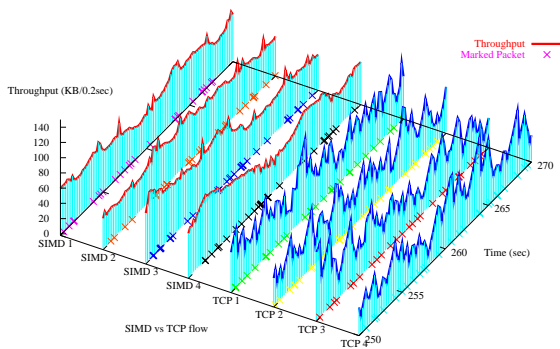
We also observe that the throughput of AIMD degrades significantly. IIAD and AIAD also gets less than their fair share. This is in part due to the reason mentioned in Section 6.1.1, that is, AIMD becomes less competitive than standard TCP in this loss regime. The other reason, we conjecture, is that AIMD algorithm 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



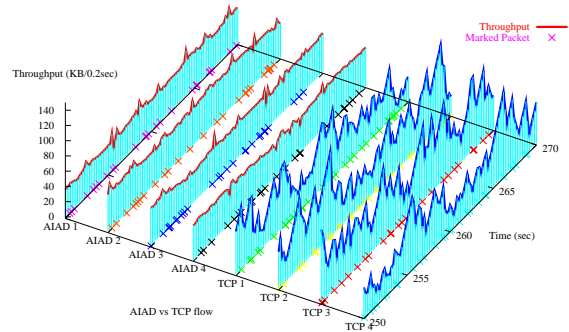
(a) AIMD(1/10,1/16) with TCP



(b) IIAD with TCP

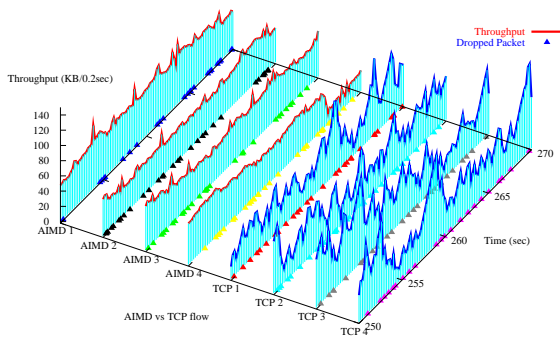


(c) SIMD(1/16) with TCP

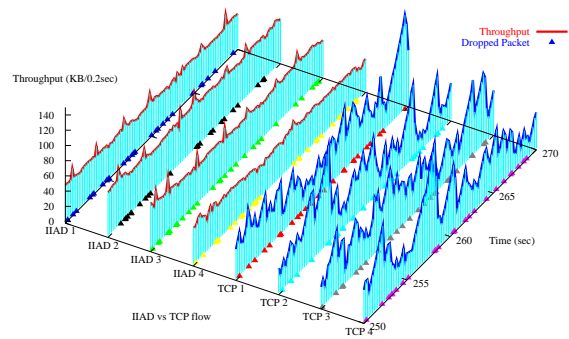


(d) AIAD(2/3) with TCP

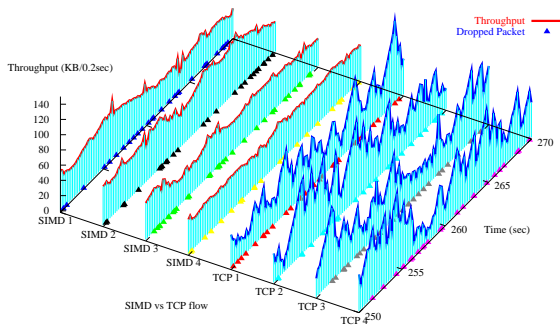
Figure 14: Traffic smoothness, 16 + 16 TCP flows, 60Mbps link, RED with ECN



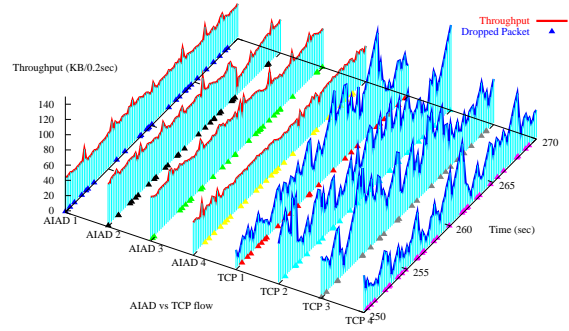
(a) AIMD(1/10,1/16) with TCP



(b) IIAD with TCP

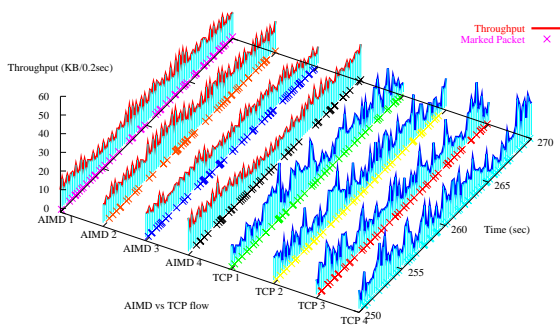


(c) SIMD(1/16) with TCP

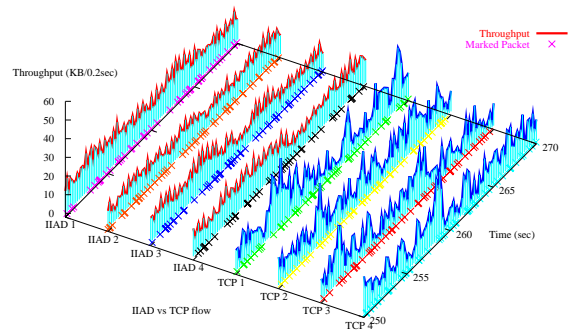


(d) AIAD(2/3) with TCP

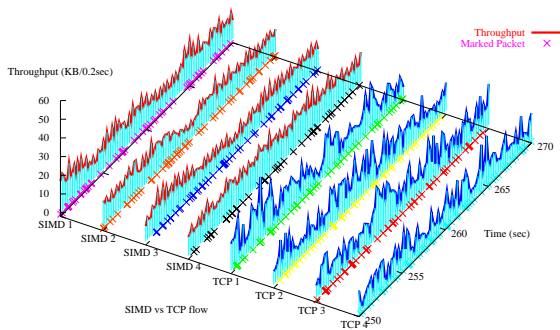
Figure 15: Traffic smoothness, 16 + 16 TCP flows, 60Mbps link, RED without ECN



