Effects of Upper-Layer Protocols on Self-Similarity of Network Traffic

Yoshiaki Sumida, Hiroyuki Ohsaki, Masayuki Murata and Hideo Miyahara
Graduate School of Engineering Science, Osaka University

***Abstract*—It has been pointed out by many researchers that network traffic in LAN and WAN environment has self-similarity. However, it has not been clear why self-similarity is observed in the network traffic. Also, its impact on network performance has not been fully investigated. In this paper, we investigate cause of traffic self-similarity in a network employing TCP/IP as its upper-layer protocols through simulation experiments. We further investigate effect of the TCP mechanism on self-similarity of the network traffic, and effect of self-similarity of the network traffic on the users' QoS.**

I. Introduction

Markovian models have been widely used as an analytic model of network traffic because of its analytical tractability. The main advantage of such Markovian models is in its memoryless property; that is, future events are completely independent from past ones. However, recent research on the real Ethernet traffic has shown the existence of *self-similarity* or *long-range dependence*, which cannot be modeled by those Markovian models [1], [2]. In [3], Garrett *et. al* have been pointed out that VBR (Variable Bit Rate) video traffic also possesses self-similarity. Several researches are actively undergoing to develop an analytic model of network traffic with self-similarity, with which effects of self-similarity on the network performance could be evaluated. For example, Willinger *et. al* have explained in [4] that self-similarity of network traffic is caused by multiplexing a number of traffic streams, each of which is modeled by ON/OFF source where the length of ON or OFF period follows heavy-tailed distribution.

However, the "discovery" made in [4] and their following papers is just an observation in the network, and it has not been fully discussed why the network traffic on Ethernet shows self-similarity. It seems true that the source of the network traffic itself has the cause of self-similarity; for example, heavy-tailed distribution of document size [5]. However, self-similarity of the network traffic observed *in the network* might be affected by the network protocols such as a TCP (Transmission Control Protocol) mechanism, which has a feedback congestion control mechanism. Accordingly, the authors in [6], [7] have investigated through simulation experiments and have indicated that the degree of self-similarity of network traffic is changed by the TCP mechanism. Moreover, they show that network performance is degraded when network traffic has strong self-similarity. However, their study on the effects caused by upper-layer protocols is limited; for example, it is not clear which part of the TCP mechanism (e.g., a window-based flow control or a packet retransmission mechanism) affects the intensity of self-similarity of the network traffic. Furthermore, it has not also been cleared how various network parameters (i.e., offered traffic load, a link bandwidth, and a buffer capacity of switch) affect self-similarity of traffic observed in the network.

Another and more important problem is how self-similarity of network traffic gives impact on users' QoS (Quality of Service). QoS provided to users is one of the most important performance measures. Previous studies focusing on only packet-level and first-order performance measures (e.g., an average packet delay and an average packet loss probability) are apparently insufficient. The main subject of this paper is, therefore, to investigate the effects of self-similarity of the network traffic on users' QoS. In our simulation setting, we consider a TCP-based network and a file transfer service as its typical application. As QoS measures, we use effective throughput and delay variation of a file transfer.

Based on the above discussion, we investigate effects of the TCP mechanism as the upper-layer protocol on self-similarity of the network traffic observed in the network through simulation experiments. Namely, we investigate (1) why the network traffic exhibits self-similarity, (2) how the upper-layer protocol affects self-similarity of the network traffic, and (3) how self-similarity of the traffic and the upper-layer protocol affect users' QoS. As a simulation model, we use a server-client network model that consists of two servers and 32 clients contending for a single bottleneck link. Each client requests a file transfer from one of two servers using the TCP mechanism. By changing characteristics of the source traffic (i.e., file size distribution), we evaluate its effect on self-similarity of the traffic observed at the bottleneck link. We then change various system parameters such as offered traffic load, bandwidth of the bottleneck link, and a buffer size of the switch, to investigate relations between self-similarity of the traffic and those system parameters. For this purpose, we compare simulation results with and without the TCP mechanism as in [6], [7], and show effects of upper-layer protocols on self-similarity clearly.

This paper is organized as follows. In Section II, we give a brief definition of self-similarity and three estimation methods of self-similarity from measured data. In

Department of Informatics and Mathematical Science Graduate School of Engineering Science, Osaka University,
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
(Phone) +81-6-6850-6588 (Fax) +81-6-6850-6589
(E-mail) {y-sumida,oosaki,murata,miyahara}@ics.es.osaka-u.ac.jp

Section III, we describe our simulation model in detail. In Section IV, we investigate effects of the TCP mechanism on self-similarity of the network traffic and users' QoS through various simulation results. Finally, we summarize this paper with a few remarks in Section V.

II. Self-Similarity

In this section, we briefly present the definition of self-similarity, and introduce three methods to estimate the intensity of self-similarity from observed data. See, for example, [8], [9] for more detail.

A. Definition

Let X be a stochastic process on a discrete-time, i.e., $X = \{X_t : t = 0, 1, 2, \ldots\}$. Let $X^{(m)}$ be a stochastic process generated from X by aggregating m events of X, i.e., $X^{(m)} = \{X_n^{(m)} : n = 0, 1, 2 \ldots\}$, where

$$X_n^{(m)} = \frac{1}{m} \sum_{j=(n-1)m+1}^{nm} X_j. \qquad (1)$$

When the auto-correlation function of $X^{(m)}$, $r^{(m)}(k)$, satisfies $r^{(m)}(k) \sim k^{-(2-2H)}$ for $1/2 < H < 1$, X is defined to possess (asymptotically second-order) *self-similarity* ($\sim$ denotes that both sides are asymptotically proportional to each other when $k \longrightarrow \infty$). In the above equation, H is called *Hurst parameter* that defines the intensity of self-similarity. It is known that asymptotically second-order self-similarity is equivalent to the long-range dependence [10].

