Real-time Network Traffic Management using the Modified BPTraSha Algorithm

Karim Mohammed Rezaul
Centre for Applied Internet Research (CAIR)
Glyndwr University, Wrexham, Wales, UK
karim@cair-uk.org

Vic Grout
Centre for Applied Internet Research (CAIR)
Glyndwr University, Wrexham, Wales, UK
v.grout@glyndwr.ac.uk

Abstract- **Various researchers have reported that traffic measurements demonstrate considerable burstiness on several time scales, with properties of self-similarity. Also, the rapid development of technologies has widened the scope of network and Internet applications and, in turn, increased traffic. The self-similar nature of this data traffic may exhibit spikiness and burstiness on large scales with such behaviour being caused by strong dependence characteristics in data: that is, large values tend to come in clusters and clusters of clusters and so on. Several studies have shown that TCP, the dominant network (Internet) transport protocol, contributes to the propagation of self-similarity. Bursty traffic can affect the Quality of Service of all traffic on the network by introducing inconsistent latency. It is easier to manage the workloads under less bursty (i.e. smoother) conditions. This paper continues the work published in [1], which introduced a novel algorithm for traffic shaping to smooth out the traffic burstiness. It was named as the Bursty Packet Traffic Shaper (BPTraSha). Experimental results show that this approach allows significant traffic control by smoothing the incoming traffic. BPTraSha can be implemented on the distribution router buffer so that the traffic's bursty nature can be modified before it is transmitted over the core network (e.g., Internet). A modified BPTraSha algorithm is proposed in this research, which can be shown to be more dynamic, and therefore responsive, than the previous one. In this case, the dynamic variation of link speed can lead to further reducing the long-range dependence of network traffic.**

Keywords:
Self-similarity, LRD, ACF, QoS, Shaping, BPTraSha.

I. Introduction

A number of factors, such as a slow start phase of the congestion window, packet losses, ack-compression of TCP traffic and multiplexing of packets at the bottleneck rate, can cause either short- or long-term burstiness in the behaviour of TCP flow [2]. The research in [3] investigates how various versions of TCP congestion control affect network performance when traffic is bursty. It shows a significant adverse impact on network performance attributable to traffic self-similarity and, while throughput declines gradually as self-similarity increases, queueing delay increases more drastically. Self-similarity is closely related to the phenomenon of heavy-tailed distributions, where the tail index of the distribution declines as a power law with small index (less than 2). TCP represents the dominant transport protocol of the network (e.g., Internet), which contributes to the propagation of self-similarity [4]. It was shown in [4] that TCP itself inherits self-similarity when it is combined with self-similar background traffic in a bottleneck buffer through the transform function of the linear system.

The research in [5] investigated the relationship between TCP's congestion control mechanism and traffic self-similarity under certain network conditions. It demonstrates that, when a TCP connection is going through a highly-lossy channel - and the loss condition is not affected by this single TCP connection's behaviour, TCP starts to produce packet trains that show pseudo-self-similarity [6] (i.e. traffic is self-similar over limited range of time scales). In fact, when the loss rate is relatively high, TCP's adaptive congestion control mechanism generates traffic with heavy-tailed off or idle periods (i.e. inter-arrival time), which in turn introduces long-range dependence into the overall traffic. The researchers in [7] analysed the traces of actual TCP transfers over the Internet and reported that individual TCP flows, isolated from the aggregated flow on the link, also have a self-similar nature. Also, the loss rate experienced by TCP flow is an important indicator of the degree of self-similarity in the network traffic. A natural construction of the extremely bursty nature of TCP traffic comes from timeouts (representing 'silent' periods) that lead to losses and, consequently, losses increase the burstiness - and higher loss rates thus lead to a higher degree of self-similarity (i.e. higher values of Hurst parameter) [7]. It has been shown [8] that if packets were to arrive according to the well-behaved Poisson process, simple retransmission mechanisms can make traffic appear self-similar over time scales and be a possible source of long-range dependence. Retransmission mechanisms can make a network congestible, because these mechanisms often cause network inefficiencies which cause throughput to degrade specifically in periods when load is already high.

One of the major drawbacks of TCP/IP is the lack of true Quality of Service (QoS) functionality. QoS in networks, in simple terms, is the ability to guarantee and limit bandwidth appropriately for certain services and users. Traffic shaping is the term used for any system by which traffic is constrained to a specific speed. Traffic shaping is an attempt to control network traffic in order to optimize, attempt to optimize or guarantee performance, low-latency and bandwidth. Traffic shaping deals with concepts of classification, queue disciplines, enforcing policies, congestion management, QoS and fairness. Shaping is the mechanism by which packets are delayed before transmission in an output queue to meet a desired output rate. This is one of the most common requirements of users seeking bandwidth control solutions. The basic principle of traffic shaping is based on the fact that the outgoing traffic from the FireBrick or router is scheduled. The FireBrick is a network appliance with a rich feature set, including a stateful firewall, router, managed switch, traffic shaping, tunneling, multilink handling, and much more.

Each packet has a time stamp, stating when it is to be sent, and all traffic is normally sent in order and not before its time. This method is used to deliberately slow responses from reject and bounce filters, as well as for speed lanes. When sending a packet, its length is considered and the transmission time added to the time for the next packet to be sent. This ensures packets can only leave at the designated rate and no faster. Shapers can smooth out bursty traffic and attempt to limit or ration traffic to meet, but not exceed, a configured rate (e.g. packets per second or bits/bytes per second). However earlier research [9, 10] reports that the strong robustness of self-similarity properties existing in traffic cannot be removed by shaping.

