TCP-friendly SIMD Congestion Control and Its Convergence Behavior*

Shudong Jin Liang Guo Ibrahim Matta Azer Bestavros
Computer Science Department
Boston University
Boston, MA 02215, USA
{jins,guol,matta,best}@cs.bu.edu

Abstract

The increased diversity of Internet application requirements has spurred recent interests in flexible congestion control mechanisms. Window-based congestion control schemes use increase rules to probe available bandwidth, and decrease rules to back off when congestion is detected. The control rules are parameterized so as to ensure that the resulting protocol is TCP-friendly in terms of the relationship between throughput and packet loss rate. In this paper, we propose a novel window-based congestion control algorithm called SIMD (Square-Increase/Multiplicative-Decrease). Contrary to previous memory-less controls, SIMD utilizes history information in its control rules. It uses multiplicative decrease but the increase in window size is in proportion to the square *of the time elapsed since the detection of the last loss event. Thus, SIMD can efficiently probe available bandwidth. Nevertheless, SIMD is TCP-friendly as well as TCP-compatible through RED routers. Furthermore, SIMD has much better convergence behavior than TCP-friendly AIMD and binomial algorithms proposed recently.*

1. Introduction

TCP uses additive-increase and multiplicative-decrease (AIMD). It probes available bandwidth by increasing its congestion window size linearly, and responds to congestion (indicated by packet losses) by decreasing the window size multiplicatively. Recently proposed congestion control mechanisms include generalizations of TCP-like window-based schemes [1, 8, 19, 22] and equation-based schemes [9, 17, 20]. One common objective of these new schemes is to reduce variations in transmission rate. Such high variations may limit network utilization. In addition, they are not desirable for emerging applications such as real-time streaming applications on the Internet.

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

It is required that new protocols implement congestion control mechanisms that interact well with TCP [7]. 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 (λ, p) relationship $\lambda = \sqrt{3/2}/(R\sqrt{p})$ should approximately hold, where λ is the throughput of a flow, p is its packet loss rate, and R is the round-trip time.

In addition, there are other requirements for congestion control algorithms. *Smoothness* measures the variation in the transmission rate of a connection using the protocol. Smoothness is important in steady state. High smoothness is desirable for some applications, e.g., Internet real-time applications. *Aggressiveness* measures how fast the connection probes extra bandwidth by opening up its window. In particular, when there is a sudden increase in available bandwidth, it is desirable that the connection acquires it quickly. *Responsiveness* measures how fast the connection reacts to increased congestion by decreasing its window size. It is desirable that the connection reduces its transmission rate to its fair share promptly. Both aggressiveness and responsiveness are measures of the transient behavior of congestion control protocols [21]. *Convergence* measures how fast competing connections converge to their fair share of bandwidth. Convergence speed is related to the aggressiveness and responsiveness indices. More aggressive and responsive protocols usually converge faster.

Several recently proposed TCP-friendly congestion control schemes, including general AIMD [8, 21], binomial algorithms [1], TFRC [9], and TEAR [19], can provide smoother transmission rate than TCP. However, these algorithms may lack the aggressiveness and responsiveness of TCP. Hence, when network conditions change drastically, these protocols can not react to the change promptly. In particular, recent studies [8, 21] have compared TCP AIMD, general AIMD, TFRC, and TEAR, and shown that higher smoothness results in lower aggressiveness. Therefore, a question is, *can TCP-friendly congestion control algorithms maintain high smoothness in steady state and still have high aggressiveness when there are drastic changes in network conditions?*

Meanwhile, it is necessary to consider the convergence of congestion control schemes. Chiu and Jain [2] showed that AIMD control converges to fairness and efficiency. Recently, it was shown that the additive increase of TCP and general AIMD control is inferior [11]. Binomial algorithms [1] using non-additive increase also possess the convergence property. Binomial algorithms are similar to AIMD in that they all use memory-less control. That is, their control rules use only the current window size. Therefore, a question is, *can one improve the convergence behavior by using history information in window-based congestion control algorithms?*