B. Estimation Methods of Hurst Parameter

Several methods for confirming existence of self-similarity in observed data, and for estimating its Hurst parameter have been proposed in the literature [8], [11], [12]. However, there is no generic and statistically rigorous one. We therefore use three methods to determine the value of the Hurst parameter in Section IV: a variance-time plot, an R/S plot, and a Whittle's estimator.

In the variance-time plot, an aggregated process $X^{(m)}$ defined by Eq. (1) is first computed from the measured process X. If $X^{(m)}$ has self-similarity, it satisfies the following property [8], [12], [13]:

$$\mathrm{Var}[X^{(m)}] \approx a \cdot m^{2H-2},$$

where a is a constant. Therefore, when $X^{(m)}$ has self-similarity, a slope of $\log(\mathrm{Var}[X^{(m)}])$ as a function of $\log m$ converges to $(2H-2)$ for a sufficiently large m.

In the R/S plot, the following equation is first computed from the measured process X:

$$\frac{R(n)}{S(n)} = \frac{\max(0, W_1, \ldots, W_n) - \min(0, W_1, \ldots, W_n)}{S(n)},$$

where $W_l = (X_1 + X_2 + \cdots + X_k) - k\overline{X}(n)$. In the above equation, $\overline{X}(n)$ and $S(n)$ are the average and the standard deviation of the process X, respectively. If X has self-similarity, the average of $R(n)/S(n)$ should converge to $a \cdot n^H$, where a is a constant [8], [12], [13]. By using this property, the Hurst parameter can be obtained by plotting $\log(R(n)/S(n))$ against $\log n$.

Although the variance-time plot and the R/S plot are useful for rough estimations of the existence of self-similarity in the measured data, these methods are not statistically rigorous. On the contrary, the Whittle's estimator can be used to obtain the Hurst parameter with its confidence interval in a statistically rigorous manner. However, the Whittle's estimator assumes the underlying process be a Gaussian process. Let $f(\lambda; z)$ and $I(\lambda)$ be a spectral density function and a periodgram of X, respectively. A value of z that minimizes $W(z)$ given by

$$W(z) \quad = \quad \int_{-\pi}^{\pi} \frac{I(\lambda)}{f(\lambda; z)} d\lambda$$

is then computed, and the Hurst parameter, H, is determined by z [11]. If the process X is not Gaussian, $X^{(m)}$ for large m is used instead because $X^{(m)}$ converges to be a Gaussian process as m goes infinity.

III. Simulation Model

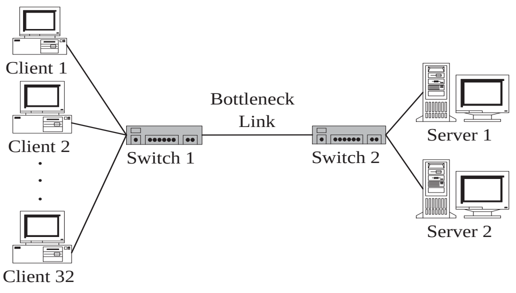

Fig. 1. Simulation model.

Figure 1 illustrates our simulation model, which is a server-client model where 32 clients are connected to two servers via a single bottleneck link. We use a simulation package called `ns` [14] by modifying some codes. In our model, each client requests a file transfer to a randomly selected server, and the server sends a file back as a series of fixed-size (i.e., 1 Kbyte) packets. As an upper-layer protocol, Reno version of TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) are used. The average file size is fixed at 22 Kbyte. The average request interval, which is duration from termination of the previous file transfer to occurrence of the next file transfer request, is set to 3.2 s. Either the exponential distribution or the Pareto distribution is used as the distribution function of file sizes, and the exponential distribution is used for request intervals. The probability density function of the Pareto distribution is defined by

$$P[X \le x] \quad = \quad \begin{cases} 1 - (k/x)^{\alpha} & k < x, \\ 0 & x \le k. \end{cases} \qquad (2)$$

Note that the Pareto distribution is *heavy-tailed* distribution. The tail part of the Pareto distribution becomes

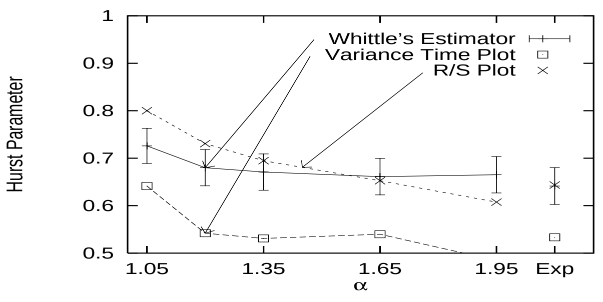

Fig. 2. Hurst parameter vs. α (UDP).

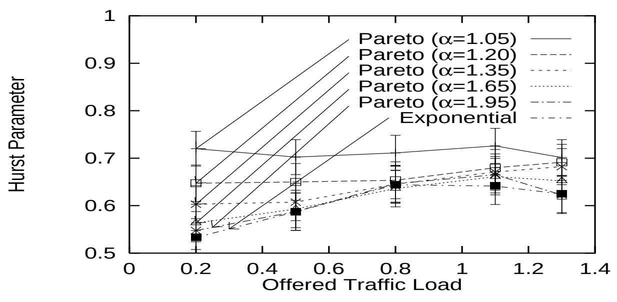

Fig. 3. Hurst parameter vs. offered traffic load (UDP).

large as the shape parameter, α (> 0), gets close to 1 [15], and k specifies the minimum value generated by the Pareto distribution. Since we fix the average file size to 22 Kbyte in simulation experiments, the value of k varies for a different value of α. For example, we use $k = 2,151$ for $\alpha = 1.05$ and $k = 10,975$ for $\alpha = 1.95$.

