How does TCP generate Pseudo-self-similarity?

LIANG GUO MARK CROVELLA IBRAHIM MATTA
Computer Science Department
Boston University
Boston, MA 02215
{guol, crovella, matta}@cs.bu.edu

Abstract—*Long-range dependence has been observed in many recent Internet traffic measurements. In addition, some recent studies have shown that under certain network conditions, TCP itself can produce traffic that exhibits dependence over limited timescales, even in the absence of higher-level variability. In this paper, we use a simple Markovian model to argue that when the loss rate is relatively high, TCP's adaptive congestion control mechanism indeed generates traffic with OFF periods exhibiting power-law shape over several timescales and thus introduces pseudo-long-range dependence into the overall traffic. Moreover, we observe that more variable initial retransmission timeout values for different packets introduces more variable packet inter-arrival times, which increases the burstiness of the overall traffic. We can thus explain why a single TCP connection can produce a time-series that can be misidentified as self-similar using standard tests.*

Keywords— **Congestion Control, Long-Range Dependence, Self-Similarity.**

I. INTRODUCTION

SELF-SIMILARITY has been observed in a large number of Internet traffic measurements (for example, [1, 2, 3]). Self-similarity refers to the condition in which the second-order statistics of traffic decay very slowly with increasing the aggregation level, compared to traditional Markovian (memory-less) models. Self-similarity is closely associated with the phenomenon of heavy-tailed distributions, which are distributions whose tails decline via a power law with small exponent (less than 2). The presence of heavy tails in lengths of individual flows can be shown to induce self-similarity in network traffic [4]. Heavy-tailed properties have been found in file sizes and user thinking time [1], flow (session) duration [5], as well as packet inter-arrival time [4] distributions in the Internet.

To date, the best-accepted hypothesis for the genesis of self-similarity on timescales from seconds to an hour is the heavy-tailed distribution that is typical of flow lengths in the Internet [1, 4]— the majority of flows are found to be very short, many are long, and some are very long. However, a number of studies have shown that interesting scaling properties can arise even when flow lengths are *not* highly variable [6, 7].

In particular, recent work by Veres *et al.* [7] shows that even without any variability in terms of flow lengths or network delays, TCP itself can sometimes exhibit "chaotic" behavior and produce traffic series that shows properties similar to those of self-similar traffic generated by synthetic methods. In this paper, we explain why this can happen, using a simple Markovian analysis. This explanation shows that such scaling behavior only exists over a finite range of timescales; thus, it should not be categorized as self-similarity in the underlying traffic. By means of *ns* simulation [8], we confirm that packet interarrival time from TCP flows appear to be drawn from a discrete power-law distribution over a limited range of timescales, under certain conditions. We show that larger loss rates bring longer-timescale dependence into the traffic. In addition, we show in this paper that, because short connections do not have enough packet samples to accurately estimate the time they need to wait for acknowledgments, the conservatively chosen value can introduce very large gaps between packets, thus making the traffic generated by (short) TCP connections even more bursty, or in other words, extending burstiness to larger timescales.

This work was supported in part by NSF grants ANI-9986397, CAREER ANI-0096045, and MRI EIA-9871022.

Heavy-tailed Distribution, Self-Similarity and Long-Range Dependence: Here we briefly review some concepts related to fractal traffic properties. A more detailed description can be found in [9, 10] and references therein.

Many of the distributions shown in today's Internet traffic have the property of being *heavy-tailed*. We say a distribution is heavy-tailed if the asymptotic shape of the distribution is power-law with exponent less than 2 regardless of the behavior of the distribution for small values of the random variable, i.e.,

$$P[X \geq x] \sim x^{-\alpha}, \quad \text{as } x \to \infty, 0 < \alpha < 2$$

The reason that such distributions are called heavy-tailed is that, compared to those more commonly used distributions such as exponential and normal distributions, a random variable that follows a heavy-tailed distribution can give rise to extremely large values with non-negligible probability. As a consequence, such random variable shows infinite variance when $\alpha < 2$. It is important to notice that heavy-tailedness is a property on the tail distributions.

Note that the heavy-tailed property, and its associated infinite variance property, only exist when the distribution's power-law shape extends to infinity. When a distribution has a power law shape that does *not* extend to infinity, we say that it is "power law over a limited range." Random variables whose distributions are power-law over a limited range may exhibit unusual scaling properties, but these properties do not persist to arbitrarily long timescales.

It has been shown in [4] that the aggregation of i.i.d. ON/OFF processes produces *self-similar* time-series, if either

ON or OFF periods of each process follow a heavy-tailed distribution. We say a time-series is (asymptotically) self-similar if the autocorrelation function of the new time-series produced by aggregating the original time-series is (asymptotically) equal to the original autocorrelation function. That is, given a stationary time-series $X = (X_t : t = 0, 1, 2, ...)$, we define the m-aggregated series $X^{(m)} = (X_k^{(m)} : k = 1, 2, 3, ...)$ by summing the original series X over non-overlapping blocks of size m. Then if X is self-similar, it has the same autocorrelation function $r(k) = E[(X_t - \overline{X})(X_{t+k} - \overline{X})]$ as the series $X^{(m)}$ for all m, where $\overline{X} = E[X]$.

As a result, self-similar processes show *long-range dependence*. A process with long-range dependence has an autocorrelation function $r(k) \sim k^{-\beta}$ as $k \to \infty$, where $0 < \beta < 1$. Thus the autocorrelation function of such a process decays hyperbolically (as compared to the exponential decay exhibited by traditional Markovian traffic models).

One of the attractive features of using self-similar models for time-series, when appropriate, is that the degree of self-similarity of a series is expressed using only a single parameter called the *Hurst* parameter $H = 1 - \beta/2$. For self-similar series, $1/2 < H < 1$. As $H \to 1$, the degree of self-similarity increases. Thus the fundamental test for self-similarity of a series reduces to the question of whether H is significantly different from $1/2$.