This paper provides answers to these questions. We study TCP-like window-based congestion control algorithms. Contrary to the memory-less AIMD and binomial algorithms [1], we consider the case where connections utilize history information, in addition to the current window size. The only history we use is the window size at the time of detecting the last loss. To this end, we propose a novel algorithm called SIMD (Square-Increase/Multiplicative-Decrease). SIMD decreases the window size multiplicatively but increases it in proportion to the *square* of the time elapsed since the detection of the last loss event. SIMD can have high smoothness in steady state, and if network conditions change drastically, SIMD can grow aggressive. On the contrary, other control schemes increase the window size linearly or sub-linearly. We show that SIMD is TCP-friendly: connections using SIMD have approximately the same throughput as TCP connections, given the same packet loss rate and round-trip time. Furthermore, SIMD has better convergence behavior than that of memory-less AIMD and binomial algorithms. We use a synchronized feedback model [2] to illustrate the convergence behavior of SIMD. In addition, using the ns simulator [5], we show that SIMD can fully capitalize on the random loss property of RED [10] to improve convergence speed.

Our SIMD algorithm 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, SIMD improves transient behavior by using history. Compared to equation-based schemes, SIMD has 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 algorithm in Section 2. We analyze its convergence behavior in Section 3. Our simulation results are described in Section 4. We revisit related work in Section 5 and finally conclude the paper.

2. SIMD Congestion Control

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 packet losses. 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 uses the following linear control rules:

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

where w_t is the window size at time t, R is the round-trip time, and δ 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 once a packet loss event is detected. Binomial algorithms [1] generalize AIMD with non-linear controls. They use the following control rules:

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

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

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 (α, β, k, and l). Neither of them utilizes history information. On the contrary, we find the window size at the end of the last congestion epoch (before the decrease) handy and useful. 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. Note, 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. We define the control rules of SIMD as:

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

Like AIMD, SIMD uses multiplicative decrease. However, SIMD uses an increase rule very different from those used by AIMD and binomial algorithms. First, SIMD uses the history information of a connection since w_0 is the window size after the last decrease. (Later, we will also show that α itself depends on w_{max}, and changes from one congestion epoch to another.) Second, the increase pattern of the window size is super-linear. To elaborate on this point, next we show that SIMD's increase rule results in a quadratic function of time t since the detection of the last loss event.

Let $w(t)$ be the continuous approximation of the window size at time t (in RTT's) elapsed since the window started to increase. By definition, $w_0 = w(0)$. Using linear interpolation and continuous approximation, from the increase rule in (1), we have

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

This gives us

$$\frac{1}{\sqrt{w(t) - w_0}} dw(t) = \alpha dt.$$

We integrate both sides to get $2\sqrt{w(t) - w_0} = \alpha t + C$. Notice that the constant $C = 0$ since when $t = 0$, $w(t) = w_0$. We then rewrite it as

$$w(t) = w_0 + \frac{\alpha^2}{4} t^2. \quad (2)$$

Therefore, SIMD can grow aggressive with time. This property is important since it allows SIMD to efficiently probe extra bandwidth when it becomes available. For SIMD, it is possible to have high smoothness (low variation of window size) in steady state by using a small β, as well as high aggressiveness when there are drastic changes in network conditions. On the contrary, if $w(t)$ is a linear or sub-linear function of t, then the connections are unable to acquire bandwidth quickly. For example, TCP-friendly AIMD algorithm needs to parameterize its control rules by defining α as a function of β [8, 22]. In particular, without considering the effect of TCP's timeout mechanisms, $\alpha = 3\beta/(2 - \beta)$. Although smoothness is possible by using moderate decrease, AIMD becomes insensitive to sudden increases in available bandwidth.