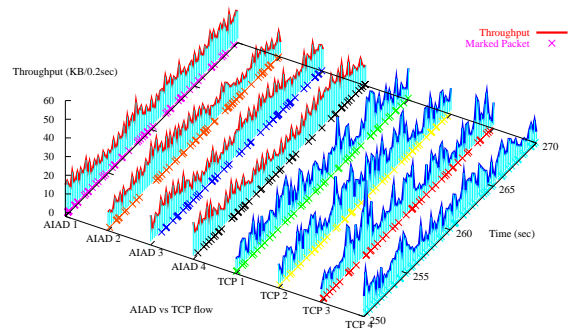
(a) AIMD(1/10,1/16) with TCP



(b) IIAD with TCP

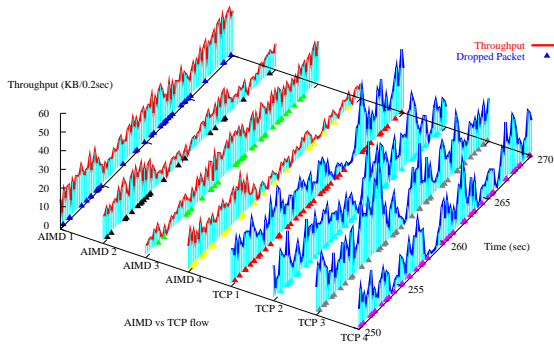


(c) SIMD(1/16) with TCP

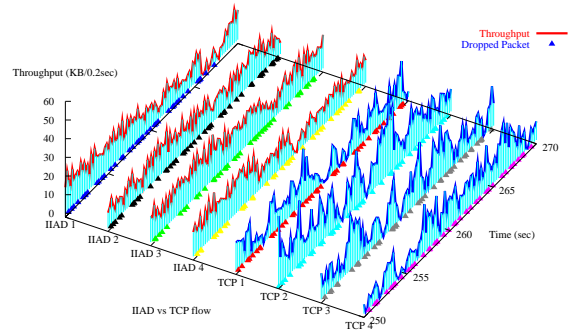


(d) AIAD(2/3) with TCP

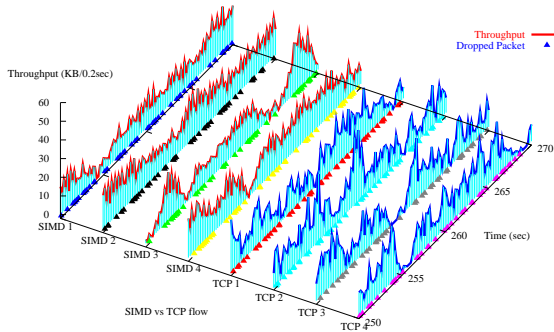
Figure 16: Traffic smoothness, 16 + 16 TCP flows, 15Mbps link, RED with ECN



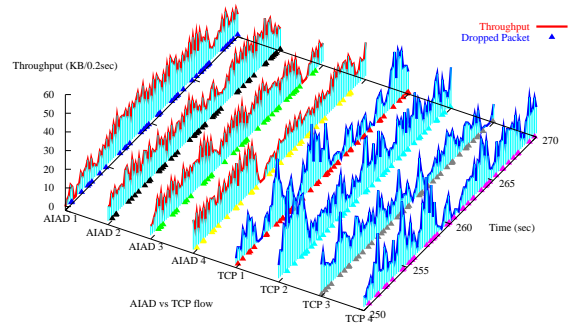
(a) AIMD(1/10,1/16) with TCP



(b) IIAD with TCP



(c) SIMD(1/16) with TCP



(d) AIAD(2/3) with TCP

Figure 17: Traffic smoothness, 16 + 16 TCP flows, 15Mbps link, RED without ECN

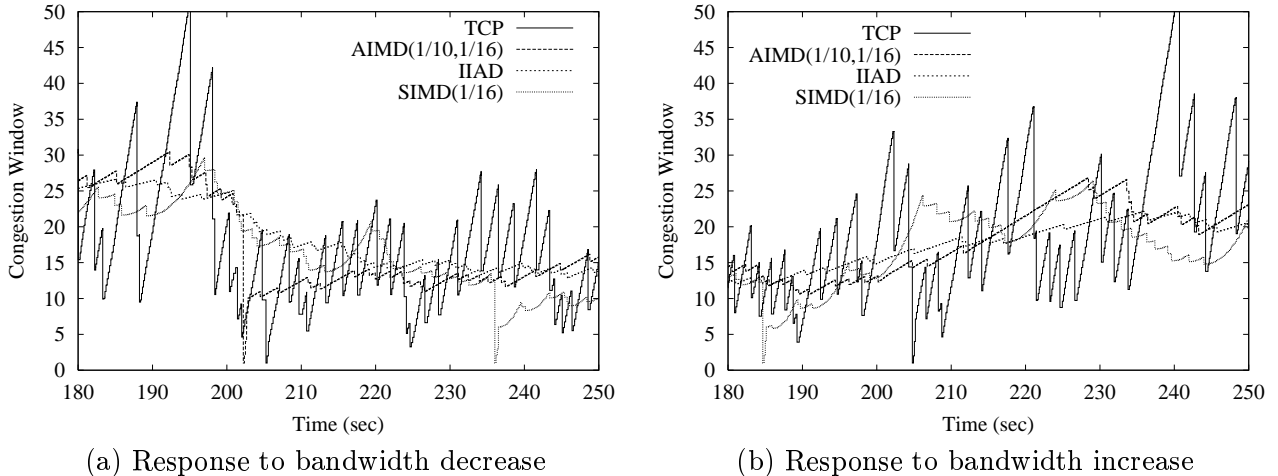


Figure 18: Response to bandwidth variation,  $n = 8$

this problem and can achieve throughput close to TCP in this scenario.

### 6.2.2 Response to step increase of congestion (Responsiveness)

We conduct the following experiment to test protocol responsiveness. One tagged flow is sharing bandwidth with  $n - 1$  other standard TCP flows in the beginning of the simulation. Then at time 200, another  $n$  TCP flows join in to compete for the bottleneck link. Figure 18(a) shows the congestion window transition of the flow under study.

Consistent with [13], there is a tradeoff between responsiveness and smoothness. Standard TCP responds to congestion very quickly, at the expense of highly variable congestion window. IIAD makes the smoothest transition, but the transition time is very long. AIMD and SIMD perform in between and achieve similar smoothness and responsiveness. Notice that here the average window size of each connection is greater than 10, so IIAD has better smoothness in agreement with our analysis in Section 5.

### 6.2.3 Response to step increase of bandwidth (Aggressiveness)

This experiment was conducted to test protocol aggressiveness. The setup is similar to the previous experiment except that all  $2n$  flows are active in the beginning, then at time 200,  $n$  TCP flows terminate their transmission. Figure 18(b) shows the congestion window transition of the flow under study.

Standard TCP is still the fastest responding protocol, and IIAD is still the slowest. However, we can observe from the figure that SIMD is more aggressive than AIMD when the increase of bandwidth is significant, which agrees with our analysis in Section 5.

