

# On the relationship between file sizes, transport protocols, and self-similar network traffic\*

Kihong Park<sup>†</sup>  
park@cs.bu.edu

Gitae Kim<sup>‡</sup>  
kgtjan@cs.bu.edu

Mark Crovella<sup>§</sup>  
crovella@cs.bu.edu

**BU-CS-96-016**

August 7, 1996

Computer Science Department  
Boston University  
Boston, MA 02215

## Abstract

Recent measurements of local-area and wide-area traffic have shown that network traffic exhibits variability at a wide range of scales—self-similarity. In this paper, we examine a mechanism that gives rise to self-similar network traffic and present some of its performance implications. The mechanism we study is the transfer of files or messages whose size is drawn from a heavy-tailed distribution. We examine its effects through detailed transport-level simulations of multiple TCP streams in an internetwork.

First, we show that in a “realistic” client/server network environment—i.e., one with bounded resources and coupling among traffic sources competing for resources—the degree to which file sizes are heavy-tailed can directly determine the degree of traffic self-similarity at the link level. We show that this causal relationship is not significantly affected by changes in network resources (bottleneck bandwidth and buffer capacity), network topology, the influence of cross-traffic, or the distribution of interarrival times.

Second, we show that properties of the transport layer play an important role in preserving and modulating this relationship. In particular, the reliable transmission and flow control mechanisms of TCP (Reno, Tahoe, or Vegas) serve to maintain the long-range dependency structure induced by heavy-tailed file size distributions. In contrast, if a non-flow-controlled and unreliable (UDP-based) transport protocol is used, the resulting traffic shows little self-similar characteristics: although still bursty at short time scales, it has little long-range dependence. If

---

\*A short version will appear in *Proc. Fourth International Conference on Network Protocols*, October, 1996.

<sup>†</sup>Supported in part by NSF grant CCR-9204284. Address after August 12, 1996: Department of Computer Sciences, 1398 Computer Science Building, Purdue University, West Lafayette, IN 47907; park@cs.purdue.edu.

<sup>‡</sup>Supported in part by NSF grant CCR-9308344.

<sup>§</sup>Supported in part by NSF grant CCR-9501822

flow-controlled, unreliable transport is employed, the degree of traffic self-similarity is positively correlated with the degree of throttling at the source.

Third, in exploring the relationship between file sizes, transport protocols, and self-similarity, we are also able to show some of the performance implications of self-similarity. We present data on the relationship between traffic self-similarity and network performance as captured by performance measures including packet loss rate, retransmission rate, and queueing delay. Increased self-similarity, as expected, results in degradation of performance. Queueing delay, in particular, exhibits a drastic increase with increasing self-similarity. Throughput-related measures such as packet loss and retransmission rate, however, increase only gradually with increasing traffic self-similarity as long as reliable, flow-controlled transport protocol is used.

# 1 Introduction

Recent measurements of local-area and wide-area traffic [15, 28, 9] have shown that network traffic exhibits variability at a wide range of scales. Such scale-invariant variability is in strong contrast to traditional models of network traffic, which show variability at short time scales but are essentially smooth at large time scales. This effect is described statistically as long-range dependence, and time series showing this effect are said to be self-similar. Since self-similarity is believed to have a significant impact on network performance, understanding the causes and effects of traffic self-similarity is an important problem.

In this paper, we study a mechanism that induces self-similarity in network traffic. We show that self-similar traffic can arise from a simple, high-level property of the overall system—the heavy-tailed distribution of file sizes being transferred over the network. We show that if the distribution of file sizes is heavy-tailed—meaning that the distribution behaves like a power law thus generating very large file transfers with nonnegligible probability—then the superposition of many file transfers in a client/server network environment induces self-similar traffic and this causal mechanism is robust with respect to changes in network configuration. Properties of the transport/network layer in the protocol stack, however, will be shown to play an important role with respect to preserving and modulating this causal relationship.

The mechanism we propose is motivated by the ON/OFF traffic model described in [28]. The ON/OFF model shows that self-similarity can arise in an idealized context with unbounded resources and independent traffic sources as a result of aggregating a large number of ON/OFF traffic streams whose ON or OFF periods are heavy-tailed. The success of the simple ON/OFF model in capturing the characteristics of measured traffic traces [28] is surprising given that it ignores interaction among traffic sources contending for network resources which in real networks can be as complicated as the feedback congestion control algorithm of TCP Reno. To apply the framework of the ON/OFF model to real networks, it is necessary to understand whether the model's limitations affect its usefulness, and if not, how those limitations are overcome in practice. These are questions that can be answered most effectively by direct experimentation.

To establish the link between file sizes and traffic self-similarity, it is necessary to show that the degree of self-similarity varies as a direct result of variation in file size distribution. In addition, we must examine the behavior of networks with a wide range of characteristics operating under various network conditions. In this paper, we show that in a “realistic” client/server network environment—i.e., one with bounded resources leading to the coupling of multiple traffic sources contending for shared resources—the degree to which file sizes are heavy-tailed directly determines the degree of traffic self-similarity. Specifically, measuring self-similarity via the Hurst parameter  $H$  and file size distribution by its power-law exponent  $\alpha$  (their definitions are given later), we show

that there is a nearly linear relationship between  $H$  and  $\alpha$  over a wide range of network conditions when subject to the influence of the protocol stack. This mechanism gives a particularly simple explanation of why self-similar network traffic may be observed in many diverse contexts.

This relationship is robust in the sense that it is present over a wide range of conditions including changes in bottleneck bandwidth and buffer capacity, interference from cross-traffic possessing dissimilar traffic characteristics, and changes in the distribution of conversation interarrival times. For example, if self-similar traffic is mixed with cross-traffic that is only short-range dependent—hence smooth at large time-scales—then self-similarity remains persistent in the aggregated traffic. In addition, if the interarrival time distribution is very heavy-tailed, it can amplify traffic self-similarity when file sizes are only moderately heavy-tailed. However, if the file size distribution is very heavy-tailed, then the interarrival time distribution has virtually no effect on traffic self-similarity.

We also discuss a traffic shaping effect of TCP that helps explain how heavy-tailed file sizes generate self-similar traffic. We find that, in practice, the presence of self-similarity depends on whether reliable and flow-controlled communication is employed at the transport layer. For example, in the absence of reliability and flow control mechanisms such as when a UDP-based transport protocol is used, much of the self-similarity of downstream traffic is destroyed as compared to the case of upstream traffic. The resulting traffic, while still bursty at short ranges, shows significantly less long-range correlation structure. In contrast, when TCP (Reno, Tahoe, or Vegas) is employed, the long-range dependence structure induced by heavy-tailed file size distributions is preserved and transferred to the link-layer, manifesting itself as scale-invariant burstiness. In essence, this is a combined effect of the input traffic being flow controlled and conserved through retransmission-based reliable transport. In the following, we will use “reliability” loosely to refer to both.

We conclude with a discussion of the effect of self-similarity on network performance. We find that as self-similarity is increased in an UDP-based non-flow-controlled<sup>1</sup> environment, performance declines drastically as measured by packet loss rate and mean queue length. However, if reliable communication via TCP Reno is used, packet loss, retransmission rate, and file transmission time decline gracefully, i.e., roughly linearly as a function of  $H$ . The exception is mean queue length, which shows the same superlinear increase as in the unreliable non-flow-controlled communication case. This graceful decline in TCP’s performance under self-similar loads comes at a cost: a disproportionately increased consumption of buffer space. The sensitive dependence of mean queue length on self-similarity agrees with previous work [16] showing that queue length distribution decays more slowly for long-range dependent traffic than for short-range dependent sources. The

---

<sup>1</sup>H-estimates and performance results when an open-loop flow control is active can be found in [20]. Increasing the degree of throttling at the source has a positive influence on both the H-values and performance, and one may think of the non-flow-controlled case as an extreme case and no compensatory actions are taken whatsoever.

graceful performance decline exhibited by reliable communication is a result of shaping a large file transfer into an on-average, “thin” packet train (stretching-in-time effect). Stretching-in-time is effected through the joint action of retransmission (or conservation) of lost packets, and the *conservative* nature of linear-increase/exponential-decrease feedback congestion control [14]. This also suggests, in part, why the ON/OFF model has been so successful despite its lack of coupling among traffic sources—a principal effect of interaction among traffic sources in an internetworked environment seems to lie in the generation of lengthy packet trains. Thus, the translation of large file transfers into well-behaved elongated packet trains may be related to system-wide traffic self-similarity via the multiplexing of a large number of traffic streams (in our simulations 32 or less are sufficient to yield high Hurst parameter estimates).

The rest of this paper is organized as follows. In the next section, we discuss related work. In the following section, we describe the network model and the details of our simulation method. This is followed by the main section which explores the effect of file size distribution on traffic self-similarity, including reliability and the role of the protocol stack, heavy-tailed versus non-heavy-tailed interarrival time distribution, resource variations, and traffic mixing. We conclude with a description of the effect of traffic self-similarity from a performance evaluation perspective, showing its quantitative and qualitative effects with respect to packet loss rate, retransmission rate, and mean queue length when both traffic self-similarity and network resources are varied.

## 2 Related work

Since the seminal study of Leland *et al.* [15] which set the groundwork for considering self-similar network traffic as an important modeling and performance evaluation problem, a string of work has appeared dealing with various aspects of traffic self-similarity [1, 2, 11, 12, 16, 18, 22, 28]. The research avenues may be broadly classified into two categories.

In the first category [11, 12, 15, 22, 28], traffic traces from physical network measurements are employed to identify the presence of scale-invariant burstiness, and models are constructed capable of generating synthetic traffic with matching characteristics. These papers show that long-range dependence is an ubiquitous phenomenon encompassing both local-area and wide-area network traffic. In addition, individual sources such as compressed VBR video streams have been shown to exhibit self-similarity as an inherent property.

In the second category are papers that have evaluated the effect of self-similar traffic on idealized or simplified networks [1, 2, 16, 18]. These papers show that long-range dependent traffic is likely to degrade performance. Principal results include the observation that the queue length distribution under self-similar traffic decays much more slowly when compared to short-range-dependent sources (e.g., Poisson). In contrast to these papers, the work we report here presents measurements of

network performance from detailed transport-level simulations reflecting more “realistic” network environments.

Our work extends the line of work of the first category of papers by formulating and verifying causal mechanisms which may be at play in real networks responsible for generating the self-similarity phenomena observed in diverse networking contexts. In particular, this paper extends and complements the framework set out in [28]. In [28], an idealized mechanism was proposed (the “ON/OFF” model) which was shown to be sufficient to generate self-similar time series as a result of the superposition of independent 0/1 renewal processes, each of which alternates between an ON and an OFF state. If the durations of the ON and OFF states are drawn from heavy-tailed distributions, then as the number of processes increases, the resulting series (formed by counting at regular intervals the number of processes in the ON state) will approach fractional Gaussian noise, a perfectly self-similar series. This result, although conceptually simple and elegant, lacks two important network modeling assumptions; one, resource-boundedness of real networks and the nonlinearity it induces, and two, feedback congestion control mechanisms employed by protocols including TCP which may introduce additional nonlinear interactions. This forms the starting point for our experimental investigations.