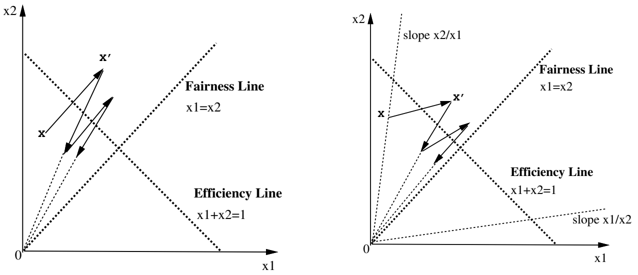
For SIMD, a remaining question is, how can we define α in the increase rule (1) such that SIMD is TCP-friendly, given β and the state variable w_{max} (or w_0)? Here we assume the multiplicative decrease factor β is a constant. We define α as follows:

$$\alpha = \frac{3\sqrt{\beta}}{(1 - 2\beta/3)\sqrt{2w_{max}}}. \quad (3)$$

Thus, during a congestion epoch, α is inversely proportional to $\sqrt{w_{max}}$. Due to space limitation, we refer the reader to [13] for why this choice of α makes our SIMD algorithm TCP-friendly assuming a random loss model. In Section 4, we use simulations to validate the TCP-friendliness of SIMD for a wide range of loss rate. From (3), Equation (2) becomes:

$$w(t) = w_0 + \frac{9\beta}{8(1 - 2\beta/3)^2 w_{max}} t^2. \quad (4)$$

We observe the following. First, the increase term of the increase rule in (1) is proportional to $\sqrt{(w_t - w_0)/w_{max}}$. Since w_t, w_0, and w_{max} are dependent on the window size, the increase term is time-varying. Therefore, SIMD can be viewed as a special AIMD whose increase parameter α (in the control rules of AIMD) is always varying. The elegance of SIMD is, by doing this, it can provide high smoothness (using small β) in steady state, and still have high aggressiveness when there is a sudden increase of available bandwidth. Second, the rate at which $w(t)$ increases is inversely proportional to w_{max}, as shown in Equation (4). Therefore, if there are two SIMD flows competing, then the flow with smaller window size is more aggressive. This property can result in better convergence behavior. AIMD does not have such property.



(a) AIMD trajectory (b) SIMD trajectory

Figure 1. Convergence of AIMD and SIMD

3. Convergence to Fairness and Efficiency

In this section, we first show that SIMD converges to fairness and efficiency under a synchronized feedback assumption. Then we show that SIMD converges faster than memoryless AIMD and binomial controls.

3.1. Convergence of SIMD

We adopt the ideal synchronized feedback assumption [2]. 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 [2] 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 1, any two-user resource allocation can be represented by a point $X(x_1, x_2)$, where x_i is the resource allocation (normalized by total capacity) for the i^{th} user, $i = 1, 2$. We define the fairness index as $\max(\frac{x_1}{x_2}, \frac{x_2}{x_1})$. If the fairness index is closer to unity, the resource allocation is more fair. The line $x_1 = x_2$ is the "fairness line". The line $x_1 + x_2 = 1$ is the "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 by a constant. Figure 1(a) shows the trajectory to X' parallel to the fairness line. This movement improves fairness (i.e., reduces the fairness index). Then both users use multiplicative decrease, which does not change fairness. Hence, as the system evolves, AIMD brings the resource allocation point toward the fairness line, finally oscillating around the efficiency line.

For SIMD control, we first observe Equation (4). We can see that the window size of a connection increases in proportion to $1/x_i, i = 1, 2$. Thus, as shown in Figure 1(b), the increase trajectory emanates from $X(x_1, x_2)$ with slope $\frac{x_1}{x_2}$. Indeed, at any point between the two lines emanating from the origin with slopes $\frac{x_1}{x_2}$ and $\frac{x_2}{x_1}$, the resource allocation X' is more fair than X as it reduces the value of the fairness index. Therefore, the increase phase of SIMD improves fairness. Since like AIMD, SIMD uses multiplicative decrease, the decrease phase of SIMD does not change fairness. Hence, SIMD converges to fairness and efficiency.

3.2. Convergence Speed

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

First, to intuitively show that SIMD converges faster than AIMD, we show that the increase trajectory of SIMD intersects the efficiency line at a point that is usually more fair than that of 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, 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 SIMD, 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 of 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 2(a). Since the size of area (1) is much larger than area (2), we intuitively say that SIMD usually converges faster than AIMD.

