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

Paper ID: 255
Total Pages: 20 (+ 8 pages of appendix, which will not appear in the final version but in a referenced technical report)

Abstract

The increased diversity of Internet application requirements has spurred a recent interest in transport protocols with flexible transmission controls. In window-based congestion control schemes, increase rules determine how to probe and efficiently consume 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 previously proposed 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. SIMD is one instance of this spectrum of algorithms, in which the congestion window is increased super-linearly with time since the last loss. Compared to recently proposed AIMD and binomial algorithms, we show the superiority of SIMD in adapting to sudden increases in available bandwidth, while maintaining competitive smoothness and responsiveness.

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 [13] 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. It has been shown that AIMD *converges to fairness* under synchronized feedback assumption [4].

Recently proposed congestion control mechanisms include equation-based schemes [10, 20, 26] and generalization of TCP-like window-based schemes [9, 23, 27]. One common objective is to tackle TCP's high rate variations. Such high variations may limit network utilization. Furthermore, they are not desirable for emerging applications such as real-time streaming applications on the Internet. However, these new protocols must implement congestion control mechanisms that interact well with TCP [8]. 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 $\lambda = 1/(\sqrt{2/3}R\sqrt{p})$ should hold, where λ is the throughput of a flow, p is the loss rate, and R is the round trip time.

In addition, *smoothness, aggressiveness*, and *responsiveness* [9, 27] are also important indices of a congestion control algorithm. Smoothness of a protocol measures the transmission rate variability of a connection using the protocol. Smoothness is important in stationary environments. Aggressiveness means how fast the connection probes extra bandwidth. Responsiveness means how fast the connection reacts to increased congestion and decreases its window size promptly. Both aggressiveness and responsiveness indicate the transient behaviors of a congestion control protocol [27]. An important observation is that there are tradeoffs among smoothness, aggressiveness, and responsiveness [9, 27]. Comparisons of TCP AIMD[1], general AIMD [9, 28], TFRC [10], and TEAR [23] indicated that typically higher smoothness means less aggressiveness and responsiveness.

1.1 Our Contribution

We consider TCP-like window-based congestion control algorithms. The first property of our algorithms is TCP-friendliness. Contrary to the memory-less TCP, we consider the case of stateful schemes, by which we mean that in addition to the current window size, connections utilize history information. The only history we use is the window size at the time of detecting the last loss. This allows us to explore much broader choices of TCP-friendly congestion control mechanisms than AIMD and other schemes [3]. The second property of our algorithms is convergence to fairness under reasonable assumptions. Under the synchronized feedback assumption used by Chiu and Jain [4], we demonstrate that our algorithms can possess better convergence behaviors than memory-less AIMD and other schemes. The third property of our algorithms is the flexibility in trading off among smoothness, aggressiveness, and responsiveness. Contrary to other schemes, we demonstrate that connections using our proposed algorithms can have smooth transmission rates while probing available bandwidth much faster than other schemes and responding to increased congestion promptly.

A TCP-like window-based congestion control scheme increases the congestion window as a result of the successful transmission of a window of packets (in one round-trip time R), and decreases the congestion window upon the detection of packet loss. We call such sequence of window increments followed by one window decrement as 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:

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

Binomial algorithms [3] use non-linear control. Neither of them utilizes history information. On the contrary, we find the window size at the end of the last congestion epoch handy and useful for control rules. Our scheme maintains such a state variable w_{max}. Let $w(t)$ be the window size at time t (in RTT's) elapsed since the window started to increase. Let $w_0 = w(0)$. Given a decrease rule, w_0 can be computed from w_{max}. Henceforth, for clarity, we use both w_{max} and w_0.

An observation is, if we use an increase rule of the form:

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

where w_t is the window size at discrete time t since the last decrease, and α is a constant independent of t, then we have the following continuous approximation of the window size as a function of time:

$$w(t) = w_0 + ct^u, \qquad u, c > 0, \tag{2}$$

where $u = 1/(k+1)$ and $c = ((k+1)\alpha)^u$. Appendix A shows that (1) and (2) are equivalent. We are interested in congestion control schemes that have various window size increasing patterns (different u's, or equivalently, different k's). This allows us to investigate various schemes with different degrees

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

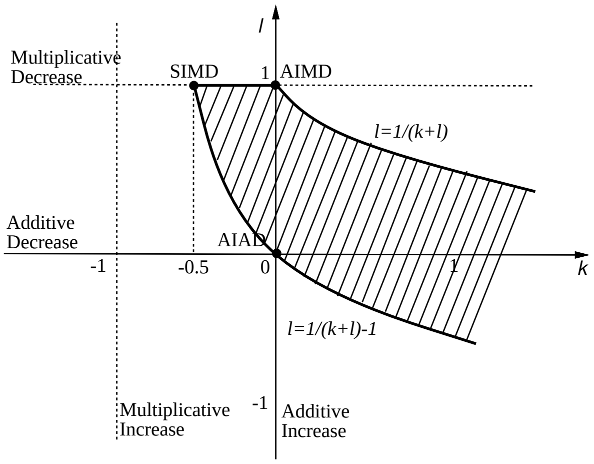


Figure 1: A spectrum of TCP-friendly congestion control algorithms.

of aggressiveness. For example, this flexibility can support applications that need to quickly acquire bandwidth as it becomes available. Thus this paper considers the following control rules:

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

We consider $k > -1$, since otherwise the window size increases exponentially (or faster) and the system is unstable. We consider $l \leq 1$, since otherwise $w_t - \beta w_t^l$ can be negative when w_t is large enough.

Our work can be summarized by Figure 1 (explained in Section 2). There are many choices of TCP-friendly congestion control algorithms using our control rules in (3). In particular, the spectrum shown by the shaded area can have better convergence to fairness behaviors than AIMD and other schemes. This provides much broader choices of congestion controls for various applications, especially applications that are required to be TCP-friendly and need low transmission rate variations. In particular, if $k < 0$, the congestion window increases super-linearly; if $k = 0$, the window increases linearly; and if $k > 0$, the window increases sub-linearly. If $l = 1$, the window decreases multiplicatively. When $k = 0$ and $l = 1$, the spectrum reduces to the memory-less AIMD control. We simulated a special case SIMD with $k = -0.5$ and $l = 1$. It uses multiplicative decrease but the window increases in proportion to the square of the time elapsed since the last loss (since $u = 2$ in Equation (2)). We show that SIMD converges to fairness faster than AIMD and other schemes, and it can achieve smoothness of transmission rates while probing bandwidth much faster.

We propose and analyze our algorithms in Section 2. Simulation results are presented in Section 3. We revisit related work in Section 4 and finally conclude the paper.

2 TCP-friendly Non-linear Controls

2.1 TCP-Friendliness

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

In Appendix B, assuming steady state and without considering the effect of TCP's timeout mechanisms, we show that the following definition of α makes our congestion control schemes approximately

TCP-friendly:

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

where $\Gamma(.)$ is the Gamma function. For positive integer n, $\Gamma(n+1) = n!$. Equivalently, according to Appendix A, c in Equation (2) is defined as:

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

Notice that when the transmission rate variation is low, $w_{max} >> \beta w_{max}^{l}$. Then α and c expressions are greatly simplified, i.e., α becomes the product of a constant and w_{max}^{kl+l-1}, and c becomes the product of a constant and $w_{max}^{l-1/(k+1)}$.

Notably, this control scheme has several special cases. When $k = 0$ and $l = 1$, from (4) we have $\alpha_{AIMD} = 3\beta/(2-\beta)$. It degenerates to the memory-less TCP-friendly AIMD control [9]. When $k = -0.5$ and $l = 1$,

$$\alpha_{SIMD} = \frac{3\sqrt{\beta}}{(1-\frac{2\beta}{3})\sqrt{2w_{max}}}. \tag{6}$$

In this case, the window size decreases multiplicatively on detecting packet losses, but increases in proportion to the *square* of the time elapsed since the last loss, since in Equation (2), $u = \frac{1}{k+1} = 2$. We call this algorithm SIMD (Square-Increase/Multiplicative-Decrease). SIMD is extensively studied in the remainder of this paper.

2.2 Convergence to Fairness and Efficiency

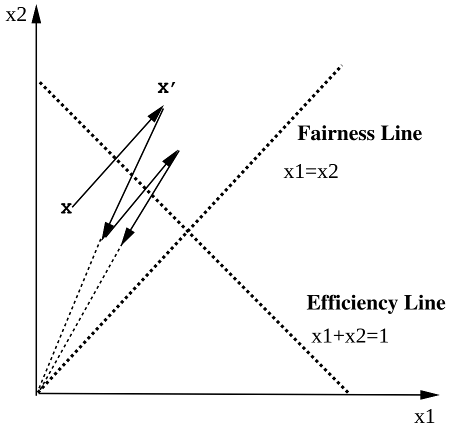
In this subsection, we first show the convergence conditions of our congestion control schemes. Then we show they can converge faster than AIMD.

