CYRF: A Theory of Window-Based Unicast Congestion Control

Nishanth R. Sastry and Simon S. Lam, *Fellow, IEEE*

***Abstract*—This work presents a comprehensive theoretical framework for memoryless window-based congestion control protocols that are designed to converge to fairness and efficiency. We first derive a necessary and sufficient condition for stepwise convergence to fairness. Using this, we show how fair window increase/decrease policies can be constructed from suitable pairs of monotonically nondecreasing functions. We generalize this to smooth protocols that converge over each congestion epoch. The framework also includes a simple method for incorporating TCP-friendliness.**

Well-studied congestion control protocols such as TCP, GAIMD, and Binomial congestion control can be constructed using this method. Thus, we provide a common framework for the analysis of such window-based protocols. We also present two new congestion control protocols for streaming media-like applications as examples of protocol design in this framework: The first protocol, LOG, has the objective of reconciling the smoothness requirement of an application with the need for a fast dynamic response to congestion. The second protocol, SIGMOID, guarantees a minimum bandwidth for an application but behaves exactly like TCP for large windows.

***Index Terms*—Congestion control, fairness, TCP-friendliness, transport protocols.**

I. Introduction

Van Jacobson's congestion control and avoidance mechanisms for TCP [9] have been instrumental for the success and stability of the Internet. Chiu and Jain [5] proved that, assuming synchronous feedback, any additive-increase multiplicative-decrease (AIMD) mechanism such as TCP converges to fairness and efficiency. Thus, these mechanisms allow flows to equitably share a bottleneck link's bandwidth by adjusting their sending rates, while at the same time efficiently utilizing all of the available bandwidth. In abstract terms, TCP can be viewed as a distributed algorithm to fairly and efficiently apportion a common resource *without having to explicitly exchange any information between the users of the resource*. This is arguably the main reason for the success of TCP.

Unfortunately, recent experience indicates that TCP may not be suitable for all applications. For example, TCP's abrupt decrease-by-half response to losses or congestion indications cannot be tolerated by streaming media applications. Consequently, there is a pressing need to design custom congestion control protocols adapted to the needs of different applications. To simplify the task of designing a protocol that not only suits a given situation, but also satisfies the nice property of convergence to fairness and efficiency, we propose a window-based congestion control framework called CYRF (for *C*hoose *Y*our *R*esponse *F*unction). By choosing specific window increase and decrease policies (or response functions), CYRF can be easily adapted to suit different application and network needs and can also be made TCP-friendly.

Our main theoretical result is that given two monotonically nondecreasing functions $f(x)$ and $g(x)$, $f(x) > 0$ and $0 < g(x) \leq 1$ for all $x \geq 1$, a set of flows using the following increase-decrease policy converges to fairness and efficiency

$$\begin{aligned}\mathcal{I} : x(t+R) &\quad\leftarrow\quad x(t) + \frac{x(t)}{f(x(t))}\\ \mathcal{D} : x(t+R) &\quad\leftarrow\quad x(t) - x(t)g(x(t)). \end{aligned}\tag{1}$$

Throughout this paper, we use $x(t)$ to represent the current window size and $x(t+R)$, the next window, after a round-trip time R. We also use $\mathcal{I}$ for the increase policy and $\mathcal{D}$ for the decrease policy.

We term these protocols as *step-wise convergent* because each application of the above policy moves the system closer to fairness. We obtain a more general class of smooth *epoch-wise convergent* protocols called 1-CYRF that allow unfair decrease steps but still converge to fairness over each congestion epoch if we drop the requirement that $g(x)$ be monotonic, and instead only ask that the product $f(x)g(x)$ be monotonically nondecreasing, and always greater than 1 for $x > 1$.

We also introduce a new characterization of TCP-friendliness and show that a smooth CYRF flow will be TCP-friendly in steady state if

$$f(x)g(x) \propto x. \tag{2}$$

Observe from (1) that a slowly increasing function for $f(x)$ results in an aggressive protocol that makes full use of network bandwidth as soon as it becomes available. Similarly a slowly increasing $g(x)$ results in a smoother response to congestion indications. We cannot choose an arbitrarily aggressive increase policy together with a very smooth decrease policy because of the above constraint (2). Thus, there is a continuum of protocols with different *degrees of smoothness*.

An interesting aspect of this work is that all commonly known window-based protocols, namely TCP, GAIMD, and Binomial Congestion Control, are special cases of CYRF (some binomial congestion control protocols are only a special case of 1-CYRF),

Manuscript received March 11, 2003; revised December 4, 2003; approved by IEEE/ACM Transactions on Networking Editor E. Knightly. This work was supported in part by the National Science Foundation under Grant CNS-0434515 and by the Texas Advanced Research Program under Grant 003658-0439-2001. This paper is an extended version of a paper presented at the IEEE International Conference on Network Protocols, Paris, France, November 2002.

N. R. Sastry is with the International Business Machines Corporation, Westford, MA 01886 USA (e-mail: nishanth_sastry@us.ibm.com).

S. S. Lam is with the Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712-0233 USA (e-mail: lam@cs.utexas.edu).

Digital Object Identifier 10.1109/TNET.2005.845545

thus providing a powerful unified framework for the analysis of these protocols. We obtain new proofs for the fairness and TCP-friendliness of these protocols as special cases of the results for CYRF. As a second application of CYRF, we describe a new classification of the space of window-based protocols.

The main application envisaged for CYRF is the design of new window-based protocols to suit different application and network needs. We briefly describe a new 1-CYRF TCP-friendly protocol called LOG that balances the need of streaming media applications for smooth changes in sending rate with the network requirement of a fast response to congestion indications. We use LOG to experimentally demonstrate the validity of the main theorems for CYRF. We also discuss a limitation of window-based TCP-friendly nonlinear protocols in the presence of severely congested (overflowing) drop-tail queues and propose a possible solution—a protocol that behaves just like TCP in the limiting case of large windows, but also guarantees a minimum throughput. We obtain a new protocol called SIGMOID from the desired shape of its response function, using the CYRF paradigm. LOG and SIGMOID are intended to be examples of protocol design in the CYRF framework for specific application and network requirements.

The rest of this paper is organized as follows. In Section II, we describe related proposals. Section III explains some of the simplifying assumptions used in our analysis and derives sufficient conditions for 2 and $n > 2$ flows to converge to fairness and efficiency. This is used in Section IV as the basis for CYRF. Section V provides the TCP-friendliness rule and describes the design of LOG. Section VI experimentally validates the main theorems using LOG and sketches the design of SIGMOID. Section VII concludes the work.

II. Related Work

The TCP algorithms introduced by Jacobson and Karels [1], [9], [23] is the most widely used congestion control mechanism today. In congestion avoidance mode, TCP increases its window size by 1 when a window is acknowledged, and decreases the window to half its previous size when a loss is detected. Thus, its increase and decrease policies are given by

$$\begin{aligned} \mathcal{I} : x(t+R) &\leftarrow x(t) + 1 \\ \mathcal{D} : x(t+R) &\leftarrow x(t) - \frac{x(t)}{2}. \end{aligned} \tag{3}$$

Chiu and Jain [5] proved that, assuming synchronous feedback, any AIMD mechanism such as TCP converges to fairness and efficiency.

The notion of TCP-friendliness [15], [16] has given rise to a number of new proposals [2], [8], [12], [20]–[22], [25] for the transport of streaming multimedia. The closest in approach to CYRF are GAIMD and Binomial Congestion Control, which are both shown to be special cases of CYRF. GAIMD [11], [25], generalizes TCP to an AIMD policy with different increase and decrease parameters

$$\begin{aligned} \mathcal{I} : x(t+R) &\leftarrow x(t) + \alpha \qquad \alpha > 0 \\ \mathcal{D} : x(t+R) &\leftarrow x(t) - \beta x(t) \quad 0 < \beta < 1. \end{aligned} \tag{4}$$

For TCP-friendliness, α and β must be related as follows [7]:[1]

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

Binomial congestion control [2] proposes the following nonlinear increase–decrease policies:

$$\begin{aligned} \mathcal{I} : x(t+R) &\leftarrow x(t) + \frac{\alpha}{x(t)^k} \qquad \alpha > 0 \\ \mathcal{D} : x(t+R) &\leftarrow x(t)\beta x(t)^l \qquad 0 < \beta < 1 \end{aligned} \tag{6}$$

with the further condition that $k + l = 1$ to ensure TCP-friendliness. We can use $l > 1$ only if we know that the maximum window size is $x < (1/\beta)^{l-1}$. Otherwise, we will need to separately deal with the possibility of negative window sizes after an application of $\mathcal{D}$. A similar policy is briefly considered in Section IV of [5]. GAIMD is a special case of the binomial algorithm (6) with $k = 0$ and $l = 1$. $\mathrm{SQRT}(k = l = 0.5)$ and $\mathrm{IIAD}(k = 1, l = 0)$ are two nonlinear binomial controls in [2] that we will use in later sections.

In this work, we do not look at multicast congestion control. We also do not consider application-specific adaptive approaches such as [19] that try to make the best use of the available network support. We consider all flows equal so that schemes like MulTCP [6] fall outside our framework. Similarly, we allow only binary feedback, either through packet loss or an ECN-like indication. Thus, proposals such as Explicit Window Adaptation [13] which requires per-flow network feedback are not considered. Finally, we confine ourselves to the classical memoryless model of congestion control. Thus, recent schemes such as SIMD [12] fall outside our scope.

III. Convergence Requirements

In this section, we first outline the simplifying assumptions made in the rest of the paper and then formalize the notions of convergence to fairness and efficiency.

A. Notation

Following Chiu and Jain [5], in the rest of this work we adopt the following conventions for notation. We use x_i to represent the current window size of the i^{th} flow. Δx_i denotes a change to it due to the application of an increase policy $\mathcal{I}$ or a decrease policy $\mathcal{D}$. In general, the numerical subscript i will be used to denote a quantity on flow i, and the number of flows is represented by n.