Then, we analytically compare the convergence time of SIMD, general AIMD [8, 22], and binomial control schemes [1]. Binomial algorithms are a family of algorithms generalizing AIMD. The control rules were shown in Section 2. We choose IIAD (Inverse-Increase/Additive-Decrease) as a representative. IIAD has an increase term inversely proportional to the current window size ($k = 1$) and a constant decrease term ($l = 0$). We still assume synchronized feedback and use Figure 2(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, W_2), where $W_1 \leq W_2$ and $W_1 + W_2 < W$, close to the efficiency line $w_1 + w_2 = W$, and T_2, the time until the difference between the two user windows stays within a certain small bound, i.e., $|w_1 - w_2| < \epsilon$. T_1 and T_2 are measured in round-trip times. We also denote the difference between the two user windows after T_1 as Δ. Due to space limitation, we only present the main results here in Table 1. The detailed analysis can be found in [13].

We numerically solve the above equations for different initial points. Figure 2(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, which supports our intuitive claim (cf. Figure 2(a)). 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.

4. Simulation Results

We use the *ns* simulator [5] to validate that with RED [10] queue management, our proposed algorithm is TCP-friendly and TCP-compatible. We also investigate the way two homogeneous flows converge to their bandwidth fair share and show that our proposed algorithm outperforms other algorithms, including TCP [12], general AIMD [8, 22], and IIAD [1]. Details about the implementation of SIMD in the ns simulator can be found in [13].

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 2.

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 × B/W delay product
DropTail queue limit	1.5 × 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 2. Network configuration

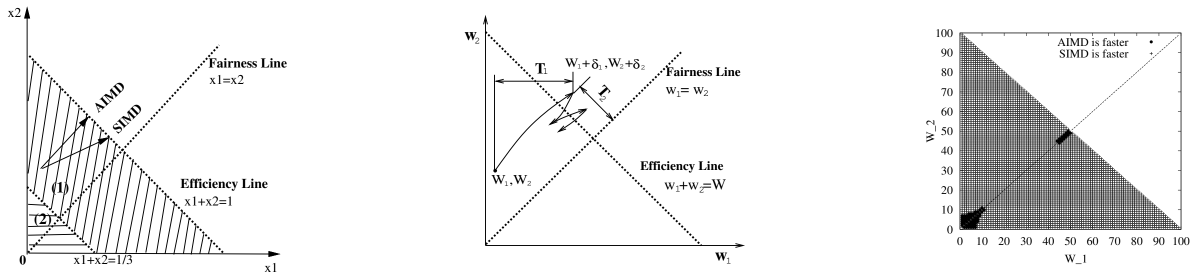
The bottleneck queue size and RED queue parameters are tuned as recommended in [3]. The "gentle_" option of RED queue is turned on as recommended in [6]. We choose $\beta = 1/16$ for SIMD and AIMD (and thus $\alpha \approx 1/10$ for AIMD to ensure TCP-friendliness). For IIAD, $\alpha = 1$ and $\beta = 2/3$. For 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 [15] for congestion detection. We also obtained similar results for other mechanisms (e.g. Reno, newReno). We assume no delayed acknowledgments.

4.1. 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 (with respect to standard TCP measured over 3000 RTT) of SIMD for different β values and plot them in Figure 3. For comparison, we also plot AIMD(1/5,1/8) throughput.

We notice that all the curves have a dip when the loss rate is moderate. A close look at the TCP-friendly equation [16] can reveal one possible explanation of this abnormality:

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



(a) Increase Trajectory (b) Metrics Definition (c) AIMD vs SIMD

Figure 2. Comparison of convergence speed

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

Table 1. Performance measures on convergence to fairness and efficiency

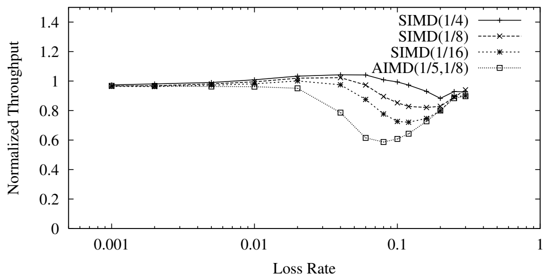


Figure 3. TCP-friendliness

When loss rate is low, TCP mainly stays in the *congestion avoidance* stage, and the AIMD algorithm dominates this equation. When loss rate is very high, TCP spends most of its time retransmitting packets, and the *exponential back-off* algorithm dominates this equation. Since all TCP variants studied in this paper use the same timeout mechanism as standard TCP, and they carefully calibrate the values of parameters during congestion avoidance to match standard TCP, they can achieve comparable throughput as standard TCP for very high and low loss rates. However, for the loss regime in between, it becomes hard, if not impossible, to obtain α and β values that would approximate well both the congestion avoidance and the exponential back-off components of this equation [22].