2.2.1 Convergence Condition

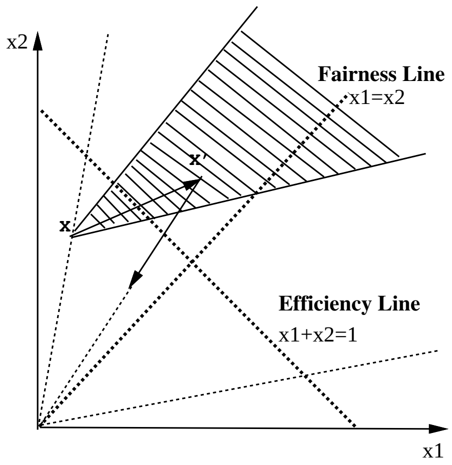
We adopt the ideal synchronized feedback assumption [4]. To show that multiple users with synchronized feedbacks using our control scheme converge to fairness, we use the vector space used by Chiu and Jain [4] to view the system state transitions as a trajectory. For ease of presentation, we show a two-user case. It is straightforward to apply the same technique to the multiple-user case to reach the same conclusion.

As shown in Figure 2(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 $\max(\frac{x_1}{x_2}, \frac{x_2}{x_1})$. The closer to unity the fairness index, the more fair the allocation. 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, without loss of generality, assuming $x_1 \leq x_2$, AIMD increases the resource allocation of both users by a constant. In Figure 2(a), the trajectory is shown as the line parallel to the fairness line. This movement improves fairness. 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.

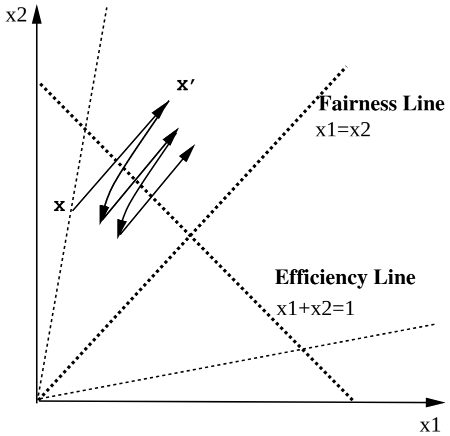
For our control schemes, we first observe Equation (2) and the definition of c in Equation (5). Ignoring the constant factor in (5), we can see that the window size of a connection increases in proportion to $x_i^{l-1/(k+1)}, i = 1, 2$. To ensure that the increase trajectory moves towards a more fair point, we can choose



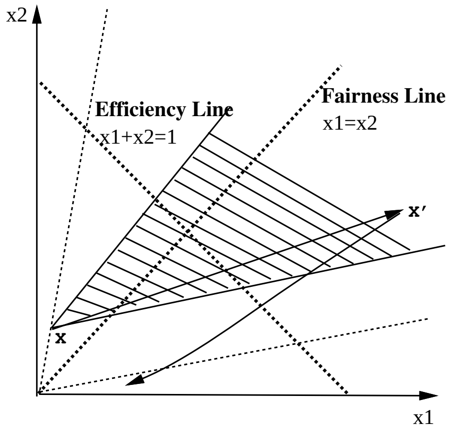
(a) AIMD convergence trajectory



(b) $l = 1$, Multiplicative Decrease



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



(d) $l < \frac{1}{k+1}$ and $l < 1$

Figure 2: Convergence of our congestion control schemes.

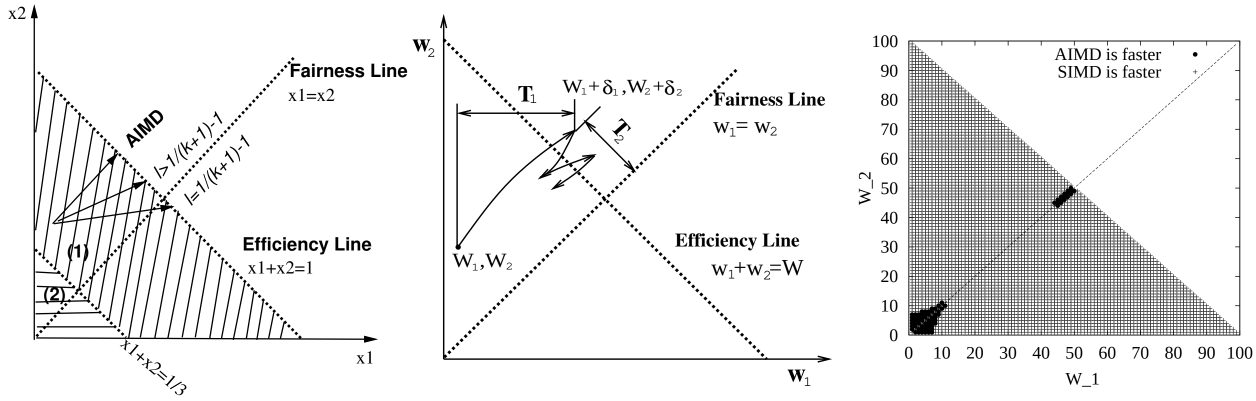
$-1 \leq l - \frac{1}{k+1} \leq 0$. This condition is shown by the shaded area in Figure 1. Under this condition, the trajectory in Figure 2(b) shows the system moves in the shaded area between the 45° additive increase line and the line emanating from $X(x_1, x_2)$ with slope $\frac{x_1}{x_2}$. However, different areas of the spectrum in Figure 1 exhibit different convergence behaviors. 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 2(b). Hence decrease does not change the fairness index. Since the increase has been shown to improve the fairness index, repeated increases and decreases consistently improve fairness. Finally, the system stays on the fairness line and oscillates around the efficiency line.

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

In this case, the window size of a connection increases with time, but the increasing pattern is *independent* of w_{max}, i.e., $x_i, i = 1, 2$. This means that the two users increase the resource allocation at the same speed. The increase trajectory follows exactly the additive increase trajectory, shown in Figure 2(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 more fair than the intersection of the increase



(a) Increase Trajectory

(b) Metrics Definition

(c) AIMD vs SIMD

Figure 3: Comparison of convergence speed

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. When the increase trajectory crosses the fairness line, shown in Figure 2(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-clocked nature of window-based controls. Under this assumption, Appendix C gives a condition to ensure the convergence of our schemes 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 conditions.

2.2.2 Convergence Speed

We first intuitively show the condition for our control schemes to converge 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, we show that the increase trajectory of our control schemes 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 3(a), thus it is easier for our schemes to reach a more fair intersection. 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 our control scheme with $l - \frac{1}{k+1} = -1$, the intersection is $(x_1 - x_2 + \frac{x_2}{x_1+x_2}, x_2 - x_1 + \frac{x_1}{x_1+x_2})$. 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 3(a). Therefore, we claim that our schemes usually converge faster than AIMD.

Then, we analytically compare the convergence time of our control schemes, general AIMD [9, 28], and binomial controls schemes [3]. For our control schemes, we choose the case of $k = -0.5$ and $l = 1$, i.e., the SIMD algorithm as a representative. For binomial controls, we choose IIAD (Inverse-Increase Additive-Decrease) [2] as a representative. We still assume synchronized feedback and use Figure 3(b) to

[2]Briefly speaking, binomial controls increase the window inversely proportional to w^k, and decrease the window propor-

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 $W_2 \geq W_1$ 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 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 Δ. The detailed analysis is given in Appendix D and we only present the main results here in Table 1.

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

Table 1: Performance measures on convergence to fairness and efficiency

We also numerically solve the above equations for different initial points. Figure 3(c) shows the regions for which SIMD (with $\beta = 1/16$) converges faster/slower (i.e., $T_1 + T_2$ is smaller/larger) than AIMD (with $\alpha = 1/10$ and $\beta = 1/16$) for $\epsilon = 1$ and $W = 100$. In most cases SIMD converges faster than AIMD, which supports our intuitive claim (cf. Figure 3(a)). Numerical results also show that IIAD (with $\alpha = 1$ and $\beta = 2/3$) is much slower than AIMD and SIMD in all cases.

2.3 Tradeoffs among Smoothness, Aggressiveness, and Responsiveness

In this subsection, we consider important indices of congestion control algorithms other than TCP-friendliness and convergence. These indices are smoothness, aggressiveness, and responsiveness. Smoothness indicates the variability of the window size of a connection. High variability is not desirable. Aggressiveness means how fast a connection probes available bandwidth. High 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 indices of transient behaviors.