The relationship between file sizes and self-similar traffic was suggested by the work described in [9] which showed that self-similarity in World Wide Web traffic might arise due to the heavy-tailed distribution of file sizes present in the Web. Our study uses simulation to show that in a generic client/server environment which includes systems such as the World Wide Web, there is indeed a strong functional relationship between file sizes and traffic self-similarity.

Finally, an important question is whether file size distributions in practice are in fact typically heavy-tailed, and whether file size access patterns can be modeled as randomly sampling from such distributions. A definitive study of this question seems to not have been done. Previous measurement-based studies of file systems have recognized that file size distributions possess long tails, but they have not explicitly examined the tails for power-law behavior [24, 26, 19, 5, 23]. However, evidence of heavy tails in the distribution of file sizes has been noted in some specific contexts. In [9], it is shown that the size distribution of files found in the World Wide Web appears to be heavy-tailed with  $\alpha$  approximately equal to 1. This result is also in general agreement with measurements reported in [4]. A general study of Unix filesystems has found size distributions that appear to approximate power-law distribution [13]. Additional evidence of power-law behavior is present in some data on transmission lengths of network transfers. The authors in [7] show that the sizes of reads and writes to an NFS server appear to show power-law behavior. And, in [22], it was found that the upper tail of the distribution of data bytes in FTP bursts was well fit to a Pareto distribution with  $0.9 \leq \alpha \leq 1.1$ .

### 3 Network model and simulation set-up

#### 3.1 Network model

The network we use is given by a directed graph  $G = (V, E)$  consisting of  $n$  nodes  $v_1, v_2, \dots, v_n$  and  $m$  links  $e_1, e_2, \dots, e_m$ . Each output link  $e_j$  has a buffer  $b_j$ , link bandwidth  $\ell_j$ , and latency  $\tau_j$  associated with it.  $v_i$  is a *server node* if it has a probability density function  $p_i(X)$  where  $X \geq 0$  is a random variable denoting file size. We will call  $p_i(X)$  the *file size distribution* of server  $v_i$ .  $v_i$  is a *client node* (it may at the same time be also a server) if it has two probability density functions  $h_i(X), d_i(Y)$ ,  $X \in \{1, \dots, n\}$ ,  $Y \in \mathbf{R}_+$ , where  $h_i$  is used to select a server, and  $d_i$  is the *idle time distribution* which is used in determining the time of next request. In the context of reliable communication, if  $T_k$  is the time at which the  $k$ 'th request by client  $v_i$  was reliably serviced, the next request made by client  $v_i$  is scheduled at time  $T_k + Y$  where  $Y$  has distribution  $d_i$ . Requests from an individual client are directed to servers at random, independently and uniformly over the set of servers. In unreliable communication, this causal requirement is waived<sup>2</sup>. A 2-server, 32-client network configuration with a bottleneck link between gateways  $G_1$  and  $G_2$  is shown in Figure 1. This network configuration is used for most of the experiments reported below. In this configuration, the vast majority of traffic is flowing from servers to clients; hence we will refer to the total traffic arriving at  $G_2$  from servers as *upstream* traffic and traffic between  $G_2$  and  $G_1$  as *downstream* traffic.

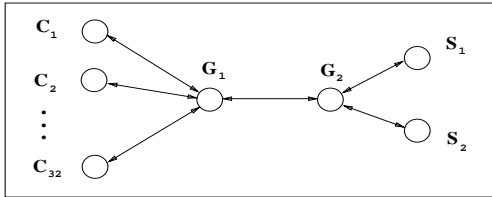


Figure 1: Network configuration.

A file is completely determined by its size  $X$  and it is split into  $\lceil X/M \rceil$  packets where  $M$  is the maximum segment size. The segments are routed through a packet-switched internetwork with packets being dropped at bottleneck nodes in case of buffer overflow. The dynamical model is given by all clients independently placing file transfer requests to servers where each request is completely determined by the file size. Notice that in the reliable communication model, clients independently determine the time of next request only in the sense that the coin corresponding to the idle time distribution  $d_i$  is independent from the ones used by other clients. All clients are coupled by *when* that coin is allowed to be tossed, i.e., the reliable completion time of the last request  $T_k$  which is a function of network state. We will require file size distribution  $p_i$  to be heavy-tailed (formal

---

<sup>2</sup>The primary motivation for this requirement was to keep our model as close to the ON/OFF model as possible making only a minimal set of essential changes.

definition is given later) but we will allow the idle time distribution  $d_i$  to be non-heavy-tailed (e.g., exponential).

To put this dynamic model into context, we compare it to the ON/OFF model [28] which provides theoretical grounding for our approach. The most serious drawback of the ON/OFF model—also the very feature which makes it simple and tractable—is the superposition of *independent* ON/OFF sources resulting in a linear system which ignores possible interactions among multiple traffic streams. However, in a real network, individual actions by the traffic sources can be as complicated as TCP Reno’s congestion control algorithm, and it is interesting that in spite of this significant difference, the traffic modeling aspect (for which the ON/OFF model was intended) has been so successful.

To provide a framework for discussing the data presented in later sections and partly address this question, we will make use of the following notions. Let  $B > 0$  be a constant denoting the bottleneck link buffer size in bytes, and let  $(X(t))_{t \geq 0}$  be a nonnegative sequence representing offered load (bytes arriving at the bottleneck link per unit time). The *clip* of  $X(t)$ ,  $X^*(t)$ , is defined as

$$X^*(t) = \max \{X(t) - B, 0\}.$$

That is,  $X^*(t)$  is just byte loss due to buffer overflow at time  $t$ . If  $s$  denotes the volume of data that is to be *reliably* transferred, then a minimum requirement on  $X(t)$  for it to have serviced this request by time  $T$  (starting at time 0) can be expressed as

$$\int_0^T X(t) dt \geq s + 2 \int_0^T X^*(t) dt.$$

Notice that the factor 2 is crucial in capturing a form of conservation or reliability (reflecting original data plus retransmissions).

This can be turned into an (static) optimization problem as follows. Assume there are  $n$  components making up the aggregate traffic stream,  $X(t) = \sum_{i=1}^n X_i(t)$ , each with its own reliable data volume  $s_i$  ( $s = \sum_{i=1}^n s_i$ ) that is to be transferred satisfying its private inequality. We seek the least  $T$  such that all  $n$  inequalities are satisfied. In a noncooperative game theory setting, each traffic component may be modeled as trying to minimize its own least completion time  $T_i$ . It is not difficult to see that per-session window-control of TCP implements such a greedy strategy. We will refer to the preference of selfish sources to minimize individual transfer time as *stretching in space* (i.e., making most of idle effective bandwidth), and the reliability or conservation constraint as *stretching in time*. Some consequences of the previous discussion to traffic self-similarity can be summarized as follows:

- (i) Under unreliable communication,  $X_i^*(t)$  large (i.e., high packet loss) has a similar effect as sampling  $s_i$  from a less heavy-tailed distribution than  $p_i$  with respect to the downstream traffic.

- (ii) Under reliable communication,  $X_i^*(t)$  large is compensated by stretching in time thus preserving the heavy-tailedness property of the downstream traffic.
- (iii) The larger the stretching in space, the weaker the long-range dependence (as defined in Sec. 4.2) of the down-stream traffic.

The third point can be understood by considering an extreme case where  $B = \infty$  and sources are maximally greedy leading to a process where the transfer of a file of size  $s$  is concentrated at a single point in time. With our assumption of non-heavy-tailed idle time distribution  $d_i$ , this leads to a stochastic process  $X(t)$  in which all of the dependency structure is attributable to  $d_i$ ; i.e.,  $X(t)$  is only short-range dependent. In Section 4, in tandem with showing that file size distribution by itself is sufficient to generate self-similarity, we will see various aspects of the interactions captured in (i)–(iii).

### 3.2 Simulation set-up

We have used the LBNL Network Simulator (**ns**) as the basis for our simulation environment [10]. **ns** is an event-driven simulator derived from S. Keshav’s REAL network simulator supporting several flavors of TCP (in particular, TCP Reno’s congestion control features—Slow Start, Congestion Avoidance, Fast Retransmit/Recovery) and router scheduling algorithms. Although not production TCP code, we have found **ns**’s emulation of TCP satisfactory for the purposes of studying congestion control as well as emulating reliable transport. A test suite description can be found in [10].

We have modified the distributed version of **ns** to model our interactive client/server environment. This entailed extending the one-way data flow restriction of a single **ns** TCP session to full-duplex, and implementing our client/server nodes as separate application layer agents. A UDP-based unreliable transport protocol was added to the existing protocol suite, and an aggressive opportunistic UDP-based agent was built to service file requests when using unreliable communication. In addition to the tracing functions that native **ns** provides, we added utilities for monitoring an expanded set of network statistics including provisions for detecting retransmission, reliable throughput, and computing file transmission completion times.

Our simulation results were obtained from several hundred runs of **ns**. Each run executed for 10000 simulated seconds, logging traffic in each 10 millisecond interval. The result in each case is a timeseries of one million data points; using such extremely long series increases the reliability of statistical measurements of self-similarity. Although most of the runs reported here were done with a 2-server/32-client bottleneck configuration as shown in Figure 1, other configurations were tested including performance runs with the number of clients varying in the range 1–132. The bottleneck link was varied from 1.5 Mbps up to OC-3 levels, and buffer sizes were varied in the range of 1K–128K. Non-bottleneck links were set at 10 Mbps the latency of each link was set to

15ms. The maximum segment size was fixed at 1K for most runs. For any reasonable assignment to bandwidth, buffer size, mean file request size, and other system parameters, it was found that by either adjusting the number of clients or the mean of the idle time distribution  $d_i$  appropriately, any intended level of network demand could be achieved. This enabled us to carry out performance evaluations by varying the two main network resources: bottleneck buffer size and bandwidth.

## 4 File size distribution and traffic self-similarity

As discussed in the Introduction and in Section 3.1, the mechanism we propose in this paper for the genesis of self-similarity is the repeated, interactive transfer of files over the network, when the sizes of those files are drawn from a heavy-tailed distribution. In this section we show that such a mechanism suffices to create self-similar network traffic, and we show that there is a tight functional relationship between the tail weight of the file size distribution and link-level self-similarity.

### 4.1 Heavy-tailed distributions

An important characteristic of our proposed mechanism for traffic self-similarity is that the sizes of files being transferred are drawn from a heavy-tailed distribution. A distribution is *heavy-tailed* if

$$P[X > x] \sim x^{-\alpha} \quad \text{as } x \rightarrow \infty$$

where  $0 < \alpha < 2$ . That is, regardless of the behavior of the distribution for small values of the random variable, if the asymptotic shape of the distribution follows a power law, we will call it heavy-tailed. One of the simplest heavy-tailed distributions is the *Pareto* distribution. The Pareto distribution is power-law over its entire range; its probability density function is given by

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

where  $\alpha, k > 0$ , and  $x \geq k$ . Its distribution function has the form

$$F(x) = P[X \leq x] = 1 - (k/x)^\alpha.$$

The parameter  $k$  represents the smallest possible value of the random variable.

