Binomial Congestion Control Algorithms

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 each congestion indication (or loss) 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, those 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 to TCP, especially across 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 [12] implemented in its dominant transport protocol, TCP [24, 30]. 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 including streaming and real-time audio and video because its reliability and ordering semantics increases end-to-end delays and delay variations. Furthermore, many of these applications do not react well to the large and abrupt reductions in transmission rate that using TCP-style AIMD would entail on packet losses. To be safe for deployment in the Internet, however, the protocols used by these applications must implement "TCP-compatible" congestion control algorithms, which interact well with TCP and maintain the stability of the Internet [4]. The idea is to 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" [16]. It is well-known that the throughput λ of a flow with TCP's AIMD congestion control is related to its loss rate p as $\lambda \propto S/(R\sqrt{p})$, where S is the packet size and R the connection's round-trip time [15, 21, 8, 22]. An algorithm is said to be "TCP-friendly" 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 on each packet loss, 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 familiar class of linear control algorithms (of which AIMD is one example), and understanding how a TCP-friendly algorithm competes with TCP for bottleneck resources. While previous work on equation-based congestion control has shown how adjusting the transmission rate as a function of the loss-rate enables interesting congestion control for streaming applications [9], our work opens up the possibility of using increase/decrease rules without tracking loss-rates.

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 simple manner:

$$\text{I: } w_{t+R} \leftarrow w_t + \alpha / w_t^k; \alpha > 0$$
$$\text{D: } w_{t+\delta t} \leftarrow w_t - \beta w_t^l; 0 < \beta < 1 \quad (2)$$

These nonlinear 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 [12]); for $k = -1, l = 0$, we get MIAD; and for $k = 0, l = 0$ we get AIAD, thereby covering the class of all linear algorithms.

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; because they possess the property that any $l < 1$ has a decrease that is in general *less than* a multiplicative decrease, they might be suitable for streaming and real-time audio and video. In particular, if there exist values of k and l (other than $k = 0, l = 1$) for which binomial algorithms are TCP-friendly, then a spectrum of potentially safe congestion management mechanisms that are usable by streaming Internet applications would become available to us. It should be noted that the generalized AIMD algorithms obtained by choosing different values of α and β in Equations 1 are also members of the binomial family. However, by just varying α and β, the window oscillations still remain multiplicative. For better control over the oscillations, binomial algorithms make α and β (in Equations 1) functions of the current window value.

Our major finding is that several TCP-friendly binomial algorithms exist. Based on theoretical analysis and simulation experiments, 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-friendly 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, co-existing binomial control algorithms 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 does not always imply 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 all competing flows. Hence, while binomial algorithms are TCP-friendly (because they satisfy the λ-p relationship), they become TCP-compatible in the presence of RED gateways.

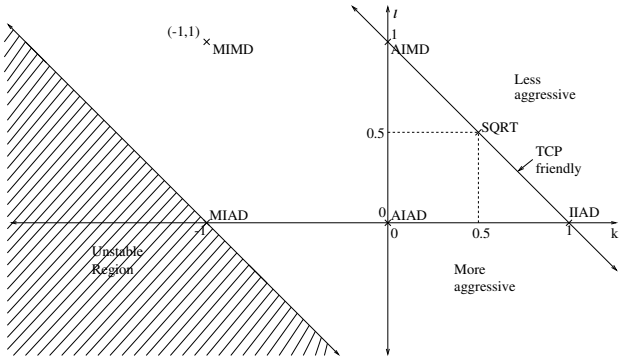

Figure 1. The (k, l) space of nonlinear controls from our family, with the $k+l = 1$ line showing the set of TCP-compatible controls.

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 algorithms, 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 for given α and β. 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. This observation shows the suitability of binomial algorithms as a good theoretical framework for evaluating additive increase/multiplicative decrease algorithms.

