In *Proceedings of the 1997 SPIE International Conference on Performance and Control of Netork Systems*

On the Effect of Traffic Self-similarity on Network Performance

Kihong Park[a] Gitae Kim[b] Mark Crovella[b]

[a]Department of Computer Sciences, Purdue University
West Lafayette, IN 47907

[b]Computer Science Department, Boston University
Boston, MA 02215

ABSTRACT

Recent measurements of network traffic have shown that self-similarity is an ubiquitous phenomenon present in both local area and wide area traffic traces. In previous work,[1] we have shown a simple, robust application layer causal mechanism of traffic self-similarity, namely, the transfer of files in a network system where the file size distributions are heavy-tailed. In this paper, we study the effect of scale-invariant burstiness on network performance when the functionality of the transport layer and the interaction of traffic sources sharing bounded network resources is incorporated.

First, we show that transport layer mechanisms are important factors in translating the application layer causality into link traffic self-similarity. Network performance as captured by throughput, packet loss rate, and packet retransmission rate degrades gradually with increased heavy-tailedness while queueing delay, response time, and fairness deteriorate more drastically. The degree to which heavy-tailedness affects self-similarity is determined by how well congestion control is able to shape a source traffic into an on-average constant output stream while conserving information.

Second, we show that increasing network resources such as link bandwidth and buffer capacity results in a superlinear improvement in performance. When large file transfers occur with nonnegligible probability, the incremental improvement in throughput achieved for large buffer sizes is accompanied by long queueing delays vis-à-vis the case when the file size distribution is not heavy-tailed. Buffer utilization continues to remain at a high level implying that further improvement in throughput is only achieved at the expense of a disproportionate increase in queueing delay. A similar trade-off relationship exists between queueing delay and packet loss rate, the curvature of the performance curve being highly sensitive to the degree of self-similarity.

Third, we investigate the effect of congestion control on network performance when subject to highly self-similar traffic conditions. We implement an open-loop congestion control using unreliable transport on top of UDP where the data stream is throttled at the source to achieve a fixed arrival rate. Decreasing the arrival rate results in a decline in packet loss rate whereas link utilization increases. In the context of reliable communication, we compare the performance of three versions of TCP—Reno, Tahoe, and Vegas—and we find that sophistication of control leads to improved performance that is preserved even under highly self-similar traffic conditions. The performance gain from Tahoe to Reno is relatively minor while the performance jump from TCP Reno to Vegas is more pronounced consistent with quantitative results reported elsewhere.

Keywords: Self-similar traffic, network performance, congestion control and quality of service

1. INTRODUCTION

Recent measurements of traffic in local area networks have shown that network traffic is often bursty on a wide range of time scales, a much wider range than is captured by traditional traffic models.[2] Scale-invariant burstiness can be described using the notion of self-similarity, and much work has been done in the last few years in verifying its ubiquitous presence in networked environments and investigating the causes of traffic self-similarity and its consequences.[3–10,1,11,12]

Other author information. K. Park: Supported in part by NSF grant CCR-9204284; park@cs.purdue.edu.
G. Kim: Supported in part by NSF grant CCR-9308344; kgtjan@cs.bu.edu.
M. Crovella: Supported in part by NSF grant CCR-9501822; crovella@cs.bu.edu.

A number of papers have studied the implications of long-range dependence for traffic modeling and network performance evaluation. The research avenues may be broadly classified into two categories, one dealing primarily with traffic characterization and modeling issues, the other concentrating on performance evaluation. In the first category of work,[5–8,11,12] traffic traces from physical network measurements are used to identify the presence of scale-invariant burstiness, and models are constructed capable of generating synthetic traffic with matching characteristics. These works have shown that long-range dependence is a phenomenon occurring in both local area and wide area network traffic, and individual traffic sources such as compressed video have been shown to possess self-similarity as a source-intrinsic property.[6,7,13] In the second category are papers that have evaluated the effect of self-similar traffic in the context of idealized or simplified networks.[3,4,14,9,10] These papers show that long-range dependent traffic has an adverse effect on network performance, the principal result being that the distribution of queue lengths under self-similar traffic decays more slowly as compared to short-range-dependent sources (e.g., Poisson). In a somewhat different approach,[15,16] it is shown that for small buffer sizes or short time scales, long-range dependence has only a marginal influence. This is as expected since low frequency components—relative to buffer size and time scale—in the frequency domain act like "DC-components," and burstiness, which is an important factor in determining performance, is dominated by high-frequency components which collectively tend to exhibit short-range correlations.

The effects of self-similarity on the performance of resource-bounded networks incorporating the influence of a protocol stack have not been extensively studied. This question is practically relevant because, although self-similarity is expected to adversely affect network performance, knowledge of the degree of performance degradation due to self-similarity allows us to assess its relative importance as a performance problem and possibly control its consequences. The control aspects are studied in Park.[17] In this paper we investigate the effects of self-similar traffic on network performance when the functionality of the protocol stack and the interaction of traffic sources sharing bounded network resources is explicitly incorporated. We employ transport/network-layer simulations of reliable and unreliable transport protocols with or without congestion control to study the impact of self-similarity under controlled environments. Setting up such a test environment is, by itself, nontrivial due to the fact that, in general, the characteristics of the observed traffic is itself influenced by the actions of the control algorithms in the protocol stack.

Our results are facilitated by a high-level approach to self-similar traffic generation[1] that allows the degree of link traffic self-similarity to be intrinsically and intimately controlled by a mechanism acting at the application layer. In a nutshell, this process involves the generation of file or object transfers whose size distribution is heavy-tailed (formal definition given later). In other words, files or objects of extremely large sizes are communicated with nonnegligible probability among hosts in a generic network system. We have shown[1] that this high-level structural mechanism suffices to generate link traffic self-similarity, and the degree of the observed self-similarity is shown to be directly related to the heavy-tailedness of the file size distribution. An important advantage of this scheme is that self-similarity can be controlled at the application layer without directly feeding a self-similar time series from a host to the network anywhere in the system. This in turn allows us to evaluate the effects of the protocol stack—in particular, the transport layer—in a manner where its effects can be discerned. Simulation issues related to the evaluation of control protocols under self-similar traffic conditions are further discussed in Park.[18]

Our work is related to the ON/OFF model of Willinger at al.[12] where it is shown that the superposition of a large number of independent 0/1 renewal reward processes with heavy-tailed ON/OFF durations results in fractional Gaussian noise when suitably normalized. The main drawback of this simple, elegant characterization which goes back to some early work by Mandelbrot[19] is the *independence* assumption of the traffic streams which ignores their interaction and mutual dependence in real networks. The latter leads to a coupling of the traffic sources which stems from the sharing of bounded network resources. This paper extends our previous work[1] and complements the findings of Willinger et al.[12] by taking into account the impact of network resources and the influence exerted by transport protocols. Our main findings can be summarized as follows.

First, we show that transport layer mechanisms—congestion control and reliable communication—are crucial components in translating heavy-tailedness of file size distribution into link traffic self-similarity. Network performance as captured by throughput, packet loss rate, and packet retransmission rate degrades gradually with increasing heavy-tailedness while queueing delay, response time, and fairness deteriorate more drastically. The degree to which heavy-tailedness affects self-similarity is determined by how well congestion control is able to shape a source traffic into an on-average constant output stream while conserving information.

Second, we show that increasing network resources such as link bandwidth and buffer space results in a super-linear improvement in performance. When large file transfers occur with nonnegligible probability, the incremental improvement in throughput achieved for large buffer sizes is accompanied by long queueing delays vis-à-vis the case when the file size distribution is not heavy-tailed. Buffer utilization continues to remain at a high level implying that further improvement in throughput is only achieved at the expense of a disproportionate increase in queueing delay. A similar trade-off relationship exists between queueing delay and packet loss rate, the curvature of the performance curve being highly sensitive to the degree of self-similarity. This implies that multi-media applications—a traffic source with voluminous data rate requirements which promotes a structural environment conducive to network traffic self-similarity—are also the traffic sources that suffer most from their sensitive quality of service (QoS) requirements unless resource reservations or stratification of best-effort services is employed.[20] Increasing link bandwidth, given a fixed large buffer capacity, has the effect of decreasing queueing delay much more drastically under highly self-similar traffic conditions than when traffic is less self-similar. This suggests that high-bandwidth communication links be employed to alleviate the exponential trade-off relationship between queueing delay and packet loss/throughput for supporting QoS-sensitive traffic. The effect of bandwidth on queueing delay is much less pronounced when buffer capacity is small. Increasing link bandwidth also has the auxiliary effect of compressing the time axis of the self-similar traffic time series and amplifying its magnitude, making it easier for long-range dependence to be detected and exploited at time scales where transport protocol control decisions are affected.