Heavy-tailed distributions have a number of properties that are qualitatively different from distributions more commonly encountered such as the exponential or normal distribution. If  $\alpha \leq 2$ , the distribution has infinite variance; if  $\alpha \leq 1$  then the distribution has also infinite mean. Thus, as  $\alpha$  decreases, a large portion of the probability mass is present in the tail of the distribution. In practical terms, a random variable that follows a heavy-tailed distribution can give rise to extremely large file size requests with non-negligible probability. Our simulations are based on heavy-tailed distributions of the requests made for files in a client/server system; although this is not the same

thing as a heavy-tailed distribution of available files, the difference is not significant in the context of this work.

## 4.2 Effect of file size distribution

First, we demonstrate our central point: that interactive transfer of files whose size distribution is heavy-tailed will generate self-similar traffic—even when realistic network dynamics, including resource limitations and the flow control mechanisms of TCP, are taken into account.

Figure 2 shows time series plots of network traffic measured at the output buffer of a bottleneck link, *i.e.*, from gateway  $G_2$  to  $G_1$  in Figure 1. Each plot shows *downstream traffic* (*i.e.*, traffic flowing on the link from  $G_2$  to  $G_1$ ), measured in bytes per time unit, as a function of time.

The figure shows plots which span five orders of magnitude in time scale and four different file size distributions. The time units used vary from 10 ms in the lowest row to 100 sec in the uppermost row. The four columns show how traffic varies when the underlying file size distribution is Pareto with  $\alpha = 1.05$  (“very heavy-tailed,” extreme left), through  $\alpha = 1.35$  and  $\alpha = 1.95$  (not very heavy-tailed, second from right). The column of plots on the extreme right shows the effects of using an exponential distribution for file sizes. For each of the cases  $\alpha = 1.05, 1.35, 1.95$ , and exponential, the mean of the distribution is held constant at  $\approx 4.1\text{kB}^3$  and identical exponential distributions with mean 600 msec are used for idle time (“OFF” period). As a result, although the traffic patterns are very different, the mean offered load<sup>4</sup> on the network is the same in all plots.

Following [15, 28], these plots depict visually when self-similarity is present (or absent) in a time series, and they show how file size distribution can affect traffic burstiness. All plots appear roughly the same in a qualitative sense at the 10 ms level. However, progressing upward in the figure toward greater degrees of aggregation, the presence of self-similarity becomes evident, and as the tail weight of the file size distribution increases, the signs of scale-invariant burstiness become more pronounced. At the 100 sec aggregation level, the difference between long-range dependent traffic (on the left) and short-range dependent traffic (on the right) is quite clear. In addition, comparing the two columns on the right-hand-side of the figure shows that the traffic pattern of a heavy-tailed distribution with  $\alpha$  close to 2.0 is not significantly different from that of an exponential distribution when traffic is aggregated over time. That is, both are smoothed out at higher aggregation levels consistent with the lack of long-range dependence.

Although the time series plots shown in Figure 2 are quite helpful in gaining an intuitive picture of the effects of file size distribution on network traffic, a quantitative measure of self-similarity can

---

<sup>3</sup>This particular file size was used because of the difficulties in generating random variates with a given mean as  $\alpha \rightarrow 1$ . As  $\alpha$  approaches 1, the distributional mean diverges rapidly; the particular mean chosen in this case was one that could be produced as the sample mean in a number of datasets based on different random seeds.

<sup>4</sup>*Offered load* is the total file size requests (in bytes) generated by the clients over the 10000sec simulation interval.

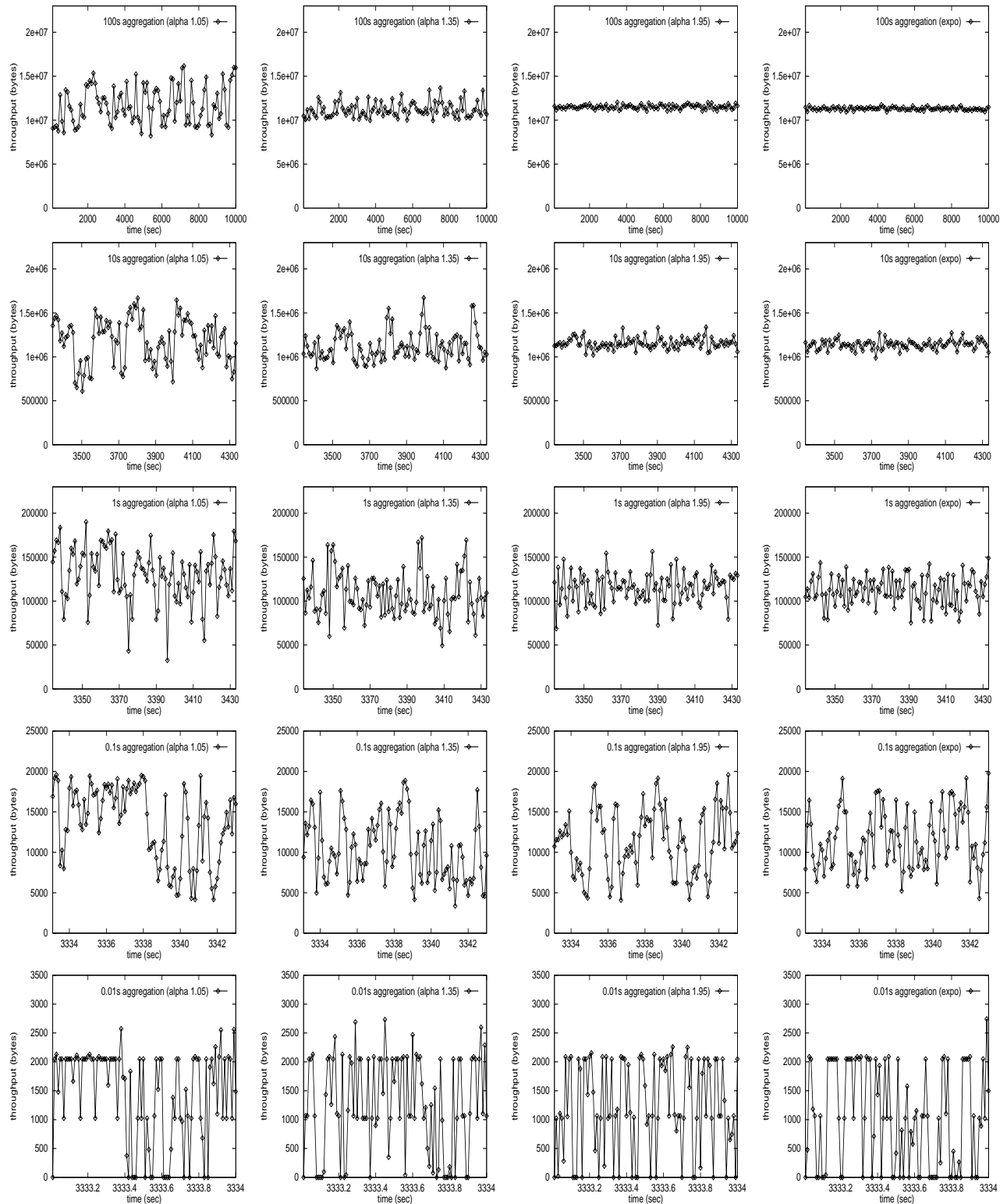


Figure 2: TCP run:  $\alpha = 1.05, 1.35, 1.95$ , and exponential. Inter-request idle time of 0.6 second and 32 clients.

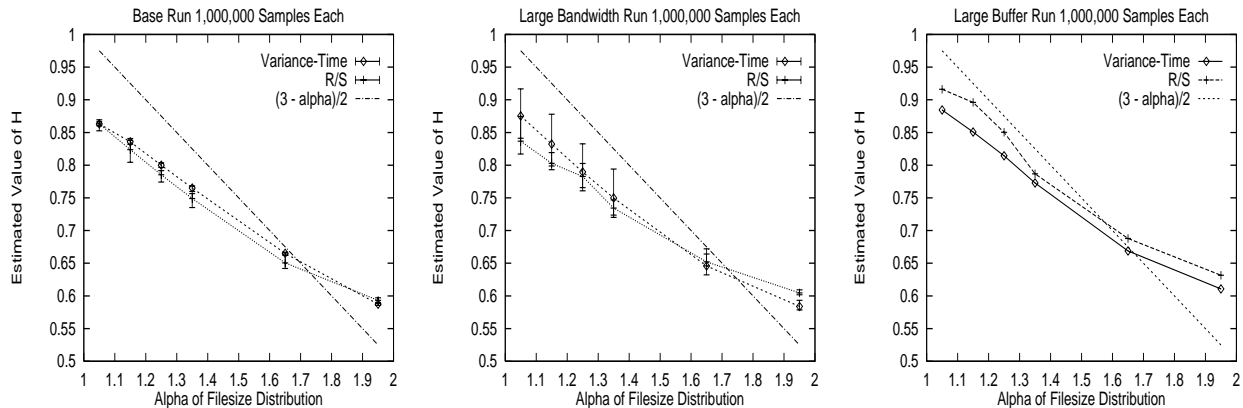


Figure 3: Hurst parameter estimates (TCP run): R/S and Variance-Time for  $\alpha = 1.05, 1.35, 1.65,$  and  $1.95$ . Base run (left), large bandwidth/large buffer (middle), large buffer (right)

be obtained using the *Hurst* parameter. The Hurst parameter  $H$  expresses the speed of decay of a series' autocorrelation function. A time series with long-range dependence has an autocorrelation function of the form

$$r(k) \sim k^{-\beta} \quad \text{as } k \rightarrow \infty$$

where  $0 < \beta < 1$ . Thus the autocorrelation function of such a process decays according to a power-law, as compared to the exponential decay exhibited by traditional traffic models. As  $\beta \rightarrow 0$ , the autocorrelation function decays more slowly, and the series has stronger long-range dependence.

The Hurst parameter is related to the rate of decay of the series' autocorrelation function by

$$H = 1 - \frac{\beta}{2}.$$

Hence, for self-similar series,  $1/2 < H < 1$ . As  $H \rightarrow 1$ , the degree of self-similarity increases. Thus an important test for self-similarity of a time series reduces to the question of determining whether  $H$  is significantly different from  $1/2$ .

In this paper we use two methods for testing self-similarity.<sup>5</sup> These methods are described more fully in [6] and are the same methods used in [15]. A summary of the relative accuracy of these methods on synthetic data sets can be found in [27]. The first method, the *variance-time plot*, relies on the slowly decaying variance of a self-similar series. The variance of  $X^{(m)}$  is plotted against  $m$  on a log-log plot where  $X^{(m)}$  is the random variable of the aggregated time series at level (or block size)  $m$ ; a straight line with slope  $\beta$  between  $-1$  and  $0$  is consistent with self-similarity, which translates to  $H$  via  $H = 1 - \beta/2$ . The second method, the *R/S plot*, uses the fact that for a self-similar data set, the *rescaled range* or *R/S* statistic grows according to a power law with

<sup>5</sup>A third method based on the periodogram was also used; however this method is believed to be sensitive to low frequency components in the series, which led in our case to wide spread in its estimates; it is omitted here.

