TCP-friendly Congestion Control for Real-time Streaming Applications

Deepak Bansal and Hari Balakrishnan
M.I.T. Laboratory for Computer Science
Cambridge, MA 02139
Email: {bansal,hari}@lcs.mit.edu

Abstract

This paper introduces and analyzes a class of nonlinear congestion control algorithms called binomial algorithms*, motivated in part by the needs of streaming audio and video applications for which a drastic reduction in transmission rate upon congestion is problematic. Binomial algorithms generalize TCP-style additive-increase by increasing inversely proportional to a power k of the current window (for TCP, $k = 0$) ; they generalize TCP-style multiplicative-decrease by decreasing proportional to a power l of the current window (for TCP, $l = 1$). We show that there are an infinite number of deployable TCP-friendly binomial algorithms, all of which satisfy $k + l = 1$, and that all binomial algorithms converge to fairness under a synchronized-feedback assumption provided $k + l > 0; k, l \geq 0$. Our simulation results show that binomial algorithms interact well with TCP across a RED gateway. We focus on two particular algorithms, IIAD (inverse-increase/additive-decrease, $k = 1, l = 0$) and SQRT ($k = l = 0.5$), showing that they are well-suited to applications that do not react well to large TCP-style window reductions. We also find that TCP-friendliness in terms of the relationship between throughput and loss rate of an algorithm does not necessarily imply fairness relative to TCP performance, especially for drop-tail bottleneck gateways.*

1 Introduction

The stability of the Internet to date has in large part been due to the congestion control and avoidance algorithms [15] implemented in its dominant transport protocol, TCP [28, 34]. Based on the principle of additive-increase/multiplicative-decrease (AIMD) [6], a TCP connection probes for extra bandwidth by increasing its congestion window linearly with time, and on detecting congestion, reducing its window multiplicatively by a factor of two. Under certain assumptions of synchronized feedback, Chiu and Jain have shown that an AIMD control scheme converges to a stable and fair operating point [6], providing a sound basis for Jacobson's algorithms found in most current TCP implementations [1].

TCP is not well-suited for several emerging applications such as streaming and real-time audio and video because the reliability and ordering semantics it ensures increases end-to-end delays and delay variations. To be safe for deployment in the Internet, however, the protocols used by these applications must implement congestion control algorithms that are stable and interact well with TCP. Such protocols are called "TCP compatible" [3] or "TCP fair". They ensure that the TCP connections using AIMD get their fair allocation of bandwidth in the presence of these protocols and vice versa. One notion that has been proposed to capture "TCP compatibility" is "TCP-friendliness". It is well known that the throughput λ of a flow with TCP's AIMD congestion control (increase factor $\alpha = 1$ packet, decrease factor $\beta = 1/2$) is related to its loss rate p as $\lambda \propto S/(R\sqrt{p})$, where S is the packet size [19, 25, 10, 26]. An algorithm is TCP-friendly [20] if its throughput $\lambda \propto S/(R\sqrt{p})$ with the same constant of proportionality as for a TCP connection with the same packet size and round-trip time.

In this paper, we present and evaluate a new class of nonlinear congestion control algorithms for Internet transport protocols and applications. Our work is motivated by two important goals. First, we seek to develop and analyze a family of algorithms for applications such as Internet audio and video that do not react well to the large "factor-of-two" rate reductions that a TCP-style multiplicative-decrease entails, because of the drastic degradations in user-perceived quality that result. Second, we seek to achieve a deeper understanding of TCP-compatible congestion control by generalizing the class of linear control algorithms, and understanding how a TCP-friendly algorithm competes with TCP for bottleneck resources.

An AIMD control algorithm may be expressed as:

$$\begin{aligned} &\text{I: } w_{t+R} \leftarrow w_t + \alpha; \alpha > 0 \\ &\text{D: } w_{t+\delta t} \leftarrow (1-\beta) w_t; 0 < \beta < 1, \end{aligned} \tag{1}$$

where I refers to the increase in window as a result of receipt of one window of acknowledgements in a RTT and D refers to the decrease in window on detection of a loss by the sender, w_t the window size at time t, R the round-trip time of the flow, and α and β are constants. We have assumed a linear increase in window in the RTT.

To better understand the notions of TCP-friendliness and the trade-offs between the increase and decrease rules, we generalize the AIMD rules in the following way:

$$\begin{aligned} &\text{I: } w_{t+R} \leftarrow w_t + \alpha / w_t^k; \alpha > 0 \\ &\text{D: } w_{t+\delta t} \leftarrow w_t - \beta w_t^l; 0 < \beta < 1 \end{aligned} \tag{2}$$

These rules generalize the class of linear controls. For $k = 0$, $l = 1$, we get AIMD; for $k = -1, l = 1$, we get MIMD (multiplicative increase/multiplicative decrease used by *slow start* in TCP [15]); for $k = -1, l = 0$, we get MIAD; and for $k = 0, l = 0$ we get AIAD, thereby covering the class of all linear controls.

We call this family of algorithms *binomial* congestion control algorithms, because their control expressions involve the addition of two algebraic terms with different exponents. They are interesting because of their simplicity, and because they possess the property that any $l < 1$ has a decrease that is in general *less than* a multiplicative decrease, a desirable property for streaming and real-time audio and video. If there exist values of k and l (other than $k = 0, l = 1$) for which binomial algorithms are TCP-friendly, then it provides a spectrum of potentially safe congestion management mechanisms that are usable by Internet applications which do not react well to large and drastic rate reductions. It should be noted that varying α and β also help in reducing the oscillations. However, they still keep the reduction multiplicative and as a result, the same value of α and β cannot be used across wide range of bandwidth*delay values by an application that desires an absolute bound on the inherent oscillations due to congestion control algorithm. For that purpose (for example, if the layers in layered media are additively spaced), α and β have to be made functions of current window values which is what binomial algorithms help achieve.

Our major finding is that TCP-friendly binomial congestion control schemes do exist. Based on the analysis and simulation, we present the following findings:

- **The λ-p relationship.** For the binomial family of controls, $\lambda \propto 1/p^{\frac{1}{k+l+1}}$. In particular, the linear control protocols MIMD and AIAD have $\lambda \propto 1/p$, which is significantly more aggressive than the AIMD TCP-compatible relationship, while MIAD is unstable.

- **The $k + l$ rule.** A binomial algorithm is TCP-friendly if and only if $k + l = 1$ and $l \leq 1$ for suitable α and β. This implies that there is a wide range of TCP-friendly binomial controls parametrized by k and l, and applications can choose from this family depending on their needs and the level of rate degradation they can sustain. Furthermore, we show that under a synchronous feedback assumption, all the binomial control protocols converge to fairness as long as $k \geq 0$, $l \geq 0$ and $k + l > 0$. In particular, all the TCP-friendly binomial algorithms converge to fair allocations.

- **IIAD and SQRT control.** Of this family, we evaluate two interesting TCP-friendly algorithms in the (k, l) space: $(k = 1, l = 0)$ and $(k = 1/2, l = 1/2)$. We call the first IIAD (inverse-increase/additive decrease) because its increase rule is inversely proportional to the current window, and the second SQRT because both its increase is inversely proportional and decrease proportional to the square-root of the current window. Our simulations show that both IIAD and SQRT interact well with TCP AIMD across a wide range of network conditions over a RED bottleneck gateway.

- **TCP-friendliness vs. TCP compatibility.** Over a wide range of parameters, we discover that TCP-friendliness does not necessarily imply TCP-compatibility. The unfairness stems from the buffer management algorithms implemented at a congested gateway and how buffers are sized at a drop-tail (FIFO) gateway. Fortunately, an active queue management scheme like Random Early Drop (RED) at the bottleneck link alleviates this unfairness problem by explicitly equalizing packet loss rates across flows. Hence, while binomial algorithms are TCP-friendly (because they satisfy (λ, p) relationship, they become TCP-compatible in the presence of RED gateways.

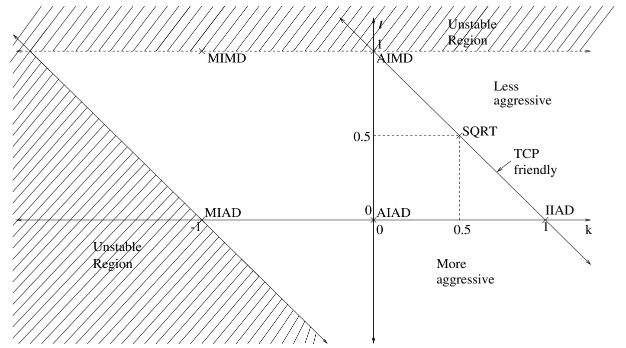

Figure 1. The (k, l) space of nonlinear controls from our family, with the $k + l = 1$ line showing the set of TCP-compatible controls.