Due to the long-range dependence property, it is hard to apply traditional methods to analyze self-similar traffic, e.g., queueing analysis based on Markovian assumption. Therefore, sometimes a long-range dependent series can be approximated by the aggregation of short-range dependent series [11, 12]. Such synthetic series are sometimes called "pseudo-self-similar" or "pseudo-long-range-dependent" series due to the fact that the scaling property disappears at large timescales.

Related Work: There has been a large body of work attempting to explain the causes of heavy-tails and self-similarity in Internet traffic. Broadly, they either attribute the causes to *application/user-level* variability (e.g., [13],[14],[1]) or to *system/network-level* complexity (e.g., [15],[7],[16],[17]).

Willinger *et al.* [13] examined Ethernet traffic at the packet level, identified flows between individual source/destination pairs, and showed that transmission and idle times for those flows are heavy-tailed. Paxson and Floyd [14] traced Internet traffic and observed that burst sizes in FTP transfers, and TELNET packet inter-arrival times (appearance of "keystroke") show heavy-tailed distributions. In [1], Crovella and Bestavros examine Web traffic. They observe that file size distributions in Unix systems as well as in Web databases are heavy-tailed. In addition, heavy-tailed distributions are found in HTTP session time and request inter-arrival time (user "thinking" time). They propose that these heavy-tailed distributions might be the primary causes of the self-similarity in Web traffic.

Other studies have argued that the chaotic nature of network protocols and variability of system conditions may also contribute to the self-similarity in traffic, especially at smaller timescales. Supported by *ns* simulation, Veres *et al.* [7] claimed that under severe network conditions, TCP congestion control protocol shows chaotic nature, and starts to generate self-similar traffic. However, they only show evidence of such chaotic nature, and do not explain why it shows up only when the network is highly congested. In [16], Peha uses simulation to argue that packet retransmission and congestion control mechanisms could cause self-similarity when congestion does happen in the network. However, no theoretical evidence is given in that paper. More recently, Veres *et al.* [17] observe from real measurements that short TCP connections produce self-similar traffic, which they attribute to the reaction of TCP congestion control to the self-similar background traffic.

Our Contribution: By carefully analyzing our extensive simulation results, we find that the "chaotic" property of TCP traffic only appears under certain network conditions. Moreover, since such chaotic property only appears over at most 4 orders of magnitude of timescales, the associated traffic should be described as "pseudo-self-similar." Our work is different from previous work in that we seek to discover the causes of protocol-induced chaos based on analytical arguments. More specifically, we illustrate in this paper that the exponential-backoff algorithm, used by TCP's congestion control mechanism under severe network congestion conditions, can cause "chaotic" behavior. Our analytical model encompasses general loss conditions, which may be due to contention among several TCP connections (as in [7]), or due to self-similar cross-traffic (as in [17]). However, our model also indicates that such "chaotic" behavior only appears in limited timescales, and thus, it is erroneous to identify the time-series as "self-similar".

Our analytical model considers the two most crucial phases of operation of TCP, namely slow-start and exponential-backoff, during which TCP generates highly variable traffic (cf. Section II). Recently, Figueiredo *et al.* [18] extended our model to include the congestion-avoidance phase as well. They show that TCP also exhibits power-law correlation structure in this phase, albeit in a very limited timescale range (less than 2 orders of magnitude).

The paper is organized as follows. In Section II we briefly review TCP's congestion control mechanism. In Section III we propose a Markovian model to describe its behavior under certain network conditions and analytically explain why the inter-arrival times of TCP packets follow a power-law distribution over limited timescales. We use simulation to confirm our analysis in Section IV and show that: *(1)* Under severe congestion conditions, TCP traffic exhibits high variability over small and medium timescales (1 RTT to 100 RTTs); and *(2)* such variability disappears at larger timescale. We discuss extensions to our analysis and future work in Section V.

II. Conservative Nature of TCP and its Pathological Effect

The objective of TCP congestion control is for each source to determine how much capacity is available in the network, so that it knows what rate it can safely send at. Therefore, the data transfer of TCP starts from a stage, called *slow-start*, in which TCP tries to increase its sending rate exponentially, until it encounters the first loss. At this point, TCP interprets packet loss as an indication of reaching the upper limit of the available bandwidth of the bottleneck link. Thus, from this point on (or following another slow-start period, depending on the implementation of TCP), it switches to another stage, called *congestion-avoidance*, in which TCP employs the *Additive Increase, Multiplicative Decrease (AIMD)* mechanism to slowly adapt to the available bandwidth.

Another important stage of TCP congestion control happens when the network is heavily loaded, during which some of the TCP connections should keep silent so as to clear congestion. This stage, referred to in this paper as *exponential-backoff*, is the regime that Karn's algorithm [19] deals with. In this regime, when TCP does not receive an acknowledgment for a packet after some timeout period, it assumes that this packet is lost, and then retransmits that packet and doubles its retransmission timeout value (RTO) for detecting packet loss. This process continues until the packet is successfully transmitted and acknowledged, up to some upper limit (usually 64 times the smallest timeout). Essentially, TCP tries to clear congestion by cutting its sending rate in half (or exponentially decreasing its rate).

Figure 1 shows a schematic view of the TCP congestion window behavior at different stages (black points on the top indicate packet losses). The value of the congestion window is proportional to the sending rate, which is roughly equal to the size of the window divided by the round-trip time (RTT). To simplify our analysis, we focus on the behavior of TCP Tahoe, and we assume that the receiver has an unlimited buffer, so there's no TCP flow control (the upper limit on the window size). We should point out that all versions of TCP behave the same when packet loss is detected by a retransmission timeout; TCP reduces its window to 1 packet and goes to slow-start phase until it reaches half the previous window, where it starts the congestion-avoidance phase. Some versions of TCP (e.g. TCP Reno, New-Reno [20]) attempt to prolong their operation in congestion-avoidance, but eventually go back to slow-start on a retransmission timeout.