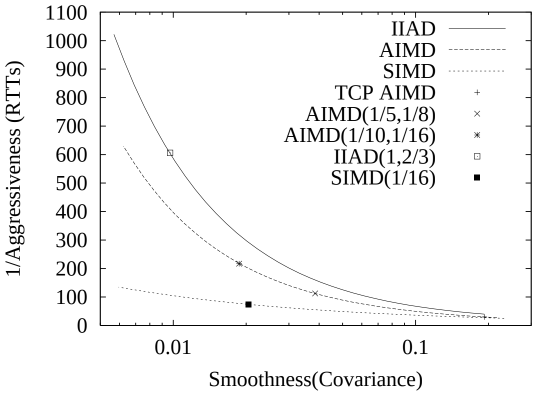
Smoothness can be observed at different time scales [9]. We consider short time scales since long-term smoothness can be affected by other dynamics in the system. We define smoothness as the variability of the window size of a connection in one congestion epoch. In particular, we use the coefficient 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 enough to give us insights into the tradeoffs. We define aggressiveness as the inverse of the time needed for the connection to increase the window size (say, from W_{aggr}), in response to a step increase of available bandwidth [27]. 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 (say, from W_{resp}) by a substantial portion, in response to a step increase of congestion [27]. That is, a substantial decrease of available bandwidth by a factor of m.

Table 2 gives the indices for the general AIMD, IIAD and SIMD algorithms. The derivations are given in Appendix E.

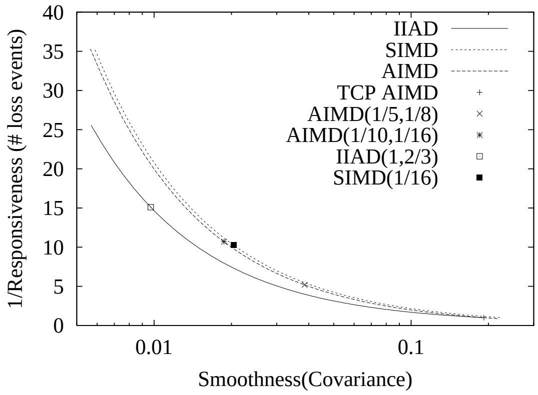
tional to w^l, so IIAD corresponds to $k = 1$, $l = 0$.

	Smoothness	1/Aggressiveness	1/Responsiveness
AIMD	$\frac{0.29\beta}{1-\beta/2}$	$\frac{m-1}{\beta}\frac{2W}{3}$	$\log_{(1-\beta)}\frac{1}{m}$
IIAD	$\frac{0.29\beta}{W-\beta/2}$	$\frac{(m^2-1)W^2}{3\beta}$	$\frac{W(1-1/m)}{\beta}$
SIMD	$\frac{0.30\beta}{(1-2\beta/3)^2}$	$\sqrt{\frac{m-1}{\beta}\frac{2\sqrt{2}W}{3}}$	$\log_{(1-\beta)}\frac{1}{m}$

Table 2: Smoothness, Aggressiveness, and Responsiveness comparisons of AIMD, binomial controls, and our control scheme.



(a) Aggressiveness vs Smoothness.



(b) Responsiveness vs Smoothness.

Figure 4: 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, which we take to be W.

Numerical analysis in Figure 4 shows the tradeoffs among smoothness, aggressiveness, and responsiveness. Figure 4(a) shows the inverse of aggressiveness of AIMD, SIMD, and IIAD. Their special cases of 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 RTT's necessary to double the window size ($m = 2$). Figure 4(b) shows the inverse of responsiveness of AIMD, IIAD, and SIMD as the coefficient of variation changes. The inverse of responsiveness is computed assuming the objective window size is half of the current window size ($m = 2$).

From this figure, we can see that SIMD has much better aggressiveness (fewer RTT's) than the others, especially when high smoothness (low coefficient of variation) is needed. Meanwhile, SIMD has comparable responsiveness index, i.e., the number of loss events necessary for SIMD to decrease the window size is close to the others. For example, we can predict that AIMD(1/20,1/30), SIMD(1/30), and IIAD(1,2/3) have comparable smoothness when the average window size is 20. However, SIMD(1/30) can react to a substantial increase of available bandwidth much faster. The smoothness-aggressiveness relationship can also be inferred from Table 2. For both AIMD and IIAD, aggressiveness varies proportional to coefficient of variation. For SIMD, aggressiveness varies proportional to the square root of coefficient of variation. Thus, for smoother transmission rates, SIMD maintains higher aggressiveness than AIMD and IIAD.

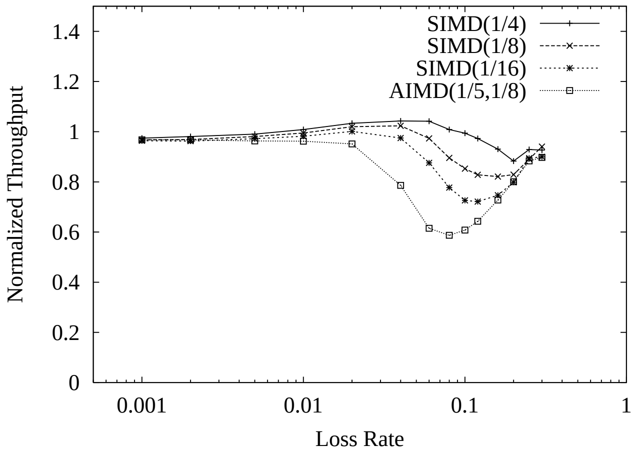


Figure 5: TCP Friendliness

3 Simulation Results

We used *ns* simulations [6] to validate that with RED [11] scheduler (or randomized dropping), our proposed algorithm is TCP friendly and TCP compatible. In addition, we compared SIMD to standard TCP [13], generalized AIMD [28], and IIAD [3], in terms of smoothness, responsiveness, and aggressiveness. We also investigated the way two homogeneous flows converge to their bandwidth fair share and found that our proposed algorithm outperforms other algorithms.

Unless explicitly specified, in all of the experiments, RED was used as the bottleneck link scheduling algorithm. The maximum queue size Q at the bottleneck link was set to the bandwidth-delay product of the path. The minimum and maximum drop thresholds (min_{th} and max_{th}) were set to $0.2 \times Q$ and $0.8 \times Q$, respectively. Packet size was set to 1000 bytes. 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 [16] for congestion detection. We also obtained similar results for other mechanisms (e.g. Reno, newReno). We assume no delayed acknowledgments.

3.1 TCP Friendliness and TCP Compatibility

3.1.1 TCP Friendliness

We conducted the following experiment to test 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 varied the loss rate p and compared the normalized (with respect to standard TCP) long-term (around 3000 RTT) throughput of SIMD for different β values and plotted them in Figure 5. For comparison, we also plotted AIMD(1/5,1/8) throughput in the figure.

We notice that all of the curves have a dip when the loss rate is moderate. A close look at the TCP friendly equation (7) [19] can reveal one possible explanation of this abnormality.

$$\lambda(p, \alpha, \beta) \approx \min\left(\frac{W_{max}}{R}, \frac{1}{R\sqrt{\frac{2\beta}{\alpha(2-\beta)}p} + T_0 \min(1, 3\sqrt{\frac{\beta(2-\beta)}{2\alpha}p})p(1+32p^2)}\right) \quad (7)$$

When loss rate is low, TCP mainly stays in the *congestion avoidance* stage, and AIMD algorithm dominates the TCP-friendly equation (7), while when loss rate is very high, TCP spends most of its time retransmitting packets, and the *exponential back-off* algorithm dominates equation (7). 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 congestion avoidance and exponential backoff components of the TCP-friendly equation [28].

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)[3]. Given the fact that most parts of the Internet are experiencing less than 5% loss rate [5], our algorithm is TCP-friendly under these conditions.

3.1.2 TCP Compatibility

We use the method described in [9] 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 have tested the cases with RED and with DropTail queues, with and without ECN. Figure 6 shows the simulation results for RED queues with ECN bit set. The bottleneck link bandwidth are 15Mbps and 60Mbps, respectively. Each point in the graph represents the throughput of an individual flow, and the dashed lines represent the average throughput of SIMD and standard TCP flows, respectively. The lower graphs show the corresponding average loss rates in each case.

As can be observed from the graphs, SIMD achieves a slightly lower throughput than standard TCP when available bandwidth is high, but it gets much lower share of the bandwidth when loss rate exceeds certain level. This agrees with the results shown in Figure 5. It also shows that TCP-friendliness implies TCP-compatibility across RED gateways [3].

