End-to-End Rate-Based Congestion Control: Convergence Properties and Scalability Analysis

Dmitri Loguinov, *Member, IEEE,* and Hayder Radha, *Senior Member, IEEE*

***Abstract*—In this paper, we study several properties of binary-feedback congestion control in rate-based applications. We first derive necessary conditions for generic binary-feedback congestion control to converge to fairness monotonically (which guarantees asymptotic stability of the fairness point) and show that AIMD is the *only* TCP-friendly binomial control with monotonic convergence to fairness. We then study steady-state behavior of binomial controls with n competing flows on a single bottleneck. Our main result here shows that combined probing for new bandwidth by all flows results in significant overshoot of the available bandwidth and rapid (often super-linear as a function of n) increase in packet loss. We also show that AIMD has the best scalability and lowest packet-loss increase among all TCP-friendly binomial schemes. We conclude the paper by deriving the conditions necessary to achieve constant packet loss regardless of the number of competing flows n and examine one new scheme with such constant packet loss called *ideally scalable congestion control* in both simulation and streaming experiments.**

***Index Terms*—Binomial algorithms, congestion control, MPEG-4, multimedia streaming, packet loss scalability.**

I. Introduction

CONGESTION is an inherent property of the currently best-effort Internet. Consequently, transport protocols commonly implement *congestion control*, which refers to end-to-end algorithms executed by a protocol in order to properly adapt the sending rate of a network flow to the available bandwidth in the network. Protocols with window-based end-to-end flow control utilize one or another version of TCP-friendly congestion control, which includes Jacobson's modifications to TCP [1], [15], TCP-like congestion control (e.g., [43]), binary-feedback algorithms (e.g., [2], [3], [6], [12], [21], [28], [38], [49]), and equation-based methods (e.g., [11], [40]). These algorithms are shown to work well in the environment where the sender relies on "self-clocking," which refers to the use of positive acknowledgment not only to recover lost packets, but also to slow down the sender during congestion [15].

However, current real-time streaming applications in the Internet [36], [44] typically rely on *rate-based* end-to-end flow control. Rate-based congestion control for transport-layer protocols has not made it very far within the IETF and so far has only been adopted in proprietary streaming applications [36], [44]. But even then, the exact operation of these controls in RealPlayer and Windows Media Player is not widely available and is believed to consist of additive-increase additive-decrease (AIAD) layer-dropping and layer-adding algorithms [36], [44].

In contrast to very few studies of rate-based controls in end-to-end applications, a large body of data-link algorithms exists [4], [22], [31], [39]; however, these methods often rely on routers to compute the sending rate of each flow (e.g., in ATM) and explicitly feed it back to the end flows. In the current Internet, such computation is considered too costly to be implemented in the network layer, which makes these methods unsuitable for end-to-end applications. Furthermore, ATM is rarely available at the desktops and few end-to-end paths are built entirely on top of the ATM technology. Thus, many emerging streaming protocols cannot use native ATM congestion control and have to rely on rate-based[1] end-to-end methods that utilize packet loss as the only feedback from the network. Our work analyzes the performance of such methods in video streaming and derives several novel results about both generic binary-feedback controls as well as their special nonlinear subclass called *binomial algorithms* [2], [3].

We first examine the problem of determining which increase–decrease functions of binary-feedback controls guarantee convergence to fairness. To keep the problem tractable, we only focus on *monotonic* convergence, which is a desirable property of congestion control since it guarantees asymptotic stability of the fairness point. Our results in this section show that AIMD is the only TCP-friendly binomial control with monotonic convergence to fairness.

We then focus on steady-state properties of binomial congestion control and derive long-term link utilization and packet-loss rates of these controls. This study shows that long-term average packet-loss rates increase as a super-linear function of the number of flows n, which prevents rate-based congestion controls from scaling to a large number of flows. Our work also finds that AIMD has the best scalability properties among TCP-friendly binomial algorithms.

We finish the paper by showing the existence of binomial controls called *ideally scalable congestion control* (ISCC) that do not suffer from the rapid packet-loss increase observed among the existing binomial schemes. We further evaluate one particular ISCC control equation both in simulation and Cisco exper-

Manuscript received August 25, 2001; revised January 22, 2002 and January 3, 2003; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Krunz.

D. Loguinov is with the Department of Computer Science, Texas A&M University, College Station, TX 77843 USA (e-mail: dmitri@cs.tamu.edu).

H. Radha is with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: radha@egr.msu.edu).

Digital Object Identifier 10.1109/TNET.2003.815291

[1]Note that window-based flow control could be used in real-time streaming, but it typically results in some form of quality-of-service (QoS) penalty (such as longer startup delays, more frequent buffer underflow events, etc.), because video is an inherently rate-based application.

iments and find it to scale significantly better than the existing methods.

Studying ISCC controls in this paper, we drift away from TCP-friendly schemes. Hence, we must mention breifly why we find such practice acceptable. In the future Internet, it is possible that UDP traffic will *not* compete with TCP in the same router queues (e.g., DiffServ may be used to separate these types of traffic at the router level). This intuition is driven by the fact that real-time flows have substantially different delay requirements from those of TCP, and it may not be practical to mix the two types of traffic in the same queues. Furthermore, rate-based applications are unlikely to be fully TCP-friendly, because they often do not follow TCP's *fast retransmit* and *timeout backoff* algorithms and do not rely on the "packet-conservation" principle [15] in their flow control.

The rest of the paper is organized as follows. Section II discusses related work. Section III studies the convergence properties of generic binary-feedback congestion control. Section IV analyzes steady-state average link utilization and packet loss of binomial schemes. Section V derives the rate of packet-loss increase as a function of the number of flows n. Section VI builds ISCC and shows the performance of one such control in both simulation and MPEG-4 streaming experiments. Section VII concludes the paper.

II. Related Work

A. End-to-End Binary-Feedback Congestion Control

Within the class of end-to-end congestion control protocols, we focus on the class of *binary-feedback* methods. Binary-feedback congestion control implements a simple reactive control system, which responds to congestion by decreasing the sending rate and responds to the absence of congestion by increasing the sending rate. Hence, at any stage, the decision of such congestion control is binary.

Furthermore, the increase and decrease functions of congestion control are assumed to be *local* [2], [6], which means that they only use the local state of a flow in computing the next value of the sending rate. In addition, many existing models use *memoryless* controls [2], [6], in which the amount of increase and decrease is based only on the value of the current sending rate rather than the history of the sending rate (e.g., several flavors of "AIMD with history" are examined in [27] and [28]). In this paper, we explicitly assume a local and memoryless model of binary-feedback congestion control.

To prevent high-frequency oscillations, congestion control is executed on discrete timescales of R time units long, where R is the delay of the control loop, which in many cases simply equals the round-trip time (RTT). Many papers study congestion control in the context of *window-based* flow control [2], [12], [28], [49] and apply control formulas to the size of congestion window *cwnd*. In such notation, assuming that the size of congestion window *cwnd* during interval i for a particular flow is given by w_i, binary-feedback congestion control can be summarized as

$$w_{i+1} = \begin{cases} w_i + W_I(w_i), & p_i = 0 \\ w_i - W_D(w_i), & p_i > 0 \end{cases} \tag{1}$$

where p_i is the congestion feedback during interval i (positive values indicate congestion), and W_I and W_D are the increase and decrease functions of *window-based* congestion control, respectively. In practice, feedback p_i is usually equal to the packet-loss rate observed by the flow during interval i.

Since our work focuses on *rate-based* streaming applications in which *cwnd* has little meaning, we must write an equivalent formulation of congestion control using the value of each flow's sending rate r_i instead of congestion window w_i. The conversion from the window-based notation to the rate-based notation is straightforward, i.e., each unit of w_i is equivalent to a rate of MTU/RTT bits/s, where the maximum transmission unit (MTU) is given in bits and the RTT is given in seconds. In other words, $r_i = w_i(\text{MTU}/\text{RTT})$.

Therefore, we can rewrite congestion control in (1) as

$$r_{i+1} = \begin{cases} r_i + R_I(r_i), & f = 0 \\ r_i - R_D(r_i), & f > 0 \end{cases} \tag{2}$$

where R_I and R_D are the increase and decrease functions of *rate-based* binary-feedback congestion control, respectively.

One special case of end-to-end congestion control is given by *binomial algorithms*, in which the increase and decrease curves are power functions of the current rate [2]

$$\begin{cases} W_I(w) = \alpha w^{-k} \\ W_D(w) = \beta w^{l} \end{cases} \Leftrightarrow \begin{cases} R_I(r) = \lambda r^{-k} \\ R_D(r) = \sigma r^{l} \end{cases} \tag{3}$$

where all constants α, β, λ, and σ are positive. For binomial algorithms, the difference between the two notations lies only in the constants in front of the corresponding power functions. Hence, the conversion from the window-based to the rate-based notation is supplied by the following formulas:

$$\lambda = \alpha \left(\frac{\text{MTU}}{\text{RTT}}\right)^{k+1} \quad \text{and} \quad \sigma = \beta \left(\frac{\text{MTU}}{\text{RTT}}\right)^{1-l}. \tag{4}$$

Throughout the rest of this paper, we will use both versions of binomial algorithms in (3), sometimes referring to constants (λ, σ) instead of constants (α, β), while keeping in mind the conversion in (4).

A special case of binomial congestion control that is implemented in TCP is called additive increase multiplicative decrease (AIMD) [6], [15]. In AIMD, $k = 0$, i.e., $W_I(w) = \alpha$ $(\alpha > 0)$, and $l = 1$, i.e., $W_D(w) = \beta w$ $(0 < \beta < 1)$. Hence, in the absence of congestion, AIMD probes for new bandwidth linearly, and during congestion, AIMD backs off exponentially. Note that notation AIMD (α, β) refers to a *window-based* version of AIMD with increase–decrease constants α and β. Therefore, AIMD(1, 1/2) (implemented in TCP) increases the rate by one packet per RTT in the absence of congestion and decreases the rate by half during congestion.

