Source-Level IP Packet Bursts: Causes and Effects[*]

Hao Jiang
College of Computing
Georgia Institute of Technology
jiang@cc.gatech.edu

Constantinos Dovrolis
College of Computing
Georgia Institute of Technology
dovrolis@cc.gatech.edu

ABSTRACT

By source-level IP packet burst, we mean several IP packets sent back-to-back from the source of a flow. We first identify several causes of source-level bursts, including TCP's slow start, idle restart, window advancement after loss recovery, and segmentation of application messages into multiple UDP packets. We then show that the presence of packet bursts in individual flows can have a major impact on aggregate traffic. In particular, such bursts create scaling in a range of timescales which corresponds to the burst duration. Uniform "spreading" of bursts in the time axis reduces the scaling exponent in short timescales (up to 100-200ms) to almost zero, meaning that the aggregate traffic becomes practically uncorrelated in that range. This result provides a plausible explanation for the scaling behavior of Internet traffic in short timescales. We also show that removing packet bursts from individual flows reduces significantly the tail of the aggregate marginal distribution, and it improves queueing performance, especially in moderate utilizations (50-85%).

Categories and Subject Descriptors: C.2.3 [Network Operations]: *Traffic modeling and analysis*

General Terms: Measurement, Performance

Keywords: scaling, network traffic, TCP, packet dispersion, packet trains, capacity estimation, correlation structure

1. INTRODUCTION

By *source-level IP packet burst*, we mean several IP packets sent back-to-back, i.e., at the maximum possible rate, from the source of a flow. Source-level bursts introduce strong correlations in the packet interarrivals of individual flows. Which protocol mechanisms create such bursts? Over which timescales do the corresponding correlations extend? Significant research efforts have focused recently on the correlation structure, or *scaling behavior*, of aggregate IP traffic in short timescales, typically up to a few hundreds of milliseconds [1, 2, 3, 4]. Is the short time scaling behavior of aggregate traffic related to the presence of packet bursts in individual flows? How will the correlation structure of aggregate traffic change if flows do not include such bursts? In terms of network performance, how will the queueing delays decrease if we remove bursts from individual flows, and in what load conditions is such a decrease most important? These are some of the questions that we investigate in this paper.

[*]This work was supported by the "Scientific Discovery through Advanced Computing" (SciDAC) program of DOE (DE-FC02-01ER25467), and by an equipment donation from Intel Corporation.

Background on scaling. The key tool that we rely on is the *wavelet-based multiresolution analysis* developed in [5] and implemented in [6]. This statistical tool allows us to observe the *scaling behavior* of a traffic process over a certain range of timescales. Consider a reference timescale T_0, and let T_j=$2^j T_0$ for j=1,2,... be increasingly coarser timescales. These timescales, or simply scales, partition a traffic trace in consecutive and non-overlapping time intervals. If t_i^j is the i'th time interval at scale j>0, then t_i^j consists of the intervals t_{2i}^{j-1} and t_{2i+1}^{j-1}. Let X_i^j be the amount of traffic in t_i^j, with $X_i^j = X_{2i}^{j-1} + X_{2i+1}^{j-1}$. The *Haar wavelet coefficients* $\{d_i^j\}$ at scale j are defined as

$$d_i^j = 2^{-j/2}(X_{2i}^{j-1} - X_{2i+1}^{j-1}) \qquad (1)$$

for $i = 1, \dots N_j$, where N_j is the number of wavelet coefficients at scale j. The *energy function* $\mathcal{E}_j$ is defined as

$$\mathcal{E}_j = E[(d_i^j)^2] \approx \frac{\sum_i (d_i^j)^2}{N_j} \qquad (2)$$

An *energy plot*, such as Figure 1, shows the logarithm of the energy $\mathcal{E}_j$ as a function of the scale j. The magnitude of $\mathcal{E}_j$ increases with the variability of the traffic process X^{j-1} at scale j-1. What is more important is *the scaling behavior of the process, i.e., the variation of $\mathcal{E}_j$ with j.* For an exactly self-similar process, such as fractional Brownian motion (fBm) with Hurst parameter H (0.5<H<1), it can be shown that $\mathcal{E}_j$=$\mathcal{E}_0 2^{j(2H-1)}$, and so the energy plot is a straight line with positive slope 2H-1. The slope of an energy plot is referred to as *scaling exponent* and is denoted by α. For fBm, α=2H-1 is constant across all timescales, and so the process is said to show *global scaling.*

To illustrate the detection of scaling in a traffic process, Figure 1 shows the energy plots for three synthetic traces, all of which have the same mean packet interarrival (50ms).

