Buffer Management for Self-Similar Network Traffic

Faranz Amin
Electrical Engineering and computer science Department
Yazd University
Yazd, Iran
farnaz.amin@stu.yazd.ac.ir

Kiarash Mizanian
Electrical Engineering and computer science Department
Yazd University
Yazd, Iran
k.mizanian@yazi.ac.ir

Abstract—**Traffic self-similar has been discovered to be a ubiquitous phenomenon in communication networks and multimedia systems and it is well known that self-similar traffic can lead to larger queuing delays and higher packet loss rates. Buffer management of queuing systems plays a crucial role in addressing the tradeoff within efficiency measured in terms of overall packet loss and fairness measured in terms of individual source packet loss. Ease of implementation is the key issue while determining the practicality of a dynamic buffer management technique. In this paper, we propose a new Queue Management based on self-similarity of network traffic. We use ns2 to simulate network configurations and we generated traffics with Pareto distribution on ns2 simulator, the numerical results demonstrate performance of proposed algorithm in compared to the other currently implemented buffer management algorithms in ns2.**

Keywords- buffer management; self-similarity; fairnes; packet loss.

I. INTRODUCTION

The growth of robust networks, such as the Internet, rests heavily upon these networks' modeling. Studies in Ethernet traffic, both local and wide area, have shown that network traffic is self-similar. Current models, like "Poisson-like" models, show that network traffic is smooth with predictable bursts. Inaccurate modeling of network traffic may lead to problems of performance budget loss.

One of the effects of self-similarity on networks is increase of packet loss [1]. The solution in order to curtail the effects of self-similarity is structural resource allocation. Bandwidth and buffer sizes are two structural sources of high importance. Bandwidth can be increased so that the burst of traffic can be swallowed. This accounts for the retransmission of packets as well. However, took down the network, additional bandwidth is wasted.

The second alternative for structural resource allocation is the buffer sizes in routers. Expanding the capacity would decline the rate of packet loss due to queues are at maximum capacity [3].

Congestion occurs in the network when link bandwidth exceeds the capacity of the router is available, in situations where the packets don't have to be resent; the quality of service is reduced. This results in long delay in data delivery and wasting of resources due to lost or dropped packets. The main function of a router is to switch packets from incoming links to outgoing links via buffer. Apart from forwarding packets, routers are involved for congestion control in network. It is known from [2] that routing algorithms on two main concepts focus queue management and scheduling.

Queue management algorithms direct the length of packet queues via dropping packets whenever necessary, whereas scheduling algorithms determine which packets must be sent next. Scheduling algorithms are used primarily to manage the allocations of bandwidth among various flows. This reduction in sending rate translates into a decrease in the incoming packet rate at the router, which effectively allows the router to clear up its queue. When rate outgoing packets higher than the rate of incoming packets, router queue size increases gradually and is lined completely.

In routing the packets there is a need to tradeoff between delay and throughput. If the queue is full or almost full, an arriving burst will cause multiple packets to be dropped. This can result in global synchronization of flows throttling back, a sustained period of lowered link utilization, reducing overall throughput. The point of buffering in the network is to absorb data bursts and to transmit them during the ensuing bursts of silence. This is essential to permit the transmission of bursty data [4]. Since, the more the traffic is self-similar, the more it needs buffer, in this paper, we have proposed an algorithm for allocation of buffer which takes into account the degree of traffic self-similarity.

The rest of the paper is organized as follows. Section Π defines self-similarity, section Ш defines Heavy-Tailed Distribution, section IV is about dependence between self-similarity and parameter α of Pareto Distribution. Section V describes the proposed algorithm and finally presents the results and some discussions.

II. SELF-SIMILARITY

In the past, so many descriptions for modeling the traffic network were developed. Next the traditional traffic models such as Poisson and Markov were replaced with these new models. For example, the reason of using Poisson process is it gave a well approximation of telephone network. Particularly, it worked well when it was used for describing times between

each call and also the duration of each calls. For describing it, exponential probability distribution is used, which is indicated by λ (i.e. the number of events per second).