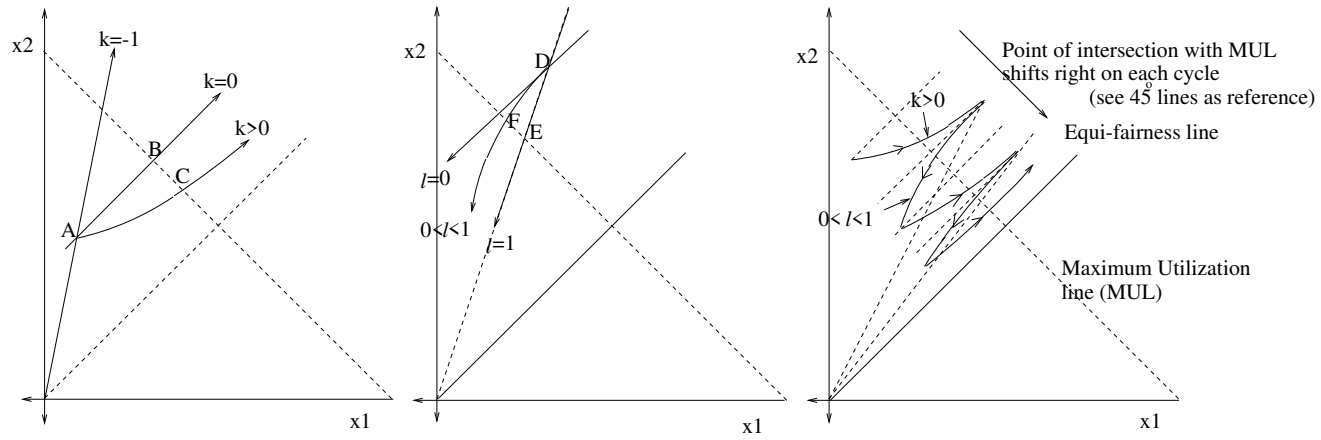

Figure 2. Sample path showing the convergence to fairness for an inverse increase proportional decrease algorithm.

2 Binomial congestion control algorithms

In this section, we discuss 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, showing that it converges to fairness under simplified conditions of synchronized feedback to sources. Although this makes some simplifying assumptions, we then 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.

We note that the window-adjustment policy is only one component of the complete congestion control protocol derived from binomial algorithms. Other mechanisms such as congestion detection (loss, ECN, etc.), retransmissions (if required), round-trip time (RTT) estimation, and connection startup (e.g., slow-start) remain the same as TCP. We evaluate a protocol that uses TCP/Reno's slow-start on startup and packet losses as an indicator of congestion, using TCP/Reno's RTT estimation. A practical protocol for Internet audio/video streaming would use the same approach, including timeouts followed by slow start on experiencing persistent congestion, but without using TCP's retransmission mechanisms.

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 relative current window sizes, x_i, of each source, normalized to a value between 0 and 1. 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 joining (x_1, x_2) to the origin. Fairness is unchanged.

Because binomial algorithms are nonlinear, 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. We also find that schemes for which $k + l \leq -1$ are unstable (Figure 1) because λ does not decrease with increasing p in this realm. Such schemes do not reduce their transmission rates in response to increasing loss-rates.

[1]For a network with n connections each with a share x_i of a resource, the fairness index $f = (\sum x_i)^2/(n \sum x_i^2)$ [13].

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. 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 (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 shape as shown in the middle picture of Figure 2; this curve is in-between the two $l = 0$ and $l = 1$ lines, and causes the system to evolve to an underutilized 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 that 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 always progress toward the fair allocation point. When $x_2 > x_1$, this continually moves downwards, while when $x_2 < x_1$, it continually moves upwards, towards the $x_2 = x_1 = 1/2$ point. Once $x_1 = x_2$, a binomial algorithm behaves like a linear algorithm, moving on the equi-fairness line, though with different magnitude of oscillation depending on k and l.

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 the point (x_1, x_2), without moving toward the fair-allocation point.

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

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

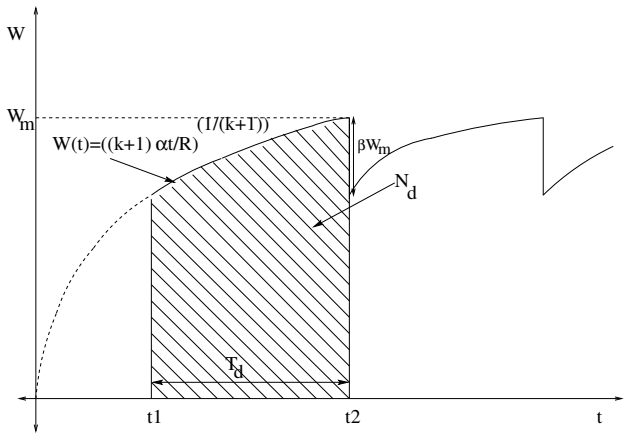

Figure 3. Functional form of window vs time curve.