Nevertheless, in the worst case with loss rate around 15%, SIMD(1/16), which is the worst among all the 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). Given the fact that most parts of the Internet are experiencing less than 5% loss rate [4], our algorithm is TCP-friendly under these conditions.

The weakness of AIMD(α, β) with small β under medium loss conditions is also reported in [8, 22]. The authors try to compensate for the bandwidth loss by increasing the value of α. However, when loss rate is small (e.g. less than 3%), AIMD with large α could achieve significantly higher bandwidth than standard TCP and become less TCP-friendly. Therefore, we maintain the theoretical α values throughout our simulations.

4.2. TCP-Compatibility

We use the method described in [8] 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 4 and Figure 5 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, SIMD achieves a slightly lower average throughput than standard TCP when

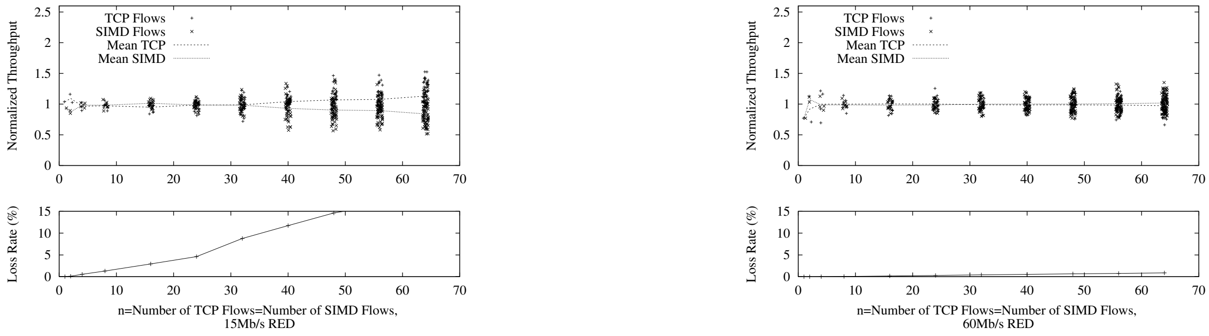


Figure 4. TCP competing with SIMD(1/16), RED with ECN

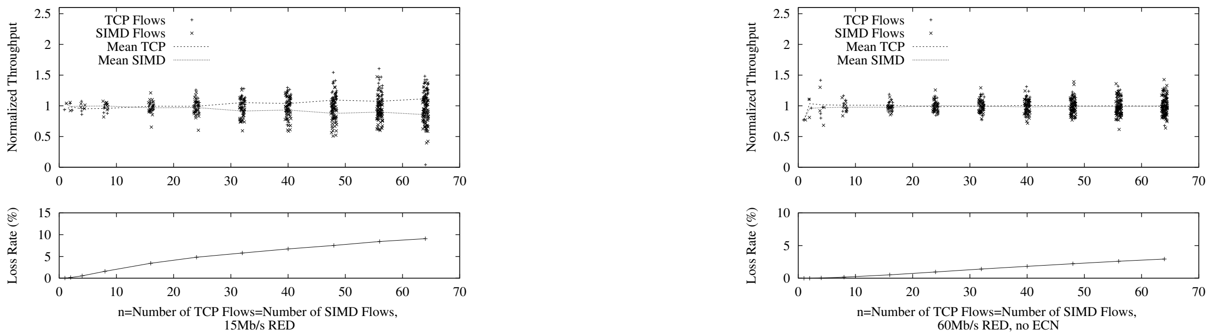


Figure 5. TCP competing with SIMD(1/16), RED without ECN

when the loss rate exceeds a certain level. This is partly due to the reason we illustrate in Figure 3. Another possible explanation is that when severe congestion happens, SIMD can not compete well against standard TCP since compared to TCP, SIMD opens its congestion window more conservatively at the beginning of each congestion epoch. Therefore, when the time between two consecutive packet losses is short, the more aggressive TCP tends to gain more throughput. However, in a reasonable loss regime (loss rate below 10%), SIMD shows very impressive TCP-compatibility. Note that in the 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.