Ideally, TCP would operate in the congestion-avoidance stage to efficiently utilize the network resources. From the traffic characteristics point-of-view, TCP generates more stable traffic in this stage than the other two stages (slow-start and exponential-backoff). Unfortunately, two factors drag TCP away from the congestion-avoidance stage: *(1)* Internet measurements [21] show that most flows (both TCP and UDP) are short in size (less than 100 packets). As mentioned earlier, TCP's adaptive control mechanism requires a certain period

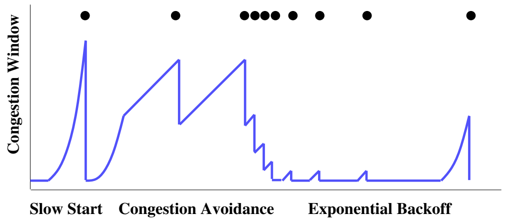


Fig. 1. Behavior of TCP congestion control

of time to learn the state of the network, such as the round-trip time and average share of the available bandwidth. Because most of the TCP connections are too short to generate enough packet samples to obtain such detailed knowledge, most of their packets are sent out at a conservatively estimated rate during slow-start; and *(2)* an even worse situation is that some parts of the current Internet can be highly congested [22, 23], and thus the packet loss rate is relatively high. Loss rates as high as 17% have been observed in portions of the Internet. High loss rate makes consecutive packet losses possible and retransmission timeouts force many TCP connections to stay in the exponential-backoff stage.

Under these conditions, a single TCP connection may spend most or all of its time in slow-start or exponential-backoff stages. This is the situation we explore in the next section.

III. Modeling TCP behavior in Slow-start and Exponential-backoff Stages

In this section, we use a discrete-time Markov chain to describe the behavior of TCP in slow-start and exponential-backoff stages and explain why it generates power-law distribution in packet inter-arrival times. To this end, we assume that TCP's window adaptation policy depends only on the last value of the congestion window. We can then draw a Markov chain as shown in Figure 2. We define the state

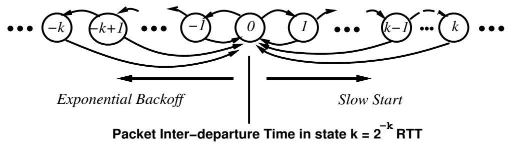


Fig. 2. Modeling short TCP congestion control

of the Markov chain to be the negation of the binary logarithm of the packet inter-departure time, normalized by the average round-trip time, for each outgoing packet. For example, the -1 state means the current packet is going to be sent out 2 round-trip times after the previous packet. Notice that for slow-start phase (state index $k > 0$), there are more than one (up to the current window size W) packet samples for each state. To simplify our analysis, we assume that all packets in the window are emitted evenly over the round-trip time. Thus each state actually contains W mini-states, each of which corresponds to one packet in the current window, with inter-departure time equals to $1/W$ round-trip time. However, for simplicity, we ignore the transition between these mini-states, but rather analyze the aggregated state with state index $\log_2 W$.

A. *Modeling Slow-Start*

Figure 3 illustrates the Markov chain for TCP slow-start stage. During this stage, TCP increases its window size by one

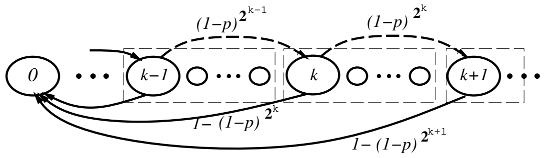


Fig. 3. Markov chain for slow-start

every time it gets an acknowledgment. Therefore essentially TCP doubles its window size every round-trip time if none of the packets in the previous window gets lost during the previous round-trip time. Assuming loss event is independent of the TCP behavior and the loss rate is constant p, then the probability of not losing any packet is $(1-p)^W$, where W is the previous round's window size. Recall that we define state k to denote a TCP window size of 2^k, or equivalently a rate of 2^k packets per RTT. Thus, the probability of transition from state k to $(k+1)$ equals to $(1-p)^{2^k}$, as shown in Figure 3. On the other hand, when packet loss is encountered,[1] TCP reduces its window size to 1, and goes to slow-start again. Thus, the transition probability from state k to state 0 is $1-(1-p)^{2^k}$.

Denote by π_k the probability that TCP is in state k, then we can write the following balance equations of the Markov chain:

$$\pi_k = (1-p)^{2^{(k-1)}} \pi_{k-1} \quad \text{for } k > 0$$

Therefore, we have

$$\begin{aligned} \pi_k &= (1-p)^{2^{(k-1)}} \pi_{k-1} \\ &= \ldots \\ &= (1-p)^{2^{k(k-1)/2}} \pi_0 \quad \text{for } k > 0 \end{aligned} \qquad (1)$$

B. *Modeling Exponential-Backoff*

The other part of the Markov chain is shown in detail in Figure 4. In this stage, TCP tries to retransmit the packet lost in

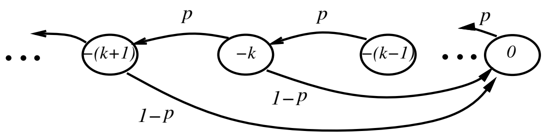


Fig. 4. Markov chain for exponential-timeout

the previous round. The window value is kept at 1, and TCP employs Karn's algorithm to estimate the round-trip time, or the maximum time it needs to wait for the acknowledgment of the retransmitted packet. Instead of using the sampled round-trip time, TCP backs off the retransmission timeout (RTO) exponentially until the packet is acknowledged, usually starting from the latest RTT sample for a new (not-retransmitted) packet. If no RTT samples have yet been received, RTO is usually set to an initial default value.