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 [14] and Floyd [8]. Using the increase rule of Equation 2, we get using a continuous fluid approximation 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 congestion occurs. 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

$$T_D \;=\; \frac{R}{\alpha(k+1)}[W_m^{k+1} - (W_m - \beta W_m^l)^{k+1}]$$

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

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:

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

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 $\frac{1}{p^{1/2}}$, and thus: [3]

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

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 1 each RTT, rather than an increase by $1/w$ on receipt of each acknowledgement. Also, we ignore timeouts in the analysis presented here, but the results do not change in any significant way. We also note that these results also hold for the *random-loss* model first analyzed by Ott *et al.* in the context of TCP [21].

3 Simulation results

In this section, we present the results of our *ns-2* [20] simulations of various binomial algorithms. We start by investigating the interactions between connections running a TCP-friendly binomial 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 binomial algorithms: IIAD (inverse-increase/additive-decrease; $k=1, l=0$) and SQRT ($k=l=0.5$; this corresponds to an increase inversely proportional to the square-root of the current window and a decrease proportional to it). We then demonstrate the effect of these algorithms on the magnitude of oscillations in the transmission rates of the congestion control protocols. We conclude this section by studying the performance of IIAD and SQRT in the presence of multiple bottlenecks.

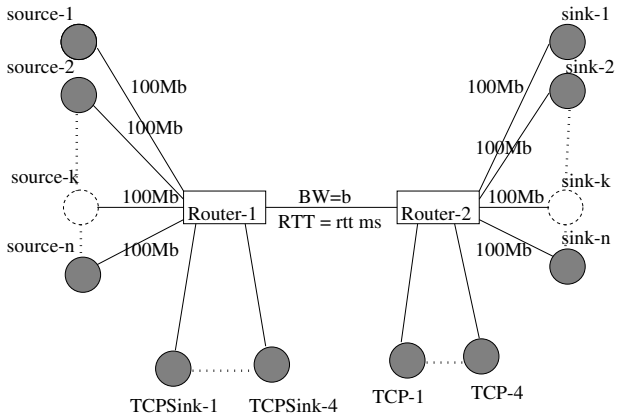

Figure 4. Simulation topology (delays of links for which no delay is specified are 1ms).

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, where all connections have an almost-identical round-trip propagation delay equal to RTT. There are 4 TCP connections in the reverse direction sharing the bottleneck to introduce ACK compression and eliminate any synchronizations. 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 the Random Early Drop (RED) buffer management algorithm at the bottleneck gateway [10]. 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 $0.2Q$ and $0.8Q$ respectively, and the connections used a packet size of 1 KByte. Each connection was started at uniformly chosen random times in

[3] A detailed analysis considering timeouts can be done in a manner almost identical to Padhye *et al.'s* analysis [22], to yield $\lambda \propto \frac{1}{R\sqrt{\frac{\beta p}{\alpha}}+T_0 min(1,3\sqrt{\frac{\beta p}{\alpha}})p(1+32p^2)}$ for TCP-friendly binomial algorithms.

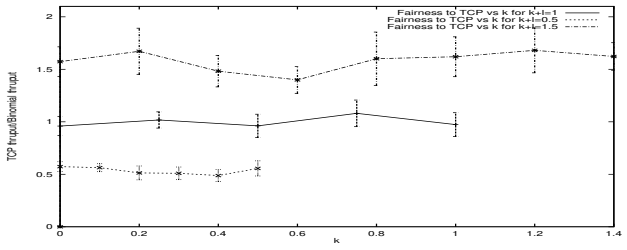

Figure 5. Ratio of the throughput of TCP AIMD to the throughput of a binomial algorithm, as a function of k. The algorithms that satisfy the $k + l$ rule are the ones for which this ratio is closest to unity. When $k + l = 0.5$, the binomial algorithm obtains noticeably higher throughput relative to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