B. The Model

The following analysis uses Chiu and Jain's *synchronous feedback* assumption [5] that all the flows in the network get the same feedback and get this feedback simultaneously. Furthermore, the feedback is *binary* and limited to a single bit indicating whether the network is overloaded (1) or if there is additional available bandwidth (0). This feedback can be implicit, for example, through packet losses, or through an explicit mechanism such as a "congestion experienced" bit in an ECN aware network [17], [18].

[1]This does not consider the effect of timeouts. Ref. [25] gives a slightly different condition for TCP-friendliness with timeouts.

If the network feedback is 1, then the next window size is determined by the decrease policy $\mathcal{D}$, otherwise, the increase policy $\mathcal{I}$ is applied. Usually, this adjustment occurs upon receiving an ACK. Here we use the *continuous fluid model* which assumes that this happens as a continuous process.

We also assume a *saturated sender* whose window size is limited only by the network, and not by the receiver's window or the amount of outstanding data at the sender. Finally, we ignore mechanisms such as slow-start by assuming that *steady state* has been reached.

It is important to note that while these assumptions simplify our proofs, they do not restrict the applicability of our results. Section VI-A shows that the results apply even in experimental simulations where the ideal-case assumptions do not hold.

C. 1-Responsiveness

Steady state can be characterized by a sequence of *congestion epochs* which we define as the largest period of time which contains (and ends with) exactly one application of the decrease policy. Most analyses (for example, [2], [7]) implicitly assume that each congestion epoch has at least one application of an increase policy, or equivalently, that each decrease is preceded by at least one increase. With the synchronous feedback assumption, this means that for each flow, the decrease in window size from a single application of $\mathcal{D}$ must at least wipe out the previous increase resulting from the last application of $\mathcal{I}$, so that the next feedback from the network does not indicate overload. In other words, the following criterion must be satisfied:

$$|\Delta(w_{\mathcal{I}})| \leq |\Delta(w_{\mathcal{D}})| \tag{7}$$

where $\Delta(w_{\mathcal{I}})$ is the increase resulting from a single application of $\mathcal{I}$ and $\Delta(w_{\mathcal{D}})$ the decrease in window size because of $\mathcal{D}$. We term protocols which satisfy (7) for *sufficiently large window sizes* as 1-*responsive* to distinguish them from k-*responsive* protocols which require $k > 1$ applications of $\mathcal{D}$ to offset an increase.

Unless otherwise stated, the protocols are assumed to be 1-responsive. As can be seen from (3), (4), and (6), many interesting protocols like TCP, GAIMD, and the TCP-friendly version of Binomial Congestion Control are 1-responsive in general.[2] Thus, this is not a very restrictive assumption.

D. Smoothness

While smoothness is not a "necessary" property, smooth protocols are now being studied with great interest as possible transport protocols for streaming media applications. We will later show how to design new smooth protocols with the useful characteristics of TCP-friendliness and epoch-wise convergence.

Smoothness has been used as a metric in previous work (e.g., [7] and [24]). Below we formalize a slightly different (but compatible) notion of smoothness. Intuitively, the smoother a flow is, the lesser will be its decrease in response to a congestion indication from the network. A window increase (decrease) policy $x_{t+R} \leftarrow x_t + \Delta x$, is said to be smooth if the window size increase (decrease) from a single application of the policy is at least an order of magnitude smaller than the current window size, for large enough windows. Formally, we write

$$|\Delta x| \ll x. \tag{8}$$

In this work, we say a protocol is *smooth* if it has a smooth decrease policy.

[2]Note that the minimum possible window size is 1 and for this window, (7) fails. But we assume a congestion epoch with a sufficiently large minimum window size.

E. Convergence to Efficiency

We require the system to react in such a way as to move the total bottleneck link utilization closer to the link capacity. This can be achieved if the total utilization across all flows (i.e., sum of window sizes) increases when the bottleneck link is under-utilized and decreases when the bottleneck link is overloaded. This is just the principle of negative feedback [5]. An easy way to achieve this is to have each flow increase its window size when the bottleneck link is under-utilized and decrease its window size when the bottleneck link is overloaded.

F. Convergence to Fairness

Fairness is an important criterion for the feasibility of any end-to-end congestion control protocol. Intuitively, this means that regardless of the initial window size values, all flows sharing a single bottleneck link must eventually end up with identical window sizes at each instant (in steady state).

In practice, we use the stronger notions of *step-wise* and *epoch-wise* convergence to fairness. Stepwise convergence requires each application of an increase or decrease policy to move the system closer to fairness. Epochwise convergence requires each congestion epoch to move the system closer to fairness, although individual increase or decrease steps may worsen fairness.

When the eventual goal of equal window sizes is not satisfied, the flows share the link unfairly. To quantify this, we use the Jain–Chiu–Hawe Fairness Index [10]

$$\mathrm{F} = \frac{(\sum x_i)^2}{n\left(\sum x_i^2\right)}. \tag{9}$$

Observe that F is a continuous differentiable function bounded from above by 1, and this upper bound is reached when the allocation is fair $(x_1 = x_2 = \cdots = x_n)$.

In this section, we use the fairness index to derive a simple sufficient condition that guarantees convergence to fairness. (It is also a necessary condition for stepwise-convergence.) The following numerical example will motivate and provide an intuitive feel for the result.

Example 1: Consider two flows with windows of size $x_1 = 8$ and $x_2 = 10$. If an application of the increase policy must result in a fair window size of 11 for both, the smaller flow must change (increase) by a larger amount: $\Delta x_1 = 3$, as compared to $\Delta x_2 = 1$. Similarly, if a decrease must result in a fair window size of 7, the smaller flow must decrease by a smaller amount, or equivalently, change by a larger amount: $\Delta x_1 = -1$ which is greater than $\Delta x_2 = -3$. ■

Thus, the *signed* change in the window size Δ_x, must be greater for the flow with the smaller window. The theorem below shows that it is sufficient for the *signed proportional* change $\Delta x_1/x_1$ to be greater.

Theorem 1 (2-Flow Fairness Condition): Two flows with window sizes x_1 and x_2, $x_1 < x_2$, sharing a bottleneck link will eventually converge to a fair allocation of bottleneck link bandwidth if the following condition is satisfied (after each application of an increase policy $\mathcal{I}$ and decrease policy $\mathcal{D}$ or over any reasonably small period of time):

$$\frac{\Delta x_1}{x_1} \geq \frac{\Delta x_2}{x_2}. \tag{10}$$

At least one of the two policies must ensure a strict inequality.

Proof: The proof proceeds as follows: Suppose two flows with windows x_1 and x_2 share a bottleneck link. Let $\Delta\mathrm{F}$ be the change in F corresponding to a small change Δx_1 in x_1 and Δx_2 in x_2. If $\Delta\mathrm{F}$ is positive at each application of an increase-decrease policy, then eventually reaches its maximum value of 1 regardless of its initial value, and the system moves to a fair allocation. Thus, we only need to ensure $\Delta\mathrm{F} \geq 0$ always.

For $n = 2$, (9) becomes

$$\mathrm{F} = \frac{(x_1 + x_2)^2}{2\left(x_1^2 + x_2^2\right)}.$$

Using

$$d\mathrm{F} = \frac{\partial \mathrm{F}}{\partial x_1} dx_1 + \frac{\partial \mathrm{F}}{\partial x_2} dx_2$$

and making the continuous fluid approximation that the changes Δx_1 and Δx_2 represent infinitesimal changes to x_1 and x_2

$$dx_1 \approx \Delta x_1 \quad dx_2 \approx \Delta x_2 \quad d\mathrm{F} \approx \Delta\mathrm{F} \tag{11}$$

we get

$$\begin{aligned}\Delta\mathrm{F} =& \frac{\left\{\left(x_1^2 + x_2^2\right)(x_1 + x_2) - (x_1 + x_2)^2 x_1\right\}\Delta x_1}{\left(x_1^2 + x_2^2\right)^2} \\ &+ \frac{\left\{\left(x_2^2 + x_1^2\right)(x_2 + x_1) - (x_2 + x_1)^2 x_2\right\}\Delta x_2}{\left(x_1^2 + x_2^2\right)^2}.\end{aligned}$$

Imposing the condition $\Delta\mathrm{F} \geq 0$, we get

$$\left(x_1^2 + x_2^2\right)(\Delta x_1 + \Delta x_2) \geq (x_1 + x_2)(x_1 \Delta x_1 + x_2 \Delta x_2). \tag{12}$$

Simplifying, and using $x_2 - x_1 > 0$, we can write

$$\frac{\Delta x_1}{x_1} \geq \frac{\Delta x_2}{x_2}.$$

Note that we need at least one of $\mathcal{I}$ or $\mathcal{D}$ to ensure $\Delta\mathrm{F} > 0$ so that F increases over each congestion epoch and eventually becomes 1. Thus, at least one of them must have a strict inequality in (10). Also, once $x_1 = x_2$, this equality is maintained under synchronous feedback and the values of the window sizes will increase or decrease in lockstep with each other. ■

We can use a linear interpolation of the window size between two applications of $\mathcal{I}$ for GAIMD and TCP, so that $dx_1 = \Delta x_1$, $dx_2 = \Delta x_2$ and $d\mathrm{F} = \Delta\mathrm{F}$ which is stronger than (11). Thus, the above proof applies to these protocols even though the changes Δx_1 and Δx_2 are not infinitesimal.

This is used in the following corollary which gives a new algebraic proof of convergence to fairness for two GAIMD (or TCP) flows. Chiu and Jain [5] give a different proof for the convergence of GAIMD under the same conditions. This validates the correctness of our results in a way. We also give the first algebraic proof of convergence for binomial congestion control. (The original proof in [2] is a geometric proof based on the Chiu–Jain phase plot.)