### 6.2.4 Impulse Response

To better illustrate the aggressiveness and responsiveness properties of different algorithms, we now study the behavior of different control algorithms responding to impulse disturbance from a periodical On/Off constant-bit-rate (CBR) flow.<sup>8</sup> The model is similar to the “square-wave” model used in the simulation study of [4]. In the experiment, we let the CBR flow alternate between On and Off state, each of which

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

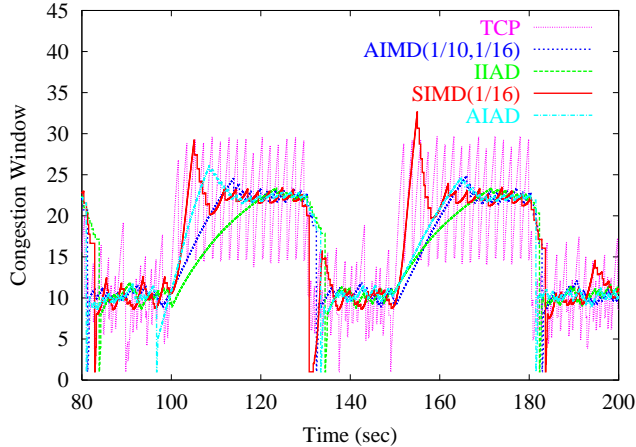


Figure 19: Impulse response to square-wave CBR flow.

Algorithm	$1/\text{Aggressiveness (RTT)}$		$1/\text{Responsiveness (losses)}$	
	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

lasts for  $t_{on}$  and  $t_{off}$ , respectively. The sending rate of the CBR flow during the active period is set to  $\gamma$  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 controlled by the algorithm under study. The results reported here are for  $C = 1.5\text{Mbps}$ , 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 19 plots the congestion window value of different control algorithms over the time period [80:200]. Here we also show results for AIAD.

We also prolong our simulation to repeat this impulse disturbance pattern and measure the average aggressiveness and responsiveness according to our definitions in Section 5 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 is 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 this scenario, AIAD is more aggressive than IIAD. In addition, after reaching steady state, AIAD is as smooth as IIAD. Notice the close match between the simulated measure of aggressiveness and the analytical results.

Aggressiveness of a congestion control algorithm is directly related to how much bandwidth a flow can get when it is competing with other flows. It has been shown in [4] that the set of slowly responsive congestion control algorithms 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 algorithm is hard to measure due to the extreme way TCP responds to burst of losses, which occurs when it sees sudden decrease of bandwidth. In

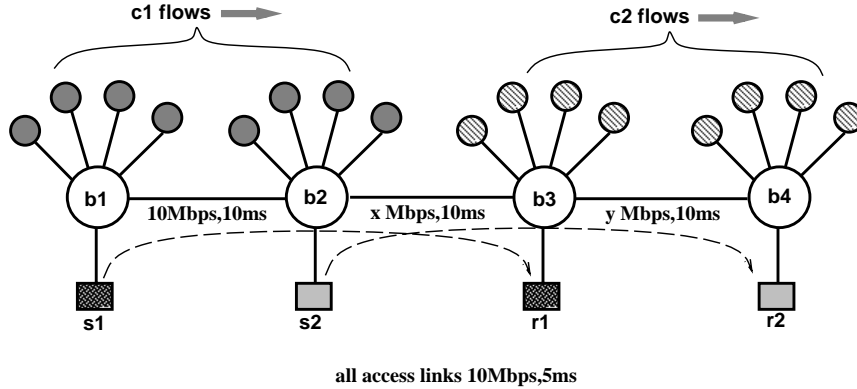


Figure 20: Simulation topology for convergence test

this case, all TCP flows reduce their congestion window to one regardless of which congestion avoidance algorithm 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 to efficiency and fairness of different algorithms.

We use the topology shown in Figure 20 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  $(b2, b3)$ .<sup>9</sup>

#### 6.3.1 $W_1 < \frac{W}{2} < W_2$ , $W_1 + W_2 = W$ (Convergence to Fairness)

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  $c_1 = 15$ ,  $c_2 = 0$ ,  $x = 6\text{Mbps}$ ,  $y = 6\text{Mbps}$ . 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 21 plots the transient behavior of the congestion window of different protocols.

It can be observed from the graph 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 4.2). The theoretical value is also given in the table for comparison. The following observations can be made from the table:

<sup>9</sup>We use packet size of 500 bytes in these experiments.

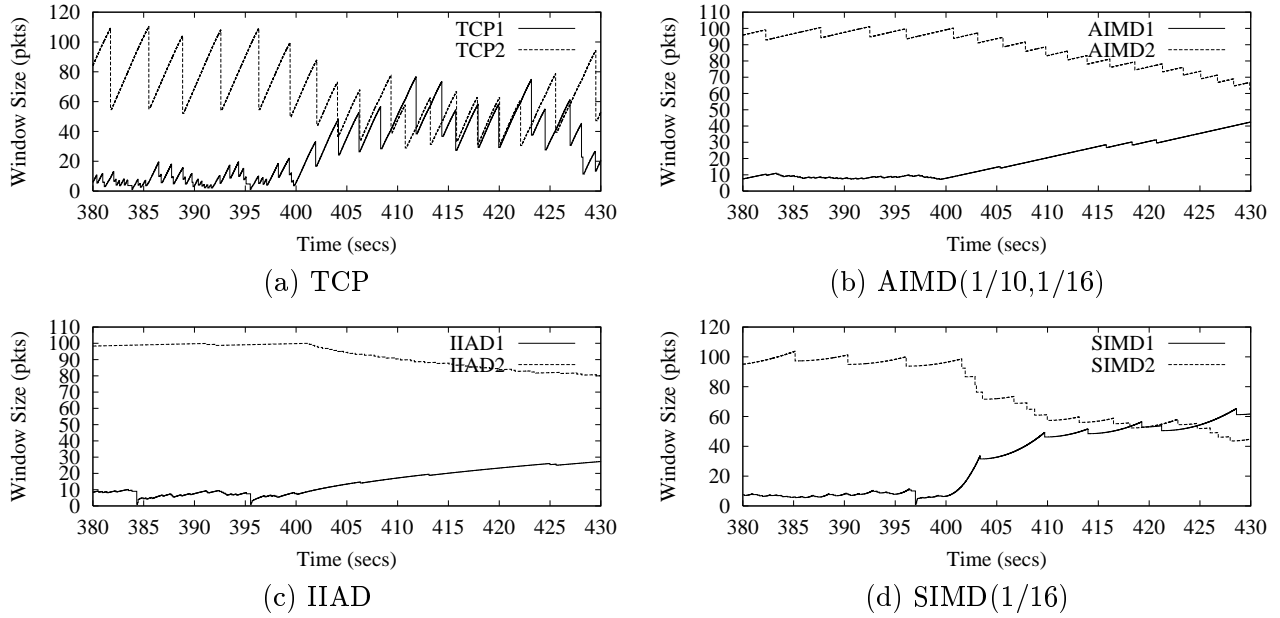


Figure 21: Two flows converge to fair share of bandwidth

Algorithm	Experiment 1				Experiment 2					
	$W_1$	$W_2$	$T_2$ (RTT)		$W_1$	$W_2$	$T_1$ (RTT)		$\Delta$ (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	<b>218</b>	<b>852</b>	10.2	33.2	90	85.1	13.6	12.3

Table 6: Quantitative measures on convergence time

- 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 to enhance the convergence rate to fairness.
- 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 less packet losses, which gives them a better chance to become more aggressive. On the contrary, AIMD does not fully capitalize on the random loss property of RED since its aggressiveness does not change. As a result, SIMD converges to fairness much faster.

