Congestion Control for Self-Similar Network Traffic*

Tsunyi Tuan Kihong Park†
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
{tsunyi,park}@cs.purdue.edu

CSD-TR 98-014
July 16, 1998

Abstract

Analytical and empirical studies have shown that self-similar network traffic can have a detrimental impact on network performance including amplified queueing delay and packet loss rate. Given the ubiquity of scale-invariant burstiness observed across diverse networking contexts, finding effective traffic control algorithms capable of detecting and managing self-similar traffic has become an important problem.

In this paper, we study congestion control algorithms for improving throughput under self-similar traffic conditions. Although scale-invariant burstiness imposes a limit on the ability to conjointly achieve high quality of service (QoS) and utilization, the long-range dependence associated with self-similar traffic leaves open the possibility that the correlation structure may be exploited for performance enhancement purposes.

We use a form of predictive congestion control called Selective Aggressiveness Control (SAC) to show that long-range dependence can be on-line detected and exploited to improve throughput. We show that the correlation structure present in long-range dependent traffic can be used to predict future traffic levels over time scales that are relevant to feedback congestion control. This on-line detected correlation structure is then used to drive a variable aggressiveness control which throttles the data rate upward if the future contention level is predicted to be low, being more aggressive the lower the predicted contention level.

The predictive congestion control module runs on top of a generic rate-based linear increase/exponential decrease feedback control and is easily portable to other contexts. We show that a significant improvement in throughput can be obtained if SAC is active and we show that the relative performance improvement increases as long-range dependence is increased. Lastly, we show that as the number of connections engaging in SAC increases both fairness and efficiency are preserved.

*Supported in part by NSF grant NCR-9714707.
†Contact author; tel.: (765) 494-7821, fax.: (765) 494-0739.

1 Introduction

Recent measurements of local-area and wide-area traffic [8, 28, 42] have shown that network traffic exhibits variability at a wide range of scales. What is striking is the ubiquitousness of the phenomenon which has been observed in diverse networking contexts, from Ethernet to ATM, LAN and WAN, compressed video, and HTTP-based WWW traffic [8, 15, 23, 42]. Such scale-invariant variability is in strong contrast to traditional models of network traffic which show burstiness at short time scales but are essentially smooth at large time scales; i.e., they lack long-range dependence. Since scale-invariant burstiness can exert a significant impact on network performance, understanding the causes and effects of traffic self-similarity is an important problem.

In previous work [33, 34], we have investigated the causal and performance aspect of traffic self-similarity, and we have shown that self-similar traffic flow is an intrinsic property of networked client/server systems with heavy-tailed file size distributions and conjoint provision of low delay and high throughput is adversely affected by scale-invariant burstiness. From a queueing theory perspective, the principal distinguishing characteristic of long-range dependent traffic is that the queue length distribution decays much more slowly—i.e., polynomially—vis-à-vis short-range-dependent traffic sources such as Poisson sources which exhibit exponential decay. A number of performance studies [1, 2, 11, 29, 32, 34] have shown that self-similarity has a detrimental effect on network performance leading to increased delay and packet loss rate. In [18, 37], the point is advanced that for small buffer sizes or short time scales, long-range dependence has only a marginal impact. This is, in part, due to a saturation effect that arises when resources are overextended whereby the burstiness associated with short-range dependent traffic is sufficient—and, in many cases, dominant—to cause significant buffer overflow.

What is still in its infancy, however, is the problem of *controlling* self-similar network traffic. By the control of self-similar traffic, we mean the problem of modulating traffic flow such that network performance including throughput is optimized. Scale-invariant burstiness introduces new complexities into the picture which makes the task of providing quality of service (QoS) while achieving high utilization significantly more difficult. First and foremost, scale-invariant burstiness implies the existence of concentrated periods of high activity at a wide range of time scales which adversely affects congestion control. Burstiness at fine time scales is commensurate with burstiness observed for traditional short-range dependent traffic models. The distinquishing feature is burstiness at coarser time scales which induces extended periods of either overload or underutilization which degrades overall performance. However, on the flip side, long-range dependence, by definition, implies the existence of nontrivial correlation structure which may be exploitable for congestion control purposes, information to which current algorithms are impervious.

In this paper, we show the feasibility of "predicting the future" under self-similar traffic conditions with sufficient reliability such that the information can be effectively utilized for congestion control purposes. First, we show that long-range dependence can be on-line detected to predict future traffic levels and contention over time scales relevant to feedback congestion control. Second, we show that

a traffic modulation mechanism exists which is able to effectively exploit this information to improve network performance, in particular, throughput. The congestion control mechanism works by selectively applying aggressiveness using the predicted future when it is warranted, throttling the data rate upward if the predicted future contention level is low, being more aggressive the lower the predicted contention level. We show that the *selective aggressiveness* mechanism is of benefit even for short-range dependent traffic, however, being significantly more effective for long-range dependent traffic leading to comparatively large performance gains. We also show that as the number of connections engaging in SAC increases, both fairness and efficiency are preserved. The latter refers to the total throughput achieved across all SAC-controlled connections.

The rest of the paper is organized as follows. In the next section, we give a brief overview of self-similar network traffic and the specific set-up employed in this paper. In Section 3, we describe the predictability mechanism and its efficacy at extracting the correlation structure present in long-range dependent traffic. This is followed by Section 4 where we describe the SAC protocol and a refinement of the predictability mechanism for on-line, per-connection estimation. In Section 5 we show performance results of SAC and show its efficacy under different long-range dependence conditions and when the number of SAC connections is varied. We conclude with a discussion of current results and future work.

2 Preliminaries

2.1 Self-Similar Traffic: Basic Definitions

Let $(X_t)_{t\in\mathbb{Z}_+}$ be a time series which, for example, represents the trace of data flow at a bottleneck link measured at some fixed time granularity. We define the aggregated series $X_i^{(m)}$ as

$$X_i^{(m)} = \frac{1}{m}(X_{im-m+1} + \cdots + X_{im}).$$

That is, X_t is partitioned into blocks of size m, their values are averaged, and i is used to index these blocks.

Let $r(k)$ and $r^{(m)}(k)$ denote the autocorrelation functions of X_t and $X_i^{(m)}$, respectively. X_t is *self-similar*—more precisely, *asymptotically second-order self-similar*—if the following conditions hold:

$$r(k) \sim \text{const} \cdot k^{-\beta}, \tag{2.1}$$

$$r^{(m)}(k) \sim r(k), \tag{2.2}$$

for k and m large where $0 < \beta < 1$. That is, X_t is "self-similar" in the sense that the correlation structure is preserved with respect to time aggregation—relation (2.2)—and $r(k)$ behaves hyperbolically with $\sum_{k=0}^{\infty} r(k) = \infty$ as implied by (2.1). The latter property is referred to as *long-range dependence*.

Let $H = 1 - \beta/2$. H is called the *Hurst parameter*, and by the range of β, $1/2 < H < 1$. It follows from (2.1) that the farther H is away from $1/2$ the more long-range dependent X_t is, and vice versa. Thus the Hurst parameter acts as an indicator of the degree of self-similarity.

A test for long-range dependence can be obtained by checking whether H significantly deviates from 1/2 or not. We use two methods for testing this condition. The first method, the *variance-time plot,* is based on the slowly decaying variance of a self-similar time series. The second method, the R/S plot, uses the fact that for a self-similar time series, the *rescaled range* or R/S statistic grows according to a power law with exponent H as a function of the number of points included. Thus the plot of R/S against this number on a log-log scale has a slope which is an estimate of H. A comprehensive discussion of the estimation methods can be found in [4, 39].

A random variable X has a *heavy-tailed* distribution if

$$\Pr\{X > x\} \sim x^{-\alpha}$$

as $x \to \infty$ where $0 < \alpha < 2$. That is, the asymptotic shape of the tail of the distribution obeys a power law. The *Pareto* distribution,

$$p(x) = \alpha k^{\alpha} x^{-\alpha-1},$$

with parameters $\alpha > 0$, $k > 0$, $x \geq k$, has the distribution function

$$\Pr\{X \leq x\} = 1 - (k/x)^{\alpha},$$

and hence is clearly heavy-tailed.

It is not difficult to check that for $\alpha \leq 2$ heavy-tailed distributions have infinite variance, and for $\alpha \leq 1$, they also have infinite mean. Thus, as α decreases, a large portion of the probability mass is located in the tail of the distribution. In practical terms, a random variable that follows a heavy-tailed distribution can take on extremely large values with nonnegligible probability.

2.2 Structural Causality

In [33], we show that aggregate traffic self-similarity is an intrinsic property of networked client/server systems where the size of the objects (e.g., files) being accessed is heavy-tailed. In particular, there exists a linear relationship between the heavy-tailedness measure of file size distributions as captured by α—the shape parameter of the Pareto distribution—and the Hurst parameter of the resultant multiplexed traffic streams. That is, the aggregate network traffic that is induced by hosts exchanging files with heavy-tailed sizes over a generic network environment running "regular" protocol stacks (e.g., TCP, flow-controlled UDP) is self-similar, being more bursty—in the scale-invariant sense—the more heavy-tailed the file size distributions are. This relationship is shown in Figure 2.1. The relationship is robust with respect to changes in network resources (bandwidth, buffer capacity), topology, the influence of cross-traffic, and the distribution of interarrival times. We call this relationship between the traffic pattern observed at the network layer and the structural property of a distributed, networked system in terms of its high-level object sizes *structural causality* [33]. $H = (3 - \alpha)/2$ is the theoretical value predicted by the ON/OFF model [42]—a 0/1 renewal process with heavy-tailed ON or OFF periods—assuming *independent* traffic sources with no interactions due to sharing of network resources.

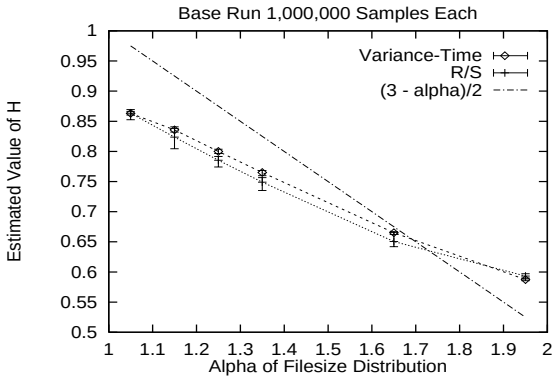

Figure 2.1: Hurst Parameter estimates (R/S and V-T) for α varying from 1.05 to 1.95

Structural causality is of import to self-similar traffic control since, one, it provides an environment where self-similar traffic conditions are easily facilitated—just simulate a client/server network—two, the degree of self-similar burstiness can be intimately controlled by the application layer parameter α, and three, the self-similar network traffic induced already incorporates the actions and modulating influence of the protocol stack since the observed traffic pattern is a direct consequence of hosts interchanging files whose transport was mediated through protocols (e.g., TCP, flow-controlled UDP) in the protocol stack. This provides us with a natural environment where the impact of control actions by a congestion control protocol can be discerned and evaluated under self-similar traffic conditions.

