LA-UR- 00-2895

Approved for public release; distribution is unlimited.

Title:	Network Traffic Characterization of TCP
Author(s):	Wu-chun Feng
Submitted to:	The 2000 Military Communications International Symposium (MILCOM 2000)

Los Alamos
NATIONAL LABORATORY

NETWORK TRAFFIC CHARACTERIZATION OF TCP [*]

Wu-chun Feng[†‡]
feng@lanl.gov

[†] Research & Development in Advanced Network Technology (RADIANT)
Los Alamos National Laboratory
Los Alamos, NM 87545

[‡] Department of Electrical & Computer Engineering
Purdue University
W. Lafayette, IN 47907

ABSTRACT

Networks from wireless to high-speed integrated services require support for the fluctuating and heterogeneous demands of end users. The ability to characterize the behavior of the resulting aggregate network traffic can provide insight into how traffic should be scheduled to make efficient use of the network, and yet still deliver expected quality-of-service to end users. These issues are of fundamental importance to the design of the Next-Generation Internet — from wireless communication to high-performance distributed computational grids such as NASA's Information Power Grid.

Many research efforts in network traffic characterization conclude that network traffic is self-similar (i.e., fractal or bursty), and thus not amenable to the statistical-multiplexing techniques currently found in the Internet. In particular, they claim that the heavy-tailed distributions of file size, packet interarrival, and transfer duration solely contribute to the self-similarity of aggregate network traffic. In contrast, we demonstrate that it is the TCP stack itself that induces much of the self-similar behavior even when aggregated application traffic should smooth out as more applications' traffic are multiplexed. Furthermore, if random early detection (RED) gateways/routers are used, we show that network performance degrades even further due to the extra burstiness induced by the gateway itself.

Keywords: *TCP, distributed computing, network traffic characterization, self-similar traffic.*

*This work was supported by the U.S. Dept. of Energy through Los Alamos National Laboratory contract W-7405-ENG-36.

INTRODUCTION

Although network-link capacity is steadily increasing, the number of Internet users is exponentially increasing. Thus, the ability of the Internet to handle the subsequent increase in traffic and to deal with widely-varying traffic characteristics becomes increasingly important. The most widely used protocol to reliably transfer information between end hosts is TCP. Over the years, it has evolved into many different implementations, e.g., Reno and Vegas, but each implementation still employs some form of an additive increase, multiplicative decrease (AIMD) algorithm to control congestion in the network.

The ability to characterize the behavior of aggregate network traffic can provide insight into how traffic should be scheduled to make efficient use of the network. These issues are fundamental to the design of the Next-Generation Internet (NGI) [9] and distributed computing environments such as NASA's Information Power Grid.

In recent years, studies in network traffic characterization have concluded that network traffic is self-similar in nature [3, 8]. That is, when traffic is aggregated over varying time scales, the aggregate traffic pattern remains bursty, regardless of the granularity of the time scale. Additional studies have concluded that the heavy-tailed distributions of file size, packet interarrival, and transfer duration fundamentally contribute to the self-similar nature of aggregate network traffic [10]. However, the proofs of the relationship between heavy-tailed distributions and self-similar traffic in [2, 10] ignore the involvement of the TCP congestion-control mechanism. Thus, while the heavy-tailed distributions of file size, packet interarrival, and transfer duration may contribute to self-similarity, many other factors which have not been investigated thoroughly.

Moreover, all the studies thus far have failed to isolate individual aspects of the end-to-end networking path in order to pinpoint the source of self-similarity; instead, various aspects have been intermingled and studied simultaneously, e.g., file-size distribution, packet interarrivals, transfer duration, TCP and its congestion-control mechanisms, and other non-TCP traffic.

A preliminary study by Feng et al. [1] addressed the above problems by focusing specifically on TCP and its congestion-control mechanisms. In this paper, we present a more comprehensive study on the adverse effects of TCP on application-generated traffic and characterize the behavior of this affected traffic via the coefficient of variation.