We also found that with DropTail queue management, as shown in Figure 7, SIMD can still be TCP-friendly and TCP-compatible. The difference, compared to the RED queue experiment, is that the variance becomes larger and SIMD now gets slightly more bandwidth than standard TCP as the bandwidth fair share becomes larger. 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). Thus, SIMD can become more aggressive than standard TCP when no loss happens for a long time. 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 DropTail queue (correlated drops for each flow) is thus not so significant. Nevertheless, we can set up a simulation scenario (not shown) in which synchronized losses cause SIMD to grab much more bandwidth than standard TCP, so SIMD becomes less TCP-friendly. The positive side, however, is that SIMD is more friendly than IIAD in the same scenario since SIMD performs multiplicative (instead of additive) decrease upon detecting congestion.

[3] The weakness of AIMD(α, β) with small β under medium loss conditions is also reported in [9]. The authors try to compensate for the bandwidth loss by increasing the value of α. However, when loss rate is small (e.g. less than 3%), AIMD with large α could achieve significantly higher bandwidth than standard TCP and become less TCP-friendly. Therefore, we maintain the theoretical α values throughout our simulations.

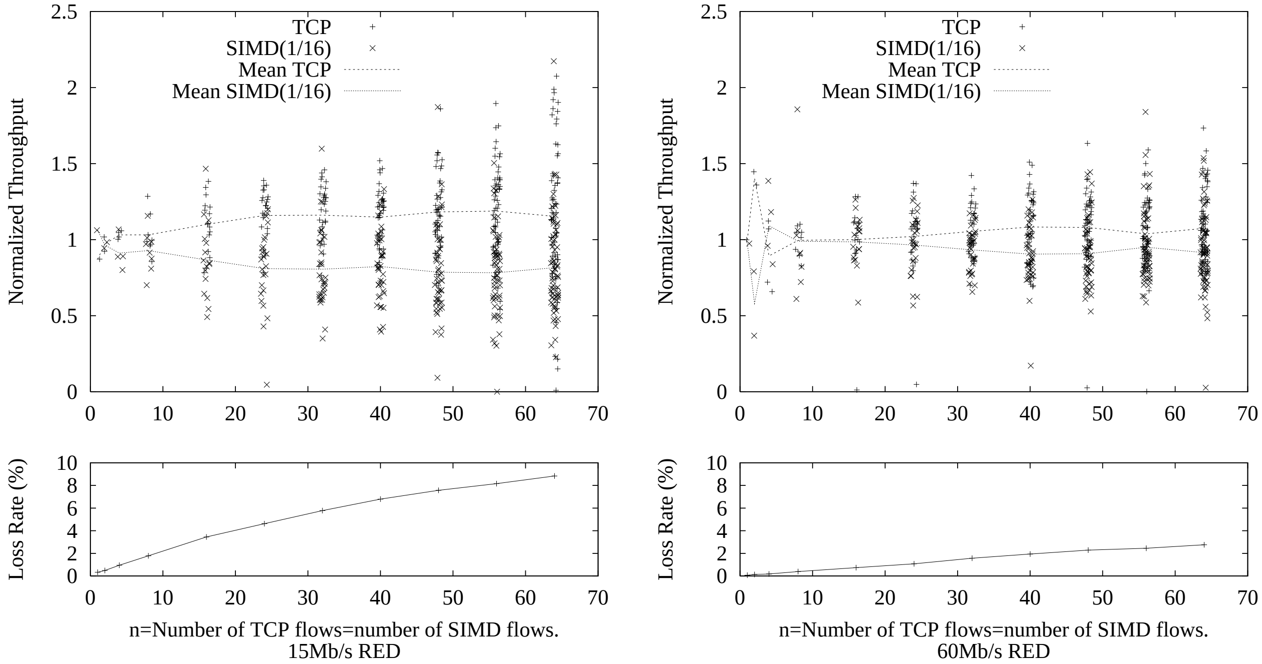


Figure 6: TCP competing with SIMD(1/16), with RED

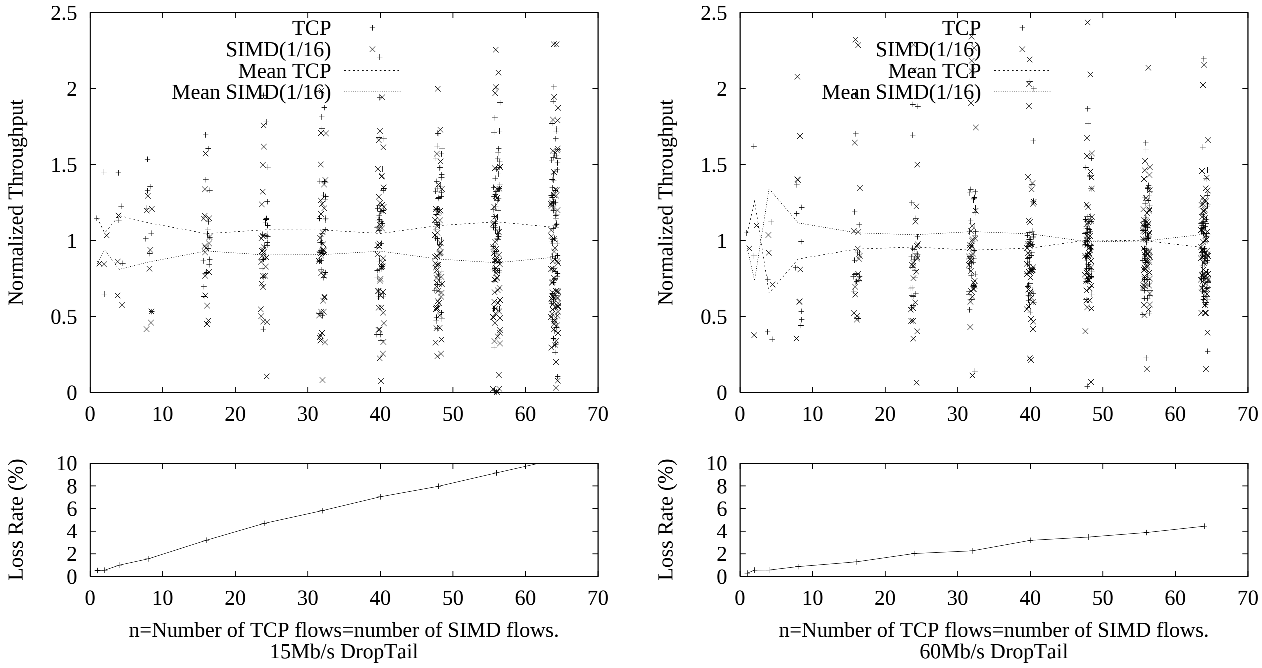
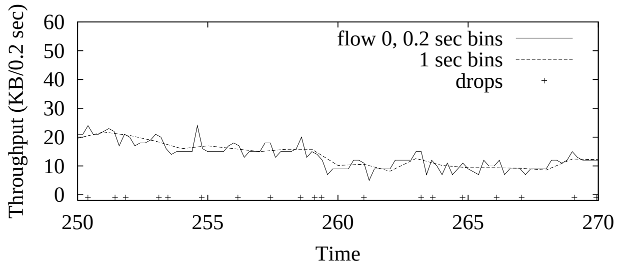
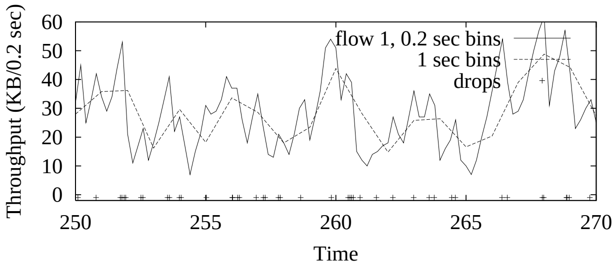


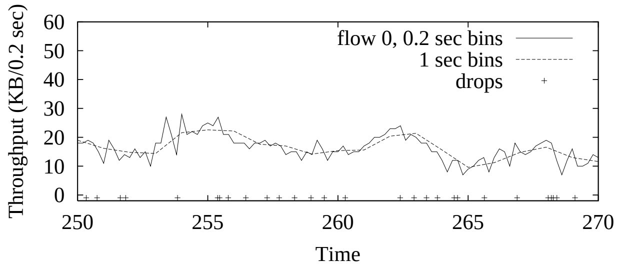
Figure 7: TCP competing with SIMD(1/16), with DropTail