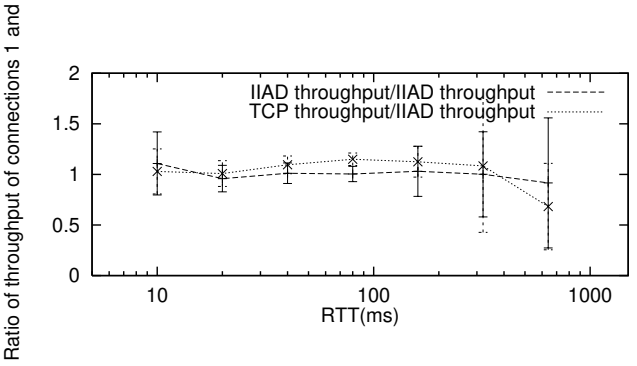

Figure 6. Ratio of throughputs of two connections sharing the bottleneck along with 95% confidence intervals ($n = 10, b = 10$ Mbps).

$[0, 2]$ seconds and throughput was calculated from $t = 50s$ to $t = 200s$.

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 and SQRT Algorithms

While IIAD is less aggressive than AIMD in the rate at which it probes for bandwidth ($k = 1$), it only reduces its window

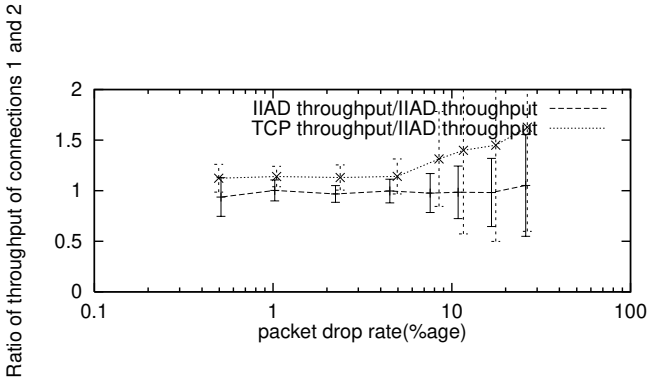

Figure 7. Ratio of throughputs of two connections sharing the bottleneck along with 95% confidence intervals ($n = 10, rtt = 100ms$).

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, $\alpha = 1.5, \beta = 1$ (we use $\beta = 1$ to reduce the errors in window adjustment due to the requirement for integral window values in *ns-2*). Although the analysis given in this paper suggests $\alpha = 2$ and $\beta = 1$, a more detailed analysis considering timeouts gives values close to $\alpha = 1.5$ and $\beta = 1$; detailed results analysing the sensitivity of α and β are available in the technical report [3], which shows that all values of α/β in the range $[0.5, 0.7]$ are roughly equivalent.

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, $n = 10$, with five connections of each kind, and each connection was started at a random time in $[0, 2]$ seconds. Each experiment was conducted using a bottleneck bandwidth $b = 10$ Mbps and a round-trip time RTT between 10ms and 640ms.

Figure 6 plots the throughput ratio for two IIAD connections on one curve and for one IIAD and one TCP connection on the other. These results show that IIAD is fair to both the IIAD and to TCP across a wide range of RTT values. At very large RTT, the standard deviation of results increases because the time for which simulations are run becomes small in terms of the number of round-trips for which the connections are active.

We also plot the effect of varying packet loss-rates in the Internet on the fairness of IIAD algorithms in figure 7. This curve also shows that IIAD connections are fair to TCP and to themselves across a wide range of loss-rates. At very high loss-rates, TCP gains because of its more aggressive probing for bandwidth, and is able to recover faster from burst losses. Similar results were obtained with SQRT algorithm as well. In fact as expected, SQRT algorithm gave faster convergence to fairness and hence was fairer to TCP over smaller time scales than IIAD algorithm [3].

We now consider the impulse-response behavior of the binomial algorithms. Our experiences with these experiments across several binomial algorithms have convinced us that slow-start (or a similar mechanism) is an important compo-

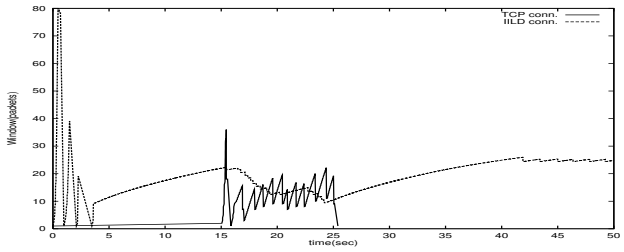

Figure 8. Response of a long-lived IIAD flow to a TCP impulse. In this experiment, $b = 1.5$ Mbps and $RTT = 50$ ms.