Nowadays, Self-similarity is the most important model that can be used for modeling the network traffic. It is in contrast to Poisson model, a traditional traffic model, which becomes very smooth during the aggregation process.

Here, we want to present a discussion of second-order self-similarity as a statistical property of time series. Intuitively, the self-similar phenomena can display structural similarities across a numerous of time scales. The degree of self-similarity is specified by measuring one or more parameters called Hurst parameter(s).

Suppose, we have $X = (X_k : k = 0, 1, 2, \dots)$ as a covariance stationary stochastic process for which μ is mean, σ^2 is variance, and $R(k)$ for $k \geq 0$ is autocorrelation function. Also, suppose that the autocorrelation function of X has the following form:

$$R(k) \sim \eta_1 k^{-\beta}, \text{ as } k \to \infty \quad (1)$$

where $0 < \beta < 1$. (constants $\eta_1, \eta_2, \dots$ are finite positive integers.) For every $m = 1, 2, \dots$ let $X^{(m)} = (X^{(m)}{}_k : k = 1, 2, 3, \dots)$ be the covariance stationary time series with corresponding autocorrelation function $R^{(m)}$ that is obtained by calculating the average of the original series X over the non-overlapping time periods of size m. That is, $X^{(m)}$ for every $m = 1, 2, \dots$ is as follow:

$$X_k^{(m)} = \frac{1}{m}(X_{km-m+1} + \dots + X_{km}), k \geq 1 \quad (2)$$

Assuming $H = 1 - \beta/2$ as the self-similarity parameter, the process X is called exactly second-order self-similar if the corresponding $X^{(m)}$ processes have the same correlation functions as X. In another words, $R^m(k) = R(k)$ for all $m = 1, 2, \dots$ and $k = 1, 2, \dots$. Moreover, the process X is called asymptotically second-order self-similar if asymptotically approaches to $R(k)$ given by (1), for large values of m and k. If the correlation functions of the aggregated processes $X^{(m)}$ are the same as the correlation functions of X or approach asymptotically to the correlation function of X, then X is called exactly or asymptotically second-order self-similar.

III. HEAVY-TAILED DISTRIBUTATIONS

Another property of the process is called long-range dependence (LRD) which satisfies the autocorrelation function in the most common definition of self-similarity [5]. Long-range dependence of the process is characterized by a slowly decaying autocorrelation function and the non-sum able autocorrelation function. By increasing the k, the autocorrelation function decays hyperbolically. This is opposed with the property of short-range dependence (SRD), where the autocorrelation function decay is exponential.

There is a common relation between short-range and long-range dependences with the value of the Hurst parameter of the self-similar process [5]:

$0 < H < 0.5 \rightarrow$SRD
$0.5 < H < 1 \rightarrow$LRD

Network traffic is a stochastic process, and is described as such. Self-similarity and long-range dependence (LRD) properties are described by heavy-tailed distributions. The heavy-tailed distributions (Pareto) are depicted as hyperbolic, while this is not true for light-tailed distributions (exponential distribution) where distributions decay exponentially.

In this paper we use the distributions with the property of being *heavy-tailed.* We say a distribution is heavy-tailed if

$$P[X \geq x] \sim x^{-\alpha}, \text{ As} x \to \infty, \ 0 < \alpha < 2 \quad (3)$$

This means that, regardless of the behavior of the distribution for small values of the random variable, if the asymptotic shape of the distribution is hyperbolic, it is considered as heavy-tailed. Based on our best knowledge, the *Pareto* distribution is the simplest heavy-tailed distribution. It is hyperbolic over its entire range. The probability mass function of *Pareto* distribution is

$$p(x) = \alpha k^{\alpha} x^{-\alpha-1} \alpha, \ k > 0 \ \ x \geq k \quad (4)$$

Also its cumulative distribution function is as follows:

$$F(x) = P[X \leq x] = 1 - \left(k/x\right)^{\alpha} \quad (5)$$

In the above equation, the parameter **k** represents the smallest possible value of the random variable. Pareto distributions have some properties that qualitatively differ from distributions more commonly encountered such as the exponential distributions, normal distributions, or Poisson distributions [6].

