Queue Management Algorithms and Network Traffic Self-Similarity

B. Sikdar, K. Chandrayana, K. S. Vastola and S. Kalyanaraman
Department of ECSE, Rensselaer Polytechnic Institute
Troy, NY 12180 USA

Abstract—**The self-similarity of network traffic has been established in a variety of environments and it is well known that self-similar traffic can lead to larger queueing delays, higher drop rates and extended periods of congestion. In this paper, we investigate the impact of various buffer management algorithms on the self-similarity of network traffic. In this paper we investigate the impact of active and passive queue management policies used at the routers on the self-similarity of TCP traffic. We also propose a modification to the RED algorithm, aimed at reducing the timeouts and exponential backoffs in TCP flows, and show that it can lead to significant reductions in the traffic self-similarity under a wide range of network conditions, as compared to the currently implemented active and passive buffer management policies. We also show that though our techniques are aimed at TCP related causes, it is also effective in reducing the degree of self-similarity in traffic even when application and user level causes are also present, as long as TCP is used as the underlying transport protocol.**

I. Introduction

The self-similar nature of network traffic has been established in a variety of network environments and its causes have been traced to various factors. These causes range from source and application level behavior of traffic sources [1], [19] to transport layer [4], [6], [8], [11], [15], [17], [18] as well as human factors [1]. In this paper we investigate the impact of active and passive buffer management policies on the protocol (TCP) related causes of self-similarity. It is also well known that the self-similar and multifractal nature of traffic can lead to a number of undesirable effects like high buffer overflow rates, large delays and persistent periods of congestion [2], [3], [12], and the severity of these conditions is directly proportional to the degree of self-similarity or the Hurst parameter. In this paper, we present a modified version of the Random Early detection (RED) [7] algorithm which can be used to reduce the degree of self-similarity in TCP traffic.

Our focus in this paper is to investigate the impact of buffer management policies on the self-similarity of network traffic. In particular, we investigate how buffer management policies can affect the TCP related causes of self-similarity. TCP related causes of self-similarity have been investigated in [4], [6], [8], [11], [15], [17], [18] and it is shown in [6], [8], [15] that TCP's reaction to losses through timeouts and exponential backoffs can contribute to the self-similarity of traffic. We evaluate the impact of current active and passive buffer management protocols on the timeouts and exponential backoffs. Based on these results, we develop a variation of the Random Early Detection (RED) queue management algorithm and show that it performs better than both passive taildrop as well as RED queues. The improvement in performance is both in terms of reduced self-similarity as well as throughput, loss rates and percentage of timeouts in the TCP flows. Our technique also works even when non TCP related causes of self-similarity are present, as long as TCP is used as the transport protocol. To support this claim, we show using simulations that even with the presence of web-traffic with heavy-tailed file sizes and typical user behavior patterns, the self-similarity of the traffic is eliminated at low loads and significantly reduced at moderate and high loads if we implement our proposed schemes.

The rest of the paper is organized as follows. Section II defines some of the terminology while Section III investigates the impact of three buffer management schemes on traffic self-similarity: taildrop, RED and a proposed modified RED algorithm. Finally, Section IV presents the results and V presents the discussions and concluding remarks.

Supported in parts by DARPA contract F30602-00-2-0537, DoD MURI contract F49620-97-1-0382 and NSF grant ANI 9806660.

II. Definitions

In this paper, we use the following definition of self-similarity: Let $X = (X_t : t = 0, 1, 2, \cdots)$ be a covariance stationary process with mean μ, variance σ^2 and autocorrelation function $r(k), k \geq 0$. For each $m = 1, 2, \cdots$, let $X^{(m)} = (X_k^{(m)} : k = 1, 2, \cdots)$ denote the new covariance stationary time series (with corresponding autocorrelation function $r^{(m)}$) given by: $X^{(m)} = 1/m(X_{km-m+1} + \cdots + X_{km}), k \geq 1$. The process X is called *(exactly) second-order self-similar* with self-similarity parameter $H = 1 - \beta/2$ if for all $m = 1, 2, \cdots$, var$(X^{(m)}) = \sigma^2 m^{-\beta}$ and

$$r^{(m)}(k) = r(k), k \geq 0 \tag{1}$$

and X is called *(asymptotically) second order self-similar* with self-similarity parameter $H = 1 - \beta/2$ if for all k large enough,