Figure 1 summarizes the qualitative features of a binomial algorithms in the (k, l) space, including the points where it corresponds to the four linear controls, the line segment where it is TCP-friendly, and the regions where it is more and less aggressive than TCP AIMD. An interesting observation that follows from this figure and our analysis is that of all the TCP-friendly binomial algorithms $(k + l = 1, l \leq 1)$, AIMD is most aggressive in probing for available bandwidth. In this sense, AIMD is the most efficient and best suited binomial algorithm for bulk data transfer applications that can tolerate large reductions in available capacity upon encountering congestion. (Note that we assume that window size is always greater than 1, so $1/W^k < 1$). This observation shows the suitability of binomial algorithms as a good theoretical framework for evaluating additive increase/multiplicative decrease algorithms.

The rest of this paper is organized as follows. In Section 2, we discuss and analyze the properties of binomial algorithms. In Section 3, we delve into the IIAD and SQRT controls, presenting several simulation results with RED gateways and evaluating fairness with competing TCP connections. In Section 4, we discuss the interactions between binomial algorithms and TCP in the presence of drop-tail gateways. We describe the performance results of our implementation of SQRT algorithm for an Internet audio application in Section 5. We compare our work to past research on congestion management in Section 7 and conclude in Section 8.

2 Binomial congestion control algorithms

In this section, we present and analyze the properties of binomial congestion control algorithms. We start by providing some intuition about the sample paths traversed by the congestion window in a binomial algorithm, and showing that it converges to fairness under simplified conditions of synchronized feedback to sources. The intuition section makes simplifying assumptions but later, we corroborate our results by deriving an analytic formula that relates the throughput of a binomial algorithm to the loss rate it observes. We then, use this formula to obtain the conditions under which a binomial algorithm is TCP-friendly.

2.1 Intuition

We use the technique of Chiu and Jain and represent the two-source case as a "phase plot," where the axes correspond to the current window sizes, x_i, of each source (for convenience, we normalize each x_i to a number between 0 and 1, so it represents the fraction of the total window size aggregated across all competing sources). As the system evolves with time, the two sources adjust their windows according to the control equations, leading to a sample path in this phase space. The key to understanding binomial controls is to realize how these paths move in the phase space. To start with, we summarize how linear controls behave [6]:

1. Additive-increase/decrease: Moves parallel to the 45^o-line. Additive-increase improves fairness (in the sense of Jain's fairness index[1]), additive-decrease reduces it.

2. Multiplicative-increase/decrease: Moves along the line connecting (x_1, x_2) to the origin. Fairness is unchanged.

Because binomial algorithms are non linear, their evolution in the phase plot is not always along straight line segments. Figure 2 shows a portion of one such sample path highlighting the increase and decrease parts. For all values of $k > 0$, the increase in x_1 and x_2 are not equal—the smaller of the two values increases *more* than the larger one. It is clear that this leads to a fairer allocation than if both sources did additive-increase by the same constant amount. On the other hand, values of $l < 1$ in the decrease phase *worsen* fairness. However, binomial algorithms still convergence to fairness as we show in Section 2.2.

The parameters k and l represent the aggressiveness of probing and conservativeness of congestion response of a binomial control algorithm. A small value for k implies that the algorithm is more aggressive in probing for additional bandwidth, while a large value of l implies that the algorithm displays large window reductions on encountering congestion. Thus, it would seem that there is a trade-off between k and l in order for for a binomial protocol to achieve a certain throughput at some loss rate.

Indeed, in Section 2.3, we show that at any loss rate, the throughput depends on the sum of the two exponents, $k + l$. As a corollary, we find that a binomial algorithm is TCP-friendly if and only if $k + l = 1$ and $l \leq 1$. We call this the *$k + l$ rule,* which represents a fundamental tradeoff between probing aggressiveness and the responsiveness of window reduction. Figure 1 shows the features of the (k, l) space. We consider schemes for which $l > 1$ as unstable (Figure 1), since there always exists a window size w_t (assuming $w_t > 1$) above which $w_t < \beta w_t^l$ for any constant β. It can be made stable by having it not cut down window by an amount more than the current window but that involves modifying the basic increase decrease rules and thus, we consider them unstable. Similarly, schemes for which $k + l \leq -1$ are also unstable because as $k + l$ rule will show, λ does not decrease with increasing p in this realm. As a result, the window for a connection will keep on increasing (or remain same) as loss rate is increased.

2.2 Convergence to fairness

In this section, we show that a network with two sources implementing the same binomial control algorithm with $k, l \geq 0$ converge to a fair and efficient operating point ($x_1 = x_2 = 1/2$), provided that $k + l > 0$. The argument is easily extended to a network of $n > 2$ sources by considering them pairwise. We assume that the network provides synchronized feedback about congestion to all the sources[2]. We do not claim that this models the reality of Internet congestion well, but this analysis provides good insight into the results that follow.

Without loss of generality, suppose $x_1 < x_2$, which corresponds to points above the $x_2 = x_1$ equi-fairness line in Figure 2 (an analogous argument can be made when $x_2 < x_1$). First, consider the left-most picture that shows how a window increase evolves. When $k = 0$, the increase is additive, parallel to the 45^o-line (along line AB). When $k > 0$, the increase curve lies below the $k = 0$ line since the amount of increase in x_1 is larger than the corresponding increase in x_2 (note $x_1 < x_2$). Therefore, it intersects the maximum-utilization line $x_1 + x_2 = 1$ at a point C, to the *right* of where the $k = 0$ line intersects it. Such an increase improves efficiency, since $x_1 + x_2$ increases, and moves towards a fairer allocation (i.e., towards the intersection of the equi-fairness and maximum-utilization lines).

Now, consider a window reduction. Observe that when $l = 0$ (additive decrease), the window reduction occurs along the 45^o line (along line DE), worsening fairness. When $l = 1$, the decrease is multiplicative and moves along the line to the origin without altering fairness. For $0 < l < 1$, the window reduction occurs along a curve with

[1] For a network with n connections each with a share x_i of a resource, the fairness index $f = (\sum x_i)^2/(n \sum x_i^2)$ [16].

[2] This is the same network model as in Chiu and Jain's work [6].

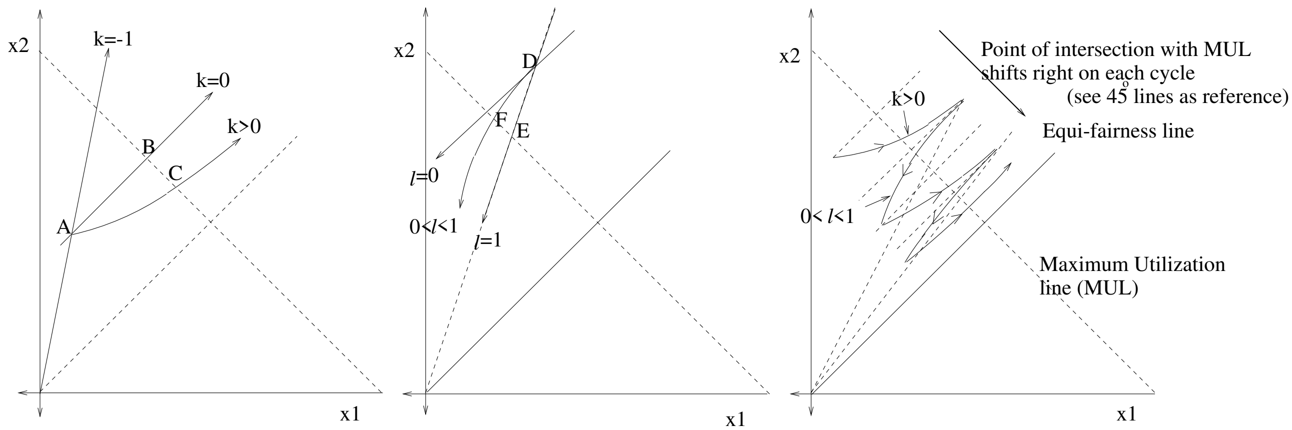

Figure 2. Sample path showing the convergence to fairness for an inverse increase proportional decrease algorithm.

shape as shown in the middle picture of Figure 2; this curve is in-between the previous two lines ($l = 0$ and $l = 1$ lines) and causes the system to evolve to an under-utilized region of the curve where $x_1 + x_2 < 1$. This curve lies strictly below the $l = 0$ line because the tangent at each point has a slope = $x_2^l / x_1^l > 1$ when $x_2 > x_1$. Therefore, it intersects the maximum-utilization line at a point F which is closer to the fair allocation point relative to the previous intersection of the sample path with that line.

The key to the convergence argument is to observe that the successive points of intersection of a binomial curve with the maximum-utilization line $x_1 + x_2 = 1$ always progress toward the fair allocation point. When $x_2 > x_1$, this continually moves downwards, when $x_2 < x_1$, it continually moves upwards towards the $x_2 = x_1$ point. Once $x_1 = x_2$, a binomial algorithm behaves exactly like a linear algorithm, moving on the $x_1 = x_2$ equi-fairness line.

It is easy to see that all we require in the above argument is for at least one of k and l to be larger than zero, since the sample path needs to move to the right at some stage. When $k = l = 0$, the algorithm is the linear additive-increase/additive-decrease scheme, which does not converge. The window evolution here remains on the 45^o-line passing through any point (x_1, x_2), without moving toward the fair allocation point.