AIMD (α, β) is *TCP-friendly* if it achieves the same average throughput when competing with a TCP connection under the

same end-to-end conditions. The necessary condition for such long-term fairness is [12], [28], [49][2]

$$\alpha = \frac{3\beta}{2-\beta}. \tag{5}$$

On the other hand, for binomial congestion control (3) to be TCP-friendly, Bansal *et al.* [2] show that $k + l$ must be equal to 1. Among such (non-AIMD) TCP-friendly binomial congestion control, they propose two methods called inverse-increase additive-decrease (IIAD) with $k = 1$, $l = 0$, and Square Root (SQRT) with $k = l = 1/2$.

Finally, we should mention that the analysis of increase–decrease congestion control typically assumes an ideal network with synchronized and immediate feedback [2], [6], [19], [27], [28]. *Synchronized* feedback means that all flows sharing a congested link receive notifications about packet loss at the same time. *Immediate* feedback means that if the capacity of any link along an end-to-end path is exceeded during interval i, feedback p_i is strictly positive. Under these ideal conditions, Chiu and Jain [6] show that all AIMD schemes converge to a fair state. In addition, Bansal *et al.* [2] show that for binomial algorithms (3) to converge to fairness, $k + l$ must be strictly greater than zero.

B. ATM Rate-Based Congestion Control

Asynchronous Transfer Mode (ATM) networks implement congestion control on the data-link layer. The available bitrate (ABR) service in ATM allows connections to send data without prior admission control or reservations (ABR is similar to the best-effort class in DiffServ). Two types of congestion control were originally proposed for ABR—credit-based [24] and rate-based [39]. The credit-based approach implements hop-by-hop window-based congestion control [39]. The rate-based methods rely on the network to provide congestion feedback in special resource management (RM) cells. In its simplest form, the feedback consists of a single-bit congestion indication. These schemes include forward explicit congestion notification (FECN), backward ECN (BECN), proportional rate controller (PRCA), and several other derivatives [39]. Upon receiving the feedback from the network, these schemes typically use linear controls to reduce or increase their sending rate. However, even the most advanced controllers in this category suffer from undesirable performance penalties when RM cells are lost (which leads to increased congestion and/or poor fairness together with oscillation).

In addition to congestion-notification schemes, the feedback in ABR may carry the *explicit rate* (ER) that the flow should use. Depending on the particular mechanism, ABR ER may or may not require the switch to maintain local per-flow state. Among a volume of work in this area, the proposed schemes range from simple [such as enhanced PRCA (EPRCA), adaptive proportional rate control (APRC), and EPRCA+ [39]] to more sophisticated with provable convergence and stability. For example, Benmohamed *et al.* [4] propose and study proportional and proportional-derivative (PD) controllers applied to the switch's queue size, Mascolo *et al.* [31] use a fair-queuing switch that feeds back the size of each flow's queue, and Kulkarni *et al.* [23] model the ABR service using a quasi-death–birth process and incorporate round-trip delays into their model. Kalyanaraman *et al.* [17] propose a rate-based control called explicit rate indication for congestion avoidance (ERICA) in which the switch estimates the number of active flows and divides the available bandwidth equally among all competing flows. Kolarov *et al.* [22] use a dual PD controller to achieve good convergence to stability during both transient and steady states. Other research related to ATM ER including stochastic modeling of queue sizes and video transport over ATM ABR can be found in [7], [13], [14], [46], and [50].

C. Continuous Feedback and Utility Functions

Continuous-feedback congestion control explicitly takes the *value* of the feedback into the control loop. In other words, these methods can be summarized as a simple differential equation

$$\frac{dr}{dt} = F(r(t), p(t)) \tag{6}$$

where $r(t)$ is the sending rate and $p(t)$ is the continuous feedback from the network (typically, packet loss). Kelly *et al.* [19] propose and study an AIMD-like continuous-feedback controller (converted to the end-to-end context)

$$\frac{dr}{dt} = \alpha - \beta r(t)p(t). \tag{7}$$

The paper shows that flows using the controls in (7) converge to so-called *proportional fairness* and maximize the combined utility of all flows, where each utility function $U(x)$ is logarithmic. Massoulié *et al.* [33] further study fairness issues in congestion control and develop *minimum potential delay fairness*, which maximizes combined utility for users with hyperbolic functions $U(x)$. Kunniyur *et al.* [25] study long-term throughput behavior of proportional and minimum potential delay fairness [33] both analytically and in simulation. Johari *et al.* [16] prove stability of end-to-end continuous feedback controller (7) under nonnegligible propagation delays. Massoulié [32] examines a similar issue in the context of heterogeneous delays. Kar *et al.* [18] study distributed convergence of proportional-fairness controllers in networks where users have different utility functions $U(x)$. Continuous feedback controllers are further analyzed in the context of active queue management (AQM), e.g., in [30] and [47].

Continuous-feedback congestion control has a nice property of converging to a single stationary point and maintaining constant (i.e., smooth) transmission rate in the steady state. However, since this stationary point has a strictly nonzero packet loss, continuous-feedback controllers are incompatible with TCP flows as they are too aggressive in comparison to any binary-feedback controller. An interesting combination of both binary and continuous feedback called additive-increase

[2]Note that some papers [2], [48], [49] use a different notation, in which $W_D(w) = (1-\beta)w$ and this formula has a different form. Furthermore, if the rate of AIMD is dominated by timeouts, the formula assumes yet another form [49].

loss-proportional-decrease (AIPD) is evaluated by Lee *et al.* in [27].

D. Other Rate-Based Congestion Control

Keshav [20] studies rate-based flow control using packet-pair sampling of the available bandwidth in fair-queuing networks. A similar method is proposed by Legout *et al.* [29] in the context of layered multicast. Fendick *et al.* [10] show that delayed feedback results in oscillatory behavior of generic rate-based controllers. Mishra *et al.* [37] use rate-based controllers inside routers that exchange queue length information with neighbors to achieve hop-by-hop congestion control.

III. Convergence Properties

Not all increase–decrease functions R_I and R_D guarantee convergence to fairness. In the context of congestion control, *convergence to fairness* is usually defined as the ability of a number of identical flows sharing a common bottleneck link to reach a state in which their rates become equal and stay equal infinitely long. Even though in practice this is a very difficult goal to achieve due to nonuniform and delayed feedback, under ideal conditions of Chiu and Jain's model, many schemes can guarantee convergence to fairness.

One of the interesting properties of congestion control that guarantees its asymptotic stability is *monotonic convergence to fairness*. If fairness during interval i is given by f_i, $0 \le f_i \le 1$, then the following conditions are necessary for monotonic convergence:

$$\forall i: \quad f_{i+1} \ge f_i \quad \text{and} \quad \lim_{i \to \infty} f_i = f^* = 1. \tag{8}$$

From the point of view of control theory, monotonic convergence is desirable even though it is not necessary. Monotonically convergent controls guarantee stability of the stationary point f^*, which means that if a system is started in some neighborhood of the stationary point, it stays there. Consider an illustration of these two types of convergence in Fig. 1. Fig. 1(a) shows a particle that oscillates around the stationary point and eventually converges to it nonmonotonically (i.e., the distance from the particle to the attractor is not monotonically decreasing at a function of time). A similar scenario with monotonic convergence is shown in Fig. 1(b).

It is important to remember that the convergence to *fairness* is different from the convergence to *efficiency* in congestion control. Binary-feedback controls do not asymptotically converge to efficiency and always oscillate around the efficiency line. On the other hand, both binary and continuous feedback controls can asymptotically converge to fairness. In what follows in this section, we study monotonic convergence to *fairness* of generic binary-feedback controls and derive conditions necessary for asymptotic stability of point $f^* = 1$.

It is common [2], [6] to examine the case of two flows sharing a link, since the extension to n flows can be performed by considering flows pairwise. It is also common to use a continuous fluid approximation model [2] and disregard the discrete nature of packets. Furthermore, in this paper, we use a max–min fairness function f_i instead of Chiu's fairness index [6]. Recall

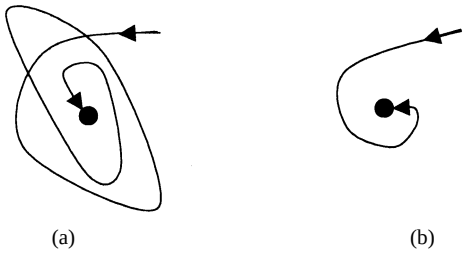


Fig. 1. (a) Nonmonotonic convergence leads to quasi-asymptotic stability. (b) Monotonic convergence leads to asymptotic stability.

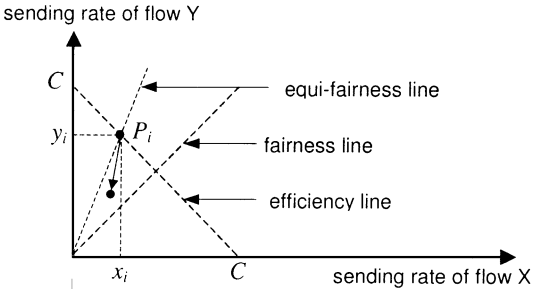


Fig. 2. Two-flow binary-feedback control system.

that max–min fairness of n flows with *nonzero* sending rates $(x_1, \ldots, x_n)$ is given by

$$f = \min_{i \ne j} \left(\frac{x_i}{x_j} \right). \tag{9}$$

Consider two flows X and Y sharing a bottleneck link under the above assumptions. Suppose that during interval i, the flows' sending rates are given by x_i and y_i, respectively. To help us understand the behavior of a two-flow control system, we use Fig. 2 from [6]. In the figure, the axes represent the sending rates of the two flows. Furthermore, line $y = x$ is known as the *fairness line* and represents points (x, y) in which fairness f equals 1. Assuming that the capacity of the bottleneck link is C, line $x + y = C$ is called the *efficiency line* and represents points in which the bottleneck link is about to overflow. Given a particular point $P_i = (x_i, y_i)$ in the figure, line $y = mx$ connecting P_i to the origin is called the *equi-fairness line* (i.e., points along the line have the same fairness $f_i = x_i/y_i = 1/m$). Furthermore, we define *efficiency* e_i of point P_i as the combined rate of both flows in that point, i.e., $e_i = x_i + y_i$.

