Electronic Commerce Research, 2: 233–253 (2002)

Performance Analysis for E-Business: Impact of Long Range Dependence

NATARAJAN GAUTAM ngautam@psu.edu
Department of Industrial Engineering, The Pennsylvania State University, 310 Leonhard Building, University Park, PA 16802

SRIDHAR SESHADRI sseshadr@stern.nyu.edu
Operations Management Department, Leonard N. Stern School of Business, New York University, NY 10012

Abstract

We consider an e-business web-server system where the network traffic exhibits self-similarity. We demonstrate that traditional techniques are unsuitable for predicting the network performance under such traffic conditions. Instead, we propose and demonstrate a novel decomposition approximation technique that helps predict delays more accurately and thus is better suited for capacity planning and network design when compared to traditional queueing network analyzers. We also consider several strategies for mitigating the effect of self-similarity, and conclude that admission control holds the greatest potential for improving service. We provide an approximation technique for computing the admission control parameter values. Numerical results and suggestions for future work are discussed.

Keywords: system performance, e-business, self-similar traffic, queueing approximations

1. Self-similarity in telecommunication network traffic

1.1. Introduction

Both data sources as well as real-time applications generate traffic patterns that exhibit certain degrees of correlation between arrivals, and show long-range dependence (LRD) in time as explained in [Sahinoglu and Tekinay, 29]. One of the first researchers to have spotted self-similar behavior (or LRD) in networks are [Leland et al., 17] in ethernet LAN traffic. Ever since, several experiments and measurements have shown that other traffic types such as WWW traffic [Crovella and Bestavros, 9], Wide Area Network traffic [Lucas et al., 18], TCP and UDP traffic [Kushida, 16], all exhibit LRD as well. Recently Menasce et al. [21] reported that the arrival process of requests to e-business servers exhibit LRD, thereby creating a need to study web server performance under self-similar request arrival.

Two of the greatest concerns due to LRD is how to model the traffic and how to obtain system performance measures. Researchers have illustrated the failure of Poisson models (which arguably is the most widely used model for traffic) and the need for more appropriate models. In particular, Willinger and Paxson [37] clearly illustrate the differences between the Internet traffic model against a Poisson model, also see [Stallings, 32]. The main

feature of traffic that exhibits LRD is that correlation dies down very slowly. The question for designers of queueing systems is what impact if any does such correlation have on network performance? How to mathematically understand the impact of LRD on queueing performance? Does the effect propagate through the network or is it only to be found at the gateways to the network? How if required can the effect of LRD be reduced? etc.

For the remainder of this Section 1 we briefly describe the literature on various aspects of self-similar models and analysis. In Section 2 we detail some e-business system analysis tools and techniques that are used for performance analysis. In Section 3 we illustrate the e-business network scenario that we consider in this paper. We illustrate the shortcomings of the traditional queueing analysis techniques and demonstrate our new algorithm in Section 4. Then we formulate and test various performance enhancement strategies in Section 5. Finally we present our concluding remarks and ideas for future work in Section 6.

1.2. Impact of self-similarity on network performance

Sahinoglu and Tekinay [29] cite several studies that indicate the importance of self-similarity to network performance (such as [Park et al., 26]). They also mention that some researchers such as [Heyman and Lakshman, 15] have reservations about the the need for capturing self-similarity (also see [Grossglauser and Bolot, 14] with regard to the debate on modeling self-similar traffic). Sahinoglu and Tekinay [29] add that one of the effects of self-similarity is that the buffers needed at switches and multiplexers must be bigger than those predicted by traditional queueing analysis and simulations. They stress that these large buffers create greater delay in individual streams than were originally anticipated. They conclude that self-similarity introduces new complexities into optimization of network performance and makes the task of providing QoS together with achieving high-utilization difficult. The reader is also referred to [Willinger et al., 38] for up-to-date developments and advances in using self-similar traffic for performance modeling of high-speed telecommunication networks. In Section 4.1, we demonstrate that not only are the effects of LRD considerable but that the large delays can render traditional network analyzers useless. Thus, setting the stage for a different approach to predicting delays.

1.3. Queueing analysis under LRD

Several researchers have attempted to analytically predict the performance under self-similar traffic in a queueing network setting. Shi and Zhu [31] explain that the superposition of self-similar (i.e., LRD) traffic results in self-similar traffic (i.e., multiplexing does not smooth traffic, also observed by Leland et al. [17]); on the same token, Bernoulli splitting of self-similar traffic results in self-similar traffic streams; also, output from a constant service time queue with self-similar input is self-similar. All these results hold under specific set of conditions described in the referenced paper. Christensen and Ballingam [7] demonstrate the reduction of self-similarity by application-level traffic shaping, but we show that from a business viewpoint such traffic shaping of *incoming traffic* does not mitigate the impact of LRD within the network.

Chan and Li [33] argue that self-similar input traffic (say fractional Brownian motion) to a single queue can be analyzed to obtain standard queue performance metrics. If some conjectures hold and if each stage is modeled as a $G/D/1$ queue, the results can be extended to network of queues where Jackson-network-type properties can be derived. The main concern apart from the validity of their conjectures is the mathematical intractability of their technique which would render the method impractical for design and capacity planning.

In contrast to the above studies, we take an engineering view to the problem and study how the effect of LRD can be reduced. We use the insights obtained from simulation experiments to propose a decomposition technique that we believe addresses the long range dependence phenomenon directly.

1.4. Modeling self-similar traffic

The most common modeling methodology for self-similar behavior is the use of heavy-tailed distributions such as the Pareto distribution (see [Addie et al., 1]) and the deployment of the Hurst parameter. Following the mathematics in [Willinger and Paxson, 37], some notations and descriptions are presented below.

Experimental traces of traffic processes exhibit high spatial variability and long-range dependence (autocorrelations with a power law decay). Heavy-tailed distributions (such as Pareto distributions) with infinite variance are used to model the extreme spatial variability. Typical probability distributions $[F(\cdot)]$ are of the form

$$1 - F(x) = \kappa_1 x^{-\beta},$$

where κ_1 is a positive (finite) constant independent of x and the tail index β is such that $0 < \beta < 2$. A fractional Gaussian noise is used to model the fractal or long-range dependent or self-similar behavior. A covariance-stationary Gaussian process $X = (X_k\colon k \geq 1)$ is called a fractional Gaussian noise with Hurst parameter $H \in [0.5, 1)$ if the autocorrelation between X_n and X_{n+k}, $k \geq 0$, is given by

$$cor(X_n, X_{n+k}) = 0.5\{(k+1)^{2H} - 2(k)^{2H} + (k-1)^{2H}\}.$$

The Hurst parameter H quantifies the strength of the fractal scaling.