$$r^{(m)}(k) \to r(k), \text{ as } m \to \infty \tag{2}$$

III. Buffer Management Policies and Self-Similarity

In this section we investigate the effect of different of buffer management policies on the self-similarity of traffic passing through it. We consider both passive and active queue management algorithms as represented by taildrop

and RED queues respectively. We also propose a change to the current RED algorithm which results in lower degrees of self-similarity. We first investigate how the loss patterns resulting from these policies affect the behavior of TCP flows in terms of the number of timeouts before presenting the results on their impact on the self-similarity of network traffic.

A. Taildrop Queues

Taildrop queues are currently the most widely implemented queueing mechanisms in routers in the Internet [10]. The first-in-first-out (FIFO) policy of taildrop queues, coupled with the bursty nature of TCP traffic implies that the packet drops from a taildrop queue become correlated and multiple packets can get dropped from the same window.

To model the effect of taildrop queues on the probability of timeouts in TCP flows, we use the correlated loss model used in [10] and [13]. In this model, a packet in a window is lost independently of losses in other rounds. However, losses within a window are correlated and all packets following the first packet to be lost in window are also assumed to be lost. As noted in [10] and [13], this model is quite realistic for taildrop queues with TCP traffic given the bursty nature of TCP sources with back to back packet transmission. With correlated losses, the probability that an arbitrary packet loss in a TCP flow with $cwnd = w$ and a loss rate of p leads to a timeout is given by [10], [13]

$$Q(w) = \begin{cases} 1 & \text{for } 1 \leq w \leq 3 \\ 1 - \frac{p(1-p)^{2w-1}}{1-(1-p)^w} & \text{for } 4 \leq w \leq 8 \\ 1 - \frac{p(2-p)(1-p)^{2w-2}}{1-(1-p)^w} & \text{for } 9 \leq w \leq W_{max} \end{cases} \tag{3}$$

where W_{max} is the maximum allowable window size.

B. RED Queues

RED is an active queue management algorithm which randomly drops packets before a queue becomes full, so that end nodes can respond to congestion before buffers overflow and was proposed in [7]. RED probabilistically drops packets even before the queue is full based on a weighted average of the queue length. An RED queue maintains two thresholds which determine the rate of packet drops: a lower threshold, min_{th}, and an upper threshold, max_{th}. For each packet arrival at the queue, based on the current average queue length k, the drop drop probability for the packet $d(k)$ is calculated and is given by

$$d(k) = \begin{cases} 0 & \text{for } k < min_{th} \\ \frac{k - minth}{max_{th} - min_{th}} max_p & \text{for } min_{th} < k < max_{th} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

where max_p is a control variable denoting the maximum drop probability. The reader is referred to [7] for the detailed RED algorithm.

For a RED queue the packet drop pattern is closely modeled by an independent loss model as noted in [14] and the references therein. When the average queue length is between min_{th} and max_{th}, as it should be when the control parameters are properly selected, the incoming packets are dropped randomly conforming to the independent loss model. In the independent loss model, losses in a window are assumed to be independent of losses in other rounds. Additionally, losses in a given window are also assumed to be independent of each other. From [9] and [14], with the independent loss model the probability that an arbitrary packet drop leads to a timeout in a TCP flow with a window of w packets experiencing a loss rate of p is given by

$$Q(w) = \begin{cases} 1 & \text{for } 1 \leq w \leq 3 \\ 1 - \frac{wp(1-p)^w}{1-(1-p)^w} & \text{for } 4 \leq w \leq 8 \\ 1 - \frac{wp(1-p)^w}{1-(1-p)^w} & \text{for } 9 \leq w \leq W_{max} \\ - \frac{w(w-1)p^2(1-p)^{w+n-2}}{1-(1-p)^w} & \end{cases} \tag{5}$$

Note that with independent losses, the probability of timeouts is much lesser than that predicted by the correlated model. This leads to the intuition that the self-similarity in traffic passing through RED queues will be much lesser than that through taildrop queues. We verify this intuition through simulations later in this section.