A. Decrease Function

To ensure monotonic convergence and proper response to congestion signals, the following four conditions must hold during each *decrease* step assuming that the system is in some point P_i just before the decrease step. First, the efficiency in the new state must be strictly less than that in the old state, i.e., $e_{i+1} < e_i$. This condition ensures that flows back off during congestion. Second, fairness must not decrease in the new state, i.e., $f_{i+1} \ge f_i$, which guarantees monotonic convergence to fairness. As pointed out before, this condition is desired, but not necessary. Third, to properly maintain convergence, the system must not arbitrarily cross or oscillate

around the fairness line,[3] i.e., it must stay on the same side of the fairness line at all times. For the case in Fig. 2, we can write $(y_i > x_i) \Rightarrow (y_{i+1} > x_{i+1})$. Finally, the system must not allow rates below or equal to zero, i.e., given an arbitrary state with $x_i > 0$ and $y_i > 0$, we must guarantee that $x_{i+1} > 0$ and $y_{i+1} > 0$.

The first condition is equivalent to

$$x_{i+1} - x_i + y_{i+1} - y_i = -R_D(x_i) - R_D(y_i) < 0 \quad (10)$$

which can be satisfied with any positive function $R_D(x) > 0$, $\forall x > 0$. The second condition is equivalent to

$$\frac{x_{i+1}}{y_{i+1}} \geq \frac{x_i}{y_i}, \qquad x_i > 0,\ y_i > 0. \quad (11)$$

Expanding (11) using (2) and generalizing by dropping the indexes (the inequality depends only on x_i and y_i)

$$xR_D(y) - yR_D(x) \geq 0, \qquad \forall x > 0, \forall y > 0, x < y. \quad (12)$$

Writing $y = x + \Delta x,\ \Delta x > 0$ and dividing both sides by Δx

$$x(R_D(x + \Delta x) - R_D(x)) - \Delta x R_D(x) \geq 0 \quad (13)$$

$$\frac{R_D(x + \Delta x) - R_D(x)}{\Delta x} \geq \frac{R_D(x)}{x}, \qquad \forall x > 0,\ \Delta x > 0. \quad (14)$$

Restricting $R_D(x)$ to be a differentiable function for all $x > 0$, (14) is equivalent to

$$R'_D(x) \geq \frac{R_D(x)}{x}, \qquad \forall x > 0. \quad (15)$$

This result is interesting in its simplicity. To understand the properties of this solution, bring $R_D(x)$ to the left and integrate both sides [both x and $R_D(x)$ are positive]:

$$\int \frac{dR_D(x)}{R_D(x)} \geq \int \frac{dx}{x}, \qquad \forall x > 0 \quad (16)$$

$$\ln R_D(x) \geq \ln x + c_1, \qquad \forall x > 0 \quad (17)$$

$$R_D(x) \geq c_2 x, \qquad \forall x > 0. \quad (18)$$

The result in (18) shows that the original condition (15) restricts $R_D(x)$ to grow *no slower* than some linear function $c_2 x$. Using similar derivations, the third (noncrossover) condition results in

$$R'_D(x) < 1, \qquad \forall x > 0 \quad (19)$$

which means that $R_D(x)$ *must* grow slower than function x [i.e., the slope of $R_D(x)$ in all points $x > 0$ must be less than 1]. Finally, the fourth condition

$$x - R_D(x) > 0, \qquad \forall x > 0 \quad (20)$$

is automatically satisfied by combining (15) and (19) above.

To summarize by combining (18) and (19), $R_D(x)$ must be a positive differentiable function for all values of $x > 0$ and must be an asymptotically (i.e., for substantially large x) *linear* function of x, with the slope strictly less than 1. For example, AIMD function $R_D(x) = \sigma x$ satisfies these conditions for $0 < \sigma < 1$. Note that nonlinear functions [such as $x - \lg(x+2)$] can also be used, but their advantage over similar linear functions is not clear.

[3]It is impossible to maintain monotonic convergence and nonnegative rates if we allow the scheme to cross the fairness line. The interested reader can show this analytically following the derivations in this section.

B. Increase Function

The analysis of increase function $R_I(x)$ is similar to the above. This time, instead of four conditions, we have only three. First, the efficiency in the new state must increase (i.e., $e_{i+1} > e_i$), which guarantees that flows will probe for new bandwidth in the absence of congestion. Second, fairness must not decrease (i.e., $f_{i+1} \geq f_i$), which is the result of the same monotonicity requirement as before. Third, the system must not cross the fairness line (i.e., $y_{i+1} > x_{i+1}$). Crossing the fairness line violates monotonic converge to fairness and, as we will see later, never happens in practice among binomial schemes.

The first condition is satisfied with any positive function $R_I(x)$, i.e., $R_I(x) > 0$, $\forall x > 0$. The second condition is the opposite of (15) due to a different sign in (2)

$$R'_I(x) \leq \frac{R_I(x)}{x}, \qquad \forall x > 0. \quad (21)$$

Finally, the third condition is similar to (19), but assumes the following shape:

$$R'_I(x) > -1, \qquad \forall x > 0. \quad (22)$$

Using (21), we find that R_I must grow no faster than some linear function $c_3 x$ and using (22), R_I cannot decay quicker than $-x$. For example, the AIMD increase function $R_I(x) = \lambda$ again satisfies all conditions of monotonic convergence for $\lambda > 0$. We look at other examples in the next section while studying binomial [2] congestion control methods.

C. Convergence

Note that the above conditions still do not guarantee convergence to fairness. In other words, the conditions guarantee that if the system converges, it will do so monotonically, but the fact of convergence has not been established yet. Hence, we impose a final restriction on R_D and R_I—either the decrease or the increase step must strictly improve fairness, i.e., one of (15), (21) must be a *strict* inequality. If (15) is made into a strict inequality, we can no longer satisfy the condition in (19). Consequently, (15) must remain in its present form, and (21) must become a strict inequality.

The proof of convergence under the above restrictions proceeds as follows. First note that point $f^* = 1$ is an upper bound for sequence $\{f_n\}$: $f_n \leq f^*$. Since $\{f_n\}$ is monotonically nondecreasing and bounded above, it must converge. Suppose that the system does *not* converge to fairness under these conditions, i.e., $f_n \to F < 1$. In other words, the system converges to some stationary (fixed) point F strictly less than 1. Restart the system in point $f_0 = F$. During each congestion control cycle that includes at least one decrease step and one increase step fairness must *strictly* improve. Therefore, after several steps, fairness will reach some point $F^* > F$. The latter condition contradicts the fact that the system previously converged to a fixed

(stationary) point F. Consequently, the flows must converge to the only stationary point $f^* = 1$.

We finish this section by making an observation that the above convergence conditions hold for window-based increase–decrease functions W_I and W_D and cases when $x_i > y_i$.

IV. Properties of Binomial Algorithms

A. Overview

Consider binomial algorithms in (3). Both functions R_I and R_D are positive for $x > 0$ and, therefore, satisfy the first condition. The second condition (i.e., monotonically nondecreasing fairness) results in the following restrictions on k and l from applying (15) and a strict form of (21):

$$\begin{cases} -\lambda k x^{-(k+1)} < \dfrac{\lambda x^{-k}}{x} \\ \sigma l x^{l-1} \geq \dfrac{\sigma x^l}{x} \end{cases} \Rightarrow \begin{cases} k > -1 \\ l \geq 1. \end{cases} \tag{23}$$

The third (i.e., noncrossover) condition derived from (19) and (22) restricts l even further, but does not impose any limit on k (assuming sufficiently large x)

$$\begin{cases} k\lambda < x^{k+1} \\ l\sigma < x^{1-l} \end{cases} \Rightarrow \begin{cases} k = \text{anything} \\ l \leq 1. \end{cases} \tag{24}$$

The restriction on l in (24) is dictated by the fact that sending rate x of a flow is not limited *a priori* and the selection of a *positive* constant σ such that it is less than x^{1-l}/l, for substantially large $x > 0$, is feasible only when power $1 - l$ is nonnegative.[4] Later in this paper, we will show how restriction $l \leq 1$ can be lifted and what kind of advantages such schemes bring to congestion control protocols.

Assuming that the upper limit on x is not known, for binomial controls to possess monotonic convergence to fairness, both (23) and (24) must be satisfied. This combination of conditions constrains l to be strictly 1. Now recall that for binomial algorithms to be TCP-friendly, $k + l$ must also be 1. Consequently, the first major result of this paper is that AIMD ($k = 0$, $l = 1$) is *the only TCP-friendly binomial algorithm with monotonic convergence to fairness*. This result shows that linear controls of AIMD are expected to be more robust in their convergence (i.e., asymptotically stable) than any other TCP-friendly binomial control under a variety of network conditions. Nonmonotonic controls discussed below (IIAD and SQRT) can also guarantee stability of $f^* = 1$, however, this stability is *quasi-asymptotic*.

B. Nonmonotonic Schemes

Among nonmonotonic binomial schemes, Bansal *et al.* [2] study controls that are forced to reduce fairness during the *decrease* step (i.e., $l < 1$) according to (23) and increase fairness during the *increase* step (i.e., $k > -1$). This is illustrated in Fig. 3(a) from [2] and [6]. The equi-fairness line in the figure is given by $k = -1$ and $l = 1$. Any deviation to the right from the equi-fairness line increases fairness ($k > -1$ or $l > 1$) and deviation to the left decreases fairness ($k < -1$ or $l < 1$). The system in the figure first reduces fairness from point A to point B during the decrease step, but then recovers and achieves fairness in point C that is higher than that in point A.

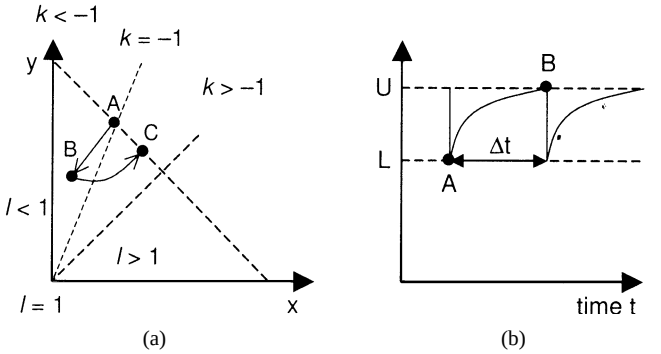


Fig. 3. (a) Nonmonotonic convergence to fairness. (b) Oscillation of the sending rate in the steady state.