IV. DEPENDENCE ON SELF-SIMILAR OF GENERATED FLOW FROM PARETO DISTRIBUTION WITH PARAMETER α

In this paper, Pareto distribution has been used for traffic generation. Therefore, in this section, it would be shown that the traffic generated via Pareto distribution is self-similar and the degree of its similarity is correlated with alpha parameter.

There exist a few methods for testing flow on self-similarity [5], [7]. The variance-plot analysis is based on property of slowly decaying variance of self-similar processes undergoing aggregation. In this process the variance of an aggregate (exactly or asymptotically) a self-similar process is [defined as:]

$$Var[X^{(m)}] = Var[X]/m^{\beta} \quad (6)$$

$$\text{Where } H = 1 - \beta/2 \quad (7)$$

The log $\{Var[X^{(m)}]\}$ is a constant independent of m. If one plots $Var[X]$versus m on a log-log graph and by fitting a least-square line through the resulting points (and also ignoring the values for m value near 0), then the result should be a straight line with a slope of $-\beta$. The values of the slope in-between (-1; 0) suggest self-similarity. The values are represented in table 1 and figure 1.

For testing dependence of self-similarity from Pareto distribution parameter α, four models are created with different values of parameter α = 1, 1.3, 1.6, 1.9. Table 2 and Fig. 2 depict the results.

Table I. Variance-plot analysis

m	Log (m)	Log (Var [X])
1	0.0000	0.51867
2	0.3010	0.52311
3	0.4771	0.48436
5	0.6989	0.41862
10	1.0000	0.45181
20	1.3010	0.47864
50	1.6989	0.41123
100	2.0000	0.41054
200	2.3010	0.34045
500	2.6989	0.38532
1000	3.0000	0.41071

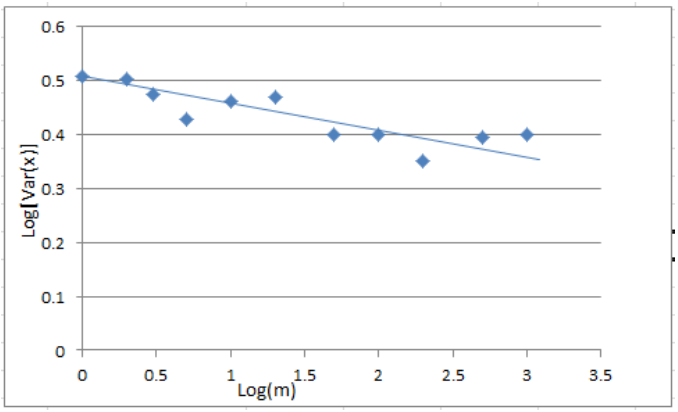

Figure I. Variance-plot analysis

Table II. Dependence of self-similarity on parameter α

α	$-\beta$	H
1	-0.03	0.98
1.3	-0.13	0.93
1.6	-0.24	0.88
1.9	-0.32	0.84

As the graph above indicates, self-similarity depends on the characteristics of ON/OFF periods, and with $\alpha \to 1$ traffic becomes more self-similar than with α value greater than 1. As described above, variance of Pareto distribution becomes infinitive when $\alpha \to 1$ and the process becomes more heavy-tailed.

Considering the findings of the study, it can be concluded that alpha parameter can be used for determining the degree of traffic self-similarity. Therefore, for generation of self-similar traffic via Pareto distribution in next section, alpha parameter would be used to determine the degree of self-similarity of the generated traffic.