Third, we show that congestion control is effective in the sense that it leads to a graceful decline in performance when subject to highly self-similar traffic conditions. We implement an open-loop flow control performing unreliable transport using UDP where the data stream is throttled at the source to achieve a fixed maximum arrival rate. Decreasing the arrival rate results in a decline in packet loss rate and an increase in link utilization. It also has the effect of making the traffic source conform more closely to the idealized assumptions of the ON/OFF model, in particular, as reflected by the independence of traffic sources when contention for network resources becomes negligible. In the context of reliable communication, we compare the performance of three versions of TCP—Reno, Tahoe, and Vegas—and we find that sophistication of control leads to improved performance that is preserved even under highly self-similar traffic conditions. The performance gain from Tahoe to Reno is relatively minor while the performance jump from TCP Reno to Vegas is more pronounced consistent with quantitative measurements reported elsewhere.[21,22]

The rest of the paper is organized as follows. In the next section, we describe the network model including the set-up of our simulation environment. This is followed by a description of the self-similar traffic generation process and dual separation of the protocol stack which are central to the understanding of the subsequent performance results. The next three sections evaluate network performance, first, with respect to self-similarity, second, as a function network resources, and lastly through the use of different congestion control algorithms. We conclude with a discussion of our results and ongoing work.

2. NETWORK MODEL AND SIMULATION SET-UP

2.1. Network Model

The network is given by a directed graph $G = (V, E)$ consisting of n nodes and m links. Each output link has a buffer, link bandwidth, and latency associated with it. $v_i \in V$ is a *server node* if it has a probability density function $p_i(X)$ where $X \geq 0$ is a random variable denoting file (or message) 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 node) if it has two probability density functions $h_i(X)$, $d_i(Y)$, $X \in \{1, \ldots, n\}$, $Y \in \mathbf{R}_+$, 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 the 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 is scheduled at time $T_k + Y$ where Y has distribution d_i. In unreliable (UDP) communication, this causal requirement is waived. A 2-server, 32-client network configuration with a bottleneck link between gateways G_1 and G_2 is shown in Figure 1.

A file is completely determined by its size X. Each file is split into $q = \lceil X/M \rceil$ packets where M is the maximum segment size. The segments are routed through a packet-switched internetwork in which packets are dropped at bottleneck nodes in case of buffer overflow. Each client alternates between independently placing file transfer requests to servers, receiving data from the server, and spending some period of time idle. $X_i \sim p_i(X)$, $Y_i \sim d_i(X)$, $i = 1, \ldots, n$, are i.i.d., and we require that $p_i(X)$ be heavy-tailed (formal definition given later). The idle time distribution, on the other hand, is allowed to be non-heavy-tailed (e.g., exponential).

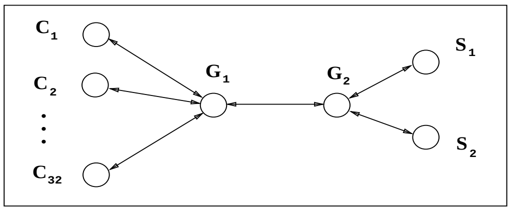

Figure 1. Network configuration.

2.2. Simulation Set-up

We use the LBNL Network Simulator, ns, as the basis for our simulation environment.[23] ns is an event-driven simulator derived from Keshav's REAL network simulator supporting several flavours of TCP (TCP Tahoe and TCP Reno)[24] and router scheduling algorithms. Although not production TCP code, we have found ns's emulation of TCP satisfactory for the purposes of studying the effect of congestion control[24,25] as well as emulating reliable transport. A test suite description can be found in Floyd.[23]

We have modified ns in order 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 the client/server nodes as separate application layer agents. A UDP-based unreliable transport protocol was added to the existing protocol suite, and an open-loop end-to-end flow control was implemented on top of it. The arrival rate was adjusted by a parameter which determined the spacing between successive packets. We also implemented TCP Vegas[21] as a module of ns and it was used in performance comparisons between the three versions of TCP under self-similar traffic conditions*. 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 at 10 millisecond granularity. The result in each case was a time series of one million data points, which ensured that our statistical measurements of self-similarity were based on time series of adequate length. Although most of the runs reported here were done with a 2-server/32-client bottleneck configuration (cf. Figure 1), other configurations were tested including performance runs with the number of clients increasing up to 132. The bottleneck link was varied from 1.5Mbps up to OC-3 levels, and buffer sizes were varied in the range 1kB–128kB. Non-bottleneck links were set at 10Mbps and the latency of each link was set to 15ms. The maximum segment size was fixed at 1kB for most runs. For any reasonable assignment to 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 d_i, appropriately, any intended level of contention could be achieved. This enabled us to carry out performance evaluations by varying the two main network resources—bottleneck buffer capacity and bandwidth—and the degree of self-similarity.

3. APPLICATION LAYER CAUSALITY OF SELF-SIMILAR LINK TRAFFIC

3.1. Dual Separation of Protocol Stack

One of the main goals of this paper is to evaluate the role of the protocol stack, in particular, the transport layer in determining network performance when subject to self-similar traffic conditions. To this end, we take a dual view of the protocol stack by partitioning its functionality into an application layer and a transport/network layer, collapsing the layers beneath the network layer into a single data link abstraction, simply referred to as the link layer or link. We also ignore finer subdivisions of the layers above the transport layer modeling it as a single entity. The application layer is completely characterized by the probability distributions mentioned in Section 2.2. File size distribution is the main control variable with idle time distribution playing a secondary role. In Park et al.,[1] we have shown that independent of idle time distribution, file size distribution alone suffices to induce self-similar link traffic by making the distribution more or less heavy-tailed. We use the Pareto distribution (definition given in Section 3.2) with shape parameter α for this purpose adjusting the location parameter k to keep the average file size—sampled over

*The TCP Vegas module in ns was implemented by Alia Atlas.

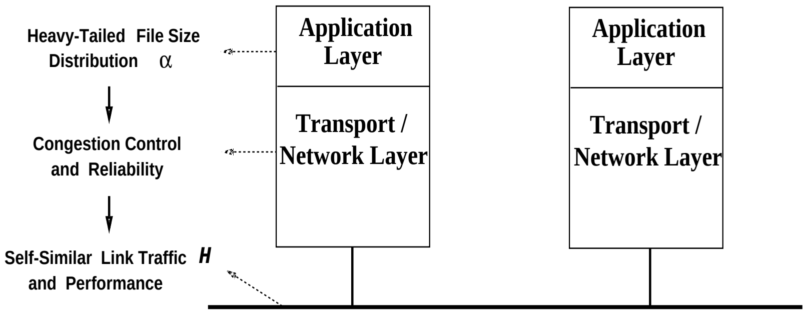

Figure 2. Transformation of the heavy-tailedness of file size distribution property at the application layer via the action of the transport/network layer to its manifestation as self-similar link traffic at the link layer.

10000 seconds—invariant across the α values 1.05, 1.35, 1.65, 1.95 considered. We have used some other distributions to generate communication demand at the application layer, and in the case of the exponential distribution which possesses a short tail, performance results were comparable to that of the Pareto distribution with $\alpha = 1.95$. These results are omitted in the paper.