BACKGROUND

The Central Limit Theorem states that the summation of a large number of finite-mean, finite-variance, independent variables, e.g., Poisson, approaches a Gaussian random variable with less variability (or less "spread" or burstiness) than the original distribution(s). So, if each random variable were to represent traffic generated by a particular communication stream, then the sum of a large number of these streams represents aggregate network traffic with less variability, and thus less variation or spread in the required bandwidth, i.e, network traffic is less bursty or more smooth. Such aggregate traffic behavior enables statistical-multiplexing techniques to be very effective over the Internet. Unfortunately, although application-generated traffic streams may have finite means and variances and may be independent, we will demonstrate that TCP can modulate these streams in such a way that they are no longer independent. Hence, the thrust of this paper is to examine how TCP modulates application-generated traffic and how it affects the statistical-multiplexing techniques used in the Internet as well as distributed computing systems.

To measure the burstiness of aggregate TCP traffic, we use the *coefficient of variation (c.o.v.)* — the ratio of the standard deviation to the mean of the observed number of packets arriving at a gateway in each round-trip propagation delay. The *c.o.v.* gives a normalized value for the "spread" of a distribution and allows for the comparison of "spreads" over a varying number of communication streams.

Rather than use the Hurst parameter from self-similar modeling as is done in many studies of network traffic [3, 6, 7, 8, 10], we use *c.o.v.* because it better reflects the burstiness of the incoming traffic, and consequently, the effectiveness of statistical multiplexing over the Internet [1]. If the *c.o.v.* is small, the amount of traffic coming into the gateway in each round-trip time (RTT) will concentrate mostly around the mean, and therefore will yield better performance via statistical multiplexing.

SIMULATION STUDY

The goal of this simulation study is to understand the dynamics of how TCP modulates application-generated traffic. Understanding how TCP modulates traffic can have a profound impact on the *c.o.v.*, and hence, throughput and packet loss percentage of network traffic. This, in turn, directly affects the performance of distributed computing systems such as NASA's Information Power Grid.

NETWORK MODEL

To characterize the TCP modulation of traffic, we first generate application traffic according to a known distribution. We then compare the *c.o.v.* of this distribution to the *c.o.v.* of the traffic transmitted by TCP. We can then determine whether TCP modulates the traffic, and if it does, how it affects the shape (burstiness) of the traffic, and hence, the performance of the network.

Consider a client-server network with one server and M clients. Each client is linked to a common gateway with a full-duplex link with bandwidth μ_c and delay τ_c. A bottleneck bandwidth of μ_s and delay of τ_s connects the gateway to the server. Each client generates Poisson traffic, i.e., single packets are submitted to the TCP stack with exponentially distributed interpacket arrival times with mean $1/\lambda$. All the clients try to send the generated packets to the server through the common gateway and bottleneck link. The model is shown in Figure 1.

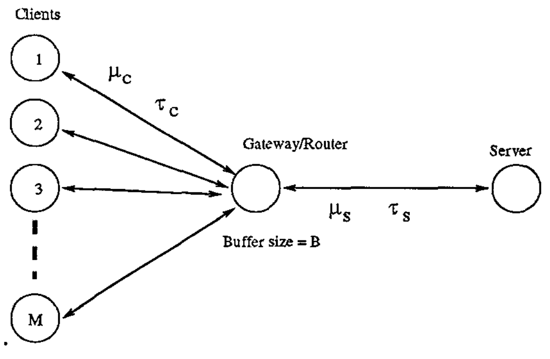

Figure 1: Network Model

In our *ns* [5] simulations, we vary the total traffic load offered by varying the number of clients M. We use UDP, TCP Reno (with delay acknowledgments both on and off), and TCP Vegas as the transport-layer protocols. We also test the effects of two queueing disciplines in the gateway, FIFO (First-In, First-Out) and RED, to see whether the queueing discipline has any effect on the burstiness generated by the TCP protocol stack. We calculate the *c.o.v.* of the aggregate traffic generated by the clients, based on the known distribution each client uses to generate its traffic, and compare it to the measured *c.o.v.* of the aggregate TCP modulated traffic as it arrives at the gateway. The parameters used in the simulation are shown in Table 1.