Hence, as long as the resulting cycle (i.e., steps between points A and C) increases fairness, the scheme will eventually converge to fairness. Parameters k and l in (3) determine the shape of the curve along which the increase and decrease steps are taken. Constants λ and σ in (3) determine the size of each step taken along the corresponding curve. Note that if $k + l = 0$, the system will oscillate along the curve given by $R_D \propto x^l$ (or $R_I \propto x^{-k}$, which is the same). Under the assumption of a continuous fluid approximation [2], condition $k = -l$ makes the protocol take decrease steps along the same curve as the increase steps, resulting in no advance toward or away from the fairness line. This can be formulated differently: a binomial scheme converges to fairness *iff* $k + l > 0$ and diverges *iff* $k + l < 0$ [2].

Consequently, combining (24) with the convergence rule $k + l > 0$, we notice that the necessary restrictions on k and l for convergence of *nonmonotonic* binomial algorithms are $k > -1$ and $l \leq 1$.

C. Efficiency

The average efficiency is an important property of a congestion control scheme, which reflects how well the scheme utilizes the bottleneck bandwidth in the steady state. Higher efficiency is more desirable (but not necessarily at the expense of other properties such as packet loss or responsiveness). Analysis in this section not only helps us study the efficiency of binomial schemes, but is also a necessary background for our packet-loss scalability analysis in the next section.

It is common to define the *average efficiency* of a scheme as the long-term link utilization once the scheme has reached its steady state. In the steady state, each flow's sending rate will oscillate between two points, which we call the *upper point* (U) and the *lower point* (L) as shown in Fig. 3(b). When a single flow is present in the network, U equals the capacity of the bottleneck link C. When n flows compete over a shared link of capacity C, U equals C/n for each flow (because the flows have reached fairness by this time). In both cases, $L = U - \sigma U^l$ according to (3). In addition, since the pattern in Fig. 3(b) is repetitive, it is sufficient to determine the average throughput

[4]We implicitly assume that x is limited below by some constant $x_{\min}$. In window-based congestion control, $x_{\min}$ is equivalent to one unit of *cwnd* (i.e., *MTU/RTT*), and in rate-based congestion control, $x_{\min}$ is the minimum rate at which real-time material can be received.

of a flow during a single oscillation (i.e., between points A and B) rather than over a longer period of time. Note that in the window-based notation of congestion control, the maximum capacity of the link is given by $W = C(\mathrm{RTT}/\mathrm{MTU})$.

Using a continuous fluid approximation and results from [2], each flow's rate $x(t)$ *during the increase phase* (i.e., between points A and B) is given by

$$x(t) = \left(\frac{\lambda(k+1)t}{R}\right)^{1/(k+1)} \tag{25}$$

where R is a fixed duration of the control interval. Following [2], duration Δt between points A and B in Fig. 3(b) is

$$\Delta t = \frac{U^{k+1}\left(1-\left(1-\sigma U^{l-1}\right)^{k+1}\right)R}{\lambda(k+1)} \tag{26}$$

and the total amount of bits transmitted during the same interval is

$$X = \frac{U^{k+2}\left(1-\left(1-\sigma U^{l-1}\right)^{k+2}\right)R}{\lambda(k+2)}. \tag{27}$$

Consequently, we derive that the flow's average sending rate during the interval is $X/\Delta t$ and the *average efficiency* (i.e., percent link utilization) of a binomial congestion control scheme is

$$e = \frac{X}{U\Delta t} = \frac{(k+1)\left(1-\left(1-\sigma U^{l-1}\right)^{k+2}\right)}{(k+2)\left(1-\left(1-\sigma U^{l-1}\right)^{k+1}\right)}. \tag{28}$$

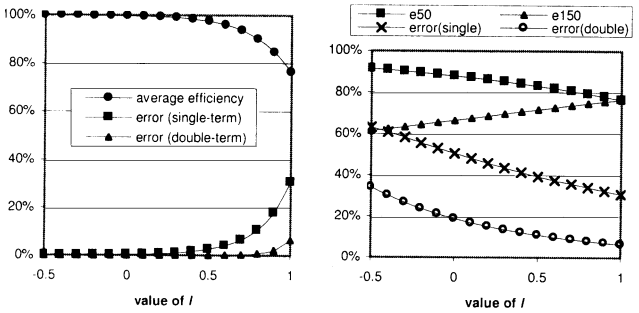
Equation (28) can be converted to the window-based notation by replacing σ with β and rate U with its window equivalent. We also note that for large n, the exact model of efficiency e in (28) becomes inapplicable when $U = C/n$ drops below $\sigma^{1/(1-l)}$. We can no longer use the above derivations, because term $1-\sigma U^{l-1}$ in the equation becomes negative. This is caused by the "drop-below-zero" effect [i.e., rate $x(t)$ becomes negative] that we tried to avoid before in (20). Nonzero rates were automatically guaranteed given monotonic convergence to fairness in (15), but in the absence of monotonicity, we must explicitly restrict n to the following:

$$n < \frac{C}{\sigma^{1/(1-l)}} = \frac{W}{\beta^{1/(1-l)}}. \tag{29}$$

We next focus on simplifying the expression in (28). Equation (28) contains two terms of the form $1-(1-z)^q$, which can be expanded using Taylor series to

$$1-(1-z)^q = qz - \frac{q(q-1)}{2}z^2 + \frac{q(q-1)(q-2)}{6}z^3 \cdots. \tag{30}$$

Note that for $l < 1$, the value of z is also less than 1, which means that the higher order terms in (30) get progressively smaller. We first consider keeping only the leading term of the Taylor expansion as used in [2], i.e., the qz term, and examine how well the resulting approximation reflects the actual values



(a) (b)

Fig. 4. (a) Exact average efficiency and relative error for one-term and two-term approximations, $n = 1$. (b) Increase in approximation error as the number of flows becomes large, $n = 50, 150$.

of X and Δt. Keeping only the leading term, average efficiency e is given by

$$e = \frac{(k+1)(k+2)\sigma U^{l-1}}{(k+2)(k+1)\sigma U^{l-1}} = 1. \tag{31}$$

The result in (31) suggests that link utilization of binomial algorithms is 100% regardless of the values of k and l even though it is clearly incorrect (e.g., AIMD never achieves 100% utilization unless $\beta = \sigma = 0$). To examine how close this equation models reality, we plotted in Fig. 4(a) the *exact* average efficiency computed from (28) and the relative error between the exact value and the one derived from (31) for a single-flow case (i.e., $U = C$). As the figure shows, for values of l below 0.5, the efficiency is very high (close to 100%) and the corresponding error is negligible. However, as l approaches 1.0, the error increases to 31%. The figure is plotted using $k = 1/2$, $W = 100$, and $\beta = 1/2$. The results do not change significantly for different values of k: the error for $l = 1$ varies monotonically from 35% for $k = -0.5$ to 28% for $k = 1$.

Since Bansal *et al.* [2] use schemes with $l \leq 0.5$ (i.e., IIAD and SQRT), their analysis does not suffer from significant round-off errors given a small number of flows n (see below for the discussion of larger n). Having established that the leading term alone in (30) is insufficient to achieve a good approximation to the exact formula for values of l close to 1, we next examine a two-term approximation to (28)

$$e = 1 - \frac{\sigma U^{l-1}}{2 - k\sigma U^{l-1}}. \tag{32}$$

To perform a self-check, we plug AIMD parameters ($l = 1$, $k = 0$) into (32) and get a familiar (and exact) formula of the average efficiency of an AIMD scheme: $e = (2-\beta)/2$ (recall that $\sigma = \beta$ in AIMD). The result of comparing (32) with (28) is also shown in Fig. 4(a). Depending on the value of l, the error of the two-term approximation (32) is between 5 and 1000 times smaller than that of the single-term approximation (31).

It is interesting that (32) depends on the value of U, which means that efficiency e will depend on the number of flows sharing a link. Consider a case of n flows instead of one flow (i.e., $U = C/n$). It is easy to notice that controllers with multiplicative decrease (i.e., $l = 1$) are the *only* schemes in which the efficiency does not change regardless of the number of flows, because $l - 1$ in (32) is zero. However, the rest of binomial

schemes with $l < 1$ suffer a reduction in efficiency as can be seen by using $U = C/n$ in (32)

$$e_n = 1 - \frac{\sigma U^{l-1}}{2n^{l-1} - k\sigma U^{l-1}}. \tag{33}$$

For larger n, term $2n^{l-1}$ becomes smaller, because l is less than 1. Consequently, both the denominator in (33) and e_n become smaller for larger n. This is illustrated in Fig. 4(b). The top two curves in the figure show how the exact efficiency changes with l for two cases of n equal to 50 and 150 (the curves marked e_{50} and e_{150}, respectively). As the figure shows, for large n, the efficiency of schemes with $l < 1$ suffers a drastic reduction compared to similar cases with fewer flows in Fig. 4(a). In addition, when n approaches its upper limit $W/\beta^{1/(1-l)}$ in (29) (which is 158 for $l = -0.5$), the performance of all schemes with $l < 1$ becomes worse than that of AIMD.

The two bottom curves in Fig. 4(b) show the amount of accumulated error when using one-term and two-term approximations for $n = 150$. The amount of error in the single-term approximation is substantially increased compared to that when n was 1 and stays between 30% and 65%. This was expected, because z in (30) becomes close to 1.0 as n becomes large and the higher order terms are no longer insignificant.

In practice, it may prove to be difficult to fine-tune binomial schemes with $l < 1$ to achieve *consistent* performance over links of all capacities C, because the average efficiency of these schemes will depend on both the number of flows n sharing the link and capacity C, none of which are usually known in advance.

We should also mention that binomial schemes with $l < 1$ possess limited scalability and can support a large number of flows n only when C is very large or σ is very small as seen in (29). When n exceeds $C/\sigma^{1/(1-l)}$, such schemes will be forced to reduce their rates to zero (or some minimum rate $x_{\min}$) upon each packet loss, which is clearly not very efficient and can be achieved with simpler methods. Consequently, we find that *AIMD may be the best solution for heterogeneous links of the current Internet*, because its average efficiency does not change from link to link or when users join and leave a particular end-to-end path.

In addition, as we will see in the next section, a more serious problem with schemes utilizing $l < 1$ is that their packet-loss rates increase at a faster pace than that in AIMD as the number of flows n becomes large.