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

To create such scenario, we let  $c_1 = 11$ ,  $c_2 = 3$ ,  $x = 6\text{Mbps}$ ,  $y = 10\text{Mbps}$ . 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  $(b_2, b_3)$ . 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 22 plots the transient behavior of the congestion window of different protocols.

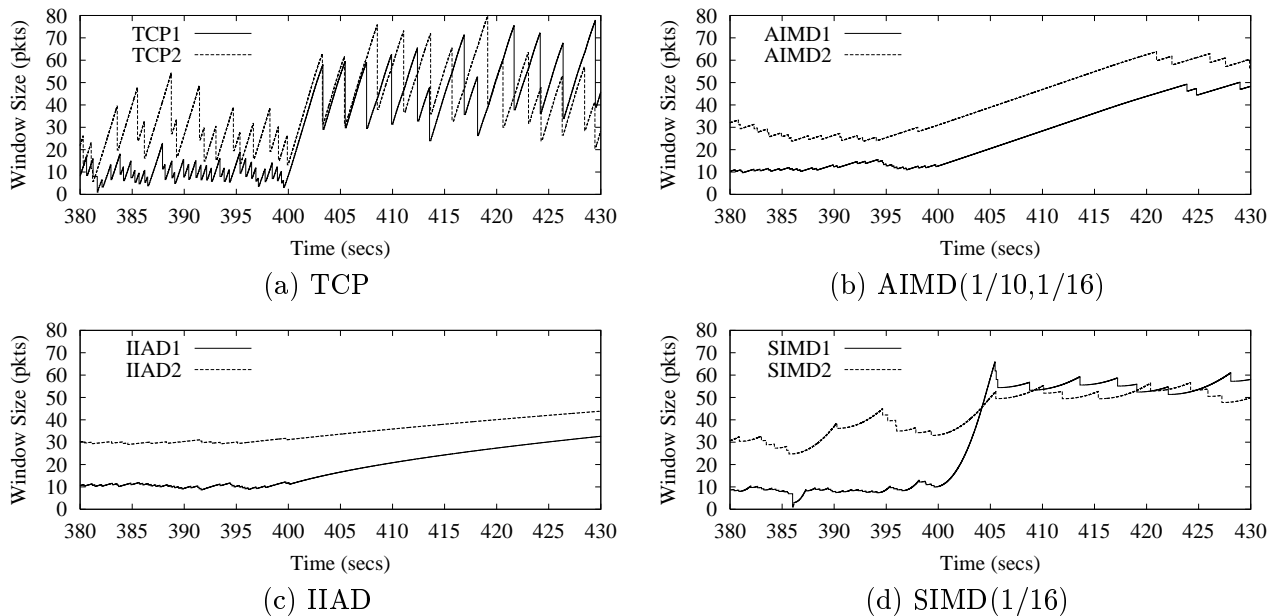


Figure 22: Two flows converge to fair share of bandwidth

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 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 window 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 smoothly and 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).

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 control algorithms known are Jacobson's TCP Reno [16] and Ramakrishnan and Jain's DECbit scheme [25]. 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 [13, 32] and TEAR [27]. These mechanisms use a moderate window decrease parameter to reduce rate variability, meanwhile use a matching window increase parameter to satisfy TCP-friendliness. There are tradeoffs between smoothness and reaction to changes in network conditions [13, 31].

Chiu and Jain also mentioned non-linear controls in [6]. They argued that non-linear controls reduce robustness and are not suitable for practical purposes. On the contrary, Bansal and Balakrishnan [3] proposed binomial algorithms that interact well with TCP AIMD. Binomial algorithms 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. Binomial

algorithms can be TCP-friendly if and only if  $k + l = 1$ . Binomial controls are memory-less in that they use only the current window size in their control rules. Our control algorithm suite is radically different from memory-less binomial algorithms. To our knowledge, our proposed scheme presents the first set of window-based TCP-friendly congestion control algorithms that use history information in their control rules. By doing so, our algorithms improve their transient behavior and convergence speed without sacrificing smoothness in steady state.

Another approach to provide smoother transmission rate is equation-based congestion controls [14, 24, 30], first proposed in [19]. In these schemes, the end-systems measure the packet loss rate and round-trip time, and use the TCP-friendly equation [23] to compute the transmission rate. Two comparisons [13, 31] 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 [4], Bansal *et al.* integrate self-clocking into the equation-based control algorithm to enhance its safety in deployment. They also compared such enhanced algorithm with other slow responsive but smooth congestion control (SlowCC) schemes such as binomial algorithms. Their simulation results show that all schemes become less competitive to standard TCP in a highly dynamic environment. SlowCC algorithms 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 [14]. 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.

Applications can be adaptive to the congestion level of the network in a TCP-friendly way. Examples include RAP [26] and LDA [29]. Applications using RAP can adapt the quality of transmitted streams based on the estimated rate. RAP employs an AIMD algorithm, similar to TCP. LDA relies on RTP [28] for feedback information about packet losses and round-trip time. The additive increase rate is estimated using reported loss, delay, and bottleneck bandwidth values.

Much of the literature has focused on the modeling of TCP congestion control [2, 10, 17, 21, 22, 23]. 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  $1/\sqrt{p}$ . A heuristic analysis in [10] shows that the throughput of a connection is inversely proportional to its round-trip time. Lakshman *et al.* [17] studied the properties of TCP in a regime where the bandwidth-delay product is high and losses are random. In [21], 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.* [23] extend this method and use a stochastic model that also captures the effect of TCP's timeout mechanism on throughput, thus provide a more accurate prediction of TCP throughput when random loss probability is moderate. Altman *et al.* [2] analyze TCP throughput under a more general loss process which is assumed to be only stationary. The model thus can account for any correlation and inter-loss time distributions. They also show that the throughput is inversely proportional to round-trip time and the square root of packet loss probability. Recently, Low *et al.* [18] presented a duality model of TCP Vegas congestion control mechanism [5].

## 8 Conclusion

We proposed a spectrum of TCP-like window-based congestion control algorithms. These algorithms are TCP-friendly. Unlike memory-less controls such as AIMD and binomial algorithms, our algorithms utilize history information. We have shown that they possess better convergence behaviors than AIMD and binomial algorithms. They possess different smoothness, aggressiveness, and responsiveness trade-

offs. 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 [9] to study their steady-state and transient behaviors. We presented simulation results of special cases SIMD and AIAD. In particular, SIMD uses multiplicative decrease but the window size increases as a quadratic function of elapsed time since detecting the last loss. Analysis and simulation were used to confirm the effectiveness of our scheme in terms of TCP-friendliness, TCP-compatibility, convergence to fairness, and the tradeoffs among smoothness, aggressiveness, and responsiveness. A release of the code for our *ns* implementations and the simulation scripts used for this paper will be available soon.