The transport/network layer is composed of a number of protocols implementing reliable/unreliable communication with congestion control, and packets are routed by the network layer across a packet-switched internetwork. The most important aspect of the dual protocol stack separation is the positioning of the "causal seed" of self-similarity at its highest point of affection—the application layer. File transfer requests are generated at the top without the imposition of an external self-similar time series anywhere in the system. The self-similar traffic observed at the link layer is an *induced* phenomenon, partly influenced by transport layer actions—i.e., traffic shaping effect—allowing for an intrinsic evaluation of the influence and impact of transport protocols on network performance.

How well the transport layer performs its function is dependent upon the availability of network resources, and we model network resources via two variables, buffer capacity and link bandwidth. Buffer capacity comes into play when multiple traffic streams are multiplexed onto the same output port of a gateway, its size determining packet loss rate as well as queueing delay. Our performance results are expressed as functions of bottleneck buffer capacity, link bandwidth, and the heavy-tailedness of file size distribution represented by α in the application layer. The characteristics of network traffic at the link layer are captured by the Hurst parameter H (defined later) which measures the degree of self-similarity of a time series. The overall structure of the set-up is depicted in Figure 2.

3.2. Heavy-Tailedness of File Size Distribution and Link Traffic Self-similarity

An important characteristic of the aforementioned mechanism for inducing self-similar link traffic from the application layer is the assumption that the sizes of files or messages being transferred are drawn from a heavy-tailed distribution. A distribution is *heavy-tailed* if it asymptotically follows a power law. That is,

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

where $0 < \alpha < 2$. One of the simplest heavy-tailed distributions is the *Pareto* distribution whose probability density function is given by

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

where α is the shape parameter, $k > 0$ is the location parameter, and $x \geq k$. The distribution function has the form

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

Heavy-tailed distributions have a number of properties that are qualitatively different from distributions more commonly encountered in networking research, in particular, the exponential distribution. If $\alpha \leq 2$, the distribution has infinite variance, and if $\alpha \leq 1$, the distribution has also infinite mean. Thus, as α decreases, a large portion of the probability mass resides in the tail of the distribution. In practical terms—and relating to our network model—a random variable that follows a heavy-tailed distribution can give rise to extremely large file size requests with nonnegligible probability.

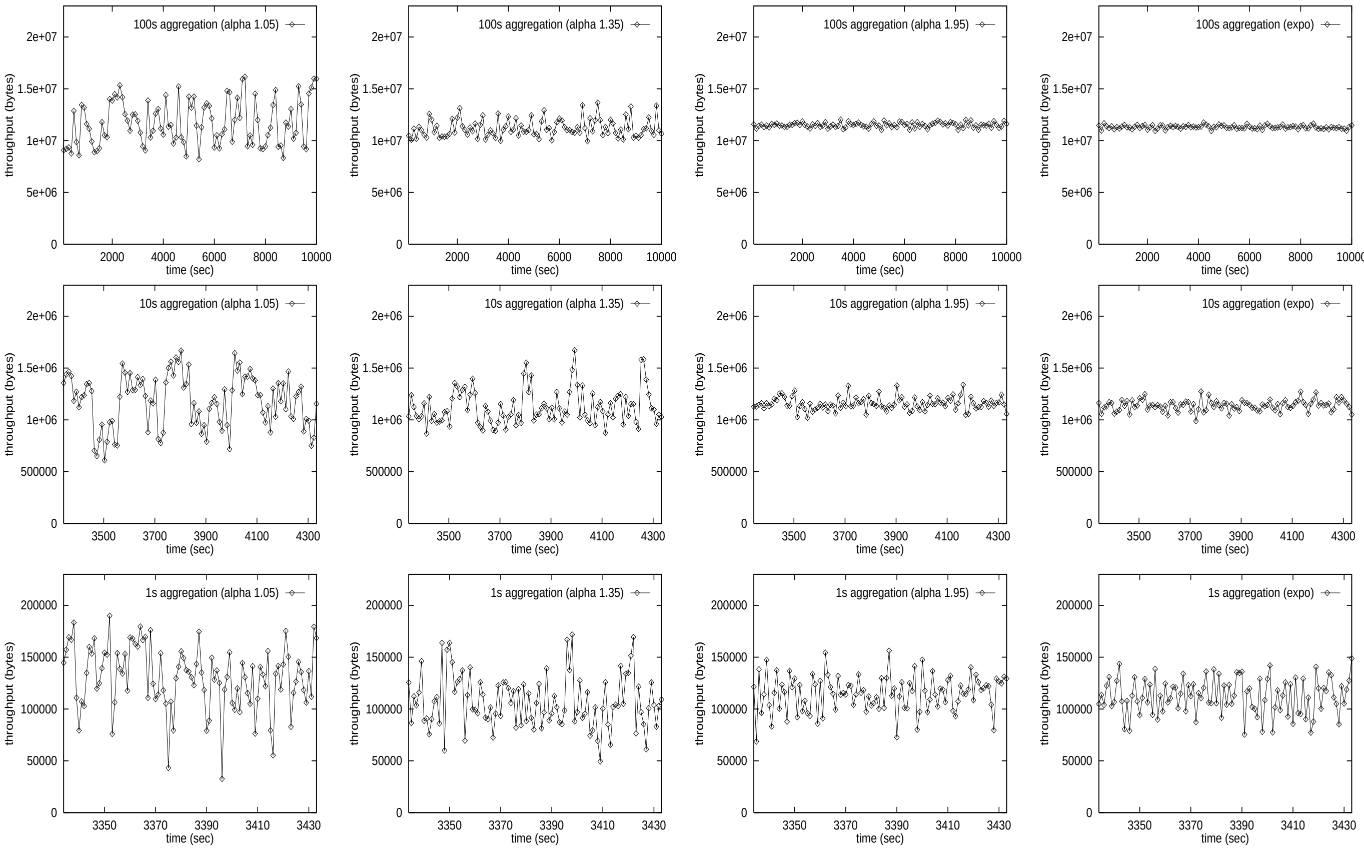

Figure 3. TCP run. Throughput as a function of file size distribution and three aggregation levels. File size distributions constitute Pareto with $\alpha = 1.05, 1.35, 1.95$, and exponential.

Figure 3 shows visually† that our set-up is able to induce self-similar link traffic, the degree of scale-invariant burstiness being determined by the α parameter of the Pareto distribution. The plots show the time series of network traffic measured at the output port of a bottleneck link (i.e., from gateway G_2 to G_1 in Figure 1). The *downstream traffic* is measured in bytes per unit time where the aggregation level or time unit varies over five orders of magnitude from 10ms, 100ms, 1sec, 10sec, to 100sec. Only the top three aggregation levels are shown in Figure 3. For α close to 2, we observe a smoothing effect as the aggregation level is increased, indicating a weak dependency structure in the underlying time series. As α approaches 1, however, burstiness is preserved even at large time scales indicating that the 10ms time series possesses long-range dependent. The first two aggregation levels (10ms, 100ms) are omitted because the difference between the aggregated time series manifests itself visually from the 1 second time scale onwards. The last column depicts time series obtained by employing an exponential file size distribution at the application layer with the mean normalized so as to equal that of the Pareto distributions. We observe that the aggregated time series between exponential and Pareto with $\alpha = 1.95$ are statistically indistinguishable.

Although Figure 3 gives clear visual evidence of the difference in self-similarity of link traffic as a function of our control variable α, a quantitative measure of self-similarity is obtained by using the *Hurst* parameter H which expresses the speed of decay of a time series' autocorrelation function. A time series with long-range dependence has an autocorrelation function of the form

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

where $0 < \beta < 1$. Hence, when compared to the exponential decay exhibited by traditional traffic models, the autocorrelation function of a long-range dependent process decays according to a power-law. The Hurst parameter

†Self-similar phenomena are well-suited to visual illustration facilitating an intuitive interpretation. This augments the statistical estimations of self-similarity given later in the paper.

is related to β via $H = 1 - \beta/2$. Hence, for self-similar time series, $1/2 < H < 1$. As $H \to 1$, the degree of self-similarity increases. A test for self-similarity of a time series can be reduced to the question of determining whether H significantly deviates from 1/2.

