CYRF: A Framework for Window-based Unicast Congestion Control *

Nishanth R. Sastry†
nishanth_sastry@us.ibm.com

Simon S. Lam
lam@cs.utexas.edu

Department of Computer Sciences,
The University of Texas at Austin
TR-02-09

January 30, 2002
Revised September 23, 2002

Abstract

This work presents a comprehensive theoretical framework of window-based congestion control protocols called CYRF (for *C*hoose *Y*our *R*esponse *F*unction) that are designed to converge to fairness and efficiency. We first derive a sufficient condition for convergence to fairness. Using this, we show how fair window increase/decrease policies can be constructed from suitable pairs of monotonically non-decreasing functions. We also give a new characterization of TCP-friendliness and simple rules for smooth CYRF flows to be TCP-friendly.

Specific protocols that meet the needs of a given situation can be designed within this framework. We experimentally investigate two CYRF protocols for streaming media-like applications: LOG, a protocol suitable for applications that need to trade-off between smoothness and aggressiveness, and SIGMOID, which is designed to be "TCP-friendly" even in a network with drop-tail queues (unlike other non-linear window based protocols)

1 Introduction

Van Jacobson's congestion control and avoidance mechanisms for TCP [11] have been instrumental for the success and stability of the Internet. Chiu and Jain [6] proved that, assuming synchronous feedback, any additive-increase multiplicative-decrease (AIMD) mechanism such as TCP converges to fairness

*Research sponsored in part by Texas Advanced Research Program grant no. 003658-0439-2001

†Currently with IBM.

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. This is arguably the main reason for the success of TCP. We say an end-to-end congestion control protocol is *feasible* if and only if it ensures convergence to fairness and efficiency, at least under "ideal" conditions.

While TCP's AIMD response function is appropriate for applications with *elastic* bandwidth requirements, it results in degraded performance for applications such as streaming media which need smoother increase/decrease policies. Consequently, several new congestion control protocols such as RAP [23], TFRC [10], General AIMD or GAIMD [31] and Binomial Congestion Control [2] have been designed targeting the needs of specific applications, especially streaming media. Because legacy TCP flows still dominate the Internet, it is desirable that these protocols be *TCP-friendly* [17,18], which roughly means that all flows must send at the same mean rate as a TCP flow. Thus, the design of such protocols is complex because of additional constraints imposed by application needs such as smoothness and network or deployment issues such as TCP-friendliness.

To simplify the task of designing a *feasible* protocol that also suits a given situation, we propose a window-based congestion control framework called CYRF (for *C*hoose *Y*our *R*esponse *F*unction) which *guarantees* convergence to fairness and efficiency. By choosing specific window increase and decrease policies (or response functions), CYRF can be easily adapted to suit different needs.

Our main theoretical result is that given two monotonically non-decreasing 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 (under assumptions detailed in Section 3):

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

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 also 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 non-decreasing, and always greater than 1 for $x > 1$.

Next, we look at a specific application of this framework: transporting streaming media-like applications. As stated before, the traditional model for this class of applications requires smoothness and TCP-friendliness. We first introduce a new characterization of TCP-friendliness and then show that a smooth

CYRF flow will be TCP-friendly in steady state if

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

Observe from Equation 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 (Equation 2). Thus there is a continuum of protocols with different *degrees of smoothness*.

While smoother protocols are better from the application's point of view, this is true only in steady-state. Aggressive protocols are also more responsive to congestion indications and thus have better transient and dynamic behaviors [3, 30]. We design and experimentally investigate a 1-CYRF protocol called LOG, with aggressiveness and smoothness properties intermediate to that of IIAD and SQRT, the two other non-linear (binomial) congestion controls that have been previously studied [2].

Finally, we briefly describe a CYRF protocol called SIGMOID that represents a novel approach to congestion control for streaming media applications. Clearly, with the traditional model, we cannot escape the fundamental trade-off between smoothness and responsiveness, which is essential in dynamic situations. Furthermore, non-linear window-based congestion control protocols such as LOG, IIAD and SQRT do not interact well with TCP in the presence of droptail queues. SIGMOID avoids both these problems by behaving exactly like TCP when the window size is large enough. It has a dynamic behaviour exactly like TCP. It also ensures a minimum window size and thus achieves a minimum throughput. Thus, when used with a playout buffer, smoothness is no longer an essential requirement. Experiments show that with the ideal minimum window size, SIGMOID interacts quite well with TCP with both RED and DropTail queues in the network.

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), thus providing a powerful unified framework for the analysis of these protocols. For example, we obtain new proofs for the fairness and TCP-friendliness of these protocols as special cases of the results for CYRF. We also discuss a classification of window-based protocols based on CYRF.

The rest of this paper is organized as follows. In Section 2, we describe related proposals. Section 3 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 4 as the basis for CYRF. We then characterize TCP-friendliness and describe LOG in section 5. Sections 6 and 7 present our simulations on LOG and the SIGMOID protocol. Section 8 concludes the work.

2 Related Work

TCP is the most widely used congestion control mechanism today. Most modern TCP implementations [29] incorporate algorithms introduced by Van Jacobson [1, 11] into 4.3BSD to fix the original 1988 congestion collapse. 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:

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

Congestion avoidance was simultaneously investigated in a series of papers by Jain, Ramakrishnan and Chiu which is summarized in [13]. Prominent congestion control protocols in classical networking literature include Clark *et al.* 's NETBLT [7], Cheriton and Williamson's VMTP [5], Ramakrishnan and Jain's DECBit [21] and Wang and Crowcroft's Tri-S [28].