C. Modified RED Queues

The independent loss model for the packet drop in a RED queue is very accurate for the cases when the average queue length stays between max_{th} and min_{th}. However, if the offered load is so high that the average queue length becomes close to max_{th}, RED fails to perform better than taildrop queues and the traffic passing through the RED bottleneck can have higher values of H as compared to taildrop queues. This is due to the fact that when the average queue length becomes greater than max_{th}, RED drops each packet with probability 1. This leads to multiple packet drop from the same window, resulting in timeouts.

To deal with this situation, we propose a change to RED's dropping policy. The new algorithm for packet dropping is shown in Algorithm 1. The idea is *not* to drop any two consecutive packets which arrive at the queue, unless of course if the queue is full. Since TCP generally sends back to back packets, ensuring that no two consecutive packets are dropped will reduce the probability that multiple packets from the same window are dropped, thereby reducing the occurrence of timeouts. Note that in the algorithm, we do not change the unconditional dropping probability $d(k)$ as calculated for the original RED algorithm.

To calculate the probability that any arbitrary packet arriving when the average queue size is k is dropped, $d'(k)$, we first refer to Figure 1. The three states, denoted by $\{i, j\}$ with $i, j \in 0, 1$ and $i = j \neq 1$, represent the possible conditions the queue can be in depending on whether the current or the previous packet was dropped or not. The global balance equations for the transition probabilities from the three

Algorithm 1 Modified Dropping algorithm of RED

```
last_drop_flag ⇐ 0
for Each Packet Arrival do
  if last_drop_flag = 1 then
    last_drop_flag = 0;
    goto enqueue;
  else if min_th < avg < max_th then
    with probability d(k), drop the packet
    if packet is dropped then
      last_drop_flag = 1;
    end if
  else if max_th < avg then
    Drop the packet;
    last_drop_flag = 1;
  else
    goto enqueue;
  end if
end for
```

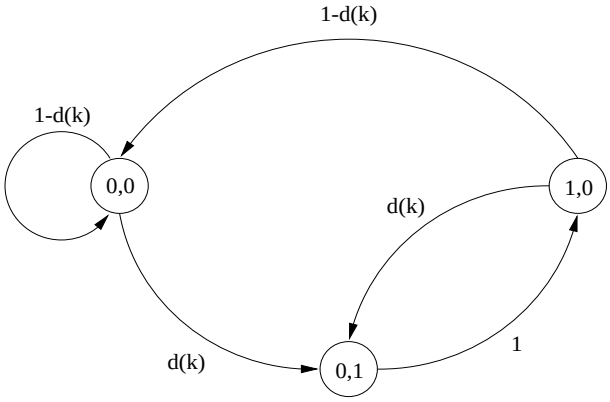

Fig. 1. Packet drop probabilities with the modified RED algorithm

states can be written as

$$\begin{aligned} P_{0,0}d(k) &= P_{1,0}(1-d(k)) \\ P_{0,1} &= P_{0,0}d(k) + P_{1,0}d(k) \\ P_{1,0} &= P_{0,1} \end{aligned} \quad (6)$$

where $P_{i,j}$ denotes the steady state probability of being in state i, j. These steady-state are then given by

$$P_{0,0} = \frac{1-d(k)}{1+d(k)} \quad P_{0,1} = \frac{d(k)}{1+d(k)} \quad P_{1,0} = \frac{d(k)}{1+d(k)}$$

In this modified RED algorithm, the probability that any arbitrary arriving packet is dropped when the average queue length is k, $0 \leq k < qlen$, is thus

$$d'(k) = P_{0,0}d(k) + P_{1,0}d(k) = \frac{d(k)}{1+d(k)} \quad (7)$$

The analysis above assumes that successive packets see the same weighted average queue length and thus the same $d(k)$.

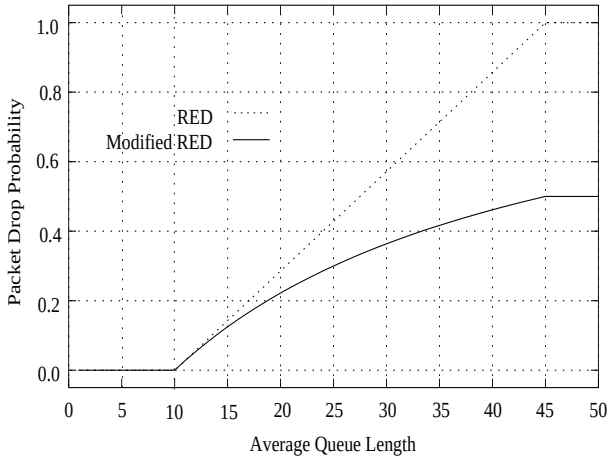

Fig. 2. Comparison of packet drop probabilities in the RED and modified RED buffer management policies.