This proof is valid under the synchronized feedback assumption and shows that a network in which all sources implement the same binomial control algorithm converges to a fair operating point. We note that it does not address the case of different binomial algorithms coexisting in the same network.

2.3 Throughput

We now analyze the throughput of a binomial algorithm as a function of the loss rate it experiences. We start with the *steady-state* model studied for TCP by Lakshman and Madhow [18] and Floyd [10]. Using the increase rule of Equation 2, we get using a continuous fluid approximation

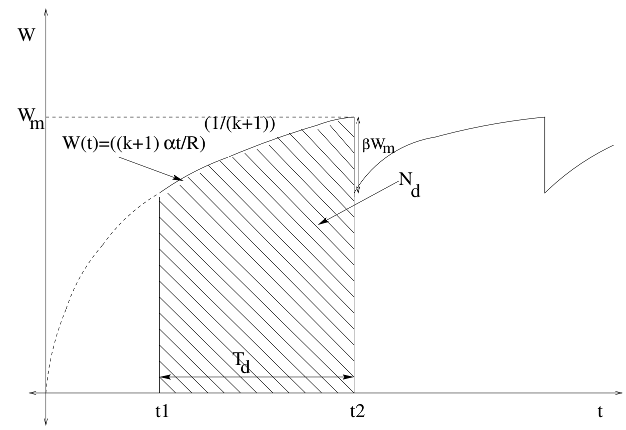

Figure 3. Functional form of window vs time curve.

and linear interpolation of the window between w_t and w_{t+R}:

$$\frac{dw}{dt} = \frac{\alpha}{w^k . R} \qquad \Rightarrow \frac{w^{k+1}}{k+1} = \frac{\alpha t}{R} + C \tag{3}$$

where C is an integration constant.

The functional form of this curve is shown in Figure 3. We are interested in two parameters marked in the figure: T_D, the time between two successive packet drops, and N_D, the number of packets between two successive drops. Both these are independent of "time-shifting" the curve along the horizontal (time) axis, which implies that one can arrange it such that a downward extrapolation of the curve passes through the origin. That is, without loss of generality and with no change to T_D and N_D, one can set $C = 0$.

Let W_m be the maximum value of the window w_t at time t_2 (Figure 3), at which a packet drop occurs signifying congestion to the sender. Then, one can write expressions for

T_D and N_D as follows:

$$T_D = t_2 - t_1$$

Substituting $w_{t_2} = W_m$ and $w_{t_1} = W_m - \beta W_m^l$ in Equation 3, we get

$$\begin{aligned} T_D &= \frac{R}{\alpha(k+1)}[W_m^{k+1} - (W_m - \beta W_m^l)^{k+1}] \\ &= \frac{RW_m^{k+1}}{\alpha(k+1)}[1 - (1 - \beta W_m^{l-1})^{k+1}] \\ &= \frac{RW_m^{k+1}}{\alpha}\beta(W_m^{l-1} + O(W_m^{2l-2})) \\ &\approx \frac{\beta RW_m^{k+l}}{\alpha}(\text{when } l < 1 \text{ and } \beta << W_m^{1-l}) \quad (4) \end{aligned}$$

The leading term in T_D therefore varies as W_m^{k+l}, with the succeeding terms becoming increasingly insignificant.

N_D is the shaded area under the curve in Figure 3.

$$N_D = (k+1)^{\frac{1}{k+1}} \int_{t_1}^{t_2} \left[\frac{\alpha t}{R}\right]^{\frac{1}{k+1}} /R\, dt \quad (5)$$

Calculating the integral, we get:

$$\begin{aligned} N_D &= \frac{1}{(2+k)\alpha} W_m^{2+k}[1 - (1 - \beta W_m^{l-1})^{2+k}] \\ &\approx \frac{1}{(2+k)\alpha} W_m^{2+k}\beta(2+k)W_m^{l-1}(\text{leading term}) \\ &= \frac{\beta}{\alpha} W_m^{k+l+1} \quad (6) \end{aligned}$$

The average throughput (in packets per second), λ of a flow using binomial congestion control is the number of packets sent in each epoch between successive drops (N_D) divided by the duration between drops (T_D). The packet loss probability, $p = 1/N_D$. Writing λ and p in terms of W_m by substituting the expressions for N_D and T_D yields:

$$\lambda = (\frac{\alpha}{\beta})^{1/(k+l+1)} \frac{1}{Rp^{1/(k+l+1)}} \quad (7)$$

Thus, $\lambda \propto \frac{1}{p^{1/(k+l+1)}}$ for a protocol in this family. This implies that for such a protocol to be TCP-friendly, λ must vary as $\frac{1}{p^{1/2}}$, which implies that:

$$k + l = 1 \quad (8)$$

To first order, choosing α/β to be the same as for TCP would achieve similar performance. Note that in our analysis above we have assumed a linear interpolation of window between w_t and w_{t+R} i.e we assume an increase in window by one in a RTT for TCP rather than an increase by 1/w on receipt of each acknowledgement.

These results also hold for the *random-loss* model first analyzed by Ott *et al.* in the context of TCP [25]. Unlike in the steady-state model where losses happen periodically when the sender's window reaches W_m, losses in the random-loss model are modeled as Bernoulli trials where each packet is independently lost with probability p.

We use the stochastic approximation technique for TCP performance described by Wang and Schwartz [36]. We treat the window value after receiving an acknowledgment with sequence number t, w_t, as a stochastic process and calculate its average value in the steady state. If we run the algorithm for a long time, the resulting congestion probability (the number of window reductions over the number of packets sent) is p. Then, in the steady state, the random process w_t evolves as follows: given w_t, with probability $(1-p)$, the packet is not lost and the sender's window increases, so $w_{t+1} = w_t + \alpha/w_t^{k+1}$, whereas with probability p, the packet is lost, forcing the window to reduce giving $w_{t+1} = w_t - \beta w_t^l$.

Using this, we can calculate the average "drift" D in the window when $w_t = w$. $D(w) = (1-p)\alpha/w^{k+1} - p\beta w^l$. Assuming that the stochastic process w_t has a stationary distribution, w_t must have most of its probability around the region for $w = W_{steady}$ such that $D(W_{steady}) = 0$. Thus:

$$\frac{(1-p)\alpha}{W_{steady}^{k+1}} = p\beta W_{steady}^l \quad (9)$$

$$\Rightarrow W_{steady} \approx (\frac{\alpha}{\beta p})^{1/(k+l+1)} \text{ if } p << 1 \quad (10)$$

We emphasize that p is the per-packet loss probability. As shown in the steady-state analysis above, W_{steady} is a good approximation to the time average of the random process w_t. The result, that $\lambda \approx (\frac{\alpha}{\beta})^{1/(k+l+1)} \frac{1}{Rp^{1/(k+l+1)}} \propto \frac{1}{p^{1/(k+l+1)}}$ therefore follows for the random-loss model as well.

This relationship establishes the fact that for a given loss rate and identical conditions, TCP AIMD and a binomial algorithm satisfying $k + l$ rule can achieve the same throughput. Further, it shows that for a given loss rate, two binomial connections will achieve same throughput provided other conditions are same.

3 Simulation results

In this section, we present the results of our *ns-2* [24] simulations of binomial control algorithms. We start by investigating the interactions between connections running a TCP-friendly binomial control algorithm (i.e., one that satisfies the $k + l$ rule) and TCP, as a function of k, which determines how aggressive the window increase factor is. We then investigate the performance of two specific members of this family: IIAD (inverse-increase/additive-decrease; $k = 1, l = 0$) and SQRT ($k = l = 0.5$; this corresponds to an increase proportional to the square-root of the current window and a decrease proportional to it). We conclude this section by studying the performance of IIAD and SQRT in the presence of multiple bottlenecks.

Our single bottleneck simulations used the topology shown in Figure 4. It consists of n connections sharing a bottleneck link with total bandwidth equal to b; all connections have an identical round-trip propagation delay equal

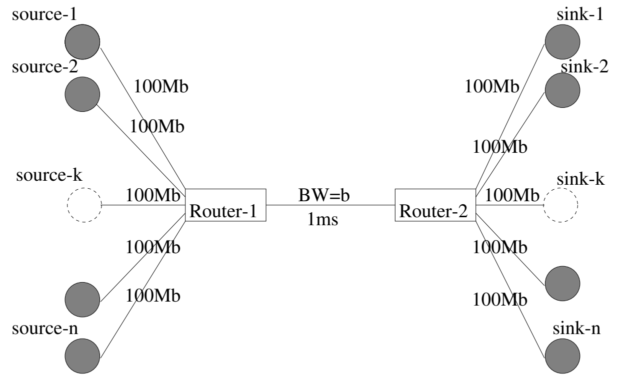

Figure 4. Simulation topology (delays of links for which no delay is specified are all equal such that the round-trip time for each connection = RTT ms).