A discrete-time, covariance-stationary, zero-mean stochastic process $X = (X_k\colon k \geq 1)$ is called exactly self-similar or fractal with scaling parameter $H \in [0.5, 1)$ if for all levels of aggregation (or resolution), $m \geq 1$,

$$X^{(m)} = m^{H-1} X,$$

where the aggregated processes $X^{(m)}$ are defined by

$$X^{(m)}(k) = \frac{X_{(m-1)k+1} + \cdots + X_{km}}{m}, \quad k \geq 1.$$

For an exactly self-similar process with scaling parameter H,

$$Var\, X^{(m)} = \kappa_1 m^{2H-2}.$$

2. Performance analysis methods

In the light of the discussions in the previous sections and the experimental results presented in Section 4.1, it appears that conventional methods for analyzing network performance need a drastic change in orientation to be successful. On the other hand, it also seems to be rather a waste to ignore the vast accumulation of knowledge about queueing system performance. Amongst methods that are used to evaluate and/or compute performance measures of e-commerce systems three techniques dominate, see [Menasce and Almeida, 19, 20]. These are: Using Standard Industry Benchmarks, Queueing Network Analysis, and Simulation.

The first method uses workload generators to test the performance of the system. Examples include, Mindcraft's Webstone, SPEC's SPECWeb96 and SPECWeb99, and SURGE (see [Barford and Crovella, 2]). The Transaction Processing Council released benchmark for e-commerce sites undertaking B2C activity, see [35]. This method can be used to compare the performance of different systems. Menasce and Almeida state that, "Analytic models, even approximate ones, are, in general, the technique of choice for scalability analysis and for capacity planning." [20, p. 321]. They propose the use of queueing network analysis with two important modifications: (i) the network is modeled as multi-class network to account for the vastly different types and nature of workloads that have to be accommodated, and (ii) the degradation in service due to bursty arrival patterns be modeled using an operational approach, see [Buzen, 5]. In case of software contention they suggest the method of Layered Queueing Networks (LQN), see, for example, [Franks et al., 11; Rolia and Sevcik, 28; Neilson et al., 24; Woodside et al., 39]. Commercial and research software is available to carry out detailed simulation of systems, for example, COMNET [8] and ns [25]. We restrict our attention to the second class of models described above. We also focus on single-class queueing networks with First Come First Served (FCFS) discipline.

3. E-business network scenario

A typical web-server for an e-business application can be modeled as a multi-stage multi-class queueing network. The ensuing model is not only hard to tackle analytically but also difficult to simulate. The aim of the analysis is to obtain understanding of the impact of self-similar traffic on a queueing network performance rather than a single stage queue. This in itself is different from previous research that has concentrated upon single stage systems. In particular, we address three questions: (i) Does the location of the bottleneck station within the network affect the performance of the network and/or the quality of approximations? Naively speaking we expect self-similarity to get filtered out by being passed through successive stages of lightly to moderately loaded stations. This phenomenon is known to occur when the inter-arrival times are extremely volatile, see, for example, [Buzacott and Shanthikumar, 3]. In that case passing the arrivals through a stage that has deterministic service time reduces the volatility of the inter-arrival times. (ii) It is well known that "the superposition of many independent streams is approximately

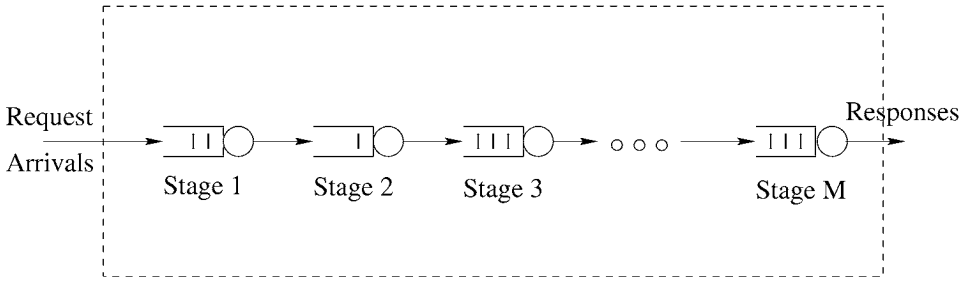

Figure 1. Multi-stage queueing network model of web-server.

Poisson" (see [Daley and Vere-Jones, 10, Section 9.2] and [Gnedenko and Kovalenko, 13, Section 2.8]). Thus, we might expect that if we replace a single server with several servers at a stage without changing that stage's capacity, the downstream stages will be benefited. That is we expect the departures from the multiple servers to act like independent streams whose composition then is Poisson. We test out this hypothesis. (iii) Finally, it is possible to selectively turn away customers when there is high congestion. This might also break up the long-range dependence if the rejection coincides with bursts in traffic. Thus, we ask whether selectively turning away customers reduces the long-range dependence?

To answer these questions we analyze a reasonably complex system. We assume that requests arrive externally and they get processed in M stages, at the end of which they exit the system, i.e., the server sends a response to the request. In this section we assume that there is a single server at each stage that is dedicated to process all the requests in a first-come-first-served fashion. This is relaxed later. See Figure 1 for an illustration of the scenario.

We restrict the study to single class tandem queueing networks. This may not be the most acceptable for a web-server. However, unless scheduling priorities are introduced, when the scheduling rule is FCFS no significant loss of detail ensues by lumping all the classes into one for analysis. In future we will address multi-class networks with priorities. It should be mentioned at this point that texts on web server planning advocate the use of product form approximation for multiple class networks. Unfortunately, the suggested technique is useful only when either special scheduling rules are used or when the mean service times are the same for all classes. The reader should note that the tandem system studied by us is adequate for changing the position of the bottleneck stage, changing from single to multiple servers and also for studying the effect of admission control on self-similarity of traffic within the network. Moreover, the queueing approximations should work well for tandem systems in the first place. If they do so, then we can apply them with greater confidence to more complex ones. Finally, if some devices are more successful at reducing long-range dependence then these should also prove to be useful in more complex settings.

The request arrival process, is modeled as an autocorrelated process, specifically a fractional brownian motion with a given Hurst parameter. This is in line with the observations made by Menasce et al. [21] where request arrival processes for e-business sites are self-similar in nature. The mean request arrival rate is denoted by λ. The successive inter-arrival

times are not only correlated, but also their coefficient of variation (i.e., the ratio of standard deviation to mean) is high (0.64 in all experiments). Let $\mu_1, \mu_2, \ldots, \mu_M$ be the service rates at stages $1, 2, \ldots, M$, respectively. The processing time distribution is exponential at all stages. This is reasonable based on the findings in [Menasce et al., 21] where the session duration (for e-business sites whose requests: (i) are not generated by robots, and (ii) have time outs) is observed to be approximately exponential.