3 Predictability of Self-Similar Traffic

3.1 Predictability Set-Up

In this section, we will show that the correlation structure present in long-range dependent (LRD) traffic can be detected and used to predict the future over time scales relevant to congestion control. Time series analysis and prediction theory has a long history with techniques spanning a number of domains from estimation theory to regression theory to neural network based techniques to mention a few [3, 17, 22, 40]. In many senses, it is an "art form" with different methods giving variable performance depending on the context and modeling assumptions. Our goal is not to perform optimal time series prediction but rather to choose a simple, easy-to-implement scheme and use it as a reference for studying congestion control techniques and their efficacy at exploiting the correlation structure present in LRD traffic for improving network performance. Our prediction method, which is described next, is a time domain technique and can be viewed as an instance of Bayesian estimation.

Assume we are given a wide-sense stationary stochastic process $(\xi_t)_{t\in\mathbb{Z}}$ and two numbers $T_1, T_2 > 0$. At time t, we have at our disposal

$$a = \sum_{i\in[t-T_1,t)} q_i$$

where q_i is a sample path of ξ_t over time interval $[t - T_1, t)$. For notational clarity, let

$$V_1 = \sum_{i \in [t-T_1, t)} \xi_i, \qquad V_2 = \sum_{i \in [t, t+T_2)} \xi_i.$$

a may be thought of as the aggregate traffic observed over the "recent past" $[t - T_1, t)$ and V_1, V_2 are composite random variables denoting the recent past and near future. We are interested in computing the conditional probability

$$\Pr\{V_2 = b \mid V_1 = a\} \tag{3.1}$$

for b in the range of V_2. For example, if a represented a "high" traffic volume, then we may be interested in knowing what the probability of encountering yet another high traffic volume in the near future would be.

Assume ξ_t has finite mean and variance. Let $\mu_k = \mathbf{E}(V_k)$, $\sigma_k^2 = \mathbf{V}(V_k)$, $k = 1, 2$, where $\mathbf{E}$ and $\mathbf{V}$ are the expectation and variance operators, respectively. To make sense of "high" and "low," we will partition the range of V_k into eight[1] *levels*

$$(-\infty, \mu_k - 3\sigma_k),\ [\mu_k - 3\sigma_k, \mu_k - 2\sigma_k),\ [\mu_k - 2\sigma_k, \mu_k - \sigma_k),\ [\mu_k - \sigma_k, \mu_k),$$
$$[\mu_k, \mu_k + \sigma_k),\ [\mu_k + \sigma_k, \mu_k + 2\sigma_k),\ [\mu_k + 2\sigma_k, \mu_k + 3\sigma_k),\ [\mu_k + 3\sigma_k, +\infty).$$

We will define two new random variables L_1, L_2 where

$$\begin{aligned}
L_k = 1 &\iff V_k \in (-\infty, \mu_k - 3\sigma_k), \\
L_k = 2 &\iff V_k \in [\mu_k - 3\sigma_k, \mu_k - 2\sigma_k), \\
&\vdots \\
L_k = 8 &\iff V_k \in [\mu_k + 3\sigma_k, +\infty).
\end{aligned}$$

In other words, L_k is a function of V_k, $L_k = L_k(V_k)$, and it performs a certain quantization. Thus if $L_k \approx 1$ then the traffic level is "low" relative to the mean, and if $L_k \approx 8$, then it is "high." Although the central limit theorem does not (mathematically) apply here, in practice, it was observed that the above partition gave a normalized measure of *deviation*.

Returning to (3.1) and prediction, for certain values of T_1, T_2, we are interested in knowing the conditional probability densities

$$\Pr\{L_2 \mid L_1 = \ell\}$$

for $\ell \in [1, 8]$. If $\Pr\{L_2 \mid L_1 = 8\}$ were concentrated toward $L_2 = 8$, and $\Pr\{L_2 \mid L_1 = 1\}$ were concentrated toward $L_2 = 1$, then this information could be potentially exploited for congestion control purposes.

[1] In general, any number of levels. Eight levels were found to be sufficiently granular for prediction purposes.

3.2 Estimation of Conditional Probability Density

To explore and quantify the potential predictability of self-similar network traffic, we use TCP traffic traces collected in [33] whose Hurst parameter estimates are shown in Figure 2.1 as the main reference point. First, we use *off-line* estimation of *aggregate* throughput traffic which is then refined to *on-line* estimation of aggregate traffic using *per-connection* traffic when performing predictive congestion control. Other traces including those collected from flow-controlled UDP runs yield similar results. The traces used are each 10000 second long at 10ms granularity. They represent the aggregate traffic of 32 concurrent TCP Reno connections recorded at a bottleneck router.

We observe that the aggregate throughput series exhibits correlation structure at several time scales from 250ms to 20sec and higher. To estimate $\Pr\{L_2 \mid L_1 = \ell\}$ from the aggregate throughput series X_t, we segment X_t into

$$N = \frac{10000 \quad (\text{sec})}{T_1 + T_2 \ (\text{sec})}$$

contiguous nonoverlapping blocks of length $T_1 + T_2$ (except possibly for the last block), and for each block $j \in [1, N]$ compute the aggregate traffic V_1, V_2 over the subintervals of length T_1, T_2.

For $\ell, \ell' \in [1, 8]$, let $h_\ell \in [0, N]$ denote the total number of blocks such that $L_1(V_1) = \ell$ and let $h_{\ell'} \in [0, h_\ell]$ denote the size of the subset of those blocks such that $L_2(V_2) = \ell'$. Then

$$\Pr\{L_2 = \ell' \mid L_1 = \ell\} = \frac{h_{\ell'}}{h_\ell}.$$

Figure 3.1 shows the estimated conditional probability densities for $\alpha = 1.05, 1.95$ traffic for time scales 500ms, 1s, and 5s. In the following, $T_1 = T_2$.

For the aggregate throughput traces with $\alpha = 1.05$—Figure 3.1 (top row)—the 3-D conditional probability densities can be seen to be skewed diagonally from the lower left side toward the upper right side. This indicates that if the current traffic level L_1 is low, say $L_1 = 1$, chances are that L_2 will be low as well. That is, the probability mass of $\Pr\{L_2 \mid L_1 = 1\}$ is concentrated *toward* 1. Conversely, the plots show that $\Pr\{L_2 \mid L_1 = 8\}$ is concentrated *toward* 8. This is more clearly seen in Figure 3.2 (left) which shows two cross-sections, i.e., 2-D projections, reflecting $\Pr\{L_2 \mid L_1 = 1\}$ and $\Pr\{L_2 \mid L_1 = 8\}$.

For the aggregate throughput traces with $\alpha = 1.95$ (Figure 3.1 (bottom row)), on the other hand, the shape of the distribution does not change as the conditioning variable L_1 is varied. This is more clearly seen in the projections of $\Pr\{L_2 \mid L_1 = 1\}$ and $\Pr\{L_2 \mid L_1 = 8\}$ shown in Figure 3.2 (right). This indicates that for $\alpha = 1.95$ traffic observing the present (over the time scales considered) does not help much in predicting the future beyond the information conveyed by the fixed *a priori distribution*. Given the definition of L_k, the Gaussian shape of the marginal densities is consistent with short-range correlations making the central limit theorem approximately applicable over larger time scales. In both cases ($\alpha = 1.05, 1.95$), the shape of the distribution stays relatively constant across a wide range of time scales 500ms–20s. For $\alpha = 1.35, 1.65$ the predictability structure lies "in-between" (not shown here).

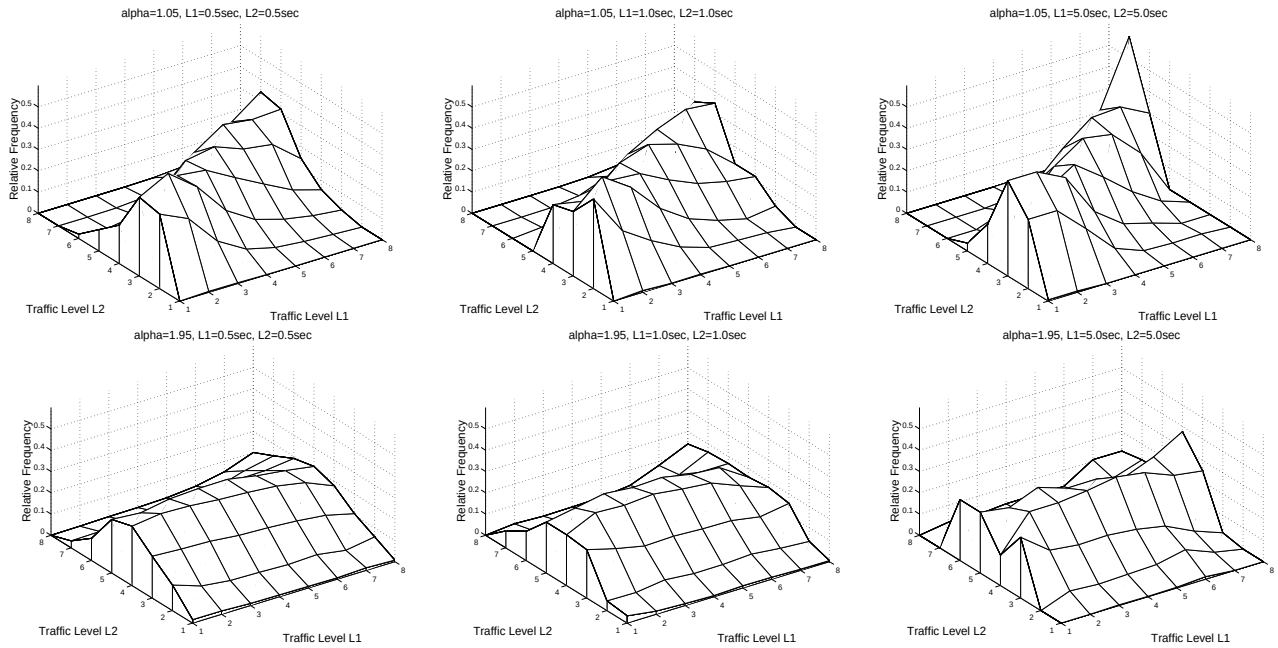

Figure 3.1: Top Row: Probability densities with L_2 conditioned on L_1 for $\alpha = 1.05$. Bottom Row: Probability densities with L_2 conditioned on L_1 for $\alpha = 1.95$.

3.3 Predictability and Time Scale

An important issue is how time scale affects predictability when traffic is long-range dependent. Going back to Figure 3.1 (top row), one subtle effect that is not easily discernible is that as time scale is increased the conditional probability densities $\Pr\{L_2 \mid L_1 = \ell\}$ become more *concentrated*. Given that $\Pr\{L_2 \mid L_1 = \ell\}$ is a function of T_1, T_2, we would like to determine at what time scale predictability is maximized.

One way to measure the "information content"—i.e., in the sense of randomness or unstructuredness—in a probability distribution is to compute its entropy. For a discrete probability density p_i, its *entropy* $S(p_i)$ is defined as $S(p_i) = \sum_i p_i \log 1/p_i$. In the case of our conditional density $\Pr\{L_2 \mid L_1 = \ell\}$,

$$S_\ell = -\sum_{\ell'=1}^{8} \Pr\{L_2 = \ell' \mid L_1 = \ell\} \log \Pr\{L_2 = \ell' \mid L_1 = \ell\}.$$

Thus entropy is maximal when the distribution is uniform and it is minimal if the distribution is concentrated at a single point. Since we are given a set of eight conditional probability densities, one for each $L_1 = 1, 2, \ldots, 8$, we define the *average entropy* $\bar{S}$ as

$$\bar{S} = \sum_{\ell=1}^{8} S_\ell / 8.$$

The average entropy remains a function of T_1, T_2, i.e., $\bar{S} = \bar{S}(T_1, T_2)$.