The notion of TCP-friendliness [17, 18] has given rise to a number of new proposals [2, 10, 23, 24, 27, 31] 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 [13, 31] generalizes TCP to an Additive Increase Multiplicative Decrease (AIMD) policy with different increase and decrease parameters. Its increase and decrease policies are given by

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

with $\alpha = 3\beta/(2-\beta)$ for TCP-friendliness[9].[1] Binomial congestion control [2] proposes the following non-linear increase/decrease policies:

$$\begin{aligned} \mathcal{I} : x(t+R) &\leftarrow x(t)+\alpha/x(t)^k \ ; & \alpha > 0 \\ \mathcal{D} : x(t+R) &\leftarrow x(t)-\beta x(t)^l \ ; & 0<\beta<1 \end{aligned} \tag{5}$$

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 4 of the Chiu-Jain paper [6].

Notice that plugging $\alpha = 1$ and $\beta = 1/2$ for GAIMD (Equation 4) gives us TCP. GAIMD itself is a special case of the binomial algorithm(Equation 5) with $k=0$ and $l=1$. SQRT ($k=l=0.5$) and IIAD ($k=1$, $l=0$) are two non-linear binomial controls in [2] that we will use in later sections.

[1] This does not consider the effect of timeouts. [31] gives a slightly different condition for TCP-friendliness with timeouts.

TFRC [10] is a rate-based scheme which directly uses the TCP throughput equation [19] to estimate its sending rate. TEAR [24] estimates the TCP-friendly rate by emulating the entire TCP state machine at the receiver. RAP [23] modifies the inter-packet gap to provide fine-grained delay-based congestion avoidance. The Loss-Delay Adjustment Algorithm [27] uses feedback from RTP [26] for rate adjustment.

In this work, we do not look at multicast congestion control. We also do not consider application-specific adaptive approaches such as [22] that try to make the best use of the available network support. We consider all flows equal so that schemes like MulTCP [8] 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 [15] 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 [14] fall outside our scope.

3 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.

3.1 Notation

Following Chiu and Jain [6], 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.

3.2 The Model

The following analysis uses Chiu and Jain's *synchronous feedback* assumption [6] 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 [20, 21].

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.

3.3 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, 9]) 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:

$$|\Delta(w_{\mathcal{I}})| \leq |\Delta(w_{\mathcal{D}})| \tag{6}$$

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 Equation 6 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 Equations 3, 4 and 5, 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.

3.4 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. [9] and [30]). 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. In this work, we say a protocol is *smooth* if its increase and decrease policies are smooth. 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

$$|\Delta x| \ll x \tag{7}$$

[2] Note that the minimum possible window size is 1 and for this window, Equation 6 fails. But we assume a congestion epoch with a sufficiently large minimum window size.

3.5 Convergence to Efficency

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 underutilized and decreases when the bottleneck link is overloaded. This is just the principle of negative feedback [6]. 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.

3.6 Convergence to fairness

Fairness is the most 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).

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 [12] F:

$$\mathrm{F} = \frac{(\sum x_i)^2}{n(\sum x_i^2)} \tag{8}$$

Observe that F is a continuous differentiable function upperbounded by 1, and this upperbound is reached when the allocation is totally fair ($x_1 = x_2 = \ldots = x_n$).

In this section, we derive a simple sufficient condition that guarantees convergence to fairness. 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 Δx, 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 and maintain a totally 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):*

$$\frac{\Delta x_1}{x_1} \geq \frac{\Delta x_2}{x_2} \tag{9}$$

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 ΔF be the change in F corresponding to a small change Δx_1 in x_1 and Δx_2 in x_2. If ΔF is positive at each application of an increase/decrease policy, then eventually F reaches its maximum value of 1 regardless of its initial value, and the system moves to a totally fair allocation. Thus we only need to ensure $\Delta \mathrm{F} \geq 0$ always.

For $n = 2$, Equation 8 becomes

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

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:

$$dx_1 \approx \Delta x_1,\ dx_2 \approx \Delta x_2 \text{ and } d\mathrm{F} \approx \Delta \mathrm{F} \tag{10}$$

we get

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

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

$$(x_1^2 + x_2^2)(\Delta x_1 + \Delta x_2) \geq (x_1 + x_2)(x_1 \Delta x_1 + x_2 \Delta x_2) \tag{11}$$

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 Equation 9. 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 Equation 10. 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 [6] 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 under the assumption of synchronized feedback*

Proof: For TCP, $\Delta x = 1$ for the increase policy and Equation 9 becomes: $1/x_1 > 1/x_2$ if $x_1 < x_2$. Similarly $\Delta x = -x/2$ for the decrease policy and Equation 9 reduces to $-(1/2) = -(1/2)$.