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  $H$ . Figure 3 shows  $H$ -estimates based on variance-time and R/S methods for three different network configurations. Each plot shows  $H$  as a function of the Pareto distribution parameter for  $\alpha = 1.05, 1.15, 1.25, 1.35, 1.65$  and  $1.95$ . Each point on the plot is the average of 3 estimates based on different random seeds, and the error bars show the spread of the maximum and minimum in the estimates obtained.

Figure 3 (left) shows the results for our baseline TCP Reno case, in which network bandwidth and switch buffer size are both somewhat limited (1.5 Mb/s and 6K), resulting in a  $\approx 4\%$  packet drop rate for the most bursty case ( $\alpha = 1.05$ ). The plot shows that the Hurst parameter estimates vary with filesize distribution in a roughly linear manner. That is, in spite of the presence of limited network resources and the consequent interaction among the 32 traffic streams as they contend for those resources, the tail weight of the file size distribution directly determines link-level traffic self-similarity of the downstream traffic. The line,  $H = (3 - \alpha)/2$ , shows the values of  $H$  that would be predicted by the ON/OFF model in the idealized case corresponding to a fractional Gaussian noise process. Although their overall trends are similar (nearly coinciding at  $\alpha = 1.65$ ), clearly, the slope of the simulated system with resource limitations and reliable transport layer running TCP Reno’s congestion control is slightly less than  $-1$ , with an offset below the idealized line for  $\alpha$  close to 1 and above the line for  $\alpha$  close to 2. Figure 3 (middle) shows similar results for the case in which there is no significant limitation in bandwidth (155 Mb/s) leading to zero packet loss. There is noticeably more spread among the estimates which we believe to be the result of more variability in the traffic patterns, since traffic is less constrained by bandwidth limitations. However, the nearly linear relationship between  $H$  and  $\alpha$  remains essentially unchanged. Figure 3 (right) shows the results when bandwidth is somewhat limited, as in the baseline case, but buffer sizes at the switch are made large (64kB). Again, a consistent roughly linear relationship between the heavy-tailedness of file size distribution ( $\alpha$ ) and self-similarity of link traffic ( $H$ ) is observed.

In order to show that this relationship is not due to specific characteristics of the TCP Reno protocol, we repeated our baseline experiments using TCP Tahoe and TCP Vegas. The results are shown in Figure 4. The figures show trends that are essentially the same as in the baseline case for TCP Reno which indicates that specific differences in implementation of TCP’s flow control between Reno, Tahoe, and Vegas do not significantly affect the resulting traffic self-similarity.

Figure 5 (top) shows the relative file size distribution of client/server conversations over the 10000 second simulation time interval, organized into file size buckets (or bins), when each file transfer request is weighted by its size in bytes before normalizing to yield the relative frequency. The top-left figure shows that the Pareto distribution with  $\alpha = 1.05$  generates file size requests which are dominated by file sizes above the 64kB range during the 10000sec interval. On the other hand, the file sizes for Pareto with  $\alpha = 1.95$  and the exponential distribution (top-middle, top-right)

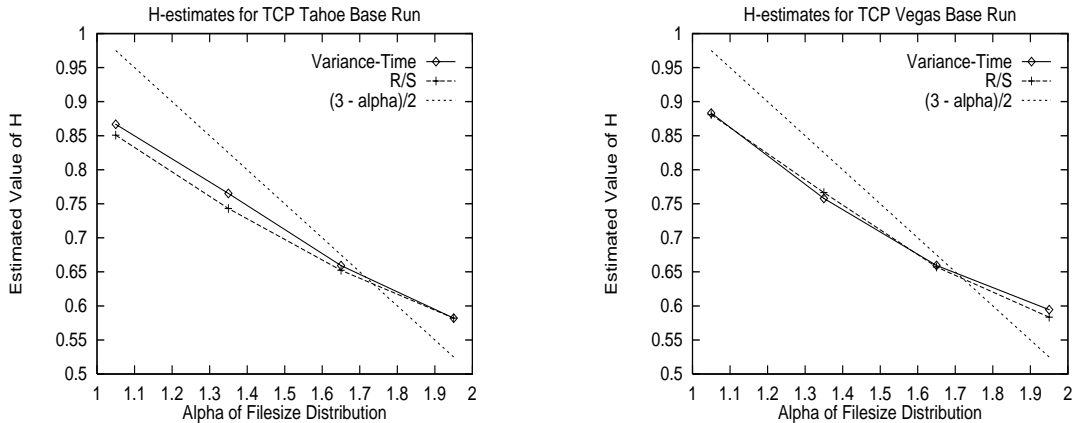


Figure 4: Hurst parameter as a function of  $\alpha$  for TCP Tahoe runs (left) and TCP Vegas runs (right) for the baseline set-up.

are concentrated on file sizes below 64kB, and in spite of their finer differences, their aggregated behavior (cf. Figure 2) is indistinguishable with respect to self-similarity<sup>6</sup>. Figure 5 (bottom) depicts the same graphs except that now file size requests are not weighted by their size. That is, the relative frequencies indicate the frequency of file size requests over the 10000sec simulation time interval. The bottom-left figure shows that small file sizes (300–700 bytes) occur most frequently even though the traffic during the simulation time is taken up by relatively few conversations whose durations dominate the network traffic due their large size. The bottom-middle and bottom-right figures show distributions which correspond more closely to their weighted counterparts in contrast to the “inverse” shape for the top-left/bottom-left pair.

Taken together these plots indicate that file size distribution, is able to directly affect the characteristics of downstream traffic with respect to self-similarity. They also indicate that the conclusion of the ON/OFF model—generation of self-similarity through the aggregation of heavy-tailed ON/OFF sources—applies at the high-level perspective of interactive file or message transfers in a client/server network environment when taking into account the influence of the transport/network layer in the protocol stack and the nonlinearity induced by traffic sources sharing bounded network resources.

### 4.3 Effect of idle time distribution and traffic mixing

The previous section showed that file size distributions are strongly correlated with traffic self-similarity. In this section we show that the relationship holds under a variety of network conditions.

<sup>6</sup>This also leads us to believe that subtle differences in how “small” file sizes are actually modeled are inconsequential to the conclusions at hand.

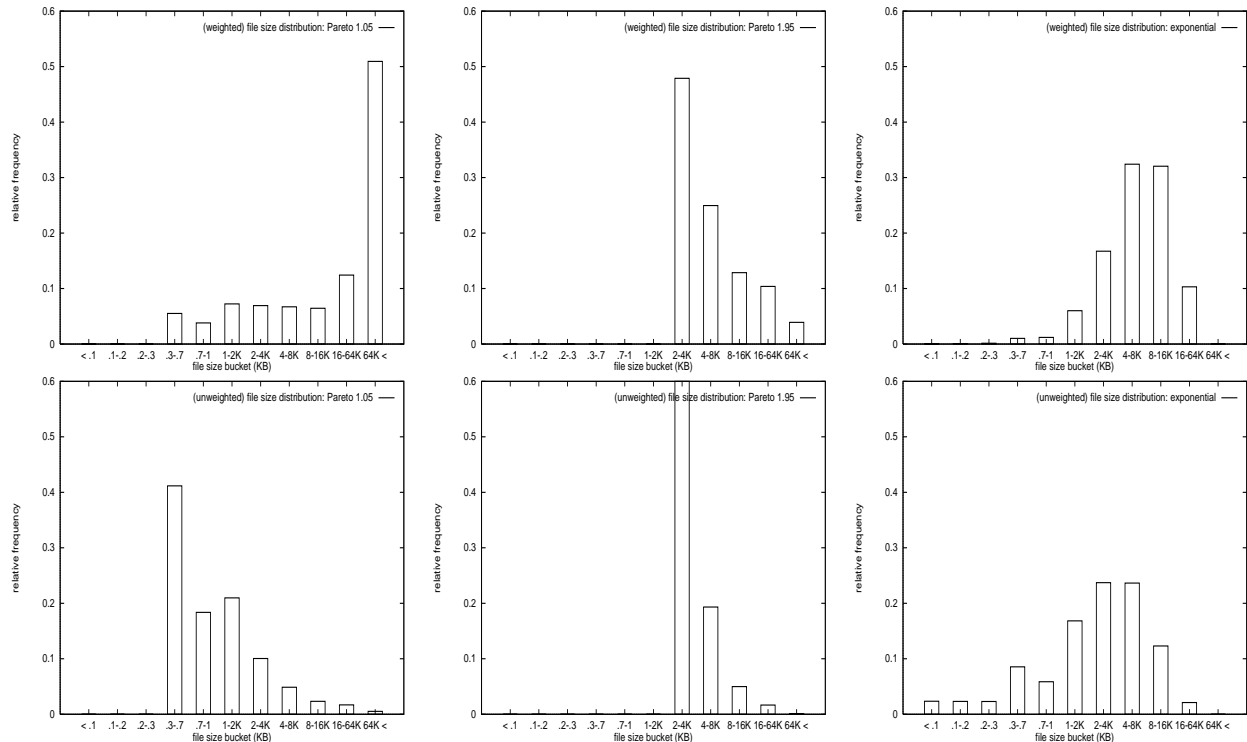


Figure 5: Top: Relative frequency of weighted file size distributions obtained from three 10000 second TCP runs—Pareto with  $\alpha = 1.05$  (left), with  $\alpha = 1.95$  (middle), and exponential distribution (right). Bottom: Relative frequency without weighting by file size.

#### 4.3.1 Idle time distribution: Exponential vs. Pareto

As noted earlier, all the runs so far were obtained with an exponential idle time distribution with mean 600 msec. Consistent with recent results reported in [29] we have found that in our simulations a heavy-tailed idle time distribution is not needed: a heavy-tailed file size distribution by itself is sufficient to produce self-similarity. Figure 6 (left) and (middle) show the H-estimates of the baseline configuration when the idle time distribution is exponential with mean 0.6 sec, and when idle time distribution is Pareto with  $\alpha = 1.05$  and mean 1.197 sec, while keeping the file size distribution Pareto. As the H-estimates show, the effect of a Pareto-modeled heavy-tailed idle time distribution is to boost long-range dependence when  $\alpha$  is close to 2, decreasing in effect as  $\alpha$  approaches 1.

This phenomenon may be explained as follows. For file size  $\alpha$  close to 2, the correlation structure introduced by heavy-tailed idle time is significant relative to the contribution of file size distribution, thus increasing the degree of self-similarity as reflected by the Hurst parameter. As file size  $\alpha$  approaches 1, however, the tail weight of the file size distribution becomes the dominating term, and the contribution of idle time with respect to increasing dependency is insignificant in comparison.

