
MELİKE EROL¹, SEMA OKTUĞ¹, TAYFUN AKGÜL²
Dept. of Computer Engineering¹, Dept. of Electronics and Communications²
Istanbul Technical University
ITU, Faculty of Electrical Engineering, 34469, Maslak, Istanbul
TURKEY

Abstract: - It is known that aggregate network traffic is long-range dependent and self-similar. Long-range dependent processes, unlike the theoretically popular Poisson-like short-range dependent processes, affect the queuing performance of the network drastically. The main causes of self-similarity are considered as the file sizes on the internet, human think-time and the transport protocol. The first two are statistics relying on observations and are rather difficult to intervene, however the latter can be modified to produce less self-similar processes. TCP is the dominant transport protocol and the adaptive nature of TCP congestion control mechanisms have been declared as a possible self-similarity generating factor. Moreover, the interaction between TCP congestion control mechanism and different buffer management policies affect the degree of self-similarity in different ways. Although queue management policies are currently hot research topics, their effects on self-similarity have been left as an open issue. To our best knowledge, there is only one recent work in literature which investigates the impacts of two of the queue management policies, droptail and RED on the self-similarity [1]. In our work, we include BLUE and hence provide a better comparison of queue management policies. We extend our investigation by using signal processing tools to understand the behavior of large number of TCP flows.

Key-Words: - Aggregate network traffic, TCP, self-similarity, Hurst estimation.

1 Introduction

Internet traffic has long ago proven to show self-similar properties and long-range dependence (LRD). These properties affect the queuing performance at gateways severely. The gateways or bottleneck routers are the critical points of the networks. Emerging high-speed transmission media has left them as the major delay introducing elements. Clearly, their limited buffers should be used wisely. Active Queue Management (AQM) techniques are deployed for this reason. In order to manage buffers, one may need to know what these suffering buffers are subject to. Today, it is known that queues are fed by long-range dependent traffic on the networks.

The pioneer paper by Leland et al. [2] introducing self-similarity of Ethernet traffic was published a decade ago. Their findings motivated researchers to focus on the self-similarity of network traffic. Soon after, self-similarity of Ethernet traffic is adapted to wide area traffic [3]. The work of Crovella et al. [4] show that the self-similarity can be explained in terms of the file system characteristics and the user behavior. Later on, self-similarity is attributed to underlying network architecture together with the congestion control algorithms of TCP-like closed-loop protocols [5]. A complementary work, on the congestion control mechanisms of TCP is given in [6] where TCP’s timeout and collusion avoidance phases are stated to be dominating the shape of TCP traffic.

The correlated nature of TCP traffic combined with AQM mechanisms affect the self-similarity in a different way than TCP applied to droptail. However, AQM algorithms can be designed to overcome the negative effects of the LRD traffic or even going one step further they can play a role in reducing the self-similarity of the input traffic [1].

The degree of self-similarity and LRD can be measured by the Hurst parameter. In this paper, $H$ is estimated by using aggregated variance, absolute values and R/S methods and a nonparametric power spectral analysis method is used to validate the estimates.

In literature, only the work of Sikdar et al. [1] is present for investigating the effects of droptail, RED and their modified version of RED on the self-similarity of the aggregate input traffic. Droptail, RED and BLUE have been compared from many different aspects but not for their performance related to self-similarity and the Hurst exponent. Following their path, we establish simulations in NS-2 to investigate the degree of self-similarity produced with the interaction between the congestion...
control mechanisms of TCP and the packet dropping policies of droptail, RED and BLUE.

This paper is organized as follows. In Section 2, we summarize the Hurst parameter estimation techniques. We describe the queue mechanisms employed in our simulations in Section 3. In Section 4, we present our simulation results. We conclude and discuss the future perspectives in Section 5.

2 Estimation of Self-Similarity Parameter

Estimation of the degree of self-similarity has been of interest in many applications from hydrology to physics. Various estimators are developed to parameterize the degree of self-similarity, namely the Hurst parameter, $H$. For self-similar stationary data, the range of $H$ is restricted to $H > 0$. For $0 < H < 1/2$ the process has short-range dependence (SRD) and the correlations are summable [7]. On the other hand, for $1/2 < H < 1$ the process is considered long-range dependent and the correlations are insummable. As $H$ approaches to 1, the underlying process becomes more bursty [8].