There are two key research issues addressed in the following sections which would be very valuable to e-business network administrators and managers:

1. *Can queueing analysis and approximations be used to predict the queue lengths?* Note that Sahinoglu and Tekinay [29] report that one of the effects of self-similarity in the arrivals is that the queues observed are much bigger than those predicted by traditional queueing analysis. We illustrate this phenomenon and then present an innovative technique that combines traditional queueing analysis with newly developed approximations to predict the queue sizes accurately (see Section 4).
2. *Can the system be modified suitably in order to reduce the queue sizes?* From a customer-satisfaction standpoint, it is very important for the e-business that the queues be kept small. In Section 5, we consider suitable enhancements to the current design/architecture and demonstrate their benefits.

We performed several experiments to answer these questions. All experiments assume that the number of stages $M = 8$. The analytical approximations based on queueing networks are carried out using the software packages in [Moses et al., 22, 23].

4. System performance analysis and experimental results

We demonstrate via experiments the failure of traditional queueing analyses in the context of self-similar traffic (see Section 4.1). In Section 4.2 we develop a novel approximation technique called the burst correction algorithm to predict the system behavior. Finally in Section 4.3 we show how well the burst correction algorithm performs using experimental results.

4.1. Traditional queueing analysis

In this section, we show using experimental results how traditional queueing analysis is unsuitable when the traffic is self-similar. A summary of the input parameters for all experiments and analytical models in this section are described in Table 1. The values are chosen in these experiments because we are doing a set of experiments that changes the place of the bottleneck. For example, the bottleneck station is the fourth, first and second, first (with a near bottleneck at station four), first and third, and first and third but separated by a fast station respectively in the five experiments. For all the simulations in this paper the 99% confidence intervals were in the region of 5–7% of the average queue length values observed.

Table 1. Input values for the experiments.

Index	Hurst parameter	Arrival rate (λ)	Service rate $\mu = (\mu_1 \ldots \mu_8)$
1	0.7	0.976	(1.3013 1.3013 1.3013 1.02 1.05 1.09 1.03 1.4)
2	0.7	0.976	(1.0167 1.00167 1.3013 1.02 1.05 1.09 1.03 1.4)
3	0.7	0.976	(1.0167 1.3013 1.3013 1.02 1.05 1.09 1.03 1.4)
4	0.7	0.976	(1.0167 1.3013 1.00167 1.02 1.05 1.09 1.03 1.4)
5	0.7	0.976	(1.0167 1.0813 1.00167 1.02 1.05 1.09 1.03 1.4)

Table 2. Experimentally observed queue lengths.

Index	Observed # in queue 1	Observed # in queue 2	Observed # in queue 3	Observed # in queue 4	Observed # in queue 5	Observed # in queue 6	Observed # in queue 7	Observed # in queue 8
1	3.2916	3.0468	2.8272	106.801	19.8556	9.2647	31.2899	1.6426
2	95.8414	203.548	2.32794	30.2297	14.4896	8.3715	23.2449	1.6377
3	143.779	2.3538	2.3441	38.4842	16.0804	8.9138	28.7454	1.6256
4	123.305	2.3751	225.3633	29.5028	14.7981	8.5044	25.4596	1.6138
5	114.4848	10.5677	186.352	31.04	14.5781	8.597	23.5157	1.6057

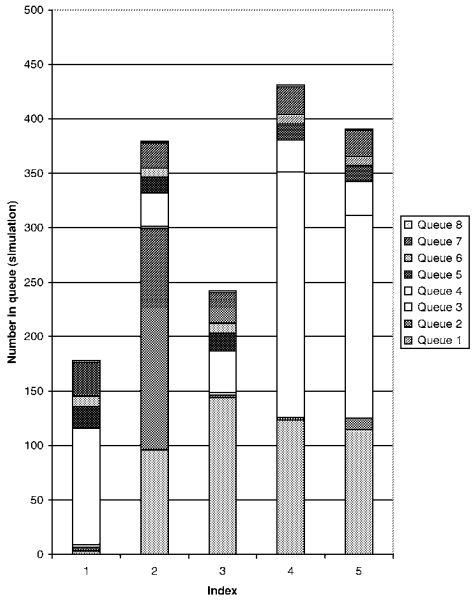

Figure 2. Experimentally observed queue lengths at the nodes.

We use [Paxson, 27], in particular we used FFT-FGN, the S version for Hurst parameter values of 0.7 for simulating 1 million self-similar arrivals into the system for each experiment. In Table 2 and Figure 2 the average number of requests pending at each of the stations are illustrated. Notice the large sizes of the queues.

Table 3. Experimental results and traditional queueing estimates.

Index	Total # waiting in queue (observed)	Total # waiting in queue (traditional)
1	178.02	66.63
2	379.69	122.2
3	242.33	87.4
4	430.92	122.2
5	390.74	128.3

Table 4. Experimentally observed number in system for different Hurst parameters.

H	# waiting in queue
0.5	107.762
0.7	390.74
0.9	6656.225

We now use traditional queueing approximation techniques and obtain the total number of requests waiting in the queues of the web servers. The results are described in Table 3.

From the results shown in Table 3 it is abundantly clear that traditional approximation methods to analyze performance of queueing networks such as Whitt [36] and Buzacott and Shanthikumar [3] severely underestimate the queue sizes. A brief description of the traditional approximation method is summarized in the Appendix. The only available upper bounds for tandem and other queueing systems (as against bounds for single stage systems) that are developed in [Seshadri and Pinedo, 30] are also inadequate. However, textbooks on web server planning continue to use similar or even Markovian models to analyze the performance. The key question is what aspect of the variability induced by long-range dependence affects the queueing phenomenon the greatest? The surprising answer is that it is not the request-to-request inter-arrival time variability but variability on a larger time scale—namely the traffic burstiness. We shall show this in various stages in this paper. We also observe that the position of the bottleneck does not affect the performance significantly.

Next we also tried different Hurst parameter values besides $H = 0.7$, namely $H = 0.5$ and $H = 0.9$ and performed experiments. The case $H = 0.5$ corresponds to renewal arrivals with no dependence between inter-arrival times. For the input parameters defined in index value 5 of Table 1, the total number in the system is experimentally obtained and illustrated in Table 4. Notice that the case $H = 0.5$ which corresponds to the i.i.d renewal case has a queue length of 107.762 which is reasonably close to the value of 128.3 (see Table 3) predicted using the traditional queueing analysis.