The following is known: when infinite ON/OFF sources, where an interval of either ON or OFF period follows the Pareto distribution with parameter α, are multiplexed, the superposed process becomes a FGN (Fractional Gaussian Noise) and the Hurst parameter of this FGN is given by $(3 - \alpha)/2$ [4]. This suggests that the aggregated traffic in our simulation model observed *before* the bottleneck link is approximated by FGN. In Section IV, we will compare the estimated value of the Hurst parameter with the theoretical value of $(3 - \alpha)/2$.

We note that document size distribution of WWW (World Wide Web) traffic is approximated by the Pareto distribution with $\alpha = 1.12$ [16]. Moreover, by using the Least Squares Estimator [15], we have found that file size distribution of a UNIX operating system [5] is approximated by the Pareto distribution with $\alpha = 0.685$, and its tail distribution (over 2 Mbyte) is approximated by the Pareto distribution with $\alpha = 1.294$.

In our simulation, we change the buffer size of the switch from 4 Kbyte to 512 Kbyte, and the bottleneck link bandwidth from 150 Kbit/s to 15 Mbit/s. The bandwidths of all other links are fixed at 10 Mbit/s. Since the link between switches is bottleneck in our simulation, we focus on the throughput at the bottleneck link (i.e., the output link of Switch 2 in Fig. 1) for a network performance measure. We also change the offered traffic load from 0.2 to 1.3 by adjusting the average request interval. We note that the traffic load is defined as a ratio of the amount of traffic arriving at the bottleneck switch, when no flow control mechanism is provided by the upper-layer protocol, to the bandwidth of the bottleneck link. Thus, actual traffic load on the bottleneck link is affected by the window flow control and packet retransmission mechanisms of the TCP mechanism.

IV. Simulation Results

A. Case of UDP as the Upper-Layer Protocol

1) Effects of File Size Distribution: In this subsection, we consider the case where the UDP mechanism is applied as the upper-layer protocol. In this case, neither flow control nor packet retransmission mechanism is employed. We set the buffer size of the switch to 128 Kbyte, and the bandwidth of the bottleneck link to 1.5 Mbit/s. The offered traffic load is 1.1, which is the normalized value by the bottleneck link bandwidth. In Figure 2, we show the Hurst parameter of the observed traffic at the bottleneck link for different values of α ($1 < \alpha < 2$), which determines the tail part of the file size distribution (see Eq.(2)) [15]. We also show the Hurst parameter when file sizes are exponentially distributed (labeled as "Exp" in the figure). We show Hurst parameters obtained by the variance-time plot, the R/S plot, and the Whittle's estimator. For the Hurst parameter estimated by the Whittle's estimator, its 95% confidence interval is also shown.

When the UDP mechanism is adopted as the upper-layer protocol, the traffic observed at the bottleneck link is modeled by multiplexed ON/OFF sources where the ON interval is determined by the file size distribution and the OFF interval is by the request interval distribution. It is expected that the Hurst parameter of such traffic becomes $(3-\alpha)/2$ when the ON interval follows the Pareto distribution, and 0.5 when both of ON and OFF intervals follow the exponential distribution [9]. However, as can be found from Fig. 2, the Hurst parameters of the UDP case are very different from the theoretical value. When the file size distribution follows the Pareto distribution, Hurst parameter becomes small as α increases. However the slope is smaller than $(3-\alpha)/2$. And when both file size distribution and request interval distribution follow the exponential distribution, Hurst parameter is about 0.64, being slightly larger than 0.5. This difference is caused by the buffer overflow at the bottleneck switch. UDP provides no congestion control mechanism so that many packets are lost (more than 10%) at the bottleneck switch, and packet losses occur intermittently. Thus, the traffic observed at the bottleneck link shows rather short-term dependency, leading to the modest level of self-similarity.

2) Effects of Offered Traffic Load: To validate above observations, we next show estimated values of the Hurst parameter and the packet loss probability at the switch in Figs. 3 and 4, respectively, where the offered traffic load is varied from 0.2 to 1.3. Figure 3 indicates that the Hurst parameter is not affected by the offered traffic load when the file size distribution follows the Pareto distribution with a small value of α. However, in the case of large α or exponential file size distribution, there exists a tendency that the Hurst parameter increases as the offered traffic load gets large, converging to about 0.65. This is because characteristics of the network traffic observed at

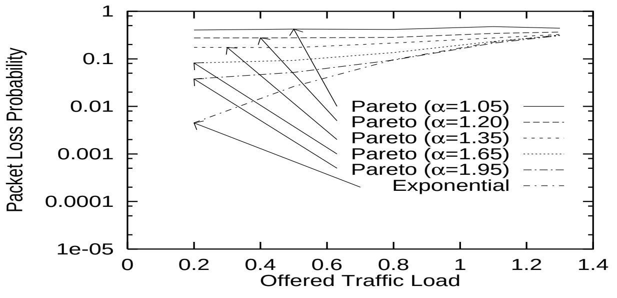

Fig. 4. Packet loss probability vs. offered traffic load (UDP).