Parameters	Value
client link bandwidth (μ_c)	10 Mbps
client link delay (τ_c)	25 ms
bottleneck link bandwidth (μ_s)	50 Mbps
bottleneck link delay (τ_s)	25 ms
TCP max advertised window	20 packets
gateway buffer size (B)	50 packets
packet size	1500 bytes
average packet intergeneration time ($1/\lambda$)	0.01 s
total test time	200 s
TCP Vegas/α	1
TCP Vegas/β	3
TCP Vegas/γ	1
RED min_{th}	10 packets
RED max_{th}	40 packets

Table 1: Simulation Parameters.

TCP RENO VS. TCP VEGAS

Since the traffic generated by the application layers is Poisson (finite mean and finite variance), the *c.o.v.* of the number of packets received during one RTT for the unmodulated aggregate traffic is $1/\sqrt{(\lambda\tau)n}$ where n is the number of clients aggregated and $\tau = RTT = 2(\tau_c + \tau_s)$. Thus, the traffic generated from the application layer becomes smoother as the number of sources increases.

HOMOGENEOUS CASE

Here we examine how TCP modulates application-generated traffic when all the clients are running the same implementation of TCP.

Figure 2 shows that UDP does not adversely modulate traffic because the *c.o.v.* of aggregated UDP traffic is very close to that of the aggregated Poisson process. This result is not surprising since UDP transmits packets received from the application layer to the network without any flow/congestion control. For TCP, we divide the results into three cases.

1. Uncongested: the amount of traffic generated is much lower than the available bandwidth, i.e., the number of clients is less than 10.

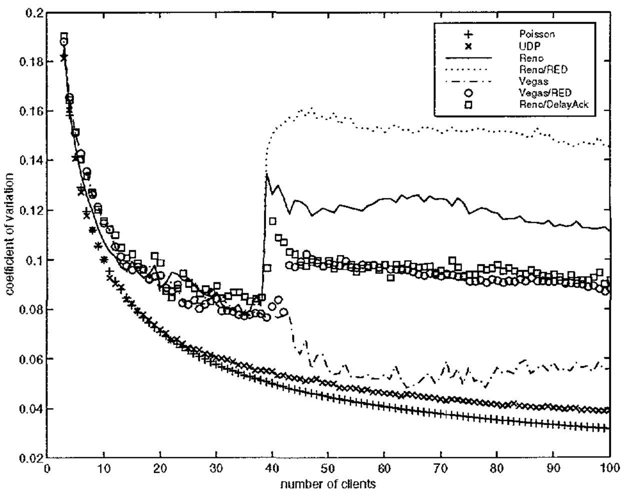

Figure 2: *c.o.v.* of Aggregated TCP Traffic.

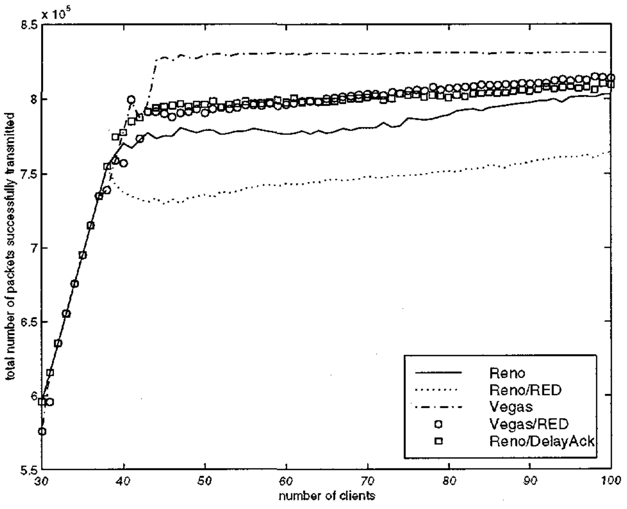

Figure 3: Throughput of Aggregated TCP Traffic.

