Multiple Time Scale Congestion Control for Self-Similar Network Traffic [1]

Tsunyi Tuan [a,2] Kihong Park [a,3]

[a] *Network Systems Lab, Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA*

Abstract

Analytical and empirical studies have shown that self-similar traffic can have a detrimental impact on network performance including amplified queuing 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 network performance—in particular, throughput—under self-similar traffic conditions. Although scale-invariant burstiness implies the existence of concentrated periods of contention and idleness, the long-range dependence associated with self-similar traffic leaves open the possibility that correlation structure at larger time scales may be exploited for performance enhancement purposes.

We construct a 2-level multiple time scale congestion control protocol that exercises congestion control concurrently across two time scales an order of magnitude apart. The first component—acting at the smaller time scale—is a generic linear increase/exponential decrease feedback congestion control that uses implicit prediction afforded by feedback to affect rate control sensitive to changes in network state at 20–200ms time scales. The second component—acting at 2–5s time scales—uses explicit prediction to detect persistent shifts in overall network contention and uses this information to modulate the aggressiveness exhibited by the first component.

We show that cooperative interaction between the two congestion control modules acting on information at different time scales leads to improved performance vis-à-vis the case when the large time scale component is absent. We show that the improvement factor increases with long-range dependence and we show that as the number of flows engaging in multiple time scale congestion control (MTSC) increases, both fairness and efficiency are preserved.

Key words: Self-Similar Traffic, Congestion Control, Rate-Based Feedback Control

1 Introduction

1.1 Motivation

Recent measurements of local-area and wide-area traffic [8,27,34,41] have shown that network traffic exhibits variability at a wide range of time scales. What is striking is the ubiquitousness of the phenomenon which has been observed in diverse networking contexts, from Ethernet to ATM, compressed VBR video, and HTTP-based WWW traffic [8,14,22,41]. Such scale-invariant variability stands in contrast with traditional models of network traffic which exhibit burstiness at short time scales but are essentially smooth at large time scales; i.e., they lack long-range dependence.

A number of performance studies [1,2,10,28,31] have shown that self-similarity can have a detrimental effect on network performance leading to increased queueing delay and packet loss rate. From a queuing theory perspective, a 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 which possess exponential decay. In [17,36], the point is advanced that for small buffer sizes or short time scales, long-range dependence has only a marginal impact on performance. This is, in part, due to a saturation effect that arises when resources are overextended and burstiness associated with short-range dependent traffic is sufficient—and, in many cases, dominant—in determining queueing and buffer overflows.

What is still in its infancy is the problem of *controlling* self-similar network traffic. By the control of self-similar traffic, we mean the problem of modulating traffic flows such that network performance including throughput is optimized. Scale-invariant burstiness introduces new complexities into the picture making the task of provisioning 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 and low 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. The distinguishing feature is burstiness at coarser time scales which induces extended periods of either overutilization or underutilization which degrades overall performance.

On the flip side, *long-range dependence*—by definition—implies the existence of nontrivial correlation structure at larger time scales which may be exploitable for congestion control purposes, information to which current algorithms are impervious. How to exploit such information effectively to improve performance is a nontrivial problem and the subject matter of this study.

[1] Supported in part by NSF grant ANI-9714707.
[2] Contact author. Tel.: (765) 494–0875; fax.: (765) 494–0739; e-mail: tsunyi@cs.purdue.edu.
[3] Additionally supported by NSF grants ANI-9875789 (CAREER), ESS-9806741, and grants from PRF and Sprint. E-mail: park@cs.purdue.edu.

1.2 New Contributions

In this paper, we show that congestion control can be exercised concurrently at multiple time scales, and by cooperatively engaging information extracted at different time scales, achieve significant performance gains vis-à-vis congestion controls that are sensitive to only a single—in particular, short-range—time scale. We construct a 2-level multiple time scale congestion control protocol that affects congestion control across two time scales more than an order of magnitude apart. The first component—acting at a smaller time scale—is a generic linear increase/exponential decrease feedback congestion control that uses *implicit prediction* afforded by feedback to impart rate control that is sensitive to changes in network state at 20–200ms time scales. The latter, in turn, is determined by the round trip time (RTT) or latency of the feedback loop.

The second component—acting at 2–5s time scales—uses *explicit prediction* to detect persistent changes in the overall network contention level and uses this information to modulate the aggressiveness exhibited by the first component. Modulation is directed at inducing more aggressive bandwidth consumption when network contention is low—i.e., available bandwidth is large—and inducing conservative bandwidth consumption when the opposite is true. Asymmetry in the linear increase/exponential decrease control law is a sufficient condition for achieving stability, ignoring the additional impact that *delayed* feedback can have on stability [33]. In [25], it is shown that conservativeness with respect to bandwidth consumption—as implied by asymmetry—can lead to low resource utilization and this is amplified the longer the feedback loop or RTT. Increasing the rate or slope of the linear increase phase to reduce the level of conservativeness, without proper contextual information, can "backfire" due to increased occurrence of exponential backoff which can offset the gains obtained from more aggressive bandwidth consumption.

The congestion control module acting at the larger time scale is called Selective Aggressiveness Control (SAC) and it uses predicted information about network state at the 2–5s level to influence the *aggressiveness* exerted by the linear increase/exponential decrease feedback congestion control during its linear increase phase. In particular, when the predicted contention level during the next 2–5s interval is low, the rate of increase during the linear increase phase is amplified so as to facilitate more aggressive bandwidth consumption. On the other hand, if the predicted contention level at the large time scale is high, then the slope of the linear increase phase is dampened to induce a more conservative bandwidth consumption.

We show that this particular form of cooperative coupling between the two congestion control modules acting at different time scales is effective at exploiting long-range correlation structure, leading to significant gains in throughput vis-à-vis the case when the large time scale component is inactive. We show that the improvement factor increases with long-range dependence. For short-range dependent traffic, the large time scale congestion control module has a marginal effect. We also show that as the number of flows engaging in MTSC increases, both fairness and efficiency are preserved. The latter is with respect to total throughput achieved across all SAC-

controlled connections. Our specific form of multiple time scale congestion control is modular and easily portable to other types of feedback congestion control algorithms, a case in point being its recent implementation in the context of TCP. With TCP Reno replacing the generic linear increase/exponential decrease component as the short-time scale congestion control, we observe similar performance improvements as those attained with generic rate-based congestion control. Finally, we note that MTSC adds proactivity to feedback congestion controls which are essentially reactive. The positive effect is most pronounced for large RTT control loops.