This paper is organised as follows. Section II highlights research related to shaping traffic. Section III describes the definitions of self-similarity, long-range dependence and the autocorrelation function. Section IV introduces the algorithm BPTraSha and its purpose. Section V discusses the performance and complexity of BPTraSha by experimental analysis. Finally we draw conclusions and suggest future work in section VI.

II. RELATED RESEARCH

Several researchers show how to control the network in situations where the distribution tail of the traffic flow process cannot be altered. In [11] it is claimed that, by incorporating shapers and policers at the edges of the networks, huge buffers are needed that result in large delays and may thus be unacceptable in practice. In [12] a Burst Shaping Queueing (BSQ) algorithm is presented, which can minimize the burstiness of traffic on packet switched routers by interleaving packets that are going to follow different links on next hops. The research in [13] discusses issues of shaping and simulated queueing performance of ATM traffic. In this work, a leaky bucket shaping method is used and the shaping effect surprisingly results in higher values for the estimated Hurst parameter (the degree of self-similarity) - that is, the estimated Hurst parameter is increased due to shaping. It is also noted that the interpretation of the estimated Hurst parameter is problematic in practice.

In [14] an optical packet assembly mechanism is proposed to function as a traffic shaper and its impact on self-similar traffic characteristics at the edge router are investigated. Simulation results demonstrate that the optical packet assembly mechanism can reduce traffic correlation and the degree of self-similarity. In [15], the three different traffic shaping techniques are presented: thinning, striping and shuffling, which can improve the queueing characteristics of data by decreasing the short-term burstiness and diminishing short-term correlations. However, none of these processes are shown to decrease the degree of Long-Range Dependence (LRD) in data. The research in [16] proposes a dual leaky bucket technique for shaping the web traffic, reducing the intensity of the long duration traffic bursts, which, in turn, reduces the Hurst parameter. The 'leaky bucket' procedure [17] is also employed in [18] to examine the effectiveness of shaping in the case of α-stable fractal traffic and it is found that shaping and policing mechanisms do not eliminate self-similarity.

III. SELF-SIMILARITY, LONG-RANGE DEPENDENCE AND AUTOCORRELATION FUNCTION

It is especially important to understand the link between self-similarity and long-range dependence of network traffic and performance of the networks because such characterization can be potentially applied for control purposes such as traffic shaping, load balancing, etc. In general two or more objects having the same characteristics are called self-similarity. A phenomenon that is self-similar looks the same or behaves the same when viewed at different degrees of magnification or different scales on a dimension and bursty over all time scales. Self-similarity is the property of a series of data points to retain a pattern or appearance regardless of the level of granularity used and is the result of long-range dependence in the data series. If a self-similar process is bursty at a wide range of timescales, it may exhibit long-range- dependence. In general lagged autocorrelations are used in time series analysis for empirical stationary tests. Self-similarity manifests itself as long-range dependence (i.e., long memory) in the time series of arrivals. The evidence of very slow, linear decay in the sample lag autocorrelation function (ACF) indicates the nonstationary behaviour [19]. Long-range-dependence means that all the values at any time are correlated in a positive and non-negligible way with values at all future instants. For a continuous time process $Y = \{Y(t), t \geq 0\}$ is self-similar if it satisfies the following condition [20]:

$$Y(t) \overset{d}{=} a^{-H} Y(a t), \quad \forall a > 0, \quad and \quad 0 < H < 1 \qquad (3.1)$$

where H is the index of self-similarity, called Hurst parameter and the equality is in the sense of finite-dimensional distributions.

The stationary process X is said to be a long-range dependent process if its autocorrelation function (ACF) is non-summable [21] meaning that $\sum_{k=-\infty}^{\infty} \rho_k = \infty$ (3.2)

The details of how ACF decays with k are of interest because the behaviour of the tail of ACF completely determines its summability. According to [22], X is said to exhibit long-range dependence if

$$\rho_k \sim L(t)\, k^{-(2-2H)}, \quad as\ k \to \infty \qquad (3.3)$$

where $\frac{1}{2} < H < 1$ and $L(.)$ slowly varies at infinity, i.e.,

$$\lim_{t \to \infty} \frac{L(xt)}{L(t)} = 1, \quad \text{for all } x > 0 \qquad (3.4)$$

Equation (3.3) implies that the LRD is characterized by an autocorrelation function that decays hyperbolically rather than exponentially fast.

LRD processes are characterised by a slowly decaying covariance function that is no more summable. When the network performance is affected by LRD the data are correlated over an unlimited range of time lags and this property results in a scale invariance phenomenon. Then no characteristic time scale can be identified in the process, they are all equivalent for describing its statistics, i.e., the part resembles the whole and vice versa. This is why LRD is also called Self-Similarity [23].