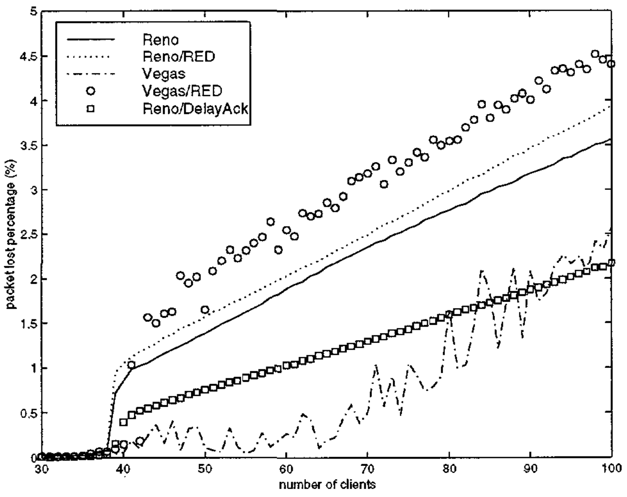

Figure 4: Packet-Loss Percentage of Aggregated TCP Traffic.

2. Moderately congested: the amount of traffic generated causes some, but not severe, congestion, i.e., the number of clients is between 10 and 38.

3. Heavily congested: the amount of traffic generated is higher than what the network can handle, i.e., the number of clients is greater than 38.

In the uncongested case, the traffic entering the gateway is very similar to the traffic that the clients generate. This result is due to the absence of congestion in the network, i.e., the congestion-control mechanism has not activated to control or modulate the application-generated traffic.

When the clients generate a moderate amount of traffic, and hence introduce intermittent congestion, the TCP congestion-control mechanism begins to modulate the application-generated traffic. We can see this effect in Figure 2 as the number of client connections varies from 10 to 38 — the TCP *c.o.v.* numbers are up to 50% higher than the aggregated Poisson, and hence indicate that the congestion-control mechanisms of TCP noticeably modulate traffic when the network is moderately congested; that is, TCP induces burstiness into the aggregate traffic stream. Because the network only experiences intermittent congestion, this induced burstiness is not strong enough to adversely impact throughput and packet loss, as shown in Figures 3 and 4. (Note: The number of clients starts at 30 for these figures because the different TCP implementations exhibit nearly identical behavior for less than 30 clients.)

Under heavy congestion, the *c.o.v.* increases sharply for all TCP implementations except TCP Vegas. The TCP Reno and TCP Reno/RED *c.o.v.* numbers are over 140% and 200% larger than the aggregated Poisson numbers, respectively. This result indicates that TCP Reno and TCP Reno/RED significantly modulate application-generated traffic (Poisson traffic) to be much more bursty. And unfortunately, this modulation is adverse enough to impact the throughput and packet loss percentage of TCP Reno and TCP Reno/RED, as shown in Figures 3 and 4. This leads us to believe that these TCP Reno implementations introduce a high level of dependency between the congestion-control mechanisms of each of the TCP streams.

HETEROGENEOUS CASE

From the results in the last section, the network performs better when all the TCP connections are running TCP Vegas. However, in wide-area distributed computing systems, it is unlikely that network users will switch to TCP Vegas all at once. So, this section examines how TCP Vegas connections perform in the presence of TCP Reno connections.

We use the same network model and parameters as before. However, here there are 20 TCP Reno clients generating background traffic. Figures 5 and 6 show the throughput and packet loss, respectively, for the clients which are added to the initial 20 TCP Reno clients. From our results, TCP Vegas still outperforms TCP Reno in this test. This coincides somewhat with the finding in [4] which states that TCP Vegas connections are favored when the gateway buffer size is small, as in our simulation (50 packets).