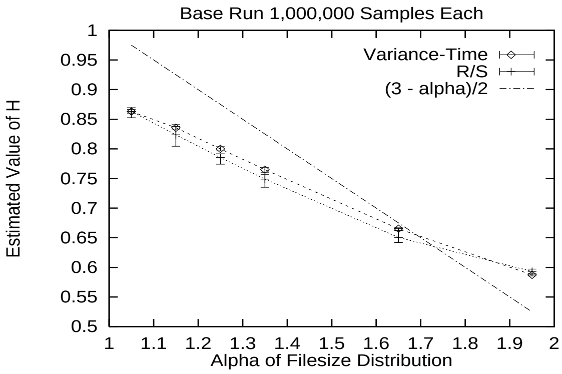

Figure 4. Hurst parameter estimates (R/S and V-T) for α varying from 1.05 to 1.95.

Figure 4 shows H-estimates based on Variance-Time (V-T) and R/S methods‡ for a baseline network configuration consisting of TCP Reno, link speed of 1.5Mbps, 6kB buffer size, and 2 server/32 clients. The 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. This particular configuration results in a packet drop rate of $\approx 4\%$ for the most bursty case ($\alpha = 1.05$). The plot shows that even when resources are bounded leading to interaction among the 32 traffic streams contending for network resources, the Hurst parameter estimates vary with file size distribution in a roughly linear manner. That is, the heavy-tailed property of the file size distribution as captured by α directly determines link-level traffic self-similarity of the downstream traffic as measured by H. The $H = (3-\alpha)/2$ relation 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 when the traffic sources are independent with constant ON/OFF amplitudes. Although their overall trends are similar (nearly coinciding at $\alpha = 1.65$), 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 α close to 1 and above the line for α close to 2.

4. PERFORMANCE EVALUATION

4.1. Effect of Self-similarity

In this section, we evaluate network performance when the degree of self-similarity is varied. The scale-invariant burstiness of link traffic is affected solely by the heavy-tailedness of file size distribution at the application layer (cf. Figure 2 in Section 3), captured by the shape parameter α of the Pareto distribution which acts as the control variable. Other candidate file size distributions possessing short tails (e.g., exponential) yield similar performance results as the Pareto distribution with $\alpha = 1.95$ and are not shown here. First, we present performance data when the transport layer implements reliable communication with congestion control. We use TCP Reno as our principal benchmark protocol. Other versions of TCP yield similar results and their performance differences are studied in Section 4.3. We also present performance results evaluating the effect of self-similarity when unreliable communication is employed at the transport layer. We use UDP-based transmission for this purpose, with a greedy transport agent driving the UDP module at maximum rate subject to bandwidth limitation and a nominal slow-down proportional to the file size which accounts for processing overhead.

4.1.1. Packet loss and retransmission

Figure 5 shows the effect of varying the heavy-tailedness of the Pareto distribution§ for $\alpha = 1.05, 1.35, 1.65$, and 1.95. The left and middle figures show packet loss rate and packet retransmission rate as a function of α for four values of

‡Variance-time (V-T) and R/S are frequently used methods[26,8] for estimating the Hurst parameter H.

§We remark that the location parameter of the Pareto distribution was adjusted so that the actual mean of the file size requests sampled over 10000 seconds remained invariant at 4.1kB for the range of α values considered.

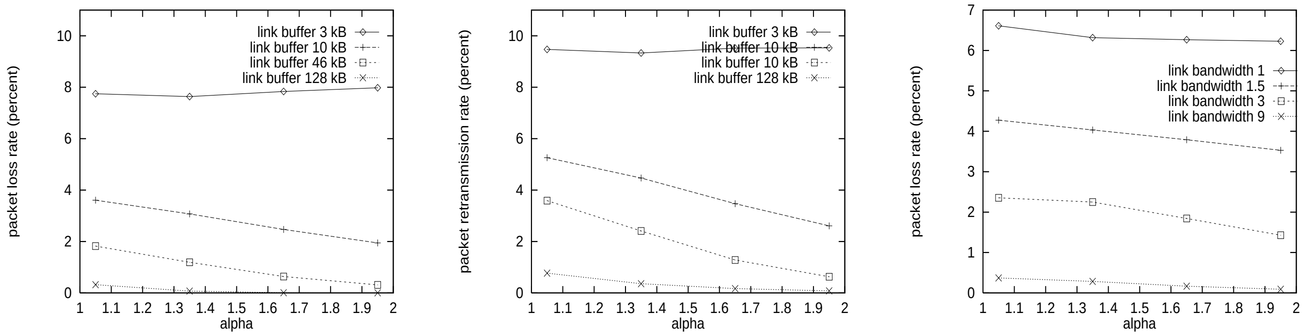

Figure 5. TCP run. Packet loss rate (left) and packet retransmission rate (middle) as a function of α for four values of buffer capacity. Packet loss rate (right) as a function of α for four values of link bandwidths.

buffer capacities, 3kB, 10kB, 46kB, and 128kB. When the bottleneck buffer size is very small (2kB), there is little difference in performance across the α values, exhibiting a flat performance curve. This is due to the burstiness at $\alpha = 1.95$ being already sufficient to induce high packet loss. Since TCP employs feedback control triggering back-off upon perceived packet loss, the potential for an amplified effect at smaller α values is muted by the *initial* packet drops occurring, on average, across all α values. The amplification effect is realized and clearly visible when feedback congestion control is not used as in the case of the non-flow-controlled UDP-based protocol (cf. Figure 7 below). When buffer size is large (128kB), the increased capacity allows even the high burstiness at $\alpha = 1.05$ to be accommodated, and we see a gradual deterioration in packet loss and retransmission as α approaches 1.

The slope of the performance curve is most pronounced for intermediate values of the bottleneck buffer capacity (10kB, 46kB) at which the difference in burstiness between $\alpha = 1.05$ and $\alpha = 1.95$ takes its highest toll. However, we note that packet loss and retransmission remain a *linear* function of α and this "robustness" with respect to self-similarity is achieved by the action of congestion control. We will see in Section 4.1.3 that in its absence, a nonlinear degradation in performance results as α approaches 1. Figure 5 (right) shows a similar behavior with respect to packet loss rate when the bottleneck bandwidth is varied (1Mb/s, 1.5Mb/s, 3Mb/s, 9Mb/s) for a fixed buffer size of 6kB.

4.1.2. Throughput and queueing delay

Figure 6 (left) shows reliable throughput as a function of α for four values of buffer capacity. *Reliable throughput* ϱ is defined as the average over all file transfers (indexed by k) over time of the ratio

$$\varrho_k = \frac{S_k}{W_k} = \frac{S_k}{S_k + H_k}$$

where S_k is the k'th file size, W_k is the total number of bytes expended to reliably transmit the k'th file, and $H_k = W_k - S_k$ is the net overhead incurred including packet retransmissions. Hence, the fewer the lost packets and

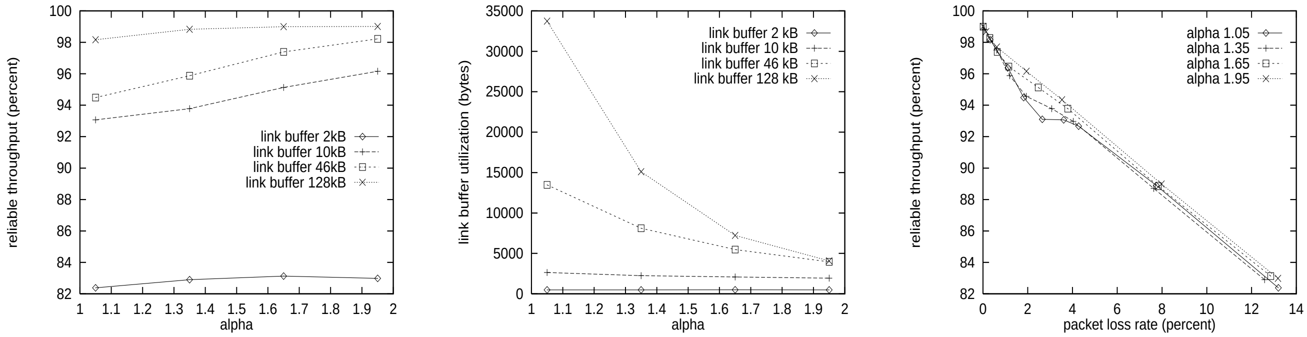

Figure 6. TCP run. Reliable throughput (left), mean queue length (middle) as a function of α; reliable throughput as a function of packet loss rate (right).

resulting retransmissions, the closer ϱ is to 1. In the case of reliable communication with TCP's window control, we use reliable throughput as a measure of "effective throughput" to better account for retransmission overhead and distinguish it from raw throughput. As with packet loss rate, Figure 6 (left) shows performance curves that are most sensitive with respect to self-similarity for intermediate values of buffer capacity. This is as expected since throughput and packet loss rate are almost linearly correlated as seen in Figure 6 (right).