To summarize, most encouragingly, by exploring a new design space where window-based congestion control mechanisms utilize history, we can improve convergence behaviors during transient periods, while providing smoothness in steady state. Given the fact that equation-based congestion control schemes use longer history, we believe comparison between equation-based schemes and our scheme remains an interesting future work.

## References

- [1] M. Allman, V. Paxson, and W. Stevens. TCP congestion control, April 1999.
- [2] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of TCP/IP with stationary random losses. In *Proceedings of ACM SIGCOMM*, August 2000.
- [3] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *Proceedings of IEEE INFOCOM*, April 2001.
- [4] D. Bansal, H. Balakrishnan, and S. Floyd. Dynamic behavior of slowly-responsive congestion control algorithms. In *Proceedings of ACM SIGCOMM*, August 2001.
- [5] L. S. Brakmo and L. L. Peterson. TCP Vegas: end to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8), October 1995.
- [6] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.
- [7] M. Christiansen, K. Jeffay, D. Ott, and F. Smith. Tuning RED for Web Traffic. In *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, Aug.-Sep. 2000.
- [8] Cooperative Association for Internet Data Analysis. The CAIDA Website. <http://www.caida.org>.
- [9] E. Amir et al. UCB/LBNL/VINT Network Simulator - ns (version 2). Available at <http://http://www.isi.edu/nsnam/ns/>.
- [10] S. Floyd. Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic. *Computer Communication Review*, 21(5), August 1991.
- [11] S. Floyd. Recommendation on using the “gentle\_” variant of RED. <http://www.aciri.org/floyd/red/gentle.html>, March 2000.
- [12] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.
- [13] S. Floyd, M. Handley, and J. Padhye. A comparison of equation-based and AIMD congestion control. <http://www.aciri.org/floyd/papers.html>, May 2000.
- [14] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM*, August 2000.
- [15] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):393–417, August 1993.
- [16] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM*, August 1988.
- [17] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Trans. on Networking*, 5(3), 1997.
- [18] S. H. Low, L. Peterson, and L. Wang. Understanding TCP Vegas: A duality model. In *Proceedings of ACM SIGMETRICS*, June 2001.
- [19] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control. Note sent to end2end-interest mailing list, 1997.
- [20] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options. Internet RFC 2018, April 1996.
- [21] M. Mathis, J. Semske, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithms. *Computer Communication Review*, 27(3), July 1997.

- [22] T. J. Ott, J. Kemperman, and M. Mathis. The stationary behavior of ideal TCP congestion avoidance. <http://www.argreenhouse.com/papers/tjo>, 1996.
- [23] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM*, 1998.
- [24] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A model based TCP-friendly rate control protocol. In *Proceedings of NOSSDAV*, June 1999.
- [25] 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. Technical report, DEC, August 1987.
- [26] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *Proceedings of IEEE INFOCOM*, April 1999.
- [27] I. Rhee, V. Ozdemir, and Y. Yi. TEAR: TCP Emulation At Receivers – flow control for multimedia streaming. Technical report, Department of Computer Science, North Carolina State University, April 2000.
- [28] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, Jan 1996.
- [29] D. Sisalem and H. Schulzrinne. The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. In *Proceedings of NOSSDAV*, July 1998.
- [30] W.-T. Tan and A. Zakhor. Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol. *IEEE Trans. Multimedia*, 1(2):172–186, June 1999.
- [31] Y. R. Yang, M. S. Kim, and S. S. Lam. Transient behavior of TCP-friendly congestion control protocols. In *Proceedings of IEEE INFOCOM*, April 2001.
- [32] Y. R. Yang and S. S. Lam. General AIMD congestion control. In *Proceedings of ICNP*, November 2000.

# Appendix

## A TCP-friendliness of Our Control Scheme

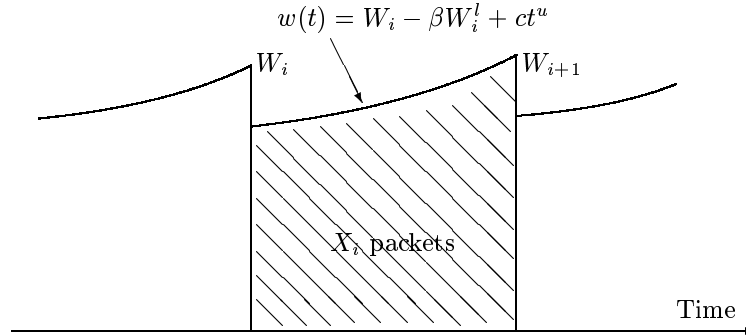


Figure 23: Window increases with time, and decreases on packet losses.

This appendix explains, without rigid proof, our choice of  $\alpha$  in Equation (4), or equivalently, the choice of  $c$  in Equation (5) to makes our control scheme TCP-friendly. We assume packet losses occur randomly with a fixed probability  $p$ , and (2) 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 a steady state, as shown in Figure 23. 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 periodical losses first.

### Periodic Losses

Under a periodical 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  $\beta W^l$ .

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

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

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  $\lambda$  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  $p = \frac{1}{X}$ . Plug them into the  $(\lambda, p)$  relationship, we get

$$c = \left(\frac{3}{2(1 - \frac{1}{k+2}\beta W^{l-1})}\right)^{\frac{1}{k+1}} \beta W^{l - \frac{1}{k+1}}. \quad (10)$$

Noticing, 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 trials: 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

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

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  $\bar{w}_i$ , i.e.,  $T_i = X_i/\bar{w}_i$ . Using (1), this results in a window increase of size

$$I_i \approx c \left( \frac{X_i}{\bar{w}_i} \right)^u.$$

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

$$I_i \approx c \left( \frac{X_i}{\bar{w}} \right)^u.$$

Then the expected window increase is:

$$\begin{aligned} E[I_i] &= \sum_{j=0}^{\infty} I_i P[X_i = j] \\ &\approx \sum_{j=0}^{\infty} c \left( \frac{j}{\bar{w}} \right)^u (1-p)^j p \\ &\approx \int_0^{\infty} c \left( \frac{x}{\bar{w}} \right)^u p e^{-px} dx \\ &= \frac{c \Gamma(u+1)}{(p\bar{w})^u}, \end{aligned} \tag{11}$$

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

$$E[I_i] = \frac{c}{(p\bar{w})^2}. \tag{12}$$

For TCP-friendliness, we need to equalize the expected window increases  $E[I_i]$  under both loss models.<sup>10</sup> Noticing the only difference between (11) and (12) is a factor of  $\Gamma(u+1)$ , we only adjust the definition in (10). 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.