4.2. *New approximation technique*

In this paper we only sketch the burst corrected algorithm. The key idea in making this approximation is to decompose the queueing behavior into two parts: one due to the bursti-

ness of traffic and the other due to an arrival process that is of the renewal type. The bursty part of traffic is estimated by feeding the arrival process to a single stage system with deterministic processing time. This approach is motivated by our experiments on traffic shaping described in Section 5.2. The rest of the approximation proceeds by using a queueing network analyzer (see [Whitt, 36]). The correction due to the burstiness of traffic obtained by analyzing the single stage system is then added to the estimates of queues produced by the network analyzer. For now we observe that the queueing phenomenon due to "burstiness" seems to be almost orthogonal to the phenomenon due to "variability of inter-arrival" times. This is a new finding by itself.

Burst Correction Algorithm

1. Determine the bottleneck station in the network.
2. For each node that experiences arrivals from external sources: simulate a single stage deterministic service time queue with arrival pattern and service rate = 0.99 or 0.995 of the bottleneck rate.
3. Record the squared coefficient of variation (SCV) of the departure process as well as any desired queueing characteristics of the single stage systems' buffers'. In this example, let Q_s be the average queue length in the shaper.
4. Use the SCV of the departure process as the SCV of the inputs to the network. Approximate the queue lengths. Call this total Q_N. The combined approximation is $Q = Q_s + Q_N$.

For other performance measures such as probability of delay greater that t, we can decompose and use convolution to estimate the combined delay. This will be done in forthcoming work.

4.3. Results

The total number of customers waiting in the 8 queues (L) are obtained by adding the correction term for traffic burstiness and analytical approximations to obtain the results in this section. For the input values described in Table 1, a comparison is made between the experimental results, the traditional queueing approximation and the newly developed burst correction approximation. These results are tabulated in Table 5 and illustrated in Figure 3.

As proof of viability, the results show that when the calculations are appropriately fine tuned to handle the bursts of arrivals as done in this paper, good approximations can be obtained. The reader will note that the approximations are much more accurate compared to traditional queueing approximations. These results are not very accurate, but for the purpose of capacity planning and analysis they are quite reasonable. Moreover the aim of this research work is to show a better way to approximate queueing behavior and to provide an understanding of what causes excessive congestion.

The results using Hurst parameter values of 0.5, 0.7 and 0.9 are illustrated in Figure 4. In the figure, the y-axis is in the logarithm (to the base 10) scale. Notice how well the burst

Table 5. Experimental results and predicted estimates.

Index	Total # waiting in queue (observed)	Total # waiting in queue (traditional queueing network)	Total # waiting in queue (approximation with correction)	Error (%) (approx. − obs.)/obs.
1	178.02	66.63	203.7	14.44
2	379.69	122.2	458.8	20.83
3	242.33	87.4	251.0	3.58
4	430.92	122.2	458.8	6.47
5	390.74	128.3	464.9	18.99

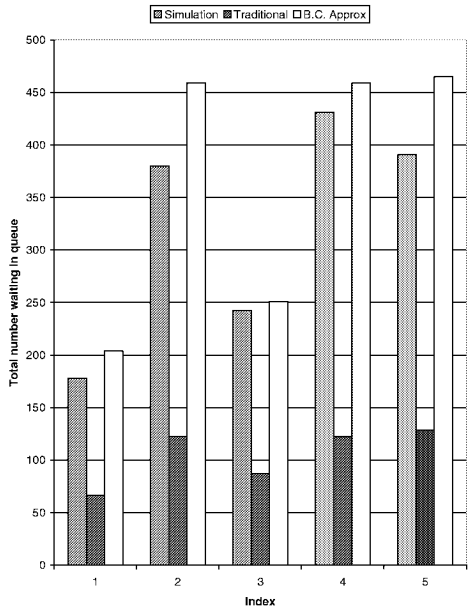

Figure 3. Predicting the total number in the system.

correction technique predicts the size of the queues especially for $H = 0.7$ and 0.9 values. The case $H = 0.5$ being i.i.d. arrivals, the traditional approximations perform very well, in fact better than the burst correction algorithm which is too conservative in this case.

5. Improving the system performance

From a customer satisfaction point of view, it is very important for e-businesses to reduce the large number of customer requests piled up in the system. Therefore we consider three schemes (described in Sections 5.1–5.3) and test them to see how much of a performance enhancement each of the schemes can produce.

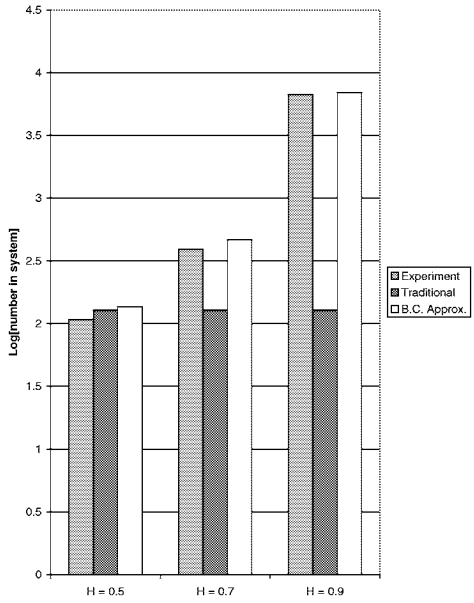

Figure 4. Comparing queue lengths at the nodes for $H = 0.5$, 0.7 and 0.9.

Table 6. Results for the multi-server case.

# of servers $N = (N_1 \ldots N_8)$	Service rate $\mu = (\mu_1 \ldots \mu_8)$	Observed # in queues	Predicted # in queues	Error (%)
(5 5 5 5 5 5 5 5)	(0.2033 0.2163 0.2003 0.204 0.21 0.218 0.206 0.28)	387.01	455.9	17.79
(5 7 3 2 1 3 5 3)	(0.2033 0.1545 0.3339 0.510 1.05 0.3633 0.206 0.4667)	393.61	458.6.9	16.51

5.1. Enhancement: multi-server system

In this section we investigate a methodology to reduce the number of customers in the system. Clearly, the main reason for the long queues is the highly correlated arrivals. Because the arrivals are split into multiple number of parallel servers, it is conjectured that the queue lengths would reduce since for each server, the arrival process is now less correlated.

An experiment similar to that in Section 4 is performed with the only exception being the enhancement of multiple servers in each stage. In order to keep the comparisons meaningful, we assume that the service rate of each server is now reduced by dividing the original service rate by the number of servers in that stage. The rest of the experiment is identical including the 1 million arrivals using the self-similar arrival process detailed in Section 4. For all experiments, the Hurst parameter is 0.7 and the arrival rate is 0.976 arrivals per unit time.

Note that in terms of mean utilizations, the parameters in Table 6 are similar to those in the last row (index 5) of Table 1. The number in the queue has remained practically

Table 7. Results for large number of servers.