During exponential-backoff, if TCP does not receive the acknowledgment of the retransmitted packet before RTO expires, either because the packet or its acknowledgment is lost (with probability p), or because the actual RTT is greater than RTO, then TCP doubles RTO and repeats the above process. The transition probability from state $-(k-1)$ to $-k$ in Figure 4 corresponds to this case.[2]

We can derive the state probabilities from the following balance equations:

$$\pi_{-k} = p\pi_{-(k-1)} \quad \text{for } k > 0$$

Therefore, we have:

$$\begin{aligned} \pi_{-k} &= p\pi_{-(k-1)} \\ &= \ldots \\ &= p^k \pi_0 \quad \text{for } k > 0 \end{aligned} \qquad (2)$$

C. *Comments and Observations on the Analysis*

Note that in Figure 2, we do not include the congestion-avoidance stage. Our objective here is to model *short* TCP connections. Many Internet measurements [5, 21] have shown that most of the Internet flows are short-lived. Since TCP needs to conservatively adapt to the available bandwidth in the network, it is often the case that short TCP connections end their transmission before they really enter the congestion-avoidance stage. Current versions of TCP try to avoid being too pessimistic when a packet loss is encountered. They employ the *fast retransmit and recovery* [20] mechanism to avoid going to exponential-backoff stage too often. To do this, if the TCP sender receives the K^{th} duplicate acknowledgment packet before the timer expires,[3] the sender will behave as if a timeout has occurred and retransmits while remaining in congestion-avoidance. However, for short connections, since they rarely have a chance to open their window big enough to have the fast retransmit mechanism workable, most of their lifetime is spent in exponential-backoff and slow-start. Thus, the model in Figure 2 is more suited for short TCP connections, which constitute the majority of TCP flows in the Internet today.

The steady-state probability distribution function for the states in the Markov chain is plotted in Figure 5:

[1] This is how TCP Tahoe always behaves upon detecting a packet loss.

[2] Note that in Karn's algorithm, if a packet has been sent more than once, the initial RTO for the next packet is set to the RTO of the previous transmission, not the estimated round-trip time. In other words, TCP stays in state $-k$ instead of going back to state 0. We ignore this detail to keep our Markovian analysis simple.

[3] K is usually 3.

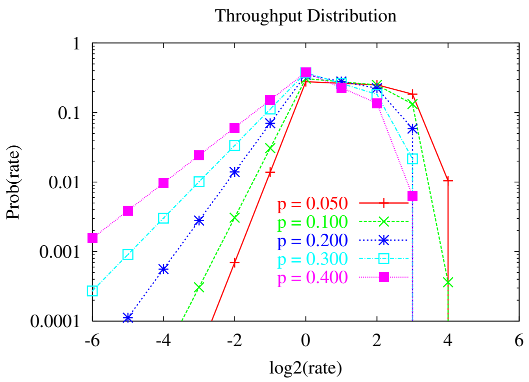


Fig. 5. State probability with different loss rate

Observing the part of the curve for the exponential-backoff stage (the left side), when p is large enough (e.g. $p > 12\%$), the distribution of packet inter-arrival times from the single TCP source starts to exhibit a tail consistent with infinite variance (if it is extended to infinity). This is shown in the following analysis. We have noticed from equation (2) that $\pi_{-k} = p^k \pi_0$, where π_{-k} corresponds to the probability of inter-arrival of (retransmitted) packets equals 2^k RTT. Denote by T the random variable which describes the packet inter-arrival time in the exponential-backoff stage, then we have:

$$\begin{aligned} Pr[T = t = 2^k] &= \pi_{-k} \\ &= p^k \pi_0 \\ &= (2^k)^{-\log_2(1/p)} \pi_0 \\ &= t^{-\log_2(1/p)} \pi_0 \\ &= t^{-(\alpha+1)} \pi_0 \end{aligned} \tag{3}$$

where α is the slope (exponent) of the corresponding complementary cumulative distribution, and is obtained by:

$$\alpha = \log_2(1/2p) \tag{4}$$

If the Markov chain were infinite, i.e., k had no upper bound, the distribution would have been categorized as heavy-tailed when $1 < \alpha < 2$, or $\frac{1}{8} < p < \frac{1}{4}$. Furthermore, the aggregation of such time-series would have shown self-similarity [4]. However, the TCP protocol defines an upper bound on the timeout value, which is $\max(64\ RTO, T_M)$, where the value of T_M depends on the implementation. Thus, usually $k \leq 6$. When combined with other source of variabilities, e.g., variability in initial RTO values (c.f. Section IV-D), this means such power-law shape can span about 3 decimal orders of magnitude of timescales. Therefore, it is more appropriate to classify such traffic as "pseudo-self-similar" because the scaling property would disappear at larger timescales. Moreover, as the loss rate increases, the chance to have longer silent period increases, and so does the variability and the timescale it spans.

IV. Simulation with Pathological TCP Connections

A. Demonstrating Pseudo-self-similarity by Simulation

We use a simple experiment with the *ns* [8] simulator to validate the above observation. As in many other similar studies [7, 23], we simulate the case in which TCP connections pass through a bottleneck link. However, because we are interested in the behavior of a single TCP flow, instead of simulating the case of many TCP flows competing for the bottleneck resource, we simulate only a single connection transmitting over a lossy link. The topology is shown in Figure 6. Node S

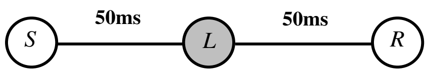


Fig. 6. Simulation topology

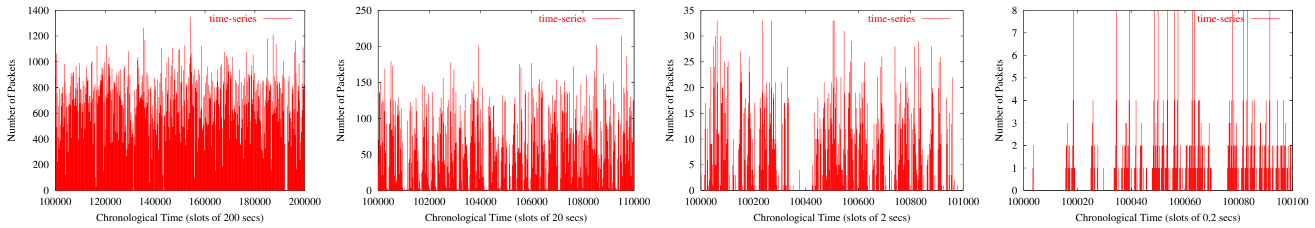
is the sender, node R is the receiver, the two-way propagation delay between S and R is set to 200 msec. We put a loss module inside node L, which drops incoming (new or retransmitted) packets uniformly with probability p. The packet size is set to be fixed at 576 bytes. The link capacity, buffer size and receiver window size are set to very large values so that only the loss module affects TCP's performance. The size of each TCP connection is set to be fixed at $FlowSize$ packets. A terminated connection is immediately replaced by a new connection (with all TCP parameters reset). We use the Tahoe version of TCP.[4]