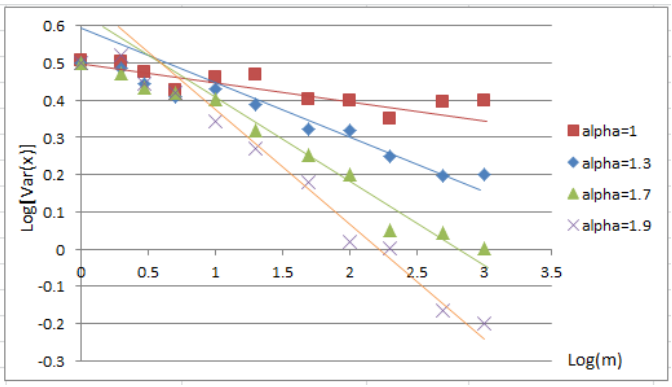

Figure II. Dependence of self-similarity on parameter α

V. QUEUE MANAGEMENT

In static queue management, no prediction about happening of congestion in future is done. Buffers have fixed lengths and limited capacities. They receive all input packets until they become full. As soon as their capacities become full, they reject all input packets until new capacity become available. Various algorithms proposed in this field have tried to either make queue management fairer or to reduce complexity and increase efficiency.

FIFO algorithm (drop tail when full) is the first algorithm used in routers. In this method, input packets are added to the end of the queue and when capacity for sending becomes available, packets are sent from the beginning of the queue. When the queue becomes full, received packets are dropped. One of the weaknesses of FIFO queuing is that it does not distinguish among packets received from different sources so there is possibility that some of these sources occupy much of the band width of a router.

FQ[1] algorithm has tried to solve this problem [8], by allocating a distinct queue for each flow and sending packets from theses queues is round robin. Therefore, no flow can occupy more band width than it allocated share. In case of congestion event, only those packets belonging to flows which have caused congestion are dropped and their influence on

[1] Fair Queuing

other flows is prevented. FQ provides several important advantages over the usual FIFO queuing algorithm: fair allocation of bandwidth, lower delay for sources using less than their full share of bandwidth, and protection from ill-behaved sources. Although FQ algorithm is advantageous in case of solving congestion problem, because of its complexity, it cannot be used in networks with speed.

We have proposed a new algorithm for queue management with two purposes of increasing efficiency and fair. In this section, algorithm is introduced then method of implementation and resulted findings would be presented.

A. PROPOSED ALGORITHM

In this paper, we have suggested a queue management algorithm for self-similar traffic which is supposed to have advantages of fair queuing and its implementation is simple. In this algorithm, for allocation of buffer to the flows, degree of self-similarity is taken into account. Three distinct queues with different sizes are considered for the flows; because of alpha parameter would be used to determine the degree of self-similarity of the generated traffic, first queue with 2/3 of the size of allocated buffer is for those flows whose is more self-similar and alpha parameter is between 1.7 And 1.9. Second queue with 1/3 of buffer size are for those flows whose degree of self-similarity is between 1.5 and 1.7. Finally, third queue with 1/3 of buffer size are for those flows whose degree of self-similarity is between 1.0 and 1.5.

Since the more the traffic is self-similar, the more it needs buffer, we allocate each flow to a distinct queue by considering its degree of self-similarity and sending packets from theses queues is round robin. Therefore, no flow can occupy more band width than it allocated share. In case of congestion event, only those packets belonging to flows in queue which have caused congestion are dropped and their influence on other queues is prevented.

VI. PERFORMANCE EVALUTION

To evaluate the performance of our algorithm, we use Network Simulator ns2 to simulate a series of scenarios. The symmetrical network topology has been chosen for the simulation. As shown in Fig. **III**, **IV**, **V** there are n connections in the network. N is a variable parameter shows the number of connections that are connected to the bottleneck link. we choose n within 16, 32 & 64.

. The larger is the number of connections; the worse is the congestion in the bottleneck. The bottleneck link's bandwidth is 10Mbps, and the n connections' output are all 10Mbps, the data transfer time is 150ms (Table 1).

For generating traffic, we used Pareto distribution in NS2 and for generating self-similar traffic, α parameter has been given to each source randomly (between 1 and 2) and each source started generating traffic at random time. NS2 includes code to simulate TCP protocol, we defined TCP agent in NS2.