For GAIMD, $\Delta x = \alpha$ for the increase policy and Equation 9 becomes: $\alpha/x_1 > \alpha/x_2$ if $x_1 < x_2$. Similarly $\Delta x = -\beta x$ for the decrease policy and Equation 9 reduces to $-\beta = -\beta$. ■

For Binomial congestion control, $\Delta x = \alpha/x^k$ for the increase policy and Equation 9 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 Equation 9 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 2, 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 SQRT ($k = 1/2,\ l = 1/2$) and 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 (Equation 6) 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 decrrease) 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 $x_1 \leq x_2$,

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

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 flows:

Theorem 4 (n-Flow Fairness Condition) *n-flows with windows x_1,x_2,... 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):*

$$\sum_{i=1}^{i=n} x_i^2 \sum_{i=1}^{i=n} \Delta x_i \geq \sum_{i=1}^{i=n} x_i \sum_{i=1}^{i=n} x_i \Delta x_i \tag{12}$$

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 Equation 11 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 Equation 12.

  In this case $\Delta x_i = \alpha$. Since F is upperbounded by 1, we get (from Equation 8)

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

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 Equation 12 with $\Delta x_i = \alpha$.

- Case 2: The decrease policy satisfies Equation 12
  In this case $\Delta x_i = \beta x_i$. Equation 12 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 Equation 12 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 4.2 as a special case of Theorem 10. Thus we have:

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

4 CYRF

In this section, we adopt a novel approach to protocol design. Since the primary motivation behind this work is the wide range of requirments 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 Equation 9?". This yields a new family of congestion control protocols that are *designed* to converge to fairness and efficiency.

4.1 $f(\cdot), g(\cdot)$ Congestion Control

Suppose two flows share a bottleneck. Assuming that Δx is some function of x, we can see that Equation 9 (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 3.5 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 Equation 1 ensure converge 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 $g(\cdot)$, the increase policy satisfies Equation 9: $1/f(x_1) \geq 1/f(x_2)$ and similarly for the decrease policy,$-g(x_1) \geq -g(x_2)$. 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 upperbound $g(x) \leq 1$ ensures that $\mathcal{D}$ does not lead to negative window sizes. ■

Because each application of an increase or decrease policy moves the system towards fairness, we call this class of protocols as *step-wise convergent*.

For protocols with a *smooth* increase policy, we can drop the requirement that $g(x)$ be a monotonically non-decreasing function; instead, we only require that $f(x)g(x)$ be monotonically non-decreasing 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 (2-Flow Fairness for 1-CYRF) *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 Equation 9 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 *non-decreasing*, then

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

is monotonically *non-increasing*. Thus the inequality required by Equation 9 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 (n-Flow Fairness for CYRF) *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 Equation 12. 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 Equation 12 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 non-increasing, so that $1/f(x_{k+1}) \leq 1/f(x_i)$ and $-g(x_{k+1}) \leq -g(x_{k+1})$ for all $1 \leq i \leq k$.

- 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 Equation 12 is satisfied with $n \leq k$ flows. Here, $\Delta x = x/f(x)$. So we get:

  $$\sum_{i=1}^{k} x_i^2 \sum_{i=1}^{k} \frac{x_i}{f(x_i)} \geq \sum_{i=1}^{k} x_i \sum_{i=1}^{k} \frac{x_i^2}{f(x_i)} \tag{13}$$

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

  $$X \sum_{i=1}^{k} \frac{Xx_i - x_i^2}{f(x_i)} \geq \frac{X}{f(X)} \sum_{i=0}^{k} (Xx_i - x_i^2) \tag{14}$$

  Rewriting, we get

  $$X^2 \sum_{i=1}^{k} \frac{x_i}{f(x_i)} + \frac{X}{f(X)} \sum_{i=1}^{k} x_i^2 \geq X \sum_{i=1}^{k} \frac{x_i^2}{f(x_i)} + \frac{X^2}{f(X)} \sum_{i=1}^{k} x_i \tag{15}$$

  Adding Equation 13 and Equation 15, 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 Equation 12 for $n = k + 1$. Hence, by the principle of mathematical induction, Equation 12 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 non-increasing 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 (n-Flow Fairness for 1-CYRF) *1-CYRF converges to fairness for $n > 2$ flows.*

4.2 Applications of CYRF

The first application of CYRF is a "natural" classification of window based protocols. Observe that any window increase or decrease function can be written in the form of Equation 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 non-decreasing, 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 Equation 1:

$$f(x) = x^{k+1}/\alpha \; ; \; g(x) = \beta x^{l-1} \tag{16}$$

However this is not CYRF for $l \leq 1$ 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.

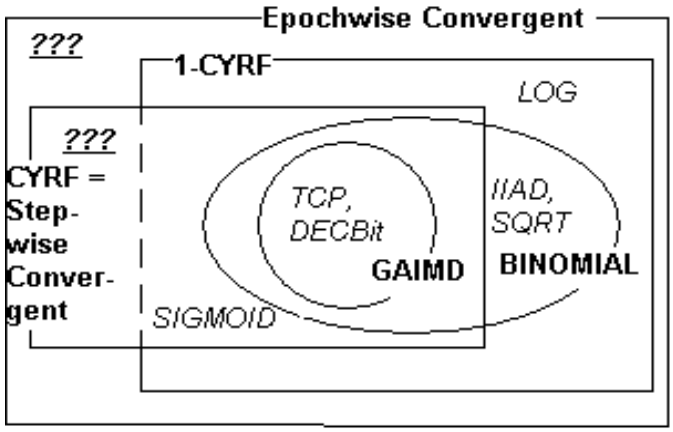

Figure 1: A Classification of Window-Based Protocols

Also, by substituting $f(x) = x$, and $g(x) = 1/2$ in Equation 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 $g(x) = \beta$).