We traced packet arrival events both at the upstream link (S, L) and downstream link (L, R) of node L.[5] Figure 7 shows four time-series plots of the traffic on the upstream link. The plots are produced by aggregating packet traffic into discrete bins of 0.2, 2, 20, or 200 seconds. We observe that traffic is bursty over all timescales shown, and that the variability does not decline rapidly as the timescale increases. This is suggestive of an unusual scaling property in the traffic trace.

To demonstrate that this traffic can exhibit properties that might be misidentified as self-similarity, we apply two popular methods used for assessing self-similarity. For these tests we used a 55-hour packet trace from a simulation with parameters

[4] We also simulated TCP Reno. The simulation results are very similar to TCP Tahoe. Note that in the regime of high and bursty packet losses we are considering in this paper, TCP Tahoe is actually more robust to multiple losses than TCP Reno. Moreover, our model has no dependence on the congestion avoidance phase of TCP and the differences between Tahoe and other versions are not significant for this loss regime, even with a maximum window size or delayed ACKs. This is especially true for short TCP flows with small window size, which is likely not to trigger the fast retransmit and recovery mechanisms.

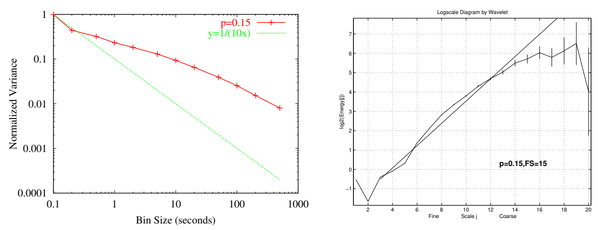
[5] All the data shown in this paper are collected at the upstream link (S, L), i.e., the data includes all packets, including those that are later lost at node L. Excluding those (eventually lost) packets affects the distributions (packet inter-arrival times, burst sizes) at small timescales, but does not affect the tail shape of the distributions (i.e., at large timescales). Thus the major conclusions of this paper still hold for packet traces collected on the downstream link (L, R), for which results are not shown.



(a) Bin Size = 200 seconds (b) Bin Size = 20 seconds (c) Bin Size = 2 seconds (d) Bin Size = 0.2 seconds

Fig. 7. Traffic burstiness over four orders of magnitude

$p = 0.15$ and $FlowSize = 15$ packets.[6] Figure 8 shows the results. Roughly, with the Aggregated Variance method, for a self-similar process, the variance of the time average goes to zero very slowly as the bin size, or aggregation level, becomes larger. In Figure 8(a), the sample variance is plotted versus bin size on a log-log plot. For a self-similar process, the result should be a straight line with a slope of $2H - 2$. If the time-series has no long-range dependence, then $H = 0.5$ and the slope should be -1 [24]. With the Wavelet Analysis Method, for a self-similar process, the degree of variability, or burstiness, decreases very slowly at larger timescales. In Figure 8(b), this variability is represented by "energy" estimates with 95% quantiles, computed using the tool of Veitch and Abry [25]. The plot of energy versus timescale should be a straight line with a slope of $2H - 1$.[7] Observe from these graphs that the traffic time-series shares properties similar to a self-similar signal, but only over the timescales from few RTTs to hundreds of RTTs.

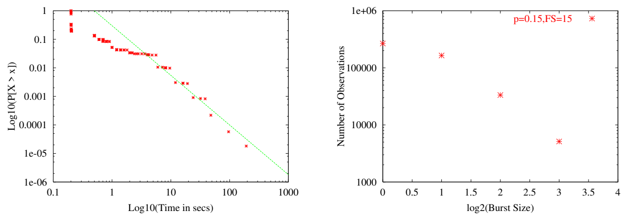


(a) (b)

Fig. 8. LRD analysis on packet trace with loss rate = 15%, (a) Aggregated Variance Method H = 0.76, (b) Wavelet Method H = 0.785 (0.783,0.787)

B. *Power laws in Packet Inter-arrivals: How and Why*

Greiner *et al.* show in [27] that a renewal process where the interarrival times have a heavy-tail yields self-similar scaling behavior. Our simulation results, which closely match our analysis in Section III, show that packet inter-arrivals under TCP can indeed have a power-tail, causing scaling behavior in the overall traffic over limited timescales. Figure 9 shows simulation results for $p = 0.15$ and $FlowSize = 15$ packets.



(a) (b)

Fig. 9. The inter-arrival time and burst size distribution of TCP in (a) exponential-timeout (CDF), (b) slow-start stage (pdf)

We consider packets whose inter-arrival time is less than 2 times the packet transmission time (about 1 millisecond in our simulation) to constitute a burst. Bursts are only emitted in slow-start, so their presence does not affect the heavy tails in our Markov chain model. The probability of a state π_k $(k > 0)$ can be thought of as the probability of a burst of size 2^k in our *ns* simulation. Therefore, we use two plots to represent the inter-packet arrival behavior. Figure 9(a) shows the log-log complementary distribution (LLCD) plot for those inter-arrival times bigger than 1 millisecond, and Figure 9(b) shows the distribution of packet arrival in bursts. Roughly, Figure 9(a) describes TCP behavior in the exponential-timeout stage (and part of the slow-start stage), while Figure 9(b) describes the slow-start stage. Note that Figures 9(a) and (b) are plotted in $\log_{10}$-$\log_{10}$ and $\log_{10}$-$\log_2$ scales, respectively. Also note that each burst corresponds to an aggregated state in the Markov chain (with state index $k > 0$).