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 performance evaluation set-up employed in this paper. In Section 2.3, we describe the predictability mechanism and its efficacy at extracting correlation structure present in long-range dependent traffic. This is followed by Section 3 where we describe the multiple time scale congestion control framework and its two specific components—generic linear increase/exponential decrease congestion control and SAC. In Section 4 we show performance results of the MTSC framework and demonstrate its efficacy under different resource configurations, long-range dependence conditions, and when the number of SAC connections sharing common network resources is varied. We conclude with a discussion of our 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 traffic 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:

$$r(k) \sim \text{const} \cdot k^{-\beta}, \tag{1}$$

$$r^{(m)}(k) \sim r(k), \tag{2}$$

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)—and $r(k)$ behaves hyper-

bolically with $\sum_{k=0}^{\infty} r(k) = \infty$ as implied by (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 (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. A comprehensive discussion of estimation methods can be found in [4,38].

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.

2.2 Structural Causality

In [32], 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 transported 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 "typical" protocols (e.g., TCP, flow-controlled UDP) is self-similar. Furthermore, traffic is more bursty—in the scale-invariant sense—the more heavy-tailed the file size distribution. This relationship is shown in Figure 1. The relationship is robust with respect to changes in network resources (e.g., bandwidth, buffer capacity), topology, the influence of cross-traffic, and the distribution of inter-arrival times. We call this relationship between the traffic pattern observed at the network layer and the structural property of a networked system in terms of its high-level object sizes *structural causality*.

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

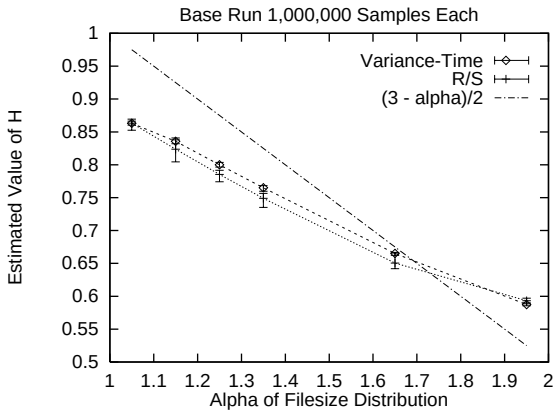

Fig. 1. Hurst Parameter estimates (R/S and V-T) for α varying from 1.05 to 1.95

network with heavy-tailed object sizes—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 reflects the actions and modulating influence of the protocol stack. The observed traffic pattern is a direct consequence of hosts exchanging files whose transmission, in turn, was mediated by transport 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.

2.3 Long-Range Dependent Traffic and Predictability

We show that correlation structure present in long-range dependent traffic can be detected online and used to predict future traffic levels over time scales relevant to congestion control. We choose a simple, easy-to-implement estimation scheme based on conditional expectation and use it as a reference for studying congestion control techniques and their efficacy at exploiting correlation structure present in long-range dependent traffic. We emphasize that investigating optimum prediction—a difficult problem for long-range dependent traffic [4]—is not our objective. In fact, our prediction scheme can be replaced by any other prediction scheme (e.g., [3,16,21,39]), and if the latter is superior, this will only amplify the efficacy of our multiple time scale congestion control results.

To fix notation, assume we are given a wide-sense stationary stochastic process $(\xi_t)_{t\in\mathbb{Z}}$ and two numbers $T_1, T_2 > 0$. Let

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

We are interested in computing the conditional probability densitity $\Pr\{V_2 \mid V_1 = a\}$ which would allow us to predict the future traffic level during the time interval $[t, t+T_2)$ given the observation a of the recent past $[t-T_1, t)$. We employ two further random variables L_1, L_2 with range $[1, m]$ that perform a certain quantization $L_k = L_k(V_k)$, $k = 1, 2$. If $L_k \approx 1$ then

the traffic level is interpreted as "low" and if $L_k \approx m$ then it is understood as "high." Thus the conditional probability density of interest is $\Pr\{L_2 \mid L_1 = \ell\}$, $\ell \in [1, m]$. A method for estimating $\Pr\{L_2 \mid L_1 = \ell\}$—off-line and on-line—is described in the Appendix.