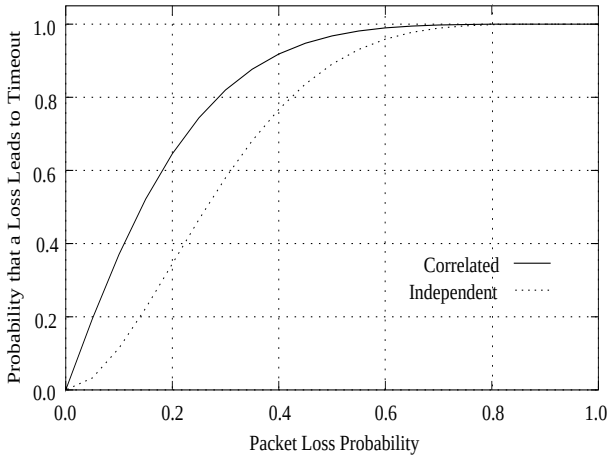

Fig. 3. Comparison of timeout probabilities for TCP flows with a window of 5 with correlated and independent loss models.

We note that if max_p is small as is suggested in literature on RED parameter configuring, then this modification does not significantly affect the drop rates while the average queue length is less than max_{th}. However, when the average queue length exceeds max_{th} but is less than $qlen$, the packet drop probability becomes 0.5 as compared to 1 in RED. This is a sufficiently high drop rate to force TCP sources to reduce their transmission rates but without inducing a lot of timeouts.

The difference in the packet drop rates from a RED and our modified RED queue is shown in Figure 2. The results are for a buffer size of $K = 50$, $min_{th} = 10$, $max_{th} = 45$, $max_p = 1.0$ and $w_q = 0.002$. Note that when the average queue size becomes greater than max_{th}, the packet loss rate for the modified RED queue is only 0.5 as compared to 1 for RED. Figure 3 gives a feel of the difference in the timeout probabilities in taildrop and RED queues where we plot the probability that a loss in a TCP flow with a congestion window of 5 leads to a timeout in the case of correlated and

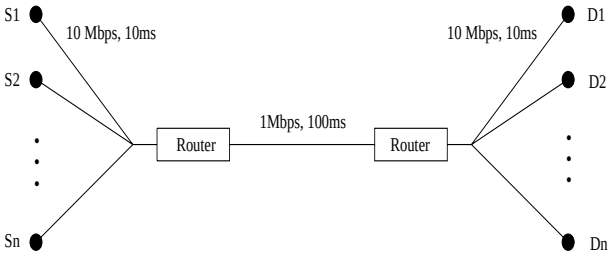

Fig. 4. Topology of the network for the validation tests.

independent losses. We see that with correlated losses, the probability of timeouts are much higher for the same loss rates. Also, since the packet drop probabilities for the modified RED queue are smaller that those for a RED queue with the same parameters for a given value of the average queue length, the corresponding probability of timeouts in the modified RED queue would also be smaller.

IV. Results

We now compare the self-similarity of the traffic with the three buffer management policies. These results were generated by simulations using the simulator *ns*. While the taildrop and RED queues are already available in the *ns* package, we implemented the modified RED algorithm in *ns*. The topology used for the simulations is shown in Figure 4. The simulations for each queueing policy is again broken in two parts: the presence or absence of web traffic. The web traffic is introduced as background traffic for more realistic wide area networking scenarios and to get a feel of the effectiveness of the buffer management techniques in reducing the self-similarity introduced by application level and human causes [1].

In the simulations with absence of web traffic, all the sources correspond to very long TCP Reno sources which are active for the entire duration of the simulation. In the simulations with web traffic, a subset of the sources and destinations act as web traffic clients and servers. The column "Configuration" in Tables I and II enumerates the number of log flows and web sessions in each simulation. The web traffic was generated according to the methodology and parameter settings described in [5].