IV. BPTRASHA: AN ALGORITHM FOR CONTROLLING BURSTY TRAFFIC

Let us assume that the client networks (such as C_1, C_2, C_3,….., C_n) are connected to the main router of Internet service provider (ISP). The packet sequences (i.e. packet size in byte) from different sources are queued at the router buffer. The packet sequences arrive at the router buffer with timestamp in second (or millisecond). Therefore we have packet size in byte for corresponding timestamp. For the experimental analysis we used Lawrence Berkeley Laboratory (LBL) TCP data which are publicly available in [24]. The bursty nature of packet sequences arrive at the router will be shaped with the fixed rate by the shaper algorithm BPTraSha. Here we mean the link speed as desired fixed rate (i.e. capacity, C) at which the packets would be transmitted. In other words, a bursty traffic in the input will be regulated with the fixed rate before they pass through the network. The algorithm is described in Figure 1. For the user's convenience, the algorithm is implemented both in Java and Matlab programming language.

```
T = timestamp
B = Packet size in bytes
TT = transmission time
bps = Bit per second
Delt = Delay in second
Tmod = Modified time
Tmod_cng = change in modified time
bps_mod = Modified bit per second
Ld = Longest delay
Sd = Shortest delay
S = sample count (e.g. number of packet sequences)
C = link speed

1. Capture B for corresponding T (i.e. T and B)
2. Count S
3. For k = 0 to (S-1)
   a)  if (k = 0)
           bps[k] = B[k] *8 / (T[k]+TT[k])
             where TT[k] = B[k] *8 / C
         else bps[k] = B[k]*8 / (T[k]-T[k-1]+TT[k-1])
   b)  if (k = 0)
           Delt[k] = 0
           Tmod[k] = T[k]
         else
              i)   Delt[k] = T[k]-(Tmod[k-1]+TT[k-1])
              ii)  if (Delt[k] >= 0)
                      Tmod[k] = T[k]
                    else
                       Tmod[k] = T[k]-Delt[k]
4. For k = 0 to (S-2)
   i)  if (k = 0)
           Tmod_cng[k] = Tmod[k]
        else
           Tmod_cng[k] = Tmod[k+1]-Tmod[k]
   ii)  set   bps_mod[k] = B[k]*8 / Tmod_cng[k]
   iii)  if (Delt[k] <0)
           find out Ld      // Longest delay
           find out Sd      // Shortest delay
5. Exit
```

Fig. 1. The algorithm, BPTraSha

The performance of algorithm has been depicted in Figure 2 to Figure 10. The Figures show how the bursty nature of traffic are smoothed out by the algorithm, i.e., bursty traffic have been shaped by the desired fixed rate. The length of packet sequences used for these experiments is N =65536. We used various types of TCP data for the experiment, but due to space limitations we provide here results from using LBL-TCP3-packet, LBL-TCP4-packet and LBL-TCP5-packet data. The link capacity (i.e. desired rate) applied here are C = 5 Mbps, C = 10 Mbps and C =15 Mbps. Figure 11 illustrates the expected longest delay observed for different link speed with the variation of length of packet sequences. It is clear from the Figure that higher capacity yields less delay and thereby provides better quality of service. Figure 12 shows the expected shortest and longest delay for different link speed while length of sequences is varied. The shortest delay found to be from 0.000001 second to 0.000004 second. The longest delay here observed to be from 0.00056 second to 0.15927 second depending on the link speed (C) and length (N) of packet sequences. The higher the link speed the shorter the observed delay. Table 1 illustrates a sample of trace files that the BPTraSha algorithm uses.

TABLE 1: SAMPLE OF A TRACE FILE

Length of samples	Timestamp (T_i)	Packet size in byte (B_i)
1	0.008185	41
2	0.010445	42
3	0.023775	42
4	0.026558	41
5	0.029002	82
6	0.032439	55
7	0.049618	41
8	0.052431	42
9	0.056457	42
10	0.057815	454
11	0.072126	40
12	0.098415	95
13	0.104465	55
14	0.122345	40
15	0.12449	40
16	0.125228	41
17	0.138935	41
18	0.13995	104
19	0.14093	41
20	0.146912	72
: :	:	:
N	T_n	B_n