In this study, three of the time-domain estimators given in [7] and [9] are selected for their common use in network literature and simplicity. Our aim is to analyze the self-similarity of aggregate incoming traffic to the bottleneck gateway. Throughout this paper, $X(k), k=0,1,2,\ldots,N$, represents the incoming traffic data of length $N$, which is manipulated to be a zero-mean discrete-time process. The following methods are employed to process $X(k)$:

2.1 Aggregated Variance Method

The aggregated variance method is based on averaging non-overlapping blocks of data of size $m$, where $i$ is the block index:

$$X^{(m)}(i) = 1/m \sum_{k=(i-1)m+1}^{im} X(k)$$

If $X(k)$ is asymptotically second-order self-similar process, then the variances of all its aggregated series, $X^{(m)}$, obey the following, as $m \to \infty$:

$$\text{var} (X^{(m)}) \approx am^{\beta}$$

where $a$ is a positive constant. Clearly, in the logarithmic scale (2) simplifies to a linear equation. The $\beta$ parameter which is now the slope, can be estimated by a linear-fit method. Note that $\beta$ is related to the Hurst parameter by

$$H = 1 - \beta/2, \quad 1/2 < H < 1$$

2.2 Absolute Value Method

The absolute value method uses the sum of the absolute values of the aggregated series, i.e.,

$$\mu(m) = \frac{1}{N/m} \sum_{k=1}^{N/m} |X^{(m)}(k)|$$

In the logarithmic scale, the slope of the plot of absolute values versus aggregation level, $m$, yield to be $H - 1$, if the process $X(k)$ has long-range dependence.

2.3 R/S Method

The R/S statistics is calculated by

$$\frac{R}{S} = \frac{1}{S(n)} \left[ \max_{0 \leq j \leq n} \left( Y(j) - \frac{j}{n} Y(n) \right) - \min_{0 \leq j \leq n} \left( Y(j) - \frac{j}{n} Y(n) \right) \right]$$

where $Y(n)$ is the partial sum

$$Y(n) = \sum_{k=1}^{n} X(k)$$

and $S^2(n)$ is the sample variance

$$S^2(n) = (1/n) \sum_{k=1}^{n} X(k)^2 - (1/n)^2 Y(n)^2$$

The expected value of $R(n)/S(n)$ shows progression related with $H$ as:

$$E[R(n)/S(n)] \propto cn^{\beta}$$

when $n \to \infty$ for fractional Gaussian noise (for details refer to [9]). Here, $E[.]$ is the expected value operator and $c$ is a positive constant. Once again in logarithmic scale $H$ can be estimated by the slope of the straight line.

It is relevant to mention that in our work, we have used other techniques such as periodogram and the wavelet based estimators. However, because of the periodicity observed in our data sets (which will be discussed in Section 4), these frequency sensitive methods estimate misleading $H$ values. Hence, the results are not included here.

3 Queue Management Techniques

3.1 Droptail

Droptail is the most widely deployed and the simplest queuing discipline. It does not apply any special decision mechanism for dropping packets. The incoming packets are accepted until the queue
overflows. When the queue overflows new incoming packets are accepted only after the head of the queue departs. This uncontrolled behavior gives rise to several problems when the network is highly loaded. Since the queue is full most of the time, the possibility of losing the packets within the same congestion window of a connection is higher. Packets that are injected to the network within a window are called back-to-back and they are supposed to be correlated. Besides many negative effects of droptail, it is claimed in [1] that the correlated losses have an increasing effect on self-similarity by intensifying burstiness.

### 3.2 RED

RED is one of the first AQM mechanisms developed [10]. The idea behind RED was that the control of network load could not be left to sources only, because each source would have information of its own state, but there was a need for a unified view. A mechanism, which would know the state of the total network and take action for all, was required.

RED marks or drops packets depending on a weighted average of queue length. In this way, congestion is notified to the sources who are invited to take action by reducing their sending rates before congestion becomes severe.

RED has five parameters to control the queue behavior, i.e., \( g_{\text{len}} \) (queue length), \( max_p \) (maximum marking probability), \( min_q \) (minimum queue length to start packet marking), \( max_q \) (maximum queue length, note that, beyond this threshold all the arriving packets are marked with probability of \( max_p \)) and \( w_q \) (weight factor). RED calculates the average queue size by using the weight factor \( w_q \). For the best performance, these parameters should be tuned considering the network architecture and traffic characteristics. This is one of the significant drawbacks of RED.

### 3.3 BLUE

BLUE uses packet loss and link utilization history to manage congestion. Whenever an overflow event is detected, BLUE increases its packet marking probability, \( p_m \). If the link is idle or in other words the queue is empty, then a queue idle event is detected and the marking probability is decreased. To take quick action for the negative events, BLUE increases \( p_m \) faster and decreases it slowly. The increments used to update \( p_m \) are \( p_i \) (probability to increase) and \( p_d \) (probability to decrease) where \( p_m \) is increased by \( p_i \) and decreased by \( p_d \) depending on the state of the queue. The time interval between two consecutive increments or decrements are denoted by \( f_i \) (interval for increment) and \( f_d \) (interval for decrement), respectively. The aggressiveness of the algorithm is determined by \( p_m \) and is learned in time.