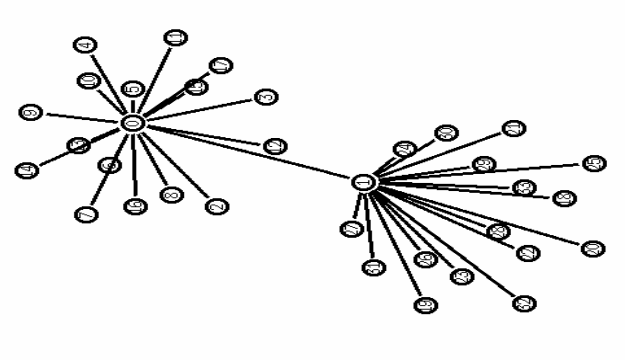

Figure VI. Simulation model of for 16 node network

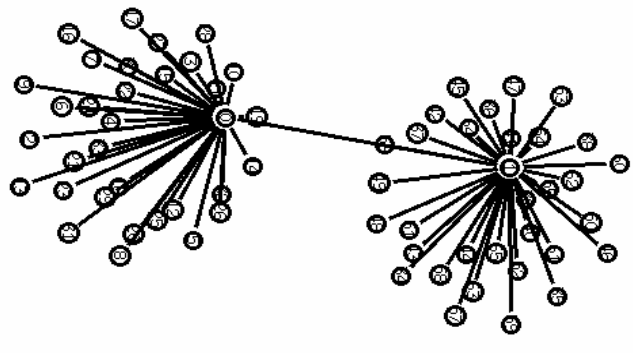

Figure VII. Simulation model of for 32 node network

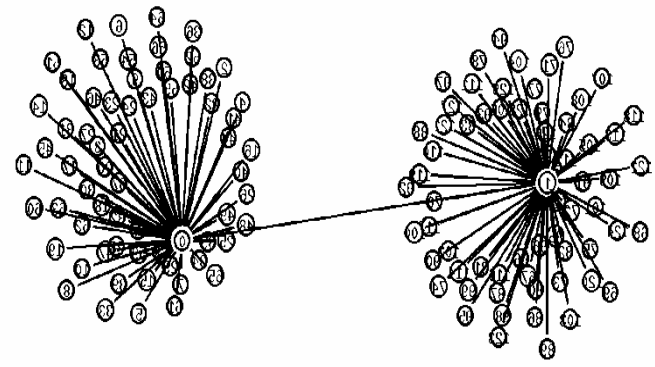

Figure VIII**.** Simulation model of for 64 node network

Table III. Simulation Parameters

Botteleneck link bandwidth	10 [Mbit/s]
Delay of the Botteleneck link	150 [ms]
Pakhet Size	1,000 [byte]
Buffer Size	100 [pakhet]
Rate	2 [M]

Table IV. Throughput

Proposed algorithm	DRR	SFQ	RED	Drop Tail	Queue Size	nodes
99.28	99.25	99.04	99.30	99.33	25	16
99.28	99.25	99.04	99.61	99.58	100	
99.28	99.25	99.04	99.63	99.74	250	
98.66	98.58	98.05	97.93	98.69	25	32
98.66	98.58	98.05	98.91	98.80	100	
98.66	98.58	98.05	98.92	99.31	250	
98.66	96.41	96.12	96.67	96.42	25	64
98.66	96.41	96.12	97.89	97.42	100	
98.66	96.41	96.12	97.60	98.13	250	

Table V. Loss Rate

Proposed algorithm	DRR	SFQ	RED	Drop Tail	Queue Size	nodes
0.72	0.75	0.96	0.7	0.67	25	16
0.72	0.75	0.96	0.39	0.42	100	
0.72	0.75	0.96	0.37	0.26	250	
1.34	1.42	1.95	2.07	1.31	25	32
1.34	1.42	1.95	1.09	1.2	100	
1.34	1.42	1.95	1.08	0.69	250	
1.34	3.59	3.88	3.33	3.58	25	64
1.34	3.59	3.88	2.11	2.58	100	
1.34	3.59	3.88	2.4	1.87	250	

We compared our algorithm with The Drop-tail and SFQ, DRR, RED that are the currently implemented buffer management algorithms in ns2. This section describes the NS2 simulation details associated with our comparison study of the performance of our algorithm. The NS2 simulator provides the ability to simulate Drop-Tail, RED, FQ, DRR routers. With different queue size 25, 100 and 250 packets.