Figure 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 italics. As the "???" marks indicate, we do not know of any useful 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 next section shows that the rules for TCP-friendliness in GAIMD and Binomial congestion control can also be derived as special cases of the results for CYRF.

In the rest of the paper, we will demonstrate a third application of this framework by designing two protocols suitable for streaming media-like applications.

5 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.

5.1 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,

$$x_{t+nR} \approx x_t + n\Delta(x_t) \tag{17}$$

for n successive applications of $\mathcal{I}$ (n being of at most the order of x so that the errors dont add up significantly). This proves the *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:

$$n = f(X)g(X) + 1 \approx f(X)g(X) \tag{18}$$

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

$$S_n = \frac{n}{2}X(2 - g(X)) \tag{19}$$

5.2 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 no flow should send 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 [18]. This is called TCP-friendliness. The following theorem gives a new characterization of TCP-friendliness in steady-state:

Theorem 11 (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:*

$$S_n \propto n^2 \tag{20}$$

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 $p = 1/{S_n}^3$ (Also, note that $S_n \neq 0$ because of 1-responsiveness). 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 12 *A smooth CYRF flow is TCP-friendly in steady-state if and only if*

$$f(x)g(x) \propto x \tag{21}$$

Proof: Substituting from Equations 18 and 19 in Equation 20, 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 Equation 21.

Note that Equation 21 also implies 1-responsiveness, which was an implicit assumption in the above theorem. ■

We observe in passing that by substituting from Equation 16 in Equation 21, we get $k + l = 1$, which is the rule for binomial congestion control to be TCP-friendly. We can also obtain the rule for GAIMD by a slightly more exact analysis that also finds the constant of proportionality in Equation 20. This is discussed next.

5.3 TCP-Compatibility

As stated in Section 5.2, the notion of TCP-compatibility requires that no flow should send 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 give a stricter version of Theorem 11 which also derives the constant of proportionality for Equation 20 and thus achieves strict TCP-compatibility. Using this, we derive the rule for CYRF to be TCP-compatible. We also show how the condition for the tcp-compatibility of GAIMD can be obtained as a special case of the result for CYRF.

[3] $p = k/S_n$ for k-responsive protocols.

Theorem 13 (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:*

$$\frac{n^2}{S_n} = \frac{2}{3} \tag{22}$$

Proof: From Theorem 11 we know that if

$$\frac{n^2}{S_n} = c \tag{23}$$

for some constant c, then the protocol is TCP-friendly in steady state.

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 $W_{TCP}/2$ and the next congestion epoch starts[4]. 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 Equation 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 have

$$W_{TCP} = W_{TCP}/2 + (n-1)$$

This gives us

$$n \approx W_{TCP}/2$$

We also get the number of packets sent during the epoch as the sum of the series:

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

Plugging these values into Equation 23, we get

$$\frac{n^2}{S_n} = \frac{2}{3}$$

for TCP-compatibility. ■

By a very similar argument, we can show that a k-responsive protocol is TCP-compatible if

$$\frac{kn^2}{S_n} = \frac{2}{3} \tag{24}$$

Corollary 14 *A smooth CYRF flow is TCP-compatible in steady-state if and only if*

$$f(X) = \frac{X(2-g(X))}{3g(X)} \tag{25}$$

[4] As in the proof of Thm. 11, this follows from our definition of the congestion epoch and the fact that TCP is 1-responsive

Proof: Substituting from Equations 18 and 19 in Equation 22 we get

$$\frac{2(1+f(X)g(X))}{X(2-g(X))} = \frac{2}{3}$$

Simplifying,

$$f(X) = \frac{X(2-g(X))}{3g(X)}$$

■

Notice that this can be reduced to the TCP-friendliness condition (Corollary 12). As in the proof of Corollary 12, if $g(X) \ll 2$ we can write $f(X)g(X) = X/3$ or $f(X)g(X) \propto X$ which is Equation 21.

Corollary 15 *GAIMD is TCP-compatible if*

$$\alpha = \frac{3\beta}{(2-\beta)} \tag{26}$$

Proof: Although GAIMD is not a smooth protocol, we can see from Equation 4 that Equations 18 and 19 also hold for GAIMD. Thus the proofs of Theorems 11 and 13 and hence Corollary 14 hold for GAIMD (and TCP) also. Substituting $f(x) = x/\alpha$, and $g(x) = \beta$, we get the result. ■

This is a simplified condition for GAIMD to be TCP-compatible ignoring the effect of timeouts. A slightly different proof of this result is given by [9]. Again, this verifies Equation 25 in a way. [31] gives a different condition taking timeouts into consideration.

5.4 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 Equation 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 (Equation 2). 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 usually have better transient and dynamic behaviors [3, 30].