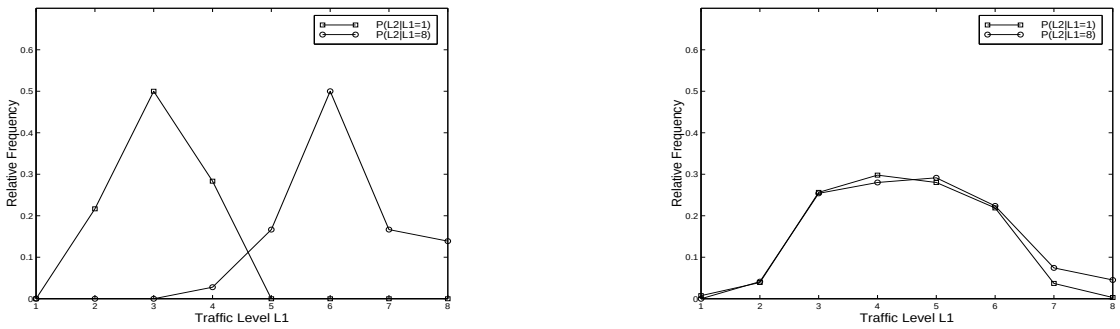

Figure 3.2: Left: Shifting effect of conditional probability densities $P(L_2|L_1 = 1)$ and $P(L_2|L_1 = 8)$ for $\alpha = 1.05$. Right: For $\alpha = 1.95$, the corresponding probabilities remain invariant.

Figure 3.3 plots $\bar{S}(T_1, T_2) = \bar{S}(T_1)$ (recall that $T_1 = T_2$) for the $\alpha = 1.05$ throughput series as a function of time scale or aggregation level T_1. Entropy is highest for small time scales in the range ~250ms, and it drops monotonically as T_1 is increased. Eventually, $\bar{S}(T_1)$ begins to flatten out near the 3–5 second mark reaching saturation, and stays so as time scale is further increased. From our analysis of various long-range dependent traffic traces, we find that the "knee" of the entropy curve is in the range of 1–5 seconds. Note that increasing T_1 further and further to gain small decreases in entropy brings forth with it an important problem, namely, if prediction is done over a "too long" time interval, then the information may not be effectively exploitable by various congestion control strategies. In the next section, we will show that one strategy—*selective aggressiveness*—is effective at exploiting the predictability structure found in the 1–5s range.

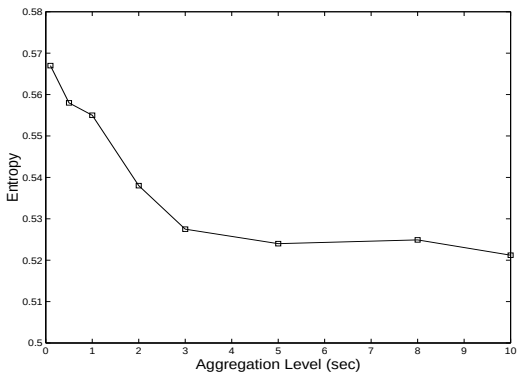

Figure 3.3: Average entropy $\bar{S}(T_1)$ plot for $\alpha = 1.05$ traffic as a function of time scale T_1.

4 SAC and Predictive Congestion Control

In this section we present a congestion control strategy called *Selective Aggressiveness Control* (SAC) and show its efficacy at exploting the predictability structure present in long-range dependent traffic for improving network performance. Our control algorithm is a form of "predictive congestion control"

whereby information about the future is utilized to make traffic control decisions. SAC is aimed to be robust, efficient, and portable such that it can be easily incorporated into existing congestion control schemes.

SAC's modus operandi is to complement and help improve the performance of existing reactive congestion controls. Toward this end, we set up a simple, generic rate-based feedback congestion control as a reference and let our control module "run on top" of it. SAC always respects the decision made by the underlying congestion control with respect to the directional change of the traffic rate—up or down—however, it may choose to adjust the *magnitude* of change. That is, if, at any time, the underlying congestion control decides to *increase* its sending rate, SAC will never take the opposite action and decrease the sending rate. Instead, what SAC will do is *amplify* or *diminish* the magnitude of the directional change based on its predicted future network state.

In a nutshell, SAC will try to aggressively soak up bandwidth if it predicts the future network state to be "idle," adjusting the level of aggressiveness as a function of the predicted idleness. We will show that the performance gain due to SAC is higher the more long-range dependent the network traffic is.

4.1 Congestion Control

4.1.1 Generic Feedback Congestion Control

Congestion control has been an active area of networking research spanning over two decades with a flurry of concentrated work carried out in the late '80s and early '90s [5, 6, 16, 19, 24, 25, 27, 30, 31, 35, 36, 38]. Gerla and Kleinrock [16] laid down much of the early groundwork and Jacobson [24] has been instrumental in influencing the practical mechanisms that have survived until today. A central part of the investigation has been the study of stability and optimality issues [5, 13, 24, 25, 30, 31, 35, 38] associated with feedback congestion control. A taxonomy for classifying the various protocols can be found in [43].

More recently, the delay-bandwidth product problem arising out of high-bandwidth networks and quality of service issues stemming from support of real-time multimedia communication [7, 10, 12, 20, 21, 41] have added further complexities to the problem with QoS reigning as a unifying key theme. One of the lessons learned from congestion control research is that end-to-end rate-based feedback control using various forms of linear increase/exponential decrease can be effective, and asymmetry in the control law needs to be preserved to achieve stability.

We will employ a simple, generic instance of rate-based feedback congestion control as a reference to help demonstrate the efficacy of selective aggressiveness control under self-similar traffic conditions. SAC is motivated, in part, by the simple yet important point put forth in [26] which shows that the conservative nature of asymmetric controls can, in some situations, lead to throughput smaller than those achieved by a "nearly blind" aggressive control. By applying aggressiveness selectively—based on the prediction of future network contention—we seek to offset some of the cost incurred for stability.

Let λ denote packet arrival rate and let γ denote throughput. Our *generic linear in-*

crease/exponential decrease feedback congestion control has a control law of the form[2]:

$$\frac{d\lambda}{dt} = \begin{cases} \delta, & \text{if } d\gamma/d\lambda > 0, \\ -a\lambda, & \text{if } d\gamma/d\lambda < 0, \end{cases} \tag{4.1}$$

where $\delta, a > 0$ are positive constants. Thus, if increasing the data rate results in increased throughput (i.e., $d\gamma/d\lambda > 0$) then increase the data rate linearly. Conversely, if increasing the data rate results in a decrease in throughput (i.e., $d\gamma/d\lambda < 0$) then exponentially decrease the data rate. In general, condition $d\gamma/d\lambda < 0$ can be replaced by various measures of *congestion*.

Of course, difficulties arise because (4.1) is, in reality, a delay differential equation (the feedback loop incurs a time lag) and the sign of $d\gamma/d\lambda$ needs to be reliably estimated. The latter can be implemented using standard techniques.

4.1.2 Unimodal Load-Throughput Relation

One item that needs further explanation is throughput γ. "Throughput" (in the sense of *goodput*) can be defined in a number of ways depending on the context, from *reliable throughput* (number of bits reliably transferred per unit time when taking into account reliability mechanism overhead), to *raw throughput* (number of bits transferred per unit time), to *power* (one of the throughput measures divided by delay). Raw throughput, denoted ν, is both easy to measure (just monitor the number of packets, in bytes, arriving at the receiver per unit time) and to attain (in most contexts $\nu = \nu(\lambda)$ is a monotone increasing function of λ, e.g., $M/M/1/n$), but it does not adequately discriminate between congestion controls that achieve a certain raw throughput without incurring high packet loss or delay and those that do.

For example, achieving reliability using ARQ with finite receiver and sender side buffers requires intricate control and coordination, and high packet loss can have a severe impact on the efficient functioning of such controls (e.g., TCP's Window control). In particular, for a given raw throughput, if the packet loss rate is high, this may mean that a significant fraction of the raw throughput is taken up by duplicate packets (due to early retransmissions) or by packets that will be dropped at the receiver side due to "fragmentation" and buffer overflow. Thus the reliable throughput associated with this raw throughput/packet loss rate combination would be low.

How severely packet loss impacts the throughput experienced by an application will depend on the characteristics of the application at hand. To better reflect such costs, we will use a throughput measure γ_k

$$\gamma_k = (1-c)^k \nu \tag{4.2}$$

that (polynomially) penalizes raw throughput ν by packet loss rate $0 \leq c \leq 1$ where the severity can be set by parameter $k \geq 0$. Thus raw throughput ν is a special instance of γ_k with $k = 0$. We will measure instantaneous throughput γ_k at the receiver and feed back to the sender for use in the

[2] We use continuous notation for expositional clarity.

control law (4.1). Figure 4.1 illustrates the relationship between γ_k and λ for a $M/M/1/n$ queueing system which shows that for $c > 0$ the load-throughput curve $\gamma_k = \gamma_k(\lambda)$ is *unimodal*. Note that c is a monotone decreasing function of λ while ν is monotone increasing. In the case of $M/M/1/n$ and most other network systems, raw bandwidth is upper bounded by the service rate or link speed—i.e., $\nu \leq \mu$—and thus most load-throughput functions of interest (not just (4.2)) will be unimodal due to the above monotonicity properties.

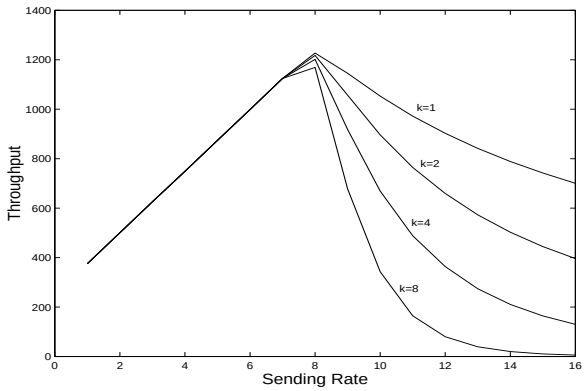

Figure 4.1: Unimodal load-throughput curve $\gamma_k = \gamma_k(\lambda)$ for $M/M/1/n$ system for $k = 1, 2, 4, 8$.

4.2 Selective Aggressiveness Control (SAC)

Assuming that future network contention is predictable with sufficient degree of accuracy, there remains the question of what to do with this information for performance enhancement purposes. The choice of actions, to a large measure, is constrained by the networking context and what degree of freedom it allows. In the traditional end-to-end congestion control setting, the network is a shared resource treated as a black box, and the only control variable available to a flow is its traffic rate λ.

In this paper, the target mechanism to be improved using predictability is the performance loss stemming conservative bandwidth usage during the *linear increase phase* of linear increase/exponential decrease congestion control algorithms [26]. Feedback congestion control protocols, including TCP, implement variants of this basic control law due to well-established stability reasons. In [26], however, it was shown in the context of TCP Reno that the asymmetry stemming from *linear* increase after *exponential* back-off ends up significantly underutilizing bandwidth such that, in some situations, a simple non-feedback control was shown to be more effective[3].