BLUE performs better than RED especially when a large number of TCP sources are active and the aggregate traffic is extremely bursty [11]. Bursty traffic often defeats RED since queue lengths may grow or shrink before RED can react.

### 4 Simulation Results

In our simulations, we create the dumbbell topology of Fig. 1, in NS-2. At the bottleneck gateway, node 0, we place three different queue management mechanisms consequently: i) droptail, ii) RED, iii) BLUE.

For each scenario, we place 30, 40, 50, 60, 70, 80 and 90 servers (on the left hand side in Fig. 1) sending FTP traffic to the corresponding clients (on the right hand side in Fig. 1). The FTP traffic between servers and clients continue throughout the simulation (long TCP sources [1]). For the transport layer, TCP Reno implementation is used.

Our detailed parameter set is given in Table 1. The queue length is kept constant at 100 packets. The other four parameters of RED is selected as: \( min_q = 30, max_q = 90, max_p = 0.1, w_q = 0.02 \). For BLUE, the parameters are set to \( p_i = 0.002, p_d = 0.02, f_i = 10, f_d = 10 \).

The simulation results are given in Table 2. The first column indicates the source configuration. Throughput values are defined as the average of total arriving bits per unit time which is actually the average traffic generated by each source. \( H_1, H_2, H_3 \) are the Hurst parameters estimated by aggregated variances, absolute values and R/S methods, respectively.

To estimate \( H \), we form the time-series by collecting the total number of packets arriving at the gateway. We observe the correlation structure at the time-scales of above average RTT since TCP congestion control mechanisms can react within one round. Therefore, there is no need to utilize the multi-fractal analysis techniques [12] which are claimed to perform better and to be essential for small-time scale analysis. We conveniently, utilize the methods described in Section 2, as TCP flows show self-similar or mono-fractal behavior above RTT [13]. The estimated \( H \) values are given in Table 2.

![Fig. 1 Dumbbell Topology](image)
In Table 2, unexpected $H$ values are marked with gray. For example, $H$ values for 30 long sources with droptail, indicate SRD whereas $H$ values for 40 long sources indicate independence. These values are unreliable.

In order to analyze these peculiar $H$ values we use a nonparametric spectrum estimation technique, namely Welch’s method (see [14] pp. 415). The analysis of various droptail data, reveal that the data has periodic nature. This periodic nature can be attributed to TCP congestion control mechanisms.

The Power Spectral Density (PSD) plots for 30 and 40 flows employed to droptail queue, are given in Fig. 3 and 4, respectively. Five dominant periods are observed at approximately $T= 5.0, 2.5, 1.25$ s, plus 625 ms and 310 ms. Spectra of other configurations with more than 40 connections look alike having a periodic component in 2.5 s. We do not include all the figures due to space limitations.

As can be seen from Figure 5 and 6, the periods of congestion windows overlap with the periods of aggregate traffic. It is not surprising to observe similar PSDs having almost identical frequency contents (periods) since the packets are injected to the network within windows.

We refer to [15,16] to explain the reasons of this periodicity. In [15], it is suggested that TCP has chaotic behavior which makes the aggregate process wander around the margins of periodicity and self-similarity, where the system characteristics is determined by the following ratio:

$$\text{buffer size / number of sources}$$ (9)

If this ratio is small the system produces periodicity, on the contrary for large ratios it produces self-similarity. Furthermore, focusing on the TCP sources with the same RTT, synchronized behavior which yields periodicity can be related to [16]

$$W_c = \text{buffer size} + \text{Wopt}$$ (10)

where $W_c$ is the total number of packets that can be hold in the buffer and $Wopt$ is the optimum window size in packets to utilize the link without significant delay. It is calculated by

$$Wopt=\text{propagation delay} \times \text{bottleneck link bandwidth}$$ (11)

Again in [16], large, medium and small pipes are defined via $W_c$ and $Conn$ (number of sources or connections). TCP flows are claimed to behave differently for large, medium and small pipes.