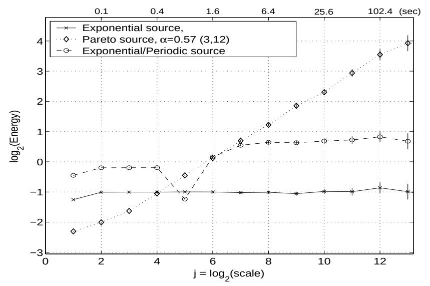

Figure 1: Energy plot examples.

At the top of the graph we show the timescale T_j that corresponds to scale j at the x-axis (T_0=25ms). The first trace is a Poisson process. The signature of uncorrelated exponential interarrivals in the energy plot is a horizontal straight line (α=0). The second trace is again a renewal process, but this time the interarrivals follow the Pareto distribution with shape parameter β=1.5. The infinite variance of the interarrivals creates global scaling. The signature of such global scaling in the energy plot is a straight line segment with positive slope (α=2-β) across all timescales. The third trace is again based on exponential interarrivals, but this time we introduce a strong periodicity at the 400ms timescale. Specifically, after each exponential interarrival we generate, with probability 0.75, another packet 400ms later (scale 4). This periodicity causes a "dip" in the energy plot at the 800ms timescale (scale 5). This is because a periodicity reduces the variability of the traffic process at the corresponding timescale. Note that the dip appears at scale 5, instead of 4, because the energy at scale j depends on the traffic process variations in scale j-1.

In practice, network traffic can show different scaling behavior across different timescales. If the slope of the energy plot is (roughly) constant over a range of timescales j to $j+k$, we say that the traffic process exhibits *local scaling* in the timescales T_j to T_{j+k}. This paper focuses on the scaling behavior of IP traffic in short timescales, typically extending up to a few hundreds of milliseconds.

Related work. Our work is mostly related to previous research on the scaling behavior of Internet traffic in short timescales. One of the first papers that reported scaling in short timescales at WAN traces was [7]. In [1, 8], Feldmann et al. used the wavelet-based multiresolution analysis technique of [5] to detect and characterize the scaling behavior of Internet traffic. The authors showed that scaling in short timescales is related to the TCP closed-loop flow control, and the cutoff between "short" and "long" timescales is, roughly, the RTT of the TCP transfers. Additionally, [1] provided empirical evidence that WAN traffic can be modeled using a *multifractal model*, similar to that developed in [2]. More recent work, however, argues that the traffic at a tier-1 ISP is well-modeled as monofractal, rather than multifractal [3].

[9] showed that IP layer scaling does not depend on the TCP flow arrival process. In a follow-up work, [4] showed that the the correlation structure of aggregate traffic in short timescales can be captured by a Poisson cluster process in which the packet interarrivals within individual clusters follow an overdispersed Gamma distribution. [3] introduced the concept of "dense flows" (i.e., flows with bursts of densely clustered packets), and showed that it is this kind of flows that create scaling in short timescales. [10] showed that scaling in fine timescales can have a significant impact on queueing performance, especially in moderate utilizations, while scaling in coarser timescales is more important in heavy utilizations. Our main result, connecting scaling in short timescales with packet bursts from individual flows, is in agreement with the results of [3, 9, 4, 10], providing a more specific explanation for the nature and causes of scaling behavior in aggregate traffic.

The traces that we used in this study are publicly available at the NLANR-MOAT site [11]. Each trace lasts for 90 seconds. The traces that we include in this paper come from OC-12 links at the Merit (MRA) and Indiana University (IND) Internet2 GigaPOPs. The rest of this paper is structured as follows. §2 gives several causes of source-level bursts in IP traffic. §3 shows that packet bursts create scaling in a range of timescales which corresponds to the burst duration. §4 investigates the effect of bursts from individual flows on aggregate traffic in terms of scaling in short timescales, marginal distribution, and queueing performance. The Appendix describes a passive capacity estimation methodology, which is required for the detection of packet bursts from individual flows at a trace.

2. CAUSES OF SOURCE-LEVEL BURSTS

We have analyzed dozens of traces, attempting to identify the most common causes of source-level bursts. Figure 2 shows nine such causes, one for UDP and eight for TCP flows. Unfortunately, our analysis is not automated, and so we cannot make quantitative statements regarding the relative frequency of each cause. We believe, however, that Figure 2 shows most, or all, major causes.