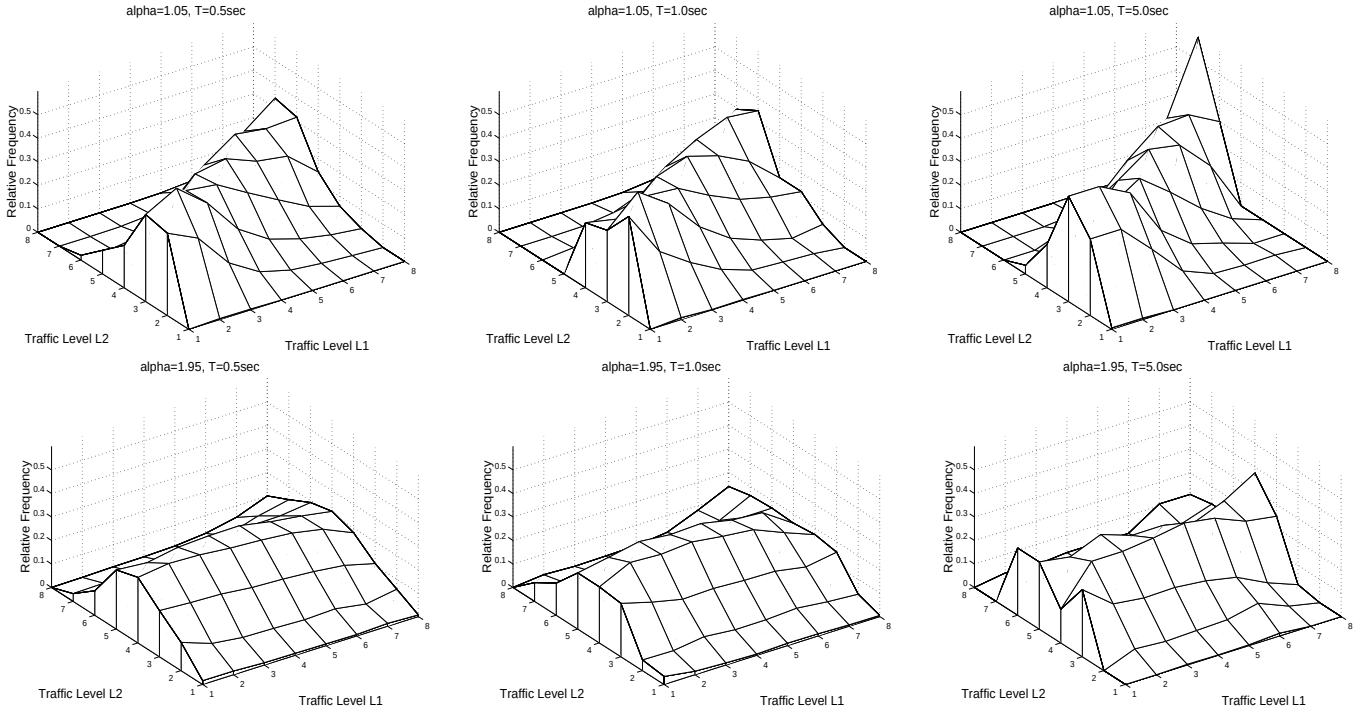

Fig. 2. Top Row: Probability densities with L_2 conditioned on L_1 for $\alpha = 1.05$. Bottom Row: Probability densities with L_2 conditioned on L_1 for $\alpha = 1.95$.

Figure 2 shows the estimated conditional probability densities for $\alpha = 1.05, 1.95$ traffic for time scales 500ms, 1s, and 5s for the time series corresponding to Figure 1. In the following, $T_1 = T_2 = 2$s and $m = 8$. For the aggregate throughput traces with $\alpha = 1.05$—cf. Figure 2 (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. Figure 2 (bottom row) shows that conditioning is ineffective for $\alpha = 1.95$ traffic.

2.4 Predictability and Time Scale

An important issue is how time scale affects predictability when traffic is long-range dependent. Going back to Figure 2 (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 and 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—of 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 $\Pr\{L_2 \mid L_1 = \ell\}$,

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

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 m conditional probability densities ($m = 8$), one for each $L_1 = 1, 2, \ldots, m$, we define the *average entropy* $\bar{S}$ as $\bar{S} = \sum_{\ell=1}^{m} S_\ell/m$. The average entropy remains a function of T_1 and T_2, $\bar{S} = \bar{S}(T_1, T_2)$.

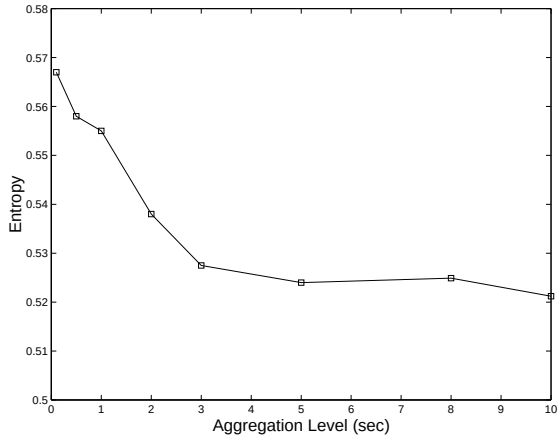

Fig. 3. Average entropy $\bar{S}(T_1)$ plot for $\alpha = 1.05$ traffic as a function of time scale T_1.

Figure 3 plots $\bar{S}(T_1, T_2) = \bar{S}(T_1)$ (recall that $T_1 = T_2$) for the $\alpha = 1.05$ traffic series as a function of time scale or aggregation level T_1. Entropy is highest for small time scales in the vicinity of 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. We find that the "knee" of the entropy curve is in the 1–5 second range. Note that increasing T_1 further to gain small decreases in entropy carries with it an important problem: if prediction is done over a "too long" time interval, then the information may not be effectively exploitable by various congestion control strategies.

3 Multiple Time Scale Congestion Control

As explicated in Section 1.2, the MTSC framework consists of two congestion control modules acting at time scales 20–200ms and 2–5s, respectively, which cooperate to affect improved performance. The short time scale component is a generic linear increase/exponential decrease rate-based congestion control whose sensitivity to changes in network state is determined by the RTT associated with the feedback loop. The long time scale component—Selective Aggressiveness Control (SAC)—uses explicit prediction of network contention levels at 2–5s time scales to

modulate the aggressiveness exhibited by the short time scale component. Coupling is achieved by an aggressiveness parameter a_t which is used by the generic linear increase/exponential decrease congestion control to set its slope during the linear increase phase. The MTSC framework is depicted in Figure 4.

SAC's modus operandi is to complement and help improve the performance of existing reactive congestion controls by imparting *proactivity*. SAC respects the decision made by the underlying short time-scale 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.

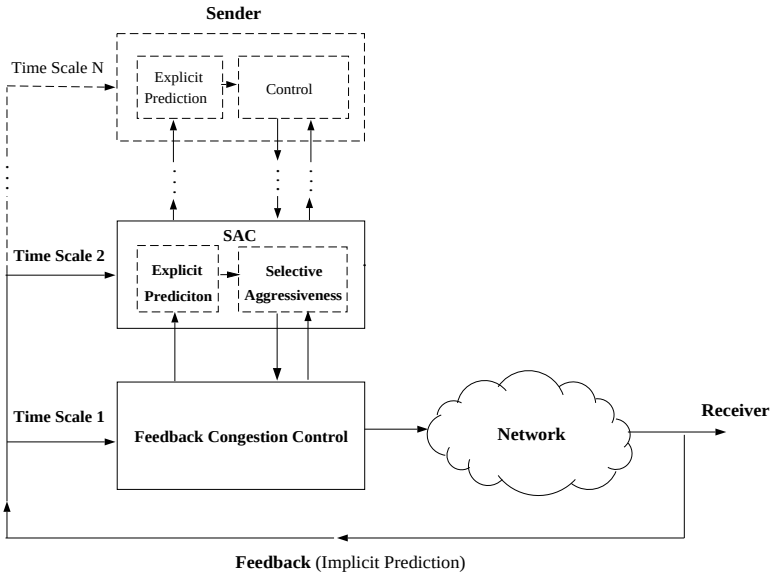

Fig. 4. Multiple time scale congestion control framework. Information at time scale 2 is used to influence the behavior of congestion control at time scale 1. Dashed lines show the potential extensibility of the framework to three or more time scales.