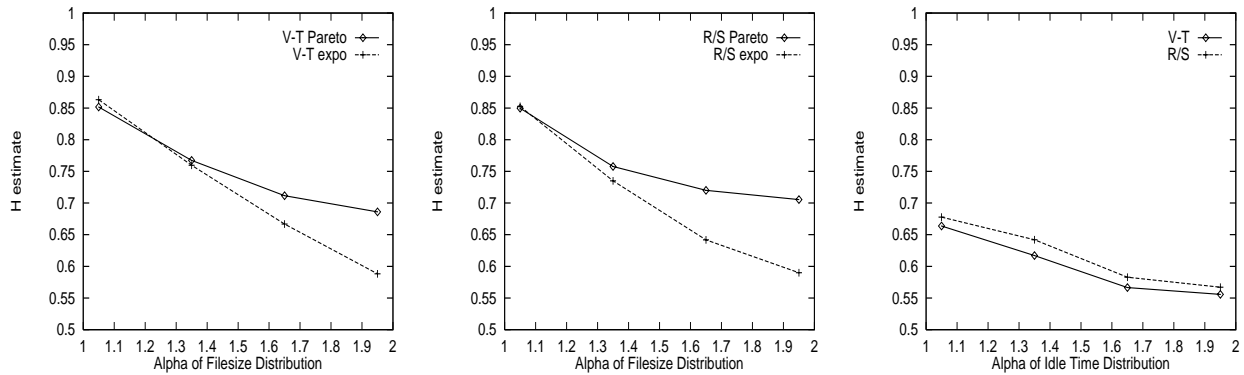


Figure 6: TCP run: Exponential idle time vs. Pareto idle time with Pareto file size distributions—Variance-Time (left), R/S (middle); Pareto idle times with exponential file size distribution (right)

Figure 6 (right) shows the Hurst parameter estimates when the file size distribution was exponential with mean 4.1kB, but the idle time distribution was Pareto with  $\alpha$  ranging between 1.05–1.95 with mean 1.197 sec at  $\alpha = 1.05$ . A positive trend in the H-estimates as the idle time distribution is made more heavy-tailed ( $\alpha \rightarrow 1$ ) is clearly discernible. However, the overall level of H-values is drastically reduced from the case when the file size distribution was Pareto indicating that the file size distribution is the dominating factor in determining the self-similar characteristics of network traffic.

### 4.3.2 Traffic mixing

Figure 7 shows the effect of making one of the file size distributions heavy-tailed ( $\alpha = 1.05$ ) and the other one exponential in the 2-server system. Downstream throughput is plotted against time where the time unit is 100 seconds. The left plot shows the case when both servers are Pareto with  $\alpha = 1.05$ . The right plot shows the case when both servers have exponential file size distributions. The middle plot is the combined case, where one server has a Pareto distribution with  $\alpha = 1.05$  and the other server has an exponential distribution. Figure 7 shows that the mixed case is less “bursty” than the pure Pareto case but more bursty than the pure exponential case. Performance indicators such as packet drop rate and retransmission rate (not shown here) exhibit a smooth linear degradation when transiting from one extreme to the other. That is, the presence of less bursty cross-traffic does not drastically smooth out the more bursty one, nor does the latter swallow up the smooth traffic entirely. Traffic mixing was applied to all combination pairs for  $\alpha = 1.05, 1.35, 1.65, 1.95$  keeping one server fixed at  $\alpha = 1.05$ . A similar “additive” mixing was observed with respect to performance. The H-values for the three cases are approximately 0.86, 0.81, and 0.54, respectively.

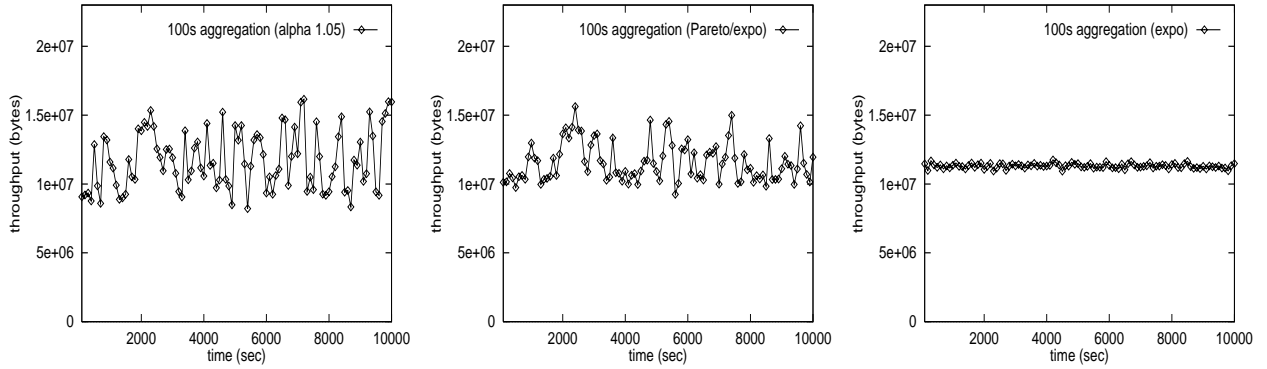


Figure 7: Traffic mixing effect for two file size distributions Pareto  $\alpha = 1.05$  and exponential at 100 second aggregation level: Both servers are Pareto (left); one server is Pareto, the other one is exponential (middle); both servers are exponential (right).

#### 4.4 Effect of network topology

In this subsection, we examine the effect of changing the network topology. Figure 8 shows a variation in network topology in which the 32 clients are organized in a caterpillar graph with 4 articulation points (gateways  $G_3, G_4, G_5, G_6$ ), each containing 8 clients, where traffic volume intensifies as we progress from gateway  $G_6$  to  $G_2$  due to the increased multiplexing effect. Link

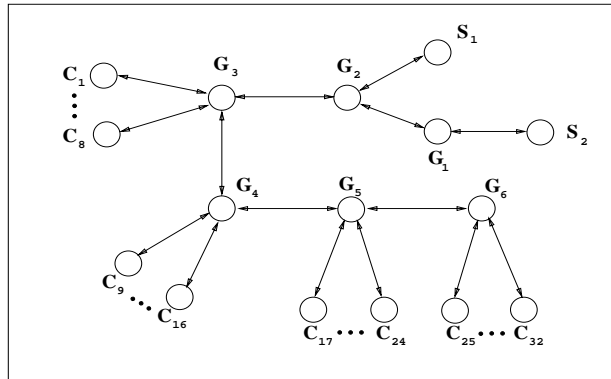


Figure 8: Variation in network topology.

traffic was measured at the bottleneck link between  $G_3$  and  $G_2$  which was set at 1.544Mbps. All other links were set at 10Mbps. Figure 9 compares the Hurst parameter estimates of the extended topology against the H-values of the base topology. We observe that for both V-T and R/S, the degree of self-similarity is not significantly different.

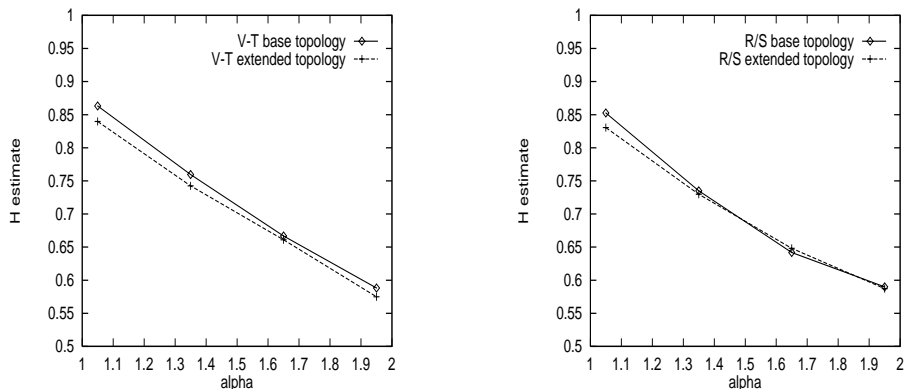


Figure 9: Hurst parameter estimates for extended topology: V-T (left) and R/S (right).

## 4.5 Protocol stack and traffic self-similarity

In this subsection, we will explore the role of the protocol stack with respect to its effect on traffic self-similarity. We will concentrate on the functionality of the transport layer and its tendency to influence the characteristics of downstream traffic via its two end-to-end mechanisms: reliable packet transport and congestion control.

### 4.5.1 Clipping

Clipping was defined in Section 3.1 as packet loss due to buffer overflow. We will further restrict our attention to packet loss where total traffic is not conserved—that is, dropped packets are not retransmitted to achieve reliability.<sup>7</sup> If  $X(t)$  is a traffic stream and  $X^*(t)$  its clip (the time series of dropped packets), then an easy consequence of  $X^*(t)$  being large is the reduction in self-similarity of the down-stream traffic due to its “equivalence” to the source having sampled from a less heavy-tailed file size distribution (observation (i) of Section 3.1).

Figure 10 shows the Hurst parameter estimates for a 32-client/2-server system with exponential idle time distribution and Pareto file size distributions for  $\alpha = 1.05, 1.35, 1.65,$  and  $1.95$ . In these experiments, communication is unreliable; they use a UDP-like transport protocol which is driven by an extremely greedy application whose output rate, upon receiving a client request, was essentially only bounded by the local physical link bandwidth. Referring back to Figure 1 in Section 3.1, traffic was measured on the outlinks going from  $S_1$  and  $S_2$  to the bottleneck node  $G_2$  (upstream traffic), and upon leaving on the bottleneck link traversing from  $G_2$  to  $G_1$  (downstream traffic) after being multiplexed at  $G_2$ . The network configuration was such that when the file size distribution

<sup>7</sup>We note that for this definition to be precise, further assumptions need to be made to handle situations such as reliable transmission through error-correcting codes where information content as captured by the description (Kolmogorov) complexity of an object may be significantly smaller than its encoded size. However, this distinction does not affect the present discussion.

was Pareto with  $\alpha = 1.95$ , very little packet loss occurred at the bottleneck node, with source burstiness (as represented by decreasing  $\alpha$ ) being the sole control variable. The H-estimates show that as source burstiness is increased, the estimated Hurst parameter of the down-stream traffic decreases relative to its value in the upstream traffic, indicating a clipping effect.

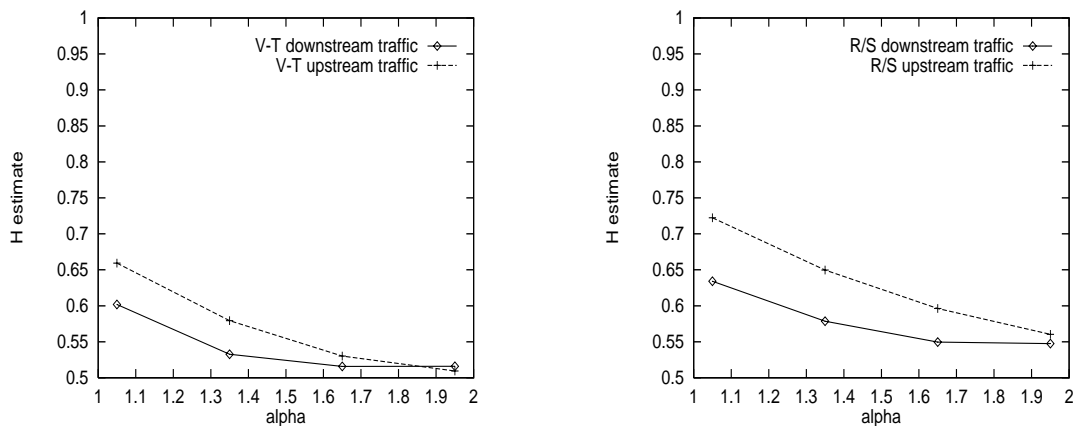


Figure 10: UDP run: Erosion of long-range dependence through excessive buffer overflow; Variance-Time (left) and R/S (right).