D. Packet Loss

The amount of packet loss during the steady state is another important property of a congestion control scheme. Consider one oscillation cycle between points A and B in Fig. 3(b) and the case of a single flow. The maximum amount of overshoot under nonideal (i.e., discrete) conditions will be the value of the increase function just before the flow reaches its upper boundary C in point B. Hence, the amount of the maximum overshoot for a single flow is given by $\lambda C^{-k} R$, where R is the fixed duration between control actions as discussed before. Knowing how many bits X were sent by the flow during the interval of duration Δt, we can write the average percentage of lost data p_1 using (27) and assuming the worst case of a maximum overshoot as

$$\begin{aligned} p_1 &= \frac{\lambda C^{-k} R}{X + \lambda C^{-k} R} \\ &= \frac{\lambda^2 (k+2)}{C^{2k+2}\left(1 - (1 - \sigma C^{l-1})^{k+2}\right) + \lambda^2 (k+2)} \\ &\approx \frac{\lambda^2 (k+2)}{C^{2k+2}\left(1 - (1 - \sigma C^{l-1})^{k+2}\right)} \end{aligned} \tag{34}$$

when $\lambda C^{-k} R \ll X$. In particular, for AIMD schemes, the average packet-loss rate given the maximum overshoot is

$$p_1 = \frac{2\lambda^2}{C^2\sigma(2-\sigma) + 2\lambda^2} \approx \frac{2\lambda^2}{C^2\sigma(2-\sigma)}. \tag{35}$$

A closer look at the last equation reveals that as the number of flows increases (i.e., C is replaced by C/n), AIMD's packet-loss rate will also increase. Furthermore, the amount of increase is proportional to n^2, where n is the number of flows. This confirms a well-known fact that AIMD scales as n^2 when it comes to packet loss [38]. Note that as $n \to \infty$, the amount of overshoot $\lambda C^{-k} R$ becomes large compared to the value of X, and the approximations above no longer work. However, the exact formulas in (34) and (35) asymptotically approach the correct value of 100%.

Consider a simple explanation of why AIMD scales quadratically. In AIMD, the increase in packet loss by a factor of n^2 comes from two places—from the reduction of both the number of discrete increase steps N between points A and B in Fig. 3(b) and interval Δt by a factor of n (because increase distance U–L becomes n times smaller). As a result, the number of bits sent during the interval (which is proportional to $N\Delta t$) is reduced by a factor of n^2, and the amount of overshoot is unchanged (i.e., λR). Consequently, the total amount of lost packets relative to the number of sent packets is increased by a factor of n^2.

For example, according to these results, a single AIMD flow may experience 0.1% packet loss over a given link. When 10 flows start sharing the same link, the loss will (theoretically) increase by a factor of 100, reaching 10%. When 100 flows are sharing the link, packet loss will (again in theory) approach 100%. There are two reasons why we do not see this kind of performance degradation in practice. First, our results in (35) are based on a continuous fluid model, which assumes that packets are infinitely divisible. However, in practice, this approximation is true only when the amount of increase λR is negligible compared to the difference between the upper and lower limits, i.e., U–L in Fig. 3(b). Hence, when the number of *discrete* increase steps N becomes equal to 1 (or approaches 1), it can no longer be reduced by a factor of n, because it must remain an integer. Taking into account values of N close to 1, the increase in packet loss becomes a subquadratic (often linear) function of n.

Second, most protocols employing AIMD rely on positive acknowledgment (ACK) in implementing congestion control. This self-clocking [15], or packet conservation, is capable of significantly improving the scalability aspects of AIMD, because the

sender does not inject more packets into the network than the network can handle at any given time. Thus, window-based flow control coupled with exponential timer backoff significantly increases stability of TCP-like protocols. On the other hand, rate-based congestion control does not have this nice cushion to fall back on, which explains why *rate-based* AIMD schemes suffer a much higher packet-loss increase than equivalent window-based schemes (see the experimental results given later in this paper).

V. Packet-Loss Scalability of Congestion Control

A. Overview

Rapid packet-loss increase of AIMD, discussed at the end of the previous section, is bad for rate-based protocols in general and Internet video streaming in particular. However, the situation is not as gloomy as it appears at first since our analysis later in the paper leads to the existence of end-to-end controls with constant packet loss and much nicer properties than AIMD.

Suppose the average packet loss when n flows share a link of capacity C is given by p_n. Let *packet-loss increase factor* s_n be the ratio of p_n to p_1. Parameter s_n specifies how fast packet loss increases when more flows share a common link and directly relates to the ability of the scheme to support a large number of flows (i.e., schemes with lower s_n scale better). Using (34)

$$p_n \approx \frac{\lambda^2(k+2)n^{2k+2}}{C^{2k+2}\left(1-\left(1-\sigma\left(\frac{C}{n}\right)^{l-1}\right)^{k+2}\right)} \tag{36}$$

and applying a two-term approximation from (30)

$$\begin{aligned} s_n &\approx \frac{n^{2k+2}\left(1-\left(1-\sigma C^{l-1}\right)^{k+2}\right)}{\left(1-\left(1-\sigma\left(\frac{C}{n}\right)^{l-1}\right)^{k+2}\right)} \\ &\approx \frac{n^{l+2k+1}\left(2-(k+1)\sigma C^{l-1}\right)}{2-(k+1)\sigma\left(\frac{C}{n}\right)^{l-1}} \\ &= O\left(n^{l+2k+1}\right). \end{aligned} \tag{37}$$

Hence, packet-loss increase factor s_n of binomial algorithms is proportional to n^{l+2k+1} for small n (when overshoot $\lambda C^{-k}R$ is small compared to X) and grows no faster than n^{l+2k+1} for the rest of n. For AIMD, we get the familiar scalability formula of $O(n^2)$, whereas IIAD ($k=1$, $l=0$) and SQRT ($k=l=1/2$) scale as $O(n^3)$ and $O(n^{2.5})$, respectively.

Among *TCP-friendly* schemes (i.e., $k+l=1$), packet-loss increase s_n is $O(n^{3-l})$, which means that TCP-friendly schemes with the *largest* l scale best. Since we already established that l must be no more than 1 (the noncrossover condition), we immediately arrive at our second major result—*among TCP-friendly binomial schemes, AIMD scales best*.

Now we should make several observations about the applicability of (37) in practice. First, we assumed in (34) that the overshoot will always be as large as possible, i.e., $\lambda U^{-k}R$. However, in many cases the actual overshoot will be some r andom value distributed between zero and $\lambda U^{-k}R$. Second, recall our discussion of AIMD's scalability in the previous section. When the increase distance U–L becomes small compared to the value of the increase step, AIMD starts scaling as a linear function rather than a quadratic function. Hence, (37) is accurate only when the increase steps are small compared to C/n. The results based on the above model can be further skewed, if $\lambda U^{-k}R$ becomes large compared to X, in which case we must use the exact formula in (34).

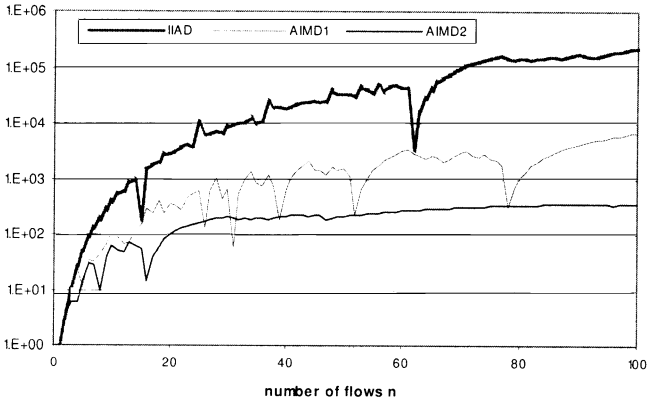


Fig. 5. Packet-loss scalability s_n of AIMD and IIAD in simulation (note the log scale of the y axis).

B. Simulation

To verify packet-loss scalability findings above and show some examples, we present simulation results of rate-based AIMD(1, 1/2) and IIAD(1, 1/2) over a T1 bottleneck link (i.e., $C = 1544$ kb/s). For AIMD, we set $\lambda = \text{MTU}/\text{RTT}$ at two constant values of 5 and 50 kb/s (the corresponding schemes are called AIMD_1 and AIMD_2) to show how their scalability changes when λ becomes large compared to the upper boundary $U = C/n$. For IIAD, we selected $\text{MTU}/\text{RTT} = 10$ kb/s to allow the scheme to maintain $p_n \ll 100\%$ (otherwise, IIAD loses its n^3 packet-loss increase). We used a discrete event simulation, in which n flows of the same type shared a common link under the standard assumption of immediate and synchronized feedback. Furthermore, the simulated flows did not use retransmission or other error control to recover lost packets.

Fig. 5 shows parameter s_n (based on the *actual* rather than the *maximum* overshoot) during the simulation as a function of n for the three types of flows. In AIMD_1 (the curve in the middle), packet-loss increase ratio s_{100} reaches a factor of 6755, which is equivalent to scalability of $n^{1.91}$ (just below predicted n^2). On the other hand, AIMD_2 (the bottom curve) maintains its quadratic packet-loss increase only until $n = 7$, at which time it switches to a linear increase. AIMD_2 reaches an increase factor of $s_{100} = 352$, which is equivalent to an overall scalability of $n^{1.27}$. It may seem at first that the larger increase step λ of AIMD_2 is better; however, due to a larger λ, AIMD_2 is much more aggressive in searching for bandwidth and suffers more packet loss than AIMD_1 for all values of n. Thus, for example, for $n = 100$, AIMD_2 loses 55% of all sent packets, while AIMD_1 loses only 10%.

Finally, IIAD's scalability performance is much worse than that of either of AIMD schemes, as can be seen in the figure (the top curve). Packet loss with 100 flows p_{100} is 219 889 times larger than packet loss with one flow p_1. Analysis of the data

using a least-squares fit to the curve shows that the overall scalability of IIAD is approximately $n^{2.67}$ (again, slightly less than predicted n^3).