Here we trade-off between smoothness and dynamic or transient behaviour by designing a protocol whose properties are "between" SQRT ($k = l = 0.5$ in eqn. 5) and IIAD ($k = 1$, $l = 0$), the two non-linear binomial controls studied in [2]. From Equation 16, 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 Figure 2,

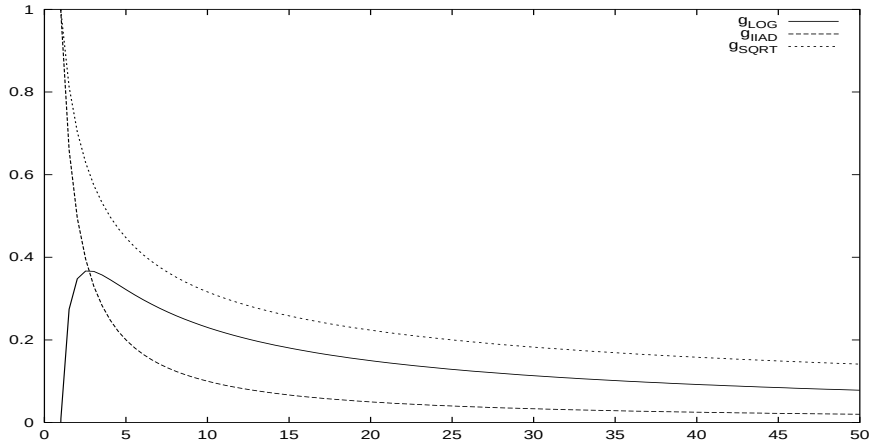

Figure 2: $g(x)$ for SQRT, IIAD and LOG

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 $f(x) \propto x^2/log(x)$ (from Corollary 12). Thus the increase and decrease policies in Equation 1 become:

$$\begin{aligned} \mathcal{I} : x(t+R) &\leftarrow x(t) + log(x)/x \\ \mathcal{D} : x(t+R) &\leftarrow x(t) - 0.5 * log(x) \end{aligned} \tag{27}$$

We call this protocol LOG. 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. It is interesting to note that LOG falls outside the binomial class of protocols, and yet it falls "between" IIAD and SQRT.

6 Experiments

We have implemented CYRF in the *ns-2* network simulator[5] 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 some of the previous 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.

[5] Available from `http://www.isi.edu/nsnam/ns/`

Most of the experiments use the standard "dumb bell" topology: a single bottleneck link with a default bandwidth of 10Mb and a delay of 1ms. 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 percent of Q_{max} are used. n flows are started at random times in the first two seconds. All flows use 1Kb packets and saturated senders are simulated by using the FTP application in *ns*. A random number of Reno TCP flows in the opposite direction form background traffic.

6.1 Verification

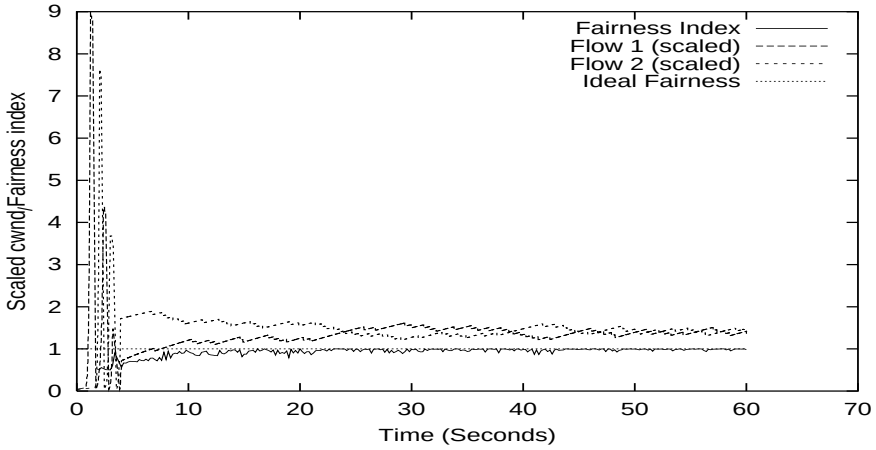

Figure 3: **Fairness Index and cwnd of 2 competing LOG flows**

In this section, we will verify the theoretical results of this paper. Fig. 3 shows the scaled cwnd values of 2 competing flows and the corresponding Chiu-Jain-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.

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 10Mb with a delay of 1ms, and the others are 100Mb with 24ms delay, designed to saturate the routers. All queues are RED. X-Flow$i$ is the