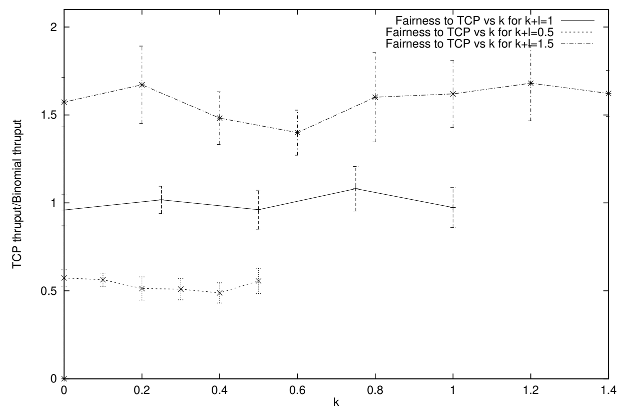

Figure 5. Ratio of the throughput of TCP AIMD to the throughput of a binomial algorithm, as a function of k. The algorithms that satisfy the $k + l$ rule are the ones for which this ratio is closest to unity. When $k + l = 0.5$, the binomial algorithm obtains much higher throughput relative to TCP and when $k + l = 1.5$, TCP obtains much higher throughput relative to binomial algorithm, as predicted by the analysis. The error bars show the 95% confidence interval of the throughput ratio. In these experiments, $b = 3$ Mbps and $RTT = 50$ ms.

to RTT (we change b and RTT in various simulations). We implemented the transport protocol by modifying the congestion avoidance algorithm used by TCP; we replaced AIMD with the binomial family. However, we did not modify the connection start-up or timeout routines; they continue to use slow-start and timeouts as before. Thus, the effect of slow start and timeouts on connection throughput is same as for a normal TCP connection. Each source always has data to send, modeled using ns's "FTP" application. In all our experiments, we simulated each topology and workload ten times and calculated both the average and sample standard deviation of the observed values. The figures and graphs display this information.

In this section, we present performance results using Floyd and Jacobson's Random Early Drop (RED) buffer management algorithm at the bottleneck gateway [12]. The maximum queue size Q at the bottleneck was set to $b \times RTT$, 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, and the connections used a packet size of 1KByte. Each connection was started at uniformly chosen random times in $[0, 2]$ seconds and throughput was calculated from $t = 10s$ to $t = 20s$ to give sufficient time for the connections to stabilize and to filter out startup transients. Later in the section, we also present results showing startup effects and impulse response of a binomial algorithm on TCP and vice versa.

3.1 TCP-compatibility

Our first set of results (Figure 5) show how binomial algorithms interact with each other and with TCP. To study the effect of k and l on TCP, we simulated two connections ($n = 2$), one TCP and the other a binomial algorithm parametrized by k. We show three sets of results corresponding to the three cases $k + l$ equal to, less than, and greater than 1. For these simple scenarios, these results validate the $k + l$ rule for TCP-friendliness, since the long-term throughput for the binomial algorithms for which $k + l = 1$ are close to that of TCP. These results also show that TCP-friendliness implies TCP-compatibility across RED gateways.

3.2 IIAD Algorithm

We now turn to IIAD, which is relatively less aggressive in the rate at which it probes for bandwidth ($k = 1$), but at the same time only reduces its window by a constant upon congestion ($l = 0$). We choose the values of α and β such that the theoretical throughput of IIAD is close to the throughput of TCP AIMD. There are an infinite number of values for α and β corresponding to this; we pick one pair, $\alpha = 1, \beta = 0.6$.

We compare the fairness of IIAD relative to another IIAD instance and to a TCP/Reno sharing the same bottleneck using the topology and workload in Figure 4. In these experiments, $n = 2$ and each connection was started at a random time in $[0, 2]$ seconds. Each individual experiment was conducted using a bottleneck bandwidth b of 1.5 Mbps and the round-trip time RTT was varied between 50 ms and 200 ms.

Figure 6 plots the throughput ratio for two IIAD connections on one curve and for one IIAD and one TCP connection on the other curve. These results show that IIAD is fair to both the IIAD and to TCP across a range of RTT values. However, the standard deviation of the results increases as RTT increases. This is because as RTT increases, each connection takes a greater amount of time to reach the optimum available bandwidth. Furthermore, when persistent losses occur leading to a timeout (recall that these experi-

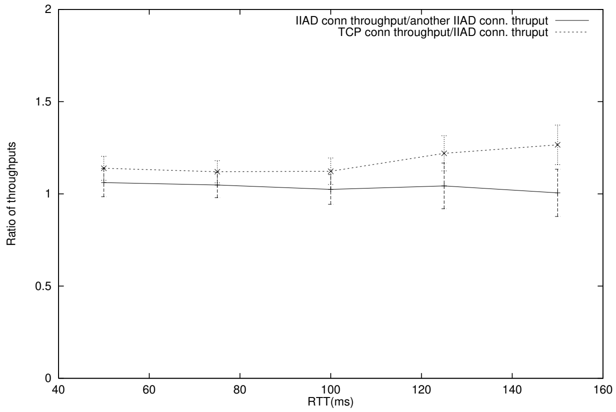

Figure 6. Ratio of throughputs of two connections sharing the bottleneck along with 95% confidence intervals ($n = 2, b = 1.5$ Mbps).

ments use a modified TCP source and sink), the congestion window shrinks to one packet and slow start occurs. The initial start-up effects and response to timeouts take longer to stabilize at higher delays, resulting in the correspondingly higher variance. The start-up effects are exacerbated for IIAD because it is less aggressive than AIMD, and responds relatively slower to any spare bandwidth. This is the reason that at higher delays, IIAD sometimes loses to TCP; the connections experience losses during slow start and IIAD takes longer to ramp to its share of bandwidth. Figure 6 shows that IIAD and TCP AIMD are fair to each other over long time scales. We observed the same results and behavior across a range of bottleneck bandwidths.

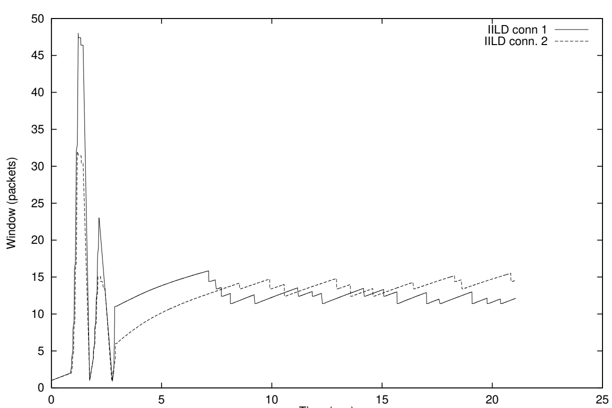

Figure 7. Congestion window evolution at the sender for two IIAD connections that start at random times in $[0, 2]$ seconds. In this experiment, $b = 1.5$ Mbps and $RTT = 50$ ms.

For one of these experiments, we look at the evolution of the congestion window. Figure 7 shows this for the two IIAD connections, showing the expected increase and decrease behavior as a function of time. For want of space, we do not show the sequence trace for these two connections here, but the two connections quickly achieve the same slopes signifying that they share bandwidth well with each other. We observed such good sharing across a range of b and RTT values.

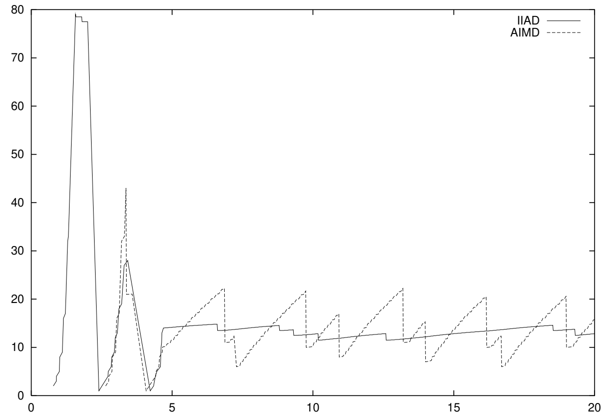

Figure 8. Congestion window variation for the IIAD and TCP Reno connections started at random times.

More interesting is the interaction between IIAD and TCP in terms of the evolution of their congestion windows (Figure 8). We observe that the window values quickly become similar, and even though the TCP connection started later, it was able to obtain its share of bandwidth without any difficulty.

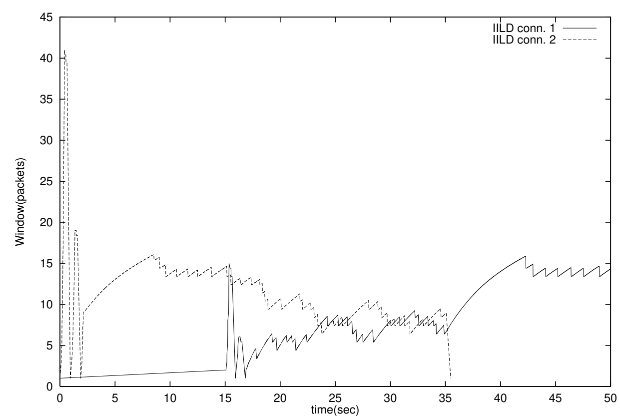

Figure 9. Slow start response of an IIAD flow to another long-running IIAD flow. In this experiment, $b = 1.5$ Mbps, $RTT = 50$ ms.