We observe that the inter-arrival time distribution in Figure 9(a) clearly has a tail that shows hyperbolic shape. The function given by the linear least-squares fit has the shape $\alpha = \log_2 3.33 = 1.74$ (implying high variance). The shape of the tail matches the value obtained by analysis from equation (4) in Section III-C. However, as discussed in our analysis, TCP defines a bound on the maximum timeout value, thus the tail of the inter-arrival time distribution is cut off at this upper bound and therefore the distribution can not be categorized as heavy-tailed. As a result, the scaling property would disappear at larger timescale.

The burst size distribution in the slow-start plot in Figure 9(b) also shows a shape similar to our analysis given in

[6] For different simulation setups, the packet trace contains hundreds of thousands to few millions of packets transmitted over the upstream link.

[7] Note in Figure 8(b), the "dip" in the Wavelet plot reflects the strong correlation introduced by the network round-trip time (of 200 msec. here), as explained in [26].

Figure 5, where it decays very fast due to the relatively high loss probability.

C. Effect of Loss Rate

We also vary the loss rate p in our simulation experiments. Figure 10 plots the Wavelet analysis and the inter-arrival time distribution for $p = 0.1$ and $p = 0.2$. Observe that the slopes of the linear least-squares fit obtained by simulation closely match those obtained by analysis (cf. equation (4)).

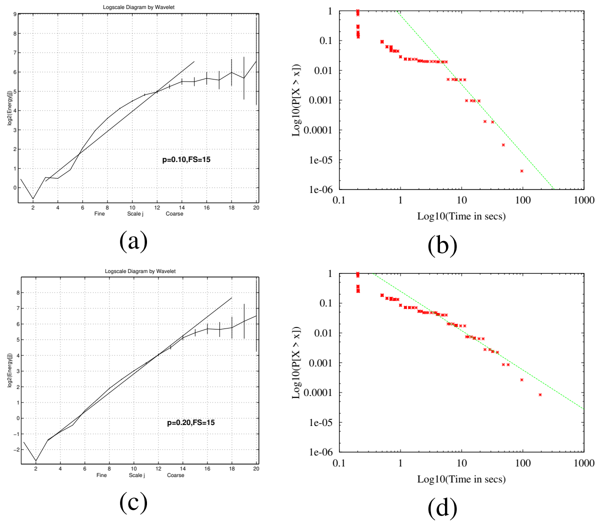


Fig. 10. Simulation with loss rates p = 0.1 (a,b), and p = 0.2 (c,d): (a) p = 0.1, Wavelet Method H = 0.761, (b) LLCD of inter-arrival times, slope = $-\log_2 5$ = -2.32, (c) p = 0.2, Wavelet Method H = 0.803, (d) LLCD of inter-arrival times, slope = $-\log_2 2.5$ = -1.32.

When the loss rate is relatively low, the scaling property exhibited by the time-series is less significant. This is because the tail of the inter-arrival time (or the OFF period) distribution is less heavy and thus at larger timescales, the traffic starts to show less variance. On the contrary, with higher loss rate, the tail of the OFF periods is heavier and the statistical scaling property is more significant.

Since the time-series is not self-similar, it is meaningless to estimate the Hurst parameter. However, we can still compute a similar measurement over the time-series, *if we assume* that there were no upper bound on the maximum timeout value, so that the power-law tail extended to all larger timescales. Such "pseudo-Hurst" parameter is estimated from the linear square fit in the wavelet plot over the timescales that the scaling property shows up, and is shown in the figures. As we observed in our analysis, when loss rate goes up, the burstiness, represented by the "pseudo-Hurst" parameter increases, and the timescale over which the scaling property shows up increases too.

D. The Variability of Initial Retransmission Timeout Value

There are two reasons to have variable initial timeout. The first is brought by variable queueing delays. TCP includes the

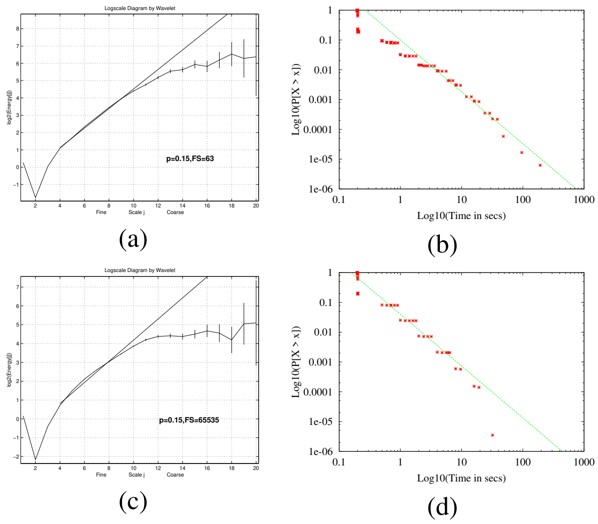


Fig. 11. Simulation with file sizes $FlowSize$ = 63 (a,b), and $FlowSize$ = 65535 (c,d): (a) Wavelet Method H = 0.780, (b) LLCD of inter-arrival times, slope = -$\log_2 3.33$ = -1.74, (c) Wavelet Method H = 0.782, (d) LLCD of inter-arrival times, slope = -$\log_2 3.33$ = -1.74.

variance of the RTT in its RTT estimate, which is then used to compute the retransmission timeout value. The second factor, which is more significant in our model, is that for the first packet, the initial timeout value is set to a default value because TCP does not have any packet sample to estimate the round-trip time. This default value is suggested to be 3 seconds in [28], and is set to 6 seconds by default in the *ns* simulator. Recall that the two-way propagation delay in our simulation is 0.2 second, so if TCP had enough packet samples to accurately estimate the round-trip time, the initial retransmission timeout value would have been set to a value close to 0.2-0.4 second.[8] This huge difference (about 15 times) results in some packet inter-arrival times to be as high as 64×6 = 384 seconds, if the transmission of a packet fails 6 times starting from a default RTO of 6 seconds. The effect of the default initial timeout value is less significant in two cases: when its value is closer to the estimated value, and when the connection size is large. For these two cases, we conducted the following two experiments.