The dips in Fig. 5 occur when capacity C happens to be a multiple of λn for AIMD (a similar situation holds for IIAD) and the amount of overshoot is significantly less than the average. As pointed out before and as shown in Fig. 5, the *actual* increase in packet loss under nonideal (i.e., discrete) conditions may be lower than that predicted by (37). Nevertheless, the theoretical result in (37) can be used as a good performance measure in comparing the scalability of different binomial schemes.

C. Feasibility of Ideal Scalability

We next examine ideal scalability of binomial schemes and derive its necessary conditions. We define a scheme to have *ideal scalability* if s_n is constant for all n. This definition is driven by the fact that no matter how small packet loss p_1 is, there will be a link of sufficient capacity that will accommodate such large number of concurrent flows n that p_n will be unacceptably high. This is especially true given the no-better-than-quadratic scalability of binomial congestion control. Consequently, the ideal situation is to have a scheme that maintains a consistent packet-loss rate regardless of the number of flows utilizing the scheme, i.e., $p_n = p_1$ for all n. Furthermore, although not necessary, it would be desirable to have a scheme that maintains the *same* packet loss over links of *different* capacity C.

To show the existence of ideal scalability, we examine (37) again and find congestion controllers that allow s_n to remain constant. The necessary conditions for ideal scalability are (recall that the second equation is needed for convergence to fairness)

$$\begin{cases} l + 2k + 1 = 0 \\ k + l > 0 \end{cases} \Rightarrow \begin{cases} k < -1 \\ l > 1. \end{cases} \tag{38}$$

The $l > 1$ condition means that if we plan to satisfy the noncrossover conditions (19) and (24), or prevent the scheme from reducing its rate below zero, ideal scalability requires the knowledge of some tight upper limit on sending rate x [see the discussion following (24)]. Consequently, only by assuming that x is limited by a constant is it possible to find such σ that will satisfy the necessary condition $\sigma < 1/lx^{l-1}$, $l > 1$ in (24) for all rates x. Hence, we come to our third major result: *among binary-feedback congestion control schemes, ideal scalability is possible only when sending rates x are limited from above by a constant, i.e., when flows have knowledge of the bottleneck capacity*.

There are two possible ways that an application can learn the value of C—by using real-time end-to-end measurements or by asking the network to provide an explicit feedback with the value of C. In the next section, we examine the viability of applying the former method to sampling the capacity of the bottleneck link and the possibility of using such estimates in ideally scalable congestion control. Even though the latter method is similar in spirit to the explicit rate (ER) service in ATM ABR [4], [22], [31], [39], it differs from the proposed work in its complexity. Most ATM ER controllers inside the switch compute a differential equation that is a function of the router's queue size (and sometimes other parameters) and feed back the *available* bandwidth to the end flows. In our approach, the router only needs to feed back the *bottleneck* bandwidth (i.e., line speed of the outgoing link), which is a more practical approach in the end-to-end context than the methods proposed for ATM.

Note that *all* flows sharing a single link must receive an estimate of C that is fairly close to the true capacity of the link. A major drawback of employing congestion control that relies on end-to-end estimates of C is that different flows may form a different estimate, which may lead to poor convergence and/or scalability depending on the amount of error. Hence, our approach in this section relaxes one condition (i.e., $l \leq 1$) but imposes a new one—all flows must measure the bottleneck capacity with high *consistency* and reasonable *accuracy*. Note that a thorough evaluation of various bandwidth estimation methods for ideally scalable congestion control is the topic of ongoing research. In Section VI, we briefly show some of our preliminary results.

To be fair, we must mention another drawback of ideal scalability—typically slower convergence to fairness due to its less aggressive probing for bandwidth and nonmonotonic convergence to fairness. Nevertheless, we believe that ideally scalable controllers present an interesting dimension to congestion control. We investigated ideally scalable congestion control until we established a working version of the algorithm, which we present in the remainder of this paper. Note that much more work in this area is required before we can recommend binary-feedback congestion control other than AIMD for practical use over the Internet.

D. Ideally Scalable Congestion Control

In this section, we introduce ideally scalable congestion control (ISCC) and show how bottleneck capacity C can be used in ISCC to adapt values of (λ, σ) to each end-to-end path. We use notation ISCC(k, l) to refer to ideally scalable schemes described in this section with powers k and l. Note that other ways (not covered by ISCC) of selecting (λ, σ) may be possible to achieve the same goal of constant s_n.

Assuming that $C(t)$ is the current estimate of the bottleneck capacity and that sending rate $x(t)$ is limited by $C(t)$ at all times t, condition $\sigma < 1/lx^{l-1}$ in (24) can be satisfied by choosing the following σ:

$$\sigma = \frac{1}{m_D C(t)^{l-1}} \tag{39}$$

where $l > 1$ and m_D is some constant greater than or equal to l.[5] It is easy to show that the decrease step of schemes with σ according to (39) is no more than x/m_D for any given state $x > 0$. Hence, rate x is guaranteed to stay positive at all times. Furthermore, by varying constant m_D, the scheme can adjust its average efficiency, where larger values of m_D mean higher efficiency.

In addition, we must carefully select the value of λ so that the negative value of power k does not cause uncontrollably high increase steps. One way to achieve this is to select a fixed

[5]From now on, without losing any functionality or convergence, we allow σ to be equal to $1/(lx^{l-1})$, in which case the system may *touch* the fairness line, but not cross it.

value α and then multiply it by $(\mathrm{MTU}/\mathrm{RTT})^{k+1}$ as shown in (4). However, the increase steps will still remain virtually unlimited, because the value of MTU/RTT has little relationship to the value of C (which is needed to effectively limit λx^{-k}). In addition, different flows may use different multiplicative factors in (4) due to the differences in the RTT or the MTU. An alternative approach is to apply a similar thinking to that used before in selecting σ—choose λ so that the increase step is always no more than x/m_I for any given rate x, where m_I is some constant greater than or equal to one. This can be written as

$$\lambda x^{-k} \leq \frac{x}{m_I}, \qquad m_I \geq 1,\ x > 0 \tag{40}$$

which is satisfied with the following choice of λ:

$$\lambda = \frac{C(t)^{k+1}}{m_I}, \qquad m_I \geq 1. \tag{41}$$

Parameter m_I can be used to vary the aggressiveness of the scheme in searching for new bandwidth, where larger values of m_I result in less aggressive behavior of the scheme. Combining (34), (39), and (41), packet loss of ISCC schemes no longer depends on n or C:

$$p_1 = \frac{k+2}{m_I^2\left(1-\left(1-\frac{1}{m_D}\right)^{k+2}\right)+k+2}. \tag{42}$$

In the next section, we compare the performance of one particular ISCC congestion control scheme in a rate-based real-time streaming application with that of IIAD, SQRT, AIMD, and TCP-friendly rate control (TFRC) [11].

VI. Simulation and Experiments

A. Choice of Powers Functions

We start with an observation that if l becomes much larger than 1.0 in an ISCC scheme and sending rate x is much smaller than capacity C (e.g., when n is large), such congestion control becomes less responsive to packet loss. Being less responsive usually results in very small rate reductions that often cannot alleviate congestion in a single step. Thus, schemes with large l usually need multiple back-to-back decrease steps to move the system below the efficiency line in Fig. 2. Our assumptions above do not model this behavior. The actual resulting packet loss in these schemes turns out to be higher than predicted by (37) and the convergence time is sometimes substantially increased. Hence, from this perspective, larger values of l are not desirable.

The only value of l that guarantees ideal scalability among TCP-friendly schemes (i.e., $k+l=1$ and $l+2k+1=0$) is quite high and equals 3. In practice, this scheme converges very slowly and may not be a feasible solution for the real Internet. Among non-TCP-friendly schemes, values of l close to 1.0 force k to come close to -1.0 (because $l+2k+1$ must still remain zero), which also results in slower convergence to fairness as sum $k+l$ approaches zero.[6]

[6]Values of $k+l$ close to zero mean that the system makes very small (if any) steps toward the fairness line and, thus, converges very slowly.

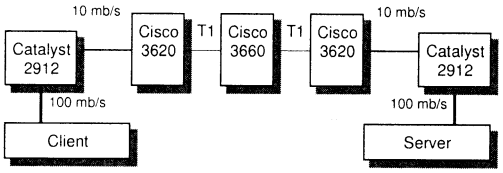


Fig. 6. Setup of the experiment.

Among an infinite number of ISCC schemes, we arbitrarily selected a scheme with $l=2 \quad (k=-1.5)$ which achieves a reasonable compromise between packet loss and convergence, and we show its performance in this paper. Note that this particular scheme is somewhat less aggressive that AIMD and typically yields bandwidth to AIMD (however, this effect becomes noticeable only when the number of flows n is large). Hence, practical application of this ISCC scheme in the Internet requires the use of new QoS methods in routers (i.e., DiffServ) as discussed in Section I.

For $l=2$, the analytical result in (38) suggests a value of k equal to -1.5. However, in simulation, we found that $k=-1.2$ was sufficient to maintain constant packet loss. In fact, $k=-1.5$ appeared to be "too successful" in controlling the combined packet loss of all flows and resulted in a slowly *decaying* function s_n (i.e., the more flows, the lower packet loss p_n). Whether this observation holds in a real network is the topic of our investigation below.

B. Real-Time Bandwidth Estimation

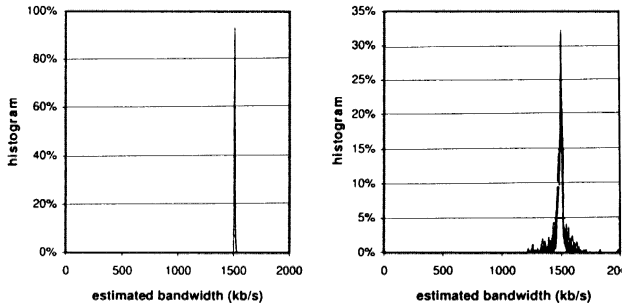
In this section, we briefly examine the accuracy of real-time bandwidth estimation in rate-based streaming applications. In the next section, we show the performance of two ISCC schemes that rely on these real-time estimates for computing the values of λ and σ.

We used a Cisco network, depicted in Fig. 6, for all real-life experiments in this paper. The server and the client were connected to Catalyst switches via two 100-Mb/s Ethernets. The switches in turn were connected to Cisco 3620 routers via 10-Mb/s Ethernets. The 3620 routers connected to each other via T1 links passing through an additional Cisco 3660 router. During the experiment, we disabled weighted random early detection (WRED) and weighted fair queueing (WFQ) on all T1 interfaces to reflect the current setup of backbone routers.