UDP message segmentation. When a UDP-based application sends a message that is larger than the path's MTU, the message is segmented by the application into multiple UDP packets, and/or it is fragmented by the operating system into multiple IP packets. The example shown in Figure 2 is from a UDP video flow which sends six packets every 40ms.

Slow start. Slow start increases the congestion window by one MSS for every new ACK. This rapid increase can double the burst length in each RTT. In the example shown, the receiver does not use Delayed-ACKs, and so the bursts are one third longer than normally.

Loss recovery with Fast Retransmit. The recovery of a lost segment through Fast Retransmit can fill in a "hole" in the receiving sliding window, and so it can cause a rapid advancement of the ACK number. The sender can then send up to $CW/2$ bytes back-to-back, where CW is the congestion window before the loss. Note that the connection of our example did not do Fast Recovery, as new segments were not sent in response to duplicate ACKs that followed the retransmission, due to congestion window constraints.[1]

Unused congestion window increases. Sometimes, mostly with applications that initially exchange control messages (such as *scp*), the congestion window increases with every ACK, but without being used by the sender. Then, when

[1]Fast Recovery can reduce the burst size after a retransmission, if new segments can be sent in response to duplicate ACKs.

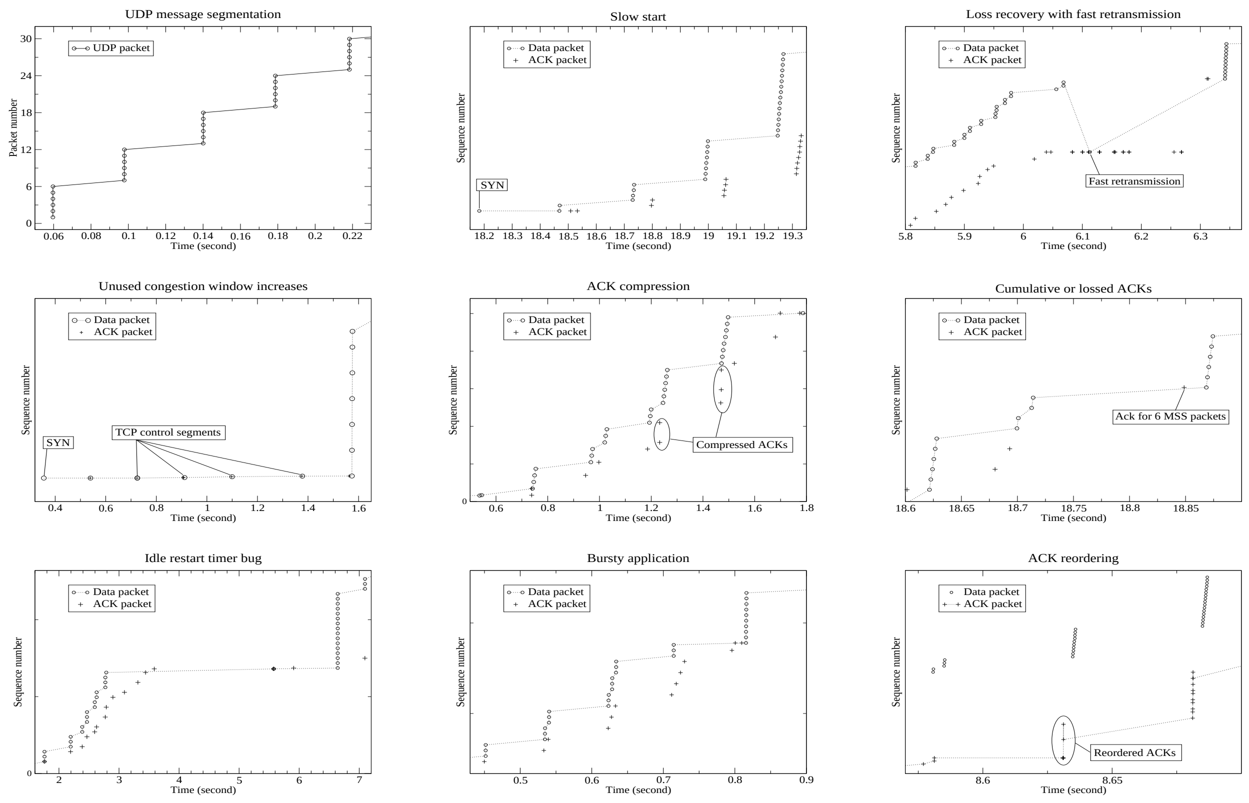

Figure 2: Major causes of source-layer bursts.

the sender is ready to transfer a large message or file, it can send a long burst to the network. Our example comes from the start of an *scp* session.