D.1 Effect of File (Flow) Size

We compare the packet traces generated from sending different file sizes over TCP. We fix the loss rate at 15%. Figure 11 shows both the Wavelet analysis on the packet trace and the packet inter-arrival time distribution for $FlowSize =$ 63 packets and $65,535$ packets.

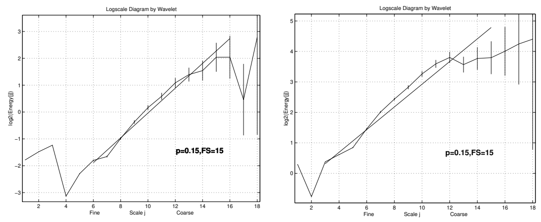
We observe that as the file size increases, the time-series shows less variations at larger time scales; this is apparent from the less heavy tail of the LLCD plot and the Wavelet plot bending down at large timescales. We attribute this reduced

[8] Observe that Figure 9 shows a minimum packet inter-arrival time of 0.2 second.

variability to the more accurate information TCP learned from increased packet samples.[9]

D.2 Effect of Default Initial Timeout Value

We conducted two experiments to reduce the difference between the default initial timeout value and the normal timeout value. In the first experiment, we increase the two-way propagation delay from 0.2 second to 1 second. In the second experiment, we reduce the default initial timeout value from 6 seconds to 0.5 second. Figure 12 shows the Wavelet analysis of the time-series produced by the captured packet traces for $FlowSize = 15$ packets and $p = 0.15$.



(a) (b)

Fig. 12. Reducing the difference between timeout values: (a) increase round-trip propagation delay to 1 second, H = 0.731, (b) decrease default timeout value to 0.5 second, measured H = 0.686.

As the variance of the timeout values becomes smaller, the pseudo-long-range dependence in the time-series becomes less significant.[10]

E. Explaining "Chaotic" TCP behavior

In [7], Veres *et al.* show that when 40 TCP connections are squeezed into a small pipe (with 1Mbps bandwidth and 30ms two-way propagation delay, so that on average, each connection can transmit less than one packet in one round-trip time), the time-series produced by each connection shows pseudo-long-range dependence. We repeated this experiment and plot the Wavelet analysis of the traffic produced by one of the TCP connections in Figure 13.

Notice that Veres' experiment produces pseudo-self-similarity as well. We can now explain this phenomena based on our analysis. The average packet loss rate measured in this experiment is around 16%. At this loss level, the burstiness measure have a value similar to our experiment results shown in Figures 8 and 11. The difference is that in Veres' experiment, each connection has infinite packets to transmit, so the effect of large initial timeout value should have vanished. However, the timescale over which the scaling property

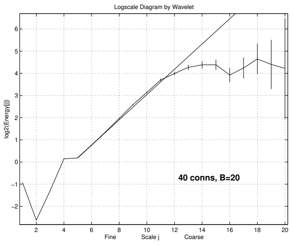
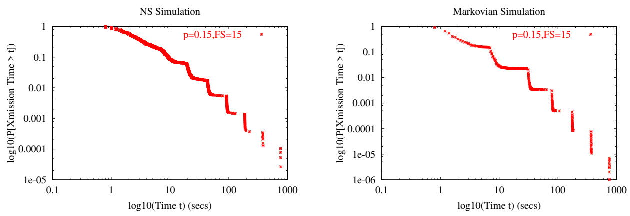


Fig. 13. Veres' experiment result over larger timescale range, $H = 0.786$.

shows up in Veres' experiment is longer than what we get in the long TCP connection case (c.f. Figure 11(c)). We attribute such longer timescale to the variabilities in timeout estimations brought by variable queueing delays, which have similar effect as different initial timeout values observed in our experiments with small file sizes. We observe that it is these two factors that contribute to pseudo-heavy-tailed packet inter-arrival times, or the "chaotic" nature of the TCP congestion control.

F. Effect of TCP Congestion Control on Transmission Times



(a) *ns* simulation (b) Simulated Markov chain

Fig. 14. Validation of high variability in transmission time, $FlowSize$ = 15, p = 15%

In [29], the authors noted that the omnipresence of heavy-tails in file size distribution in the majority of file systems is not the only reason behind the heavy-tailed distribution of HTTP file transfers. They pointed out that the network condition is also important by observing that there is *no* strong correlation between file transmission time and file size. However, that paper made no attempt to explore alternate reasons for heavy-tailed transmission times.

We observe that in addition to these reasons, TCP's adaptive congestion control mechanism in the presence of severe network conditions can be another major contributor to power-law tails, over a limited range. We run simulations on both the *ns* simulator and the Markov chain described in Section II to validate this observation. Figure 14 shows the LLCD plot of the transmission time distribution for transferring a 15-packet file over a lossy channel with 15% loss rate. The stair-step shape of the plot is due to the small file size (15 packets) leading to a small set of possible packet-loss events and thus transmission times. Nonetheless it shows an approximate power-law shape (the general trend apart from the stair-stepping) that

[9] We note that this result does not contradict those in [17] regarding long TCP flows propagating self-similar traffic in the presence of self-similar background traffic. Veres $et\ al.$ [17] do not explain the origin of this background self-similar traffic. Our results provide high loss rate leading to exponential backoffs in short TCP flows as a contributing factor. Note that a number of measurements show that the majority of TCP flows are short.