We now consider the impulse-response behavior of the binomial control algorithms. Our experiences with these experiments across several binomial algorithms have convinced us that slow start (or a similar mechanism) is an important component of any practical protocol to ensure that a connection converges relatively quickly, within a few RTTs to the fair value. We show an example of this in Figure 9, which shows how slow start enables a new IIAD connection to catch up and share bandwidth with another long-running IIAD connection.

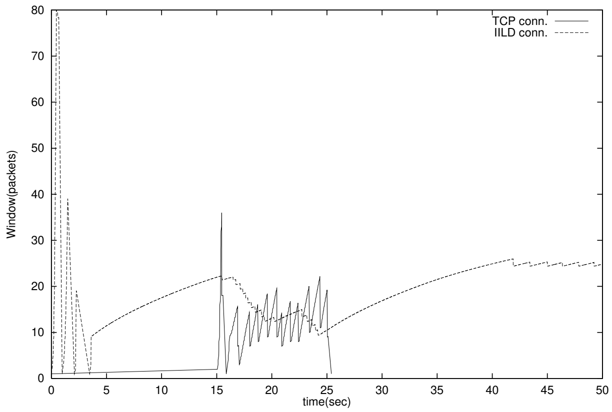

Figure 10. Response of a long-lived IIAD flow to a TCP impulse. In this experiment, $b = 1.5$ Mbps and $RTT = 50$ ms.

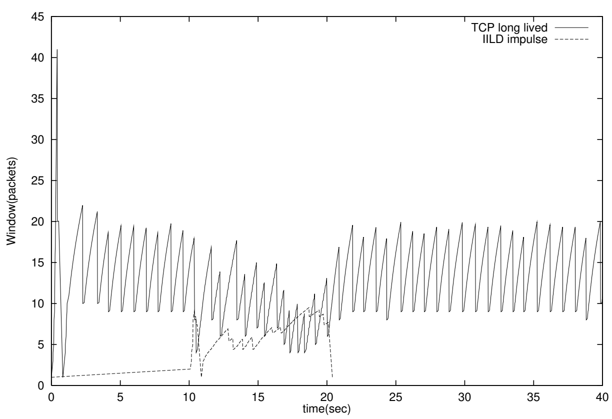

Figure 11. Response of a long-lived TCP flow to an IIAD impulse. In this experiment, $b = 1.5$ Mbps and $RTT = 50$ ms.

We observe the same behavior when a new TCP AIMD connection impulse encounters a long-running IIAD, as shown in Figure 10. The results of the converse experiment are shown in Figure 11, where an IIAD impulse meets a long-lived TCP AIMD. These figures show that IIAD and TCP converge to their fair bandwidth share, with the impulses using slow start.

These results hold when the number of connections is increased as well. Figure 12 shows the window variation for five of fifteen concurrent IIAD flows sharing the bottleneck link. At time $t = 20$ seconds, a TCP AIMD connection starts

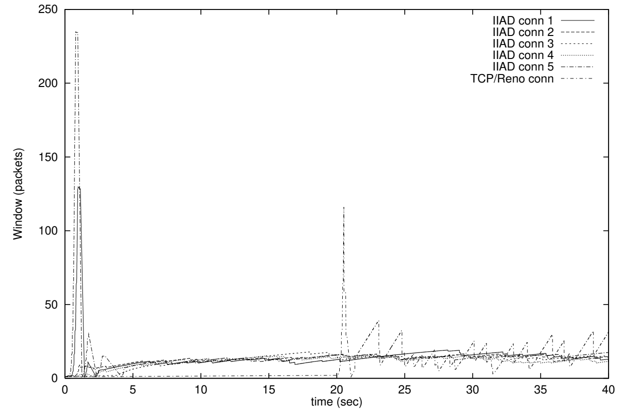

Figure 12. A TCP AIMD connection grabs its fair share in the presence of many long-lived IIAD flows ($n = 16, b = 9$ Mbps, $RTT = 50$ ms). For clarity, we only show the window sizes of five IIAD connections and the TCP connection.

(the impulse at $t = 20$), and is able to grab its share of bandwidth even in the presence of several other long-lived IIAD connections as can be seen from the TCP window evolution after about $t = 25$ seconds. Conversely, Figure 13 shows the window evolution of an IIAD flow that starts at time $t = 20$ seconds in the presence of five concurrent long-lived TCP connections (for clarity, we only plot the windows of two of the TCPs).

3.3 SQRT algorithm results

We now investigate the performance of the SQRT algorithm, which has $k = l = 0.5$.

Although we use $\alpha = 1$ and $\beta = 0.6$ in the reported experiments; we found that performance is relatively insensitive to small changes in β. We do not report the results of all the experiments we conducted with SQRT; in particular, we found that the long-term fairness of SQRT connections with each other and with TCP are high. We obtained results very similar to Figure 6 (the IIAD experiments), across a wide range of RTT and b values, showing that when the number of connections is small, SQRT connections share bandwidth fairly with each other and with TCP AIMD.

SQRT differs from IIAD in its aggressiveness in increasing its window and in reducing its window upon congestion. As a result, the details of its interaction with TCP AIMD are different from those of IIAD. This is shown in Figure 14, where a TCP impulse encounters a long-running SQRT connection at the bottleneck. We see that the two connections converge to their share of the bandwidth in fewer number of round-trip times than in the TCP-IIAD case (Figure 10).

Figure 15 plots the time evolution of the congestion window for concurrent SQRT and AIMD connections. As expected, the window variations for SQRT are larger than for IIAD (Figure 8), but its variations are markedly smaller than TCP. As with IIAD, this does not affect bandwidth sharing

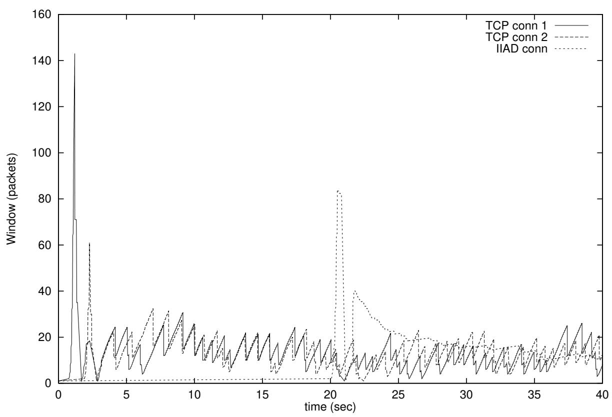

Figure 13. An IIAD flow grabs its fair allocation in the presence of many long-lived TCP flows ($n = 6, b = 9$ Mbps, $RTT = 50$ ms). For clarity, we only show the window sizes of two TCP connections and the IIAD connection.

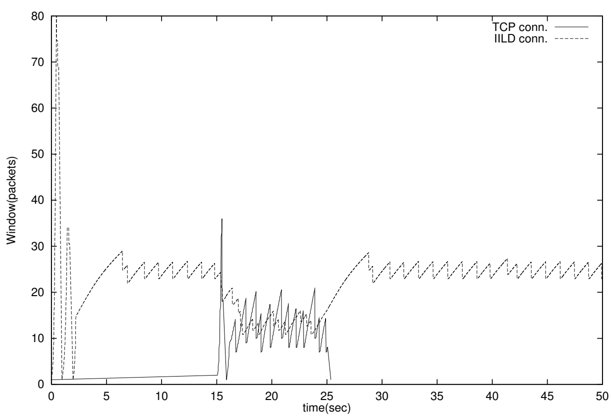

Figure 14. Response of a long-lived SQRT flow to a TCP impulse. In this experiment, $b = 3$ Mbps and $RTT = 50$ ms.

as can be seen from the window plot. We observed similar results for a range of RTT and b values, and when we scaled n to higher values.

Finally, we consider the sensitivity of our results to the value of β in the binomial algorithms by showing our results for SQRT case (we obtained similar results for IIAD as well). Figure 16 plots the effect of β on fairness to a TCP connection sharing the bottleneck. We observe little variation in the throughput ratio, but notice that fairness to TCP reduces at the extremes. When β is close to 0, TCP AIMD loses and when β is close to 1, SQRT loses. This is expected behavior based on the magnitude of SQRT window reduction upon congestion.

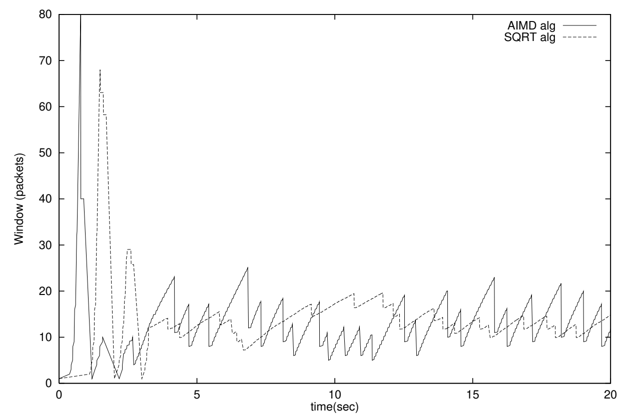

Figure 15. Graph showing window variation for SQRT and TCP AIMD connections sharing the bottleneck ($b = 3$ Mbps, $RTT = 50$ ms).