Given that linear increase/exponential decrease is widely used in congestion control protocols including TCP, we seek to target the *linear increase* part of such protocols such that, when deemed beneficial, and only then, a more aggressive bandwidth consumption is facilitated. This *selective* application of aggressiveness, when coupled with predictive capability, will hopefully lead to a more effective

[3]This potential problem was also recognized in Jacobson's seminal paper [24] which, in part, motivated TCP Tahoe's Slow Start feature.

use of bandwidth resulting in improved performance. Without selective, controlled application of aggressiveness, however, the gain from aggressiveness may be cancelled out (or even dominated) by its cost—aggressiveness, under high network contention conditions, can lead to deteriorated performance, even congestion collapse—thereby making predictability and its appropriate exploitation a nontrivial problem.

Our protocol—*Selective Aggressiveness Control* (SAC)—is composed of two parts, prediction and application of aggression, and they are described next.

4.2.1 Per-Connection On-Line Estimation of Future Contention

In the end-to-end feedback congestion control context, the two principal problems that a connection faces when estimating future network contention are:

(i) need to estimate "global" network contention using "local" *per-connection* information,

(ii) need to perform *on-line* prediction.

First, with respect to requirement (i), since the network is a black box as far as an end-to-end connection or flow is concerned, we cannot rely on internal network support such as congestion notification via router support to reveal network state information. Instead, we need to gleam, in our case, predict *future* network state using information obtained from a flow's input/output interaction with the network. For this to work, two assumptions need to hold in practice. One, due to the coupling induced by sharing of common resources, a connection's individual throughput when engaging in feedback congestion control (such as (4.1)) is correlated with the aggregate flow accessing the same resources. Two, aggregate traffic level, when partitioned according to the quantization scheme $L_k(V_k)$ of Section 3.1, is correlated to the contention level at the router that the aggregate traffic enters.

Second, with respect to requirement (ii), it turns out that *on-line* estimation of the conditional probability density $\Pr\{L_2 \mid L_1 = \ell\}$ is easily and efficiently accomplished using $O(1)$ cost update operations. On the sender side, SAC maintains a 2-dimensional array or table

$$\texttt{CondProb}[\cdot][\cdot]$$

of size 8×9, one row for each $\ell \in [1, 8]$. The last column of `CondProb`, `CondProb`$[\ell][9]$, is used to keep track of h_ℓ, the number of blocks observed thus far whose traffic level map to ℓ, i.e., $L_1(V_1) = \ell$ (see Section 3.1). For each $\ell' \in [1, 8]$, `CondProb`$[\ell][\ell']$ maintains the count $h_{\ell'}$. Since $\Pr\{L_2 = \ell' \mid L_1 = \ell\} = h_{\ell'}/h_\ell$, having the table `CondProb` means having the conditional probability densities.

All that is needed to maintain `CondProb` is a clock or alarm of period 2 which, starting at time $t = 0$, goes off at times

$$t \;=\; T_1,\; T_1 + T_2,\; T_1 + T_2 + T_1,\; T_1 + T_2 + T_1 + T_2,\; \ldots$$

If a feedback packet containing an instantaneous throughput γ measured at the receiver arrives during the period

$$[i(T_1 + T_2),\; i(T_1 + T_2) + T_1], \qquad i \geq 0,$$

it is added to V_1. When the alarm goes off at $t = i(T_1+T_2)+T_1$, V_1 is used to compute the updated mean μ_1 and standard deviation σ_1 of V_1. This can be done incrementally using $O(1)$ operations for both mean and standard deviation since variance can be expressed as a sum $\mathbf{E}\{(X - \mathbf{E}(X))^2\} = \mathbf{E}(X^2) - \mathbf{E}(X)^2$. Now $\ell = L_1(V_1)$ is computed using the updated μ_1, σ_1, and `CondProb`$[\ell][9]$ is incremented by 1. During interval

$$[i(T_1 + T_2) + T_1,\ (i + 1)(T_1 + T_2)], \qquad i \geq 0,$$

a similar operation is performed, however, now, with respect to V_2. At the end of the interval, the updated μ_2, σ_2 are computed, and $\ell' = L_2(V_2)$ is computed using the updated mean and standard deviation. Finally, `CondProb`$[\ell][\ell']$ is incremented by 1, and V_1, V_2 are reset to 0 to start the process anew. The number of operations within a time interval of length $T_1 + T_2$ is $O(1)$.

It should be noted that the conditional densities computed from `CondProb` at time t are *approximations* to the conditional probability densities computed off-line for the period $[0, t]$ since in the on-line algorithm running sums are used to compute μ_k, σ_k, $k = 1, 2$. Thus at time t when μ_k, σ_k are updated, the previous classifications made of V_k need not hold under the new L_k since the latter is a function of μ_k, σ_k.

4.2.2 Selective Application of Aggressiveness

SAC aims to "expedite" the bandwidth consumption process during the linear increase phase of linear increase/exponential decrease feedback congestion control algorithms—in our case, represented by the *generic feedback congestion control algorithm* (4.1)—when such actions are warranted.

The actuation part of the interface between SAC and (4.1) is defined as follows: Let λ_t denote the newly updated rate value at time t—by (4.1)—and let $\lambda_{t'}$ be the most recently ($t' < t$) updated rate value previous to t.

SAC (actuation interface):

1. If $\lambda_t > \lambda_{t'}$ then update $\lambda_t \leftarrow \lambda_t + \epsilon_t$.
2. Else do nothing.

Here, $\epsilon_t \geq 0$ is an aggressiveness factor that is determined by SAC based on the current state of `CondProb`. Notice that SAC kicks into action only during the linear increase phase of (4.1), i.e., when $\lambda_t > \lambda_{t'}$. The magnitude of ϵ_t determines the degree of aggressiveness, and it is determined as a function of the predicted network state as captured by `CondProb` and its conditional probability densities.

At time t, the algorithm used to determine ϵ is as follows: Let S_t be the aggregate throughput reported by the receiver via feedback over time interval $[t - T_1, t]$.

SAC (ϵ determination):

1. Let $\ell = L_1(S_t)$.

2. Compute $\bar{\ell}' = \mathbf{E}(L_2 \mid L_1 = \ell) = \sum_{\ell'=1}^{8} \ell' \Pr\{L_2 = \ell' \mid L_1 = \ell\}$.

3. Set $\epsilon = \epsilon(\bar{\ell}')$.

Thus, the current traffic level S_t is normalized and mapped to the index $\ell = L_1(S_t)$ which is then used to calculate the expectation of L_2 conditioned on ℓ, $\bar{\ell}'$. The latter is then finally used to index into a table $\epsilon(\bar{\ell}')$ called the *aggressiveness schedule*. The intuition behind the aggressiveness schedule $\epsilon(\cdot)$ is that if the expected future contention level is low (i.e., $\bar{\ell}'$ close to 1) then it is likely that applying a high level of aggressiveness will pay off. Conversely, if the expected future contention level is high (i.e., $\bar{\ell}'$ near 8) then applying a low level of aggressiveness is called for. One schedule that we frequently use is the *inverse schedule*

$$\epsilon(\bar{\ell}') = 1/\bar{\ell}'.$$

Other schedules of interest include the *threshold schedule* with threshold $\theta \in [1, 8]$ and aggressiveness factor θ^* where $\epsilon = \theta^*$ if $\bar{\ell}' \leq \theta$, and 0, otherwise.

$L_1 \backslash L_2$	1	2	3	4	5	6	7	8	$E[L_2 \mid \cdot]$	$\epsilon(\cdot)$
1	0.667	0.333	0	0	0	0	0	0	1.3	0.769
2	0.003	0.568	0.306	0.093	0.027	0.003	0	0	2.6	0.384
3	0	0.126	0.468	0.262	0.116	0.023	0.003	0	3.4	0.294
4	0	0.035	0.205	0.368	0.305	0.077	0.201	0	4.2	0.238
5	0	0.003	0.078	0.296	0.356	0.205	0.060	0.002	4.9	0.204
6	0	0	0.012	0.099	0.285	0.418	0.182	0.003	5.7	0.175
7	0	0	0.018	0.079	0.245	0.443	0.213	0.003	5.8	0.172
8	0	0	0	0	0.333	0	0.500	0.167	6.5	0.153

$L_1 \backslash L_2$	1	2	3	4	5	6	7	8	$E[L_2 \mid \cdot]$	$\epsilon(\cdot)$
1	0.155	0.116	0.155	0.233	0.165	0.078	0.087	0.097	3.8	0.263
2	0.043	0.058	0.179	0.272	0.257	0.128	0.054	0.008	4.3	0.232
3	0.023	0.049	0.132	0.306	0.273	0.161	0.054	0.003	4.5	0.222
4	0.020	0.058	0.135	0.274	0.286	0.167	0.055	0.004	4.5	0.222
5	0.012	0.039	0.134	0.273	0.307	0.183	0.044	0.008	4.6	0.217
6	0.017	0.058	0.141	0.243	0.325	0.166	0.044	0.007	4.5	0.222
7	0.008	0.042	0.126	0.211	0.322	0.195	0.088	0.008	4.8	0.208
8	0	0	0.167	0.233	0.233	0.300	0.067	0	4.8	0.208

Table 1: Top: A snapshot of `CondProb` for $\alpha = 1.05$, 10000s after connection has been established. Bottom: An snapshot of `CondProb` for $\alpha = 1.95$, 10000s after connection has been establised.

Figure 1 shows the `CondProb` table for two runs corresponding to $\alpha = 1.05$ (top) and $\alpha = 1.95$ (bottom) traffic conditions. The column containing h_ℓ has been omitted and the entries show actual relative frequencies rather than $h_{\ell'}$ counts for illustrative purposes. Clearly, the conditional probability densities are skewed diagonally for $\alpha = 1.05$ traffic whereas they are roughly invariant for $\alpha = 1.95$ traffic. The expected future contention level $\bar{\ell}' = \mathbf{E}(L_2 \mid L_1 = \ell)$ and aggressiveness schedule (inverse)

are shown as separate columns. For $\alpha = 1.05$ traffic, the expected future contention level $E[L_2|\cdot]$ varies over a wide range which is a direct consequence of the predictability—i.e., skewedness—present in the correlation structure. For $\alpha = 1.95$ traffic, however, $E[L_2|\cdot]$ is fairly "flat" indicating that conditioning on the present does not aid significantly in predicting the future.

5 Simulation Results

5.1 Congestion Control Evaluation Set-Up

We use the LBNL Network Simulator, *ns* (version 2), as the basis of our simulation environment. *ns* is an event-driven simulator derived from Keshav's REAL network simulator supporting several flavors of TCP and router packet scheduling algorithms. We have modified *ns* in order to model a bottleneck network environment where several concurrent connections are multiplexed over a shared bottleneck link. A UDP-based unreliable transport protocol was added to the existing protocol suite, and our congestion control and predictive control were implemented on top of it.

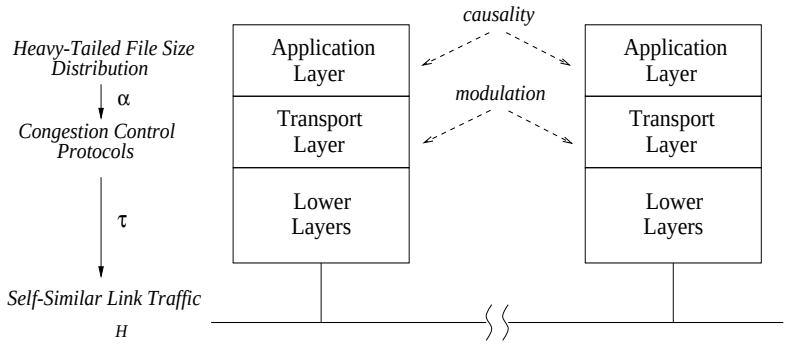

Figure 5.1: Transformation of the heavy-tailedness of file size distribution property at the application layer via the action of the transport layer into its manifestation as self-similar aggregated traffic at the link layer.