---

<sup>10</sup>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.

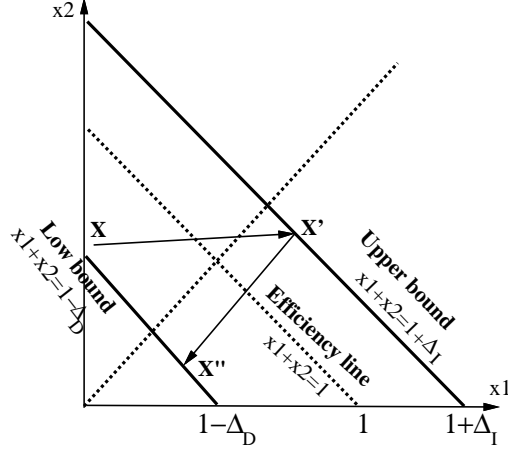


Figure 24: Bounded increases and decreases ensure convergence.

## B Condition of Convergence

We assume that the overshoot of the increase trajectory and the undershoot of the decrease trajectory are bounded. Then we give a sufficient condition so that our scheme converges when  $0 \leq l < 1$  and  $\frac{1}{k+1} - 1 \leq l < \frac{1}{k+1}$ . Let  $\Delta_I$  and  $\Delta_D$  denote the maximum overshoot and undershoot, respectively. Therefore, the trajectory in steady state moves between  $x_1 + x_2 \geq 1 - \Delta_D$  and  $x_1 + x_2 \leq 1 + \Delta_I$ , as shown in Figure 24. Without loss of generality, we assume the initial  $X(x_1, x_2)$  satisfies  $1 - \Delta_D \leq x_1 + x_2 < 1$  and  $x_1 < x_2$ .

Consider the extreme case of  $l - \frac{1}{k+1} = -1$  and  $l = 0$ . We consider  $l - \frac{1}{k+1} = -1$  since it gives us the maximum increase beyond the fairness line. We consider  $l = 0$  since additive decrease gives us the maximum unfairness after decrease. Thus, the condition derived for this extreme case will be sufficient for other cases. Since in this case the increase is in proportion to  $1/x_i, i = 1, 2$ , the increase trajectory reaches at most the point  $X'(x_1 - x_2 + \frac{x_2(1+\Delta_I)}{x_1+x_2}, x_2 - x_1 + \frac{x_1(1+\Delta_I)}{x_1+x_2})$ . We need only to consider the case when the trajectory has increased beyond the fairness line. That is,

$$x_1 - x_2 + \frac{x_2(1 + \Delta_I)}{x_1 + x_2} > x_2 - x_1 + \frac{x_1(1 + \Delta_I)}{x_1 + x_2},$$

from which we have

$$1 + \Delta_I > 2(x_1 + x_2). \quad (13)$$

Since the decrease is additive, the decrease trajectory reaches at most the point  $X''(x_1 - x_2 + \frac{x_2(1+\Delta_I)}{x_1+x_2} - \frac{\Delta_I+\Delta_D}{2}, x_2 - x_1 + \frac{x_1(1+\Delta_I)}{x_1+x_2} - \frac{\Delta_I+\Delta_D}{2})$ . We need

$$x_2 - x_1 + \frac{x_1(1 + \Delta_I)}{x_1 + x_2} - \frac{\Delta_I + \Delta_D}{2} > 0.$$

Since we have (13),  $\frac{1+\Delta_I}{x_1+x_2} > 2$ , so it suffices to have  $x_1 + x_2 > \frac{\Delta_I+\Delta_D}{2}$ . Since we have  $x_1 + x_2 \geq 1 - \Delta_D$ , we need only  $1 - \Delta_D > \frac{\Delta_I+\Delta_D}{2}$ . That is,

$$\Delta_I + 3\Delta_D < 2. \quad (14)$$

Interestingly, we find that this condition is sufficient for  $X''$  to be more fair than  $X$ . Hence, with this condition, when the system evolves, fairness is improved.

This condition can be easily satisfied when the rate of decrease and increase is not very high. This is the case when we consider *smooth* transmission rate.

## C Convergence Time

We use Figure 4(b) to illustrate how we compute the convergence time. We use the phase plot as in the convergence analysis. Assuming we start from an arbitrary point  $(W_1, W_2)$  in the graph, also assume this point is below the efficiency line  $w_1 + w_2 = W$ , where  $W$  is the bottleneck resource (measured in terms of packets). Without loss of generality, we assume  $W_2 \geq W_1$ . We assume synchronized feedback. We can then analyze the time it takes a control mechanism to bring this starting point close to the fairness line, or specifically,  $|w_2 - w_1| < \epsilon$ , and then oscillate around the efficiency line. For ease of analysis, we divide this convergence time into two parts:  $T_1$ , the time it takes to converge to the efficiency line, i.e., the time from the initial point to the first time loss is detected (window is then decreased), measured in number of round trip times; and  $T_2$ , the time needed to converge to the fairness line, measured in number of congestion epochs. We also derive  $T_2$  in terms of number of RTT's at the end of this section.

### C.1 Convergence Time to Efficiency

At time  $T_1$ , the trajectory crosses the efficiency line. We can thus compute  $T_1$  as follows.

For AIMD, we have that the window increments of  $w_1$  and  $w_2$  are the same, i.e.,  $\delta_1 = \delta_2$ . Since

$$w_1 + w_2 = W_1 + \delta_1 + W_2 + \delta_2 = W$$

we now have:

$$\delta_1 = \delta_2 = (W - W_1 - W_2)/2$$

Since in each RTT, windows are increased by 1 in TCP and by  $\alpha = 3\beta/(2 - \beta)$  in general AIMD [13], we now have:

$$\begin{aligned} T_1^{TCP} &= \frac{W - W_1 - W_2}{2} \\ T_1^{AIMD} &= \frac{W - W_1 - W_2}{2\alpha} = \frac{(W - W_1 - W_2)(2 - \beta)}{6\beta} \end{aligned}$$

Note that at this moment, the difference between the two window values remains the same, i.e.,

$$\Delta^{TCP} = \Delta^{AIMD} = w_2 - w_1 = W_2 - W_1$$

For SIMD, we have that the window increment is inversely proportional to  $W_i$ , so  $\delta_1/\delta_2 = W_2/W_1$ , therefore, we have:

$$\begin{aligned} \delta_1 &= \frac{W_2}{W_1 + W_2}(W - W_1 - W_2) \\ \delta_2 &= \frac{W_1}{W_1 + W_2}(W - W_1 - W_2) \end{aligned}$$

Since the window increase function is  $w(t) = w_0 + \frac{\alpha^2}{4}t^2$ , where  $\alpha$  is defined in Equation (4), the time it takes each user to come to this point is:

$$T_1^{SIMD} = \gamma \sqrt{\frac{W_1 W_2 (W - W_1 - W_2)}{W_1 + W_2}}$$

where  $\gamma = 2(1 - \frac{2\beta}{3})\sqrt{\frac{2}{9\beta(1-\beta)}}$ . And the difference between the two new values becomes:

$$\Delta^{SIMD} = |2(W_2 - W_1) - \frac{W}{W_1 + W_2}(W_2 - W_1)|$$

Given IIAD rules, since the trajectory is inversely proportional to the current window size, the window growth follows the function  $w(t) = \sqrt{2\alpha t} = \sqrt{3\beta t}$ .<sup>11</sup> By solving the equation  $w_1 + w_2 = \sqrt{3\beta T_1} + W_1^2 +$

<sup>11</sup>Since  $\frac{dw}{dt} = \alpha/w$ , and  $w(0) = 0$ , we get  $w(t) = \sqrt{2\alpha t}$ . Also,  $\alpha = \frac{3}{2}\beta$  for TCP-friendliness [3].