Index	# of servers $N = (N_1 \ldots N_8)$	Service rate $\mu = (\mu_1 \ldots \mu_8)$
1	(1 1 1 1 1 1 1 1)	(1.0167 1.0813 1.00167 1.02 1.05 1.09 1.03 1.4)
2	(5 7 3 2 1 3 5 3)	(0.2033 0.1545 0.3339 0.510 1.05 0.3633 0.206 0.4667)
3	(100 7 3 2 1 3 5 3)	(0.010167 0.1545 0.3339 0.510 1.05 0.3633 0.206 0.4667)

Index	Observed # in queue 1	Observed # in queue 2	Observed # in queue 3	Observed # in queue 4	Observed # in queue 5	Observed # in queue 6	Observed # in queue 7	Observed # in queue 8
1	114.4848	10.5677	186.352	31.04	14.5781	8.597	23.5157	1.6057
2	111.7459	8.60714	203.4265	29.7812	13.505	7.3411	18.0585	1.1409
3	104.1756	8.4747	185.9625	36.3992	13.9563	7.5888	17.9111	1.1523

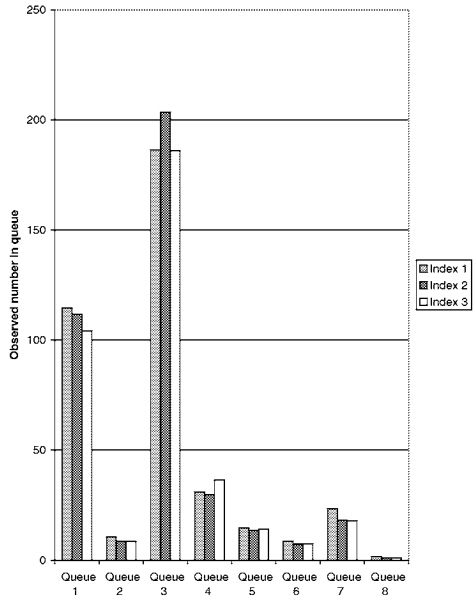

Figure 5. Experimentally observed number in each node with many servers.

unchanged. Also note that the predictions based on the new approximation proposed by us match the experimental findings reasonably accurately.

Even with 100 servers, the total number of requests in the queue can be brought down to only about 375. Moreover, even having 100 servers in the first stage does not affect subsequent queue lengths (see Table 7). The results in Table 7 are depicted in Figure 5. In conclusion, we expected that by adding multiple servers, the self-similar pattern could be broken. However that did not happen, which demonstrates that long-range dependence can not be significantly reduced through splitting, filtering, and then recombining the arrival stream.

5.2. *Enhancement: shaping arrivals*

In this section we investigate another methodology to reduce the number of customers in the system besides increasing the number of parallel servers. It is conjectured that by shaping the traffic before entering the server, the queues would reduce. This is because shaping smoothes the traffic and gets rid of the peaks, thus lowers the variance of the inter-departure events. In other words after the shaper the inter-message time's coefficient of variation (ratio of standard deviation to mean) is significantly reduced. This phenomenon definitely occurs even in the presence of long-range dependence (see Table 8).

The question is whether cutting down the variability reduces the queues downstream of the shaper? An experiment which is an enhancement to that in Section 5.1 is performed with an additional deterministic server before the first set of parallel queues to evaluate the effect of traffic smoothing. The remainder of the experiment is identical including the 1 million arrivals using the self-similar arrival process detailed in Section 4. For all experiments, the Hurst parameter is 0.7 and the arrival rate is 0.976 arrivals per unit time. Also, the number of servers $N = (N_1 \ldots N_8)$ is (5 7 3 2 1 3 5 3), and the service rate $\mu = (\mu_1 \ldots \mu_8)$ is (0.2033 0.1545 0.3339 0.510 1.05 0.3633 0.206 0.4667).

In fact we assume that the shaper takes away the self-similarity and the output traffic from the shaper is deterministic. Thus the shaper queue can be estimated and the rest of the network queues can also be estimated. Notice that because we have introduced an additional processing step, if each estimate is an upper bound then their sum gives an upper bound to the delay. Once again, from Table 9 note that in terms of mean utilizations, the above two results are similar to the last row of results in Table 1. The number in the system has again reduced but not enough. In fact the total backlog including the messages in the shaper buffer actually increases due to the shaper speed reducing. This is also a learning point not found in the literature. Note that the new approximation results match the experimental findings reasonably accurately. Before we discuss the approximation, it is important to understand and appreciate why traffic shaping need not reduce delays.

Table 8. Traffic characteristics before and after the shaper.

	Mean	Variance
Before shaper	1.0356	0.4356
After shaper	1.0356	0.0911

Table 9. Results when system uses a shaper.

Service rate of shaper	# in shaper queue	Observed total number in all stages incl shaper	Estimated total number in all stages incl shaper	Error (%)
1.00167	251.9716	425.3383	455.6	7.12
1.0167	102.8485	334.4206	452.3	35.26
1.167	5.3541	402.0668	447.9	7.12

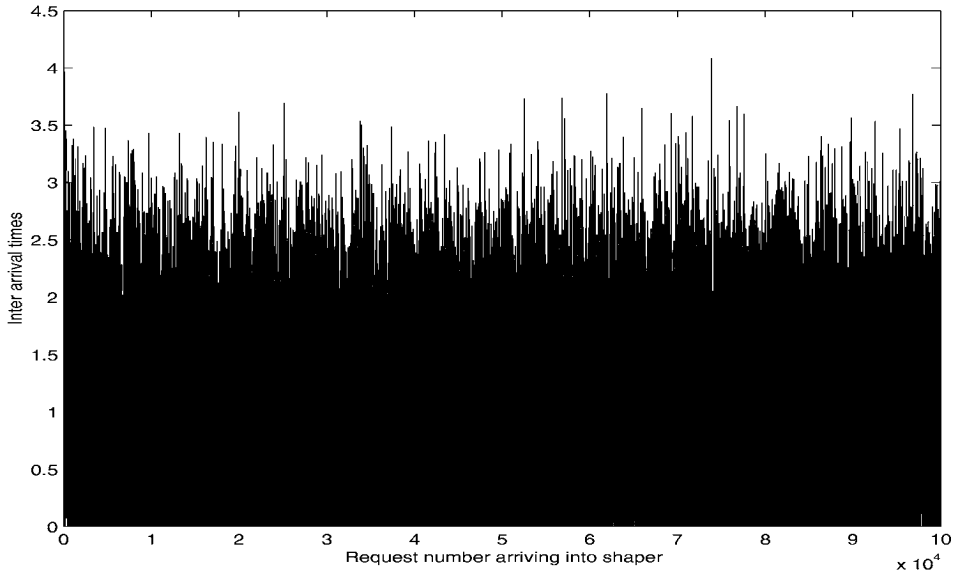

Figure 6. Traffic before shaper.