Corollary 2: Two TCP or GAIMD flows converge to fairness

Proof: For TCP, $\Delta x = 1$ for the increase policy and (10) becomes: $1/x_1 > 1/x_2$ if $x_1 < x_2$. Similarly $\Delta x = -x/2$ for the decrease policy and (10) reduces to $-1/2 = -1/2$.

For GAIMD, $\Delta x = \alpha$ for the increase policy and (10) becomes: $\alpha/x_1 > \alpha/x_2$ if $x_1 < x_2$. Similarly $\Delta x = -\beta x$ for the decrease policy and (10) reduces to $-\beta = -\beta$. ■

For Binomial Congestion Control, $\Delta x = \alpha/x^k$ for the increase policy and (10) becomes: $\alpha/x_1^{k+1} \geq \alpha/x_2^{k+1}$ if $x_1 < x_2$. Similarly $\Delta x = -\beta x^l$ for the decrease policy and (10) reduces to $-\beta x_1^{l-1} \geq -\beta x_2^{l-1}$ if $x_1 < x_2$. Thus, the increase and decrease policy separately ensure convergence to fairness only if $k > -1$ and $l > 1$.

However, SQRT and IIAD, the two instances of binomial congestion control experimentally evaluated in [2] have values of $l < 1$. Also, as discussed in Section II, we can use $l > 1$ only if we know the maximum window size. The following corollary shows that binomial congestion control converges to fairness if $k, l \geq 0$, which is satisfied by both $\mathrm{SQRT}(k = 1/2, l = 1/2)$ and $\mathrm{IIAD}(k = 1, l = 0)$.

The proof proceeds as follows: We have shown above that each application of an increase policy increases fairness if $k > -1$. Thus, any sequence of window size updates using only the increase policy increases fairness. The proof shows that even though the decrease policy will worsen fairness when $0 \leq l < 1$, the increase in fairness from the previous application of the increase policy more than offsets this decrease in fairness. Also, for sufficiently large window sizes, binomial congestion control is 1-responsive. Thus, each application of a decrease policy is always preceded by an increase policy, so that the value of F increases over each congestion epoch.

Corollary 3: Binomial congestion control converges to fairness if $k, l \geq 0$.

Proof: Clearly, binomial congestion control with $k, l \geq 0$ satisfies the 1-responsiveness criterion (7) for sufficiently large window sizes. Thus, each application of a decrease policy is preceded by an application of the increase policy.

Suppose the window sizes of the two flows are $x_1, x_2 (x_1 < x_2)$, just before the application of the increase policy that is followed by an application of the decrease policy. It is sufficient to show that the fairness index increases over this subsequence of window size adjustments (an increase followed by a decrease) since we already know that sequences consisting only of applications of increase policy improve the fairness index if $k > -1$. (Because of 1-responsiveness we need not consider a subsequence with two or more window decreases.)

We need to show that if $x_1 \leq x_2$

$$\frac{\frac{\alpha}{x_1^k} - \beta\left(x_1 + \frac{\alpha}{x_1^k}\right)^l}{x_1} \geq \frac{\frac{\alpha}{x_2^k} - \beta\left(x_2 + \frac{\alpha}{x_2^k}\right)^l}{x_2}.$$

Approximating $\beta(x + \alpha/x^k)^l \approx \beta x^l$, we need to prove

$$\frac{\frac{\alpha}{x_1^k} - \beta(x_1)^l}{x_1} \geq \frac{\frac{\alpha}{x_2^k} - \beta(x_2)^l}{x_2}$$

or

$$\frac{\alpha - \beta x_1^{k+l}}{x_1^{k+1}} \geq \frac{\alpha - \beta x_2^{k+l}}{x_2^{k+1}}.$$

But when $x_1 \leq x_2$, we have $1/x_1^{k+1} \geq 1/x_2^{k+1}$ and $\alpha - \beta x_1^{k+l} > \alpha - \beta x_2^{k+l}$ because $k + l > 0$.

Thus, the decrease in fairness due to the application of the decrease policy is offset by the increase in fairness resulting from the previous increase in window size. Thus, F always improves over a congestion epoch, and binomial congestion control converges to fairness even if $0 \leq l < 1$. ■

Theorem 1 can be extended for $n > 2$ flows also.

Theorem 4 (n-Flow Fairness Condition): n-flows with windows $x_1, x_2, \ldots, x_n$ converge to fairness if the following condition is satisfied (after each application of an increase policy $\mathcal{I}$ and decrease policy $\mathcal{D}$ or over any reasonably small period of time)

$$\sum_{i=1}^{i=n} x_i^2 \sum_{i=1}^{i=n} \Delta x_i \geq \sum_{i=1}^{i=n} x_i \sum_{i=1}^{i=n} x_i \Delta x_i. \tag{13}$$

Again, at least one of the two policies should ensure a strict inequality.

The proof is very similar to the 2-flow case. Notice that the n-flow result reduces to (12) for $n = 2$.

Corollary 5: n GAIMD or TCP flows converge to fairness

Proof: We derive the results for GAIMD. We can show that n TCP flows converge to fairness in exactly the same way. In fact, the result for GAIMD implies the result for TCP because TCP is a special case of GAIMD.

- Case 1: The increase policy satisfies (13). In this case $\Delta x_i = \alpha$. Since F is bounded from above by 1, we get [from (9)]

  $$1 \geq \frac{(\sum x_i)^2}{n\,(\sum x_i^2)}.$$

  Rewriting this, we get

  $$\sum_{i=1}^{i=n} x_i^2 \sum_{i=1}^{i=n} 1 \geq \sum_{i=1}^{i=n} x_i \sum_{i=1}^{i=n} x_i \cdot 1.$$

  Multiplying both sides by α

  $$\sum_{i=1}^{i=n} x_i^2 \sum_{i=1}^{i=n} \alpha \geq \sum_{i=1}^{i=n} x_i \sum_{i=1}^{i=n} x_i \cdot \alpha$$

  which is (13) with $\Delta x_i = \alpha$.
- Case 2: The decrease policy satisfies (13). In this case $\Delta x_i = \beta x_i$. Equation (13) becomes

  $$\sum_{i=1}^{i=n} x_i^2 \sum_{i=1}^{i=n} \beta x_i = \sum_{i=1}^{i=n} x_i \sum_{i=1}^{i=n} x_i \cdot \beta x_i.$$

Thus, GAIMD ensures that each application of the increase policy leads to an increase in fairness, but maintains the fairness index when the decrease policy is applied. ■

It can be shown that n binomial flows also satisfy (13) and hence converge to fairness. The proof sketch is very similar to the proof for Theorem 9. However, this result is easily obtained in Section IV-B as a special case of Theorem 10. Thus, we have the following:

Corollary 6: $n > 2$ binomial flows converge to fairness.

IV. CYRF

In this section, we adopt a novel approach to protocol design. Since the primary motivation behind this work is the wide range of requirements of different applications, we would like to know what latitude an application can have in choosing a response function. We ask the question: "What is the class of increase-decrease policies that satisfy (10)?". This yields a new family of congestion control protocols that are *designed* to converge to fairness and efficiency.

A. $f(\cdot)$, $g(\cdot)$ *Congestion Control*

Suppose two flows share a bottleneck. Assuming that Δx is some function of x, we can see that (10) (and Example 1) implies some kind of monotonicity for Δx. Also, this function must be differentiable for the proof of Theorem 1 to apply. Furthermore, for convergence to efficiency, the principle of negative feedback discussed in Section III-E must be satisfied. These requirements are expressed in the following theorem.

Theorem 7 (2-Flow Fairness for CYRF): Let $f(x)$ and $g(x)$ be any differentiable monotonically non-decreasing functions (at least one of them strictly increasing) with $f(x) > 0$ and $0 < g(x) \leq 1$ for all $x > 1$. Then the increase and decrease policies in (1) ensure convergence to fairness and efficiency for two flows sharing a bottleneck link.

Proof: $\Delta x = x(t)/f(x(t))$ for the increase policy and $\Delta x = -x(t)g(x(t))$ for the decrease policy.

Convergence to fairness: It is easy to see that, if $x_1, x_2 (x_1 < x_2)$, are the two window sizes, then because of the monotonicity of $f(\cdot)$ and $g(\cdot)$, $1/f(x_1) \geq 1/f(x_2)$ and $-g(x_1) \geq -g(x_2)$; so the increase and decrease policies satisfy (10). Note that since at least one of the two functions is strictly increasing, we have a strict inequality for at least one of the two policies as required by Theorem 1.

Convergence to efficiency: For CYRF to be efficient, the principle of negative feedback must apply and $\mathcal{I}$ must increase the window and $\mathcal{D}$ must decrease the window size. This is clearly satisfied for all window sizes $x(t) \geq 1$, because Δx is positive for the increase policy and negative for the decrease policy (due to the constraints $f(x(t)) > 0$ and $g(x(t)) > 0$).

The upper bound $g(x) \leq 1$ ensures that $\mathcal{D}$ does not lead to negative window sizes. ■

Because each application of an increase or decrease policy moves the system toward fairness, CYRF belongs to the class of *step-wise convergent* protocols. For protocols with a *smooth* increase policy, we can drop the requirement that $g(x)$ be a monotonically nondecreasing function; instead, we only require that $f(x)g(x)$ be monotonically nondecreasing and greater than 1 for $x > c$, for some small constant c. We then obtain *epoch-wise convergent* protocols that only converge over each congestion epoch. Note that we need either $f(x)$ or $f(x)g(x)$ to be strictly increasing to meet the "strict inequality" requirement of Theorem 1. We call this class of protocols 1-CYRF because if $f(x)g(x) > 1$, the protocol is 1-responsive.

Theorem 8: 1-CYRF converges to fairness for $n = 2$ flows.