ACK compression. Queueing in the reverse path of a TCP flow, can cause the almost simultaneous arrival of successive ACKs at the sender. This can break TCP's self-clocking and cause long bursts [12].

Cumulative or lost ACKs. Sometimes the receiver generates an ACK for multiple received segments. Such a "super-cumulative" ACK can trigger a burst at the sender. The same effect can occur if one or more ACKs are lost. In that case, the first non-lost ACK will trigger a burst.

Idle restart timer bug. A TCP sender is recommended to use slow start and return to the initial congestion window after it has not sent anything for the duration of the Idle Restart timer (typically equal to RTO) [13]. Unfortunately, several operating systems do not support, or they do not implement correctly, this feature [14]. As a result, a TCP sender can send a long burst after an idle time period.

Bursty applications. Even if the Idle Restart timer is implemented correctly, it is possible that the application itself is bursty, meaning that it writes bytes to the TCP send-socket sporadically. If the time between bursts is sufficiently short so that Idle Restart is not activated, TCP's self-clocking breaks and TCP can send long bursts.

Packet reordering. Reordering of ACKs scrambles self-clocking and can trigger a burst at the sender. In the example shown, the out-of-order ACK acknowledges ten more segments, causing a large burst at the sender. Data segment reordering can also interrupt self-clocking and cause bursts [15].

3. PACKET BURSTS AND SCALING

In this section, we show the connection between source-level bursts and scaling, and identify the timescales in which such bursts create scaling behavior. Consider a source that generates a sequence of packet trains. A train consists of N packets, each of length L bytes. If the capacity of the source is C, the *dispersion* of each packet in the time axis is $\frac{L}{C}$, while the dispersion of the entire train is $\frac{NL}{C}$. Suppose that the interarrival time T_{off} between successive trains is exponentially distributed. We next show that this traffic process shows local scaling in the timescales between $\frac{L}{C}$ and $\frac{NL}{C}$.

Figure 3 shows the autocorrelation function and the energy plot for a synthetic trace that follows the previous packet train model. In this trace, we have that $\frac{L}{C}$=4ms, N=16, $\frac{NL}{C}$=64ms, and $E[T_{off}]$=2000ms. Consider first the discrete-time process of packet arrivals in successive non-overlapping intervals of length $\frac{L}{C}$; this time series takes the values 0 and 1. The autocorrelation $R(\tau)$ of this process, for $\tau = 0, 1, 2, \ldots,$ is positive when $\tau{<}N$, due to the strongly correlated interarrivals within a packet train. For larger lags $\tau{>}N$, $R(\tau)$ is almost zero because the correlations between packets of different trains are weak ($T_{off}{\gg}\frac{NL}{C}$), and the time interval between successive trains is exponentially distributed.

Observe now the energy plot of Figure 3. The linearly

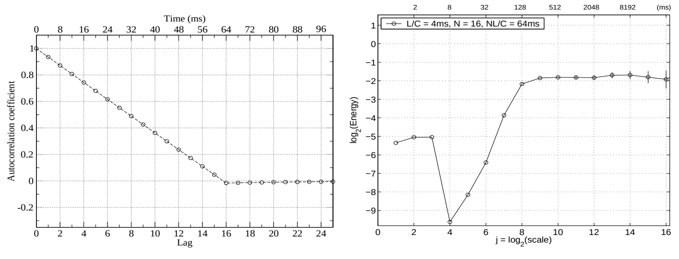

Figure 3: Autocorrelation and energy plot of the packet train traffic model.

increasing segment, between scales 4 and 8, represents local scaling in the timescales from 4ms ($\frac{L}{C}$) to 64ms ($\frac{NL}{C}$).[2] The strong positive correlations in the lags that correspond to the train duration ($\tau = 0, \ldots 15$) are reflected in the energy plot as local scaling in the corresponding timescales. The scaling exponent is almost zero in longer timescales (higher than 64ms), due to the exponential train interarrivals. Also note that the negative dip at scale-4, which corresponds to the packet spacing L/C, is due to the periodic arrival of a new packet every 4ms during packet trains.