We also found that with DropTail queue management, as shown in Figure 6, 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 bandwidth than standard TCP compared to the previous experiment. Note that the assumption of randomized packet losses made in our analysis does not apply to DropTail. Under DropTail, packet losses tend to be 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,

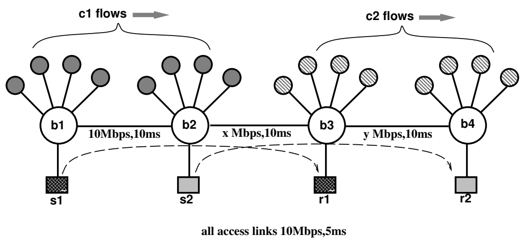


Figure 7. Simulation topology for convergence test

and the side effect of a DropTail queue (correlated drops for each flow) is thus not so significant.

4.3. Convergence to Fairness and Efficiency

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

We use the topology shown in Figure 7 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

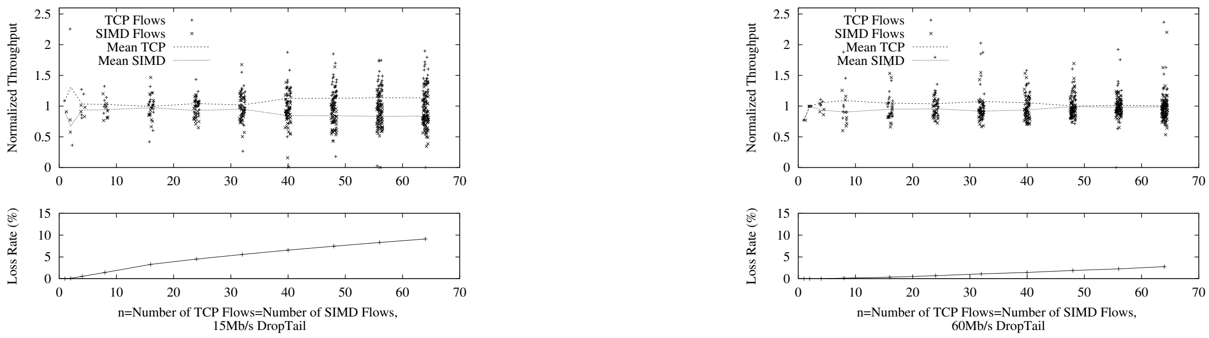


Figure 6. TCP competing with SIMD(1/16), with DropTail

(s2-r2) stay to compete for the bottleneck link ($b2$,$b3$). We use packet size of 500 bytes in these experiments.

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

We create this scenario to study the convergence time to fairness given that the initial point (W_1, W_2) is on the efficiency line ($w_1 + w_2 = W$). To create this setup, we let $c_1 = 15$, $c_2 = 0$, $x = 6$Mbps, $y = 6$Mbps. So the bottleneck link for flow (s2,r2) remains link ($b2$,$b3$), but for flow (s1,r1), the bottleneck changes from link ($b3$,$b4$) to ($b2$, $b3$) at time 400. We can also compute that: $W \approx 110$, $W_1 \approx 7$, and $W_2 \approx 100$. Figure 8 plots the transient behavior of the congestion window of different protocols.

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

Table 3 gives the convergence time to fairness (T_2). Here we use $\epsilon = 10$ packets (cf. Section 3.2). The theoretical value is also given in the table for comparison. The following observations can be made from the table.

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

Second, SIMD benefits from RED much more than other schemes. The T_2 value from simulations is much smaller than the value obtained from analysis (shown in boldface). This is because RED allows SIMD flows with smaller windows to experience fewer packet losses, which gives them a better chance to become more aggressive. 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.