**Table1** Fixed parameter set for all simulations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation duration</td>
<td>3600s</td>
</tr>
<tr>
<td>Initialization period</td>
<td>100s</td>
</tr>
<tr>
<td>Number of nodes on server side</td>
<td>90</td>
</tr>
<tr>
<td>Number of nodes on client side</td>
<td>90</td>
</tr>
<tr>
<td>Round trip propagation delay</td>
<td>240ms</td>
</tr>
<tr>
<td>Average RTT (round trip time)</td>
<td>1s</td>
</tr>
<tr>
<td>Delay between servers and n(0)</td>
<td>10ms</td>
</tr>
<tr>
<td>Delay between clients and n(1)</td>
<td>10ms</td>
</tr>
<tr>
<td>Delay of bottleneck link</td>
<td>100ms</td>
</tr>
<tr>
<td>Bandwidth between servers and n(0)</td>
<td>10Mbps</td>
</tr>
<tr>
<td>Bandwidth between clients and n(1)</td>
<td>10Mbps</td>
</tr>
<tr>
<td>Bandwidth of bottleneck link</td>
<td>1Mbps</td>
</tr>
<tr>
<td>Queue length of node 0</td>
<td>100 pkts</td>
</tr>
<tr>
<td>Packet size</td>
<td>1040 B</td>
</tr>
<tr>
<td>TCP window size</td>
<td>10 pkts</td>
</tr>
<tr>
<td>TCP maximum congestion window size</td>
<td>10 pkts</td>
</tr>
</tbody>
</table>

In the case when: i) $W_c > 3\ Conn$, ii) $Conn < W_c < 3\ Conn$, iii) $W_c < Conn$, the pipes are called large, medium and small, respectively.

In large pipes, sources tend to maximize their sending rates. When the total number of sent packets exceeds $W_c$, the sources realize an indication of loss and decrease their sending rates all together. This causes global synchronization.

In our simulations $Wopt$ is 30 and $W_c$ is 130 packets. The large pipe case is applicable to our simulations with 30 and 40 long sources with droptail scenario.

In [16], there are calculations for RED queues as well. If the buffer size of droptail is replaced with the $maxq$ of RED, a $W_c$ of 120 packets is calculated. In the scenario of 90 sources with RED queue some of the sources cease transmission. This explains the peculiar $H$ values for 90 long flows with RED.

As a result, 30 and 40 flows for droptail and 90 flows for RED lead to periodicity related to the simulation scenario and invite unreliable $H$ estimates [14].
Table 2 Throughput Values (In Bits/Sec) and H Estimates for Droptail, Red and Blue Queues (H1, H2, H3 are Estimated with Aggregated Variances, Absolute Values And R/S Method, respectively.)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Droptail</th>
<th>RED</th>
<th>BLUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Throughput</td>
<td>H1</td>
<td>H2</td>
</tr>
<tr>
<td>30 long</td>
<td>34692.2</td>
<td>0.51</td>
<td>0.27</td>
</tr>
<tr>
<td>40 long</td>
<td>26542.3</td>
<td>0.50</td>
<td>0.56</td>
</tr>
<tr>
<td>50 long</td>
<td>21532.4</td>
<td>0.74</td>
<td>0.74</td>
</tr>
<tr>
<td>60 long</td>
<td>18201.8</td>
<td>0.75</td>
<td>0.78</td>
</tr>
<tr>
<td>70 long</td>
<td>15740.8</td>
<td>0.65</td>
<td>0.67</td>
</tr>
<tr>
<td>80 long</td>
<td>13873.4</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>90 long</td>
<td>12400.4</td>
<td>0.65</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Fig. 3. The PSD plot for total arriving bytes from 30 long sources to droptail queue (x-axis is normalized frequency.)

Fig. 4. The PSD plot of total arriving bytes from 40 long sources to droptail queue (x-axis is normalized frequency.)

Fig. 5. The PSD plot of congestion window for 10 selected sources of 30 long sources with droptail (x-axis is normalized frequency.)

Fig. 6. The PSD plot for congestion window of 10 selected sources of 40 long sources with droptail (x-axis is normalized frequency.)
5 Conclusion

In this work, we analyze the degree of self-similarity generated by the interaction of TCP congestion control mechanisms and queue management policies. We conclude that due to the periodicities produced by a large number of TCP connections having the same RTT applied to a single bottleneck, there is no deterministic and obvious improvement of $H$ for AQM techniques. We could not witness a lessening of the degree of self-similarity for AQM techniques (unfortunately, we failed to obtain identical results with [1].) The periodicity produced by synchronization of congestion windows of a large number of TCP flows is directly related with the choice of identical RTTs. We conclude with a final remark that before investigating the self-similarity of any simulated data, periodic behavior should be checked. This periodic behavior could be removed with adding random RTTs and random processing times [16]. We are currently working on this issue.

References