The previous model may seem too artificial, as all packets appear in bursts. Source-level bursts can create scaling even if they occur less frequently however. Consider a source with two states: the "random" state and the "bursty" state. In the random state, the source generates exponential interarrivals with a mean of 100ms. In the bursty state, the source generates a single train of N=16 packets. The transition probability from the random state to the bursty state is 0.05, while the transition probability in the reverse direction is 1. Figure 4 shows the energy plot for such a source, when $\frac{L}{C}$=1ms. Notice the emerging scaling behavior between scales 2 and 6, which correspond to the timescales 1ms to 16ms. Even though the scaling exponent is not constant across these timescales, the range in which α is positive matches the extent of packet bursts, from $\frac{L}{C}$ to $\frac{NL}{C}$.

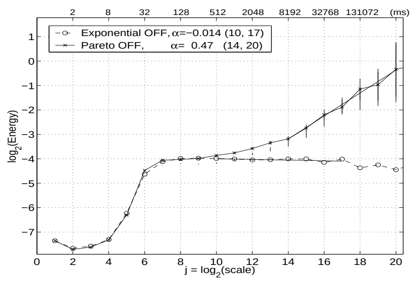

Figure 4: Bi-scaling behavior.

On the other hand, *source-level bursts do not contribute to the scaling behavior in long timescales.* To illustrate this point, consider the previous two-state model, but now suppose that the random state generates Pareto interarrivals with β=1.5. Figure 4 shows the resulting energy plot. The infinite variance of the Pareto interarrivals creates scaling at large timescales. The scaling exponent above scale 14 is estimated as α≈0.5, which is consistent with the shape parameter β=1.5. The scaling behavior in short timescales, on the other hand, is due to packet trains, and it remains roughly the same as in the case of exponential interarrivals. This is an example of *bi-scaling* behavior, i.e., different scaling exponent in short vs. long timescales, which is often seen in the energy plot of WAN traces [1].

[2]We remind the reader that the energy $\mathcal{E}_j$ is computed based on the variations of the traffic process at scale j-1.

4. EFFECTS OF PACKET BURSTS

In this section, we show the effect of packet bursts from individual flows in three different, but related, characteristics of aggregate IP traffic: scaling behavior in short timescales, marginal distribution, and queueing performance.

Burst identification. First, we describe how to identify packet bursts from individual flows in a trace of aggregate traffic. Consider a TCP flow f with source $\mathcal{S}_f$. A packet trace is collected at the output of a link $\mathcal{T}$ in f's path. In the appendix, we give a methodology for the estimation of the *pre-trace capacity* $\tilde{C}_f$ of flow f, i.e., the minimum link capacity along the path *between* $\mathcal{S}_f$ *and* $\mathcal{T}$. *A packet burst from flow f is defined as a sequence of packets from f that arrive at $\mathcal{T}$ with a rate that is roughly $\tilde{C}_f$.* It is important to note that we cannot determine whether these packets were sent from $\mathcal{S}_f$ back-to-back; we can only determine whether they arrive at $\mathcal{T}$ back-to-back. A source-level burst will be detected as a packet burst at $\mathcal{T}$, but not every packet burst at $\mathcal{T}$ will be a source-level burst. For this reason, this section refers to the effects of packet bursts, as opposed to source-level packet bursts, from individual flows.

In practice, the rate between successive packets in a burst may fluctuate above or below $\tilde{C}_f$ because of cross traffic queueing at links before $\mathcal{T}$. So, we require the following, less restrictive, condition: *a sequence of packets $P_f(i), \ldots P_f(i+j)$ from flow f is a packet burst of length j+1, if j>0 is the maximum positive number that satisfies the following two conditions:*

$$\frac{\sum_{k=i}^{i+j-1} S_f(k)}{\Delta_f(i,j)} > \frac{\tilde{C}_f}{a} \tag{3}$$

$$\frac{S_f(k)}{\Delta_f(k,k+1)} > \frac{\tilde{C}_f}{b} \text{ for all } k = i, \ldots j-1 \tag{4}$$

where $S_f(k)$ is the size of packet $P_f(k)$, and $\Delta_f(m,n)$ is the dispersion (time distance) between the start of packets $P_f(m)$ and $P_f(n)$ at $\mathcal{T}$ ($m < n$). If a>1 and b>1, these conditions require that the burst's average rate is larger than a fraction $1/a$ of $\tilde{C}_f$, and that the rate between successive packets in the burst is larger than a fraction $1/b$ of $\tilde{C}_f$.

To illustrate the frequency and length of packet bursts in real Internet traffic, Figure 5 shows the CDF of burst lengths for a trace from the OC-12 Merit link (MRA). This graph is derived based on TCP flows for which we have a pre-trace capacity estimate (about 83% of the TCP bytes in the trace). We show three curves for different parameters a and b. Note that the burst length distribution does not depend significantly on these two parameters; in the rest of this paper we use a=2 and b=4. Also note that 40% of the bytes in this trace are transferred in bursts of at least four packets, while 10% of the bytes are in bursts of more than twelve packets.