[10] This observation agrees with [16], in which Peha observes that for larger or more random timeouts, self-similar behavior is observed at larger timescales.

is indicative of heavy tails in transmission times. Larger files would presumably show a smoother curve. This surprising effect means that under certain network conditions, huge difference in transmission times can occur even in the absence of *any* variability in file sizes.[11]

V. Conclusion and Future Work

The principle of TCP congestion control is to avoid overloading the network while still maintaining network usage efficiency. The TCP control algorithm is simple but effective. However, this simplicity does not come for free. The behavior of TCP becomes less predictable or even chaotic when the network condition is out of TCP's control. In this paper, we have demonstrated that when a TCP connection is going through a highly lossy channel and the loss condition is not affected by this single TCP connection's behavior, TCP starts to produce packet trains that show pseudo-self-similarity. We used a simple Markovian model to demonstrate why limited-range power-laws show up in packet inter-arrival times and file transmission times. Our analysis sheds some light on the relationship between TCP conservative control mechanism and pseudo-self-similarity observed in Internet traffic, especially over small timescales [7, 16, 17].

Our future work includes extending our analytical model to consider the variability in round-trip time estimation, and verifying our analytical model in more complex network settings. We also plan to study possible improvements to TCP congestion control mechanism under severe network conditions. We are also planning to investigate the relationship between network-level self-similarity and application-level self-similarity (especially for network-aware adaptive applications).

References

[1] M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Transactions on Networking*, pp. 835–846, December 1997.

[2] A. Feldmann, A.C. Gilbert, W. Willinger, and T.G. Kurtz, "The Changing Nature of Network Traffic: Scaling Phenomena," *ACM Computer Communication Review*, pp. 5–29, April 1998.

[3] 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*, pp. 1–15, Feburary 1994.

[4] W. Willinger, V. Paxson, and M.S. Taqqu, "Self-Similarity and Heavy-Tails: Structural Modeling of Network Traffic," in *A Practical Guide To Heavy Tails: Statistical Techniques and Applications*, R.J. Adler, R.E. Feldman and M.S. Taqqu, editors. ISBN 0-8176-3951-9. Birkhäuser, Boston, 1998.

[5] K. Thompson, G.J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE/ACM Transactions on Networking*, pp. 10–23, November 1997.

[6] R.T. Morris, *Scalable TCP Congestion Control*, Ph.D. thesis, Harvard University, Cambridge MA, The Divison of Engineering and Applied Sciences, Januray 1999.

[7] A. Veres, B. Hungary, and M. Boda, "The Chaotic Nature of TCP Congestion Control," in *Proceedings of IEEE INFOCOM'2000*, March 2000.

[8] UCB, LBNL, and VINT, "Network simulator - ns (version 2)," http://www-mash.cs.berkeley.edu/ns/, 1999.

[11] This high variability in file transmission times is consistent with that observed in [6].

[9] R.E. Feldman R.J. Adler and M.S. Taqqu, *A Practical Guide To Heavy Tails: Statistical Techniques and Applications*, ISBN 0-8176-3951-9. Birkhäuser, Boston, 1998.

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

[11] S. Robert and J. Y. Le Boudec, "On a Markov Modulated Chain Exhibiting Self-Similarities Over Finite Timescale," *Performance Evaluation*, pp. 159–173, October 1996.

[12] Anja Feldmann and Ward Whitt, "Fitting Mixtures of Exponentials to Long-tail Distributions to Analyze Network Performance Models," *Performance Evaluation*, 1998.

[13] 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," *IEEE/ACM Transactions on Networking*, pp. 71–86, Feburary 1997.

[14] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, pp. 236–244, June 1995.

[15] J.H.B. Deane, C. Smythe, and D.J. Jefferies, "Self-Similarity in a Deterministic Model of Data Transfer," *Journal of Electronics*, vol. 80, no. 5, pp. 677–691, 1996.

[16] J. M. Peha, "Retransmission Mechanisms and Self-Similar Traffic Models," in *Proceedings of IEEE/ACM/SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, Phoenix, Arizona, Jan. 1997, pp. 47–52.

[17] A. Veres, Zs. Kenesi, S. Molnár, and G. Vattay, "On the Propagation of Long-Range Dependence in the Internet," in *Proceedings of ACM SIGCOMM 2000*, Stockholm, Sweden, Aug.-Sep. 2000.

[18] D. Figeiredo, B. Liu, V. Misra, and D. Towsley, "On the Autocorrelation Structure of TCP Traffic," Tech. Rep. TR00-55, UMass CMPSCI, November 2000, ftp://gaia.cs.umass.edu/pub/.

[19] P. Karn and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," *ACM Transactions on Computer Systems (TOCS)*, vol. 9, pp. 365–373, 1991.

[20] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP," *ACM Computer Communication Review*, pp. 5–21, July 1996.

[21] K. Claffy, G. Miller, and K. Thompson, "the Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," in *Proceedings of INET'98*, http://www.caida.org/outreach/papers/Inet98 1998.

[22] V. Paxson, "End-to-End Internet Packet Dynamics," in *Proceedinds of ACM SIGCOMM'97*, Cannes, France, September 1997.

[23] R.T. Morris, "TCP Behavior with Many Flows," in *IEEE International Conference on Network Protocols (ICNP'97)*, Atlanta, Georgia, October 1997.

[24] Murad Taqqu, "Statistical Methods for Long-Range Dependence," http://math.bu.edu/people/murad/methods/index.html.

[25] D. Veitch and P. Abry, "A Wavelet Based Joint Estimator of the Parameters of Long-Range Dependence," *IEEE Transactions on Information Theory*, April 1999.

[26] A. Feldmann, A.C. Gilbert, P. Huang, and W. Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proceeding of ACM/SIGCOMM'99*, Boston, Mass, September 1999.

[27] M. Greiner, M. Jobmann, and L. Lipsky, "The Importance of Power-Tail Distributions for Modeling Queueing Systems," *Operations Research*, vol. 47, no. 2, March-April 1999.

[28] R. Braden, "Requirements for Internet Hosts – Communication Layers," Internet RFC 1122, October 1989.

[29] M.E. Crovella, M.S. Taqqu, and A. Bestavros, "Heavy-Tailed Probability Distributions in the World Wide Web," in *A Practical Guide To Heavy Tails: Statistical Techniques and Applications*, R.J. Adler, R.E. Feldman and M.S. Taqqu, editors. ISBN 0-8176-3951-9. Birkhäuser, Boston, 1998.