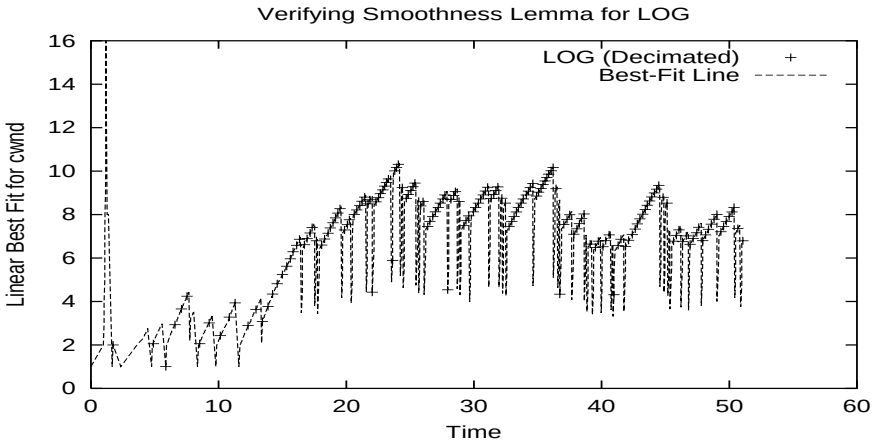

Figure 4: **LOG Congestion Window Evolution**

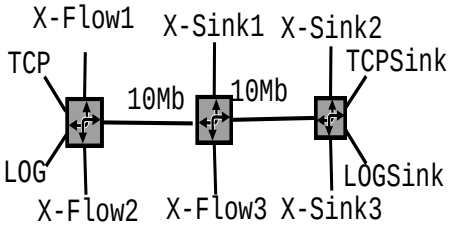

Figure 5: **Multiple-Bottlenecks**:Topology

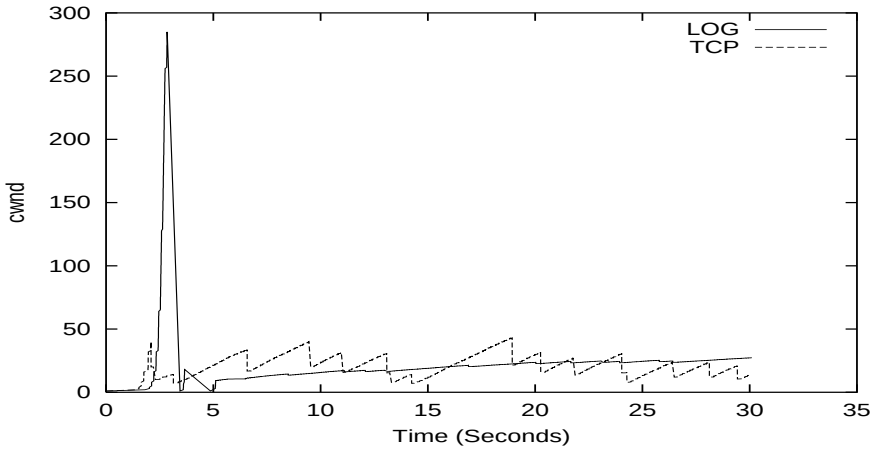

Figure 6: **Multilink-Bottlenecks**:Window Size Variation

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 dumb-bell topology.

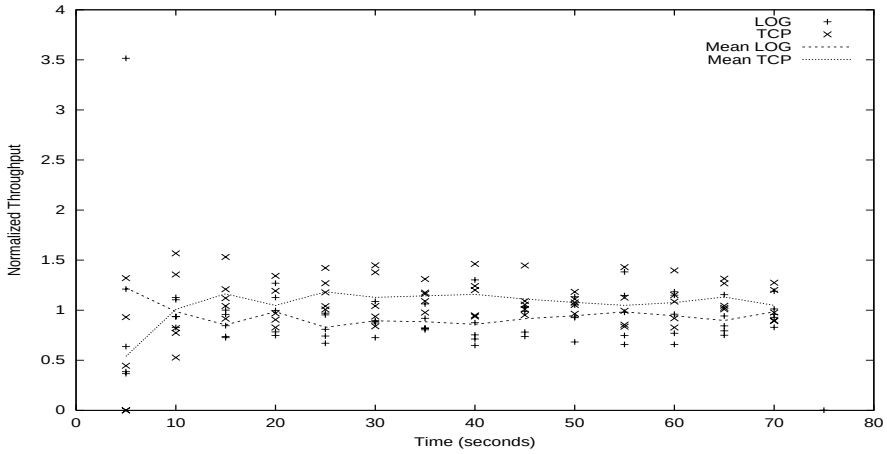

Figure 7: **Normalized Throughput (Kb/500ms)**

Fig. 7 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).

6.2 The 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 dumb bell topology described previously. A CBR flow with a rate equal to half the bottleneck link bandwidth also flows in the same direction. Background traffic in the opposite direction consists of a random number of TCP flows. At time $t_1 = 50$ seconds, the CBR flow is stopped. At $t_2 = 80$ seconds, 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 1. 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 $t = 30$ seconds, 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

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

Table 1: **Trough-Test Metrics**

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 percent 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 (upto 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 percent) and the least drop rate (3.76 percent). Each number is marginally better than the other protocols.

7 SIGMOID

All the previous experiments used RED queues. Now we show that LOG (and other non-linear window-based "TCP-friendly" protocols) become "TCP-unfriendly" in the face of drop-tail queues and severely congested situations. Fig. 8 (b) shows the window sizes of competing LOG and TCP flows in the