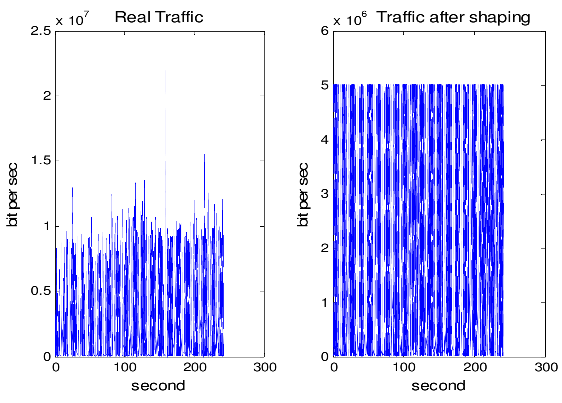

Fig. 2. LBL-tcp3-pkt, C = 5 Mbps, N = 65536

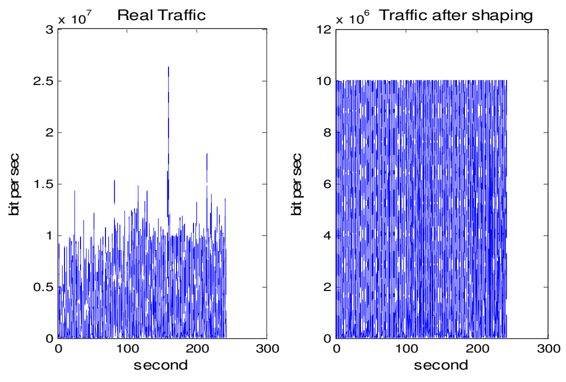

Fig. 3. LBL-tcp3-pkt, C = 10 Mbps, N = 65536

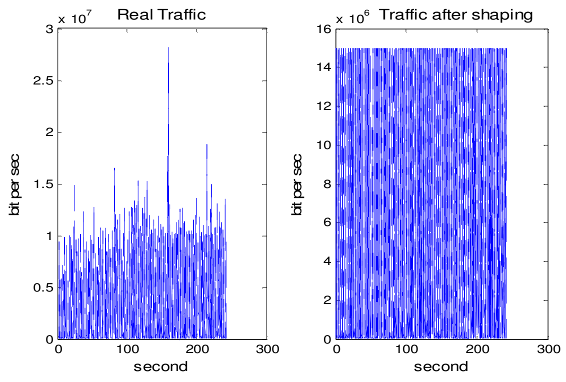

Fig. 4. LBL-tcp3-pkt, C = 15 Mbps, N = 65536

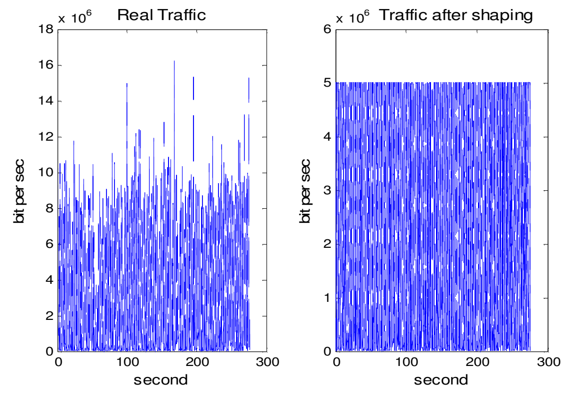

Fig. 5. LBL-pkt-4_tcp, C = 5 Mbps, N = 65536

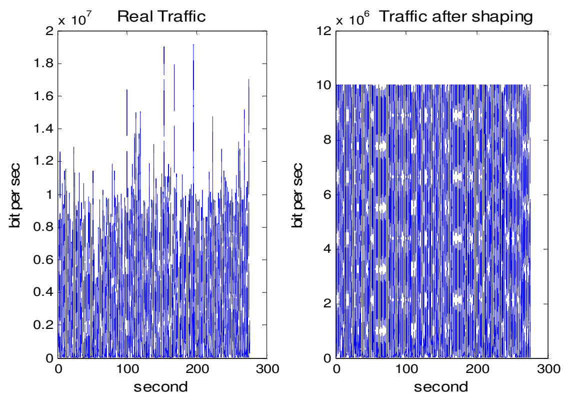

Fig. 6 . LBL-pkt-4_tcp, C = 10 Mbps, N = 65536

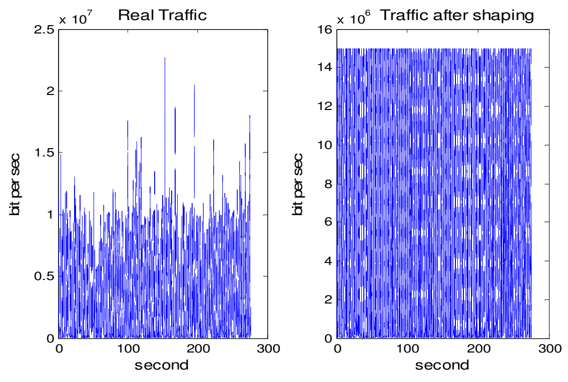

Fig. 7. LBL-pkt-4_tcp, C = 15 Mbps, N = 65536

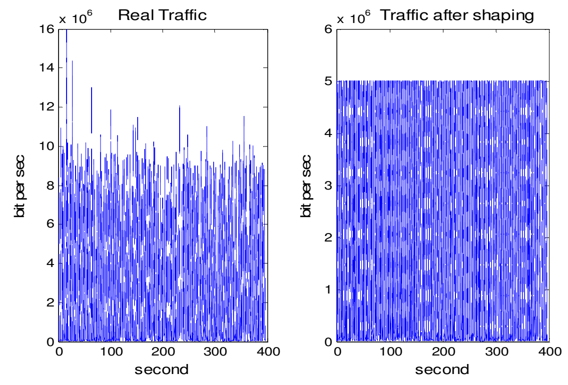

Fig. 8. LBL-pkt-5_tcp, C = 5 Mbps, N = 65536

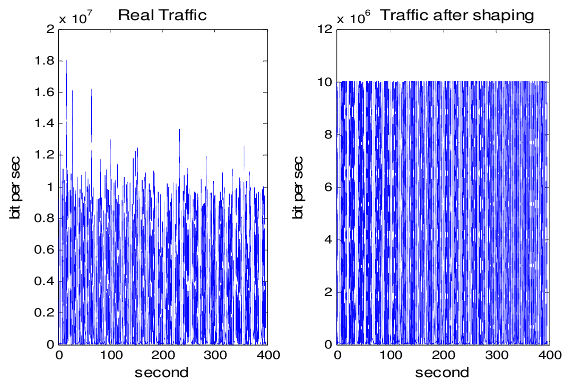

Fig. 9. LBL-pkt-5_tcp, C = 10 Mbps, N = 65536

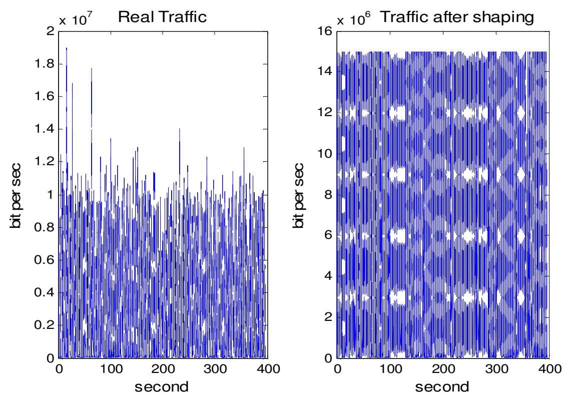

Fig. 10. LBL-pkt-5_tcp, C = 15 Mbps, N = 65536

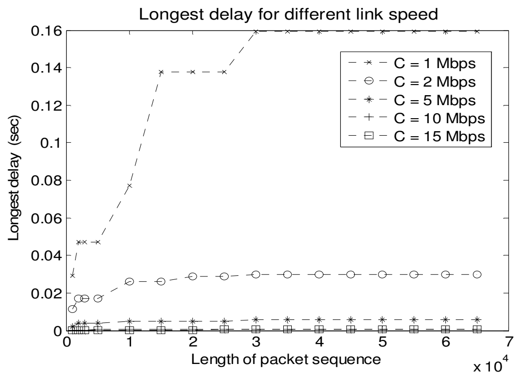

Fig. 11. Performance of BPTrasha algorithm: Observation of longest delay. Variation of link speed with different length of packet sequences.

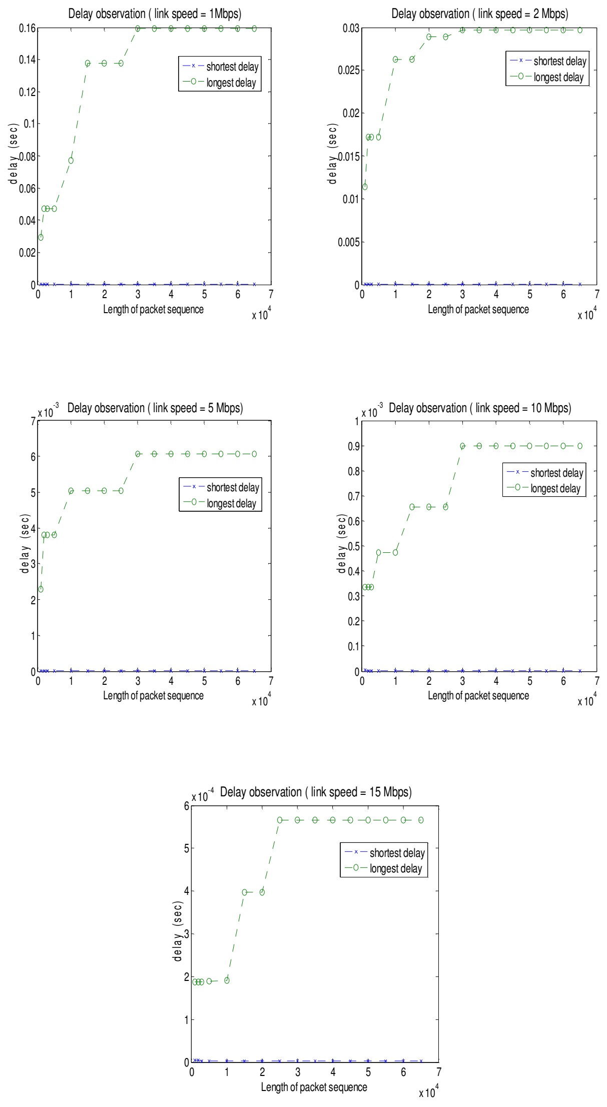

Fig. 12. Performance of BPTrasha algorithm: Observation of shortest and longest delay, for different length of packet sequences.

V. Complexity of the algorithm, BPTraSha