The relatively small improvement due to traffic shaping can be explained as follows. The output from the shaper would contain very long bursts such that the traffic generated into a downstream buffer may be much larger than the service rate of the downstream buffer. By burst we mean the length of the total number of back-to-back messages which also correspond to the busy period of the server. In fact the bursts have a squared coefficient of variation of zero. In summary, although the time between messages has become less variant (i.e. variance lowers), the string of back-to-back messages (i.e. bursts) have increased. This leads to extremely large queues in the downstream buffer if this traffic arrives faster than what the stage can process. Figure 6 represents the inter arrival times of the traffic before the shaper and Figure 7 represents the inter departure times after the shaper. Note the large bursts (corresponding to the troughs) in Figure 7. The low points in the graphs suggest extremely high arrival rates. Therefore though we reduce the traffic variance, we have introduced on–off behavior with very long on times. If we slow down the shaper, the queues at the shaper itself build up. Thus, the total queues increase whether we pass on the burst or contain it. The phenomenon can be understood also as follows, adding a shaper will only increase delay thus the queue lengths. Thus, what use is it?

There are however several benefits due to a shaper: (1) It motivates the idea of approximating and eliminating the effect of bursts. We noticed that if the shaper's service rate is slightly lower than that of the slowest stage in the rest of the system then the queue lengths in the rest of the system can be predicted quite accurately. Thus, we can decompose the traffic into a bursty and a non-bursty part and evaluate each separately. The queue at the shaper can be predicted accurately using simulation (as it is a single stage and simple system). Thus, we arrive at our key approximation idea. Simulate just the shaper with its

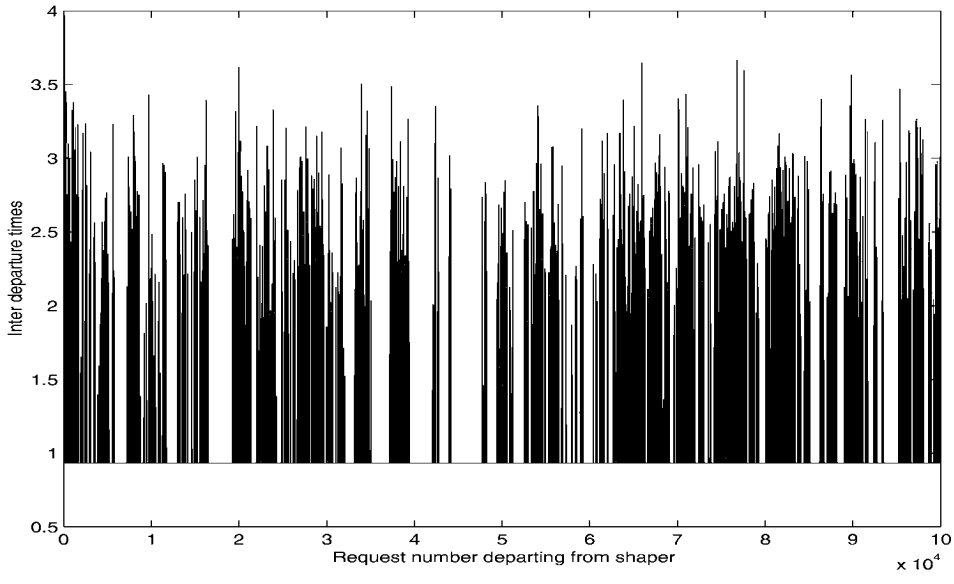

Figure 7. Traffic after shaper.

speed slightly below the bottleneck station's queue. Add this estimate to the traditional approximation. This should be higher than the actual queues observed if the traditional approximation gives an upper bound. See the *Burst Correction Algorithm* in Section 4.2. (2) The shaper can be potentially useful if containing the traffic within the shaper will save valuable buffer space and control cost within the system because most of the bursts stay outside the system. (3) The shaper makes the work in the rest of the system predictable. (4) Also, the shaper leads to lower loss when admission is restricted, see next section.

5.3. Enhancement: Enforcing admission control

None of the enhancements in the above sections were able to reduce the number in the system significantly. One of the main contributions of this research is an innovative technique which can dramatically reduce the number of requests at the price of rejecting some requests when queues build up. Results are described below based on the admission control policy: when the number of requests waiting exceeds a value K, reject arriving requests. The table below represents different thresholds K for the cases of (i) single servers at all stages without shaper, (ii) multiple servers at all stages without shaper, and (iii) multiple servers at all stages with shaper in front of the first stage.

Experiments are performed for the above cases by simulating 1 million arrivals using the self-similar arrival process detailed in Section 4. For all experiments, the Hurst parameter is 0.7 and the arrival rate is 0.976 arrivals per unit time. For the single server cases, the service rate $\mu = (\mu_1 \ldots \mu_8)$ is (1.0167 1.0813 1.00167 1.02 1.05 1.09 1.03 1.4).

Table 10. Results with admission control.

Index	# of servers	Shaper present?	K value	loss rate	Observed # in queue	Estimated # in queue	Error (%)
1	Single	No	150	0.00771	202.5728	159.0	−21.51
	Single	No	85	0.0148	147.1979	121.7	−17.29
	Single	No	15	0.0603	51.02436	52.8	3.45
2	Multiple	No	150	0.00801	204.6737	151.8	−25.82
	Multiple	No	85	0.0150	133.8948	114.7	−14.33
	Multiple	No	15	0.0562	48.5925	49.4	1.7
3	Multiple	Yes	150	0.0101	173.663	154.0	−11.31
	Multiple	Yes	85	0.0153	138.45	119.7	−13.57
	Multiple	Yes	15	0.0455	62.8229	58.7	−6.56

For the multiple server cases the number of servers $N = (N_1 \ldots N_8)$ is (5 7 3 2 1 3 5 3), and the service rate $\mu = (\mu_1 \ldots \mu_8)$ is (0.2033 0.1545 0.3339 0.510 1.05 0.3633 0.206 0.4667). When there is a shaper, its service rate is 1.00167 units.

Once again, note that in terms of mean utilizations, the above two results are similar to the last row (index 5) of parameters in Table 1. The number in the system has significantly reduced. Note that the approximations match the experimental findings very accurately. In summary, it may be worth it in terms of customer satisfaction for an e-commerce site to perform admission control by rejecting few customers in order to provide far superior service to others. Notice that by rejecting only about 5% of the customers, the queue lengths reduce by 90%. By rejecting only 1% the queues reduce by 50%. This is a surprising finding and is quite fascinating. Notice that such dramatic reduction in congestion is not normally expected. For example, the initial bottleneck utilization levels (with zero-rejection) are around 0.967/1.00167 or 96.53%. Thus, $1/(1-\rho) = 28.8$. When 1% is rejected, $1/(1-\rho) = 22.4$. When 5% is rejected it evaluates to 11.8. Therefore, the actual reductions seen are indeed dramatic. They underscore the fact that when overflows occur they occur in a correlated fashion. Thus, remove some of the long-range dependence from the process.