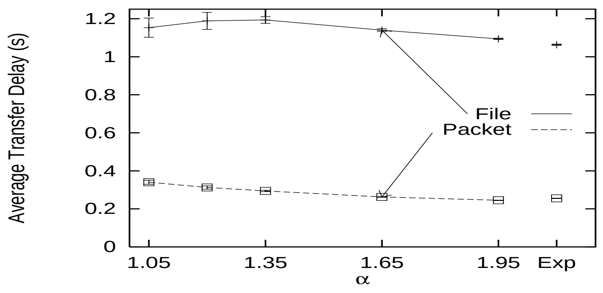

Fig. 6. Average packet transfer delay vs. α (TCP).

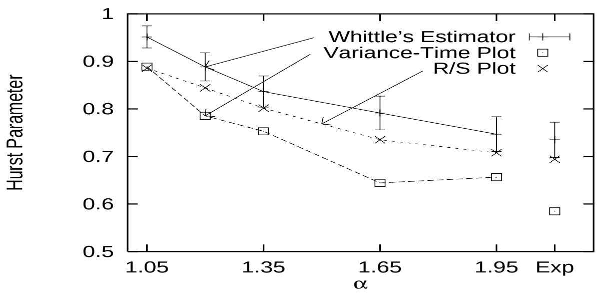

Fig. 5. Hurst parameter vs. α (TCP).

the bottleneck link are almost determined by the pattern of packet losses occurring at the switch buffer; when α is small, more than 10% of packets are lost in the switch buffer regardless of the offered traffic load (see Fig. 4). So the Hurst parameter is mostly dependent on the pattern of packet losses. In the case of large α or the exponential file size distribution, however, packet loss probability is rather small when the offered traffic load is not high. The Hurst parameter is therefore close to the theoretical value of $(3-\alpha)/2$. However, packet loss probability increases when the offered traffic load is high, leading the Hurst parameter around 0.65.

B. Case of TCP as the Upper-Layer Protocol

In this subsection, we show simulation results when the TCP mechanism is used as the upper-layer protocol. It is noted that Figs. 5, 8, and 9 presented in this section correspond to Figs. 2 through 4 in Section A, respectively. Note that "packet loss probability" (i.e., the buffer overflow probability at the bottleneck switch while the TCP mechanism recovers lost packets) increases up to about 0.1% as α decreases.

1) Effects of File Size Distribution: We first show relation between α of the Pareto distribution for file sizes and the Hurst parameter of the traffic observed on the bottleneck link in Fig. 5. By comparing Fig. 5 with Fig. 2 of the UDP case, it can be found that the Hurst parameter gets larger particulary when α is close to 1. Note that the theoretical value of the Hurst parameter of the FGN given by $(3-\alpha)/2$ is according to the line obtained by the Whittle's estimator. This indicate that the self-similarity of the incoming traffic (i.e., aggregated traffic before the bottleneck link) is preserved in the measured traffic on the bottleneck link. Namely, since packet loss probability is greatly reduced because of the window flow control of the TCP mechanism, self-similarity of the incoming traffic is left unchanged.

In the figure, it can also be found that the Hurst parameter in the case of the exponentially distributed file size is around 0.73, which is larger than that of the UDP case. This is because of packet retransmissions of the TCP mechanism. Namely, when some of packets consisting of a file are lost in the network, lost packets are retransmitted by the TCP mechanism. This effect can be considered as stretching the file size so that the tail of the Pareto distribution is expanded. Consequently, the incoming traffic holds higher self-similarity, leading to a larger value of the Hurst parameter.

We next investigate the packet/file transfer delay from the server to each client. The packet transfer delay includes delays resulted from packet retransmissions by the TCP mechanism. The file transfer delay is the time between when the server sends the first packet of the file and when the server receives the ACK (ACKnowledgement) packet for the last packet from the client. When the file size distribution follows the Pareto distribution with small α, it is expected that the variance of packet/file transfer delay would be large. In the following results, we therefore show the average value of packet/file transfer delays obtained from 30 simulation runs and their 95% confidence intervals.

Figure 6 indicates that the average packet transfer delay increases as α decreases. This can be explained as follows: Packet loss probability increases for a small value of α due to heavy-tailed distribution of file sizes, and then dropped packets are retransmitted by the TCP's mechanism. In other words, the average packet transfer delay and the packet loss probability increase as the self-similarity of the incoming traffic becomes strong even when the TCP mechanism is employed. Therefore, it is expected that the users' QoS (file transfer delay in the current case) would be further degraded when the network traffic has strong self-similarity.

However, the average file transfer delay is *not* affected by α as can be seen in Fig. 6. This phenomenon can be explained as follows. Basically, the smaller file size becomes, the faster file transmission speed becomes when both retransmission and queuing delays are ignored. However, we recall that the TCP mechanism adopts a slow-start mechanism, which exponentially increases its window-size at each receipt of ACK packet. For instance, assuming no

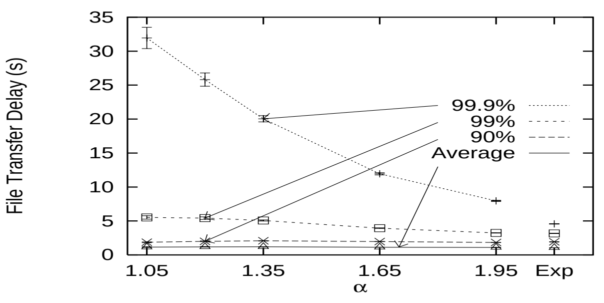

Fig. 7. File transfer delays vs. α (TCP).