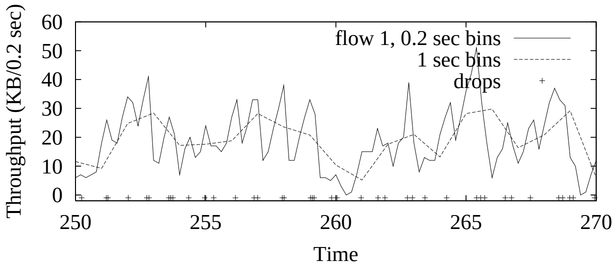
(a) AIMD(1/10,1/16) with TCP



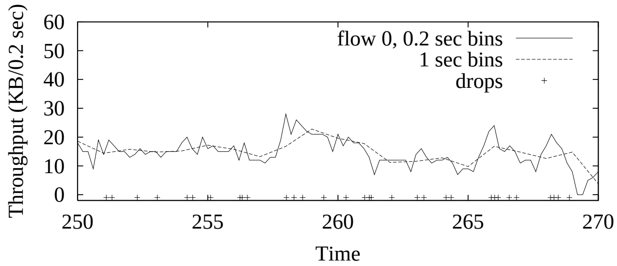
(b) TCP with AIMD(1/10,1/16)



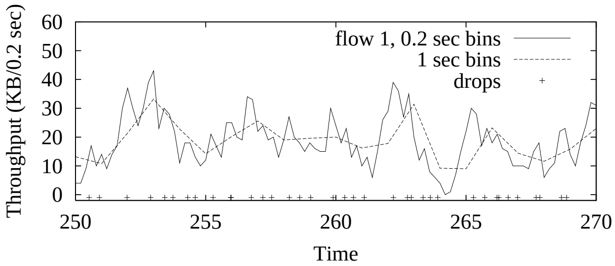
(c) IIAD with TCP



(d) TCP with IIAD



(e) SIMD(1/16) with TCP



(f) TCP with SIMD(1/16)

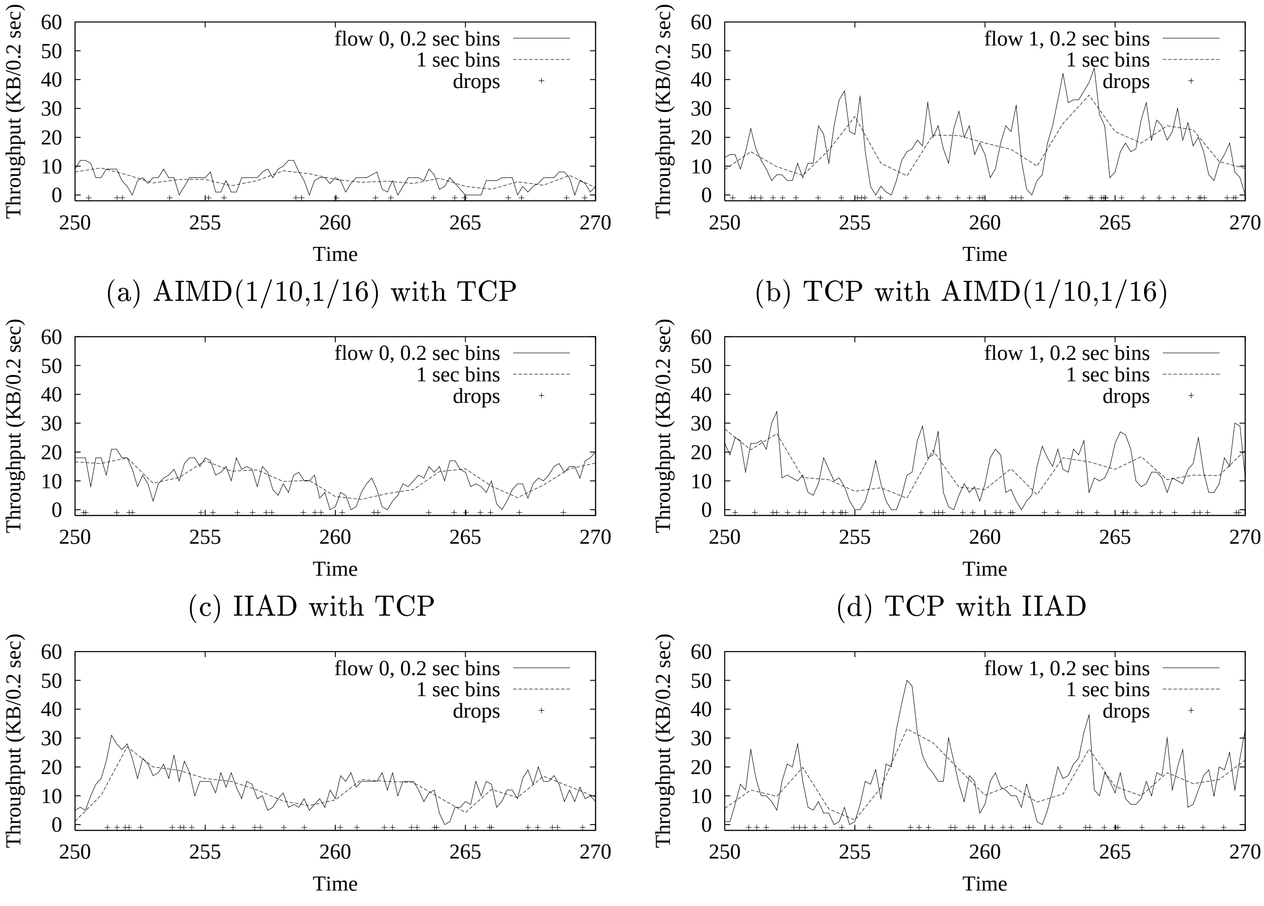
Figure 8: Traffic smoothness, 10 + 10 TCP flows

3.2 Smoothness, Responsiveness and Aggressiveness

3.2.1 Smoothness

As revealed by the study in [9], 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 [9] 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 8 we show the case for $n = 10$ and $C = 15Mbps$, which corresponds to roughly 3% packet drop rate. 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 and 1 second intervals, which correspond to twice and ten times a typical round-trip time for this simulation, respectively. As in [9], we also plot the time at which a packet is dropped by a "+" sign at the bottom of the graph. The queues use RED management without ECN. Similar results were obtained when ECN is turned on.

We can observe from the graphs that all three algorithms AIMD, IIAD and SIMD have roughly the same scale of short-term burstiness. This agrees with our analysis (cf. Section 2.3). In particular, by



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

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

(c) IIAD with TCP

(d) TCP with IIAD

(e) SIMD(1/16) with TCP

(f) TCP with SIMD(1/16)

Figure 9: Traffic smoothness, 16 + 16 TCP flows

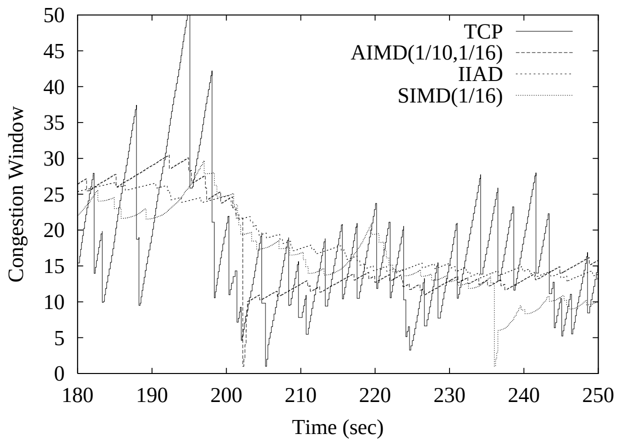
plugging in equations of Table 2 the values we choose in our simulation of $\beta = 1/16$ for AIMD and SIMD, and $\beta = 2/3$ for IIAD, and since the average window size in this simulation is about 10 packets, or $W = 10$, we find that the coefficients of variation of these three algorithms are close to each other. Our experiment results show that this is indeed the case.

We also increased n to 16 (thus increase the congestion level to roughly 5% loss rate) in another experiment set. The results are shown in Figure 9.

We found that in this scenario, the throughput of AIMD degrades significantly. This is in part due to the reason mentioned in Section 3.1.1, 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 3.3). Thus, when no loss happens, TCP increases its congestion window more aggressively and gets higher throughput than AIMD, which eventually gives up the fair share it deserves. SIMD overcomes this problem and can achieve similar throughput as TCP in this scenario. Since the average throughput of SIMD is higher than that of AIMD, the variability of SIMD traffic is also higher since they have similar coefficient of variation (cf. Section 2.3).

3.2.2 Response to step increase of congestion (Responsiveness)

We conducted 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,



(a) Response to bandwidth decrease

(b) Response to bandwidth increase

Figure 10: Response to bandwidth variation, $n = 8$

another n TCP flows join in to compete for the bottleneck link. Figure 10(a) shows the congestion window transition of the flow under study.