3.4 Multiple connections and multiple bottlenecks

This section investigates the impact of scale on binomial algorithms along two dimensions: (i) increasing the number of concurrent connections across a single bottleneck, and (ii) investigating performance across multiple bottleneck links.

To understand how several connections using different TCP-friendly binomial algorithms interact with each other, our first series of experiments looks at several concurrent connections running different algorithms sharing the bottleneck. The topology we use is same as in Figure 4 with $b = 50$ Mbps and $RTT = 50$ ms. We choose values of $k = \{0, 0.25, 0.5, 0.75, 1\}$ and $l = 1 - k$, and vary the total number of connections n. For each value of k, we set up $n/5$ connections, and start each connection at a random time in the interval $[0, 2]$ seconds. In Figure 17, we plot the mean value of Jain's fairness index (ten runs for each data point) along with 95% confidence intervals.

To study the impact of multiple bottlenecks and background traffic on the performance and fairness of binomial algorithms, we simulated the topology shown in Figure 18. The maximum number of HTTP connections for each HTTP source was set to five and all other parameters were set to the default values from *ns-2* for the HTTP and CBR sources and sinks. The window variation for the TCP AIMD and IIAD sources are shown in figure 19. As can be seen from this figure, the bottleneck bandwidth gets distributed fairly among these sources even in the presence of multiple bottlenecks. We observed the same behavior for other sources in this simulation and also when we replaced the IIAD source a SQRT source.

4 TCP friendliness vs TCP Compatibility

In this section, we study the interactions between binomial algorithms and TCP AIMD over a drop-tail bottleneck gate-

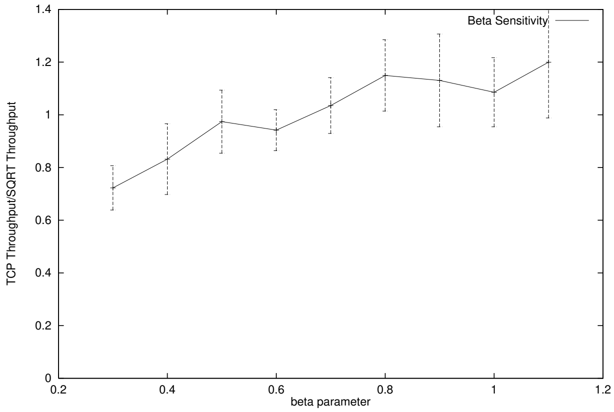

Figure 16. Plot showing fairness of SQRT to TCP/Reno (throughput ratio) along with the 95% confidence-interval for various values of β of the SQRT algorithm (b = 3Mbps, $RTT = 50$ ms).

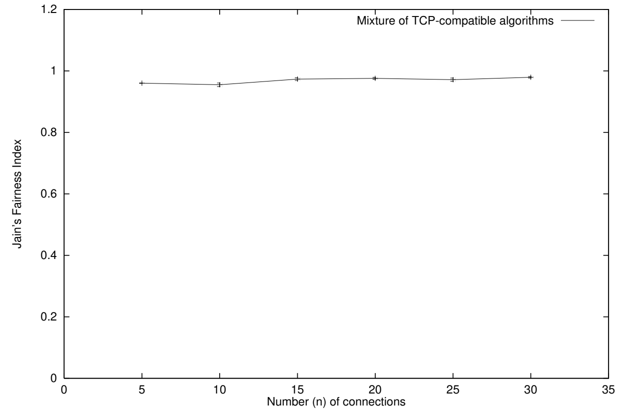

Figure 17. Plot showing Jain's Fairness Index as a function of the number of TCP-compatible connections sharing a bottleneck. In each experiment, the total number of connections was divided equally into five categories corresponding to $k = 0, 0.25, 0.5, 0.75, 1$. In these experiments, $b = 50$ Mbps, $RTT = 50$ ms. The (tiny) error-bars show 95% confidence intervals.

way, observing some surprising effects. Figure 20 shows the window variation and bottleneck buffer occupancy for two connections, one TCP AIMD and the other IIAD, sharing a drop-tail bottleneck gateway. We see that TCP starts losing out and its window keeps on decreasing until it starts to oscillate below its fair share because no buffers are available to it. On the other hand, IIAD starts grabbing more and more bandwidth. We observed similar behavior with other binomial algorithms as well.

At first, we found this result puzzling because the theory and the $k + l$ rule had predicted that as long as $k + l = 1$, the long-term throughput of a binomial algorithm would be equal to TCP AIMD. However, closer examination of the bottleneck buffer occupancy revealed the problem. In a congested network, the "steady state" of the bottleneck queue is close to full. IIAD is less aggressive than AIMD, and when it reduces its window, does not completely flush the queue. When a drop-tail gateway has been configured with a queue size of $b \times RTT$, it ensures that TCP-style "factor-of-two" multiplicative decrease brings the reducing connection's contribution to the bottleneck occupancy down to (or close to) 0. This allows other competing connections to ramp up and also ensures that sufficient buffers are available for the window to increase before another "factor-of-two" reduction happens. In contrast, a non-AIMD TCP-friendly binomial algorithm, by its very design, ensures that window reductions are not drastic. As a result, it ends up with more than its fair share of the bottleneck; and a window reduction does not flush all of its packets from the queue. In fact, the competing AIMD window oscillates as if it sees buffers equal to the additive decrease term (the amount of buffer freed by IIAD on a reduction) of the IIAD algorithm. The result is that drop rates observed by the two flows competing at a drop-tail bottleneck are not equal. This argument also shows how buffer provisioning is intimately tied to the window adjustment algorithm of the end-systems for drop-tail gateways.

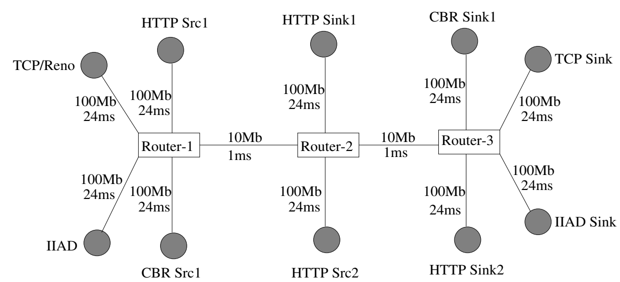

Figure 18. Topology with multiple bottlenecks and background traffic.

In contrast, RED gateways are designed to accommodate bursts and maintain small average queue sizes by providing early congestion indications. They seem ideally suited to binomial algorithms because they do not tie buffer sizing closely to the precise details of window adjustment of the end-points. Instead they vary the drop rate as a function of queue size making all flows see the same drop rate. This is yet another among the many other compelling reasons for the Internet infrastructure to move to a more active queue management scheme like RED.

We do not view the TCP-unfairness of the binomial algorithms across drop-tail gateways as a deployment problem: first, the binomial algorithms obtain better throughput than TCP AIMD with drop-tail gateways, which augurs well for applications using them (and also provide an incentive for Internet Service Providers to move to better queue management schemes)! Second, any scalable scheme for detecting flows using more than their share of bandwidth would likely

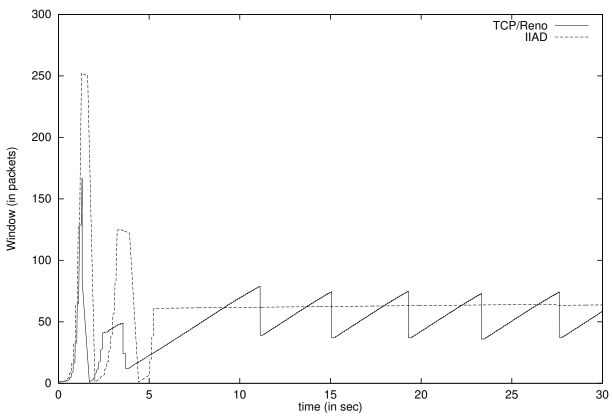

Figure 19. Window variation vs. time for the topology with multiple bottlenecks.

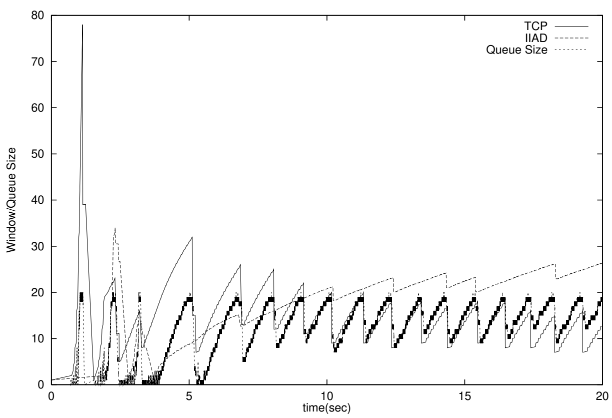

Figure 20. Plot showing window/queue size variation for TCP/Reno and SQRT algorithms sharing the bottleneck with drop-tail gateways($b = 3$ Mbps, $RTT = 50$ ms).

use an active queue management scheme and not a drop-tail gateway, which would ensure that true fairness to TCP is achieved. We emphasize that the adverse interactions of the binomial algorithms with TCP are primarily a consequence of the ill effects of drop-tail queue management.