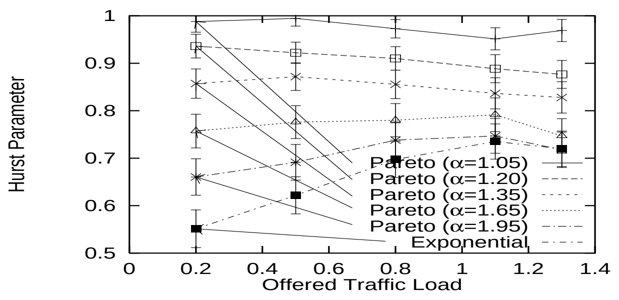

Fig. 8. Hurst parameter vs. offered traffic load (TCP).

packet loss and no queuing delays, average file transfer delay for a file of 3 Kbyte would be twice of the round-trip time. Moreover, for 11 Kbyte and 20 Kbyte files, 4 times and 5 times of the round-trip time would be required. On the other hand, when the file size distribution follows the Pareto distribution with small α, the probability that relatively small size file is generated is larger than that of with large α. Henceforth, it is expected that the average file transfer delay increases as α increases. However, as Fig. 6 indicates, the average file size is not affected by the value of α. It is because the difference in the average file transfer delay for different values of α caused by the TCP's slow-start mechanism is compensated by the difference resulted from the TCP's retransmission mechanism. Thus, we conclude that the average file transfer delay is not so influenced by the existence of the self-similarity of the incoming traffic.

In Figure 7, we show the average file transfer delay for different values of α. In this figure, 90%, 99%, and 99.9% file transfer delays are also shown to investigate effect of self-similarity on the users' QoS. n% file transfer delay is defined as the value that covers n% of all file transfer delays. This figure clearly indicates that n% file transfer delay becomes extremely large when α is close to 1 (i.e., when the incoming traffic possesses strong self-similarity). For example, the 99.9% file transfer delay for $\alpha = 1.05$ is as four times as that for $\alpha = 1.95$. Thus, we conclude that the self-similarity of the network traffic has influence not on the average file transfer delay but on the 99.9% file transfer delay.

When we think of document transfer from WWW servers, the average file transfer delay would not be affected by existence of the self-similarity that the distribution of WWW documents intrinsically holds. However, if high-quality network service is expected, the 99.9% file transfer delay would be of concern in addition to the average file transfer delay. So, the network must be provisioned carefully by taking account of the self-similarity of the network traffic. In other words, to provide better QoS's to users, the buffer size at the switch and the capacity of the bottleneck link bandwidth must be chosen carefully. We investigate effects of the buffer size at the switch and the bottleneck link bandwidth on users' QoS in Section IV-C.

2) Effects of Offered Traffic Load: Figure 8 shows relation between the offered traffic load and the Hurst parameter of the traffic observed at the bottleneck link. It can be found from the figure that the Hurst parameter increases as the offered traffic load increases when α is large. However, when α is small, the Hurst parameter decreases as the offered load increases. This phenomenon can be explained by packet loss probability shown in Fig. 9; when the packet loss probability is large, the traffic pattern on the bottleneck link is almost determined by the pattern of packet losses. Namely, long-range dependence of the bottleneck traffic is lost because of a great number of packet losses so that the Hurst parameter would converge to a fixed value.

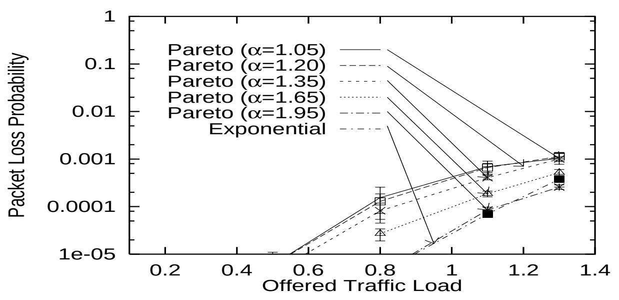

Fig. 9. Packet loss probability vs. offered traffic load (TCP).

We next show the effect of the offered traffic load on the average packet/file transfer delays in Figs. 10 and 11, respectively. We also show relation between the offered traffic load and the 99.9% file transfer delay in Fig.12. In this case, the average file size is set to 22 Kbyte and the buffer size at the switch is to 128 Kbyte. Figure 10 shows that the average packet transfer delay in the case of low traffic load condition is almost independent of α. This is because few packet losses occur at the switch when

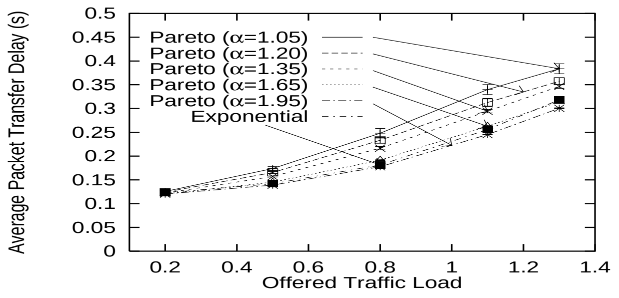

Fig. 10. Average packet transfer delay vs. offered traffic load (TCP).