Proof: In 1-CYRF, each application of the increase policy will still increase the fairness index. However, the decrease policy can now worsen fairness if $g(x)$ is not monotonic. But if the increase in F can offset the decrease, the system will still converge to fairness over each *congestion epoch*. To accomplish this, we impose a stronger constraint that the increase in F from a single application of $\mathcal{I}$ must be more than the decrease in F from a single application of $\mathcal{D}$ and use the 1-responsiveness condition to ensure that each application of a decrease policy is preceded by at least one increase. Thus, in deterministic steady state, F will still increase over each congestion epoch.

A single application of $\mathcal{I}$ followed by an application of $\mathcal{D}$ increases fairness if (10) is satisfied. Here $\Delta x = x/f(x) - (x + x/f(x))g(x + x/f(x)) \approx x(1/f(x) - g(x))$ because of smoothness. Thus, we require $1/f(x_1) - g(x_1) \geq 1/f(x_2) - g(x_2)$ if $x_1 \leq x_2$. Notice that if $f(x)g(x)$ is monotonically *nondecreasing*, then

$$\frac{1}{f(x)} - g(x) = \frac{1 - f(x)g(x)}{f(x)}$$

is monotonically *nonincreasing*. Thus, the inequality required by (10) will be automatically satisfied. ■

CYRF was *designed* to converge to fairness for the two-flow case. The following theorem proves that CYRF converges to fairness for $n > 2$ flows also.

Theorem 9: CYRF converges to fairness for $n > 2$ flows.

Proof: We will prove a stronger result, viz., that CYRF flows satisfy the sufficiency condition given by (13). The proof proceeds as follows: Without loss of generality, we will order the flows by increasing window size. We then use mathematical induction to show that if (13) is satisfied with k flows, adding a $(k+1)$th flow with a larger window size preserves the fairness condition. The key fact used is that $1/f(x)$ and $-g(x)$ are both monotonically nonincreasing, so that for all $1 \leq i \leq k$, we have $1/f(x_{k+1}) \leq 1/f(x_i)$ and $-g(x_{k+1}) \leq -g(x_i)$.

- Case 1: The increase policy increases fairness. Theorem 7 forms the base case. Without loss of generality let $x_1 \leq x_2 \leq \ldots \leq x_k \leq x_{k+1} = X$ be the window sizes of $k+1$ flows. Suppose (13) is satisfied with $n \leq k$ flows. Here, $\Delta x = x/f(x)$. So we get

$$\sum_{i=1}^{k} x_i^2 \sum_{i=1}^{k} \frac{x_i}{f(x_i)} \geq \sum_{i=1}^{k} x_i \sum_{i=1}^{k} \frac{x_i^2}{f(x_i)}. \tag{14}$$

Consider $n = k+1$. Because $x_{k+1} = X \geq x_i$, for all $1 \leq i \leq k$, and $f(\cdot)$ is monotonically nondecreasing, we can write

$$X \sum_{i=1}^{k} \frac{X x_i - x_i^2}{f(x_i)} \geq \frac{X}{f(X)} \sum_{i=0}^{k} \left(X x_i - x_i^2\right). \tag{15}$$

Rewriting, we get

$$X^2 \sum_{i=1}^{k} \frac{x_i}{f(x_i)} + \frac{X}{f(X)} \sum_{i=1}^{k} x_i^2 \geq X \sum_{i=1}^{k} \frac{x_i^2}{f(x_i)} + \frac{X^2}{f(X)} \sum_{i=1}^{k} x_i. \tag{16}$$

Adding (14) and (16), adding $X^3/f(X)$ to both sides and factoring, we get

$$\left(\sum_{i=1}^{k} x_i^2 + X^2\right)\left(\sum_{i=1}^{k} \frac{x_i}{f(x_i)} + \frac{X}{f(X)}\right) \geq \left(\sum_{i=1}^{k} x_i + X\right)\left(\sum_{i=1}^{k} \frac{x_i^2}{f(x_i)} + \frac{X^2}{f(X)}\right).$$

This is just (13) for $n = k+1$. Hence, by the principle of mathematical induction, (13) holds for any number of flows, n.

- Case 2: The decrease policy increases fairness. The algebra is exactly the same as above, except that $1/f(\cdot)$ is replaced by $-g(\cdot)$, which is also a nonincreasing function.

Since one of $f(x)$ or $g(x)$ is strictly increasing, one of the two policies $\mathcal{I}$ or $\mathcal{D}$ will ensure a strict inequality as required. ■

Note that the previous argument for convergence to efficiency still holds for the n-flow case and need not be repeated again.

It can also be shown that 1-CYRF converges to fairness for n-flows. The proof is very similar to Theorem 9. The increase case is exactly the same. In the decrease case, instead of $-g(\cdot)$, we use $1/f(x) - g(x)$, which is a decreasing function as shown above. Thus, we have:

Theorem 10: 1-CYRF converges to fairness for $n > 2$ flows.

B. Applications of CYRF

We discuss three applications of the CYRF framework: a classification of window-based protocols, a tool for unifying analyses of protocols, and constructing new protocols for different application and network needs.

Theorem 11: Window-based protocol families can be classified as follows:

(a) CYRF is necessary and sufficient for stepwise-convergence; 1-CYRF is sufficient for epochwise-convergence.
(b) Smooth, 1-responsive CYRF protocols are also 1-CYRF but the converse is not true.
(c) TCP, GAIMD $\in$ 1-CYRF; TCP-friendly binomial $\in$ 1-CYRF.

Proof: The sufficiency conditions in (a) are obvious. To prove that CYRF is is necessary for stepwise-convergence, observe that any window increase or decrease function can be written in the form of (1). However, if there is a range $[x_1, x_2]$ where the f and g are not monotonic, the function is not CYRF. In such a case we can construct an increase-decrease step similar to Example 1 with window sizes in $[x_1, x_2]$ which *decreases* fairness. Thus, CYRF implies stepwise convergence and vice-versa.

Notice that if $f(x), g(x)$ are monotonically nondecreasing, so is their product $f(x)g(x)$. Thus, if a CYRF protocol is smooth and 1-responsive (and all important protocols are), it is also 1-CYRF. Clearly the converse is not true—Binomial Congestion Control can be written in terms of (1)

$$f(x) = \frac{x^{k+1}}{\alpha} \quad g(x) = \beta x^{l-1}. \tag{17}$$

However, this is not CYRF for $l < 1$ because $g(x)$ will then monotonically *decrease*. Fortunately, all TCP-friendly Binomial controls satisfy $k+l = 1$ [2] so that $f(x)g(x) = \beta x/\alpha > 1$ for $x > (\alpha/\beta)$. Thus, TCP-friendly binomial congestion control is 1-CYRF but not CYRF.

Also, by substituting $f(x) = x$, and $g(x) = 1/2$ in (1), we see that TCP is a special case of CYRF. Similarly, GAIMD is also a special case of CYRF ($f(x) = x/\alpha$, and $g(x) = \beta$). ■

Fig. 1 summarizes the relationships. The rectangles represent new classes introduced in this work, and the ovals show known classes. Specific protocols in each class are given in

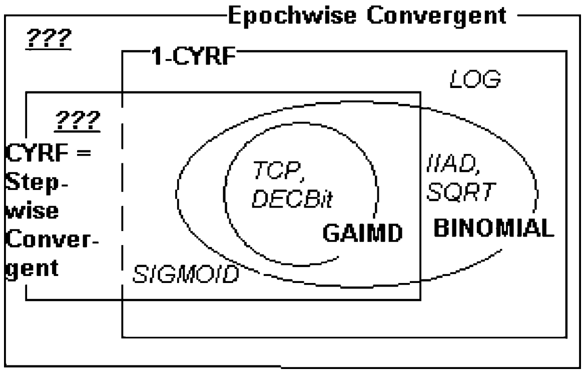

Fig. 1. Classification of window-based protocols.

italics. In particular, observe that SIGMOID is in the class of step-wise convergent protocols wheras LOG is a 1-CYRF protocol and thus is only epochwise convergent. As the "???" marks indicate, we do not know of any protocols that are 1-CYRF but not CYRF or protocols that are epoch-convergent but not 1-CYRF. We believe that TCP-friendly 1-CYRF may represent the widest class of smooth window-based memoryless binary feedback TCP-friendly congestion control protocols that always converge to fairness.

CYRF can also be thought of as a framework for analyzing window-based protocols. For example, it follows from the above discussion that Binomial, GAIMD and TCP converge to fairness and efficiency because they fall within the CYRF framework.

The third (and main) application envisaged for the CYRF framework is the design of new protocols to suit different application and network needs. Protocols in the CYRF framework provably converge to fairness as shown above. The next section derives a simple relation between the $f(\cdot)$ and $g(\cdot)$ that guarantees TCP-friendliness for smooth protocols. Thus, designing protocols for different situations reduces to choosing the appropriate $f(\cdot)$ and $g(\cdot)$.

V. TCP-Friendly CYRF

Smooth protocols are now being widely studied because streaming media flows require smooth transport for effective playout at the receiver end. Such protocols are also required to be *TCP-friendly*. We first obtain a useful approximation valid in smooth protocols and characterize TCP-friendliness in steady state. Using this, we obtain a simple rule for CYRF to be TCP-friendly. We then design a TCP-friendly 1-CYRF protocol called LOG with the aim of reconciling the conflicting needs of smoothness and a fast dynamic response to congestion.