The server supplied the client with real-time bandwidth-scalable MPEG-4 video, which included an fine-granular scalable (FGS) enhancement layer [42] and a regular base layer. At any time t, the server was able to adapt its streaming rate to the rate requested by the client, as long as it was no less than the rate of the base layer b_0 and no more than the combined rate of both layers.

We used a 10-min MPEG-4 video sequence with the base layer coded at $b_0=14$ kb/s and the enhancement layer coded up to the maximum rate $R_{\max}=1190$ kb/s. Note that two concurrent flows were needed to fully load the bottleneck link. Hence, our experiments below do not cover the case of $n=1$, and s_n is defined as the ratio of p_n to p_2.

During the experiment, the client applied a simple packet-bunch estimation technique [5], [41] to the server's video packets. To simplify the estimation of the bottleneck



(a) (b)

Fig. 7. The histogram of bandwidth estimates with (a) 2 and (b) 32 AIMD(1, 1/2) flows over a shared T1 link.

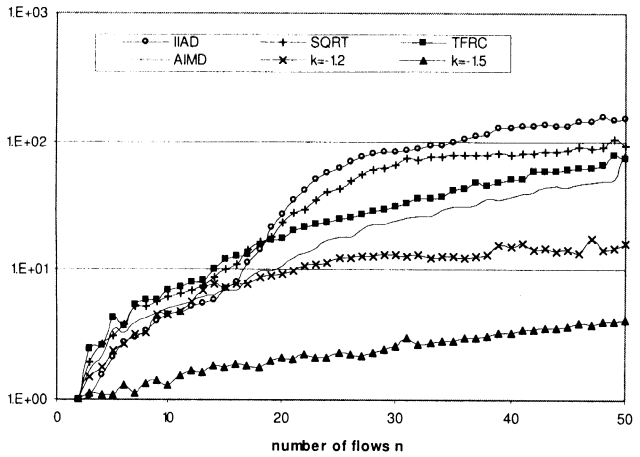


Fig. 8. Packet-loss increase factor s_n for the Cisco experiment.

bandwidth, the server sent its packets in bursts of a predefined length. A bandwidth sample was derived from each burst that contained at least three packets.

To establish a baseline performance, Fig. 7(a) plots the histograms of IP bandwidth estimates obtained by two AIMD(1, 1/2) flows over the topology in Fig. 6 (both flows used a fixed value of $\lambda = \text{MTU}/\text{RTT}$ equal to 30 kb/s). As the figure shows, the flows measured the IP bottleneck bandwidth to be 1510 kb/s, which is very close to the actual T1 rate of 1544 kb/s (the discrepancy is easily explained by the data-link/physical layer overhead on the T1 line). Furthermore, both flows were in perfect agreement, and 99.5% of estimates of each flow were between 1500 and 1520 kb/s.

Fig. 7(b) shows the histograms of bandwidth estimates obtained by 32 simultaneous AIMD(1, 1/2) flows running over the same topology and with the same value of MTU/RTT. This time, the majority of estimates lie in the proximity of 1490 kb/s, and 95.5% of estimates are contained between 1400 and 1620 kb/s (i.e., within 7% of 1510 kb/s). The lower accuracy of bandwidth estimation in the second case is explained by the lower average sending rate of each flow (i.e., 36 kb/s compared to 559 kb/s in the first case) and higher overall packet loss.

C. Scalability Results

In our rate-based streaming application, all flows used slow start at the beginning of each transfer; however, the results below exclude the behavior of the network during the transient phase and focus on the steady-state performance of the schemes in the interval starting 10 s after the last flow finished its slow start and ending when the first flow terminated.[7] This interval was 520–600 s long (depending on the number of flows) and included a combined transfer of approximately 60 000 packets.

During the experiment, we tried to select the parameters of the schemes so that the average packet loss of two competing flows using each scheme was between 0.3% and 0.6%. Using a fixed value of $\lambda = \text{MTU}/\text{RTT}$ equal to 100 kb/s, this constraint resulted in selecting the following parameters: AIMD(0.19, 1/2), SQRT(0.18, 1/2), and IIAD(0.10, 1/2). The value of the *MTU variable* in TFRC's equation [11] was selected to be 180 bytes, whereas the actual packet size used during the experiment was 1500 bytes for all schemes. Note that TFRC was the only protocol that used real-time measurements of the RTT in its computation of the rate.

The efficiency and aggressiveness parameters of the ISCC schemes were set with the same goal in mind to maintain low initial packet loss p_2: $m_D = 2$ and $m_I = 20$. These parameters guaranteed that each flow did not decrease its rate by more than 1/2 and did not probe for new bandwidth more aggressively than by 5% (i.e., 1/20) of the current sending rate.

The results of the experiment are summarized in Fig. 8, which shows packet-loss increase factor s_n for six different schemes and values of n between 2 and 50. The results of the experiment show that all nonscalable schemes maintained a steady packet-loss increase to well over 15%. For example, IIAD reached $p_{50} = 45\%$, SQRT 28%, AIMD 25%, and TFRC 21%.

Recall that our simulation suggested that $k = -1.2$ was adequate for the $l = 2$ schemes. In practice, however, the value of $k = -1.2$ turned out to be insufficient to balance out the large positive value of l in the scalability power $l+2k+1$, as shown in Fig. 8. Contrary to simulation results, ISCC(-1.2,2) maintained a steady packet-loss increase for all values of n and reached a reasonably high loss rate $p_{50} = 12\%$. The overall scalability of the scheme using a least-squares fit of a power function was found to be $n^{0.85}$.

On the other hand, packet loss of ISCC(-1.5,2) climbed only to 2.3% over the same range of flows n. A least-squares fit suggests that the increase in packet loss was slow but noticeable. Even though ISCC(-1.5,2) was not able to achieve constant packet loss in practice, it did show a substantially better performance than any other scheme. The discrepancy between the continuous fluid model for ISCC studied earlier in this paper and experimental results is easily explained by delayed and lost feedback, nonuniform packet loss, and errors in measuring the value of C.

As expected, high packet loss during the experiment resulted in a large number of underflow events, which are produced when a frame is missing from the decoder buffer at the time of its decoding. Recall that underflow events in the base layer are most severe, because each one of them results in a "freeze-frame"

[7]Flows were started with a 1.5-s delay.

TABLE I
SUMMARY OF EXPERIMENTAL RESULTS

Scheme	p_2	p_{50}	s_{50}	Scales as	Efficiency
IIAD(0.10,1/2)	0.29%	44.7%	156.9	$n^{1.98}$	74%
SQRT(0.18,1/2)	0.26%	27.8%	105.8	$n^{1.60}$	85%
AIMD(0.19,1/2)	0.30%	25.5%	86.3	$n^{1.31}$	75%
TFRC(180)	0.27%	20.9%	77.8	$n^{1.30}$	92%
ISCC(−1.2,2)	0.69%	12.0%	17.4	$n^{0.85}$	80%
ISCC(−1.5,2)	0.57%	2.3%	4.1	$n^{0.49}$	77%

effect. If an enhancement frame is not received, or is received partially by the time of its decoding, such underflow events are not severe, because the base layer can be played without the enhancement layer or even with partially received enhancement-layer pictures.

Under the worst conditions (i.e., $n \approx 50$), our data show that the nonscalable protocols maintained a "frozen" picture between 13% and 68% of the corresponding session. These results indicate that high packet loss is very harmful to video applications even in the presence of low RTTs (50–200 ms), large startup delays (3 s in our case), and an efficient packet-loss recovery mechanism (our retransmission scheme was able to recover all base-layer packets before their deadlines until loss rates exceeded approximately 15%).

On the other hand, ISCC(−1.2,2) recovered all base-layer frames before their deadlines, but had some missing enhancement-layer frames. ISCC(−1.5,2) recovered all base-layer and all enhancement-layer pictures before their decoding deadlines, representing an ideal streaming situation for the end user.

Table I summarizes our observations during the experiment and further shows that both ISCC schemes maintained a reasonably high average efficiency (for the comparison to be meaningful, we selected the value of n when all flows achieved approximately the same packet loss, i.e., $n = 2$). The table shows that ISCC's efficiency was in fact better than that of both AIMD and IIAD.

We further experimented with random early detection (RED) and studied the effect it had on the performance of rate-based binomial controllers. In a separate set of experiments over the same topology, we enabled Cisco's WRED on all T1 interfaces and used the default parameters suggested by Cisco Internetwork Operating System (IOS). Interestingly, RED did not have any major impact on packet-loss increase curves in Fig. 8 or the average efficiency. However, we did notice a slight improvement in fairness between the individual flows. Little overall effect of Cisco's WRED on congestion control and/or link utilization has also been noticed in [8] and [34].

In this section, we found that traditional schemes (including AIMD, IIAD, SQRT, and TFRC) are poorly suited for rate-based protocols that do not utilize self-clocking. Furthermore, we observed that ideally scalable schemes promise to provide a constant packet-loss scalability not only in simulation but also in practice. Nevertheless, further study is required in this area to understand the tradeoffs between the different values of l and k, as well as establish whether slower convergence to fairness found in simulation has any strong implications in large networks (i.e., in the real Internet).

VII. CONCLUSION

In this paper, we studied the binary-feedback class of rate-based congestion control and derived very simple conditions on increase and decrease functions that guarantee monotonic convergence to fairness (i.e., asymptotic stability of the fairness point). Interestingly, our derivations show that AIMD is the only binomial controller with monotonic convergence to fairness. We also showed that all binomial controls with a nonmultiplicative decrease function suffered from reduced link utilization as the number of flows n increased.

In the second half of this paper, we studied the origin of significant packet-loss increase in rate-based binomial schemes as the number of flows becomes large. This phenomenon is caused by the reduction of the fair-share bandwidth allocated to each flow by a factor of n and unchanged (or even greater) overshoot of this fair bandwidth during the increase phase. One implication of this result for ISPs is that they should scale their bandwidth proportionally to the number of users (flows) that their networks support.