As a logical follow-up experiment, we test this setup with a larger gateway buffer size of 1500 packets with and without the presence of a RED gateway. We also use two different sets of RED parameters — RED1 where the min_{th} is 300 packets and the max_{th} is 1200 packets, and RED2 with min_{th} = 75 packets and max_{th} = 300 packets.

Figures 7 and 8 show the throughput and packet loss for this follow-up experiment. While the throughput in all cases are relatively close together, the relative differences in packet loss percentage are more pronounced (although the absolute differences are smaller due to the larger buffer).

CONCLUSION

We showed that the congestion-control mechanisms of TCP Reno and Vegas modulate traffic generated by the application layer. However, the congestion-control mechanism in Reno more significantly and adversely modulates traffic to be more bursty, which subsequently affects the performance of statistical multiplexing in the gateway; this modulation occurs for two primary reasons: (1) the rapid fluctuation of the congestion window sizes caused by the continual "additive increase / multiplicative decrease (or re-start slow start)" probing of the network state and (2) the dependency between the congestion-control decisions made by multiple TCP streams which increases as the number of streams increase [1]. As a result, TCP Reno traffic does not smooth out even when a large number of streams are aggregated. On the other hand, TCP Vegas, during congestion avoidance, does not modulate the traffic to be as bursty as TCP Reno; this translates to smoother aggregate network traffic, and hence better overall network performance.

We also demonstrated that TCP Vegas outperforms TCP Reno in both the homogeneous and heterogeneous cases although the differences in overall performance are noticeably smaller in the heterogeneous case. While the work of [4] may discourage researchers in high-performance computing and communications to switch from TCP Reno to TCP Vegas, our work indicates that TCP Vegas performs better than TCP Reno, particularly for smaller buffers (as found in many vendors' routers).

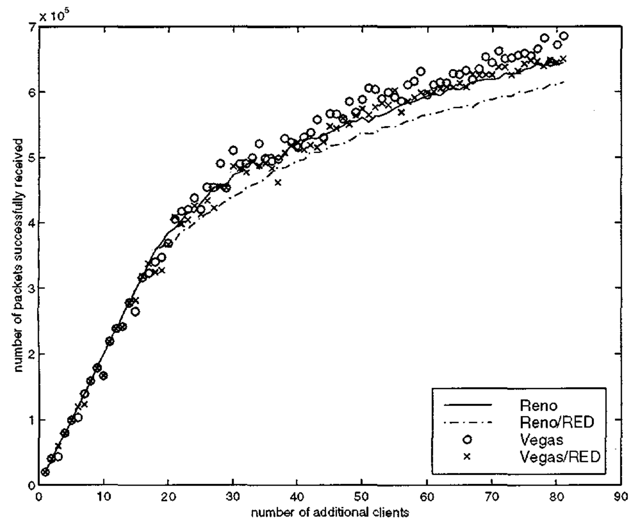

Figure 5: Throughput of Additional Clients (Buffer: 50).

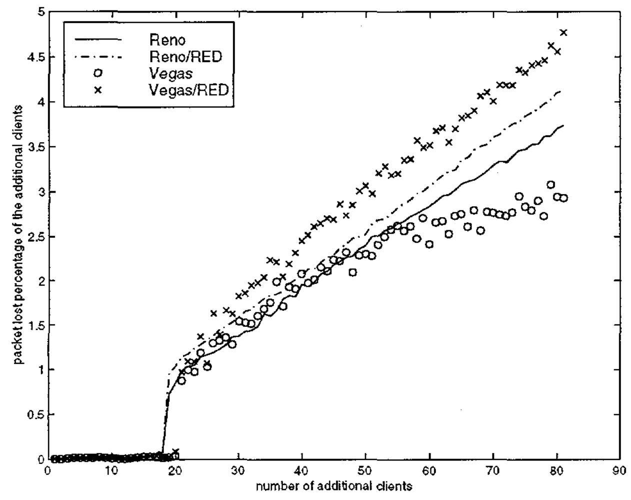

Figure 6: Packet-Loss Percentage of Additional Clients (Buffer: 50).