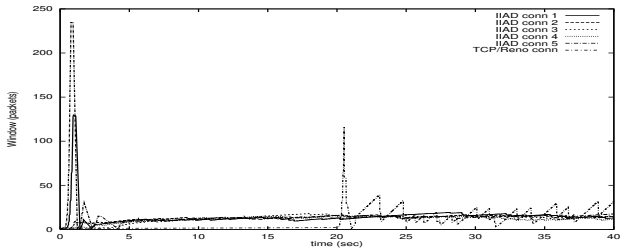

Figure 9. A TCP AIMD connection grabs its fair share of bandwidth in the presence of many long-lived IIAD flows ($n = 16, b = 9$ Mbps, $RTT = 50$ ms). For clarity, we only show the window sizes of five IIAD connections and the TCP connection.

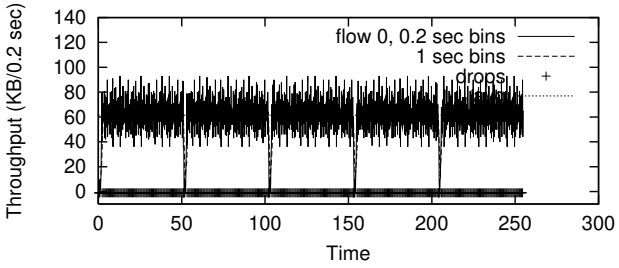

Figure 10. A TCP flow responding to periods of low loss-rate interspersed with periods of high loss-rate.

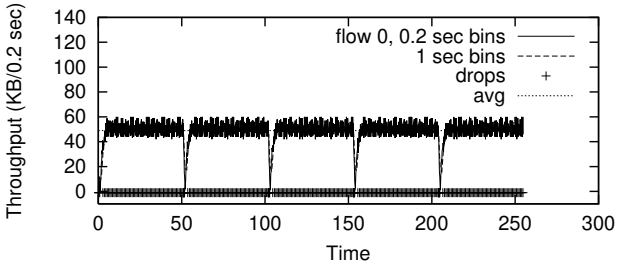

Figure 11. An IIAD flow responding to periods of low loss-rate interspersed with periods of high loss-rate.

nent 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 8, which shows how slow-start enables a new TCP connection to catch up and share bandwidth with a long-running IIAD connection.

These results also hold when the number of connections is increased. Figure 9 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 (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. Similar results hold for IIAD and SQRT connections in the presence of TCP connections [3].

3.3 Reduction in oscillations

In this section, we demonstrate the effectiveness of TCP-friendly binomial algorithms in reducing bandwidth oscillations. The loss-rate at the bottleneck link was switched between 25% and 0.5%. The low loss-rate period lasted for 50s while the high loss-rate period lasted for 1s. We study the effect of this loss pattern on TCP AIMD, IIAD, SQRT, and a generalized-AIMD control with increase/decrease parameters set to $\alpha = 0.31$ and $\beta = 0.125$ in Equation 1.

Figure 10 shows the packet drops and the transmission rate, averaged over 0.2s, 1s, and the entire connection, of a TCP AIMD connection for this loss pattern. As expected TCP AIMD shows a noticeable amount of oscillation in transmission rate even when the loss rate is constant. On the positive side, it responds fast to changes in loss rate. Figures 11 and 12 show the same results for IIAD and SQRT. IIAD shows negligible oscillations, but is slower to respond to bandwidth changes. We also plot the oscillations resulting from using SQRT and GAIMD. These results show the tradeoffs between increase/decrease rules, where slower oscillations also result in slower response in general. The relative merits of these various slowly responsive congestion control algorithms, including a comparison to equation-based approaches, is a topic of on-going research.

3.4 Multiple connections and 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, we simulate several concurrent connections running different algorithms sharing the bottleneck. The topology we use is shown 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 14, we

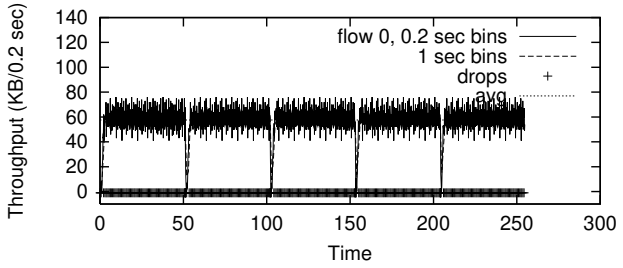

Figure 12. A SQRT flow responding to periods of low loss-rate interspersed with periods of high loss-rate.