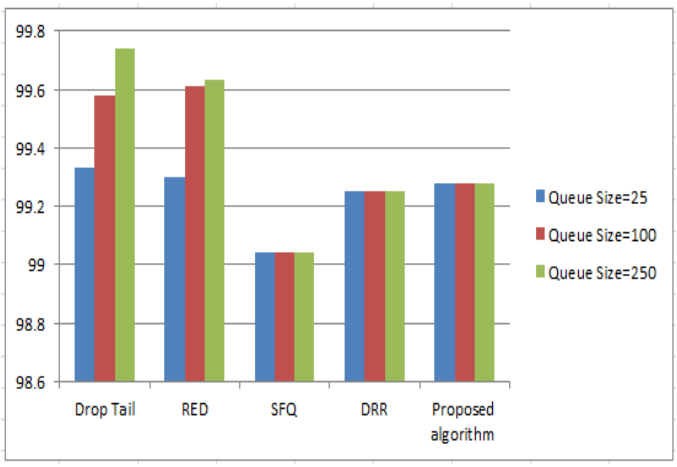

Figure IX. Throughput with 10 sec simulation Time with 16 nodes

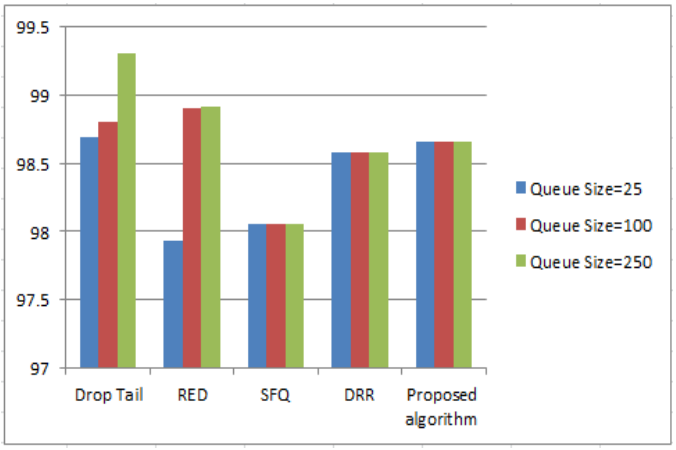

Figure X. Throughput with 10 sec simulation Time with 32 nodes

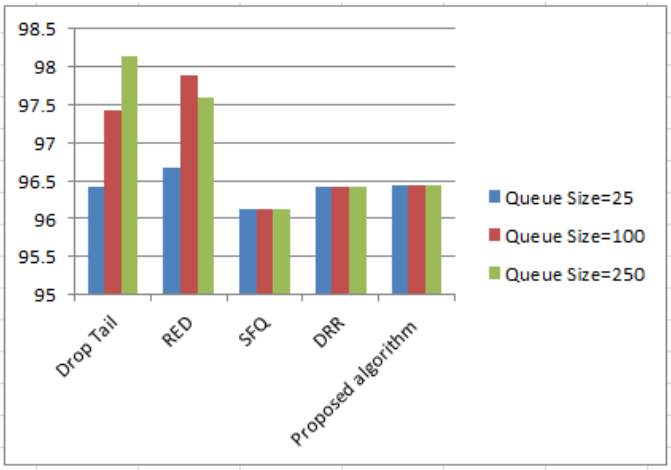

Figure XI. Throughput with 10 sec simulation Time with 64 nodes

Figures 6-8 shows the loss rate and throughput of simulation results of Drop-Tail, RED, SFQ, DRR queues of 25, 100, and 250 packets. As shown, when the queue length is increased from 25 to 100 packets as well as from 100 to 250 packets, the packet loss rate will be reduced. These results show the trade-off between small queue length (25 and 100 packets) and large queue length (250 packets). A small drop-tail queue obtains small queuing delay but results a high packet loss rate (for example, the packet loss rate was 3.58 % for a queue of 25 packets but only 1.87 % for a queue of 250 packets at 64 node network). Because of this, flows that do not experience any packet loss enjoy good response times with a small drop-tail queue, especially those that are small. On the other hand, the more the size of drop-tail queue can reduce the packet loss rate but subjects flows to more queuing delay.

Hence, for large flows that dominate the links (in terms of number of packets) and are likely to experience some packet loss (assuming that packets are dropped randomly and uniformly), the impact of increased queuing delay is out weighted by the effects of reduced packet losses. Thus, large flows receive better performance under a large drop-tail queue.

As you know, network performance will be decreased after the increasing of nodes. According to these figures, in all these queues, we witness such situation.

The performance of our proposed algorithm in all 3 networks is better than SFQ and DRR static queues, but it is worse than drop-tail and RED queues. In contrast to RED and drop-tail queues, we have used more than one buffer for queuing of packets (like SFQ, DRR), so buffer increasing does not have an obvious influence on network performance.
Because classification has been used for queuing, this algorithm offers more fair (balance) among the flows in comparison to RED and drop-tail and prevents influence of other flows in case of congestion.

VII. DISCUSSION