An important consequence of the above findings and arguments is that TCP-friendliness does not necessarily imply TCP-compatibility since the theory assumes that drop rates for competing flows are equal at a gateway. We therefore conclude that justifying a congestion control algorithm as safe for deployment on the Internet purely on the basis of the TCP-friendly equation is dangerous. While our experience indicates that this is reasonable with certain types of queue management (such as RED), it is incorrect when congestion occurs at a drop-tail gateway. We believe that deriving a set of sufficient conditions for TCP-fair congestion control in drop-tail networks requires further research.

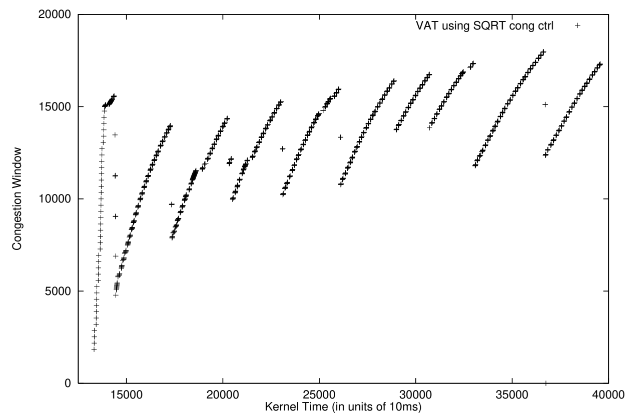

Figure 21. Window variation for a *vat* session using SQRT congestion control with a bottleneck configured using Dummynet ($b = 50$ Kbps,$RTT = 900$ ms).

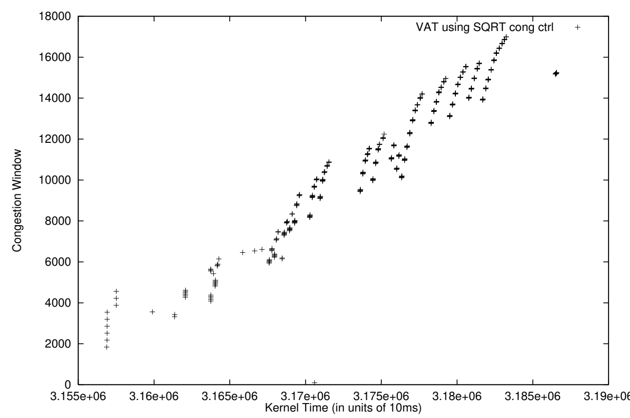

Figure 22. Window variation for a *vat* session using SQRT congestion control across an Internet path.

5 Implementation

We have implemented the SQRT congestion control algorithm in the Linux Kernel (version 2.2.9) to provide congestion controlled UDP sockets. We experimented with the Internet audio conferencing tool, *vat* in unicast mode. Figure 21 shows the congestion window variation for a transfer as a function of 10ms time intervals for an audio session between two Linux machines. These machines were on the same LAN but had a pipe of bandwidth 50Kbit/s and RTT 900ms between them, configured using Dummynet [8]. The figure shows the effectiveness of SQRT congestion control in alleviating the large TCP-style "factor-of-two" reductions. The magnitude of oscillations are smaller than what AIMD would observe.

Figure 22 shows the congestion window variation for a *vat* transfer between two Linux machines, one at MIT and the other at University of California, Berkeley. Again, the magnitude of oscillations are much smaller than with AIMD. The window keeps increasing because the bandwidth available between these two machines was much higher than the 64Kbps, rate at which *vat* samples audio data. This graph also demonstrates the working of SQRT across the Internet, since the occasional reductions are not drastic.

We note that our algorithms can be incorporated in the Congestion Manager (CM) architecture to provide applications tunable congestion control [2]. The CM exports an API that allows applications to learn about and adapt to the network conditions; this API can easily be extended to allow an application to pick the parameter (k) of a binomial algorithm, with the CM using the $k + l$ rule to ensure that the choice is TCP-friendly. An audio or video application can then, easily use one of the TCP-friendly binomial algorithms over a UDP-based transport protocol. While it is unlikely that elastic applications that run well over TCP today will use a non-AIMD binomial algorithm, an implementor can change the TCP sender with little implementation effort.

In fact, a protocol may switch between different TCP-friendly binomial algorithms in real-time, adapting to changing network conditions. In particular, it can probe more aggressively if it observes no congestion for a while, and probe less aggressively otherwise.

6 Deployment of Binomial Controls in the Internet

An issue of concern for wide scale deployment of IIAD algorithms in the Internet may be that their relatively mild reduction in window on experiencing congestion may affect Internet stability. However, we believe that the primary threat to the Internet stability comes not from the flows using some form of TCP-compatible congestion control but from flows that do not use any congestion control at all. Moreover, prevention of congestion collapse does not require that flows reduce their sending rate by half in response to a single congestion indication. The binomial algorithms, being window based and TCP-friendly, cannot cause congestion collapse simply by the fact that they send out data only in response to receipt of successful acknowledgements (except during timeouts) at the receiver. Moreover, in response to even a single loss, these algorithms reduce their window and hence alleviate congestion by stopping data transmission till the network acknowledges (or delivers) sufficient data (equal to cutdown) after that loss.

Another issue may be that α and β values of TCP's AIMD can be adjusted to provide a more stable congestion control as against using binomial controls. As mentioned earlier, adjusting α and β can only provide relative and not absolute (or fixed) bounds on oscillations due to congestion control. Moreover, binomial controls provide a framework for studying window based congestion control for multimedia applications of which TCP AIMD is one possibility.

7 Related work

There has been significant work over the past fifteen years in the area of network congestion management, especially on end-system mechanisms.

Chiu and Jain analyze the performance of linear controls, deriving the conditions for efficient convergence to fairness under a synchronized-feedback assumption [6]. They allude to non-linear controls and briefly discuss some of their properties, concluding that they seem more complex and inferior to linear controls. To our knowledge, a thorough analysis and evaluation of any family of nonlinear congestion control algorithms has not been done until now. We also focus on TCP-compatibility, recognizing the large deployed base of TCP AIMD algorithms.

Much of the classical literature on end-system congestion management was motivated primarily by reliable unicast transport, and included both window- and rate-based approaches. In addition to Jacobson's TCP algorithms [15], prominent examples and studies include Ramakrishnan and Jain's DECBit scheme that was linear with a multiplicative-decrease factor (β) of $7/8$, rate-based control in the Versatile Message Transport Protocol (VMTP) [5], Clark *et al.*'s NETBLT [7], Keshav's packet-pair approach (which requires flow isolation at congested routers) [17], and Faber *et al.*'s Dynamic Time Windows [9].

Subsequent to the development and deployment of TCP's algorithms in the Internet, Wang and Crowcroft proposed "Tri-S" (slow start and search) [37] to improve TCP's slow start. Brakmo and Peterson proposed TCP Vegas [4], which attempted to improve TCP's congestion avoidance and loss recovery. More recently, a number of enhancements have been proposed to TCP congestion control, including the persistent fast recovery of Newreno [14], selective acknowledgments (SACK) [22], and forward acknowledgments (FACK) [21]. RFC 2581 describes the recommended algorithms for proper congestion control in TCP [1]. [13] has formulated congestion control as a global optimization problem and has proposed a class of congestion control policies based on rewards and costs.

While these TCP enhancements are interesting, significant recent trends in Internet applications and traffic have led to a renewed interest in in end-system congestion control protocols. Several emerging applications including unicast audio and video are best transported over an application-level protocol running over UDP, rather than over TCP because they do not require a fully-reliable in-order delivery abstraction. Using TCP leads to a large delay variation caused by retransmissions, and perceptual quality shows sudden degradations in the face of a TCP-style window reduction for these applications.

Much recent work has focused on congestion control for adaptive applications. Rejaie *et al.*'s Rate Adaptation Protocol (RAP) uses AIMD, relying on frequent receiver acknowledgments to adjust the sender's rate [31]. They also propose a quality adaptation algorithm for discretely-layered streams at the receiver to handle the rate variations triggered by AIMD [30]. In the context of multicast, McCanne *et al.*'s receiver-driven layered multicast (RLM) incorporates a probing and rate reduction mechanism for layered video [23]. Sisalem and Schulzrinne's Loss-Delay-based Adjustment (LDA) scheme uses an AIMD rate control at the sender, using RTCP [32] for feedback [33]. Schemes like RAP and LDA can use a binomial algorithm (e.g., IIAD or SQRT) to avoid drastic rate reductions on encountering congestion.