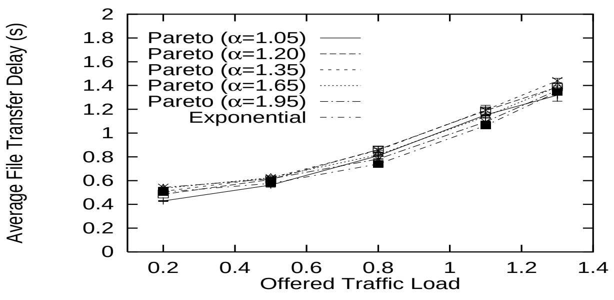

Fig. 11. Average file transfer delay vs. offered traffic load (TCP)

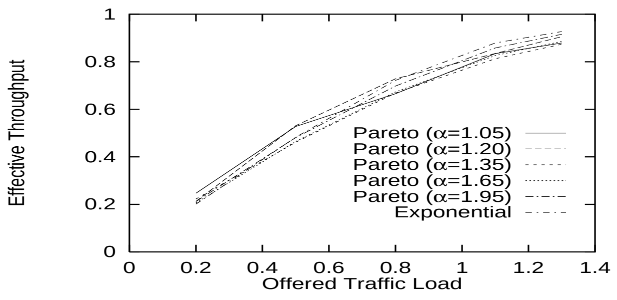

Fig. 13. Effective throughput vs. offered traffic load (TCP).

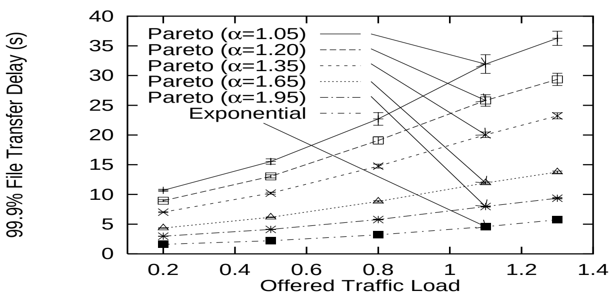

Fig. 12. 99.9% file transfer delay vs. offered traffic load (TCP)

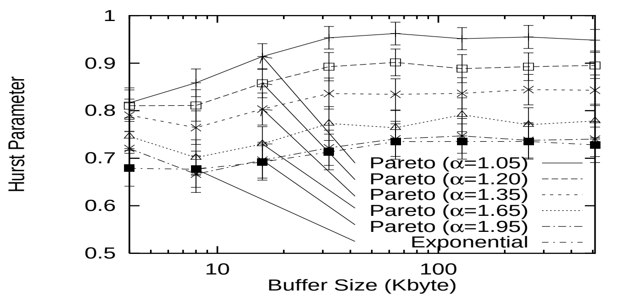

Fig. 14. Hurst parameter vs. buffer size (TCP).

the offered traffic load is low so that retransmission delay caused by the TCP mechanism is negligible. On the other hand, when the offered traffic load is not low, the smaller α is, the larger the average packet transfer delay becomes. This is resulted from retransmission delays caused by the TCP mechanism due to the high packet loss probability.

We next focus on the average file transfer delay. It can be found from the figure that the average file transfer delay is also independent of α in the case of low traffic load. This can be explained by the same reason as in the case of Fig. 6; when α becomes small, reduction of the file transfer delay caused by the slow-start mechanism is canceled by increase caused by the retransmission mechanism. However, it should be noted that the self-similarity of the incoming traffic gives a serious effect on the 99.9% file transfer delay. For example, when α is rather small and the offered traffic load is 1.3, the 99.9% file transfer delay increases from 9.3 s to 36.3 s. It is because a small value of α means generation of extremely large file with a certain probability. When many packets arrive at the switch continuously, the switch buffer suddenly grows and overflows, which leads to increased packet losses. Moreover, the retransmission of the TCP mechanism further increases the 99.9% file transfer delay.

For users' QoS, effective throughput is also important. We show relation between the offered traffic load and the effective throughput in Fig. 13. The effective throughput increases almost linearly as the offered traffic load increases. In other words, the effective throughput is not affected by the intensity of the self-similarity of the network traffic when the TCP mechanism is employed as the upper-layer protocol. From the above discussion, we conclude that the self-similarity of the network traffic has little impact on the first-order network performances such as the average packet/file transfer delays and the effective throughput. However, it cannot be negligible when it comes to second-order performance measures such as a variance of file transfer delays.

C. Effects of System Parameters

In the previous section, we have investigated effects of the characteristics of the incoming traffic (e.g., the file size distribution and the offered traffic load) on the self-similarity of the traffic observed on the bottleneck link. Simulation results have shown that self-similarity of the observed traffic has strong relation with the packet loss probability at the bottleneck switch. Since system parameters such as the buffer size of the bottleneck switch and the link bandwidth have direct influence on packet loss probability, it is expected that those parameters also affect self-similarity of the observed traffic. In what follows, we investigate relation between those system parameters and self-similarity of the network traffic.

1) Effects of Buffer Size: In Figure 14, we first show the Hurst parameters for different buffer sizes ranging from 4 Kbyte to 512 Kbyte. It can be found from Fig. 14 that the Hurst parameter converges to about 0.7 regardless of a value of α when the buffer size is quite small (e.g., 4 Kbyte). It can be explained as follows. When the buffer size is small, the number of packets that can be queued at the switch buffer is only four since we set the packet size to 1 Kbyte. Consequently, the packet loss probability becomes large (i.e., about 10%), and therefore characteristic of the incoming traffic is almost lost. In addition, the behavior of the queue length is not so changed, and then the Hurst parameters become identical even with differ-

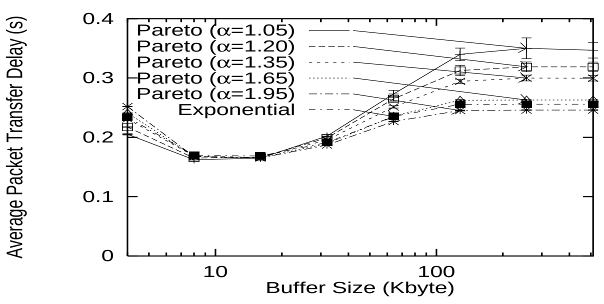

Fig. 15. Average packet transfer delay vs. buffer size (TCP).