In this paper, the simulation results with different traffic and a uniform RTT distribution were presented for Drop tail, RED, SFQ, DRR of Queue Managements that have been currently implemented in ns2 as well as our algorithm.

We have proposed a new algorithm which does not require modification to all end system TCP/IP stacks but can be solely implemented in routers. Our scheme helps to reduce dropped packets in comparison to SFQ and DRR static algorithms, the impact of increased queuing delay is out weighted by the effects of reduced packet losses, but it is worse than drop-tail and RED queues.

We think that low efficiency of this algorithm in comparison to RED is due to the fact that it is static. Also the length of buffer in all three queues is fixed. Therefore, if queue becomes full, it will drop the packets like FIFO.

Finally, it's noticeable that the proposed algorithm is just a simple and novel idea. We aim to improve it in near future and make it more efficient.

ACKNOWLEDGEMENTS

This research was sponsored by the Iran Telecommunication Research Center.

REFERENCES

[1] Leland, W. E, Taqqu, M. S., Willinger, W., Wilson, D. V. (1994). On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM Transactions on Networking*, 2 (1), 1-15.

[2] Braen,B., Clark,D.,et.al "Recommendations on queue management and congestion avoidance in the Internet" IETF RFC (Information)2309.April 1998.

[3] Park, K., Kim, G., Crovella, M. (1997). On the Effect of Traffic Selfsimilarity on Network Performance. *In Proceedings of SPIE International Conference on Performance and Control of Network Systems*. 1-39.

[4] G.Thiruchelvi, J.Raja. "A Survey On Active Queue Management Mechanisms". IJCSNS.VOL.8 No.12,December 2008.

[5] K. Park in W. Willinger, *Self similar network traffic and performance evaluation,* Wiley, 2000.

[6] Mark E. Crovella, Azer Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Transactions on Networking, Vol. 5, NO. 6, Dec 1997.

[7] M. Gospodinov in E. Gospodinova, "The graphical methods for estimating hurst parameter of self similar network traffic", International conference on computing systems and tehnology – CompSysTech' 2005.

[8] Demers, A., Keshav, S. and Shenker, S., "Analysis and Simulation of a Fair Queueing Algorithm", Journal of Internetworking Research and Experience, September 1990, 3-26;. also *In Proceedings of the ACM SIGCOMM '89* , pp1-12, Sept. 1989.