We now briefly address how should a manager go about choosing the right rejection level? We adopt a simulation-cum-optimization approach used in [Glassey et al., 12]. In this approach the queue lengths are determined for a few select values via simulation. The rest interpolated using a quadratic approximation. Notice that we only need to simulate the shaper with the rejection control in place. For example, for the last experiment (index 3 in Table 10) we obtain expected queue length, $Q = 41.53 + 1.48K - 0.004K^2$. Thus, if the manager wanted an average total delay of 100 units of time when the arrival rate is 0.976 per unit time then s/he solves for $(100)(0.976)(1 - 0.025) = 41.53 + 1.48K - 0.004K^2$; obtaining $K = 40$ with 0.025 being the estimated fraction of rejected arrivals. If we simulate using this value of K we obtain a queue length equal to 97.596, a loss fraction of 0.0257 and a delay of 102.63 units of time, thus confirming the usefulness of the approach.

Plots for simulated values versus fitted values using Table 10 are depicted in Figures 8–10. Figures 8, 9 and 10 correspond to the index values of 1, 2 and 3 in Table 10. For all three indices a quadratic fit is obtained to approximate the relationship between K

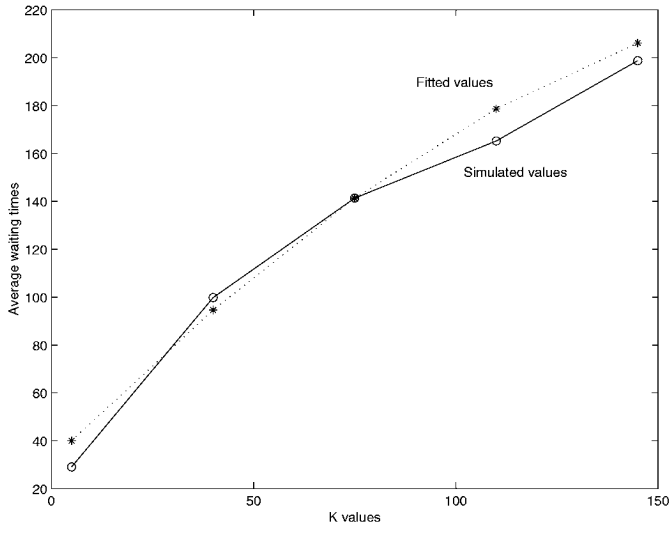

Figure 8. Fitted versus simulated values for index value 1.

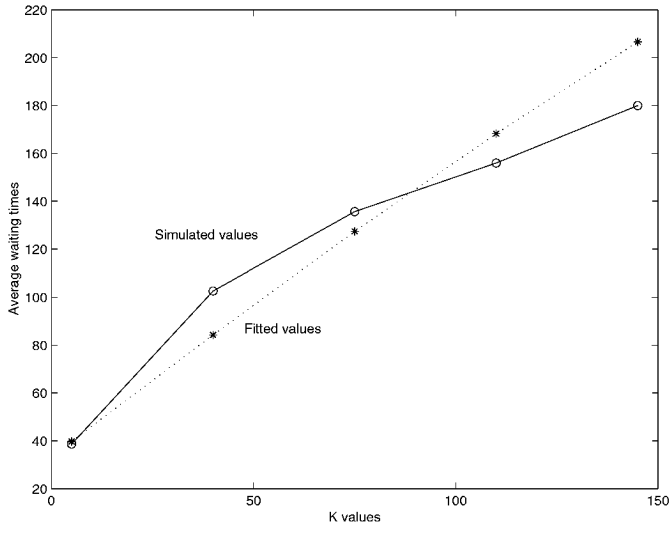

Figure 9. Fitted versus simulated values for index value 2.

and the delay. Notice from Figures 8–10 that the fitted values match the simulated results extremely well. This simplistic engineering approximation technique can be used very effectively in choosing the appropriate K value if the desired performance level is known.

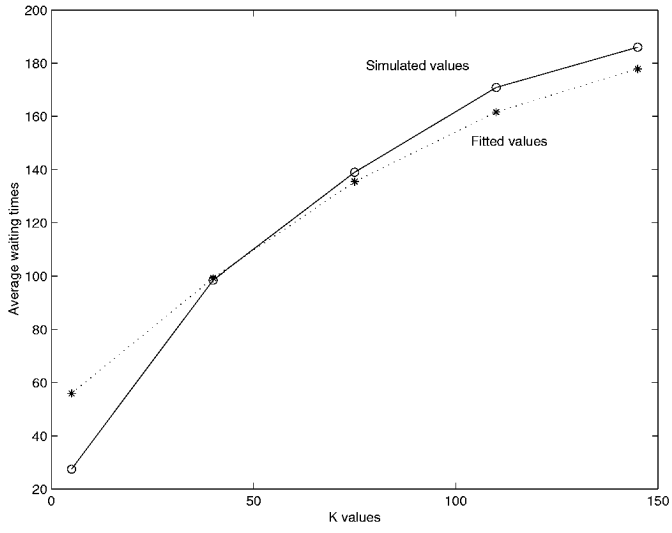

Figure 10. Fitted versus simulated values for index value 3.

6. Conclusions

We have demonstrated the potentially high impact of self-similar traffic on performance degradation in e-commerce systems. Such traffic can not be easily managed by resorting to the use of parallel servers or traffic shaping. However, if a small fraction of the traffic, during bursty periods, is turned away then the performance improvement is quite significant. We have also shown that by decomposing traffic into bursty and non-bursty components it is possible to obtain a reasonable approximation to waiting times in the system. Our proposed method overestimates the queue lengths when there is no admission control and underestimates them when there is admission control. The phenomenon is almost symmetric such that the average error over all experiments shown in this paper is 2.48% (the average absolute error is 14.1%) compared to the average error of 65.2% for traditional methods.

In future work we aim to develop these ideas in two ways: (i) Test them in the context of queueing networks with random routing, multiple classes of arrivals and with synthetic as well as actual traffic arrival patterns. (ii) Develop a theoretical framework for decomposing the traffic and produce bounds for the expected queue lengths using the decomposition approach.

Acknowledgments

The authors thank the associate editor and the anonymous reviewers for their comments and suggestions that led to considerable improvements in the content and presentation of this paper.