3.1 Generic Rate-Based Feedback Congestion Control

3.1.1 Linear Increase/Exponential Decrease Congestion Control

Congestion control has been an active area of networking research spanning over two decades with a flurry of work carried out in the late '80s and early '90s [5,6,15,18,23,24,26,29,30,33,35,37]. Gerla and Kleinrock [15] laid down much of the early groundwork and Jacobson [23] 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 [5,12,23,24,29,30,33,37]

associated with feedback congestion control. A taxonomy for classifying the various protocols can be found in [42].

More recently, the delay-bandwidth product problem arising from high-bandwidth networks and quality of service issues stemming from support of real-time multimedia communication [7,9,11,19,20,40] have added further complexities to the problem. 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 affected to achieve stability.

We will employ a simple, generic 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 observation put forth in [25] which states 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 predicted future network contention—we seek to offset some of the cost incurred for achieving stability.

Let λ denote packet arrival rate and let γ denote throughput. Our *generic linear increase/exponential decrease feedback congestion control* has a control law of the form [4]:

$$\frac{d\lambda}{dt} = \begin{cases} \delta, & \text{if } d\gamma/d\lambda > 0, \\ -a\lambda, & \text{if } d\gamma/d\lambda < 0, \end{cases} \tag{3}$$

where δ, $a > 0$ 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*. Difficulties arise because (3), in reality, is a delay differential equation—the feedback loop incurs a time lag—and the sign of $d\gamma/d\lambda$ needs to be reliably estimated. These issues can be addressed using a number of techniques [33].

3.1.2 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. They range 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* which is throughput 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., $M/M/1/n$.

[4] We use continuous notation for expositional clarity. Their discrete counterparts are straightforward.

Raw throughput, however, does not adequately discriminate between congestion controls that achieve a certain throughput without incurring high packet loss 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 exert a severe impact on the efficient functioning of such controls. 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 (e.g., 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

$$\gamma_k = (1-c)^k \nu \tag{4}$$

that polynomially penalizes raw throughput ν by packet loss rate c, $0 \leq c \leq 1$, where the level of 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 control law (3). Figure 5 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)) will be unimodal due to the above monotonicity properties.

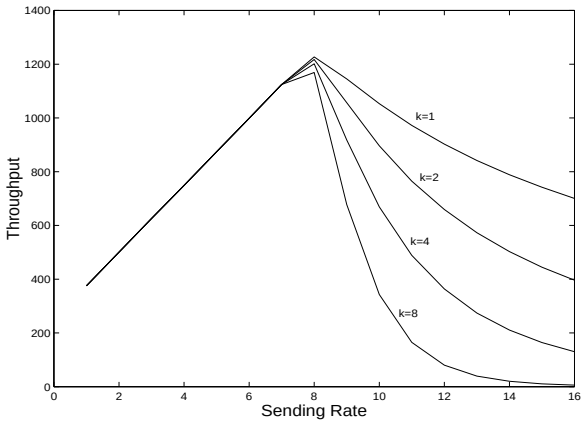

Fig. 5. Unimodal load-throughput curve $\gamma_k = \gamma_k(\lambda)$ for $M/M/1/n$ system for $k = 1, 2, 4, 8$.

3.2 Selective Aggressiveness Control

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 penalty stemming conservative bandwidth usage during the *linear increase phase* of linear increase/exponential decrease congestion control algorithms [25]. Feedback congestion control protocols, including TCP, implement variants of this basic control law due to well-established stability reasons. In [25], however, it was shown that in the context of TCP Reno 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 [5] .

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, is aimed at facilitating a more effective use of bandwidth resulting in improved performance. Without selective, controlled application of aggressiveness, however, the gain from aggressiveness may be cancelled or even dominated by its cost—aggressiveness, under high network contention conditions, can lead to deteriorated performance, even congestion collapse—thus making predictability and its appropriate exploitation a nontrivial problem. The SAC protocol is composed of two parts, prediction and application of aggression, and they are described next.

3.2.1 Per-Connection On-Line Estimation of Contention Level

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 end-to-end control 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,

[5] This potential problem was also recognized in Jacobson's seminal paper [23] which, in part, motivated TCP Tahoe's Slow Start feature.

predict *future* network state using information obtained from a flow's input/output behavior when interacting with the network. For this to work, two assumptions need to hold in practice. One, due to the coupling stemmping from sharing of common resources, a connection's individual throughput when engaging in feedback congestion control such as (3) is correlated with the aggregate flow accessing the same resources. Two, the aggregate traffic level, when partitioned according to the quantization scheme $L_k(V_k)$, is correlated with the contention level at routers 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 $m \times (m+1)$, one row for each $\ell \in [1, m]$. The last column of `CondProb`, $\texttt{CondProb}[\ell][m+1]$, is used to keep track of h_ℓ, the number of blocks observed thus far whose traffic level map to ℓ, i.e., $L_1(V_1) = \ell$. For each $\ell' \in [1, m]$, $\texttt{CondProb}[\ell][\ell']$ 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. The estimation procedure for `CondProb` is described in the Appendix.

3.2.2 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* (3)—when such actions are warranted.

The actuation part of the interface between SAC and (3) is defined as follows: Let λ_t denote the newly updated rate value at time t—by (3)—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 $\lambda_t \leftarrow \lambda_t + a_t$.
2. Else do nothing.

Here, $a_t \geq 0$ 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 (3), i.e., when $\lambda_t > \lambda_{t'}$. The magnitude of a_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 a is as follows: Let S_t be the aggregate throughput

reported by the receiver via feedback over time interval $[t - T_1, t]$.

SAC (aggressiveness determination):

1. Let $\ell = L_1(S_t)$.
2. Compute $\bar{\ell}' = \mathbf{E}(L_2 \mid L_1 = \ell) = \sum_{\ell'=1}^{m} \ell' \Pr\{L_2 = \ell' \mid L_1 = \ell\}$.
3. Set $a_t = \epsilon(\bar{\ell}')$.