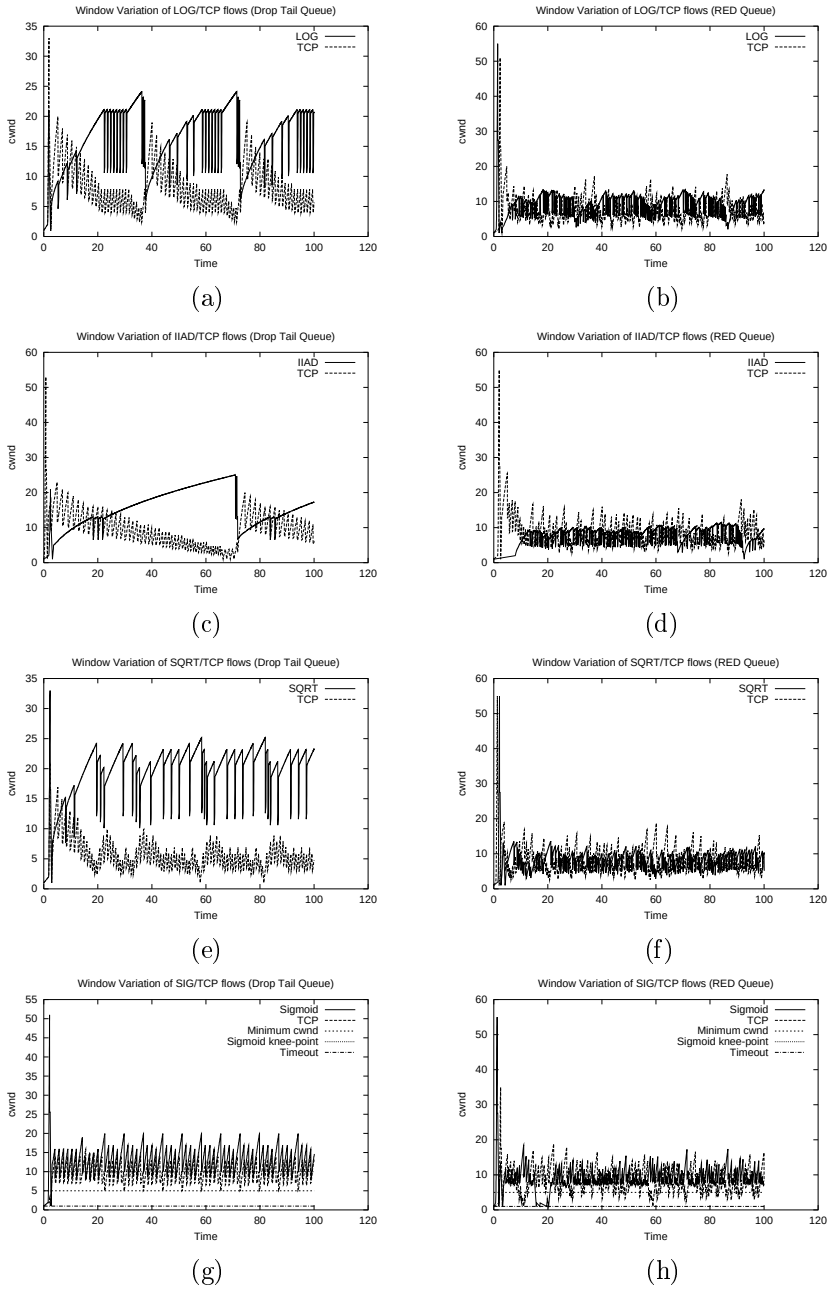

Figure 8: **RED-Effect**:(a) and (b) LOG, (c) and (d) IIAD, (e) and (f) SQRT, (g) and (h) SIGMOID. The left hand figures use drop-tail queues, and the right hand figures use RED queues.

standard bottleneck link topology described in the previous section, with a bottleneck bandwidth changed to 1Mb (from 10Mb in the other experiments) and RED queues in the bottleneck routers. Fig. 8 (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. Figs. 8 (c) to (f) confirm their results.

This happens because drop tail queues can back-up and overflow 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 [16]. 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.

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 througput "guarantee". Figures 8 (g) and (h) show that SIGMOID performs much better than the other non-linear 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. This is easily done in the CYRF framework. Clearly, we need a function $g(x)$ that is similar to that of TCP($g_{\mathrm{TCP}}(x) = 0.5$) for large windows and near zero for smaller windows. The sigmoid function

$$\frac{c}{1 + e^{(-a(x-k))}} \tag{28}$$

has the shape we are looking for. c is the maximum of this function. a determines how smoothly the function changes from 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. 9 shows the possibilities. The increase function

can be the same as TCP. Thus we get the following increase/decrease policy for SIGMOID:

$$\begin{aligned} \mathcal{I} : x(t+R) &\leftarrow x(t)+1 \\ \mathcal{D} : x(t+R) &\leftarrow x(t) - cx(t)/(1+e^{-a(x(t)-k)}) \end{aligned} \tag{29}$$

Figs. 8 (g) and (h) use $c = 0.5$, $a = 0.5$ and $k = 10$. Above the knee point, SIGMOID behaves exactly like TCP because $g(x)$ saturates to a value of $g_{\mathrm{TCP}}(x) = 0.5$. 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.

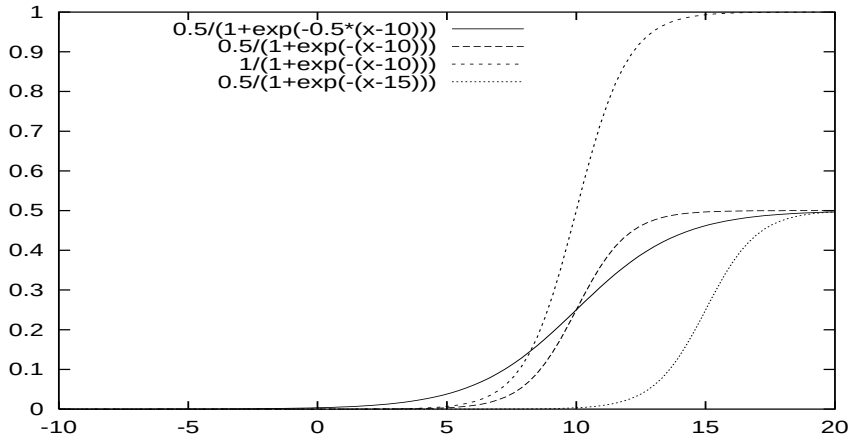

Figure 9: **Different SIGMOID Functions**: Varying a, k and c

8 Conclusion

In this paper, we presented CYRF, a novel approach to protocol design that is that it is *designed* 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 droptail queues. It circumvents the smoothness requirement for streaming media by taking a radically different approach.