A. Smoothness

Recall that a window increase or decrease policy $x_{t+R} \leftarrow x_t + \Delta x$, is smooth if $|\Delta x| \ll x$. Suppose the increase policy of a smooth protocol is successively applied two times: $x_{t+R} \leftarrow x_t + \Delta x_t, x_{t+2R} \leftarrow x_{t+R} + \Delta(x_{t+R})$. Because $|\Delta x| \ll x$ and $|\Delta(x_{t+R})| \ll x_{t+R}$, we can write $x_{t+2R} \approx x_t + 2\Delta(x_t)$ or in general

$$x_{t+nR} \approx x_t + n\Delta(x_t) \tag{18}$$

for n successive applications of $\mathcal{I}$ (n being of at most the order of x so that the errors do not add up significantly). This gives the following *smoothness lemma*.

Lemma 1 (Smoothness Lemma): In 1-responsive protocols, the successive window sizes after each application of a smooth increase policy during a congestion epoch can be approximated by an arithmetic series.

In particular, this applies to smooth CYRF protocols in steady state. Suppose X is the maximum window size in the epoch, just before the application of $\mathcal{D}$. For 1-responsive protocols, $\mathcal{D}$ is also the last window change in the current epoch. The window size just after the application of $\mathcal{D}$, $X - Xg(X)$, is also the initial window size for the epoch because of the steady-state condition. Using this with the arithmetic series approximation, the number of applications of the increase policy is given by the number of terms in the series

$$n = f(X)g(X) + 1 \approx f(X)g(X) \tag{19}$$

and number of packets sent during the epoch is the sum of the terms

$$S_n = \frac{n}{2}X\big(2 - g(X)\big). \tag{20}$$

B. TCP-Friendliness

An arbitrarily smooth and aggressive protocol is dangerous because it cannot respond fast enough to congestion indications. To protect all flows, and in particular, legacy TCP flows that dominate the Internet, we require the notion of TCP-compatibility [4], which states that a flow should send no more than a comparable conformant TCP flow (same RTT, MTU, etc). This is easily achieved by maintaining the arrival rate to at most some suitable constant c, times the square root of the packet loss rate p [16]. This is called TCP-friendliness. The following theorem gives a new characterization of TCP-friendliness in steady state:

Theorem 12 (TCP-Friendliness): A 1-responsive flow is TCP-friendly in deterministic steady state if and only if the number of packets, S_n, sent during a congestion epoch is related to n, the number of applications of the increase policy during that epoch as

$$S_n \propto n^2. \tag{21}$$

Proof: Suppose the packet size is B and the steady-state (or average) round-trip time is R. Then the (long-term) throughput over the epoch is given by $T = S_n B / nR$. Note that B and R are constant for a given flow, so $T \propto S_n/n$.

$\Rightarrow$ Assuming $S_n \propto n^2$, or equivalently, $n \propto \sqrt{S_n}$ we have $T \propto \sqrt{S_n}$ But by definition, the loss rate p for 1-responsive protocols in steady state is given by $p = 1/S_n$.[3] Thus, we get $T \propto 1/\sqrt{p}$, which is the standard condition for TCP-friendliness.

$\Leftarrow$ Assume $T \propto 1/\sqrt{p}$ or equivalently $T \propto \sqrt{S_n}$. Since $T \propto S_n/n$, we get $S_n \propto n^2$. ■

Corollary 13: A smooth CYRF flow is TCP-friendly in steady state if and only if

$$f(x)g(x) \propto x. \tag{22}$$

[3]$p = k/S_n$ for k-responsive protocols. Also, note that $S_n \neq 0$ because of 1-responsiveness.

Proof: Substituting from (19) and (20) in (21), we see that $X(2-g(X))/2 \propto f(X)g(X)$. if a CYRF flow must be TCP-friendly. If $g(\cdot) \ll 2$ (Fig. 2 shows that this is valid for the three protocols discussed in the next section), we get (22).

Note that (22) also implies 1-responsiveness, which was an implicit assumption in the above theorem. ■

We observe in passing that by substituting from (17) in (22), we get $k+l=1$, which is the rule for binomial congestion control to be TCP-friendly. In Appendix A, we derive rules for strict TCP-compatibility in CYRF, and obtain the rule for GAIMD (5) from it.

In the next section, we sketch the design of an example TCP-friendly protocol.

C. Log

CYRF represents a wide class of window-based protocols. Applications can choose different $f(x)$ and $g(x)$ to get different window-based protocols. For example, from (1) it is easy to see that an application that uses a slowly increasing function for $f(x)$ will be more aggressive and make full use of network bandwidth as soon as it becomes available. Similarly a slowly increasing $g(x)$ results in a smoother response to congestion indications. We cannot choose an arbitrarily aggressive increase policy together with a very smooth decrease policy because of the TCP-friendliness constraint [15], [16] which requires all flows to limit their sending rate to that of a comparable TCP flow. Thus, there is a continuum of protocols with different *degrees of smoothness*.

While smoother protocols are better from the application point of view (for streaming media applications), this is true only in steady state. Aggressive protocols are also more responsive protocols and usually have better transient and dynamic behaviors [3], [24].

Here we trade-off between smoothness and dynamic or transient behavior by designing a protocol whose properties are "between" SQRT ($k=l=0.5$ in (6)) and IIAD($k=1, l=0$), the two nonlinear binomial controls studied in [2]. From (17), we see IIAD is smoother than SQRT because $g_{\mathrm{IIAD}}(x)=1/x$ is always less than $g_{\mathrm{SQRT}}(x)=1/\sqrt{x}$.

As shown in Fig. 2, by choosing a function that lies between these two, such as $g(x)=log(x)/x$, we get a new protocol with intermediate properties. For this to be TCP-friendly, we need $f(x) \propto x^2/log(x)$ (Corollary 13). Thus, the increase and decrease policies in (1) become

$$\begin{aligned} \mathcal{I}: x(t+R) \quad &\leftarrow \quad x(t)+\frac{\log(x)}{x} \\ \mathcal{D}: x(t+R) \quad &\leftarrow \quad x(t)-0.5*\log(x). \end{aligned} \tag{23}$$

We call this protocol LOG.[4] From Fig. 2, we see that just like IIAD and SQRT, $g(x)$ for LOG is a decreasing function and hence LOG is only 1-CYRF. The decrease policy will worsen fairness. However, unlike IIAD or SQRT, $g(x)$ increases for very small window values; thus LOG is CYRF in this region. Due to this, it is likely to perform better in extremely congested situations when the window sizes are small.

[4]Note that Corollary 15 in the Appendix would give a decrease factor of $0.3\log(x)$. Experimentally, we find that the protocol is not very sensitive to small changes in the multiplicative constant.

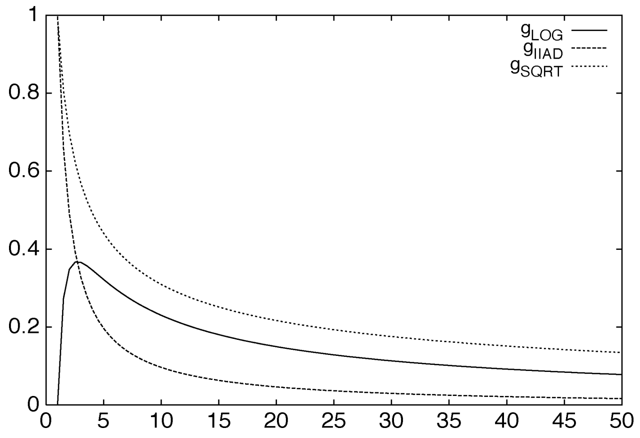

Fig. 2. Smoothness hierarchy through functions of different orders of magnitude: $g(x)$ for SQRT, IIAD and LOG.

VI. Simulation-Based Validation

We have implemented CYRF in the *ns*-2 network simulator by changing TCP's congestion avoidance mechanism to use arbitrary increase or decrease functions. CYRF inherits other mechanisms such as slow-start and timeouts from TCP. We use LOG as an example CYRF protocol to verify the main theoretical results. We will also examine the properties of LOG and then propose SIGMOID, a protocol that overcomes the "TCP-unfriendliness" of IIAD, SQRT, and LOG in the presence of droptail queues.

Most of the experiments use the standard "dumbbell" topology: a single bottleneck link with a default bandwidth of 10 Mb and a delay of 1 ms.[5] RED queues with a maximum queue size $Q_{\max}$ equal to the bandwidth-delay product and maximum and minimum drop thresholds at 20% and 80% of $Q_{\max}$ are used. n flows are started at random times in the first two seconds. All flows use 1 Kb packets and saturated senders are simulated by using the FTP application in *ns*. A random number of Reno TCP flows form background traffic.

A. Validating Theoretical Results

In this section, we will verify the theoretical results of this paper. Fig. 3 shows the scaled values for the congestion window variables (cwnd_) of two competing flows, and the corresponding Jain–Chiu–Hawe Fairness Index. These results show that LOG rapidly converges to a value of F near 1 verifying the results of Theorems 1 and 7. The repeated instantaneous decreases in fairness are because of packet drops (mostly random RED drops). Note that LOG is a 1-CYRF protocol and drops invoke the decrease policy which worsens fairness.

Fig. 4 shows the congestion window evolution of a single LOG flow. The points are decimated by plotting one in every 20 cwnd_ values for clarity. We also give the linear best-fit for the cwnd_ values (taking all cwnd_ values into account) which clearly shows the validity of the arithmetic-series approximation in Lemma 1.

[5]This was chosen to approximate the "synchronous feedback" assumption in the theorems. The results do not change significantly for larger delay values.

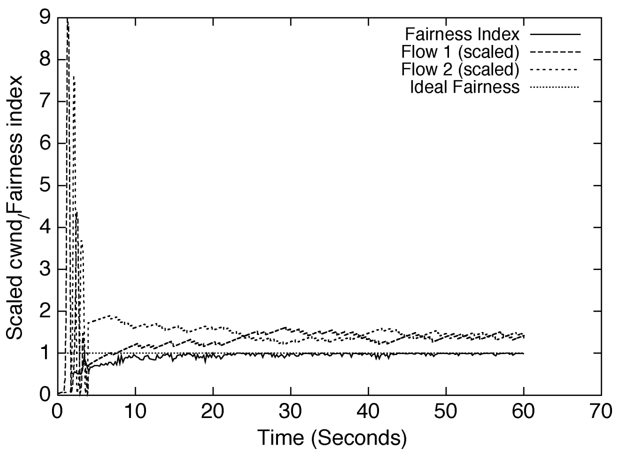

Fig. 3. Fairness index and `cwnd_` of two competing LOG flows.