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 such schedule is the *inverse schedule*

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

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

$L_1 \backslash L_2$	1	2	3	4	5	6	7	8	$E[L_2\|\cdot]$	$\epsilon(\cdot)$
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	$E[L_2\|\cdot]$	$\epsilon(\cdot)$
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 $\alpha = 1.05$, over a 10000s duration. Bottom: An snapshot of `CondProb` for $\alpha = 1.95$.

Table 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. 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 inverse aggressiveness schedule 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.

4 Simulation Results

4.1 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.

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 [32]. 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 6.

Figure 7 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 conges-

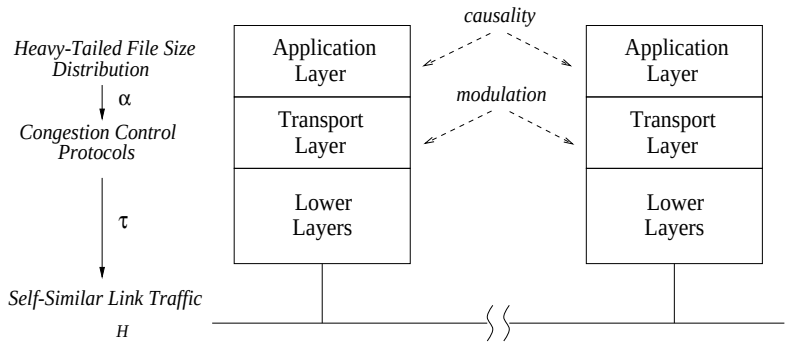

Fig. 6. Transformation of the heavy-tailedness of file size distribution property at the application layer via the action of the transport layer into its manifestation as self-similar aggregated traffic at the link layer.

tion 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.

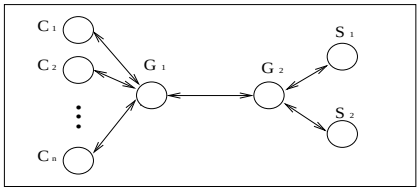

Fig. 7. Network configuration with bottleneck link (G_1, G_2).

In a typical configuration, the first 32 connections served as "background traffic" transferring files from clients to servers where the file sizes were drawn from Pareto distributions with shape parameter $\alpha = 1.05, 1.35, 1.65, 1.95$. As in [32], 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.

4.2 Per-Connection On-Line Predictability

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 3.2.1. 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). 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 8 (left) shows the shifting effect of the conditional probabilities for $\alpha = 1.05$ 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 8 (right)) the probability densities stay largely invariant.

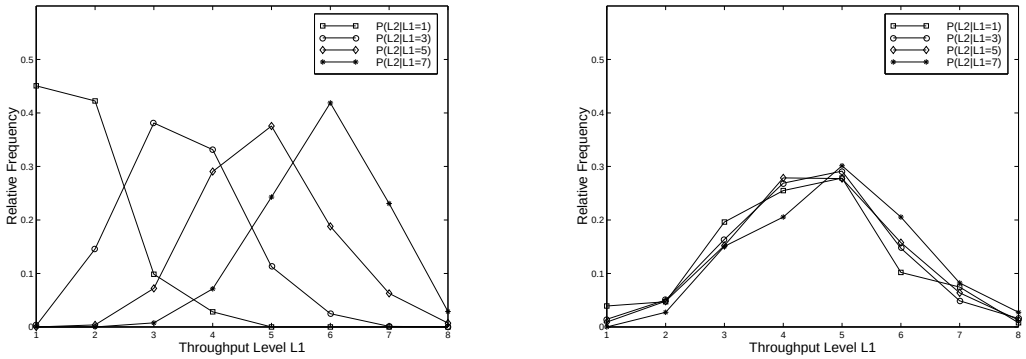

Fig. 8. Left: Shifting effect of conditional probability densities $P(L_2|L_1 = 1)$ and $P(L_2|L_1 = 8)$ under $\alpha = 1.05$ background traffic. Right: Shifting effect of conditional probability densities $P(L_2|L_1 = 1)$ and $P(L_2|L_1 = 8)$ under $\alpha = 1.95$ traffic.

4.3 Performance Measurement of SAC

4.3.1 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 3.2.2) where aggressive action is taken if and only if $\bar{\ell}' \leq \theta$ where θ is the aggressiveness threshold. Figure 9 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 causing a decrease in throughput. If $\theta = 8$, then this corresponds to the case where aggressiveness is applied at all times, i.e., there is *no selectivity*.

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.

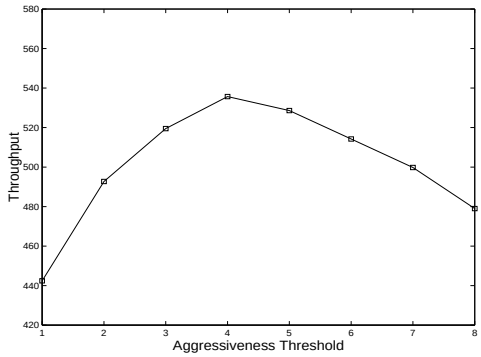

Fig. 9. Unimodal throughput curve as a function of aggressiveness threshold θ for $\alpha = 1.05$ traffic.

The shape of the curve is dependent upon the relative magnitude of available resources 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 10.

Figure 10 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.

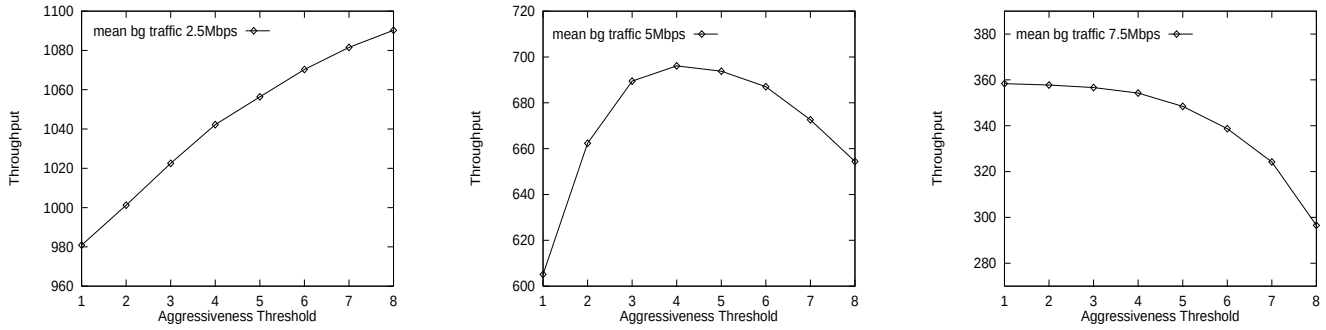

Fig. 10. Shape of throughput curve as a function of aggressiveness threshold for three levels of background traffic 2.5Mbps (left), 5Mbps (middle), and 7.5Mbps (right).