To overcome rapid packet-loss increase, we developed and studied a new class of ideally scalable controllers (ISCC), which keep the amount of overshoot proportional to the amount of data sent by a flow during each oscillation cycle. Even though ISCC offers much better packet-loss characteristics under a variety of simulation and real-life scenarios, its requirement for end flows to measure the bottleneck bandwidth and slower convergence for large n leave room for future research.

Among non-ISCC binomial schemes, our conclusion is that linear controls of AIMD offer the most robust behavior across a wide range of paths, lowest packet loss, and fastest convergence to fairness under a variety of conditions. Non-AIMD binomial schemes may possess a certain level of appeal (such as smoother rates); however, their use in rate-based applications leads to rapid packet-loss increase, which may turn out to be a serious drawback in practical implementations.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for providing excellent suggestions and comments.

REFERENCES

[1] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," IETF, RFC 2581, Apr. 1999.
[2] D. Bansal and H. Balakrishnan, "Binomial congestion control algorithms," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 631–640.
[3] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic behavior of slowly-responsive congestion control algorithms," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 263–274.
[4] L. Benmohamed and S. M. Meerkov, "Feedback control of congestion in packet switching networks: The case of a single congested node," *IEEE/ACM Trans. Networking*, vol. 1, pp. 693–708, Dec. 1993.
[5] R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in packet switched networks," *Int. J. Perform. Eval.*, pp. 27–28, 1996.
[6] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comput. Networks ISDN Syst.*, vol. 17, pp. 1–14, 1989.
[7] S. Chong, S. Lee, and S. Kang, "A simple, scalable, and stable explicit rate allocation algorithm for max–min flow control with minimum rate guarantee," *IEEE/ACM Trans. Networking*, vol. 9, pp. 322–335, June 2001.
[8] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for web traffic," in *Proc. ACM SIGCOMM*, Aug. 2000, pp. 139–150.

[9] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 905–914.
[10] K. W. Fendick, M. A. Rodriguez, and A. Weiss, "Analysis of a rate-based feedback control strategy for long haul data transport," *Perform. Eval.*, vol. 16, pp. 67–84, 1992.
[11] S. Floyd, M. Handley, and J. Padhye, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM*, Sept. 2000, pp. 43–56.
[12] S. Floyd, M. Handley, and J. Padhye. (2000, May) A comparison of equation-based and AIMD congestion control. [Online]. Available: http://www.aciri.org/tfrc/aimd.pdf
[13] C. Fulton, S. Q. Li, and C. S. Lim, "An ABR feedback control scheme with tracking," in *Proc. IEEE INFOCOM*, Apr. 1997, pp. 805–814.
[14] O. C. Imer, S. Compans, T. Basar, and R. Srikant, "ABR congestion control in ATM networks," *IEEE Control Syst. Mag.*, vol. 21, pp. 38–56, 2001.
[15] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM*, 1988, pp. 314–329.
[16] R. Johari and D. Tan, "End-to-end congestion control for the Internet: Delays and stability," *IEEE/ACM Trans. Networking*, vol. 9, pp. 818–832, Dec. 2001.
[17] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore, "The ERICA switch algorithm for ABR traffic management in ATM networks," *IEEE/ACM Trans. Networking*, vol. 8, pp. 87–98, Feb. 2000.
[18] K. Kar, S. Sarkar, and L. Tassiulas, "A simple rate control algorithm for maximizing total user utility," in *Proc. IEEE INFOCOM*, 2001, pp. 133–141.
[19] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, pp. 237–252, 1998.
[20] S. Keshav, "A control-theoretic approach to flow control," in *Proc. ACM SIGCOMM*, 1991, pp. 3–15.
[21] T. Kim, S. Lu, and V. Bharghavan. (1999, July) Loss proportional decrease based congestion control in the future Internet. Tech. Rep. Univ. Illinois, Chicago, IL. [Online]. Available: http://timely.crhc.uiuc.edu/Drafts/tech.lipd.ps.gz
[22] A. Kolarov and G. Ramamurthy, "A control theoretic approach to the design of closed loop rate based flow control for high speed ATM networks," in *Proc. IEEE INFOCOM*, Apr. 1997, pp. 293–301.
[23] L. Kulkarni and S. Q. Li, "Performance analysis of rate based feedback control for ATM networks," *IEEE/ACM Trans. Networking*, vol. 6, pp. 797–810, Dec. 1998.
[24] H. T. Kung, T. Blackwell, and A. Chapman, "Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation, and statistical multiplexing," in *Proc. ACM SIGCOMM*, 1994, pp. 101–114.
[25] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: Utility functions, random losses and ECN marks," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 1323–1332.
[26] T. V. Lakshman, P. P. Mishra, and K. K. Ramakrishnan, "Transporting compressed video over ATM networks with explicit rate feedback control," in *Proc. IEEE INFOCOM*, Apr. 1997, pp. 38–47.
[27] K.-W. Lee, T. Kim, and V. Bharghavan. (2000) A comparison of end-to-end congestion control algorithms: The case of AIMD and AIPD. Tech. Rep. Univ. Illinois, Chicago, IL. [Online]. Available: http://timely.crhc.uiuc.edu/~kwlee/ psfiles/infocom2001.ps.gz
[28] K.-W. Lee, R. Puri, T. Kim, K. Ramchandran, and V. Bharghavan, "An integrated source coding and congestion control framework for video streaming in the Internet," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 747–756.
[29] A. Legout and E. W. Biersack, "PLM: Fast convergence for cumulative layered multicast transmission schemes," in *Proc. ACM SIGMETRICS*, 2000, pp. 13–22.
[30] S. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a scalable control," in *Proc. IEEE INFOCOM*, June 2002, pp. 239–248.
[31] S. Mascolo, D. Cavendish, and M. Gerla, "ATM rate based congestion control using a Smith predictor: An EPRCA implementation," in *Proc. IEEE INFOCOM*, Mar. 1996, pp. 569–576.
[32] L. Massoulié, "Stability of distributed congestion control with heterogeneous feedback delays," *IEEE Trans. Automat. Contr.*, vol. 47, pp. 895–902, June 2002.
[33] L. Massoulié and J. Roberts, "Bandwidth sharing: Objectives and algorithms," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 1395–1403.
[34] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," in *Proc. IEEE/IFIP IWQoS*, June 1999, pp. 260–262.
[35] A. Mena and J. Heidemann, "An empirical study of real audio traffic," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 101–110.
[36] Microsoft Corp. Windows Media Player. [Online]. Available: http://www.microsoft. com/windows/mediaplayer/
[37] P. Mishra and H. Kanakia, "A hop by hop rate-based congestion control scheme," in *Proc. ACM SIGCOMM*, 1992, pp. 112–123.
[38] T. Nandagopal, K.-W. Lee, J. R. Li, and V. Bharghavan, "Scalable service differentiation using purely end-to-end mechanisms: Features and limitations," in *Proc. IFIP/IEEE IWQoS*, June 2000, pp. 31–41.
[39] H. Ohsaki, M. Murata, H. Suzuki, C. Ikeda, and H. Miyahara, "Rate-based congestion control for ATM networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 2, pp. 60–72, Apr. 1995.
[40] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM*, Sept. 1998, pp. 303–314.
[41] V. Paxson, "Measurements and analysis of end-to-end Internet dynamics," Ph.D. dissertation, Comput. Sci. Dept., Univ. California, Berkeley, 1997.
[42] H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen, "Scalable Internet video using MPEG-4," *Signal Processing: Image Commun.*, vol. 15, pp. 95–126, Jan. 1999.
[43] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for real-time streams in the Internet," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 1337–1345.
[44] Real Networks. RealPlayer. [Online]. Available: http://www.real.com
[45] I. Rhee, V. Ozdemir, and Y. Yi, "TEAR: TCP emulation at receivers—Flow control for multimedia streaming," North Carolina State Univ., Raleigh, Tech. Rep., Apr. 2000.
[46] M. Ritter, "The effect of bottleneck service rate variations on the performance of ABR flow control," in *Proc. IEEE INFOCOM*, Apr. 1997, pp. 815–822.
[47] S. Shakkottai and S. Srikant, "How good are deterministic fluid models of Internet congestion control?," in *Proc. IEEE INFOCOM*, June 2002, pp. 497–505.
[48] Y. R. Yang, M. S. Kim, and S. S. Lam, "Transient behaviors of TCP-friendly congestion control protocols," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 1716–1725.
[49] Y. R. Yang and S. S. Lam, "General AIMD congestion control," Univ. Texas, Austin, TX, Tech. Rep., May 2000.
[50] Y. Zhao, S. Q. Li, and S. Sigarto, "A linear dynamic model for design of stable explicit-rate ABR control schemes," in *Proc. IEEE INFOCOM*, Apr. 1997, pp. 283–292.

Dmitri Loguinov (S'99–M'03) received the B.S. degree (with honors) in computer science from Moscow State University, Moscow, Russia, in 1995 and the Ph.D. degree in computer science from the City University of New York, New York, in 2002.

Since September 2002, he has been an Assistant Professor of computer science with Texas A&M University, College Station. His research interests include Internet video streaming, congestion control, image and video coding, Internet traffic measurement and modeling, scalable overlay and peer-to-peer networks, and emerging quality-of-service architectures.

Hayder Radha (M'92–SM'01) received the B.S. degree (with honors) from Michigan State University (MSU), East Lansing, in 1984, the M.S. degree from Purdue University, West Lafayette, IN, in 1986, and the Ph.M. and Ph.D. degrees from Columbia University, New York, in 1991 and 1993, respectively, all in electrical engineering.

He joined MSU in 2000 as an Associate Professor in the Department of Electrical and Computer Engineering. From 1996 to 2000, he was with Philips Research USA, where he initiated the Internet Video project and led a team of researchers working on scalable video coding and streaming algorithms. Prior to joining Philips, he was a Distinguished Member of Technical Staff with Bell Labs, where he worked from 1986 to 1996 in the areas of digital communications, signal/image processing, and broad-band multimedia. His research interests include image and video coding, wireless technology, multimedia communications, and networking. He holds more than 20 patents in these areas. He served as Co-Chair and Editor of the ATM and LAN Video Coding Experts Group of the ITU-T during 1994–1996.

Dr. Radha is a Philips Research Fellow.