Consistent with [9], there's a tradeoff between responsiveness and smoothness. Standard TCP responds to congestion very quickly, at the expense of highly variable congestion window. IIAD made the smoothest transition, but the transition time is very long. AIMD and SIMD perform in between and achieve similar smoothness and responsiveness. Notice that now the average window size of each connection is greater than 10, so IIAD algorithm has better smoothness, which again agrees with our analysis in Section 2.3.

3.2.3 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 10(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 2.3.

3.2.4 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 standard TCP flow.[4] In our simulation, we let a standard short-lived (20 secs) TCP flow disturbs the transmission of a long flow controlled by the algorithm under study. The long-lived flow is transmitting packets over a 1.5Mbps link with RTT=50ms. The short-lived flow is started at time 60, terminated at time 80, then started again at time 110. Figure 11 plots the congestion window value of different control algorithms over time.

[4] 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.

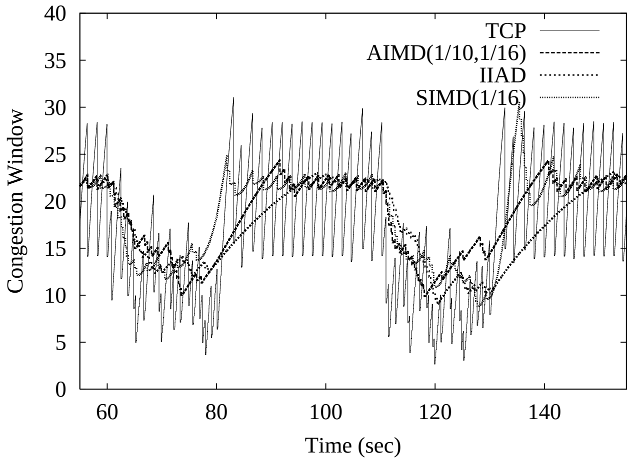


Figure 11: Impulse response to short-lived TCP flows.

We also prolonged our simulation to repeat these impulse disturbance pattern and measured the average aggressiveness and responsiveness according to our definition in Section 2.3 (here $m = 2$, $W_{aggr} = 11$, $W_{resp} = 22$) and reported these data in Table 3.[5]

Algorithm	**1/Aggressiveness (RTT)**		**1/Responsiveness (losses)**	
	simulation	analysis	simulation	analysis
TCP	13.5	14.7	1.6	1
AIMD	84.4	117.6	18.6	10.74
IIAD	157.6	181.5	22.9	16.5
SIMD	34.9	41.5	15.8	10.74

Table 3: Quantitative Measures

Again, 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 increase. AIMD has similar responsiveness as SIMD, but is less aggressive than SIMD when the short TCP flow terminates.

3.3 Convergence to Fairness and Efficiency

In this section, we assume 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 (cf. Section 2.2.2).

We use the topology shown in Figure 12 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), (s2-r2) stay to compete for the bottleneck link ($b2$,$b3$).[6]

[5]We choose the steady-state error to be 1 packet within the target windows $m\ W_{aggr}$ and W_{resp}/m, and take the two-way propagation delay as expected round trip time.

[6]We use packet size of 500 bytes in these experiments.

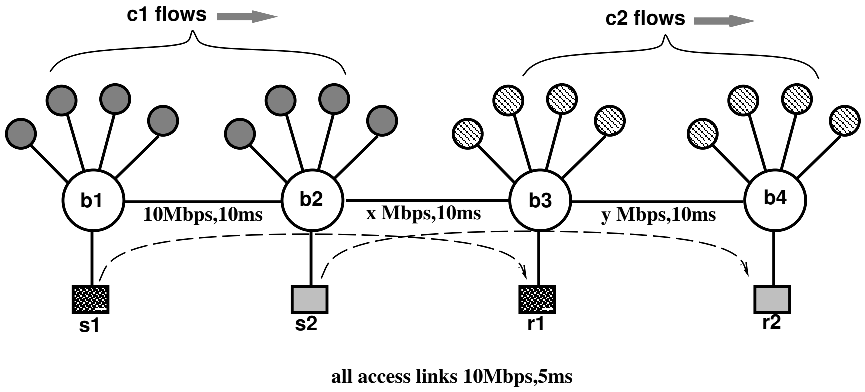
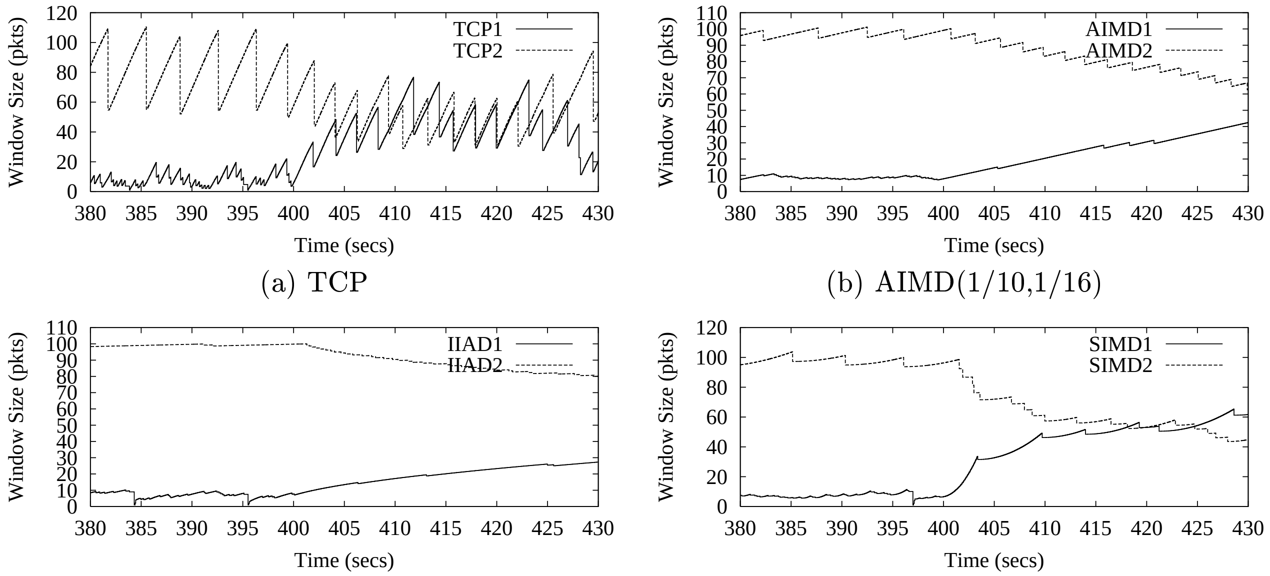


Figure 12: Simulation topology for convergence test

3.3.1 $W_1 < \frac{W}{2} < W_2$, $W_1 + W_2 = W$ (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 $c_1 = 15$, $c_2 = 0$, $x = 6$Mbps, $y = 6$Mbps, so $W \approx 110$, $W_1 \approx 7$, and $W_2 \approx 100$. Figure 13 plots the transient behavior of congestion windows of different protocols.



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

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

Figure 13: Two flows converge to fair share of bandwidth

It can be observed from the graph that standard TCP has the highest convergence rate, and since now the average window size is much higher than 10, 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 4 gives the convergence time to fairness (T_2). Here we use $\epsilon = 10$ packets (cf. Section 2.2.2). The theoretical value is also given in the table for comparison. The following observations can be made

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	6.11	99.65	68.0	88.7	8.75	13.79	55	43.7	5.8	5.96
AIMD	7.89	99.15	776.1	1217.1	12.72	31.02	349	342.3	18.6	18.3
IIAD	7.71	99.82	4232.3	6684.4	11.79	31.15	1284	1242.3	8.1	7.6
SIMD	6.55	96.32	218.5	852.3	10.19	33.21	90	85.1	13.6	12.3

Table 4: Quantitative measures on convergence time

from the table:

- 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 because our analysis is based on synchronized feedback assumption, routers do not differentiate among users when dropping packets. In contrast, in the simulation, we use RED, so users with higher window values would see more packet drops. Thus, our analysis give an upper bound on T_2. In other words, RED helps to enhance the convergence rate to fairness.
- SIMD benefits from RED much more than other schemes. This is because RED allows SIMD users with smaller windows to experience less packet losses, which makes them much more aggressive than users with large windows, and thus convergence to fairness is much faster.