Another interesting point is the already low Hurst estimate of the up-stream traffic at Pareto  $\alpha = 1.05$ . We believe this is due to the *stretching in space* effect discussed in Section 3.1. That is, given an exponential idle time distribution, the extremely greedy nature of the UDP-based application encourages traffic to be maximally stretched out in space, and stretching in time is achieved only for very large file size requests. For UDP-based applications, files need at least  $s/B$  seconds to be transferred, where  $s$  is the file size and  $B$  denotes the local physical link bandwidth (the interface buffer size was made sufficiently large so as to accommodate the maximum link rate  $B$  without packet loss). By observation (iii) of Section 3.1, stretching in space, by concentrating more of its mass on a shorter time interval, decreases the dependency structure at lower time scales, making the traffic less self-similar. Of course, when very large file sizes occur with non-negligible frequency ( $\alpha$  close to 1), stretching in space will be achieved proportional to  $s/B$ , producing long-range dependence, albeit at a reduced level when compared with reliable communication.

#### 4.5.2 Stretching

The previous section explored the stretching effect (in space or time) in the context of unreliable communication, and found that for extremely greedy, unreliable communication, stretching in time occurs only for very large file transfers, the amount of stretching being proportional to file size. Here, we explore the stretching issue when resources at gateways are allowed to vary.

First, consider the case when the bottleneck link output buffer is allowed to vary. The primary

performance effect of increasing the buffer size lies in reducing packet loss (see Section 4.6 for performance evaluations). In both reliable and unreliable communication, the increased “reservoir” allows for more packets to be in-waiting, thus by the “reverse” file size sampling argument increasing the dependency structure of the downstream traffic. Figure 11 shows an upward trend in the Hurst parameter as buffer size is increased when unreliable communication is employed. The gap is most pronounced between the Hurst parameter estimates when the smallest (3kB) and the largest (64kB) buffer sizes are compared. However, little can be said about their magnitude due to the size of the variations in the estimators relative to the magnitude of the shift. Figure 12 (top row) shows a similar effect when packet transport is reliable. Again, only a general trend is discernible with the shift effect most evident between the largest and smallest buffer sizes.

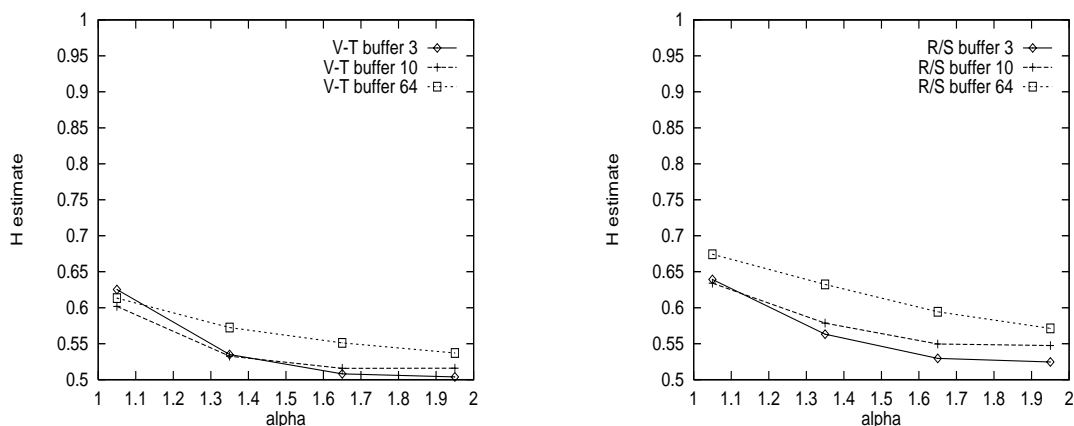


Figure 11: UDP run: Hurst parameter estimates for  $\alpha = 1.05, 1.35, 1.65,$  and  $1.95$  as bottleneck buffer size is varied. Variance-Time (left) and R/S (right)

Second, let us consider the case when the bottleneck link bandwidth is varied while keeping everything else fixed. If the bandwidth is large, then stretching in space is encouraged which results in decreased self-similarity in the downstream traffic. Figure 12 (bottom row) shows a slight decreasing trend in the H-estimates as link bandwidth is increased when reliable communication is employed. Thus, even though increasing either resource results in smaller packet loss and improved performance, their effect on the characteristics of downstream traffic is dissimilar. Whereas increased buffering is conducive to preserving dependency in the downstream traffic, increasing bandwidth has the opposite effect albeit less pronounced due to the measured nature of TCP Reno’s flow control.<sup>8</sup> The cost of the latter (stretching-in-space) is reflected by an increase in uncorrelated large bursts which without further control actions may lead to severe packet loss and

<sup>8</sup>Although Fast Retransmit/Recovery somewhat alleviates TCP Reno’s conservative stretching-in-time policy for achieving stability, the reliance of Congestion Avoidance on packet loss detection for triggering compensatory actions causes retransmit waits which further add to the stretching-in-time effect. TCP Vegas’ enhanced Congestion Avoidance feature may be viewed as counteracting this tendency [8, 3].

performance degradation (cf. Figure 11 in Section 4.5.1 and Figure 16 in Section 4.6.1). The effect of prolonged stretching in time, on the other hand, is conducive to amplifying long-range dependency by generating long packet trains which in the presence of multiple sources is prone to result in self-similar traffic patterns. The previous observation also points to why the linear ON/OFF model may have been successful in modeling the output characteristics of a complicated nonlinear system. In some sense, the effect of the unaccounted-for nonlinearity seems to be reflected back as a stretching effect, thus conforming with the model’s original suppositions. That is, interaction among traffic sources contending for network resources seems to result in mutual stretching-in-time, generating elongated packet trains with “holes and dips” due to retransmit waits.

The resiliency of reliable communication under self-similar traffic conditions is achieved by stretching a conversation over time so as to reduce the probability of packet loss thus attaining efficient reliable transmission in the sense of Section 3.1. The conservativeness inherent in additive-increase/multiplicative-decrease feedback algorithms including TCP Reno [14] suggests that too much time-stretching is already being done, whereas theoretical characterizations on the difficulty of achieving stability and optimality in a rate-controlled dynamic queueing network [25, 17, 21] suggest that perhaps not *very* much can be done to alleviate this problem in a fundamental way.

## 4.6 Network performance

In previous sections we showed that even though a number of the simplifying assumptions underlying the ON/OFF model are unrealistic in general, the effect of reliable, flow-controlled transmission as facilitated by TCP Reno lies in preserving the long-range dependence of the input traffic by stretching the transmission over time. In part, this has the effect of “incorporating” the interaction among multiple traffic streams—ignored in the ON/OFF model—by subsuming one of its main consequences (i.e., stretching in time) via conformation with the heavy-tailedness assumption of ON/OFF packet trains. In this section, we show performance results which further corroborates this point. A more complete study of the performance implications of self-similarity is presented in [20]. We first present performance characteristics of TCP Reno under a self-similar load; then in the next section we contrast this behavior with a UDP-like transport protocol.

### 4.6.1 Performance evaluation under reliable communication

**Control variables: bottleneck buffer size and self-similarity.** The smooth performance degradation of TCP in the presence of extremely bursty traffic as bottleneck buffer size and traffic self-similarity are varied is shown in Figure 13. The leftmost plot in the top row of the figure shows packet loss rate as  $\alpha$  is varied for buffer sizes ranging from 2kB to 64kB. It shows that for small buffer sizes, packet loss rate is virtually unaffected by the self-similarity of traffic; however, when

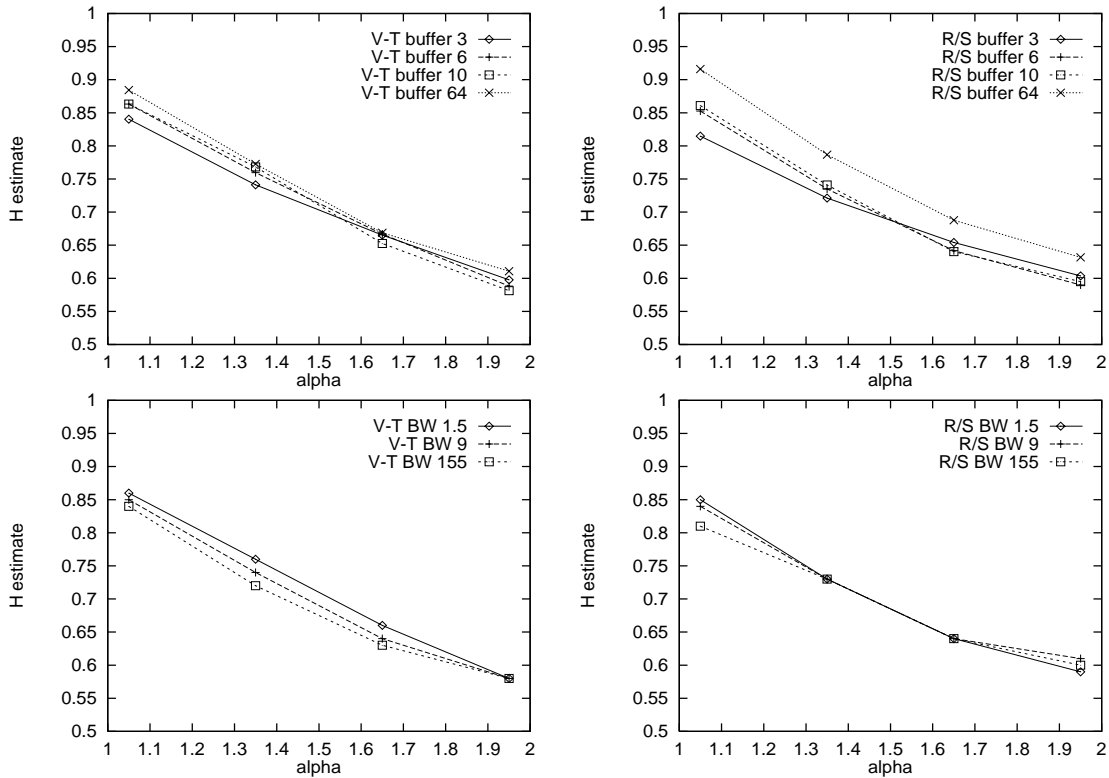


Figure 12: TCP run: Hurst parameter estimates for  $\alpha = 1.05, 1.35, 1.65,$  and  $1.95$  as bottleneck buffer size is varied (top row), and bottleneck bandwidth is varied (bottom row)—Variance-Time (left column) and R/S (right column).

buffer sizes get large,  $\alpha = 1.95$  traffic shows virtually no packet drops while strongly correlated traffic incurs a drop rate of about 1%. A similar effect is shown in the middle plot of the top row of the figure, which depicts packet retransmission rate for the same range of conditions. What is striking about these two figures is the lack of drastic change in the two performance variables as self-similarity is varied. Figures 13 (left, bottom row) and (middle, bottom row) show the same data set but with buffer size in the abscissa. For any fixed  $\alpha$ , a saturation effect is visible as buffer size is increased. It also again shows that for small buffer sizes, there is little performance difference between highly self-similar ( $\alpha \approx 1.05$ ) and less bursty traffic ( $\alpha \approx 1.95$ ).