Figure 11 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 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.

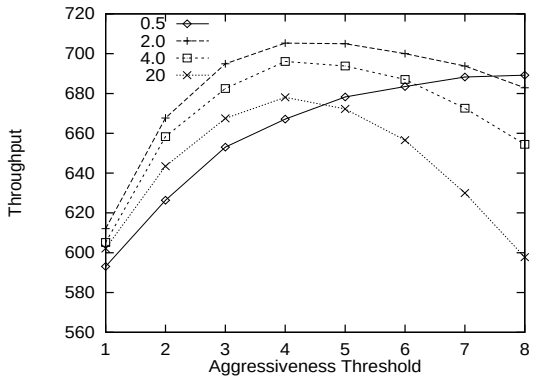

Fig. 11. Change in shape of throughput curve as the aggressiveness schedule $\epsilon(\cdot)$ is shifted (upwards) by 0.5, 2.0, 4.0, 20.0.

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 state to another.

4.3.2 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.

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 12 (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

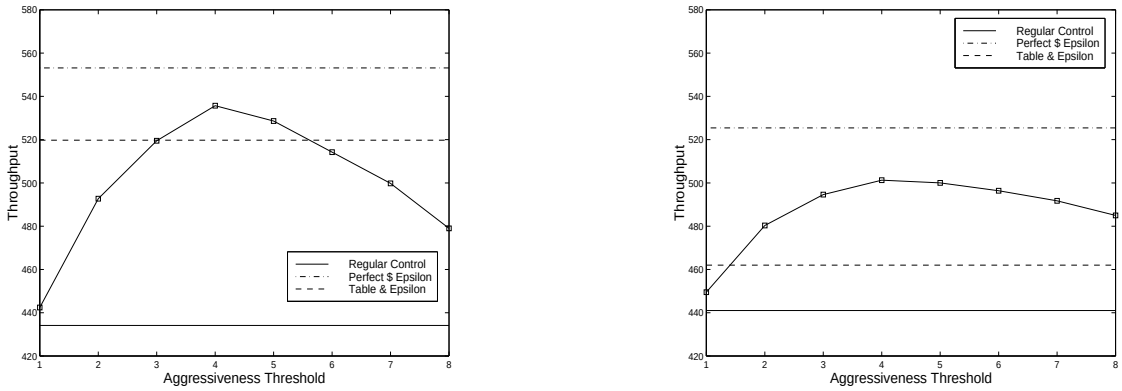

Fig. 12. Left: The horizontal lines show throughput when different control strategies are employed (top line: perfect prediction with inverse schedule; middle line: on-line table with inverse schedule; bottom line: generic linear increase/exponential decrease congestion control without SAC) for $\alpha = 1.05$ traffic. Right: Corresponding throughput plot for $\alpha = 1.95$ traffic.

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 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 12 (right) shows the corresponding plots for $\alpha = 1.95$ traffic where a similar ordering relation is observed.

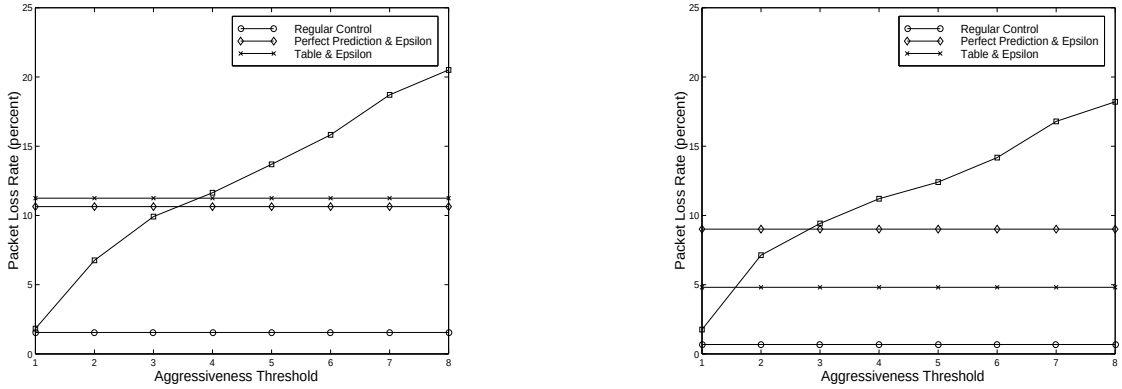

Fig. 13. Left: Corresponding packet loss rates for the control strategies shown in Figure 12 for $\alpha = 1.05$ background traffic. Right: Corresponding packet loss rates for $\alpha = 1.95$.

Figure 13 shows the packet loss rates corresponding to the throughput plots shown in Figure 12. 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 13 (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.

4.3.3 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.

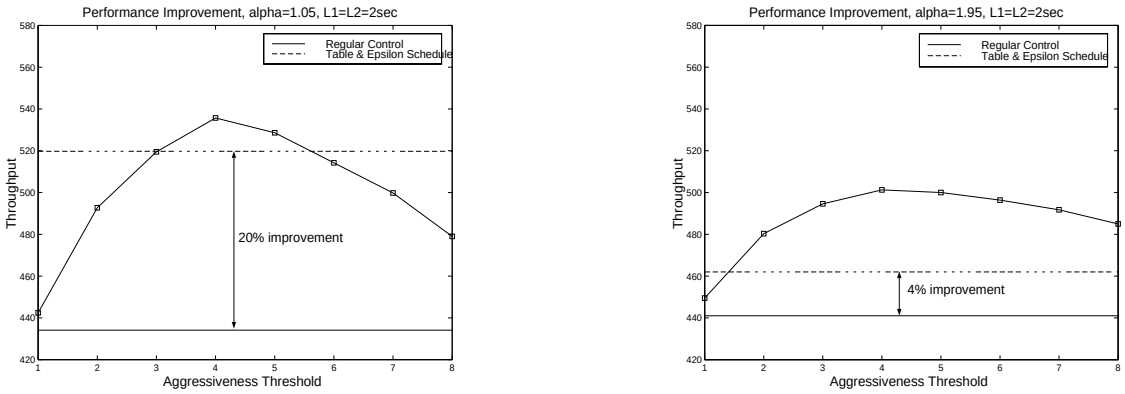

Fig. 14. Left: Under $\alpha = 1.05$ traffic, the performance improvement is about 20% when using SAC with on-line table and inverse schedule. Right: Under $\alpha = 1.95$ traffic, the performance improvement is only 4%.