To explore the complexity of BPTraSha we chose six workstations with different specifications, which are represented in Table II. We investigated several lengths of packet sequences such as N = 1000, N = 2000, N = 3000, N = 5000, N = 10000, N = 15000, N = 20000, N = 25000, N = 30000, N = 35000, N = 40000, N = 45000, N = 50000, N = 55000, N = 60000 and N = 65000. In our research we mainly emphasise the time (as opposed to space) complexity of the algorithm.

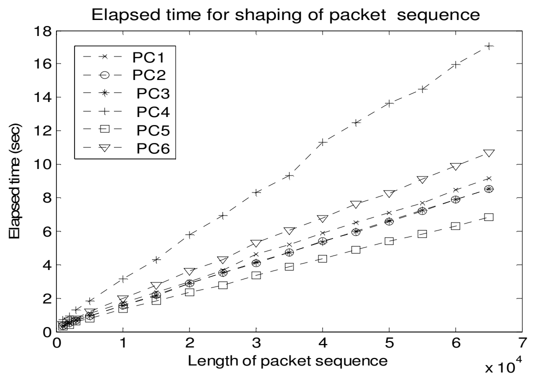

Fig. 13. Observation of elapsed time for different length of packet sequences with different PC's

Figure 13 depicts the observation of elapsed (execute) time for different lengths of packet sequences with different PCs. It is obvious that PC5 yields better performance as it possesses higher specifications. Figure 14 shows a percentage of affected packets due to delay for different lengths of packet sequences. Here higher capacity (C) signifies better performance due to less affected packets. However, the elapsed time for executing the algorithm does not significantly vary for different link speeds with the variation in lengths of packet sequences, a feature that can be observed in Figure 15.