An important feature of the set-up is the mechanism whereby *self-similar traffic conditions* are induced in the network. One possibility is to have a host inject self-similar time series into the network. We follow a different approach based on the notion of *structural causality* (see Section 2.2) whereby we make use of the fact that in a networked client/server environment with clients interactively accessing files or objects with heavy-tailed sizes from servers across the network leads to aggregate traffic that is self-similar [33]. Most importantly, this mechanism is robust and holds when the file transfers are mediated by transport layer protocols executing reliable flow-controlled transport (e.g., TCP) or unreliable flow-controlled transport. The separation and isolation of "self-similar causality" to the highest layer of the protocol stack allows us to interject different congestion control protocols in the transport layer, discern their influence, and study their impact on network performance. This is illustrated in Figure 5.1.

Figure 5.2 shows a 2-server, n-client ($n \geq 33$) network configuration with a bottleneck link connecting gateways G_1 and G_2. The link bandwidths were set at 10Mbps and the latency of each link was set to 5ms. The maximum segment size was fixed at 1kB for all runs. Some of the clients engage in interactive transport of files with heavy-tailed sizes across the bottleneck link *to the servers* (i.e., the nomenclature of "client" and "server" are reversed here), sleeping for an exponential time between successive transfers. Others act as *infinite sources* (i.e., they have always data to send) executing the generic linear increase/exponential decrease feedback congestion control—with and without SAC—in the protocol stack trying to maximize throughput. For any reasonable assignment of bandwidth, buffer size, mean file request size, and other system parameters, we found that by either adjusting the number of clients or the mean of the idle time distribution between successive file transfers appropriately, any target contention level could be achieved.

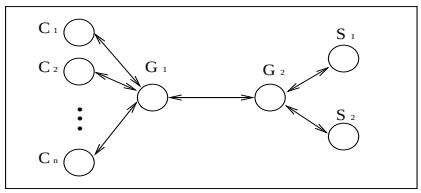

Figure 5.2: Network configuration with bottleneck link (G_1, G_2).

In a typical configuration, the first 32 connections served as "background traffic" transferring files from clients to servers (or sinks) where the file sizes were drawn from Pareto distributions with shape parameter $\alpha = 1.05, 1.35, 1.65, 1.95$. As in [33], there was a linear relation between α and the long-range dependence of aggregate traffic observed at the bottleneck link (G_1, G_2) as captured by the Hurst parameter H. H was close to 1 when α was near 1, and H was close to 1/2 when α was near 2. The 33rd connection acted as an infinite source trying to maximize throughput by running the generic feedback control, with or without SAC. In other settings, the number of congestion-controlled infinite sources was increased to observe their mutual interaction and the impact on fairness and efficiency. A typical run lasted for 10000 or 20000 seconds (simulated time) with traces collected at 10ms granularity.

5.2 Per-Connection On-Line Predictability

5.2.1 Predictability Structure

One of the first items to test was estimation of the conditional probability densities $\Pr\{L_2 \mid L_1\}$ using the per-connection, on-line method described in Section 4.2.1. We observe the same skewed diagonal shift characteristics as seen in the off-line case for $\alpha = 1.05$ traffic and the relatively invariant shape of the probability densities for $\alpha = 1.95$ traffic (we omit the 3-D plots due to space constraints). Also, as in the off-line case, as we increase the time scale (i.e., T_1) from 500ms to 1s and higher, for $\alpha = 1.05$ traffic the probability densities become more concentrated thus increasing the accuracy of predictability. Figure 5.3 (left) shows the shifting effect of the conditional probabilities for $\alpha = 1.05$

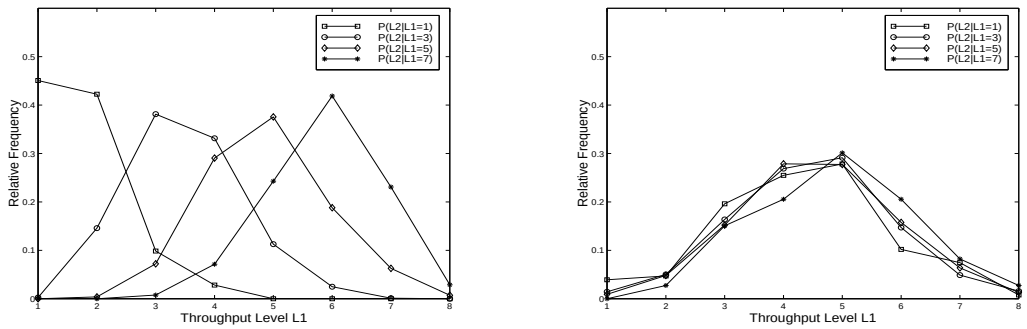

Figure 5.3: Left: Shifting effect of conditional probability densities $P(L_2|L_1 = 1)$ and $P(L_2|L_1 = 8)$ under $\alpha = 1.05$ background traffic. Right: Shifting effect of conditional probability densities $P(L_2|L_1 = 1)$ and $P(L_2|L_1 = 8)$ under $\alpha = 1.95$ traffic.

traffic via a 2-D projection that shows the marginal densities. Whereas the shifting effect is evident for $\alpha = 1.05$ traffic, for $\alpha = 1.95$ traffic (Figure 5.3 (right)) the probability densities stay largely invariant.

5.2.2 Time Scale

In Section 3.2, in connection with off-line estimation, we showed using entropy calculations that the conditional probability densities became more concentrated as time scale was increased flattening out eventually near the 4–5s mark. We observe a similar behavior with respect to the entropy curve in the on-line case (omitted due to space constraints). Locating the knee of the entropy curve is of import for on-line prediction and its use in congestion control since the size of the time scale will influence whether a certain procedure for exploiting predictability will be effective or not. As an extreme case in point, if the time scale of prediction were, say 1000s, it is difficult to imagine a mechanism that would be able to effectively exploit this information for congestion control purposes—too many changes may be occurring during such a time period that may be both favourable and detrimental to a constant control action yielding a zero net gain.

On the other hand, if control actions capable of spanning large time frames such as bandwidth reservation or pricing-based admission control were made part of the model, then even large time scale predictability may be exploited, with some effectiveness, for performance enhancement purposes. In the present context, we set $T_1 = 2$s when incorporating predictability into feedback congestion control via SAC. Our experience with different traffic traces suggests that the knee of the entropy curve, for many practical situations, may be located in the 1–6s range.

5.2.3 Convergence Rate of On-Line Conditional Probabilities

When using conditional probability densities computed from per-connection, on-line estimations, it becomes important to know when the estimations have converged or stabilized. If inaccurate information is used for selective aggressiveness control, it is possible that rather than helping improve performance, it may hurt performance.

SAC uses a distance measure to decide whether a particular conditional probability density is stable enough to be used for congestion control or not. Let $\texttt{CondProb}_t$, $\texttt{CondProb}_{t'}$ be two instances of the conditional probability density table measured at time instances $t > t'$ at least $T_1 + T_2$ apart. Then for each L_1 condition $\ell \in [1, 8]$, SAC computes

$$||\texttt{CondProb}_t[\ell][\cdot] - \texttt{CondProb}_{t'}[\ell][\cdot]||_2 < \Theta,$$

and if the check passes, allows this particular conditional probability density to be used in the actuation part of SAC when updating the data rate (see Section 4.2.2). Here, $\Theta > 0$ is an accuracy parameter and $||\cdot||_2$ is the "L_2" norm (not to be confused with the random variable L_2).

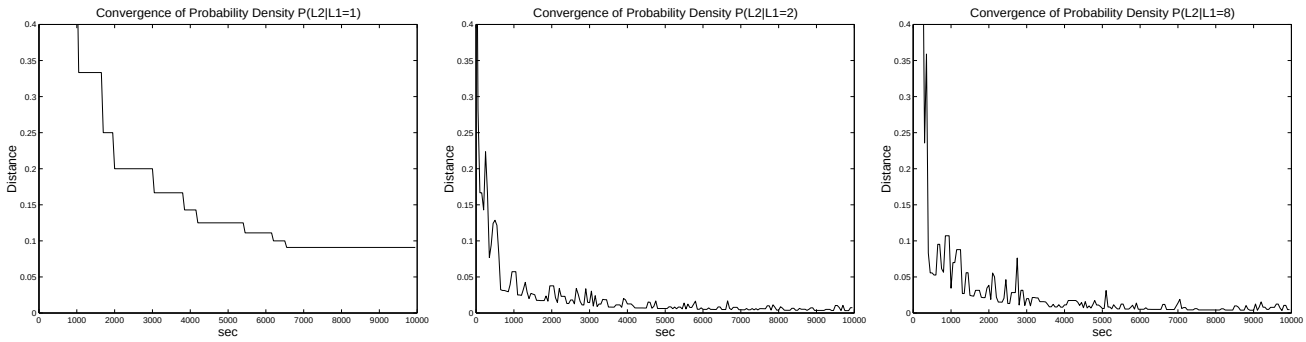

Figure 5.4: Convergence of conditional probability densities for $\alpha = 1.05$ traffic: $\ell = 1$ (left), $\ell = 2$ (middle), and $\ell = 8$ (right).

Figure 5.4 depicts the convergence rate of three conditional probability densities conditioned on $\ell = 1, 2,$ and 8 by plotting the distance measure computed above. The middle ($\ell = 2$) and right ($\ell = 8$) figures are the typical plots whose convergence rate is followed by the ones in the range $2 < \ell < 8$ as well. The leftmost plot ($\ell = 1$), however, is atypical and only holds for condition $L_1 = 1$. This mainly stems from the fact that for condition $L_1 = 1$, very few sample points (< 100) arise even during a 10000s run, and as a result, the estimated probabilities are volatile. This, in turn, can be attributed to the fact that we are computing statistics for a long-range dependent process, and it is well-known that for such processes a very large number of samples is needed to compute even its first-order statistics accurately. A related discussion can be found in [9].

The fact that this volatility stems from too few sample points also delimits the impact of this phenomenon. Namely, by Amdahl's law, the instances (over time) where the conditional probability $\Pr\{L_2 \mid L_1 = 1\}$ may have been invoked are so few to begin with so that the loss from not having taken any actions at those instances are negligible with respect to overall performance.

5.3 Performance Measurement of SAC

5.3.1 Incremental Gain of Selective Aggressiveness

Unimodal Throughput Curve In this section, we evaluate the relative performance of SAC and its

predictability gain. We measure the *incremental* benefit gained by applying aggressiveness *selectively*, first, by applying it only when the chances for benefit are highest (i.e., $\bar{\ell}' = \mathbf{E}(L_2 \mid L_1 = \ell) = 1$ for some $\ell \in [1, 8]$), then second highest ($\bar{\ell}' = 2$), and so on. Eventually, we expect to hit a point when the cost aggressiveness outweighs its gain, thus leading to a net decrease in throughput as the stringency of selectivity is further relaxed.