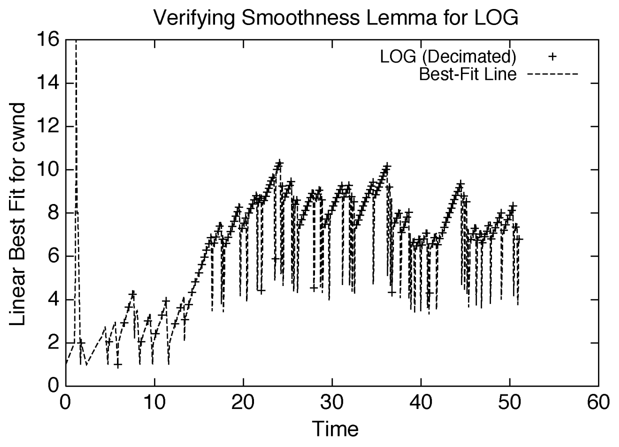

Fig. 4. LOG congestion window evolution.

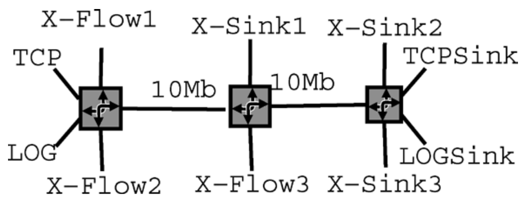

Fig. 5. Multiple-bottlenecks: topology.

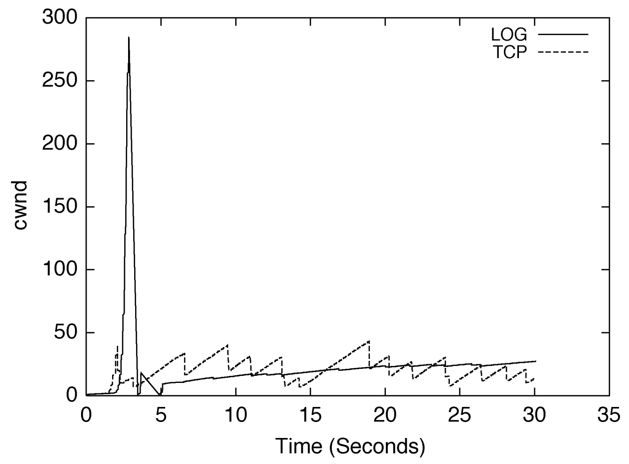

Fig. 6. Multilink-bottlenecks: window size variation.

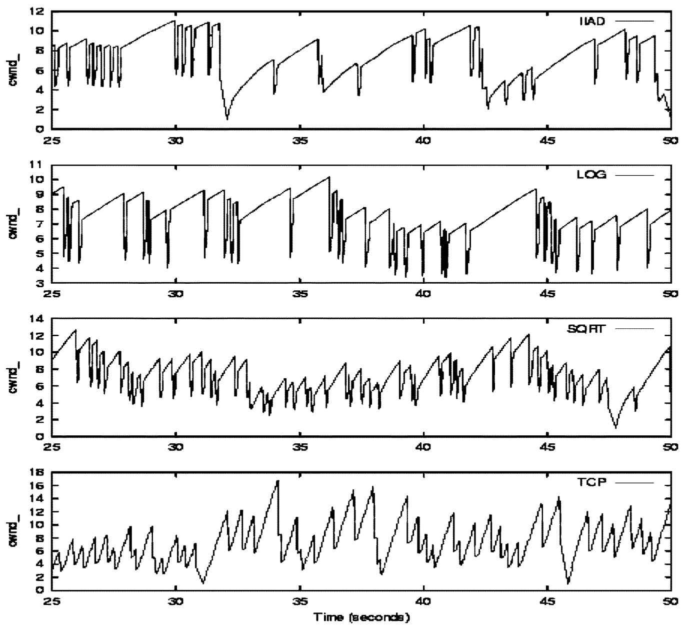

Fig. 7. Smoothness hierarchy: `cwnd_` variations of (from the top) IIAD, LOG, SQRT and TCP.

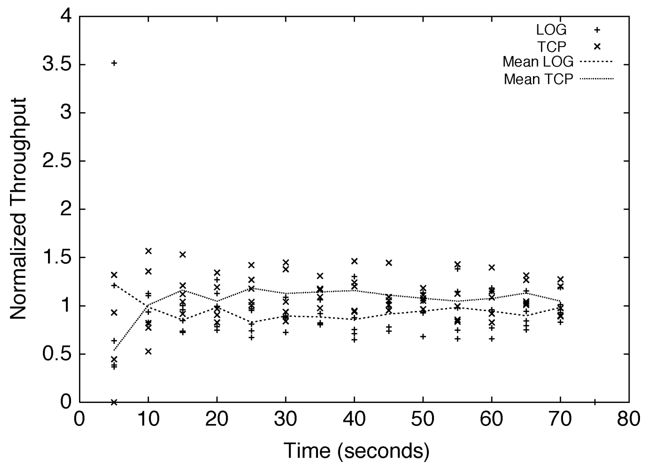

Fig. 8. Normalized throughput (Kb/500 ms).

In the next experiment, we investigate the interaction between LOG and TCP. We use the multiple-bottleneck topology shown in Fig. 5. The inter-router links are 10 Mb with a delay of 1 ms, and the others are 100 Mb with 24 ms delay, designed to saturate the routers. All queues are RED. X-Flow*i* is the source of a TCP/Reno cross-flow to X-Sink*i* that starts at a random time in the first two seconds. A TCP/Reno flow from TCP to TCPSink and a LOG flow from LOG to LOGSink are examined. Fig. 6 shows that the congestion window of the LOG flow varies much more smoothly than TCP and shares the bandwidth effectively. Similar results were obtained for the dumbbell topology.

Fig. 7 depicts the congestion windows of competing TCP, SQRT, LOG and IIAD flows over a window of time from $t = 25$ s to $t = 50$ s in the above setup. This validates the basis for the design of LOG in Section V-C, namely protocols with a smaller $g(x)$ experience smoother variations in window size.

Fig. 8 shows the normalized throughputs of n TCP and n LOG flows in the single bottleneck case. This shows the TCP-friendliness of LOG as predicted by theory. (LOG/IIAD and LOG/SQRT interactions are quite similar and are omitted for space reasons).

TABLE I
TROUGH-TEST METRICS

Protocol	Utilization Metric(%)	Drop Metric(%)	Steady State Utilization(%)	Steady State Drop Rate(%)
TCP	20.8557	1.94642	94.8312	5.59690
SQRT	22.4542	3.98332	95.0987	4.47275
LOG	21.8173	3.94338	95.1109	3.76404
IIAD	25.0437	4.87134	94.9525	3.82518

B. Trough-Test Benchmark

In order to study the relative dynamic behaviors of IIAD, SQRT and CYRF(LOG), we performed the following experiment: n TCP flows and n flows of one of these protocols share the dumbbell topology described previously. A CBR flow with a rate equal to half the bottleneck link bandwidth also flows in the same direction. Background traffic consists of a random number of TCP flows. At time $t_1 = 50$ s, the CBR flow is stopped. At $t_2 = 80$ s, it starts again. This "*bandwidth trough*" caused by stopping the CBR flow between t_1 and t_2 allows us to measure several useful metrics. These are summarized in Table I. Each metric was measured over 10 repeated experiments. The figures in the table represent the average values. In the rest of this section, "steady state" is used to refer to a time betwen $t = 20$ and $t = 30$ s, when all the startup transients have been stabilized, and the CBR flow is still running.

When additional bandwidth becomes available at t_1, smoother protocols take more time to make use of it. Similarly, an overly aggressive protocol can ramp up too much and may have to reduce its window, thus losing some utilization. We define the *Utilization Metric* of a protocol as the product of the *drop* in utilization at t_1 (from the utilization at steady state) and the amount of time it takes to get back to 95% of the steady-state utilization. This is a measure of the additional amount of data that could have been sent by an "ideal protocol" that adapts to the available bandwidth. (The utilization metric, multiplied by the bottleneck link bandwidth gives the additional amount of data in terms of bits or bytes.) Obviously, this number must be as small as possible. The table shows that LOG hits a "sweet spot" between overly aggressive and overly smooth protocols (All numbers are percentage values).

Similarly, we see an increased drop rate (up to 10–15 times the steady-state drop rate!) at t_2. This is because the smoother protocols cannot reduce their sending rates fast enough to accomodate the CBR flow and thus saturate the link. We measure this by the *Drop Metric*, which is given by the product of the additional drop rate (as compared to the steady-state drop rate) and the time required to get back to 1.5 times the steady-state drop rate. This is similar to the metric defined in [3], except that we use the difference in the drop rates at t_2 and at steady state, instead of using the drop rate at t_2 directly. Again, this number should be as low as possible and the table shows that LOG achieves a reasonably low metric (numbers given are percentage values). Observe that smoothness can make a huge difference in times of sudden congestion such as at t_2. The metric for TCP is considerably smaller than the others, and IIAD performs much worse than the other protocols.

Finally, we measured the average drop rate and link utilization in steady state, when the CBR flow is running, the results show that LOG achieves the highest utilization (95.11%) and the least drop rate (3.76%). Each number is marginally better than the other protocols.