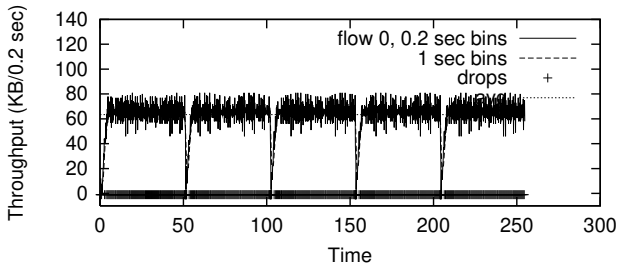

Figure 13. A generalized AIMD flow responding to periods of low loss-rate interspersed with periods of high loss-rate.

plot the mean value of the fairness index (ten runs for each 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 15. The maximum number of HTTP connections for each HTTP source was set to 5 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 16. 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 IIAD with SQRT.

4 TCP-friendliness vs TCP-compatibility

We now study the interactions between binomial algorithms and TCP AIMD over a drop-tail bottleneck gateway, observing some surprising effects. Figure 17 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 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.

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

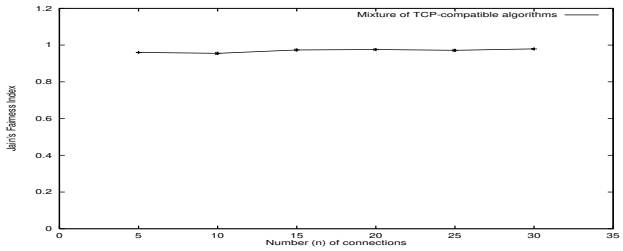

Figure 14. Plot showing Jain's Fairness Index as a function of the number of TCP-compatible connections sharing a bottleneck. In each experiment, the total number of connections was divided equally into five categories corresponding to $k = 0, 0.25, 0.5, 0.75, 1$. In these experiments, $b = 50$ Mbps, $RTT = 50$ ms. The (tiny) error-bars show 95% confidence intervals.

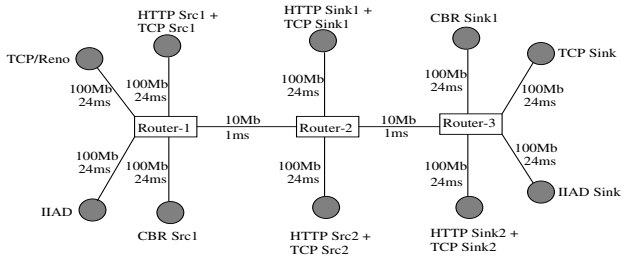

Figure 15. Topology with multiple bottlenecks and background traffic.

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 a queue optimized for TCP-style AIMD. 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 the loss-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.

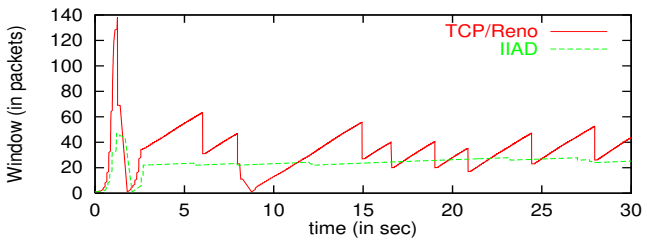

Figure 16. Window variation vs. time for the topology with multiple bottlenecks.

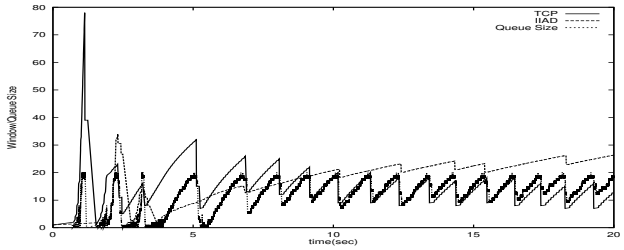

Figure 17. Plot showing window/queue size variation for TCP/Reno and SQRT algorithms sharing the bottleneck with drop-tail gateways($b = 3$ Mbps, $RTT = 50$ ms).