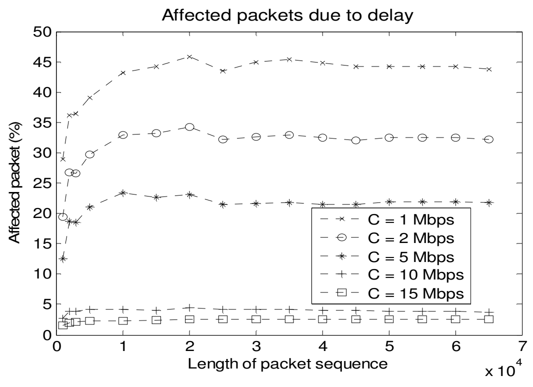

Fig. 14. Affected packets due to delay for different length of packet sequences

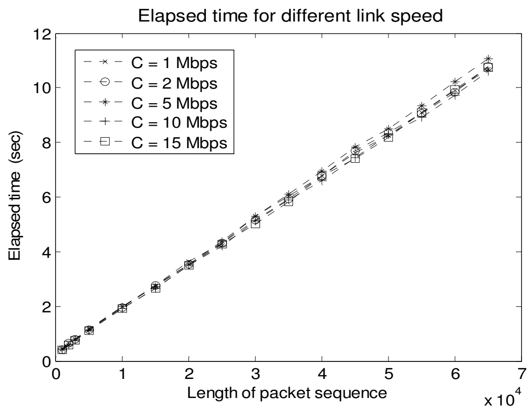

Fig. 15. Elapsed time for different length of packet sequences with the variation of link speed.

TABLE II. WORKSTATIONS WITH DIFFERENT SPECIFICATION

Work station	Specification
PC1	Intel Pentium (R) 4, CPU 2.4 GHz, 512 MB of RAM
PC2	Intel Pentium (R) 4, CPU 3.0 GHz, 0.99 GB of RAM
PC3	Intel Pentium (R) 4, CPU 3.0 GHz, 504 MB of RAM
PC4	Intel Pentium (R) 3, CPU 866 MHz, 384 MB of RAM
PC5	Intel Centrino Duo Core, CPU T2250 @ 1.73 GHz, 1024 MB of RAM
PC6	Intel Pentium (R) 4, CPU 1.80 GHz, 256 MB of RAM

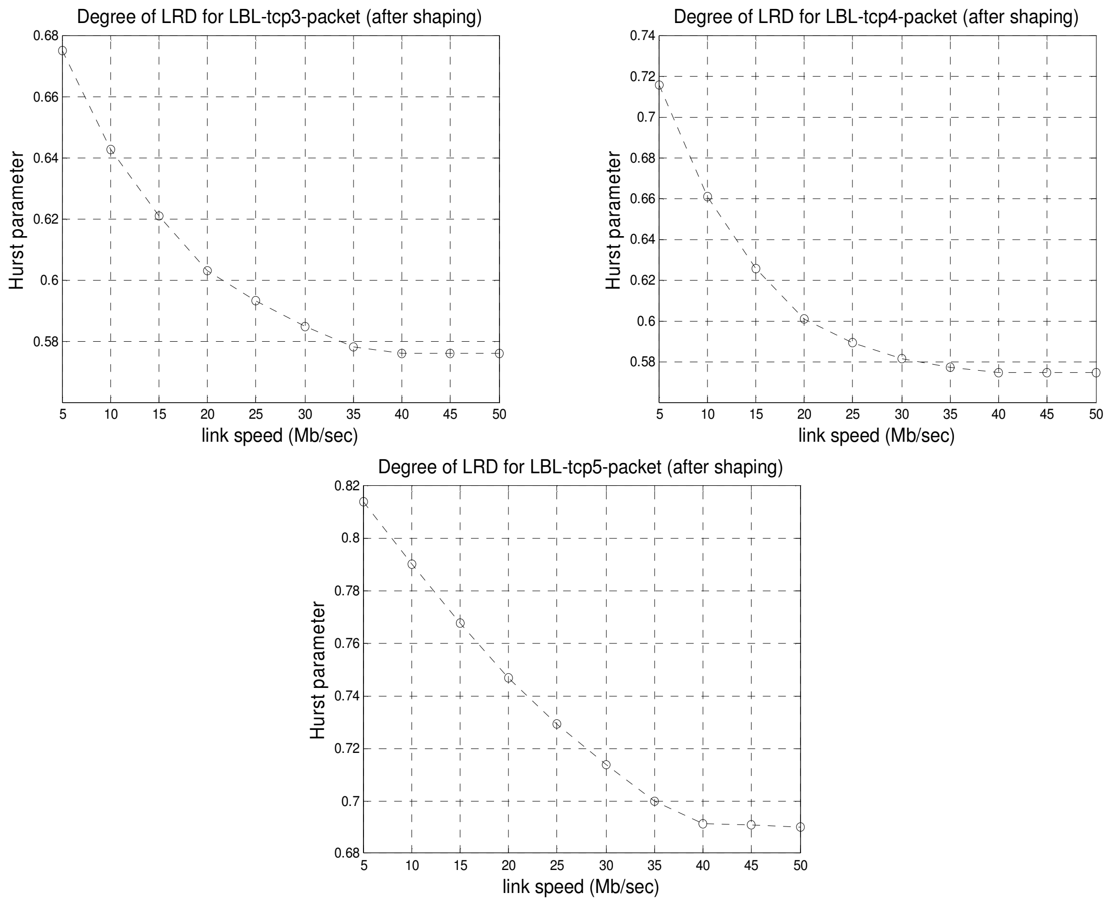

Fig 16: Variation of the degree of LRD with different link speed (C). Before shaping, the estimated H = 0.66, H = 0.68 and H = 0.7968 for LBL-TCP3-packet, LBL-TCP4-packet and LBL-TCP5-packet respectively.