C. SIGMOID

All the previous experiments used RED queues. Now we show that LOG (and other nonlinear window-based "TCP-friendly" protocols) become "TCP-unfriendly" in the face of drop-tail queues and severely congested situations. Fig. 9(b) shows the window sizes of competing LOG and TCP flows in the standard bottleneck link topology described in the previous section, with a bottleneck bandwidth changed to 1 Mb (from 10 Mb in the other experiments) and RED queues in the bottleneck routers. Fig. 9(a) shows that LOG grabs an unfair share of bandwidth in the same simulation with drop-tail queues. Similar results have been reported in [2] for IIAD and SQRT. Fig. 9(c)–(f) confirms their results.

This happens because drop tail queues can back-up and over-flow in congested situations. TCP reduces its sending rate by half to flush the queue. The "smoother" non-TCP flows reduce by a smaller amount. Thus, they grab more bandwidth and the queue does not get flushed. Because they use different window increase-decrease policies, TCP and other non-TCP flows see different drop rates when the queue becomes full. However, the non-TCP flows are *designed* to send at a rate inversely proportional to the square root of the loss-rate which causes the disparity. RED varies the drop rate as a function of the queue size making all TCP-friendly flows see the same drop rate thus eliminating this problem.

Clearly, the root cause for the problem is smoothness. Also, smooth flows with larger window sizes cause more damage [14]. We now propose a solution called SIGMOID that works around this issue by behaving exactly like TCP when the window size is large enough. Thus, it has a dynamic behavior exactly like TCP for large windows. It also tries to ensure a minimum window size by reducing the window decrease for smaller windows and thus achieves a minimum throughput. Thus, when used with a playout buffer, smoothness is no longer an essential requirement—When there is available bandwidth, SIGMOID ramps up quickly exactly like TCP, thus filling the playout buffer. In times of congestion, a continuous playout stream is still possible because of the achieved minimum throughput together with the reserve playout minutes built up in the buffer. It gracefully gives up any additional bandwidth by quickly reducing its window size just as TCP would, thus flushing a backed-up drop-tail queue.

Thus, SIGMOID works by replacing the TCP-friendliness requirement with the requirement that the flow should be considerate to TCP when the window size is large enough. The smoothness requirement is obviated by allowing a fast TCP-style window increase together with playout buffers and a minimum throughput "guarantee." Fig. 9(g) and (h) shows that SIGMOID performs much better than the other nonlinear congestion controls in the same conditions as before with virtually no difference between the drop-tail and RED queue configurations.

We conclude by showing how to implement the SIGMOID requirements in the CYRF framework. Clearly, we need a decrease function $g(x)$ that is similar to that of TCP ($g_{\mathrm{TCP}}(x) =$

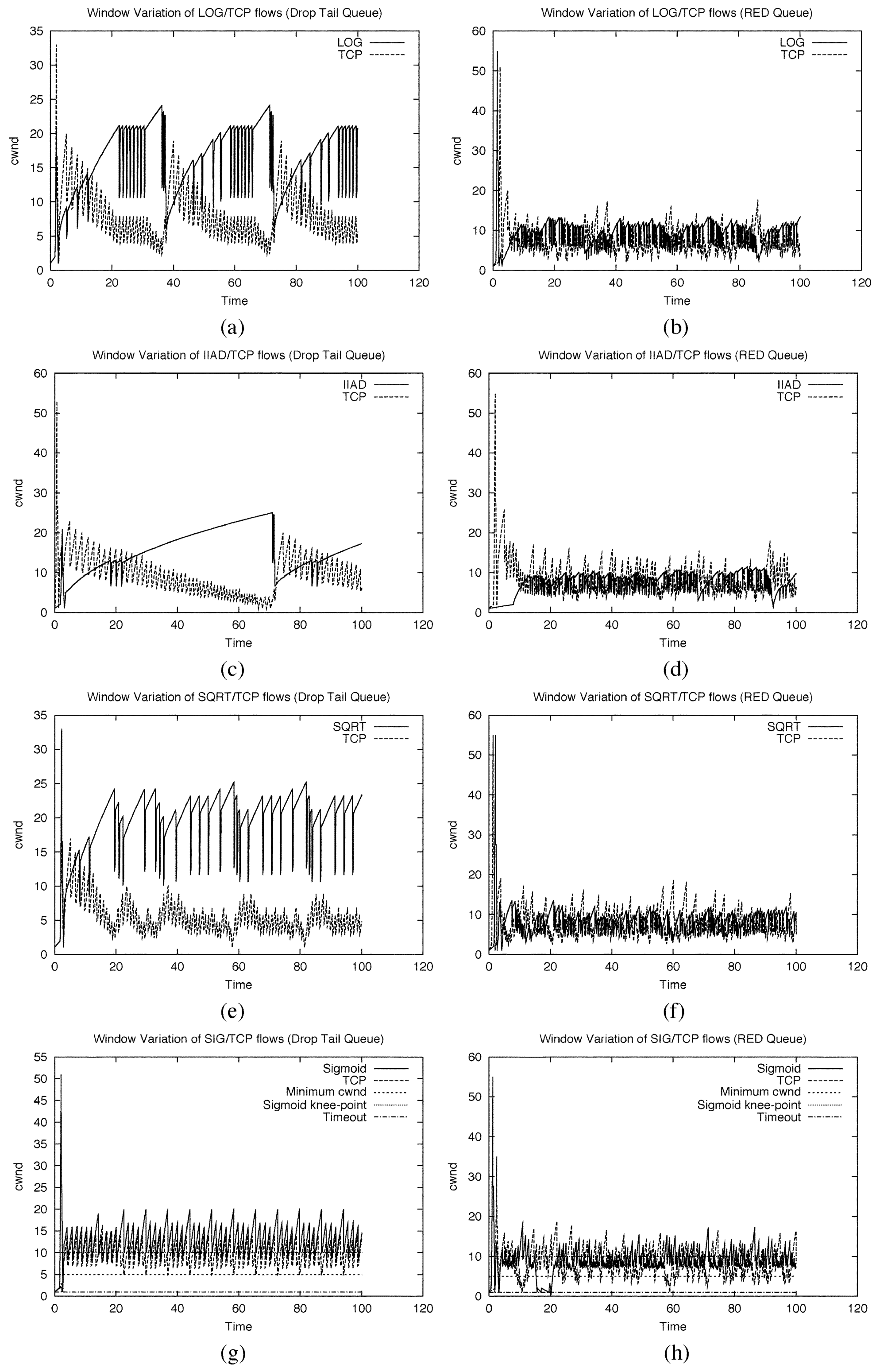

Fig. 9. RED-Effect: (a) and (b) LOG, (c) and (d) IIAD, (e) and (f) SQRT, (g) and (h) SIGMOID. The left-hand figures use drop-tail queues, and the right-hand figures use RED queues.

$0.5 = c$) for large windows and near zero for smaller windows. We consider functions of the form $g(x) = cg'(x)$ with $g'(x) \approx 0$ for small x, and $g' \to 1$ as x increases. If the congestion window x was small at the time a congestion indication is received, $g'(x) \approx 0$, and the window size is decreased very little, or not at all. On the other hand, if a window of data large enough to fill the receiver's playout buffer had been sent previously, $g'(x) \approx 1$, and the window size is halved exactly like TCP.

The well-known sigmoid function

$$g(x) = \frac{c}{1 + e^{(-a(x-k))}} \tag{24}$$

has the shape we are looking for. c is the maximum of this function. a determines how smoothly the function changes from

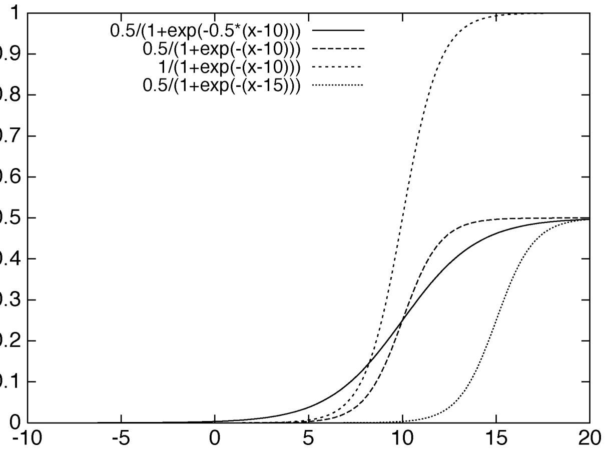

Fig. 10. Different SIGMOID functions: varying a, k, and c.

a value near 0 to a value near c. Smaller values of a give a smoother knee. The knee-point on the x axis can be changed by altering k. Fig. 10 shows the possibilities.

The increase function can be the same as TCP. Thus, we get the following increase–decrease policy for SIGMOID:

$$\begin{aligned}\mathcal{I} : x(t+R) &\leftarrow x(t)+1 \\ \mathcal{D} : x(t+R) &\leftarrow x(t) - \frac{cx(t)}{\left(1+e^{-a(x(t)-k)}\right)}.\end{aligned} \tag{25}$$

Fig. 9(g) and (h) uses $c = 0.5$, $a = 0.5$, and $k = 10$. Above the knee point, SIGMOID behaves exactly like TCP because $g(x)$ saturates to a value of $g_{\text{TCP}}(x) = 0.5$. Below it, $g(x) \approx 0$, so that the decrease policy leaves the window size virtually unchanged and thus a minimum throughput is guaranteed. Choosing the right knee point is extremely important for the proper operation of SIGMOID since this determines the minimum throughput of the SIGMOID flow and thus the maximum throughput of the other flows. If the knee point is too high for a given bottleneck, SIGMOID may never decrease its window. A useful rule of thumb is to have $k = 0.1$ssthresh_-in congestion avoidance mode; the slow-start threshold, ssthresh_, represents a "safe" estimate of the flow's share of the bottleneck, and a safety factor of $s = 0.1$ allows the queue to be flushed after getting backed up. A different safety factor can be chosen depending on the application's need for minimum bandwidth guarantees.