3.3.2 $W_1 < W_2 < \frac{W}{2}$ (Convergence to Efficiency)

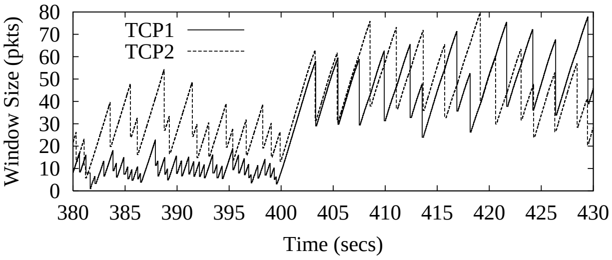
To create such scenario, we let $c_1 = 11$, $c_2 = 3$, $x = 6$Mbps, $y = 10$Mbps, so $W \approx 110$, $W_1 \approx 10$, and $W_2 \approx 30$. We can then study T_1, the convergence time to efficiency line of different control schemes. Figure 14 plots the transient behavior of congestion window of different protocols.

The advantage of 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 algorithm 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, SIMD algorithm allows the two flows to smoothly and quickly transit to the fair steady state, since the flow with smaller window is more aggressive than the one with larger window. IIAD takes a much longer time to converge due to its inherent weak aggressiveness.

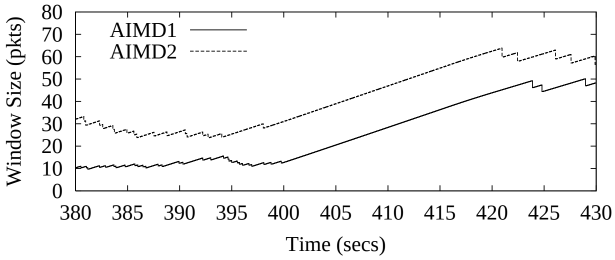
We also give convergence time to efficiency (T_1) in Table 4. Analytical results closely match the simulation results.

4 Related Work

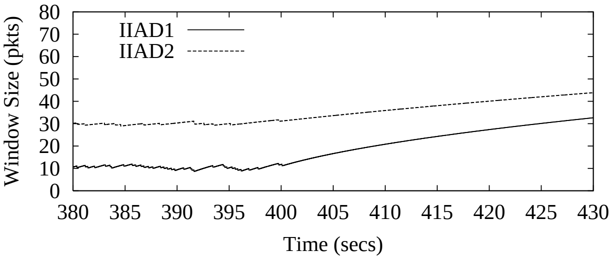
Under synchronized feedback assumption, Chiu and Jain [4] analyze AIMD control, thus provide a sound basis for Jacobson's TCP algorithm [13] and Ramakrishnan and Jain's DECbit scheme [21]. To provide smoother transmission rate than that given by TCP, several TCP-like window-based congestion control mechanisms have been proposed, including the general AIMD [9, 28] and TEAR [23]. These mechanisms use moderate window decrease parameter to reduce rate variability, meanwhile use matching window increase parameters to satisfy TCP-friendliness. There are tradeoffs between smoothness and reaction to changes in network conditions [9, 27].



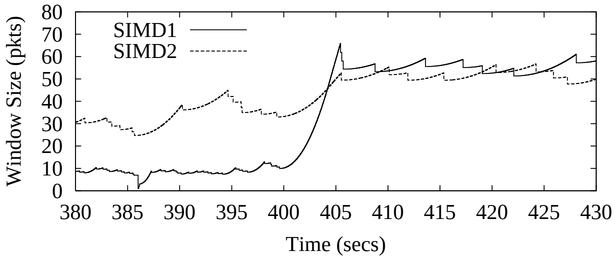
(a) TCP



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



(c) IIAD



(d) SIMD(1/16)

Figure 14: Two flows converge to fair share of bandwidth

Chiu and Jain also mentioned non-linear controls in [4]. 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 control rules. We provide much broader choices of TCP-friendly congestion control schemes. Our schemes differ from binomial algorithms in many respects, including the increase and decrease rules, convergence behaviors, and smoothness-aggressiveness tradeoffs. In particular, unlike memory-less controls [3, 12], our work proposes the first class of algorithms that exploit history information to increase convergence speed without sacrificing stability.

Another approach to provide smoother transmission rate is equation-based congestion controls [10, 20, 26], first proposed in [15]. In these schemes, the end systems measure the packet loss rate and round-trip time, and use the TCP-friendly equation [19] to compute the transmission rate. Two comparisons [9, 27] 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 have better short-term smoothness. Notably, equation-based schemes use more history information.

Applications can be adaptive to the congestion level of the network in a TCP-friendly way. Examples include RAP [22] and LDA [25]. 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 [24] for feedback information about 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 modeling of TCP congestion control [2, 7, 14, 17, 18, 19]. Ott *et al.* derived 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 [7] shows that the throughput of a connection is inversely proportional to its round-trip time. Lakshman *et al.* [14] studied the properties of TCP in a regime where the bandwidth-delay product is high and losses are random. In [17], Mathis *et al*. studied the relationship between TCP throughput and packet loss rate when TCP is in congestion avoidance stage and came up with the well-known TCP-friendly equation. Padhye *et al*. [19] 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 RTT and the square root of packet loss probability.

5 Conclusion

We proposed a spectrum of TCP-like window-based congestion control algorithms. These algorithms are TCP-friendly. They possess better convergence behaviors than AIMD and binomial algorithms. They possess different smoothness, aggressiveness, and responsiveness tradeoffs. Thus instances from our spectrum can be chosen as the transport schemes of various applications, for example, streaming applications on the Internet which are required to be TCP-friendly and need smoothness of transmission rates. We conducted extensive simulations using *ns* simulator. In particular, we presented simulation results of a special case called SIMD. 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 schemes in terms of TCP-friendliness, TCP-compatibility, convergence to fairness, and the tradeoffs among smoothness, aggressiveness, and responsiveness. In particular, unlike memory-less controls, our algorithms exploit history information to increase convergence speed without sacrificing stability. Future work includes comparing our control schemes to equation-based congestion control schemes.

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. TCP-friendly congestion control for real-time streaming applications. In *Proceedings of IEEE INFOCOM*, April 2001.

[4] 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.

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

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

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

[8] 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.

[9] 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.

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

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

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

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

[14] 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.

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

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

[17] 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.

[18] T. J. Ott, J. Kemperman, and M. Mathis. The stationary behavior of ideal TCP congestion avoidance. http://www.argreenhouse.com/papers/tjo, 1996.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, Jan 1996.

[25] D. Sisalem and H. Schulzrinne. The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. In *Proceedings of NOSSDAV*, July 1998.

[26] 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.

[27] 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.

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

A Equivalence of the Increase Rule and Increasing Function

We show here that the window increase rule in (3) and the window increasing function in (2) are equivalent. Using linear interpolation and continuous approximation, from (3), 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

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

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

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

B TCP-friendliness of Our Control Scheme

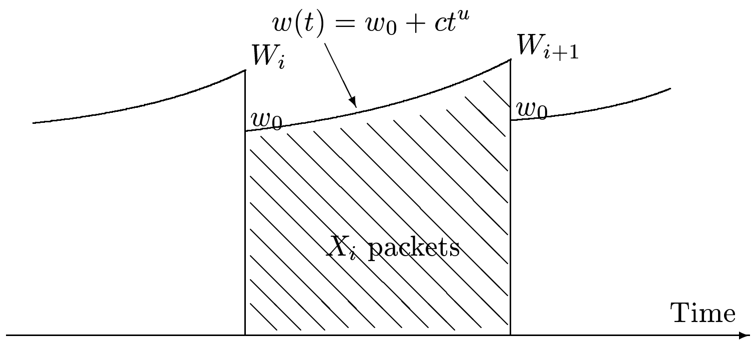


Figure 15: Window increases with time, and decreases on random packet losses.

This appendix shows that the definition of α in equation (4), or equivalently, the definition of c in equation (5) makes our control scheme TCP-friendly. We assume random losses and do not consider the effect of TCP's timeout mechanisms.

Consider many congestion epochs where the window increases and decreases alternatively in a stationary environment, as shown in Figure 15. Let W_i be the window size before the last decrease. In the i-th epoch, the window size is increased from $W_i - \beta W_i^l$ by I_i before the first packet loss happens. We have the following stochastic process:

$$W_{i+1} = (W_i - \beta W_i^l) + I_i. \tag{8}$$

Let $w_i(t)$ denote the window size at time t in the i-th epoch, $w_i(t) = w_0 + ct^u$, where c is defined as (5), $u = 1/(k+1)$, and $w_0 = W_i - \beta W_i^l$. Notice that w_{max} in c is defined by W_i.