Figure 6 (middle) shows mean queue length at the bottleneck link output buffer as a function of α for four values of buffer capacity, 2kB, 10kB, 46kB, and 128kB. For small values of buffer size, even at $\alpha = 1.95$ the buffer overflow is high, and because of TCP's congestion control (see the discussion above), buffer utilization remains invariant across all α values. When buffer capacity is large (128kB), the highly self-similar traffic at $\alpha = 1.05$ is able to utilize the increased queueing capacity to achieve further improvement in throughput. This occurs, however, at a disproportionate—i.e., exponential—increase in queueing delay relative to the change in α. We will see in Section 4.2 that this nonlinear relationship also manifests itself as a sharp increase in the curvature of the delay-throughput curve.

4.1.3. Unreliable non-flow-controlled transport

Figure 7 shows packet loss rate, link utilization, and mean queue length as a function of α for four values of buffer capacity when an unreliable UDP-based non-flow-controlled transport protocol is used. An extension of the unreliable non-flow-controlled protocol to a flow controlled version implementing a simple source-throttled open-loop control is discussed in Section 4.3. The present scheme serves to show the influence of the protocol stack in determining network performance as a function of self-similarity at one extreme end of the spectrum of possible transport layer controls.

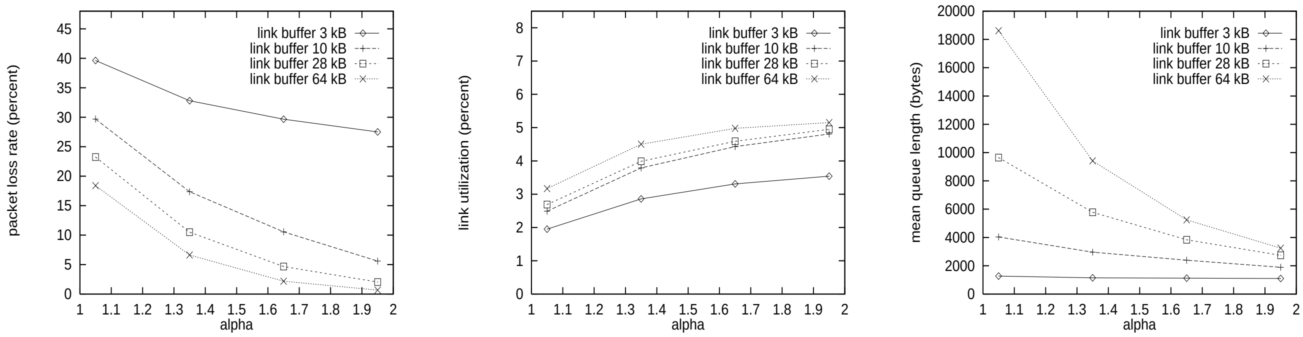

Figure 7. UDP run. Packet loss rate (left), link utilization (middle), and mean queue length as a function of α.

In contrast to the gradual increase in packet loss achieved when TCP is employed (see Figure 5), we observe a *superlinear* degradation in the packet loss rate as α approaches 1. Throughput as captured by link utilization improves only *sublinearly* as a function of α. Thus, TCP's congestion control and reliability mechanism have the effect of pulling down the superlinear dependence of network performance on self-similarity to a more robust linear relationship with respect to packet loss and throughput. Figure 7 (right) shows that mean queue length continues to depend superlinearly on α for large buffer sizes. It is not surprising that UDP-based unreliable transport without additional control performs poorly under various network conditions. However, the fact that TCP's congestion control and reliability mechanism are able to achieve a *linear* dependence on the degree of self-similarity is *empirical* and we will study this feature, especially its high cost with respect to increased queueing delay, in the next section.

4.2. Effect of Network Resources

The previous section has shown that self-similarity as captured by the shape parameter α of the Pareto distribution acting at the application layer, exerts a direct influence on network performance through its manifestation as self-similar traffic at the link layer. How and with what magnitude it affects performance is modulated by the protocols acting at the transport/network layer. In this section, we study the effect of network resources on performance, especially with respect to the trade-off relationship between throughput and delay as well as packet loss and delay, all under varying self-similar traffic conditions.

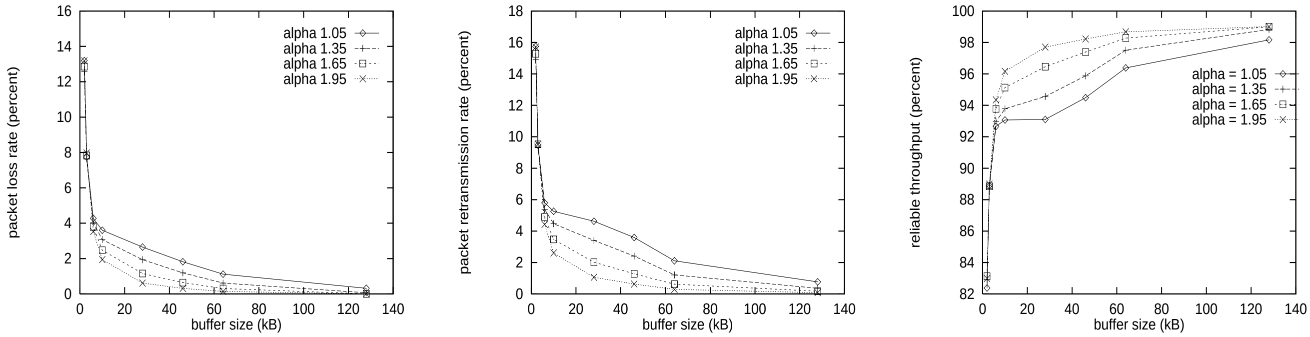

Figure 8. TCP run. Packet loss rate (left) and packet retransmission rate (middle) as a function of buffer capacity. Reliable throughput (right) as a function of buffer capacity.

4.2.1. Buffer capacity

Figure 8 shows the effect of varying the bottleneck link buffer capacity for buffer sizes in the range 2kB–128kB on packet loss rate, packet retransmission rate, and reliable throughput using TCP Reno. In Figure 8 (left), we see a sharp threshold behavior of packet loss rate as a function of buffer size, "partitioning" buffer capacity into a highly sensitive and a saturated domain. This holds, modulo a shift effect, for all the α values considered. A similar observation holds for packet retransmission rate and reliable throughput as the graphs in Figure 8 (middle) and (right) indicate. However, the dependence on buffer size for small values of α is less smooth and exhibits stretches of abruptness.

Whereas for throughput and packet loss rate the improvement in performance for large buffer sizes was gradual, Figure 9 (left) shows that the qualitative dependence of mean queue length for large buffer sizes is determined by the degree of self-similarity. For α close to 2, the functional dependence of mean queue length on buffer capacity is clearly sublinear (roughly logarithmic) whereas for α close to 1, the dependence becomes linear. For very large values of the buffer size we expect the delay curve to saturate even when α is close to 1 although the practical significance of the difference in the magnitude of their queueing delay will be preserved. The proportional increase in buffer occupancy for α close to 1 indicates that, even for large buffer sizes, traffic burstiness is high enough to cause constant utilization.