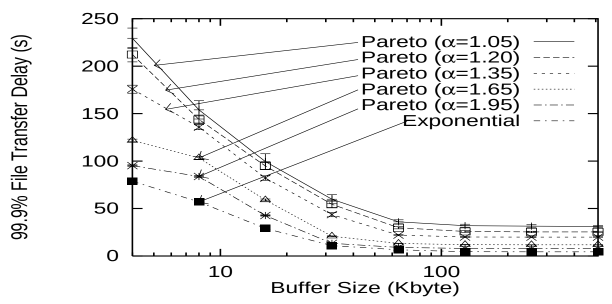

Fig. 17. 99.9% file transfer delay vs. buffer size (TCP).

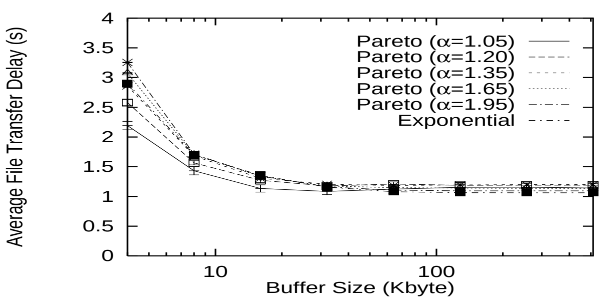

Fig. 16. Average file transfer delay vs. buffer size (TCP).

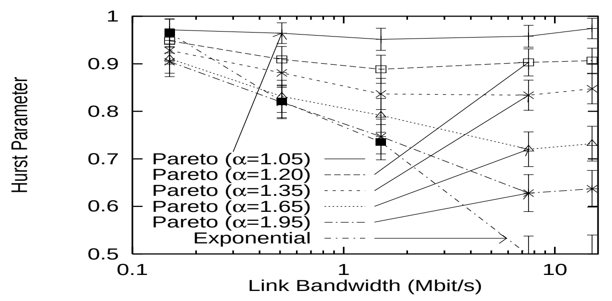

Fig. 18. Hurst parameter vs. link bandwidth (TCP).

ent values of α's. Figure 14 also indicates that the Hurst parameter increases as the buffer size increases. However, when the buffer size is over 100 Kbyte, the Hurst parameter is not affected by the buffer size. It is because when the buffer size is large enough, the window flow control of the TCP mechanism prevents buffer overflow at the switch. Therefore, the self-similarity of the incoming traffic is mostly kept unchanged.

We next show the average packet transfer delay in Fig. 15 for different buffer sizes. This figure shows an important result; when the buffer size is smaller than 32 Kbyte, the average packet transfer delay decreases as the buffer size increases. However, when the buffer size becomes larger than 32 Kbyte, the average packet transfer delay increases again. In this region, the average packet transfer delay increases as α decreases. This phenomenon is explained as follows. Because of the window-flow control of the TCP mechanism, packet losses at the switch are avoided when the buffer size is more than 32 Kbyte. Therefore, if the buffer size is greater than this, the increased queuing delay caused by the large switch buffer results in increase of the average packet transfer delay. This tendency becomes more apparent when α is small since a small value of α causes bursty arrival of packets so that the average queue length becomes large.

On the contrary, if we focus on the average file transfer delay, performance degradation caused by the extremely large buffer size disappears as shown in Fig. 16. We also show the 99.9% file transfer delay in Fig. 17, which shows that the average file transfer delay is not affected by α when the buffer size is larger than 100 Kbyte. This phenomenon can also be explained by the same reason as discussed in Section IV-B: cancellation of the difference caused by the retransmission mechanism and the slow-start mechanism. However, the 99.9% file transfer delay is heavily dependent on α as can be seen in Fig. 17. In particular, the 99.9% file transfer delay is still affected by α even with sufficiently large (e.g., 100 Kbyte) switch buffer.

2) Effects of Bottleneck Link Bandwidth: Finally, we show effect of the bottleneck link bandwidth; the Hurst parameter and the 99.9% file transfer delay for different values of the bottleneck link speed (i.e., from 150 Kbit/s to 15 Mbit/s) are shown in Fig. 18 through 20. We adjusted the average request interval to make the offered traffic load constant (1.1 in this case). One can find from Fig. 18 that the Hurst parameter converges to about 0.95 irrespective of α when the bottleneck link bandwidth is quite small. This is because the bottleneck link is fully utilized almost all the time regardless of α, leading strong self-similarity of the traffic observed on the bottleneck link.

However, the larger the bottleneck link bandwidth becomes, the larger effect of α is observed in the Hurst parameter. It is because the increase of the bottleneck link bandwidth leads to decrease of packet losses at the bottleneck switch and henceforth characteristics of the incoming traffic directly affect the characteristics of the traffic observed on the bottleneck link.

Figures 19 and 20 show the average and the 99.9% file transfer delays, respectively. From these figures, one can find that the effect of α on the file transfer delay diminishes as the bottleneck link bandwidth is large. This can be explained as follows. When bottleneck bandwidth is small (e.g., 150 Kbit/s), the difference in file transfer delays for different values of α is caused by the queuing delays at the switch buffer. For instance, the average queue

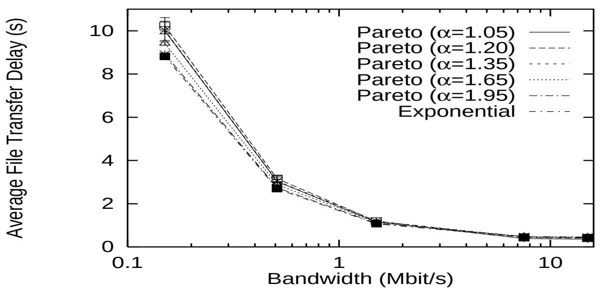

Fig. 19. Average file transfer delay vs. link bandwidth (TCP).