Figure 13 (right, top row) gives some evidence of why TCP is able to avoid drastic increases in the packet drop rate. The plot depicts the average buffer occupancy for the same range of buffer sizes. Unlike the case of packet loss and retransmission rate, the buffer occupancy when buffer size is 64kB increases sharply with increasing self-similarity. In other words, when the buffer size is small such that weakly correlated traffic is sufficient to affect significant packet loss, increasing self-similarity has little additional effect with respect to future packet drops since TCP Reno’s feedback

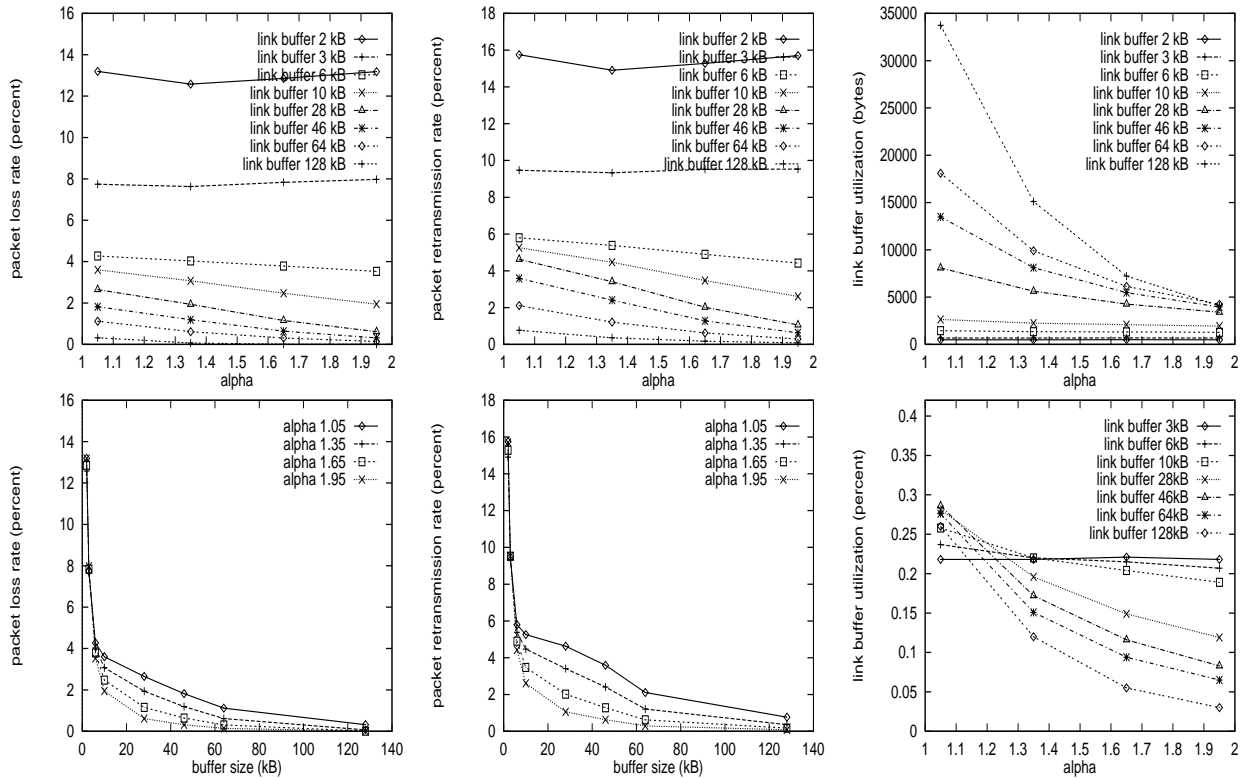


Figure 13: TCP run. Top row: Packet loss rate, packet retransmission rate, and mean queue length as a function of  $\alpha$ . Bottom row: Packet loss rate (left), packet retransmission rate (middle) as a function of buffer size; bottleneck buffer utilization (right) as a function of  $\alpha$ .

control is already causing the transmission rate to back off. However, when the buffer size is large so that the higher burstiness of long-range dependent traffic is allowed to make a differentiated effect and “stand out,” a proportionately larger buffer occupancy is required to yield a smooth degradation in performance. That is, the marginal utility of additional buffer space is exhibiting diminishing returns. Figure 13 (right, bottom row) is another depiction of the data in the top-right figure where the ordinate represents mean buffer occupancy normalized by the buffer capacity. Likewise, our measurements of reliable throughput (not shown) indicate a similarly smooth, linear degradation in throughput for all buffer sizes as  $\alpha$  decreases.

**Control variables: bottleneck bandwidth and self-similarity.** Whereas in the previous case the bottleneck buffer capacity was the main control variable in conjunction with the degree of traffic self-similarity, in this case we vary bottleneck bandwidth and evaluate its effect. Figure 14 (left, top row) shows the effect of varying  $\alpha$  for bandwidths in the range 1.5Mbps–9Mbps with respect to packet loss rate. A gradual (linear to sublinear) change in performance as a function of  $\alpha$  is observed. Figure 14 (right, top row) shows an analogous plot with packet retransmission rate in

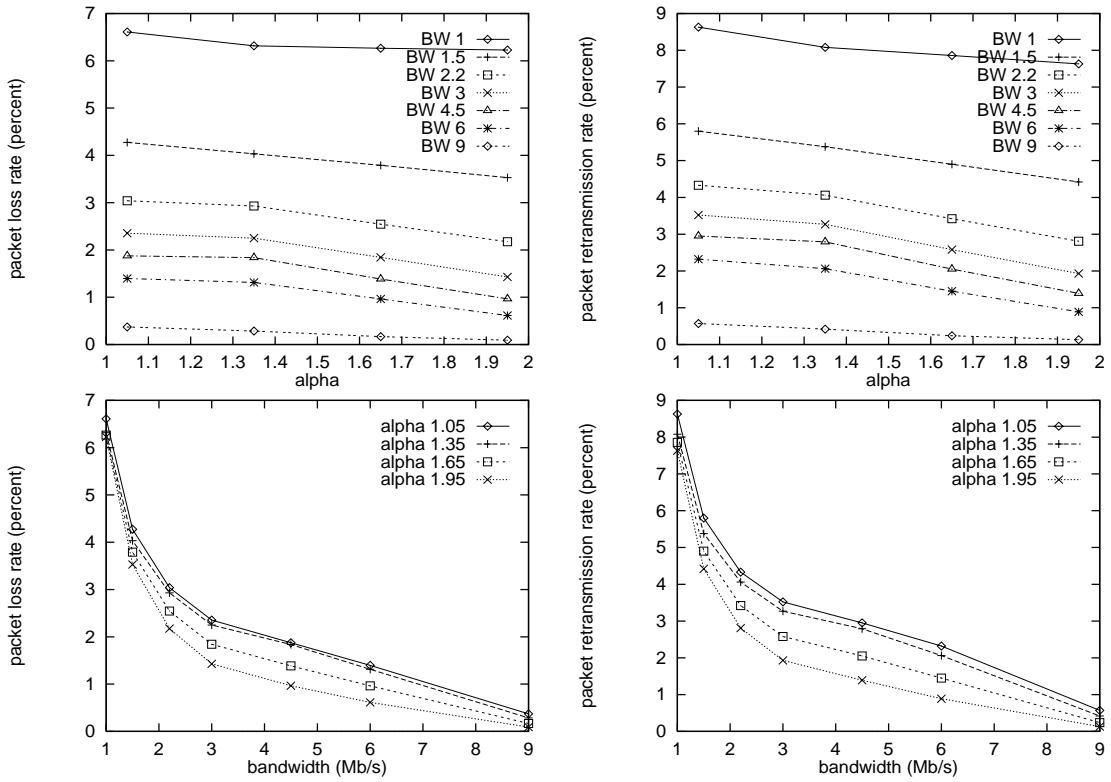


Figure 14: TCP run. Top row: Packet loss rate (left) and packet retransmission rate (right) as a function of  $\alpha$ . Bottom row: Packet loss rate (left) and packet retransmission rate (right) as a function of bandwidth.

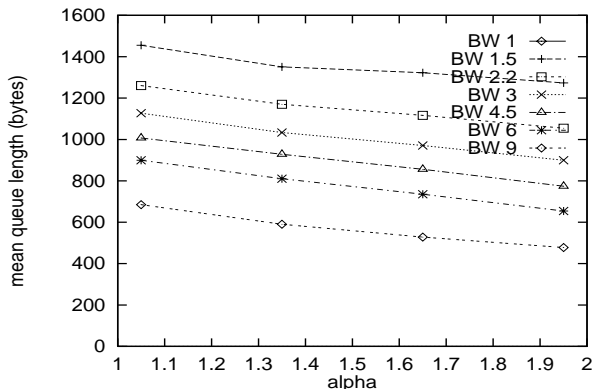


Figure 15: TCP run. Mean queue length as a function of  $\alpha$  for different bottleneck bandwidths.

place of packet loss rate. The bottom row plots of Figure 14 depict the same data set with the abscissa representing link bandwidth. For any fixed  $\alpha$ , a saturation effect kicking in between 1.5 Mbps–3 Mbps is visible. Figure 15 shows the mean queue length when buffer size is 6kB for different values of  $\alpha$  and bottleneck link bandwidth. Whereas in Figure 13 (right, top row) the curvature of the mean queue length curve increases as buffer size is increased (64kB) corresponding to simulation runs with link bandwidth fixed at 1.5 Mbps, increasing the link bandwidth while keeping the buffer capacity fixed at 6kB does not carry an analogous effect. Even though both buffer size 64kB/link bandwidth 1.5Mbps and buffer size 6kB/link bandwidth 9Mbps runs yield low packet loss rates of 1.1%, 0.4%, respectively (for  $\alpha = 1.05$ ), the latter does not seem to induce an increased buffer occupancy as  $\alpha$  is decreased.

#### 4.6.2 Performance evaluation under unreliable communication

To highlight the performance-moderating effects of TCP, we contrast its performance with that of a UDP-like protocol in this section.