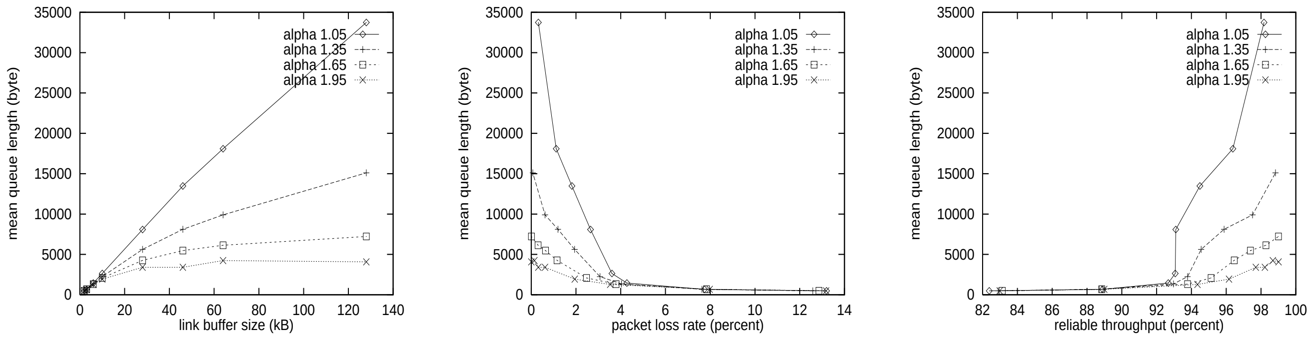

Figure 9. TCP run. Mean queue length as a function of buffer capacity for various values of α (left), mean queue length vs. packet loss rate trade-off relation (middle), and mean queue length vs. reliable throughput trade-off relation (right).

The large difference in the growth of queueing delay vis-à-vis reliable throughput and packet loss rate as a function of buffer capacity manifests itself in a trade-off relationship which is shown in Figures 9 (middle) and (right). Figure 9 (middle) shows the delay-packet loss curve for four values of self-similarity, $\alpha = 1.05, 1.35, 1.65$, and 1.95. For α close to 1, we observe an exponential trade-off relationship between queueing delay and packet loss rate, with significant differences in the curvature among the four performance curves. In the context of facilitating multi-media traffic such as video and voice in a best-effort manner while satisficing their diverse quality of service

(QoS) requirements, the performance curves in Figure 9 (middle) show that low packet loss, on average, can only be achieved at a significant increase in queueing delay and vice versa. Since video and voice traffic tend to be large-volume transfers whose increased presence causes the intrinsic self-similarity of the overall system to be amplified, it is the QoS-sensitive traffic that both promotes scale-invariant burstiness as well as suffers most from its consequences via the sensitive dependence of the delay-packet loss trade-off relation on the degree of self-similarity in the system. Figure 9 (right) shows the delay-throughput curve which exhibits a trade-off relationship similar to queueing delay vs. packet loss rate, its curvature increasing as α approaches 1.

4.2.2. Bottleneck link bandwidth

Figure 10 (left) shows the effect of varying bottleneck link bandwidth on packet loss rate for four values of α. As in the buffer capacity case, we see a threshold effect with packet loss rate being most sensitive for small values of link bandwidth followed by a gradual decline in packet drop as bandwidth is increased. A similar situation holds for packet retransmission rate which is omitted here. Figure 10 (middle) shows the effect of increasing bandwidth on queueing delay for $\alpha = 1.05, 1.35, 1.65, 1.95$ when buffer capacity is fixed at 128kB. At $\alpha = 1.95$, we see a saturation effect as bandwidth is increased from 1.5Mbps to 9Mbps producing successively smaller improvements in mean queue length which indicates a diminishing return of the marginal utility of link bandwidth with respect to queueing delay. As α approaches 1, however, the situation reverses such that at $\alpha = 1.05$, the opposite is true. That is, as bandwidth is increased, we see an accelerated decline in mean queue length vis-à-vis the case when α is large. Figure 10 (right) shows performance results for the same set-up except that the bottleneck buffer size was fixed at the smaller value of 6kB. Unlike before, we now observe a constant slope in the delay curve as a function of α across all the bandwidth values considered, and it does not exhibit a saturation effect nor acceleration effect at either end of α. This implies that when the bottleneck buffer capacity is small, then increasing bandwidth carries only a gradual improvement in queueing delay independent of whether traffic is highly self-similar or not.

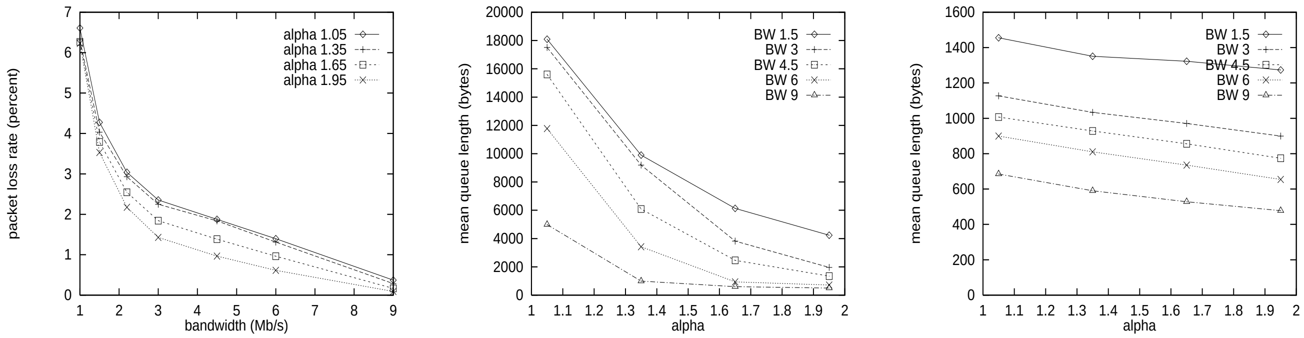

Figure 10. TCP run. Packet loss rate as a function of link bandwidth for four values of α (left), mean queue length as a function of α for five values of bandwidths 1.5Mbps, 3Mbps, 4.5Mbps, 6Mbps, 9Mbps for a fixed buffer capacity of 128kB (middle), and for fixed buffer capacity 6kB (right).

In the context of network design, the previous observations imply that when the overall system is prone to high levels of long-range dependence as in the case of multi-media traffic, increasing buffer capacity alone is a bad idea since it carries a significant queueing delay penalty while improving packet loss only gradually. A bandwidth-driven network resource allocation policy, assuming a fixed overall traffic demand, is a more effective approach since it improves both throughput and queueing delay gradually, while potentially yielding a more significant improvement in queueing delay if buffer capacity is large.

Figure 11 shows an auxiliary effect of increasing link bandwidth, namely, compression of the time axis of the self-similar link traffic time series as well as amplification of its amplitude. The variance/mean ratio of link traffic is related to the index of dispersion count and measures the variance of the aggregated time series at five aggregation levels—10ms, 100ms, 1sec, 10sec, 100sec—against the mean. The larger this ratio for α small vis-à-vis when α is large, the more “bursty” the underlying time series and the easier it is to detect statistical differences between highly self-similar and less-so traffic. Moreover, if the difference is preserved for fine levels of aggregation, then the easier it is observe and exploit the dependency structure in $\alpha \approx 1$ traffic.

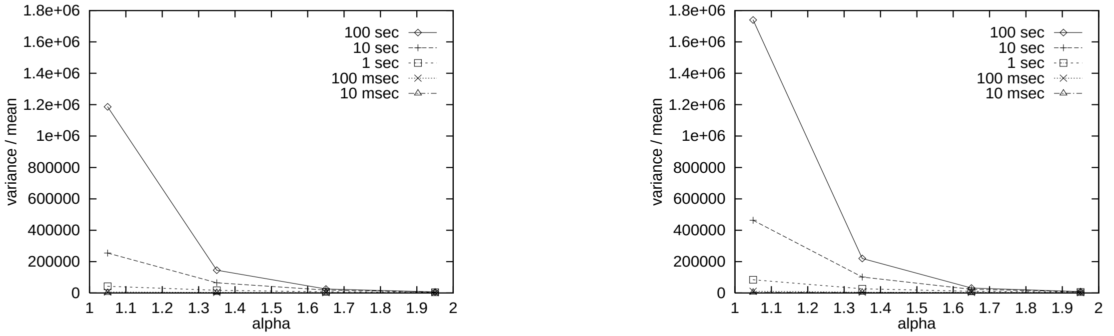

Figure 11. TCP run. Variance/mean ratio for link throughput as a function of time-scale and α when bottleneck bandwidth is 3Mb/s (left) and when bottleneck bandwidth is 6Mb/s (right).