Figure 16 depicts the variation of the degree of LRD with different link speeds (C) while shaping the traffic. Before shaping, the estimates are H = 0.66, H = 0.68 and H = 0.7968 for the LBL-TCP3-packet, LBL-TCP4-packet and LBL-TCP5-packet respectively. Clearly the Hurst parameter (H) decreases with increasing link speed (C), meaning that long-range dependent traffic can be reduced by the BPTraSha algorithm. As link speed is inversely proportional to link utilisation, a higher C designates lower utilisation and thereby reduces traffic bit rate, which is consistent with our findings. In Figure 2 to Figure 10 it is clear that the traffic burstiness (i.e., large variation of traffic bit rate) is reduced with the desired rate (i.e., C) by the shaping algorithm. Note that LRD cannot be reduced by simply increasing the link speed but it (C) plays an important role when using BPTraSha algorithm. Also note that at a certain limit (C = 40, 45 and 50 Mbps) the Hurst parameter remains unchanged: that is, no reduction of LRD is possible any more. The Hurst parameter is estimated here by HEAF(2) [25, 26]. Since there is an obvious correlation existing between link speed (C) and Hurst parameter (H), the algorithm has been modified as shown in Figure 17. In

Figure 17, the modified part of BPTraSha is evident in step 3 (i.e., Estimate LRD by H). The link speed is chosen according to the intensity of existing LRD traffic, estimated by H. It is clear from step 3 that the value of link speed falls into different ranges of H.

VI. Conclusion and Future Work

In this research, we present a modified version of the algorithm, BPTraSha, to control the bursty nature of network traffic. Experimental results show that the BPTraSha algorithm is capable of smoothing out the bursty nature of traffic packets received at the router buffer before they are transmitted to the core network (Internet). According to complexity and delay analyses, it is clear that the algorithm is not dependent on the size of the network or the amplitude of the spike. We naturally found that the higher the link speed, the shorter the observed delay.

Also, it is clear from Figure 16, that LRD can be reduced by BPTraSha with increasing link speeds. In the modified algorithm, the dynamic variation of link speed (C) has been shown, depending on the intensity of LRD, which is measured by Hurst parameter (H). As the main function of BPTraSha is to shape the bursty packet traffic, it can contribute to reducing the network load and lead to the improvement of QoS in future network (e.g., Internet) performance. Future work will include an evaluation of the applicability of the modified BPTraSha algorithm to real-time implementation at the FireBrick or router.

```
T = timestamp
B = Packet size in bytes
TT = transmission time
bps = Bit per second
Delt = Delay in second
Tmod = Modified time
Tmod_cng = change in modified time
bps_mod = Modified bit per second
Ld = Longest delay
Sd = Shortest delay
S = sample count (e.g. number of packet sequences)
C = link speed

1. Capture B for corresponding T (i.e. T and B)
2. Count S
3. Estimate LRD by Hurst parameter (H)
   if  H <= 0.5
       C = 1+Math.random( )*3  // to generate random C
                                  between 1 and 3
   elseif  0.5 < H <= 0.6
       C = 4+Math.random( )*15
   elseif  0.6 < H <= 0.7
       C = 15+Math.random( )*35
   else  0.7 < H <= 1
       C = 35+Math.random( )*50
4. For k = 0 to (S-1)
 (a)   if (k = 0)
          bps[k] = B[k] *8 / (T[k]+TT[k])
            where TT[k] = B[k] *8 / C
       else bps[k] = B[k]*8 / (T[k]-T[k-1]+TT[k-1])
 (b)   if (k = 0)
          Delt[k] = 0
          Tmod[k] = T[k]
       else
           i)   Delt[k] = T[k]-(Tmod[k-1]+TT[k-1])
           ii)  if (Delt[k] >= 0)
                    Tmod[k] = T[k]
                else
                    Tmod[k] = T[k]-Delt[k]
5. For k = 0 to (S-2)
   i)  if (k = 0)
          Tmod_cng[k] = Tmod[k]
       else
          Tmod_cng[k] = Tmod[k+1]-Tmod[k]
   ii)  set    bps_mod[k] = B[k]*8 / Tmod_cng[k]
  iii) if (Delt[k] <0)
         find out Ld      // Longest delay
         find out Sd      // Shortest delay
6. Exit
```

Fig. 17. Modified BPTraSha algorithm

REFERENCES

[1] Karim M. Rezaul & Grout V., BPTraSha: A Novel Algorithm for Shaping Bursty Nature of Internet Traffic, Proceedings of the 3rd IARIA/IEEE Advanced International Conference on Telecommunications (AICT 2007), May 13-19, 2007, Mauritius.

[2] Amit Aggarwal, Stefan Savage and Thomas Anderson, Understanding the Performance of TCP Pacing. *Proc. of the IEEE INFOCOM 2000 Conference on Computer Communications*, March 2000, pp. 1157 - 1165.

[3] K. Park, G. Kim, and M. Crovella, On the effect of self-similarity on network performance, *In Proceedings of the SPIE International Conference on Performance and Control of Network System*, November 1997, pp. 296-310.