On the theoretical front, we gave a new intuitive 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.

9 Note

This work is an extension of the first author's Master's thesis [25]. The theorems in Sections 3, 4 and 5 were originally presented in that report.

References

[1] Mark Allman, Vern Paxson, and W. Stevens. *TCP Congestion Control.* Internet Engineering Task Force, April 1999. RFC 2581 (Standards Track).

[2] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *Proceedings of IEEE INFOCOM 2001*, April 2001.

[3] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker. Dynamic behavior of slowly-responsive congestion control algorithms. In *Proceedings of ACM SIGCOMM 2001*, San Diego, CA, August 2001.

[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, April 1998. RFC 2309 (Informational).

[5] D. Cheriton and C. Williamson. VMTP as the transport layer for high-performance distributed systems. *IEEE Communications Magazine*, pages 37–44, June 1989.

[6] Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[7] D. Clark, M. Lambert, and L. Zhang. NETBLT: A high throughput transport protocol. In *Proceedings of ACM SIGCOMM '88*, August 1988.

[8] J. Crowcroft and P. Oechslin. Differentiated end-to-end internet services using a weighted proportional fair sharing TCP. *ACM Computer Communication Review*, 28(3), July 1998.

[9] S. Floyd, M. Handley, and J. Padhye. A comparison of equation-based and AIMD congestion control. Available from `http://www.aciri.org/tfrc`, May 2000.

[10] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM 2000*, August 1988. Extended version available as International Computer Science Institute tech report TR-00-03, March 2000.

[11] Van Jacobson and Mike Karels. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, August 1990. Revised version of Sigcomm '88 paper.

[12] R. Jain, D-M. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report TR-301, DEC Research Report, September 1984.

[13] R. Jain, K. K. Ramakrishnan, and D-M. Chiu. Congestion avoidance in computer networks with a connectionless network layer. Technical Report DEC-TR-506, Digital Equipment Corporation, 1988. Reprinted in C. Partridge, Ed., *Innovations in Internetworking*, published by Artech House, October 1988.

[14] Shudong Jin, Liang Guo, Ibrahim Matta, and Azer Bestavros. Tcp-friendly simd congestion control and its convergence behavior. In *Proceedings of International Conference on Network Protocols 2001.*

[15] Lampros Kalampoukas, Anujan Varma, and K. K. Ramakrishnan. Explicit window adaptation: A method to enhance TCP performance. In *Proceedings of IEEE INFOCOM '98*, San Francisco, California, March/April 1998.

[16] R. Mahajan and S. Floyd. Controlling high-bandwidth flows at the congested router. In *Proceedings of International Conference on Network Protocols 2001*, Nov 2001.

[17] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control. Technical Note sent to the end2end-interest mailing list, January 1997. Available from `http://www.psc.edu/networking/papers/tcp_friendly.html`.

[18] J. Mahdavi and S. Floyd. The TCP-friendly website. `http://www.psc.edu/networking/tcp_friendly.html`, June 1999.

[19] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM SIGCOMM '98*, September 1998.

[20] K. K. Ramakrishnan, S. Floyd, and D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP.* Internet Engineering Task Force, September 2001. RFC 3168 (Standards Track).

[21] K. K. Ramakrishnan and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Transactions on Computer Systems*, 8(2):158–181, May 1990.

[22] R. Rejaie, M. Handley, and D. Estrin. Quality adaptation for unicast audio and video. In *Proceedings of ACM SIGCOMM '99*, September 1999.

[23] R. Rejaie, M. Handley, and D. Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the internet. In *Proceedings of IEEE INFOCOM '99*, March 1999.

[24] I. Rhee, V. Ozdemir, and Y. Yi. TEAR: TCP emulation at receivers – flow control for multimedia streaming. Technical report, North Carolina State University, April 2000.

[25] Nishanth R. Sastry. Application Specific Unicast Congestion Control. Master's thesis, Department of Computer Sciences, The University of Texas at Austin, December 2001. Also available as Technical Report TR-01-51, Department of Computer Sciences, The University of Texas at Austin, and from `http://www.cs.utexas.edu/users/nishanth/thesis.html`.

[26] H. Schulzrinne, S. Cassner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, January 1996. RFC 1889 (Standards Track).

[27] D. Sisalem and H. Schulzrinne. The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme. In *Proc. 8th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 1998.

[28] Z. Wang and J. Crowcroft. A new congestion control scheme: Slow start and search (tri-s). *ACM Computer Communications Review*, 21:32–43, Jan 1991.

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

[30] Y. R. Yang, M. S. Kim, and S. S. Lam. Transient behaviors of TCP-friendly congestion control protocols. In *Proceedings of IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[31] Y. R. Yang and S. S. Lam. General AIMD congestion control. In *Proceedings of the 8th IEEE International Conference on Network Protocols*, Osaka, Japan, November 2000.