Burst removal. If we can identify source-level bursts, we can also modify a trace so that we remove those bursts. We use this technique to investigate how would the statistical

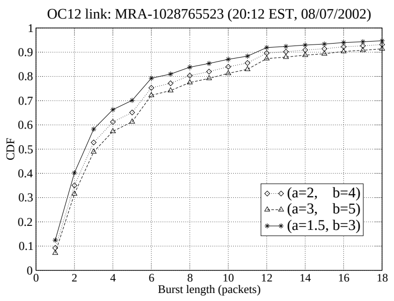

Figure 5: Parameter sensitivity of burst identification algorithm.

profile of the trace change, if individual flows did not generate packet bursts. Such a "semi-experimental" approach has been also followed in [9, 10].

Suppose that a burst $B_f(k)$ of flow f starts at time $t_f(k)$, while the first packet of f *after* this burst appears at time $t_f(k+)$. *We remove the burst* $B_f(k)$ *by artificially spacing the packets of the burst uniformly between* $t_f(k)$ *and* $t_f(k+)$. Note that the packets of flow f remain in their original order after respacing the bursts. Also note that this burst removal procedure cannot be performed on-line by a source or router, as it requires knowledge of $t_f(k+)$ when a burst starts. Also, it is not equivalent to flow shaping or pacing; these latter approaches would transmit the packets of a burst at a fixed rate. We refer to the resulting trace as *manipulated*, to distinguish it from the *original* trace.

Effect of bursts. Figure 6 compares the original and manipulated traces, from two OC-12 links, in terms of three aspects: energy plots and scaling behavior, tail distribution, and queueing performance. At the left, we show the energy plot of the traces in timescales that extend from less than a millisecond to a few seconds. Notice that both traces show clear bi-scaling behavior, with a scaling exponent of 0.35 for the MRA trace and 0.26 for the IND trace in short timescales (less than $25 - 200$ms). The scaling exponent at large timescales is 0.99 and 0.90, respectively, but its estimation is less accurate due to the short duration of these traces. The key observation, however, *is the difference between the original and manipulated traces: the scaling behavior in short timescales has been dramatically reduced, dropping the scaling exponent to almost zero.* This implies that removing packet bursts would lead to almost uncorrelated packet arrivals over a range of short timescales that extends up to 100-200ms. As expected, the scaling behavior in longer timescales has not been affected.

The middle graphs of Figure 6 show the tail distribution of the amount of bytes in non-overlapping 10ms intervals. The average of this distribution is 189KB for the MRA trace and 32KB for the IND trace. Note that the removal of packet bursts from individual flows reduces significantly the probability of having bursts in the aggregate trace. This was expected, as most bursts at the aggregate trace are due to individual flows, instead of different flows. The removal of bursts from the aggregate trace hints that the queueing performance would also improve significantly. Indeed, the right graphs of Figure 6 show the maximum queue size that would develop at a link that services the aggregate traffic, as we vary the link's capacity. The reduction in the maximum queue size, after we remove the source-level bursts, is significant especially in moderate utilizations, between 50% to 85%. This result agrees with the findings of [10].

5. SUMMARY AND FUTURE WORK

This paper focused on the causes and effects of packet bursts from individual flows in IP networks. We showed that such bursts can create scaling in short timescales, and increased queueing delays in traffic multiplexers. We identified several causes for source-level bursts, investigating the "microscopic" behavior of the UDP and TCP protocols. Some of these causes, such as the implementation of the Idle Restart timer, can be eliminated with appropriate changes in the TCP protocol or implementation. Some other causes, however, such as the segmentation of UDP messages in multiple IP packets, are more fundamental in nature and they may not be avoidable.

Even though we identified a plausible explanation for the presence of scaling in short timescales, we do not claim that source-level bursts are the only such explanation. In ongoing work, we investigate other important factors, such as the effect of TCP self-clocking. We also study the effect of per-flow shaping and TCP pacing on the correlation structure and marginal distributions of aggregate IP traffic.

6. REFERENCES

[1] A. Feldmann, A.C.Gilbert, and W.Willinger, "Data Networks as Cascades: Investigating the Multifractal Nature of the Internet WAN Traffic," in *Proceedings of ACM SIGCOMM*, 1998.