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

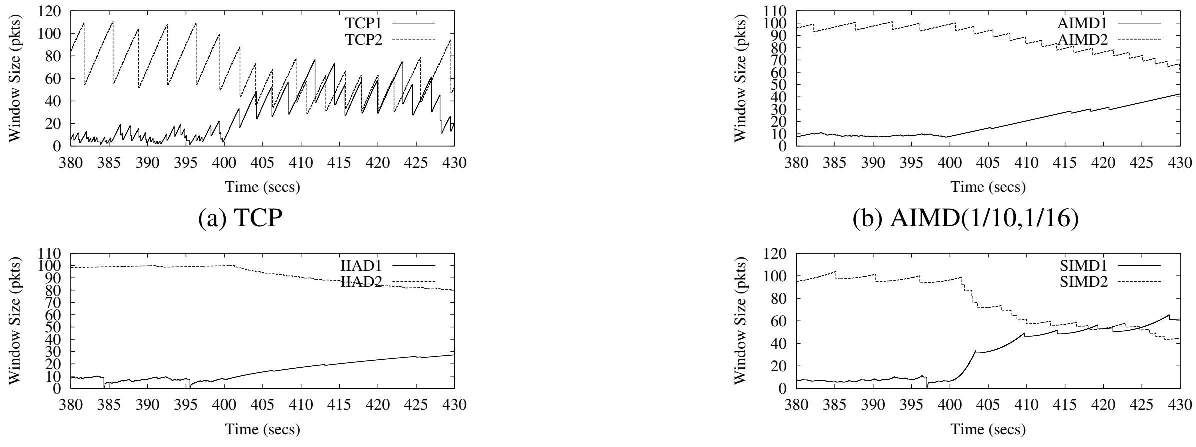
To create such scenario, we let $c_1 = 11$, $c_2 = 3$, $x = 6$Mbps, $y = 10$Mbps. So initially the bottleneck link for flow (s1,r1) is ($b1$,$b2$), and for flow (s2,r2) the bottleneck is ($b3$,$b4$). But at time 400, both of them switch to link ($b2$, $b3$). Roughly, we have $W \approx 110$, $W_1 \approx 10$, and $W_2 \approx 30$. We can then study T_1, the convergence time to efficiency of different control schemes. Figure 9 plots the transient behavior of the congestion window of different protocols.

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 windows at the same rate and thus do not efficiently converge to the fair share. On the contrary, our SIMD algorithm allows the two competing flows to 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 3. Analytical results closely match the simulation results.

5. Related Work

The earliest congestion controls include Jacobson's TCP algorithm [12] and Ramakrishnan and Jain's DECbit scheme [18]. Under a synchronized feedback assumption, Chiu and Jain [2] analyze AIMD control. 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 [8, 22] and TEAR [19].



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

(c) IIAD (d) SIMD(1/16)

Figure 8. 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)		Δ (pkts)	
			simu	anal			simu	anal	simu	anal
TCP	6.1	99.6	68.0	88.7	8.8	13.8	55	43.7	5.8	6.0
AIMD	7.9	99.2	776	1217	12.7	31.0	349	342	18.6	18.3
IIAD	7.7	99.8	4232	6684	11.8	31.2	1284	1242	8.1	7.6
SIMD	6.6	96.3	**218**	**852**	10.2	33.2	90	85.1	13.6	12.3

Table 3. Quantitative measures on convergence time

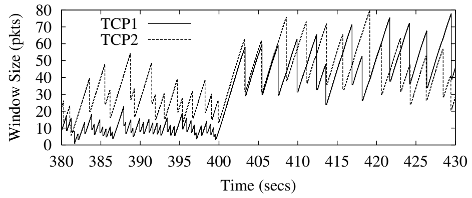
These mechanisms use a moderate window decrease parameter to reduce rate variability, meanwhile use a matching window increase parameter to satisfy TCP-friendliness. There are tradeoffs between smoothness and reaction to changes in network conditions [8, 21].

Non-linear controls were considered less robust and not suitable for practical purposes [2]. On the contrary, Bansal and Balakrishnan [1] proposed binomial algorithms that interact well with TCP AIMD. Binomial controls are memory-less in that they use only the current window size in their control rules. SIMD is radically different from memory-less binomial algorithms. To our knowledge, SIMD is the first window-based TCP-friendly congestion control algorithm using history information in its control rules. By doing so, SIMD improves its transient behavior and convergence speed without sacrificing smoothness in steady state.