4.2.3. Unreliable non-flow controlled transport

The sensitivity results and trade-off relationships of Section 4.2.1 are confirmed in the case of the non-flow-controlled UDP-based unreliable transport protocol discussed earlier. Packet loss rate and mean queue length, as a function of buffer capacity, exhibit qualitative shapes similar to their corresponding graphs for flow-controlled reliable transport using TCP Reno in Section 4.2.1. In fact, they have an even more sensitive dependence on α. The plots are omitted here for brevity.

Figure 12 depicts the delay-packet loss and delay-throughput trade-off relationships showing a sharp increase in the curvature of the performance graphs as a function of self-similarity. The delay-link utilization curve, although amplified in shift, is smoother than its corresponding counter-part for flow-controlled reliable communication—the delay-reliable throughput curve. This suggests that the jaggedness in Figure 9 was caused by TCP's reliability and congestion control mechanism.

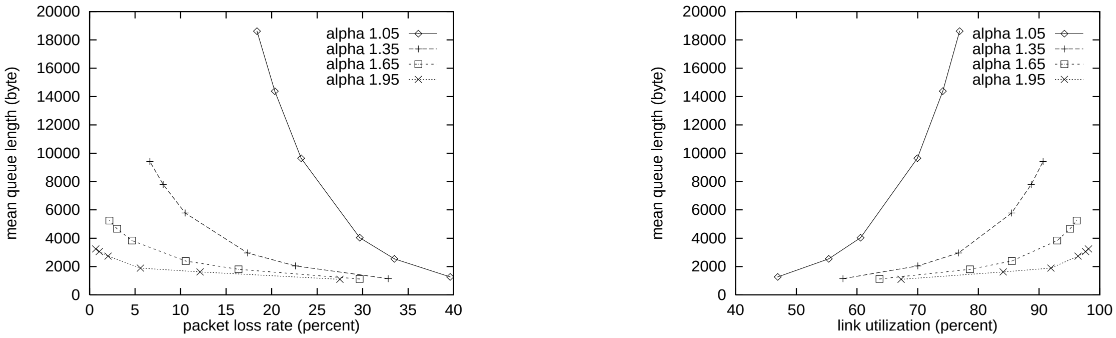

Figure 12. UDP run. Mean queue length vs. packet loss rate (left) and mean queue length vs. link utilization (right).

4.3. Effectiveness of Congestion Control

The previous sections have shown performance results as a function of self-similarity and network resources, and the impact of the transport layer in affecting performance. In this section, we compare the performance results across several congestion control algorithms, first, in the case of flow-controlled unreliable transport using UDP, and second, by comparing performance differences across three versions of TCP—Reno, Tahoe, and Vegas.

4.3.1. Flow-controlled unreliable transport

We employ a simple open-loop flow-controlled unreliable transport protocol based on UDP to test the effect of stretching out a file transfer session into an output stream of "on-average" constant height. The arrival rate is throttled by a rate parameter τ which determines the minimum (time) spacing between successive packets. Thus, by suitably adjusting the inter-packet spacing parameter τ, the maximum arrival rate of a traffic source is controlled.

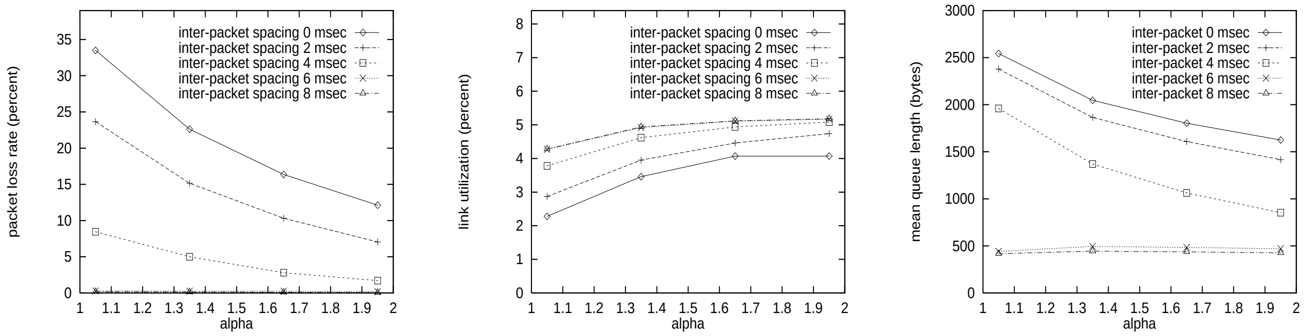

Figure 13. Flow-controlled UDP for four values of interpacket spacings: Packet loss rate (left), link utilization (middle), and mean queue length (right) as a function of α.

As $\tau \to \infty$, the traffic source becomes more and more like the ON/OFF model of Willinger et al.[12]—except for the *finite* number of traffic sources—due to the decoupling achieved by making the consequences of resource contention negligible.

Figure 13 (left) shows packet loss rate as a function of α for five values of τ, $\tau =$ 0ms, 2ms, 4ms, 6ms, 8ms. For small τ, we observe the characteristic nonlinear increase in packet loss rate as $\alpha \to 1$. As τ is increased, however, packet loss rate flattens out until the curve becomes horizontal and packet loss approaches 0. Figure 13 (middle) shows the corresponding effect on link utilization. We observe that link utilization approaches saturation as τ is increased for all values of α. The rightmost plot shows the effect of τ on mean queue length which, after exhibiting a nonlinear dependence on α for small values of τ, flattens out as τ is increased.

Figure 14 shows the Hurst parameter estimates using the V-T and R/S methods. The curves exhibit an upward shift as the interpacket spacing parameter τ is increased. This is due to fewer packets being discarded because of buffer overflow, thus better preserving the tail of the file size distribution which manifests itself as large file transfer sessions at the transport layer.

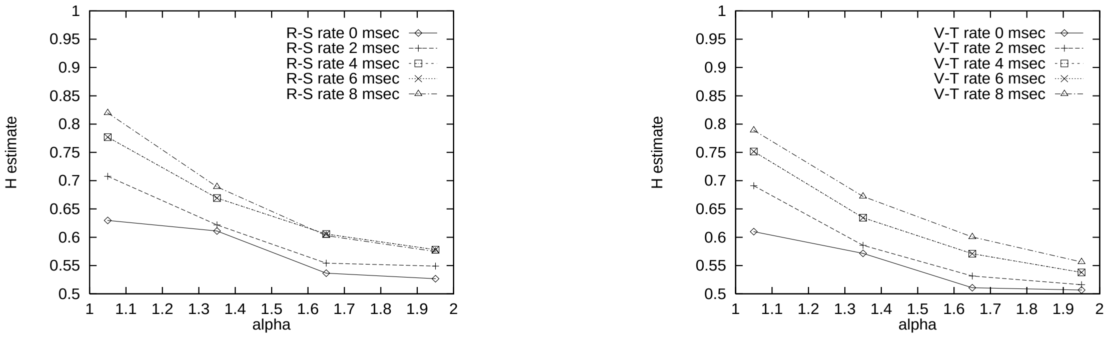

Figure 14. Flow-controlled UDP run. Hurst parameter estimates using Variance-Time (left) and R/S (right) for five values of interpacket spacings.