B.1 Computing $E[I_i]$

We first compute the expected value of I_i. 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

$$P[X_i = j] = (1-p)^j p, \quad j = 0, 1, 2, ...$$

Let $t_i(X_i)$ denote the number of rounds between two consecutive loss events. A continuous expression of $t_i(X_i)$ can be obtained. However, a simpler approximation is X_i divided by the average window size in the i-th epoch, that is

$$t_i(X_i) \approx \frac{X_i}{E[w_i(t)]}.$$

This results in a window increase of size

$$I_i = c(\frac{X_i}{E[w_i(t)]})^u.$$

Consider all possible choices for X_i and compute the expected value of I_i, we have

$$\begin{aligned} E[I_i] &= \sum_{j=0}^{\infty} I_i P[X_i = j] \\ &= \sum_{j=0}^{\infty} c(\frac{j}{E[w_i(t)]})^u (1-p)^j p. \end{aligned}$$

If u is an integer, we can compute the above sum by computing the sum of power series of the form $n^u x^n, |x| < 1, n = 0, 1, 2, ...$ The result is $E[I_i] = \frac{c(u! + O(p))}{(pE[w_i(t)])^u}$. When $p << 1$, $E[I_i] \approx \frac{u!c}{(pE[w_i(t)])^u}$. If u is not an integer, it is hard, if not impossible to obtain the exact expression. However, we can use the continuous approximation $(1-p)^j p \approx pe^{-pj}$, and compute $E[I_i]$ as

$$\begin{aligned} E[I_i] &= \int_0^{\infty} c(\frac{x}{E[w_i(t)]})^u pe^{-px} dx \\ &= \frac{c\Gamma(u+1)}{(pE[w_i(t)])^u}, \end{aligned} \tag{9}$$

where $\Gamma(.)$ is the Gamma function. Notice that it is consistent when u is an integer since $\Gamma(u+1) = u!$.

B.2 Stationarity Assumptions

We assume that the stochastic process (8) is stationary. A consequence of this assumption is that the window size increase and decrease must be in equilibrium. It allow us to assume that when $W_i = E[W_i]$, the expected window increase is approximately equal to the decrease. That is, $E[I_i] \approx \beta W_i^l$. By (9),

$$\frac{c\Gamma(u+1)}{(pE[w_i(t)])^u} \approx \beta W_i^l. \tag{10}$$

Then we estimate $E[w_i(t)]$ as follows. We consider the average case when the window size increases by $E[I_i]$ in the epoch. Its duration is $(\frac{E[I_i]}{c})^{1/u}$. Then the mean window size is computed as:

$$\begin{aligned} E[w_i(t)] &= \frac{1}{(\frac{E[I_i]}{c})^{1/u}} \int_0^{(\frac{E[I_i]}{c})^{1/u}} (W_i - \beta W_i^l + ct^u) dt \\ &= (W_i - \beta W_i^l) + E[I_i]/(u+1). \end{aligned}$$

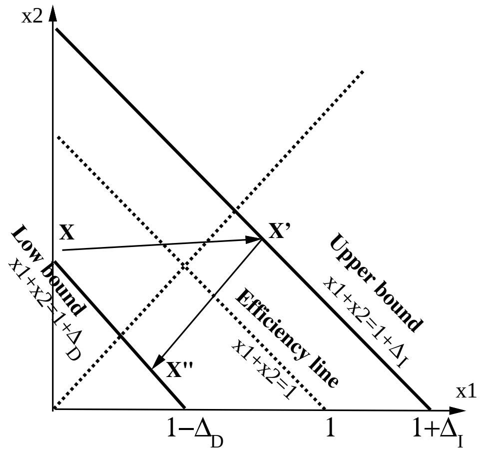


Figure 16: Bounded increases and decreases ensure convergence.

Using equation (10), we replace $E[I_i]$ with βW_i^l to obtain

$$E[w_i(t)] \approx W_i - \frac{u}{u+1}\beta W_i^l. \tag{11}$$

B.3 TCP-friendliness

When rate variation is small, $E[w_i(t)]$ is a close approximation to the time average of window size. In order for our scheme to be TCP-friendly, it must satisfy the (λ, p) relationship, $\lambda = 1/(\sqrt{2/3}R\sqrt{p})$, where R is RTT. Since $\lambda \approx E[w_i(t)]/R$, we need $\frac{1}{p} \approx \frac{2}{3}(E[w_i(t)])^2$. By equation (10), it is equivalent to

$$c\Gamma(u+1)(\frac{2}{3}E[w_i(t)])^u \approx \beta W_i^l. \tag{12}$$

It is straightforward to verify that equation (12) indeed holds by replacing c by equation (5), u by $1/(k+1)$, $E[w_i(t)]$ by equation (11), and noticing that $w_{max} = W_i$.

C 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 in order that our schemes converge when $0 \le l < 1$ and $\frac{1}{k+1} - 1 \le l < \frac{1}{k+1}$. Let Δ_I and Δ_D denote the maximum overshoot and undershoot, respectively. Therefore, the trajectory in steady state moves between $x_1 + x_2 \ge 1 - \Delta_D$ and $x_1 + x_2 \le 1 + \Delta_I$, as shown in Figure 16. Without loss of generality, we assume the initial $X(x_1, x_2)$ satisfies $1 - \Delta_D \le x_1 + x_2 < 1$ 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 proportional 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 have 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

$$1 + \Delta_I > 2(x_1 + x_2). \tag{13}$$

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,

$$\Delta_I + 3\Delta_D < 2. \tag{14}$$

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.

D Convergence Time

We use Figure 3(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.

D.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 $\alpha = 3\beta/(2-\beta)$[9], 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 increment rule is $w(t) = w_0 + ct^2$ for SIMD, the time it takes each user to come to this point is:

$$T_1^{SIMD} = \gamma\sqrt{\frac{W_1W_2(W - W_1 - W_2)}{W_1+W_2}}$$

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 $w(t) = \sqrt{2\alpha t} = \sqrt{3\beta t}$.[7] By solving the equation $w_1 + w_2 = \sqrt{3\beta T_1 + W_1^2} + \sqrt{3\beta T_1 + W_2^2} = W$, we get:

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

And the difference becomes:

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

D.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 Δ 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 relationship:

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

[7]Since $\frac{dw}{dt} = \alpha/w$, we get $w(t) = \sqrt{2\alpha t}$

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 size, 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:

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

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

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 Δ, we then have:

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

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 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:

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

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 $\beta^{SIMD} << 1$, $\beta^{IIAD} << W$, we have $L^{SIMD} \approx \sqrt{2}W/3$, $L^{IIAD} \approx W/3$. If we can 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:

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

E Smoothness, Aggressiveness, and Responsiveness of several algorithms

E.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.

(a) AIMD(α, β) Consider a congestion epoch in steady state where the window size increases from w_0, then decreases on the drop of a packet. Assume x packets have been acknowledged before the first drop. If the transmission rate is smooth, then the time of drop T is approximated by $\frac{x}{W}$. In this congestion epoch, $w(t) = w_0 + \alpha t, 0 < t < T$, where $\alpha = 3\beta/(2-\beta)$. We compute the mean window size, variance, and coefficient of variation as

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

$$E[(w - E[w])^2] \approx (\frac{3\beta}{2-\beta})^2 \frac{x^2}{12W^2}.$$

$$CoV \approx \frac{\sqrt{3}\beta}{4-2\beta}\frac{x}{W^2}.$$

Considering all possible choices of x, we compute the expected CoV as

$$E[CoV] \approx \int_0^\infty CoV pe^{-px} dx = \frac{0.289\beta}{1-\beta/2}.$$

(b) IIAD(α, β) Assume $\beta << W$. 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.

$$E[CoV] \approx \frac{0.289\beta}{W - \beta/2}.$$

(c) SIMD(β) The window size function is $w(t) = w_0 + ct^2$, where $c = \frac{9\beta}{8(1-\frac{2\beta}{3})^2 w_0}$. Following the same steps as for AIMD, we compute the mean window size, variance, and coefficient of variation as

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

$$Var = E[(w - E[w])^2] \approx (\frac{9\beta}{8(1-\frac{2\beta}{3})^2})^2 \frac{4x^4}{45W^6}.$$

$$CoV = \frac{\sqrt{Var}}{E[w]} \approx \frac{3\beta}{4\sqrt{5}(1-\frac{2\beta}{3})^2}\frac{x^2}{W^4}.$$

Consider all possible x, we compute the expected CoV as

$$E[CoV] \approx \int_0^\infty CoV pe^{-px} dx = \frac{0.298\beta}{(1-\frac{2\beta}{3})^2}.$$

E.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), $c_{SIMD} = \frac{9\beta}{8(1-2\beta/3)^2 W}$. 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 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 $\frac{(m^2-1)W^2}{3\beta}$.

E.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 β. Thus, they need $\log_{(1-\beta)} \frac{1}{m}$ loss events. IIAD decreases the window size by a constant β on detecting each loss event. Therefore, it needs $\frac{W-W/m}{\beta}$ loss events.

F 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 SIMD(β), we have the following equation:

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

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

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

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

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

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