This phenomenon can be demonstrated using the *threshold aggressiveness schedule* of SAC (see Section 4.2.2) where aggressive action is taken if and only if $\bar{\ell}' \leq \theta$ where θ is the aggressiveness threshold. Figure 5.5 shows the throughput vs. aggressiveness threshold curve for threshold values in the range $1 \leq \theta \leq 8$ for $\alpha = 1.05$ traffic. We observe that the gain is highest when going from $\theta = 1$ to 2, then successively diminishes until it turns to a net loss thereby decreasing throughput. If $\theta = 8$, then this corresponds to the case where aggressiveness is applied at all times, i.e., there is no selectivity.

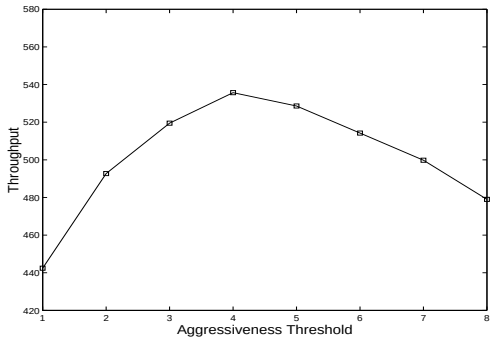

Figure 5.5: Unimodal throughput curve as a function of aggressiveness threshold θ for $\alpha = 1.05$ traffic.

Monotone Throughput Curve Although the unimodal, dome-shaped throughput curve (as a function of the aggressiveness threshold) is a representative shape, two other shapes—monotonically increasing or decreasing—are possible depending on the network configuration. The shape of the curve is dependent upon the relative magnitude of available resources (e.g., bandwidth) vs. the magnitude of aggressiveness as determined by the aggressiveness schedule $\epsilon(\cdot)$. If resources are "plentiful" then aggressiveness is least penalized and it can lead to a monotonically increasing throughput curve. On the other hand, if resources are "scarce" then aggressiveness is penalized most heavily and this can result in a monotonically decreasing throughput curve. This phenomenon is shown in Figure 5.6.

Figure 5.6 shows the throughput curves under the same network configuration except that the available bandwidth is decreased from the leftmost to the rightmost figure. This is affected by increasing the background traffic level from 2.5Mbps (left) to 5Mbps (middle) to 7.5Mbps (right). We observe that the curve's shape transitions from monotone increasing to unimodal dome-shaped to monotone decreasing. In addition, due to the decrease in available bandwidth, overall throughput drops as the background traffic level is increased.

Figure 5.7 shows the change in the shape of the throughput curve as the aggressiveness schedule $\epsilon(\cdot)$ is shifted (or translated) upwards—i.e., made overall more aggressive—by 0.5, 2.0, 4.0, and 20.0 while keeping everything else fixed. We observe that an overall increase in the magnitude of aggressiveness

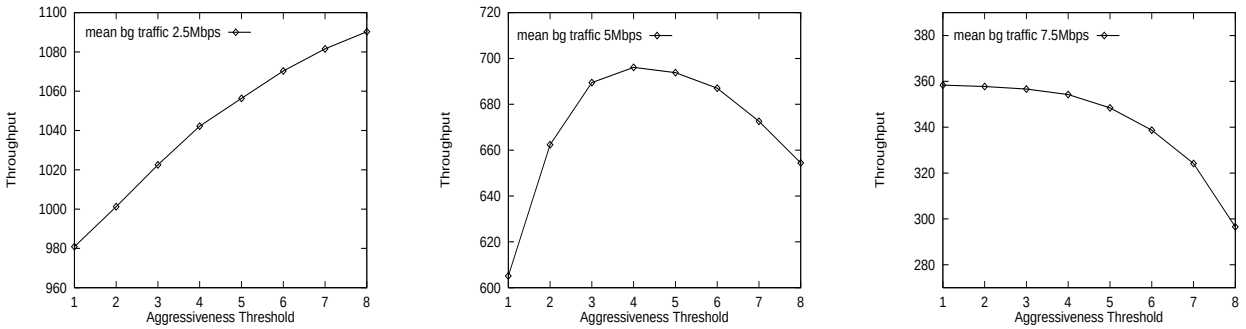

Figure 5.6: Shape of throughput curve as a function of aggressiveness threshold for three levels of background traffic 2.5Mbps (left), 5Mbps (middle), and 7.5Mbps (right).

can help improve throughput transforming a monotone increasing throughput curve into a unimodal curve whose maximum throughput has increased. However, as the overall aggressiveness level is further increased, the cost of aggressiveness begins to outweigh its benefit and we observe a downward shift in the unimodal throughput curve.

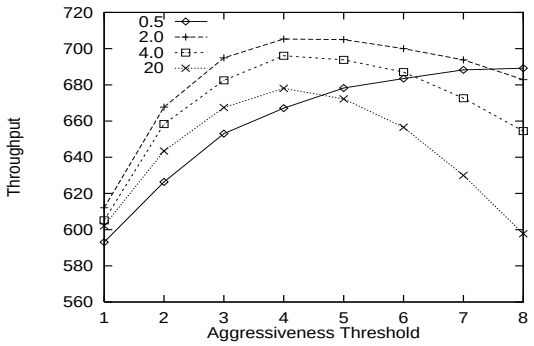

Figure 5.7: Change in shape of throughput curve as the aggressiveness schedule $\epsilon(\cdot)$ is shifted (upwards) by 0.5, 2.0, 4.0, 20.0.

SAC is designed to operate under all three network conditions finding a near-optimum throughput in each case. The most challenging task arises when the network configuration leads to a unimodal throughput curve for which finding the maximum throughput is least trivial. That is, neither blindly applying aggressiveness nor absteining from it are optimal strategies. SAC's adaptivity is also useful in nonstationary situations where the network configuration can shift from one quasi-static throughput state to another.

5.3.2 Perfect Prediction, Uncertainty, and Aggressiveness

Now that we have shown that selective aggressiveness can help but indiscriminate aggressiveness can hurt, we seek to understand three further aspects of SAC, one, how much performance is gained by applying selective aggressiveness (vis-à-vis not applying at all), two, how much performance is lost due

to prediction inaccuracies, and three, what is a practical aggressiveness schedule to use since we cannot assume to know the aggressiveness threshold for which maximum throughput is achieved (when using a threshold schedule).

The practical aggressiveness schedule that we found effective is the *inverse schedule* given by $\epsilon(x) = 1/x$. That is, the magnitude of aggressiveness is inverse-proportionally diminished as a function of the expected future traffic level. To measure the performance loss due to inaccuracies arising from using per-connection on-line prediction of future traffic levels, we observe the performance of SAC when, instead of using the on-line `CondProb` table, a perfect knowledge of future aggregate traffic is assumed and employed in conjunction with the inverse schedule. Finally, to compare the net gain of having used a practical version of SAC—in this case, predicted future using per-connection on-line table and inverse aggressiveness schedule—we observe the generic linear increase/exponential decrease feedback congestion control without SAC active.

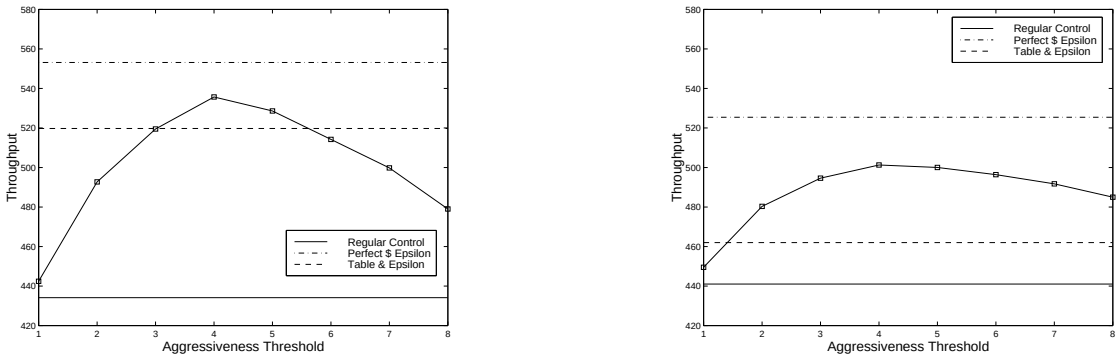

Figure 5.8: Left: The horizontal lines show throughput when different control strategies are employed (top line: perfect prediction with inverse schedule; middle line: on-line table with inverse schedule; bottom line: generic linear increase/exponential decrease congestion control without SAC) for $\alpha = 1.05$ traffic. Right: Corresponding throughput plot for $\alpha = 1.95$ traffic.

Figure 5.8 (left) shows the original throughput vs. threshold schedule curve superimposed with the throughput achieved by using SAC with perfect future knowledge and inverse aggressiveness schedule (topmost line), using SAC with predicted future and inverse schedule (middle line), and using the generic linear increase/exponential decrease feedback congestion control without SAC (bottom line). We observe that the generic feedback congestion control performs worst among the four—we are counting the family of SAC algorithms for the threshold schedule as one—which is mainly due to the costly nature of exponential backoff when coupled with conservative linear increase. For our purposes, the absolute magnitudes do not matter so much as the relative magnitudes which demonstrate a qualitative performance relationship. SAC with perfect information and inverse epsilon schedule achieves the highest throughput (even higher than the peak threshold schedule throughput) and SAC with predicted future and inverse schedule achieves a performance level in between. Figure 5.8 (right) shows the corresponding plots for $\alpha = 1.95$ traffic where a similar ordering relation is observed.

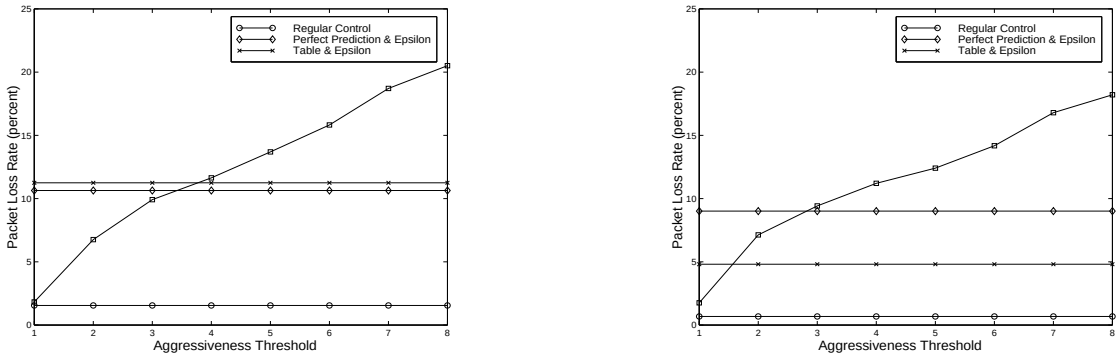

Figure 5.9: Left: Corresponding packet loss rates for the control strategies shown in Figure 5.8 for $\alpha = 1.05$ background traffic. Right: Corresponding packet loss rates for $\alpha = 1.95$.

Figure 5.9 shows the packet loss rates corresponding to the throughput plots shown in Figure 5.8. As expected, for the threshold schedule, packet loss rate increases monotonically as the aggressiveness threshold is increased. The generic (or regular) linear increase/exponential decrease congestion control incurs the least packet loss rate among the controls due to its conservativeness in the linear increase phase, albeit, at the cost of reduced throughput. Comparing Figures 5.9 (left) and (right) we observe that the overall packet loss rates for $\alpha = 1.05$ traffic is higher than that of $\alpha = 1.95$ traffic which is expected due to the higher level of self-similar burstiness.