4.3.2. Performance comparison of TCP Reno, Tahoe, and Vegas

In this section, we present preliminary findings on the performance and effectiveness of three versions of TCP implementing progressively more sophisticated congestion control features when going from TCP Tahoe to Reno to the recently proposed TCP Vegas.[21] Figure 15 (left) compares the packet loss rate of Tahoe, Reno, and Vegas as a function of self-similarity under identical network conditions. Bottleneck buffer capacity was set to 6kB and link bandwidth was set at 1.5Mbps. All three protocols exhibit the characteristic linear dependence on α as seen in the more comprehensive study of TCP Reno under various buffer capacities and link bandwidths (Section 4.1). The gap in performance between Tahoe, Reno, and Vegas remains consistent, and its width stays "roughly" the same across all α values. That is, even under highly self-similar traffic conditions, the performance gap is preserved across the three congestion control algorithms. The performance gain from Tahoe to Reno is relatively minor while the performance jump from Reno to Vegas is more pronounced, its magnitude falling into the same ballpark range as the quantitative measurements reported elsewhere.[21,22]

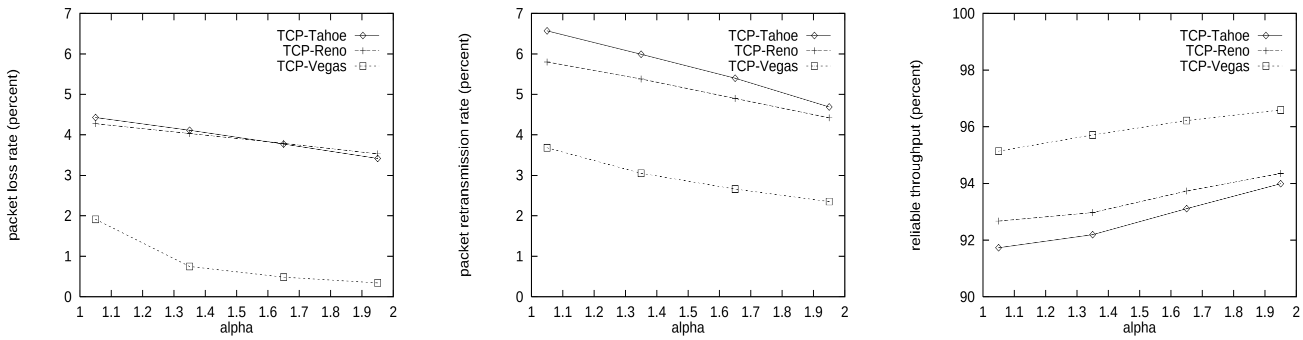

Figure 15. TCP Reno vs. Tahoe vs. Vegas run. Packet loss rate (left), packet retransmission rate (middle), and reliable throughput (right) as a function of α.

5. CONCLUSION

In this paper, we have taken an integrated look at network performance under self-similar traffic conditions by studying the interaction of three components spanning the protocol stack: heavy-tailed file size distribution in the application layer, congestion control and reliable message transmission at the transport layer, and network traffic observed at the link layer. We have examined network performance from three viewpoints, first, with respect to the impact of self-similarity, second, as a function of network resources and the resulting trade-off relationship of performance variables, and third, by comparing the relative performance of several congestion control schemes and their effectiveness.

We have shown that self-similar traffic can have serious adverse impact on network performance. While throughput declines gradually as self-similarity increases, queueing delay increases more drastically. When traffic is highly self-similar, we find that queueing delay grows nearly *proportionally* to the buffer capacity present in the system. This effect is consistent with the observation from queueing analysis that queue length distribution decays much more slowly for long-range dependent traffic. Taken together, these two observations have potentially dire implications for quality of service provision in networks. To achieve a constant level of throughput or packet loss as self-similarity is increased, extremely large buffer capacity is needed. However, increased buffering leads to large queueing delays and thus we find that the effect of self-similarity is to significantly steepen the tradeoff curve between throughput/packet loss and delay.

While these results indicate that QoS provision is a difficult problem in self-similar environments, our results also indicate some ways in which the effects of self-similarity may be mitigated. In particular, we have shown that one of the characteristics of highly self-similar traffic is that increasing network bandwidth can result in more pronounced improvement in packet delay than is the case when traffic is weakly self-similar. In fact, the reduction in packet delay from increased bandwidth is much larger for self-similar traffic than for non-self-similar traffic when the underlying buffer capacity is not too small. This suggests that rather than reducing buffer capacity to improve queueing delay, an alternative—and possibly more effective—strategy in self-similar environments is to place emphasis on bandwidth when architecting network systems.

There are a number of limitations to the present work. One, we have focused on the tails of the distributions of file sizes and we have concentrated on the Pareto distribution for simplicity. Distributions of file sizes observed in practice, although still heavy-tailed, are known to have a different shape for small values. While we do not believe this difference will significantly affect our results, we are examining this issue now. Two, our network topology is not representative of real networks, again adopted mainly for simplicity. This issue is also currently being explored.

REFERENCES

1. 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*, pp. 171–180, October 1996.

2. W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of ethernet traffic," in *Proc. ACM SIGCOMM '93*, pp. 183–193, 1993.
3. A. Adas and A. Mukherjee, "On resource management and QoS guarantees for long range dependent traffic," in *Proc. IEEE INFOCOM '95*, pp. 779–787, 1995.
4. R. Addie, M. Zukerman, and T. Neame, "Fractal traffic: measurements, modelling and performance evaluation," in *Proc. IEEE INFOCOM '95*, pp. 977–984, 1995.
5. 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.
6. M. Garret and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," in *Proc. ACM SIGCOMM '94*, pp. 269–280, 1994.
7. 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*, pp. 114–125, 1995.
8. W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking* **2**, pp. 1–15, 1994.
9. N. Likhanov and B. Tsybakov, "Analysis of an ATM buffer with self-similar ("fractal") input traffic," in *Proc. IEEE INFOCOM '95*, pp. 985–992, 1995.
10. I. Norros, "A storage model with self-similar input," *Queueing Systems* **16**, pp. 387–396, 1994.
11. V. Paxson and S. Floyd, "Wide-area traffic: the failure of Poisson modeling," in *Proc. ACM SIGCOMM '94*, pp. 257–268, 1994.
12. 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*, pp. 100–113, 1995.
13. D. Heyman and T. Lakshman, "What are the implications of long-range dependence for VBR-video traffic engineering?," *IEEE/ACM Transactions on Networking* **4**, pp. 301–317, June 1996.
14. P. Jacquet, "Queueing with aggregated up/down arrivals," 1996. Preprint.
15. M. Grossglauser and J.-C. Bolot, "On the relevance of long-range dependence in network traffic," in *Proc. ACM SIGCOMM '96*, pp. 15–24, 1996.
16. 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*, pp. 3–14, 1996.
17. K. Park, "Self-similar network traffic and its control," To appear in *Proc. 25th Allerton Conference on Communication, Control and Computing* , October 1997.
18. K. Park, "On the effect and control of self-similar network traffic: a simulation perspective," To appear in *Proc. 1997 Winter Simulation Conference* , December 1997.
19. B. B. Mandelbrot, "Long-run linearity, locally gaussian processes, h-spectra and infinite variances," *Intern. Econom. Rev.* **10**, pp. 82–113, 1969.
20. K. Park, M. Sitharam, and S. Chen, "Quality of service provision in noncooperative network environments," Tech. Rep. CSD-TR-97-039, Department of Computer Sciences, Purdue University, 1997.
21. L. Brakmo and L. Peterson, "TCP Vegas: end to end congestion avoidance on a global internet," *IEEE J. Select. Areas Commun.* **13**(8), pp. 1465–1480, 1995.
22. J. Ahn, P. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: Emulation and experiment," in *Proc. ACM SIGCOMM '95*, pp. 185–195, 1995.
23. S. 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.
24. V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM '88*, pp. 314–329, 1988.
25. K. Park, "Warp control: a dynamically stable congestion protocol and its analysis," *Journal of High Speed Networks* **2**(4), pp. 373–404, 1993.
26. J. Beran, *Statistics for Long-Memory Processes*, Monographs on Statistics and Applied Probability, Chapman and Hall, New York, NY, 1994.