For the simulations, the buffer size of the taildrop as well as the RED and modified RED queues was kept at 100 packets. For the RED and modified RED queues, the other parameters were $min_{th} = 30$, $max_{th} = 90$, $max_p = 0.1$ and $w_q = 0.002$. All the simulations were conducted for a "simulated" time of 3600.0 seconds. The self-similarity results corresponds to the aggregated traffic trace at the bottleneck while the throughput corresponds to the statistics of the long TCP flows. For estimating the Hurst parameters, we used three of the widely used methods [16]: the absolute value method, R/S statistics method and the periodogram method. The values of H obtained from each of the three methods are very close and lie within ± 0.03 of each other.

In Tables I and II we compare the values of H for the three queueing disciplines considered, for simulations with and without web traffic respectively. For the case without web traffic (Table I), we note that as expected, the taildrop queue always performs worse that the other two disciplines in terms of the degree of self-similarity. Also, the modified RED queue performs better than the other two queueing disciplines. We note that for the case with 60 flows the RED and the modified queue perform almost the same. The reason behind this is that with such a large number of flows, the average queue length is approximately max_{th}. This leads to *consistent* multiple drops from the same window and thus the flows go into timeouts approximately equally often. However, this particular case corresponds to the situation where the RED and the modified RED queue parameters are not properly tuned. In cases where the average queue length stays between max_{th} and min_{th} (30, 40 and 50 flows) we see that modified RED performs better than simple RED.

For the results in the simulations with web traffic (Table II) we see a slightly different trend in the results. We see that for lower loads the Hurst parameters corresponding to the taildrop queue are higher than both the RED and the modified RED algorithms. As the load on the network increases, the Hurst parameter of the traffic in the RED queue becomes more than the other two scheduling disciplines. However, we note that for all cases, the modified RED algorithm performs better than the others or equals the best performance. The difference in the trend in the traffic self-similarities in this case is due to the introduction of heavy tails through the web traffic.

In Tables I and II we also compare the throughputs of the long TCP flows in the simulation scenarios. The improvement in the throughput with RED queues over taildrop queues is around 1-3% for the cases without web traffic and 5-11% in the presence of web traffic. In the absence of web traffic, the throughput of the RED and modified RED are almost identical. However in the presence of web traffic, the throughput of the modified RED generally increases and the increase is around 5%.

V. Summary

In this paper we explored the effect of buffer management policies on the self-similarity of TCP traffic. Researchers have shown that the timeouts and exponential backoffs in TCP flows in the presence of losses can contribute to the self-similarity of network traffic. We investigated the impact of taildrop and RED queues on the number of timeouts and degree of self-similarity of TCP flows. We observed that for moderate and low loads, both the incidence of timeouts and Hurst parameter are lower with RED queues. However, this is no longer true when the network load increases where RED's performance resembles that of taildrop queues. To address this issue, we proposed a modification to RED which ensures that no two consecutive pack-

Configuration	Taildrop		RED		Modified RED	
	Throughput	H	Throughput	H	Throughput	H
30 Long	34738.67	0.61	35105.33	0.50	34952.59	0.50
40 Long	26514.67	0.75	26780.50	0.50	26750.33	0.50
50 Long	21545.11	0.80	22086.89	0.55	21819.80	0.50
60 Long	18253.89	0.82	18877.33	0.69	18824.07	0.70

TABLE I

QUEUES WITHOUT WEB TRAFFIC: THROUGHPUT (IN BITS/SEC) AND HURST PARAMETERS FOR THE THREE BUFFER MANAGEMENT POLICIES.

Configuration	Taildrop		RED		Modified RED	
	Throughput	H	Throughput	H	Throughput	H
10 Long, 5 web	63988.44	0.58	72982.00	0.50	63055.77	0.50
15 Long, 5 web	42099.70	0.67	44785.19	0.60	47422.37	0.57
10 Long, 10 web	33005.78	0.74	37165.11	0.76	34517.33	0.75
15 Long, 10 web	22318.52	0.77	23449.48	0.84	25204.88	0.77
20 Long, 10 web	17508.89	0.80	19952.00	0.91	21275.88	0.80

TABLE II

QUEUES WITH WEB TRAFFIC: THROUGHPUT (IN BITS/SEC) AND HURST PARAMETERS FOR THE THREE BUFFER MANAGEMENT POLICIES.

ets are dropped thereby reducing correlated losses and timeouts. Our results show that while the RED queue leads to lower degrees of self-similarity when only long TCP flows are present, in the presence of web-traffic, taildrop queues can lead to lower degrees of self-similarity at high loads. However, the modified RED queue consistently gives the lowest degree of self-similarity for all these scenarios. Also, the modified RED algorithm is able to eliminate the self-similarity of traffic at low and moderate loads for the topology and traffic sources considered and only at high loads does the traffic become self-similar.