[4] A. Veres, Zs. Kenesi, S. Molnár, G. Vattay, TCP's Role in the Propagation of Self-Similarity in the Internet, *Computer Communications*, Special issue on Performance Evaluation of IP Networks and Services, Vol. 26, Issue 8, May 2003, pp. 899-913.

[5] L. Guo, M. Crovella, and I. Matta, TCP congestion control and heavy tails, *Technical Report*: BUCSTR -2000-017, Computer Science Dept - Boston University, 2000.

[6] Liang Guo, Mark Crovella and Ibrahim Matta, How does TCP Generate Pseudo-Self-Similarity? *In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS~'01).* Cincinnati, Ohio, pp. 215-223.

[7] Biplab Sikdar and Kenneth S. Vastola, On the Contribution of TCP to the Self-Similarity of Network Traffic, *Lecture Notes In Computer Science, Proceedings of the Thyrrhenian International Workshop on Digital Communications: Evolutionary Trends of the Internet*, Vol. 2170, 2001, Springer-Verlag, London, UK, pp. 596 – 613.

[8] J. M. Peha, Protocols can make traffic appear self-similar*, In Proceedings of the 1997 IEEE/ACM/SCS Comm. Networks and Distributed System. Modeling and Simulation Conference*, Jan 1997, pp. 47-52.

[9] A. L. Neidhardt and A. Erramilli, Shaping and policing of fractal traffic, *In 10th ITC Specialists Seminar on Control in Communications*, 1996, pp. 253-264.

[10] S. Vamvakos and V. Anantharam, On the departure process of a leaky bucket system with long-range dependent input traffic, *Queuing Systems: Theory and Applications*, vol. 28, 1998, pp. 191-214.

[11] Pruthi, P. and Popescu, A., Effect of Controls on Self-Similar Traffic*, In Proceedings of the 5th IFIP ATM Workshop*, Bradford, UK, July 1997.

[12] Vasilios Darlagiannis, Martin Karsten, and Ralf Steinmetz. Burst Shaping Queueing. In *Computer Networks and Distributed Systems (WMC)*, SCS, January 2003, pp. 65-70.

[13] S. Molnár and A. Vidács, On Modeling and Shaping Self-Similar ATM Traffic, *Proc. in 15th International Teletraffic Congress (ITC15)*, Washington, USA, July, 1997.

[14] Fei Xue, S. J. Ben Yoo, Self-similar traffic shaping at the edge router in optical packet-switched networks, *Proc. IEEE International Communication Conference (ICC 2002)*, vol.4, April 28- May 2, New York, 2002, pp.2449-2453.

[15] Dennis Bushmitch, S. S. Panwar, and A. Pal, Thinning, striping and shuffling: Traffic shaping and transport techniques for variable bit rate video, *In proceedings of the IEEE Globecom 2003*, vo.2, November 17-21, Taipei, pp.1485-1491.

[16] K. Christensen, V. Ballingam, Reduction of Self-Similarity by Application-level Traffic Shaping, *In Proc. IEEE 22nd Annual Conference on Local Computer Networks*, November 1997, pp. 511 – 518.

[17] J. Turner, New directions in communications, or which way to the information age?, *IEEE Communications Magazine*, vol. 24, 1986, pp. 8-15.

[18] Harmantzis F.C. , Hatzinakos D. and Katzela I. , Shaping and policing of fractal α-stable broadband traffic, *Proc. Canadian Conf. on Elec. and Comp. Engineering (CCECE)*, Toronto, Canada, May 2001, pp. 697-202.

[19] Brocklebank J. and D. Dickey. SAS System for Forecasting Time Series. *SAS Institute Inc*. Cary NC. 1986.

[20] Walter Willinger, Vern Paxson, and Murad Taqqu, Self-similarity and Heavy Tails: Structural Modeling of Network Traffic, Adler, R., Feldman, R., and Taqqu, M.S., (editors), *In A Practical Guide to Heavy Tails: Statistical Techniques and Applications*, Birkhauser, 1998.

[21] Cox D., Long-Range Dependence: a Review. H. A. David and H. T. David (eds.), *In Statistics: An Appraisal, Iowa State Statistical Library*, The Iowa State University Press, 1984, pp.55-74.

[22] Leland Will E. Taqqu M. S., Willinger W. and Wilson D. V., On the Self-similar nature of Ethernet Traffic (Extended version), *IEEE/ACM Transactions on Networking*, February 1994, Vol. 2, No. 1, pp. 1-15.

[23] Antoine Scherrer, Antoine Fraboulet, Tanguy Risset, Multi-phase On-chip Traffic Generation Environment, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, *INRIA, Research Report No. 2006-22*, June 2006.

[24] *Internet Traffic Archive*: http://ita.ee.lbl.gov/html/traces.html

[25] Karim M. Rezaul and Grout V., Exploring the Reliability and Robustness of HEAF(2) for Quantifying the Intensity of Long-Range Dependent Network Traffic, International Journal of Computer Science and Network Security, Vol. 7, No. 2, February 2007, pp. 221-229.

[26] Karim M. Rezaul, Pakštas A., Gilchrist R. and Chen T.M., HEAF: A Novel Estimator for Long-Range Dependent Self-similar Network Traffic, Y. Koucheryavy, J. Harju, and V.B. Iversen (Eds.): Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN), May 29 - June 2, 2006, LNCS 4003, pp. 34 – 45.