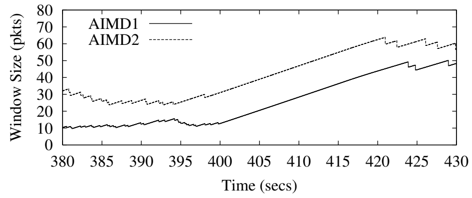
Another approach to provide smoother transmission rate is equation-based congestion controls [9, 17, 20], first proposed in [14]. In these schemes, end-systems measure the packet loss rate and round-trip time, and use the TCP-friendly equation [16] to compute the transmission rate. Two comparisons [8, 21] of equation-based and window-based congestion controls have shown that equation-based schemes and window-based AIMD share similar transient behaviors 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-clocked mechanism for overload protection as in window-based control. Notably, equation-based schemes use more history information (up to eight congestion epochs [9]). Therefore, SIMD is a step toward exploring the design space between window-based memory-less control schemes and equation-based schemes that make use of longer history.

6. Conclusion

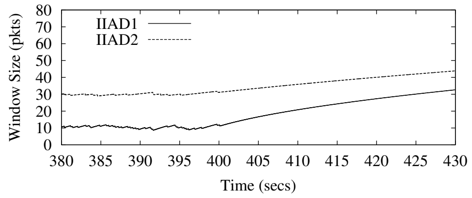
We proposed a novel window-based congestion control algorithm called SIMD (Square-Increase/Multiplicative-Decrease). Contrary to previous memory-less controls, SIMD utilizes history information in its control rules. It uses multiplicative decrease but the window size increases in proportion to the square of the time elapsed since the detection of the last loss event. Thus, SIMD can maintain smoothness in steady state, while efficiently probing available bandwidth when there are drastic changes in network conditions. We have shown that SIMD is TCP-friendly as well as TCP-compatible under RED. We have also shown that SIMD has faster convergence than TCP-friendly memory-less AIMD and binomial algorithms. Our simulations using the ns simulator have demonstrated the superiority of SIMD.



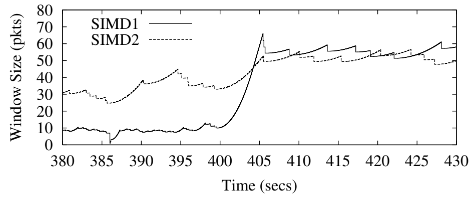
(a) TCP



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



(c) IIAD



(d) SIMD(1/16)

Figure 9. Two flows converge to fair share of bandwidth

SIMD is the first example of window-based congestion control algorithms that uses history information in its control rules. It explores a new design space between memoryless window-based congestion control schemes and equation-based schemes that use history spanning many congestion epochs. Indeed, this new space defines a new class of TCP-friendly window-based congestion control algorithms, of which SIMD is an instance [13]. Future work includes comparisons between equation-based schemes and SIMD.

References

[1] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *Proceedings of IEEE INFOCOM*, April 2001.

[2] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[3] M. Christiansen, K. Jeffay, D. Ott, and F. Smith. Tuning RED for Web Traffic. In *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, Aug.-Sep. 2000.

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

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

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

[7] 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.

[8] 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.

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM*, August 2000.

[10] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):393–417, August 1993.

[11] S. Gorinsky and H. Vin. Additive increase appears inferior. Technical Report TR2000-18, Department of CS, Univ. of Texas at Austin, May 2000.

[12] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM*, August 1988.

[13] S. Jin, L. Guo, I. Matta, and A. Bestavros. A spectrum of TCP-friendly window-based congestion control algorithms. Technical Report BU-CS-2001-015, Computer Science Department, Boston University, July 2001. Available at http://www.cs.bu.edu/techreports/2001-015-spectrum-tcp-friendly.ps.Z.

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

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

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] Y. R. Yang and S. S. Lam. General AIMD congestion control. In *Proceedings of ICNP*, November 2000.