5.3.3 Impact of Long-Range Dependence

The previous results, in addition to demonstrating a specific way to utilize correlation structure in self-similar traffic, showed that selective aggressiveness when coupled with predictability can lead to performance improvement above and beyond what a generic linear increase/exponential decrease feedback congestion control can achieve. The latter is of import since one of the practical applications of SAC is targeted at improving the performance of existing protocols.

In this section, we show that the relative performance gain due to selective aggressiveness control and predictability grows as long-range dependence increases. Figure 5.10 compares the relative performance gain stemming from employing predicted inverse schedule SAC for $\alpha = 1.05$ and $\alpha = 1.95$ background traffic. First, note that the throughput level for the genetic feedback congestion control is higher for $\alpha = 1.95$ traffic than $\alpha = 1.05$ traffic. This is as expected since self-similar burstiness is known to lead to degraded performance unless resources are overextended at which point the burstiness associated with short-range dependent traffic is dominant in determining queueing behavior. More importantly, we observe that the throughput gain relative to the generic feedback congestion control is about 20% in the $\alpha = 1.05$ case vs. about 4% for the $\alpha = 1.95$ case. This indicates that self-similar burstiness—although detrimental to network performance, in particular, QoS—possesses structure that can be exploited to dampen its negative impact. In fact, the more long-range dependent, the more structure there is to exploit effectively.

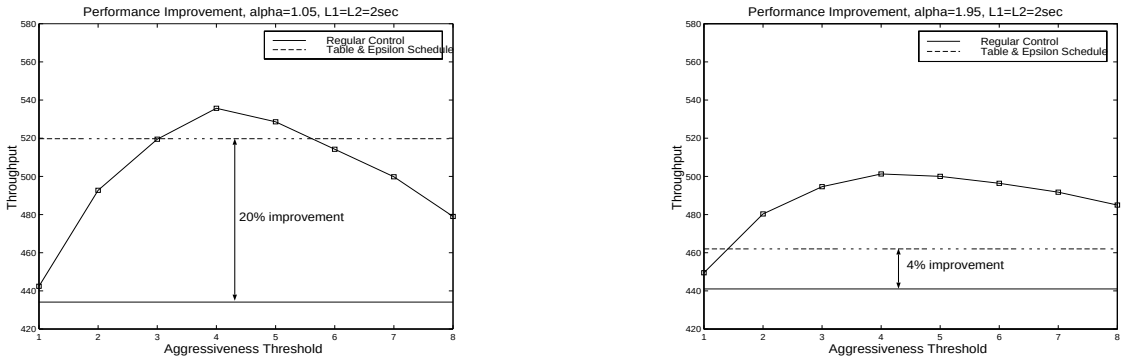

Figure 5.10: Left: Under $\alpha = 1.05$ traffic, the performance improvement is about 20% when using SAC with on-line table and inverse schedule. Right: Under $\alpha = 1.95$ traffic, the performance improvement is only 4%.

An important point to note is that we have held the mean of the background traffic levels for both $\alpha = 1.05$ and $\alpha = 1.95$ constant to achieve comparability. This is a nontrivial matter since for $\alpha = 1.05$, the mean traffic level estimated by using the Pareto distribution will overestimate the sample mean observed in practice, even if the system is run for 10000 seconds. Figure 5.11 shows the predictability gain in terms of throughput achieved for four background traffic cases $\alpha = 1.05, 1.35, 1.65, 1.95$. Interestingly, the throughput gain shows a superlinear increase as α approaches 1 (i.e., becomes more long-range dependent).

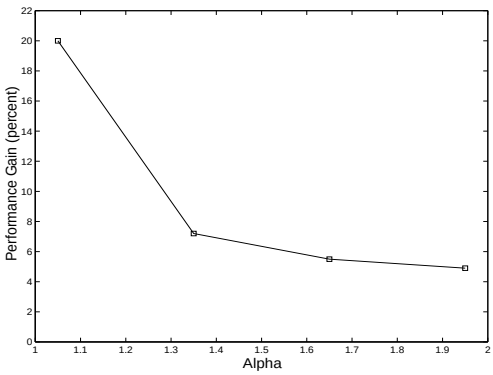

Figure 5.11: Performance gain due to predictability for $\alpha = 1.05, 1.35, 1.65, 1.95$ background traffic.

5.3.4 Convergence Rate and Performance

In Section 5.2.3 we have shown the rapid convergence rate of the on-line conditional probability table. The faster the convergence, the earlier the conditional probabilities can be employed for congestion control purposes resulting in higher throughput gain. Other things being equal, early activation of SAC induces a trade-off relation between the benefit obtained by applying predictive information for congestion control and the cost of engaging SAC based on possibly inaccurate conditional probability

estimates.

Figure 5.12 (left) shows the impact of inaccuracies in the conditional probability density estimates on performance. The top graph plots throughput as a function of *training time*—i.e., time spent in estimating the conditional probability densities—when the conditional probability table is subsequently fixed and used in a 10000 second throughput measurement run. This allows us to assess the impact of inaccurate prediction estimates on throughput performance. As the top graph shows, convergence is rapid after which the incremental gain obtained via further accuracies saturates. Notice that due to rapid convergence, the net gain in throughput due to increased prediction accuracy is below 5%. Contrast this with the bottom two graphs of Figure 5.12 (left) which show 10000 second throughput measurements when the conditional probability table used is that of $\alpha = 1.95$ traffic (trained over 10000 seconds) and random, respectively. The gap shows that even inaccurate prediction estimates are significantly more useful than random or otherwise-structured information for performance enhancement purposes.

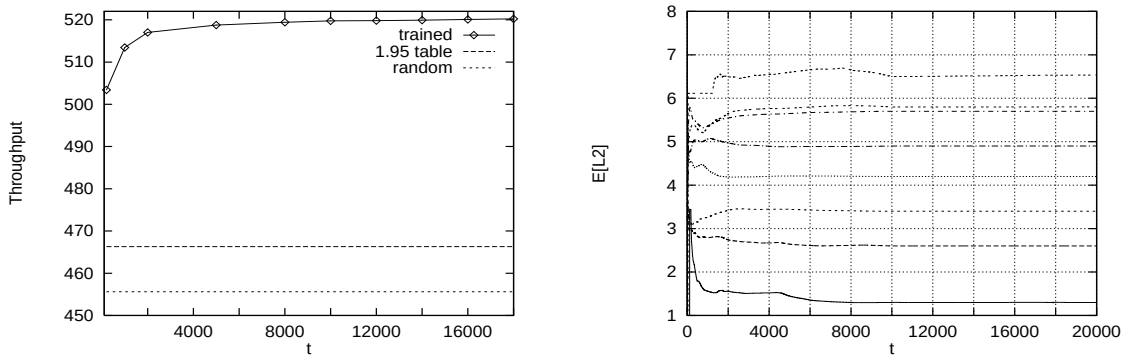

Figure 5.12: Left: Throughput as a function of SAC conditional probability table training time. Also shown: SAC throughput with random and $\alpha = 1.95$ conditional probability table. Right: Convergence property of $\mathbf{E}(L_2 \mid \cdot)$. Fast convergence to linear order $\mathbf{E}(L_2 \mid L_1 = 1) < \mathbf{E}(L_2 \mid L_1 = 2) < \cdots < \mathbf{E}(L_2 \mid L_1 = 8)$.

The fast convergence property can be further explained by Figure 5.12 (right) which plots the computed conditional expectation $\mathbf{E}(L_2 \mid \cdot)$ as a function of training time. Recall that in both the threshold and inverse schedules $\mathbf{E}(L_2 \mid \cdot)$ (quantized or not)—and not the conditional probability table proper—is used in computing the aggressiveness level. Figure 5.12 (right) shows that the functional $\mathbf{E}(L_2 \mid \cdot)$ quickly converges to the linear ordering $\mathbf{E}(L_2 \mid L_1 = 1) < \mathbf{E}(L_2 \mid L_1 = 2) < \cdots < \mathbf{E}(L_2 \mid L_1 = 8)$ as would be expected by the skewedness of the 3-D conditional probability densities. The magnitudes of $\mathbf{E}(L_2 \mid \cdot)$, after some undulation, stabilized to fixed values.

The quick establishment of the linear ordering property and the convergence of $\mathbf{E}(L_2 \mid \cdot)$ to fixed values leaves open the possibility that *a priori* conditional probabilities may be used for predictive purposes which is especially useful for short-lives connections for which per-connection conditional probability tables are impossible to establish.

5.3.5 Multiple Concurrent SAC Connections

The SAC protocol is designed to run in shared network environments where different connections compete for available resources. In this section, we investigate the behavior of the SAC protocol with respect to fairness and efficiency when multiple connections engage in SAC. The results are based on the same set-up as in Figure 5.2 except that we increase the bottleneck link bandwidth to 20Mbps to accommodate up to 10 SAC connections. The mean traffic rate of the first 32 connections—i.e., non-SAC background traffic sources—is kept at 5Mbps. We increase the number of SAC connections from 1 to 10 (33rd connection and beyond) and observe whether bandwidth is shared fairly and efficiently. The latter refers to the question of whether the total throughput achieved across all SAC connections remains conserved—increased competition can create overhead and inefficiencies—as the number of SAC connections is increased.

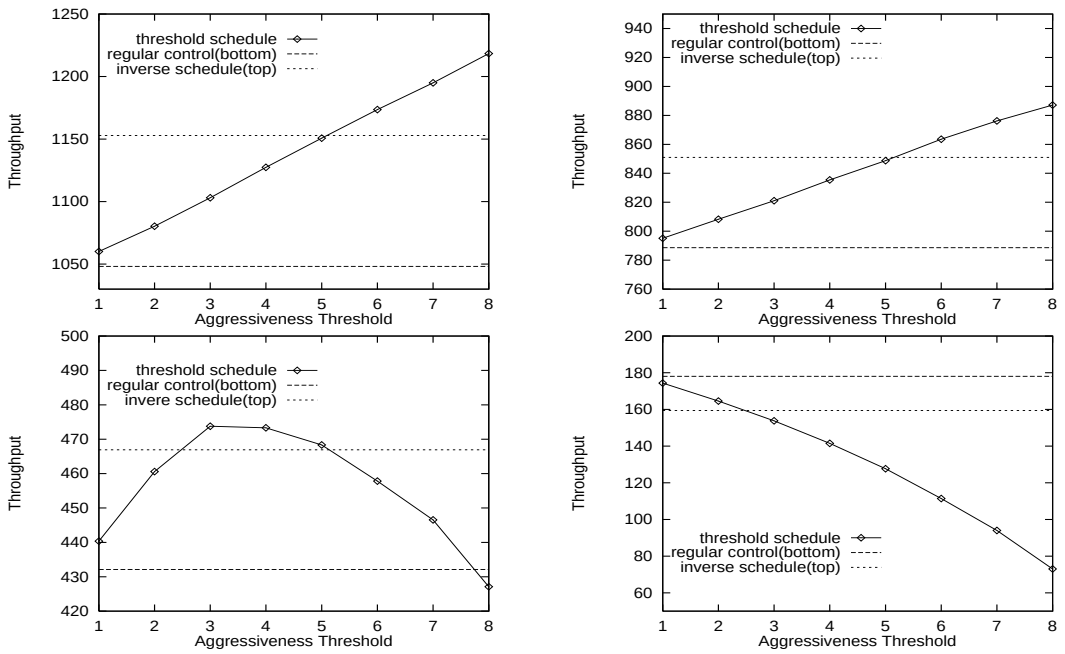

Figure 5.13: Average (per-connection) throughput as a function of aggressiveness threshold for multiple SAC connections. The top horizontal line shows the throughput achieved with the inverse schedule. The bottom line represents the throughput of the generic linear increase/exponential decrease feedback congestion control. Top left: single SAC connection; top right: two SAC connections; bottom left: four SAC connections; bottom right: ten SAC connections.