[2] R. Riedi, M. S. Crouse, V. Ribeiro, and R. G. Baraniuk, "A Multifractal Wavelet Model with Application to Network Traffic," *IEEE Transactions on Information Theory*, vol. 45, no. 3, pp. 992–1019, Apr. 1999.

[3] Z.-L. Zhang, V. Ribeiro, S. Moon, and C. Diot, "Small-Time Scaling behaviors of Internet backbone traffic: An Empirical Study," in *Proceedings of IEEE INFOCOM*, Apr. 2003.

[4] N. Hohn, D. Veitch, and P. Abry, "Cluster Processes, a Natural Language for Network Traffic," *IEEE Transactions on Signal Processing, special issue on "Signal Processing in Networking"*, 2003, Accepted for publication.

[5] P. Abry and D. Veitch, "Wavelet Analysis of Long-Range Dependent Traffic," *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 2–15, Jan. 1998.

[6] D. Veitch, "Code for the Estimation of Scaling Exponents," http://www.cubinlab.ee.mu.oz.au/∼darryl, July 2001.

[7] A. Feldmann, A.C.Gilbert, W.Willinger, and T. G. Kurtz, "The Changing Nature of Network Traffic: Scaling Phenomena," *ACM Computer Communication Review*, Apr. 1998.

[8] 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 *Proceedings of ACM SIGCOMM*, 1999.

[9] N. Hohn, D. Veitch, and P. Abry, "Does fractal scaling at the IP level depend on TCP flow arrival processes?," in *Proceedings Internet Measurement Workshop (IMW)*, Nov. 2002.

[10] A. Erramilli, O. Narayan, A. L. Neidhardt, and I. Saniee, "Performance Impacts of Multi-Scaling in Wide-Area TCP/IP Traffic," in *Proceedings of IEEE INFOCOM*, Apr. 2000.

[11] NLANR MOAT, "Passive Measurement and Analysis," http://pma.nlanr.net/PMA/, May 2003.

[12] J. C. Mogul, "Observing TCP dynamics in real networks," in *Proceedings of ACM SIGCOMM*, Aug. 1992.

[13] M. Allman, V. Paxson, and W. Stevens, *TCP Congestion Control*, Apr. 1999, IETF RFC 2581.

[14] A. Hughes, J. Touch, and J. Heidemann, *Issues in TCP Slow-Start Restart After Idle*, Mar. 1998, IETF Internet Draft, draft-ietf-tcpimpl-restart-00.txt (expired).

[15] J.C.R. Bennett, C. Partridge, and N. Shectman, "Packet Reordering is Not Pathological Network Behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789–798, Dec. 1999.