Figure 16 (left) shows the packet loss rate as  $\alpha$  is varied for different values of bottleneck buffer capacity. Compared to the smooth, linear increase seen in Figure 13 (left, top row) for the TCP Reno case, unreliable transport induces a drastic, superlinear increase in packet loss as  $\alpha \rightarrow 1$ . That is, the high burstiness associated with self-similar traffic is directly reflected in high packet drops at the bottleneck link, without the intervention of TCP’s reliable transport mechanism and congestion control to stem the flow. In Section 4.5.1, we have shown an effect of this traffic clipping to be a reduction in self-similarity of downstream traffic as reflected by the decrease in the Hurst parameter. However, the offered load still shows the effects of strong self-similarity, as shown in Figure 16 (right). This figure shows a superlinear degradation in link utilization as self-similarity is increased. Also, notice the extremely low utilization level (1–6%) despite the massive packet drops indicated by the plots on the right side of the figure. Link utilization in the TCP Reno case

(not shown here) is in the range of 60-70%, and is much more level due to its traffic-shaping effect. Finally, Figure 17 also shows the increasing curvature of the mean queue length graphs as buffer size is increased. This behavior is very similar to that in the case of TCP as seen in Figure 13 (right, top row). It indicates that, regardless of transport protocol, queue length distribution for highly self-similar traffic is much more slowly decaying than weakly self-similar traffic and Poisson sources which is consistent with the observations made in [16, 18].

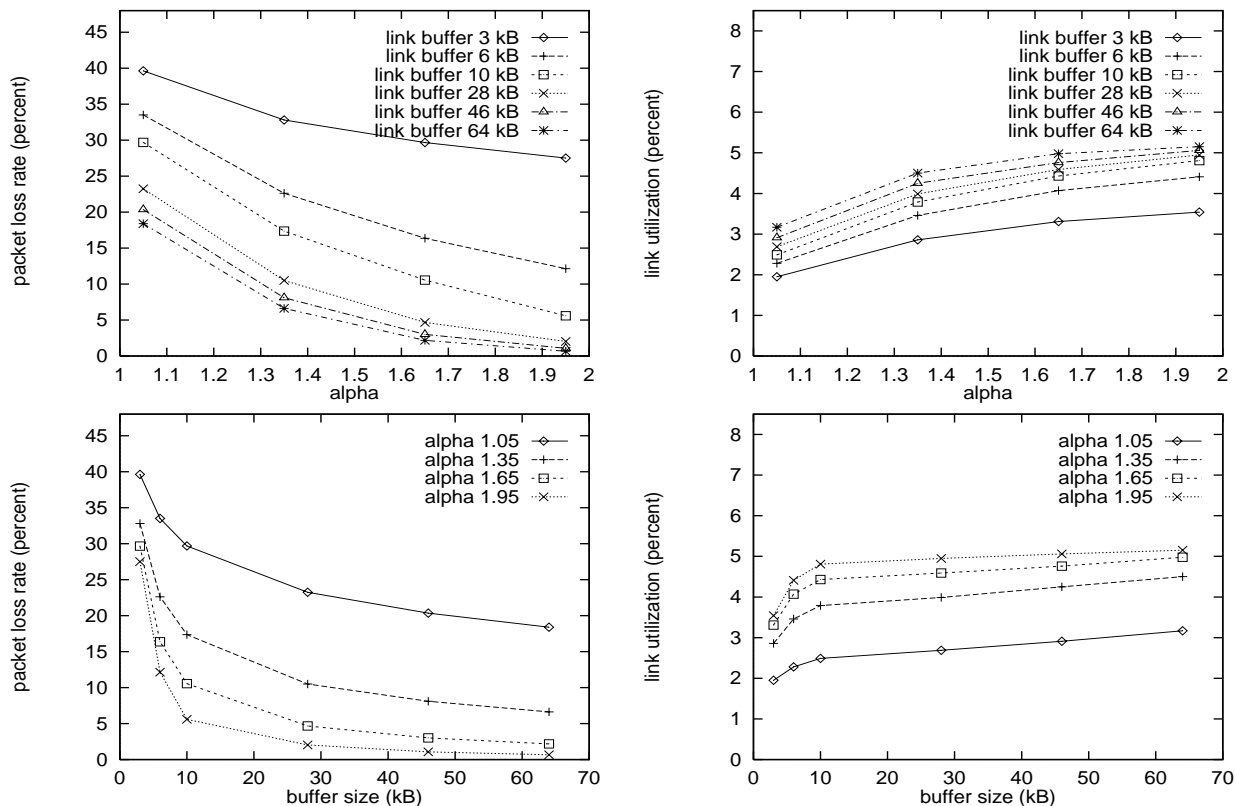


Figure 16: UDP run: packet loss rate and link utilization as a function of  $\alpha$  and buffer size (top). Same data with the abscissa denoting buffer size instead of  $\alpha$ .

## 5 Conclusion

In this paper, we have shown that self-similarity in network traffic can arise due to a particularly simple cause: the reliable transfer of files drawn from heavy-tailed distributions. Such a high-level explanation of the self-similarity phenomenon in network traffic is appealing because there is evidence that file systems indeed possess heavy-tailed file size distributions [9, 4, 13, 22]. It also relates a networking problem—traffic characterization—to a system-wide cause which has traditionally been considered outside the networking domain. The growth and prevalence of multi-

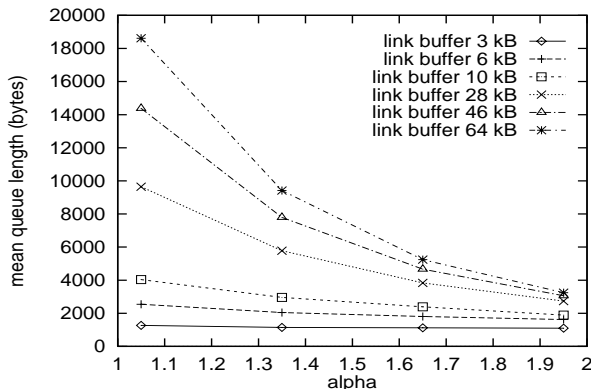


Figure 17: UDP run. Mean queue length as a function of  $\alpha$  and buffer size.

media traffic only aggravates the situation by facilitating the structural conditions for inducing self-similar network traffic, and our work supports recent efforts directed at managing network resources in a more integrated way (“Middleware” research) in which issues such as caching and server selection may turn out to be relevant in formulating effective solutions for congestion control.

We have shown that the relationship between file size distribution and traffic self-similarity is not significantly affected by changes in network resources, topology, traffic mixing, or the distribution of interarrival times. We have also shown that reliability and flow control mechanisms in the transport layer of the protocol stack give rise to a traffic-shaping effect that preserves self-similarity in network traffic. This helps explain why the ON/OFF model [28], in spite of ignoring traffic interactions through resource limitations and feedback control, may have been successful in modeling observed traffic characteristics. The coupling between traffic sources sharing and contending for common network resources leads to a stretching-in-time effect which reflects back to the ON/OFF model by conforming, at a qualitative level, to its simplifying suppositions.

Finally, we have shown that network performance, as measured by packet loss and retransmission rate, declines smoothly as self-similarity is increased under reliable, flow-controlled packet transport. The only performance indicator exhibiting a more sensitive dependence on self-similarity was mean queue length, and this concurs with the observation that queue length distribution under self-similar traffic decays more slowly than with Poisson sources. In contrast, we showed that performance declines drastically with increasing self-similarity when a UDP-like unreliable transport mechanism was employed. This gives a sense of the moderating effect of TCP on network performance in the presence of highly bursty traffic. A more detailed study of the performance evaluation question including quality-of-service (QoS) trade-offs under self-similar traffic conditions and the relative effectiveness of increasing link bandwidth and buffer capacity can be found in [20].

## 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] J. Ahn, P. Danzig, Z. Liu, and L. Yan. Evaluation of TCP Vegas: Emulation and experiment. In *Proc. ACM SIGCOMM '95*, pages 185–195, 1995.
- [4] M. F. Arlitt and C. L. Williamson. Web server workload characterization: The search for invariants, May 1996. Preprint. To appear in *Proc. 1996 ACM SIGMETRICS*.
- [5] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout. Measurements of a distributed file system. In *Proceedings of the Thirteenth ACM Symposium on Operating System Principles*, pages 198–212, Pacific Grove, CA, October 1991.
- [6] Jan Beran. *Statistics for Long-Memory Processes*. Monographs on Statistics and Applied Probability. Chapman and Hall, New York, NY, 1994.
- [7] R. R. Bodnarchuk and R. B. Bunt. A synthetic workload model for a distributed system file server. In *Proceedings of the 1991 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 50–59, 1991.
- [8] 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.
- [9] M. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes, May 1996. Preprint. To appear in *Proc. 1996 ACM SIGMETRICS*.
- [10] Sally Floyd. Simulator tests. Available in <ftp://ftp.ee.lbl.gov/papers/simtests.ps>. Z. ns is available at <http://www-nrg.ee.lbl.gov/nrg>, July 1995.
- [11] M. Garrett and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In *Proc. ACM SIGCOMM '94*, pages 269–280, 1994.
- [12] 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.
- [13] Gordon Irlam. Unix file size survey - 1993. Available at <http://www.base.com/gordoni/ufs93.html>, September 1994.
- [14] Hyogon Kim. *A Non-Feedback Congestion Control Framework for High-Speed Data Networks*. PhD thesis, University of Pennsylvania, 1995.
- [15] 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.
- [16] 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.
- [17] A. Mukherjee and J. Strikwerda. Analysis of dynamic congestion control protocols - a Fokker-Planck approximation. In *Proc. ACM SIGCOMM '91*, pages 159–169, 1991.
- [18] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16:387–396, 1994.

- [19] J. K. Ousterhout, H. Da Costa, D. Harrison, J. a Kunze, M. Kupfer, and J. G. Thompson. A trace-driven analysis of the UNIX 4.2 BSD file system. In *Proceedings of the Tenth ACM Symposium on Operating System Principles*, pages 15–24, Orcas Island, WA, December 1985.
- [20] K. Park, G. Kim, and M. Crovella. On the effect of traffic self-similarity on network performance. Technical report, Boston University Computer Science Department, 1996.
- [21] Kihong Park. Warp control: a dynamically stable congestion protocol and its analysis. In *Proc. ACM SIGCOMM '93*, pages 137–147, 1993.
- [22] V. Paxson and S. Floyd. Wide-area traffic: the failure of Poisson modeling. In *Proc. ACM SIGCOMM '94*, pages 257–268, 1994.
- [23] K. K. Ramakrishnan, P. Biswas, and R. Karedla. Analysis of file I/O traces in commercial computing environments. In *Proceedings of the 1992 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 78–90, June 1992.
- [24] M. Satyanarayanan. A study of file sizes and functional lifetimes. In *Proceedings of the Eighth ACM Symposium on Operating System Principles*, December 1981.
- [25] Scott Shenker. A theoretical analysis of feedback flow control. In *Proc. ACM SIGCOMM '90*, pages 156–165, 1990.
- [26] Alan Jay Smith. Analysis of long term file reference patterns for application to file migration algorithms. *IEEE Transactions on Software Engineering*, 7(4):403–410, July 1981.
- [27] M. S. Taqqu, V. Teverovsky, and W. Willinger. Estimators for long-range dependence: an empirical study, 1995. Preprint.
- [28] 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.
- [29] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level. In *Proc. SIGCOMM '95*, pages 100–113, Boston, MA, 1995.