While we considered only the causes of self-similarity from the TCP point of view, our solutions are also effective against other causes of self-similarity like session interarrival times and heavy tailed distributions in the file sizes, as long as the traffic is carried using TCP. We carried out simulations where background web traffic generated according to empirical distributions was also present and showed that the modified RED algorithm can reduce the degree of self-similarity in those cases also.

REFERENCES

[1] M. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: Evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835-846, Dec 1997.

[2] A. Erramilli, O. Narayan, A. Neidhardt and I. Sainee, "Performance impacts of multi-scaling in wide area TCP/IP traffic," *Proceedings of IEEE INFOCOM*, pp. 352-259, Tel Aviv, Israel, March 2000.

[3] A. Erramilli, O. Narayan and W. Willinger, "Experimental queueing analysis with long-range dependent packet traffic," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 209-223, April 1996.

[4] A. Feldmann, A. C. Gilbert and W. Willinger, "Data networks as cascades: Investigating the multifractal nature of Internet WAN traffic," *Computer Communications Review*, vol. 28, no. 4, pp. 42-58, 1998.

[5] 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," *Proceeding of ACM SIGCOMM*, Boston, MA, August 1999.

[6] D. R. Figueiredo, B. Liu, V. Misra and D. Towsley, "On the autocorrelation structure of TCP traffic," Technical Report TR 00-55, University of Massachusetts, Computer Science Department, Amherst, MA, 2000.

[7] S. Floyd and V. Jacobson, "Random early detection gateways for TCP congestion avoidance," *IEEE/ACM Transactions on Networking* vol. 1, no. 4, pp. 397-413, August 1993.

[8] L. Guo, M. Crovella and I. Matta, "How does TCP generate pseudo-self-similarity?," *Proceedings of MASCOTS*, Cincinnati, OH, August 2001.

[9] A. Kumar, "Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 485-498, August 1998.

[10] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133-145, April 2000.

[11] K. Park, G. Kim, and M. Crovella, "On the relationship between file sizes, transport protocols, and self-similar network traffic," *Proceedings of ICNP*, pp. 171-180, Columbus, OH, October 1996.

[12] V. Paxson and S. Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226-244, June 1995.

[13] B. Sikdar, S. Kalyanaraman and K. S. Vastola, "An integrated model for the latency and steady state throughput of TCP connections," *Performance Evaluation*, vol. 46, no. 2-3, pp. 139-154, September 2001.

[14] B. Sikdar, S. Kalyanaraman and K. S. Vastola, "TCP Reno with random losses: Latency, throughput and sensitivity analysis," *Proceedings of IEEE IPCCC*, pp. 188-195, Phoenix, AZ, April 2001.

[15] B. Sikdar and K. S. Vastola, "The effect of TCP on the self-similarity of network traffic," *Proceedings of the 35th Conference on Information Sciences and Systems*, Baltimore, MD, March 2001.

[16] M. S. Taqqu and V. Teverovsky, "On estimating long-range dependence in finite and infinite variance series," *A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, R. J. Adler, R. E. Feldman and M. S. Taqqu editors, pp. 177-217, Birkhauser, Boston, 1998.

[17] A. Veres and M. Boda, "The chaotic nature of TCP congestion control," *Proceedings of IEEE INFOCOM*, pp. 1715-1723, Tel-Aviv, Israel, 2000.

[18] A. Veres, Z. Kenesi, S. Molnár and G. Vattay, "On the propagation of long-range dependence in the Internet," *Proceedings of ACM SIGCOMM*, pp. 243-254, Stockholm, Sweden, September 2000.

[19] 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*, vol. 5, no. 1, pp. 71-86, 1997.