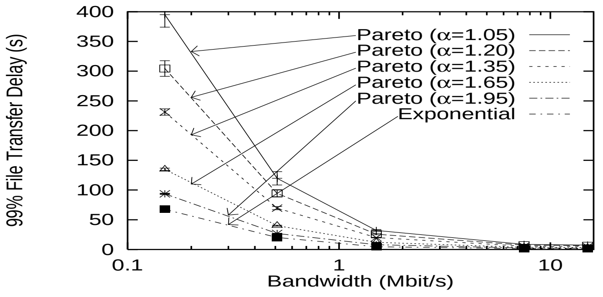

Fig. 20. 99.9% file transfer delay vs. link bandwidth (TCP).

length is 34 packets for $\alpha = 1.05$ while 27 packets for $\alpha = 1.95$. However, as the bottleneck link bandwidth gets larger, the difference in the average file transfer delays becomes smaller since the queuing delay experienced at the switch buffer becomes relatively small. However, Fig. 20 also shows that one can dramatically improve the 99.9% file transfer delay by increasing the bottleneck bandwidth if α is small (i.e., the incoming traffic has strong self-similarity). For example, when the file size distribution follows the exponential distribution, the 99.9% file transfer delay is decreased from 67 s to 1.12 s by increasing the bottleneck link bandwidth from 150 Kbit/s to 15 Mbit/s. And if the incoming traffic has strong self-similarity (e.g. $\alpha = 1.05$), it is decreased from 395 s to 7.47 s. Thus, we conclude that the increase of the bottleneck link bandwidth is very effective way to improve second-order network performances if the network traffic holds strong self-similarity.

V. Conclusion and Future Works

In this paper, we have investigated effects of the upper-layer protocol on self-similarity of the network traffic and the effect of self-similarity of the network traffic on the users' QoS. We have found that when the TCP mechanism is used as the upper-layer protocol, the characteristic of the traffic observed on the bottleneck link is directly affected by the characteristic of the incoming traffic since the TCP mechanism greatly decreases packet losses in the network. However, as the packet loss probability increases, the intensity of the self-similarity of the network traffic becomes very different from that of the incoming traffic.

We have also found that the average file transfer delay and the effective throuput are not strongly effected by the self-similarity of the network traffic. However, the 99.9% file transfer delay increases as the self-similarity of the network traffic becomes strong. From these observations, we have concluded that the self-similarity of the network traffic has little impact on the first-order network performances. However, it cannot be negligible when it comes to second-order performance measures such as the variance of file transfer delays.

As a future work, we should take account of effect of lower-layer protocols such as a CSMA/CD protocol on self-similarity of the network traffic, to reveal the cause of self-similarity found in the real Ethernet traffic.

References

[1] W. E. Leland and D. V. Wilson, "High Time-resolution Measurement and Analysis of LAN Traffic: Implications for LAN Interconnection," in *Proceedings of IEEE INFOCOM '91*, (Bal Harbour, FL), pp. 1360–1366, April 1991.

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

[3] M. W. Garrett and W. Willinger, "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic," in *Proceedings of SIGCOMM 94*, vol. 24, pp. 269–280, October 1994.

[4] W. Willinger, M. S. Taqqu, R. Sherman, and D. Wilson, "Self-similarity through High-variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," in *Proceedings of ACM SIGCOMM*, pp. 100–113, August 1995.

[5] G. Irlam, "UNIX File Size Survey – 1993," September 1994. available at http://www.base.com/gordoni/ufs93.html.

[6] K. Park, G. Kim, and M. Crovella, "On the Relationship Between File Sizes, Transport Protocols, and Self-similar Network Traffic," in *Proceedings of International Conference on Network Protocols*, pp. 171–180, October 1996.

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

[8] J. Beran, *Statistics for Long-Memory Processes. Monographs on Statistics and Applied Probability.* New York: Champman and Hall, 1994.

[9] B. K. Ryu, *Fractal Network Traffic: From Understanding to Implications.* PhD thesis, Columbia Univ., 1996.

[10] D. R. Cox, "Long-Range Dependence: A Review," in *Statistics: An Appraisal* (H. A. David and H. T. David, eds.), pp. 55 – 74, Ames (IA): The Iowa State University Press, 1984.

[11] M. S. Taqqu, V. Teverovsky, and W. Willinger, "Estimators for Long-range Dependence: An Empirical Study," *Fractals*, vol. 3, no. 4, pp. 785–798, 1995.

[12] S. Molnár, A. Vidács, and A. A. Nilsson, "Bottlenecks on the Way Towards Fractal Characterization of Network Traffic: Estimation and Interpretation of the Hurst Parameter," in *Proceedings of PMCCN*, pp. 125–144, November 1997.

[13] M. S. Borella, S. Uludag, G. B. Brewster, and I. Sidhu, "Self-Similarity of Internet Packet Delay," in *Proceedings of IEEE ICC '97*, pp. 513–517, January 1997.

[14] "LBNL network simulator (ns)." available at http://www-nrg.ee.lbl.gov/ns/.

[15] N. L. Jhonson and S. Kotz, *Continuous Univariate Distributions-1.* New York: Wiley-interscience, 1970.

[16] M. E. Crovella and A. Bestavros, "Explaining World Wide Web Traffic Self-similarity," Tech. Rep. TR-95-015, Boston Univ., October 1995.