To combat the ill-effects of multiplicative decrease on a single packet loss, various researchers have been looking at the class of "equation-based control algorithms" [20, 27, 35]. These are schemes where the sender measures the packet loss rate and round-trip time over some past time and uses these estimates to determine a TCP-compatible transmission rate based on an equation relating TCP throughput to the loss rate [26]. The effectiveness of such schemes depends critically on the method used to estimate loss rate [29, 11]. It will be interesting to compare binomial algorithms with equation-based control.

8 Concluding remarks

In this paper we presented and evaluated a new family of nonlinear congestion management algorithms, called *binomial algorithms.* They generalize the familiar class of linear algorithms; during the increase phase, $w_{t+r} = w_t + \alpha/w_t^k$ and on experiencing a loss , $w_{t+\delta t} = w_t - \beta w_t^l$. We showed that a network with sources running the same binomial algorithm converges to fairness under a synchronized-feedback assumption if $k + l > 0$ and at least one of k or l is positive, and that the throughput of a binomial algorithm $\lambda \propto 1/p^{\frac{1}{k+l+1}}$, where p is the loss rate it encounters. As a corollary, a binomial algorithm is TCP-friendly if and only if $k + l = 1$ and $l \leq 1$ (the $k + l$ rule).

The $k+l$ rule represents a fundamental trade-off between probing aggressiveness and congestion responsiveness, with small values of l being less drastic in window reduction. Hence, we believe that binomial algorithms with $l < 1$ are well-suited to applications like audio and video that do not react well to drastic multiplicative decrease. Our preliminary experiments seem to justify this hypothesis, although more validation and research is needed before widespread deployment can be recommended. For applications that simply want to transmit as much data as quickly as they can without worrying about the degree of rate variations while doing so, the $k + l$ rule shows that AIMD is a very good strategy. Of all the TCP-friendly binomial algorithms, AIMD is the most efficient in aggressively probing for bandwidth.

Our simulation results showed good performance and interactions between binomial algorithms and TCP, especially using RED. We also found that TCP-friendliness does not necessarily imply TCP-compatibility in a network with drop-tail gateways—a binomial algorithm like IIAD or SQRT obtains higher long-term throughput than TCP because of a higher average buffer occupancy. Active queue management schemes like RED allow binomial algorithms and TCP to interact well with each other, which may be viewed as another among many important reasons to eliminate drop-tail gateways from the Internet infrastructure.

We believe that the results presented in this paper lead to a deeper understanding of the issues involved in the increase and decrease phases of a congestion management algorithm and in the notions of TCP-friendliness and TCP-fairness. We hope that our findings will spur further research into congestion control dynamics to obtain a fundamental understanding of a future Internet with multiple coexisting congestion control algorithms and protocols.

Acknowledgments

This work was supported by research grants from the NTT Corporation, DARPA (Grant No. MDA972-99-1-0014), and IBM Corporation. We thank David Andersen, John Byers, Dah Ming Chiu, David Clark, Sally Floyd, Allen Miu, Srinivasan Seshan, Alex Snoeren, and Roshni Srinivasan for helpful comments on earlier drafts of this paper.

References

[1] ALLMAN, M., AND PAXSON, V. *TCP Congestion Control*. Internet Engineering Task Force, April 1999. RFC 2581.

[2] BALAKRISHNAN, H., RAHUL, H. S., AND SESHAN, S. An Integrated Congestion Management Architecture for Internet Hosts. In *Proc. ACM SIGCOMM* (Sep 1999).

[3] BRADEN, B., CLARK, D., CROWCROFT, J., DAVIE, B., DEERING, S., ESTRIN, D., FLOYD, S., JACOBSON, V., MINSHALL, G., PARTRIDGE, C., PETERSON, L., RAMAKRISHNAN, K., SHENKER, S., WROCLAWSKI, J., AND ZHANG, L. *Recommendations on Queue Management and Congestion Avoidance in the Internet*. Internet Engineering Task Force, Apr 1998. RFC 2309.

[4] BRAKMO, L. S., O'MALLEY, S. W., AND PETERSON, L. L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proc. ACM SIGCOMM '94* (Aug. 1994).

[5] CHERITON, D., AND WILLIAMSON, C. VMTP as the Transport Layer for High-Performance Distributed Systems. *IEEE Commun. Mag.* (June 1989), 37–44.

[6] CHIU, D.-M., AND JAIN, R. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems 17* (1989), 1–14.

[7] CLARK, D., LAMBERT, M. L., AND ZHANG, L. NETBLT: A High Throughput Transport Protocol. In *Proc. ACM SIGCOMM* (Aug. 1988).

[8] Dummynet. `http://www.iet.unipi.it/~luigi/ip_dummynet`, Sept. 1998.

[9] FABER, T., LANDWEBER, L., AND MUKHERJEE, A. Dynamic Time Windows: Packet Admission Control with Feedback. In *Proc. ACM SIGCOMM* (1992).

[10] FLOYD, S., AND FALL, K. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Trans. on Networking 7*, 4 (Aug. 1999).

[11] FLOYD, S., HANDLEY, M., PADHYE, J., AND WIDMER, J. Equation-Based Congestion Control for Unicast Applications. `http://www.aciri.org/tfrc/`, June 2000.

[12] FLOYD, S., AND JACOBSON, V. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking 1*, 4 (Aug. 1993).

[13] GOLESTANI, S. J., AND S., B. A Class of End-to-End Congestion Control Algorithms for the Interney. In *Proc. ICNP* (1998).

[14] HOE, J. C. Improving the Start-up Behavior of a Congestion Control Scheme for TCP. In *Proc. ACM SIGCOMM '96* (Aug. 1996).

[15] JACOBSON, V. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM* (Aug 1988).

[16] JAIN, R. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.

[17] KESHAV, S. Packet-Pair Flow Control. *IEEE/ACM Transactions on Networking* (Feb. 1995).

[18] LAKSHMAN, T. V., AND MADHOW, U. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Trans. on Networking 5*, 3 (1997).

[19] LAKSHMAN, T. V., MADHOW, U., AND SUTER, B. Window-based Error Recovery and Flow Control with a Slow Acknowledgement Channel: A study of TCP/IP Performance. In *Proc. Infocom 97* (April 1997).

[20] MAHDAVI, J., AND FLOYD, S. The TCP-Friendly Website. http://www.psc.edu/networking/tcp_friendly.html, 1998.

[21] MATHIS, M., AND MAHDAVI, J. Forward Acknowledgement: Refining TCP Congestion Control. In *Proc. ACM SIGCOMM* (Aug 1996).

[22] MATHIS, M., MAHDAVI, J., FLOYD, S., AND ROMANOW, A. *TCP Selective Acknowledgment Options*. Internet Engineering Task Force, 1996. RFC 2018.

[23] MCCANNE, S., JACOBSON, V., AND VETTERLI, M. Receiver-driven Layered Multicast. In *Proc ACM SIGCOMM* (Aug. 1996).

[24] ns-2 Network Simulator. http://www-mash.cs.berkeley.edu/-ns/, 1998.

[25] OTT, T., KEMPERMAN, J., AND MATHIS, M. The Stationary Distribution of Ideal TCP Congestion Avoidance. ftp://ftp.bellcore.com/pub/tjo/TCPwindow.ps, 1996.

[26] PADHYE, J., FIROIU, V., TOWSLEY, D., AND KUROSE, J. Modeling TCP throughput: A Simple Model and its Empirical Validation. In *Proc. ACM SIGCOMM* (Sept. 1998).

[27] PADHYE, J., KUROSE, J., TOWSLEY, D., AND KOODLI, R. A Model Based TCP-friendly Rate Control Protocol. In *Proc. NOSSDAV* (July 1999).

[28] POSTEL, J. B. *Transmission Control Protocol*. Internet Engineering Task Force, September 1981. RFC 793.

[29] RAMESH, S., AND RHEE, I. Issues in Model-Based Flow Control. *Technical Report TR-99-15, Department of Computer Science, North Carolina State University* (1999).

[30] REJAIE, R., HANDLEY, M., AND ESTRIN, D. Quality Adaptation for Unicast audio and video. In *Proc. ACM SIGCOMM* (September 1999).

[31] REJAIE, R., HANDLEY, M., AND ESTRIN, D. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proc. IEEE INFOCOM* (March 1999).

[32] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, Jan 1996. RFC 1889.

[33] SISALEM, D., AND SCHULZRINNE, H. The Loss-Delay Adjustment Algorithm: A TCP-friendly Adaptation Scheme. In *Proc. NOSSDAV* (Jul 1998).

[34] STEVENS, W. R. *TCP/IP Illustrated, Volume 1*. Addison-Wesley, Reading, MA, Nov 1994.

[35] TAN, W., AND ZAKHOR, A. Real-time Internet Video Using Error Resilient Scalable Compression and TCP-friendly Transport Protocol. *IEEE Trans. on Multimedia* (May 1999).

[36] WANG, A., AND SCHWARTZ, M. Achieving bounded fairness for multicast and TCP traffic in the Internet. In *Proc. ACM SIGCOMM* (September 1998).

[37] WANG, Z., AND CROWCROFT, J. A New Congestion Control Scheme: Slow Start and Search (Tri-S). *ACM Computer Comm. Review 21*, 1 (Jan. 1991), 32–43.