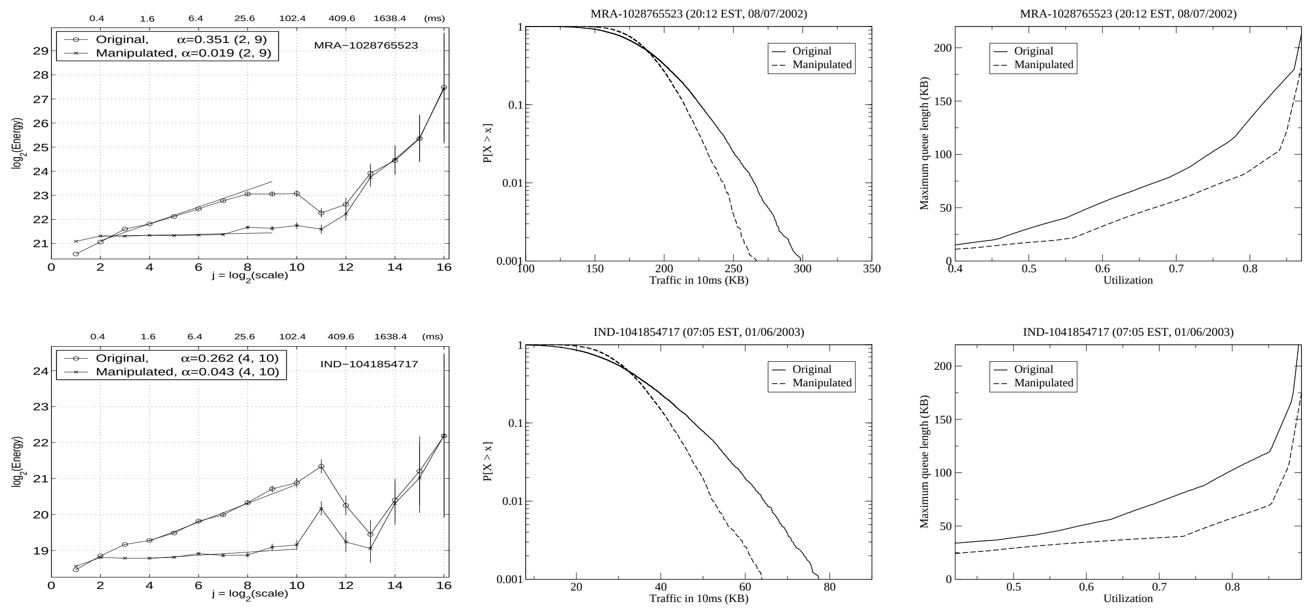

Figure 6: Effect of source-level bursts on scaling, tail distribution, and queueing performance.

[16] C. Dovrolis, P. Ramanathan, and D. Moore, "What do Packet Dispersion Techniques Measure?," in *Proceedings of IEEE INFOCOM*, Apr. 2001, pp. 905–914.

Appendix: Passive capacity estimation

The identification of packet bursts from a flow f at a trace point $\mathcal{T}$ requires an estimate of the *pre-trace capacity* $\tilde{C}_f$ of flow f. Here, we summarize a statistical methodology that estimates $\tilde{C}_f$ for TCP flows, using the timing of the flow's data packets. The methodology is based on the dispersion (time distance) of packet pairs [16].

For a TCP flow f, let $S_f(i)$ be the size of the i'th data packet, and $\Delta_f(i)$ be the dispersion measurement between data packets i and $i{+}1$. When packets i and $i{+}1$ are of the same size, we compute a bandwidth sample $b_i = S_f(i)/\Delta_f(i)$. Packets with different sizes traverse the network with different per-hop transmission latencies, and so they cannot be used with the packet pair technique [16]. Based on the delayed-ACK algorithm, TCP receivers typically acknowledge pairs of packets, forcing the sender to respond to every ACK with at least two back-to-back packets. So, we can estimate that roughly 50% of the data packets were sent back-to-back, and thus they can be used for capacity estimation. The rest of the packets were sent with a larger dispersion, and so they will give lower bandwidth measurements. Based on this insight, we sort the bandwidth samples of flow f, and then drop the lower 50% of them. To estimate the capacity of flow f, we employ a histogram-based method to identify the strongest mode among the remaining bandwidth samples; the center of the strongest mode gives the estimate $\tilde{C}_f$. The bin width that we use is $\omega = \frac{2(IRQ)}{K^{1/3}}$ (known as "Freedman-Diaconis rule"), where IRQ and K is the interquartile range and number, respectively, of bandwidth samples. We have verified this technique comparing its estimates with active measurements. The results are quite positive, but due to space constraints we do not include them in this paper.

Figure 7 shows the distribution of capacity estimates in two traces. Note that the CDF is plotted in terms of TCP bytes, rather than TCP flows. In the top graph, we see four dominant capacities at 1.5Mbps, 10Mbps, 40Mbps, and 100Mbps. These values correspond to the following common link bandwidths: T1, Ethernet, T3, and Fast Ethernet. The bottom graph shows the capacity distribution for the outbound direction of the ATM OC-3 link at University of Auckland, New Zealand. This link is rate-limited to 4.048Mbps at layer-2. We observe two modes, at 3.38Mbps and 3.58Mbps, at layer-3. The former mode corresponds to 576B IP packets, while the latter mode corresponds to 1500B IP packets. The difference is due to the overhead of AAL5 encapsulation, which depends on the IP packet size. We finally note that our capacity estimation methodology cannot produce an estimate for interactive flows, flows that consist only pure-ACKs, and flows that carry just a few data packets. We were able, however, to estimate the capacity for 83% of the TCP bytes in the MRA-1028765523 trace, 92% of the TCP bytes in the IND-1041854717 trace, and 82% of the TCP bytes in the Auckland trace.

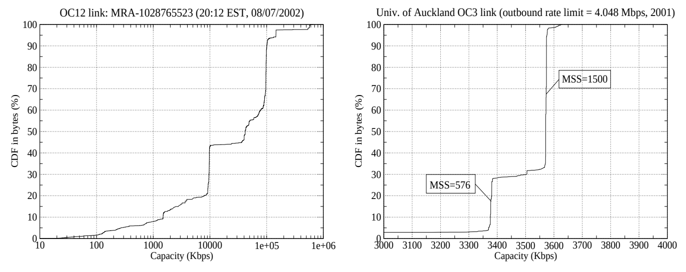

Figure 7: Capacity distribution in terms of bytes at two links.