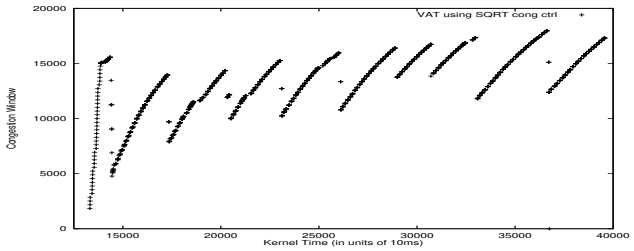

Figure 18. Window variation for a *vat* session using SQRT congestion control with a bottleneck configured using Dummynet ($b = 50$ Kbps,$RTT = 900$ ms).

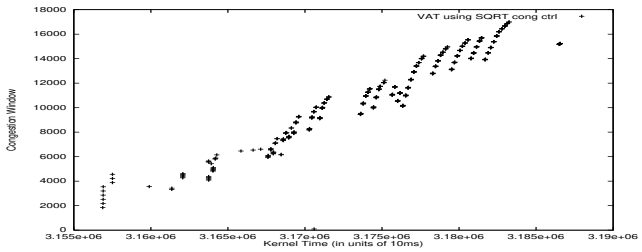

Figure 19. Window variation for a *vat* session using SQRT congestion control across an Internet path.

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 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 adverse 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.

5 Implementation

We implemented the SQRT congestion control algorithm in the Linux 2.2.9 kernel as part of the Congestion Manager (CM) [2] to provide congestion-controlled UDP sockets. We experimented with the Internet audio conferencing tool, *vat*, in unicast mode. Figure 18 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 [7]. 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 19 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.

One concern for the large-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, the prevention of congestion collapse *does not* require that flows reduce their sending rate by half in response to a single congestion indication. Furthermore, our protocols do reduce their transmission window to one upon persistent congestion (e.g., on a timeout), which helps during times of extreme congestion. We believe that it is important to have such mechanisms in any congestion control derived from TCP-friendly binomial controls. The aim of this paper has been to present binomial controls as a viable option; further real-world experimentation needs to be done to engineer these controls before widespread deployment can occur.

6 Related work

Chiu and Jain analyzed the performance of linear controls, deriving the conditions for efficient convergence to fairness under a synchronized-feedback assumption [6]. 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 by reliable unicast transport, and included both window- and rate-based approaches. In addition to Jacobson's TCP algorithms [12] and various recent TCP enhancements (e.g., [5, 18, 17]), a prominent example is Ramakrishnan and Jain's linear DECBit scheme that used a multiplicative-decrease factor of $7/8$.

Recent trends in Internet applications and traffic have led to a renewed interest 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 [27]. They also propose a quality adaptation algorithm for discretely-layered streams at the receiver to handle the rate variations triggered by AIMD [26]. In the context of multicast, McCanne *et al.*'s receiver-driven layered multicast (RLM) incorporates a probing and rate reduction mechanism for layered video [19]. Sisalem and Schulzrinne's Loss-Delay-based Adjustment (LDA) scheme uses an AIMD rate control at the sender, using RTCP [28] for feedback [29]. Schemes like RAP and LDA can use a binomial algorithm (e.g., IIAD or SQRT) to avoid drastic rate reductions on encountering congestion. Golestani has formulated congestion control as a global optimization problem and has proposed a class of congestion control policies based on rewards and costs [11].

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" [9, 23, 31]. 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 [22]. The effectiveness of such schemes depends critically on the method used to estimate loss rate [9, 25]. Rhee *et al.* recently proposed the TEAR scheme where receivers estimate the TCP-friendly rate for senders to use. A comparison of binomial algorithms with equation-based approaches and TEAR is a topic of on-going research.

7 Conclusion

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.

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] BANSAL, D., AND BALAKRISHNAN, H. TCP-friendly Congestion Control for Real-time Streaming Applications. Tech. Rep. MIT-LCS-TR-806, MIT Laboratory for Computer Science, May 2000.

[4] 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.

[5] 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).

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

[8] 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).

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

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

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

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

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

[14] 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).

[15] 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).

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

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

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

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

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

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

[22] 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).

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

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

[25] 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).

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

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

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

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

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

[31] 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).