VII. Conclusion and Scope for Future Work

In this paper, we presented CYRF, a novel approach to protocol design that is *guaranteed* to converge to fairness and efficiency. This allows protocol designers to choose the appropriate response function given the application and network issues at hand, without having to worry about fairness and efficiency. Such protocols can also be made TCP-friendly easily. Using this framework, we designed and evaluated a protocol called LOG, with intermediate smoothness and aggressiveness. We also briefly discussed SIGMOID, a CYRF protocol which tries to address the "TCP-unfriendliness" of window-based non-TCP protocols in the presence of drop-tail queues. It circumvents the smoothness requirement for streaming media by taking a radically different approach.

On the theoretical front, we gave a necessary and sufficient condition for convergence to fairness and a new characterization of smoothness and TCP-friendliness in steady state. Both of these results can easily be made use of outside the CYRF framework. CYRF itself includes most well-known window-based protocols as special cases and can be used to understand these protocols better. Finally, we gave a new classification of window-based protocols based upon the results of this paper.

Clearly, the basic framework presented here can be improved if it can also ensure other cross-protocol concerns such as scalability. The window increase–decrease policies are restricted to be functions of the current window size. The framework can be extended allowing more parameters, such as a history of previous window sizes. The theorems can be strengthened by relaxing some of the assumptions of the Chiu–Jain model used here. We are currently looking into some of these possibilities.

Appendix A
TCP-Compatibility

As stated in Section V-B, the notion of TCP-compatibility requires that a flow should send no more than a comparable conformant TCP flow. TCP-friendliness advocates maintaining the arrival rate to at most some suitable constant c, times the square root of the packet loss rate. We determine c, and using this, we derive the rule for CYRF to be TCP-compatible.

Theorem 14 (TCP-Compatibility): A 1-responsive protocol with a congestion epoch of size n during which S_n packets are sent is TCP-compatible in deterministic steady state if and only if

$$3n^2 = 2S_n. \tag{26}$$

Proof: From Theorem 12 we know that a protocol is TCP-friendly in steady state if and only if $n^2/S_n = c$ for some constant c. To get the proportionality constant, we just need to plug in the values of n and S_n for some TCP-compatible protocol. In particular, we know that TCP itself is TCP-compatible.

Suppose the maximum window size in a TCP congestion epoch is W_{TCP}. This is the window size after the last increase in the epoch. It is followed by an application of the decrease policy, which decreases the window to $W_{\text{TCP}}/2$ and the next congestion epoch starts.[6] In steady state, the initial window size of each congestion epoch is the same and hence equal to the final window size of the epoch. Also, from (3) we can see that the successive window sizes during a sequence of applications of the increase policy form an arithmetic series with a term difference of 1. Thus, we can write $W_{\text{TCP}} = W_{\text{TCP}}/2 + (n-1)$ or $n \approx W_{\text{TCP}}/2$. We also get the number of packets sent during the epoch as the sum of the series

$$S_n = \frac{n}{2}\left(2\frac{W_{\text{TCP}}}{2} + (n-1)\cdot 1\right) \approx \frac{3W_{\text{TCP}}^2}{8}.$$

Plugging these values into the relation $n^2/S_n = c$, we get the required result. ■

By a very similar argument, we can show that a k-responsive protocol is TCP-compatible if $kn^2/S_n = 2/3$.

[6]As in the proof of Theorem 12, this follows from our definition of the congestion epoch and the fact that TCP is 1-responsive.

Using (19) and (20) in (26), we get the following.

Corollary 15: A smooth CYRF flow is TCP-compatible in steady state if and only if

$$f(X) = \frac{X\,(2 - g(X))}{3g(X)}. \tag{27}$$

Notice that when $g(X) \ll 2$ we get back the TCP-friendliness condition of Corollary 13. As another quick check, substituting the f and g values for GAIMD, we get (5).

References

[1] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," Internet Engineering Task Force, RFC 2581 (Standards Track), 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*, San Diego, CA, Aug. 2001, pp. 263–273.

[4] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, and S. Floyd, "Recommendations on queue management and congestion avoidance in the Internet," Internet Engineering Task Force, RFC 2309 (Informational), Apr. 1998.

[5] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comput. Netw. ISDN Syst.*, vol. 17, pp. 1–14, 1989.

[6] J. Crowcroft and P. Oechslin, "Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP," *ACM Comput. Commun. Rev.*, vol. 28, no. 3, pp. 53–67, Jul. 1998.

[7] 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

[8] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. ACM SIGCOMM*, Aug. 2000, pp. 43–56. Extended version available as International Computer Science Institute Tech. Report TR-00-03, Mar. 2000.

[9] V. Jacobson and M. Karels, "Congestion avoidance and control," *ACM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, Aug. 1990. Revised version of an ACM Sigcomm'88 paper.

[10] R. Jain, D.-M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," Digital Equipment Corporation, DEC Research Report, Tech. Rep. TR-301, Sep. 1984.

[11] R. Jain, K. K. Ramakrishnan, and D.-M. Chiu, "Congestion avoidance in computer networks with a connectionless network layer," Digital Equipment Corporation, Tech. Rep. DEC-TR-506, 1988. Reprinted in *Innovations in Internetworking*, C. Partridge, Ed. Norwood, MA: Artech House, 1988.

[12] S. Jin, L. Guo, I. Matta, and A. Bestavros, "TCP-friendly SIMD congestion control and its convergence behavior," in *Proc. 9th IEEE Int. Conf. Network Protocols (ICNP)*, Nov. 2001, pp. 156–164.

[13] L. Kalampoukas, A. Varma, and K. K. Ramakrishnan, "Explicit window adaptation: a method to enhance TCP performance," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar./Apr. 1998, pp. 242–251.

[14] R. Mahajan and S. Floyd, "Controlling high-bandwidth flows at the congested router," in *Proc. 9th IEEE Int. Conf. Network Protocols (ICNP)*, Nov. 2001, pp. 192–201.

[15] J. Mahdavi and S. Floyd. (1997, Jan.) TCP-friendly unicast rate-based flow control. [Online]. Available: http://www.psc.edu/networking/papers/tcp_friendly.html

[16] J. Mahdavi and S. Floyd. (1999, Jun.) The TCP-friendly web-site. [Online]. Available: http://www.psc.edu/networking/tcp_ friendly.html

[17] K. K. Ramakrishnan, S. Floyd, and D. Black, "The addition of Explicit Congestion Notification (ECN) to IP," Internet Engineering Task Force, RFC 3168 (Standards Track), Sep. 2001.

[18] K. K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 158–181, May 1990.

[19] R. Rejaie, M. Handley, and D. Estrin, "Quality adaptation for unicast audio and video," in *Proc. ACM SIGCOMM*, Sep. 1999, pp. 189–200.

[20] ——, "RAP: an end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *Proc. IEEE INFOCOM*, Mar. 1999, pp. 1337–1345.

[21] I. Rhee, V. Ozdemir, and Y. Yi, "TEAR: TCP Emulation at Receivers—Flow Control for Multimedia Streaming," North Carolina State Univ., Raleigh, NC, Technical Report, Apr. 2000.

[22] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: a TCP-friendly adaptation scheme," presented at the 8th Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Cambridge, U.K., 1998.

[23] G. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, MA: Addison Wesley, 1995.

[24] Y. R. Yang, M. S. Kim, and S. S. Lam, "Transient behaviors of TCP-friendly congestion control protocols," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001, pp. 1716–1725.

[25] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proc. 8th IEEE Int. Conf. Network Protocols*, Osaka, Japan, Nov. 2000, pp. 187–198.

Nishanth R. Sastry received the Bachelor's degree in computer science and engineering in 1999, graduating with distinction from the R.V. College of Engineering, Bangalore University, India. He received the Master's degree in computer science from the University of Texas at Austin in 2001.

During 1999–2000, he was with Cisco Systems, working on ATM and LAN emulation in the catalyst switches. Since 2002, he has been with IBM, working on groupware for work-place team collaboration. His interests include all aspects of designing practical distributed systems, from low-level network protocol design and congestion control to high-level distributed collaborative environments.

Simon S. Lam (S'71–M'74–SM'80–F'85) received the B.S.E.E. degree with distinction from Washington State University, Pullman, in 1969, and the M.S. and Ph.D. degrees in engineering from the University of California at Los Angeles (UCLA) in 1970 and 1974, respectively.

From 1971 to 1974, he was a Postgraduate Research Engineer at the ARPA Network Measurement Center, UCLA, where he worked on satellite and radio packet switching networks. From 1974 to 1977, he was a Research Staff Member at the IBM T. J. Watson Research Center, Yorktown Heights, NY. Since 1977, he has been on the faculty of the University of Texas at Austin, where he is Professor and Regents Chair in Computer Sciences, and served as Department Chair from 1992 to 1994. His current research interests are in network protocol design and analysis, distributed multimedia, and Internet security services.

Dr. Lam served on the editorial boards of IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, IEEE TRANSACTIONS ON COMMUNICATIONS, *Proceedings of the IEEE*, and *Performance Evaluation*. He was Editor-in-Chief of IEEE/ACM TRANSACTIONS ON NETWORKING from 1995 to 1999. He currently serves on the editorial board of *Computer Networks*. He organized and was Program Chair of the inaugural ACM SIGCOMM Symposium held at the University of Texas at Austin in 1983. He is a founding Steering Committee member of the IEEE International Conference on Network Protocols. He received the 2004 ACM Software System Award, the 2004 ACM SIGCOMM Award, the 2004 W. Wallace McDowell Award from IEEE Computer Society, and the 1975 Leonard G. Abraham Prize and the 2001 William R. Bennett Prize from the IEEE Communications Society. He has been a Fellow of the Association for Computing Machinery (ACM) since 1998.