Figure 5.13 depicts *average per-connection throughput* as a function of the aggressiveness threshold for one (top-left), two (top-right), four (bottom-left), and ten (bottom-right) SAC connections. Superimposed we also show the throughput achieved by the inverse schedule and generic feedback congestion control, respectively. First, we observe that the shape of the throughput curve changes from monotonically increasing to unimodal to monotonically decreasing as the number of SAC connections

is increased. This is to be expected since, other things being equal, increasing the number of SAC connections amplifies the net aggressiveness level since there is no distributed control or cooperation among the SAC flows to maintain a constant overall aggressiveness level. Second, since we plot average per-connection throughput, we observe the per-connection throughput drop accordingly. Third, we observe that the performance of the threshold schedule eventually deteriorates below that of the generic feedback congestion control while the performance of the inverse schedule stays at a high level.

Threshold Schedule	1	2	3	4	5	6	7	8
1 connection	1066.1	1080.3	1103.0	1127.4	1150.8	1173.6	1194.9	1218.4
2 connections	1591.7	1617.6	1644.2	1672.7	1698.5	1728.6	1755.4	1776.1
4 connections	1765.5	1845.7	1896.4	1893.8	1877.3	1832.2	1788.8	1712.6
8 connections	1800.3	1750.3	1653.1	1537.6	1376.2	1230.6	1051.7	869.4
10 connections	1748.9	1649.0	1540.2	1417.9	1276.4	1109.2	925.7	743.0

Table 2: Total throughput across all SAC connections for the threshold schedule as the number of SAC connections is increased.

Table 2 shows the total throughput achieved across all SAC connections for the threshold schedule as the number of SAC connections is increased. For each threshold level, we observe a unimodal change in throughput as the number of connections is increased from 1 to 10 with the peak occurring earlier the higher the threshold level. This indicates a trade-off relation whereby, at first, the net increase in aggressiveness due to the increased number of SAC connections leads to a net increase in total SAC throughput. However, as the number of SAC connections is further increased, the amplification of the overall aggressiveness level asserts a negative impact on throughput eventually yielding a net decrease. A similar phenomenon is observed for the inverse schedule which is shown in Figure 3. The onset of the peak is a function of available resources and it can be further delayed by decreasing the overall aggressiveness of each connection. Note that the multiple connection throughput behavior is achieved for the network configuration shown in Figure 5.2 which, due to its uniform link latencies, can be prone to synchronization effects [14].

Inverse Schedule	1 conn.	2 conn.	4 conn.	8 conn.	10 conn.
throughput	1152.9	1702.0	1947.7	1676.4	1594.4

Table 3: Total throughput across all SAC connections for the inverse schedule as the number of SAC connections is increased.

Figure 5.14 plots the individual throughput achieved by each SAC connection when a total of 10 are present. We observe that fairness, for the network configuration shown in Figure 5.2, is well preserved. As the configuration becomes less uniform, access discrepancies as with TCP and other feedback congestion control algorithms are bound to arise which is a generic problem not specific to SAC.

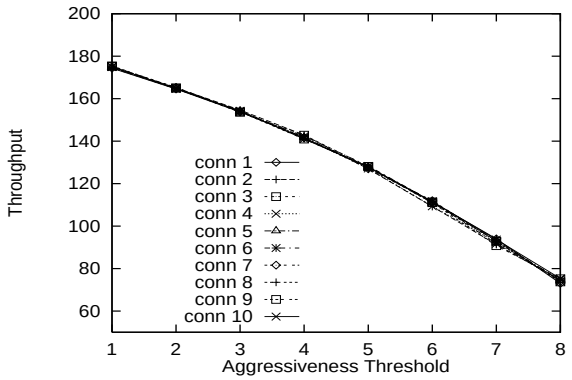

Figure 5.14: Fair bandwidth access: Individual throughput plot with 10 SAC connections present.

6 Conclusion

In this paper we have shown that a form of predictive congestion control called *Selective Aggressiveness Control* (SAC) is able to exploit correlation structure present in long-range dependent traffic for performance improvement purposes. We have shown that predictability can be implemented on-line with sufficient accuracy such that when coupled with a selective aggressiveness control mechanism, the cost associated with conservative bandwidth consumption during the linear increase phase of a generic linear increase/exponential decrease feedback congestion control algorithm can be offset to bring forth improved throughput. Furthermore, we show that the relative performance gain is higher the more long-range dependent the underlying network traffic. We have also shown that SAC preserves fairness to the degree expected of a feedback congestion control and its efficiency property is desirable within a range.

Current work is directed along three avenues, one, in devising new mechanisms capable of exploiting predictability at longer time scales, two, in further optimizing the various components of SAC, and three, in implementing the ideas of SAC in TCP. We hope to report some of these results in the near future.

References

[1] A. Adas and A. Mukherjee. On resource management and QoS guarantees for long range dependent traffic. In *Proc. IEEE INFOCOM '95*, pages 779–787, 1995.

[2] R. Addie, M. Zukerman, and T. Neame. Fractal traffic: measurements, modelling and performance evaluation. In *Proc. IEEE INFOCOM '95*, pages 977–984, 1995.

[3] Theodore Anderson. *The Statistical Analysis of Time Series.* John Wiley & Sons, 1994.

[4] Jan Beran. *Statistics for Long-Memory Processes.* Monographs on Statistics and Applied Probability. Chapman and Hall, New York, NY, 1994.

[5] Jean-Chrysostome Bolot and A. Udaya Shankar. Analysis of a fluid approximation to flow control dynamics. In *Proc. IEEE INFOCOM '92*, pages 2398–2407, 1992.

[6] L. Brakmo and L. Peterson. TCP Vegas: end to end congestion avoidance on a global internet. *IEEE J. Select. Areas Commun.*, 13(8):1465–1480, 1995.

[7] Imrich Chlamtac and William R. Franta. Rationale, directions, and issues surrounding high speed networks. *Proc. IEEE*, 78(1):94–120, 1990.

[8] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, May 1996.

[9] M. Crovella and L. Lipsky. Long-lasting transient conditions in simulations with heavy-tailed workloads. To appear in *Proc. 1997 Winter Simulation Conference*, 1997.

[10] R. Dighe, C. J. May, and G. Ramamurthy. Congestion avoidance strategies in broadband packet networks. In *Proc. IEEE INFOCOM '91*, pages 295–303, 1991.

[11] N. G. Duffield and N. O'Connel. Large deviations and overflow probabilities for the general single server queue, with applications. Technical Report DIAS-STP-93-30, DIAS Technical Report, 1993.

[12] A. E. Eckberg. B-ISDN/ATM traffic and congestion control. *IEEE Network*, pages 28–37, September 1992.

[13] K. Fendick, M. Rodrigues, and A. Weiss. Analysis of a rate-based control strategy with delayed feedback. In *Proc. ACM SIGCOMM '92*, pages 136–148, 1992.

[14] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. In *Proc. ACM SIGCOMM '93*, pages 33–44, 1993.

[15] M. Garret and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In *Proc. ACM SIGCOMM '94*, pages 269–280, 1994.

[16] M. Gerla and L. Kleinrock. Flow control: a comparative survey. *IEEE Trans. Commun.*, 20(2):35–49, 1980.

[17] G. C. Goodwin and K. S. Sin. *Adaptive Filtering, Prediction and Control*. Prentice Hall, 1984.

[18] M. Grossglauser and J-C. Bolot. On the relevance of long-range dependence in network traffic. In *Proc. ACM SIGCOMM '96*, pages 15–24, 1996.

[19] Z. Haas and J. Winters. Congestion control by adaptive admission. In *Proc. IEEE INFOCOM '91*, pages 560–569, 1991.

[20] Zygmunt Haas. A communication architecture for high-speed networking. In *Proc. IEEE INFOCOM '90*, pages 433–441, 1990.

[21] Duke Hong and Tatsuya Suda. Congestion control and prevention in ATM networks. *IEEE Network Magazine*, pages 10–16, July 1991.

[22] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(23):359–366, 1989.

[23] C. Huang, M. Devetsikiotis, I. Lambadaris, and A. Kaye. Modeling and simulation of self-similar variable bit rate compressed video: a unified approach. In *Proc. ACM SIGCOMM '95*, pages 114–125, 1995.

[24] Van Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM '88*, pages 314–329, 1988.

[25] S. Keshav. A control-theoretic approach to flow control. In *Proc. ACM SIGCOMM '91*, pages 3–15, 1991.

[26] Hyogon Kim. *A Non-Feedback Congestion Control Framework for High-Speed Data Networks*. PhD thesis, University of Pennsylvania, 1995.

[27] H. T. Kung, T. Blackwell, and A. Chapman. Credit-based flow control for ATM networks: credit update protocol, adaptive credit allocation, and statistical multiplexing. In *Proc. SIGCOMM '94*, pages 101–114, 1994.

[28] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2:1–15, 1994.

[29] N. Likhanov and B. Tsybakov. Analysis of an ATM buffer with self-similar ("fractal") input traffic. In *Proc. IEEE INFOCOM '95*, pages 985–992, 1995.

[30] D. Mitra and J. Seery. Dynamic adaptive windows for high speed data networks: theory and simulations. In *Proc. ACM SIGCOMM '90*, pages 30–37, 1990.

[31] A. Mukherjee and J. Strikwerda. Analysis of dynamic congestion control protocols - a Fokker-Planck approximation. In *Proc. ACM SIGCOMM '91*, pages 159–169, 1991.

[32] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16:387–396, 1994.

[33] K. Park, G. Kim, and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proc. IEEE International Conference on Network Protocols*, pages 171–180, October 1996.

[34] K. Park, G. Kim, and M. Crovella. On the effect of traffic self-similarity on network performance. In *Proc. SPIE International Conference on Performance and Control of Network Systems*, pages 296–310, 1997.

[35] Kihong Park. Warp control: a dynamically stable congestion protocol and its analysis. In *Proc. ACM SIGCOMM '93*, pages 137–147, 1993.

[36] K. K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. In *Proc. ACM SIGCOMM '88*, pages 303–313, 1988.

[37] B. Ryu and A. Elwalid. The importance of long-range dependence of VBR video traffic in ATM traffic engineering: myths and realities. In *Proc. ACM SIGCOMM '96*, pages 3–14, 1996.

[38] Scott Shenker. A theoretical analysis of feedback flow control. In *Proc. ACM SIGCOMM '90*, pages 156–165, 1990.

[39] M. S. Taqqu, V. Teverovsky, and W. Willinger. Estimators for long-range dependence: an empirical study, 1995. Preprint.

[40] H. L. van Trees. *Detection, Estimation and Modulation Theory.* John Wiley, 1968.

[41] Y. T. Wang and B. Sengupta. Performance analysis of a feedback congestion control policy under non-negligible propagation delay. In *Proc. ACM SIGCOMM '91*, pages 149–157, 1991.

[42] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. In *Proc. ACM SIGCOMM '95*, pages 100–113, 1995.

[43] C. Yang and A. Reddy. A taxonomy for congestion control algorithms in packet switching networks. *IEEE Network*, pages 34–45, July/August 1995.