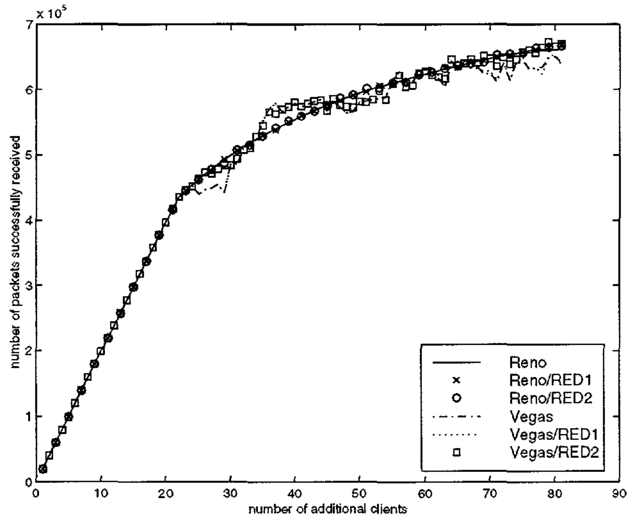

Figure 7: Throughput of Additional Clients (Buffer: 1500).

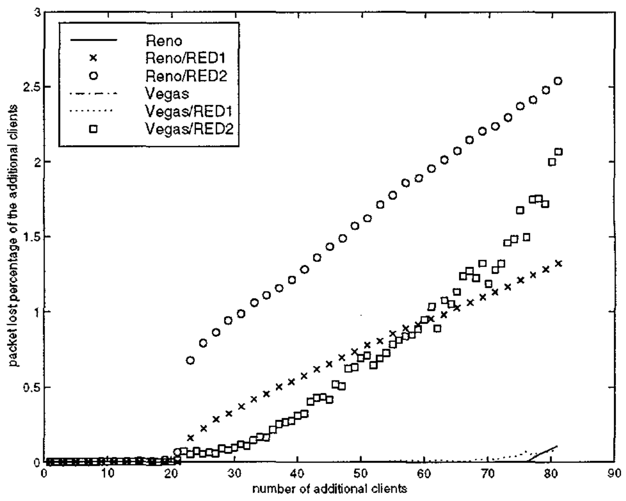

Figure 8: Packet-Loss Percentage of Additional Clients (Buffer: 1500).

References

[1] W. Feng, P. Tinnakornsrisuphap, and I. Philp. On the Burstiness of the TCP Congestion-Control Mechanism in a Distributed Computing System. In *20th IEEE International Conference on Distributed Systems*, April 2000.

[2] T. G. Kurtz. Limit Theorems for Workload Input Models. *Stochastic Networks: Theory and Applications*, pages 339–366, 1996.

[3] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Transaction on Networking*, 2(1):1–15, February 1994.

[4] J. Mo, R. J. La, V. Anantharam, and J. Walrand. Analysis and Comparison of TCP Reno and Vegas. In *Proceedings of INFOCOM'99*, March 1999.

[5] ns. UCB/LBNL/VINT Network Simulator. *http://www-mash.cs.berkeley.edu/ns.*

[6] K. Park, G. Kim, and M. Crovella. On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic. In *Proceedings of the 4th International Conference on Network Protocols*, October 1996.

[7] K. Park, G. Kim, and M. Crovella. On the Effect of Traffic Self-Similarty on Network Performance. In *Proceedings of the SPIE International Conference on Performance and Control of Network Systems*, 1997.

[8] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transaction on Networking*, 3(3):226–244, June 1995.

[9] S. Shenker. Fundamental Design Issues for the Future Internet. *IEEE Journal of Selected Areas in Communications*, 13(7):1176–1187, 1995.

[10] W. Willinger, V. Paxson, and M. Taqqu. Self-Similarity and Heavy Tails: Structural Modeling of Network Traffic. *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, pages 27–53, 1998.