In this section, we show that the relative performance gain due to selective aggressiveness control and predictability grows as long-range dependence increases. Figure 14 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.

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 15 (left) 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).

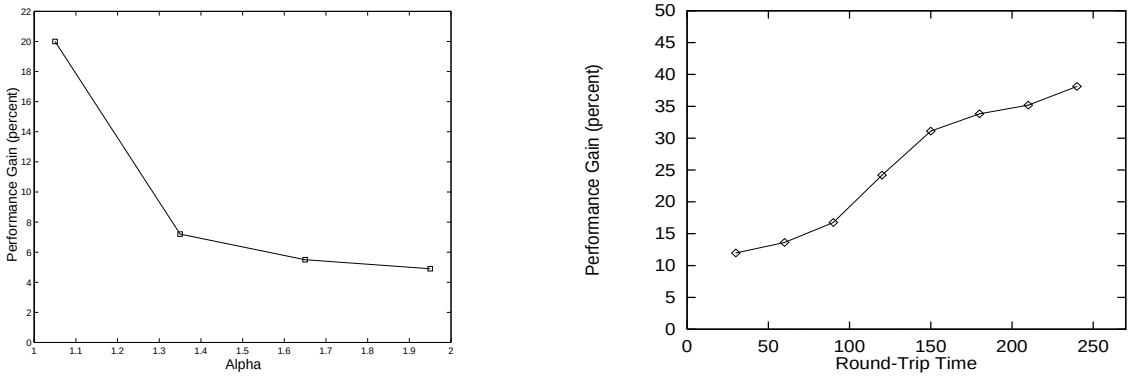

Fig. 15. Left: Performance gain due to predictability for $\alpha = 1.05, 1.35, 1.65, 1.95$ background traffic. Right: Performance gain as a function of RTT.

Another positive aspect of MTSC in the context of long-range dependent traffic is that it adds proactivity to short time scale feedback congestion control which partly offsets the negative effect of long time lags in the feedback loop on performance. Figure 15 (right) shows performance gain (%) as a function of RTT. We observe that the performance gain is amplified with increased RTT when SAC is active vis-à-vis when it is not.

4.3.4 Convergence Rate and Performance

The faster the convergence of the on-line conditional probability table, 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 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 16 (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 16 (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.

The fast convergence property can be further explained by Figure 16 (right) which plots the

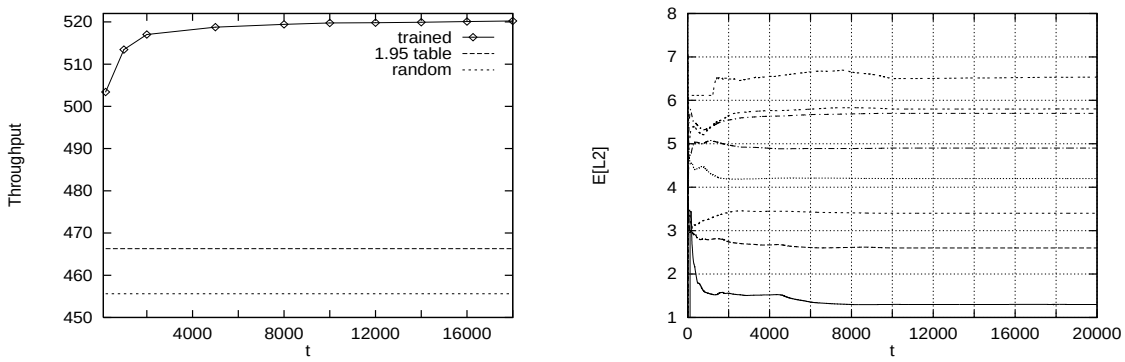

Fig. 16. Left: Throughput as a function of SAC conditional probability table training time. Also shown: SAC throughput with random and $\alpha = 1.95$ conditional probability table. Right: Convergence property of $\mathbf{E}(L_2 \,|\, \cdot)$. Fast convergence to linear order $\mathbf{E}(L_2 \,|\, L_1 = 1) < \mathbf{E}(L_2 \,|\, L_1 = 2) < \cdots < \mathbf{E}(L_2 \,|\, L_1 = 8)$.

computed conditional expectation $\mathbf{E}(L_2 \,|\, \cdot)$ as a function of training time. Recall that in both the threshold and inverse schedules $\mathbf{E}(L_2 \,|\, \cdot)$ (quantized or not)—not the conditional probability table proper—is used in computing the aggressiveness level. Figure 16 (right) shows that the functional $\mathbf{E}(L_2 \,|\, \cdot)$ quickly converges to the linear ordering $\mathbf{E}(L_2 \,|\, L_1 = 1) < \mathbf{E}(L_2 \,|\, L_1 = 2) < \cdots < \mathbf{E}(L_2 \,|\, L_1 = 8)$ as would be expected by the skewedness of the 3-D conditional probability densities. The magnitudes of $\mathbf{E}(L_2 \,|\, \cdot)$, after some undulation, stabilized to fixed values.