$\sqrt{3\beta T_1 + W_2^2} = W$ , we get:

$$T_1^{IIAD} = \frac{1}{12\beta W^2} (W_1^4 + W_2^4 + W^4 - 2W_1^2 W_2^2 - 2W^2 W_2^2 - 2W_1^2 W^2)$$

And the difference becomes:

$$\Delta^{IIAD} = w_2 - w_1 = \frac{|W_2^2 - W_1^2|}{W}$$

## C.2 Convergence Time to Fairness

After time  $T_1$ , the trajectory will oscillate around the efficiency line. We now redefine the initial point  $(W_1, W_2)$  to be the starting point of each congestion epoch. We can then derive the change in  $\Delta$  after each congestion epoch.

For TCP/AIMD, at the end of each congestion epoch, the window values evolve to  $(W_1 + \delta_1, W_2 + \delta_2)$ . Thus the starting point for the next congestion epoch will be:  $((1 - \beta)(W_1 + \delta_1), (1 - \beta)(W_2 + \delta_2))$ . Since the point is oscillating around the efficiency line, the sum of the two window values at the beginning of each congestion epoch should be the same. We therefore have:

$$\begin{aligned} (1 - \beta)(W_1 + \delta_1 + W_2 + \delta_2) &= W_1 + W_2 \\ \delta_1 &= \delta_2 \end{aligned}$$

Therefore,  $\delta_1 = \delta_2 = \frac{\beta}{1 - \beta} \frac{W_1 + W_2}{2}$ . After each congestion epoch, the difference between the two window values becomes:

$$\Delta' = (1 - \beta)|(W_2 + \delta_2) - (W_1 + \delta_1)| = (1 - \beta)\Delta$$

For SIMD, since the window increments are inversely proportional to  $W_i$ 's, we can then have the following relationships:

$$\begin{aligned} (1 - \beta)(W_1 + \delta_1 + W_2 + \delta_2) &= W_1 + W_2 \\ \delta_1/W_2 &= \delta_2/W_1 \end{aligned}$$

We can then get:

$$\Delta' = (1 - 2\beta)|W_2 - W_1| = (1 - 2\beta)\Delta$$

For IIAD, it is hard to derive the relationship in steady state, but when  $W_1$  and  $W_2$  are large, we can use the window values at the initial point to approximate the current window sizes, and thus make the window increments again inversely proportional to  $W_i$ 's. Note that this approximation is actually an upper bound on convergence rate, which implies IIAD will converge slower than this rate. Thus:

$$\begin{aligned} W_1 + \delta_1 - \beta + W_2 + \delta_2 - \beta &= W_1 + W_2 \\ \delta_1/W_2 &= \delta_2/W_1 \end{aligned}$$

We can then get:

$$\Delta' = \left(1 - \frac{2\beta}{W_1 + W_2}\right)\Delta \approx \left(1 - \frac{2\beta}{W}\right)\Delta$$

The last approximation is valid when  $W_1 + W_2 \approx W$ , or  $W \gg \beta$ .

Assume we need to have some bounds on the difference between the two windows, i.e.,  $|w_1 - w_2| < \epsilon$ , and also assume the initial window difference is  $\Delta$ , we then have:

$$T_2^{TCP} = \log_{1/2} \frac{\epsilon}{\Delta}$$

$$\begin{aligned}
T_2^{AIMD} &= \log_{1-\beta} \frac{\epsilon}{\Delta} \\
T_2^{SIMD} &= \log_{1-2\beta} \frac{\epsilon}{\Delta} \\
T_2^{IIAD} &\geq \log_{1-2\beta/W} \frac{\epsilon}{\Delta}
\end{aligned}$$

Note that  $T_2$ 's are measured in number of congestion epochs. To convert it to RTT's, we need to compute the length of congestion epoch  $L$  (in RTT's) for each mechanism given the initial window values of each congestion epoch. Given the window increase and decrease rules, it is straightforward to obtain the following results<sup>12</sup>:

$$\begin{aligned}
L^{TCP} &= W/4 \\
L^{AIMD} &= \frac{\beta W}{2\alpha} = \frac{(2-\beta)W}{6} \\
L^{SIMD} &= \frac{2}{3} \frac{1-\frac{2\beta}{3}}{1-\beta} \sqrt{2W_1W_2} \\
L^{IIAD} &= \frac{4}{3} \frac{W_1W_2}{(W-2\beta)} \approx \frac{4}{3} \frac{W_1W_2}{W}
\end{aligned}$$

We can then iteratively compute the value of  $T_2$  in terms of RTT's by summing up the length of each congestion epoch. However, note that in steady state, i.e.,  $W_1 \approx W_2 \approx \frac{W}{2}$ . Assuming  $\beta^{SIMD} \ll 1$ , we have  $L^{SIMD} \approx \sqrt{2}W/3$ . Assuming  $\beta^{IIAD} \ll W$ , we have  $L^{IIAD} \approx W/3$ . If we assume that the length of the transient congestion epoch values are close to the steady state values, we can then approximate the convergence time  $T_2$  (in RTT) to fairness as follows:

$$\begin{aligned}
T_2^{TCP} &= \frac{W}{4} \log_{1/2} \frac{\epsilon}{\Delta} \\
T_2^{AIMD} &= \frac{(2-\beta)W}{6} \log_{1-\beta} \frac{\epsilon}{\Delta} \\
T_2^{SIMD} &\approx \frac{\sqrt{2}W}{3} \log_{1-2\beta} \frac{\epsilon}{\Delta} \\
T_2^{IIAD} &\geq \frac{W}{3} \log_{1-2\beta/W} \frac{\epsilon}{\Delta}
\end{aligned}$$

## D Smoothness, Aggressiveness, and Responsiveness of several algorithms

### D.1 Smoothness

We compute the expected coefficient of variation of the window size in one congestion epoch. Let  $W$  denote the mean window size. Packet drops are random, with drop probability  $p$ . We have  $1/p = 2W^2/3$  from the TCP-friendly equation.