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

4.3.5 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 7 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 17 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 num-

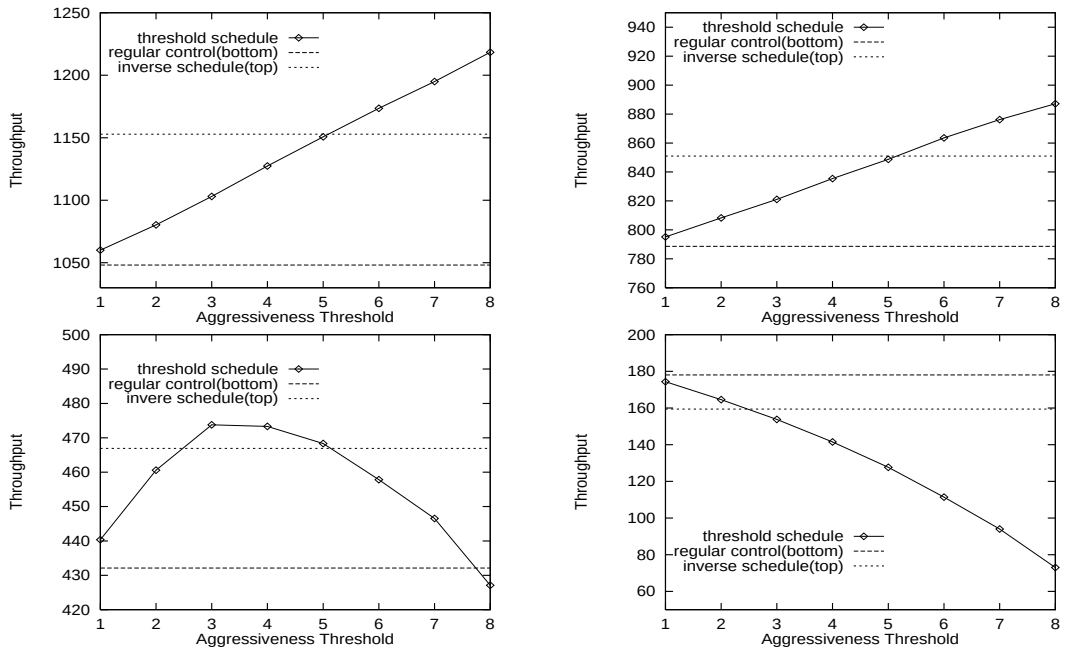

Fig. 17. Average (per-connection) throughput as a function of aggressiveness threshold for multiple SAC connections. The top horizontal line shows the throughput achieved with the inverse schedule. The bottom line represents the throughput of the generic linear increase/exponential decrease feedback congestion control. Top left: single SAC connection; top right: two SAC connections; bottom left: four SAC connections; bottom right: ten SAC connections.

ber 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 Table 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 7 which, due to its uniform link latencies, can be prone to synchronization effects [13].

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 18 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 7, 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.

5 Conclusion

In this paper we have shown a multiple time scale congestion control (MTSC) framework that exercises congestion control concurrently across two time scales—more than an order of magnitude apart—to affect improved throughput. We have shown that cooperative coupling between the long time scale component (SAC) and the short time scale component (generic feedback congestion control) via an aggressiveness parameter is effective at achieving significant gains in throughput. We have shown that relative performance gain is higher the more long-range dependent the underlying network traffic and the longer the RTT in the feedback loop of the generic feedback congestion control. We have also shown that fairness and efficiency are preserved as the number of SAC connections sharing common network resources is increased.

In just-completed work, we have shown that SAC can be used on top of TCP Reno to affect significant throughput gains. Current work is directed at exploring alternative ways of structuring and implementing the MTSC framework including different forms of coupling and exploiting correlation structure across multiple time scales.

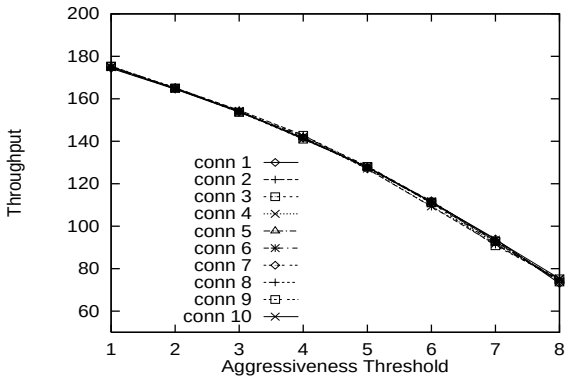

Fig. 18. Fair bandwidth access: Individual throughput plot with 10 SAC connections present.

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

[10] 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.

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

[12] 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.

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

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

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

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

[17] 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.

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

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

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

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

[22] 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.

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

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

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

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

[32] 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, 1996.

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

[34] V. Paxson and S. Floyd. Wide-area traffic: the failure of Poisson modeling. In *Proc. ACM SIGCOMM '94*, pages 257–268, 1994.

[35] 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.

[36] 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.

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

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

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

[40] 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.

[41] 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.

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

A Predictability Set-Up

A.1 Off-Line 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

$$\Pr\{V_2 = b \mid V_1 = a\} \tag{A.1}$$

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 m $(m = 8)$ *levels*

$$\begin{gathered}(-\infty, \mu_k - 3\sigma_k),\ [\mu_k - 3\sigma_k, \mu_k - 2\sigma_k),\ [\mu_k - 2\sigma_k, \mu_k - \sigma_k),\ [\mu_k - \sigma_k, \mu_k), \\ [\mu_k, \mu_k + \sigma_k),\ [\mu_k + \sigma_k, \mu_k + 2\sigma_k),\ [\mu_k + 2\sigma_k, \mu_k + 3\sigma_k),\ [\mu_k + 3\sigma_k, +\infty).\end{gathered}$$

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*. An alternative classification which is based on the uniform distribution has been found to be more universally effective.

Returning to (A.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.

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

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

A.2 On-Line Estimation

All that is needed to maintain `CondProb` is a clock 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.

Curriculum Vitae

Tsunyi Tuan received the B.S. degree in Mechanical Engineering from the National Tsing-Hua University, Hsinchu, Taiwan in 1990, and the M.S. degree in Electrical Engineering from Columbia University, New York, in 1994. He is currently working toward the Ph.D. degree in Electrical Engineering at Purdue University. He has been a teaching assistant in the Mathematics Department and is presently a research assistant in the Network Systems Lab in the Department of Computer Sciences at Purdue University. His research interests include congestion control for self-similar network traffic and its application to TCP and real-time traffic management.

Kihong Park received his B.A. from Seoul National University, Korea, and his Ph.D. in Computer Science from Boston University (1996). Presently, he is an assistant professor of computer science at Purdue University. His research centers around design and control issues in high-speed multimedia networks including congestion control, quality of service provision architectures, routing, and the facilitation of adaptive, fault-tolerant computing on large-scale distributed systems. He has over 40 technical publications and has served on several international program committees. He was a Presidential University Fellow at Boston University, is a recipient of the NSF CAREER Award, and is a member of several professional societies including ACM and IEEE.