**(a) AIMD( $\alpha, \beta$ )** Consider a congestion epoch in steady state where the window size increases from  $w_0$ , then decreases upon the drop of a packet. We assume the window size variation is small, such that the mean window size in this epoch is close to  $W$ . Thus the following approximation works reasonably well. Assume  $x$  packets have been acknowledged before the first drop. The time of drop  $T$  is approximated by  $\frac{x}{W}$ . In this congestion epoch, the window increase function is  $w(t) = w_0 + \alpha t, 0 < t < T$ , where

<sup>12</sup>For example, for TCP, since the total window size increases by  $\frac{W}{2}$  in one congestion epoch at 2-packet increments (one per user), we get  $L = \frac{W}{4}$  RTT's.

$\alpha = 3\beta/(2 - \beta)$ . To simplify the notation, we use  $w$  instead of  $w(t)$  in our derivation. We compute the mean window size and variance as follows:

$$E[w] = w_0 + \frac{3\beta}{2 - \beta} \frac{T}{2}.$$

$$\begin{aligned} \text{Var}[w] &= E[(w - E[w])^2] \\ &= \frac{1}{T} \int_0^T (w - E[w])^2 dt \\ &= \left(\frac{3\beta}{2 - \beta}\right)^2 \frac{T^2}{12} \\ &\approx \left(\frac{3\beta}{2 - \beta}\right)^2 \frac{x^2}{12W^2}. \end{aligned}$$

Then using this approximation, we compute the mean variance and CoV as:

$$\begin{aligned} E[\text{Var}[w]] &= \int_0^\infty \text{Var}[w] p e^{-px} dx \\ &\approx \int_0^\infty \left(\frac{3\beta}{2 - \beta}\right)^2 \frac{x^2}{12W^2} p e^{-px} dx \\ &= \left(\frac{3\beta}{2 - \beta}\right)^2 \frac{1}{6W^2 p^2} \\ &= \left(\frac{\beta}{1 - \beta/2}\right)^2 \frac{W^2}{6}. \end{aligned}$$

$$\begin{aligned} \text{CoV} &= \sqrt{E[\text{Var}[w]]}/W \\ &\approx \frac{0.41\beta}{1 - \beta/2}. \end{aligned}$$

**(b) IIAD**( $\alpha, \beta$ ) Assume  $\beta \ll W$ . Using linear interpolation, IIAD is approximated by AIMD( $\alpha/W, \beta/W$ ), where  $\alpha = 3\beta/(2 - \beta)$ . Hence, we can compute the coefficient of variation for IIAD, similar to AIMD.

$$\text{CoV} \approx \frac{0.41\beta}{W - \beta/2}.$$

**(c) AIAD**( $\beta$ ) Assume  $\beta \ll W$ . In steady state, AIAD( $\beta$ ) is approximated by AIMD( $\alpha/W, \beta/W$ ), where  $\alpha = 3\beta/(2 - \beta)$ . Hence, we can compute the coefficient of variation for AIAD, similar to AIMD.

$$\text{CoV} \approx \frac{0.41\beta}{W - \beta/2}.$$

**(d) SIMD**( $\beta$ ) The window increase function is  $w(t) = w_0 + ct^2, 0 < t < T$ , where  $c = \frac{9\beta}{8(1 - \frac{2\beta}{3})^2 w_0}$ . We take  $w_0 \approx W$  in computation. Following the same steps as for AIMD, we compute the mean window size and variance as:

$$E[w] = w_0 + \frac{9\beta}{8(1 - \frac{2\beta}{3})^2 w_0} \frac{T^2}{3}.$$

$$\begin{aligned}
Var[w] &= E[(w - E[w])^2] \\
&= \frac{1}{T} \int_0^T (w - E[w])^2 dt \\
&= \left( \frac{9\beta}{8(1 - \frac{2\beta}{3})^2} \right)^2 \frac{4}{45W^2p^4} \\
&\approx \left( \frac{9\beta}{8(1 - \frac{2\beta}{3})^2} \right)^2 \frac{4x^4}{45W^6}.
\end{aligned}$$

Then using this approximation, we compute the mean variance and CoV as:

$$\begin{aligned}
E[Var[w]] &= \int_0^\infty Var[w]pe^{-px} dx \\
&\approx \int_0^\infty \left( \frac{9\beta}{8(1 - \frac{2\beta}{3})^2} \right)^2 \frac{4x^4}{45W^6} pe^{-px} dx \\
&= \left( \frac{\beta}{(1 - 2\beta/3)^2} \right)^2 \frac{8W^2}{15}.
\end{aligned}$$

$$\begin{aligned}
CoV &= \sqrt{E[Var[w]]/W} \\
&\approx \frac{0.73\beta}{(1 - \frac{2\beta}{3})^2}.
\end{aligned}$$

## D.2 Aggressiveness

We compute the number of RTT's necessary for a connection to increase its window size by a factor of  $m$ . Let  $W$  denote the window size before the transition. For AIMD, since the window size increase by  $\alpha = \frac{3\beta}{2-\beta}$  in each RTT, the total number of RTT's is  $\frac{(m-1)W}{\alpha} \approx \frac{2(m-1)W}{3\beta}$ . For SIMD, according to Equation (5),  $c_{SIMD} = \frac{9\beta}{8(1-2\beta/3)^2W}$ . Therefore, the number of RTT's is  $\sqrt{(m-1)W/c_{SIMD}} \approx \sqrt{\frac{m-1}{\beta} \frac{2\sqrt{2}W}{3}}$ . For AIAD, since its increase constant is  $\approx \frac{3\beta}{2W}$ , it takes  $\frac{2(m-1)W^2}{3\beta}$  RTT to increase the window size by  $(m-1)W$ . For IIAD, the window size function is  $w(t) = \sqrt{2\alpha t}$ ,  $t$  is in RTT [3]. Since  $\alpha \approx 3\beta/2$  in order for IIAD to be TCP-friendly,  $w(t) = \sqrt{3\beta t}$ . When  $t = \frac{W^2}{3\beta}$ , the window size is  $W$ . When  $t = \frac{m^2W^2}{3\beta}$ , the window size is  $mW$ . Thus the number of RTT's necessary is  $\frac{(m^2-1)W^2}{3\beta}$ .

## D.3 Responsiveness

We compute the number of loss events necessary for a connection to decrease its window size by a factor of  $m$ . Both AIMD and SIMD use multiplicative decrease with parameter  $\beta$ . Thus, they need  $\log_{(1-\beta)} \frac{1}{m}$  loss events. Both IIAD and AIAD decrease the window size by a constant  $\beta$  on detecting each loss event. Therefore, they need  $\frac{W-W/m}{\beta}$  loss events.

## E Implementation

To implement 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 have to 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  $\text{SIMD}(\beta)$ , we have the following equation:

$$w_{new} = w_{old} + \alpha \frac{\sqrt{w_{old} - w_0}}{w_{old}}, \quad (15)$$

where  $\alpha$  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:

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

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:

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