Appendix

We are given that the arrival process of requests is of the renewal type with mean inter-arrival time equal to $1/\lambda$ and squared coefficient of variation (scv) of the inter-arrival time equal to C_a^2. The service times at each resource as well as the inter-arrival times to each resource can not be approximated by exponential distributions. The requests arriving to a resource are processed according to the first come first served (FCFS) service protocol. There are m resources where resource i has c_i identical facilities or parallel servers. The service times at resource i are i.i.d., with mean service time equal to $1/\mu_i$ and scv equal to $C_{S_i}^2$. The utilization of each resource is less than one. The other parameters that are specified are: (i) the expected number of visits that a typical request requires at resource i, which is denoted as v_i, (ii) the fraction of requests that arrive to the network that first visit resource i, denoted as γ_i, where $\sum_{i=1}^{m} \gamma_i = 1$, and (iii) the probabilities that a request upon completion of service at resource j requires resource i (denoted as p_{ji}). The p_{ji}'s are called the switching probabilities.

Approximation procedure We first define the traffic equations and then outline the approximation steps for determining the expected number of requests at each resource. The reader is referred to [Buzacott and Shanthikumar, 3] for further details.

Traffic equations Let the external arrival rate of requests to resource of type i be denoted as λ_i. We are given that the expected number of visits to center i to complete the processing of a request is v_i. Therefore,

$$v_i = \frac{\lambda_i}{\lambda}. \tag{1}$$

Equating the expected number of visits into and out of resource i, we get the traffic equations

$$v_i = \gamma_i + \sum_{j=1}^{m} v_j p_{ji}, \quad \text{for } i = 1, \ldots, m. \tag{2}$$

Let the utilization of resource i be denoted as ρ_i. Then

$$\rho_i = \frac{\lambda_i}{c_i \mu_i}, \quad \text{for } i = 1, \ldots, m. \tag{3}$$

Approximation for the expected number of requests in the system Following Buzacott and Shanthikumar [4], the arrival process to resource i is assumed to be of the renewal type with scv equal to $C_{a_i}^2$. Let $C_{d_i}^2$ be the scv of the departure process from resource i. Then $C_{a_i}^2$ is approximated by

$$C_{a_i}^2 = \frac{1}{\lambda_i} \sum_{j=1}^{m} \lambda_j p_{ji} \left[p_{ji} C_{d_j}^2 + (1 - p_{ji}) \right] + \frac{\lambda \gamma_i}{\lambda_i} \left[\gamma_i C_a^2 + (1 - \gamma_i) \right]. \tag{4}$$

The following approximation for $C_{d_i}^2$ is based on approximation number two given on page 75 of [Buzacott and Shanthikumar, 3].

$$C_{d_i}^2 = 1 + \frac{\rho_i^2}{c_i}\left(C_{S_i}^2 - 1\right) + \left(C_{a_i}^2 - 1\right)\frac{(1-\rho_i^2)(2-\rho_i) + \rho_i C_{S_i}^2 (1-\rho_i)^2}{2 - \rho_i + \rho_i C_{S_i}^2}.$$

The squared coefficients of variation of the arrival processes are computed by solving this system of linear equations. Let $E[W(\lambda, \mu)]_{M/M/c}$ be the expected waiting time of a customer in an $M/M/c$ queue with arrival rate λ and service rate μ. Let the expected number of requests at resource i be $E[N_i]$ and in the entire system be $E[N]$. Then we may approximate

$$E[N_i] \approx \frac{E[W(\lambda_i, \mu_i)]_{M/M/c_i}}{E[W(\lambda_i, c_i\mu_i)]_{M/M/1}} \frac{\rho_i(1 + C_{S_i}^2)}{2 - \rho_i + \rho_i C_{S_i}^2} \frac{\rho_i(2-\rho_i)C_{a_i}^2 + \rho_i^2 C_{S_i}^2}{2v_i(1-\rho_i)} + \rho_i \tag{5}$$

and

$$E[N] = \sum_{i=1}^{m} E[N_i]. \tag{6}$$

References

[1] Addie, R.G., M. Zukerman, and T.D. Neame. (1998). "Broadband Traffic Modeling: Simple Solutions to Hard Problems." *IEEE Communications Magazine* 88–95, August.

[2] Barford, P. and M. Crovella. (1998). "Generating Representative Web Workloads." In *Proc. of 1998 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, Madison, WI, June 22–26, pp. 151–160.

[3] Buzacott, J.A. and J.G. Shanthikumar. (1992). *Stochastic Models of Manufacturing Systems*. New York: Prentice-Hall.

[4] Buzacott, J.A. and J.G. Shanthikumar. (1992). "Design of Manufacturing Systems Using Queueing Models." *Queueing Systems* 12, 135–214.

[5] Buzen, J.P. (1978). "Operational Analysis: An Alternative to Stochastic Modeling." In *Performance of Computer Installations*, North-Holland, pp. 175–194.

[6] Chao, X., M. Miyazawa, and M. Pinedo. (1999). *Queueing Networks: Customers, Signals and Product Form Solutions*, New York: John Wiley & Sons.

[7] Christensen, K.J. and V. Ballingam. (1997). "Reduction of Self-Similarity by Application-Level Traffic Shaping." In *Proc. of 22nd Annual Conf. on Local Computer Networks*, pp. 511–518.

[8] COMNET. `http://www.compuware.com/products/ecosystems/comnet.htm`.

[9] Crovella, M.E. and A. Bestavros. (1997). "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes." *IEEE/ACM Transactions on Networking* 5(6), 835–846.

[10] Daley, D.J. and D. Vere-Jones. (1988). *An Introduction to the Theory of Point Processes*. New York: Springer.

[11] Franks, G., A. Hubbard, S. Majumdar, D. Petriu, J. Rolia, and C.M. Woodside. (1996). "A Toolset for Performance Engineering and Software Design of Client-Server Systems." *Performance Evaluation Journal* 24(1–2), 117–135.

[12] Glassey, C.R., S. Seshadri, and J.G. Shanthikumar. (1996). "Linear Control Rules for Production Control of Semiconductor Fabs." *IEEE Transactions on Semiconductor Manufacturing* 9(4), 536–549.

[13] Gnedenko, B.V. and I.V. Kovalenko. (1989). *Introduction to Queueing Theory*. Boston, MA: Birkhäuser.

[14] Grossglauser, M. and J.-C. Bolot. (1999). "On the Relevance of Long-Range Dependence in Network Traffic." *IEEE/ACM Transactions on Networking* 7(5), 629–640.

[15] Heyman, D.P. and T.V. Lakshman. (1996). "What Are the Implications of Long-Range Dependence for VBR-Video Traffic Engineering?" *IEEE/ACM Transactions on Networking* 4(3).

[16] Kushida, T. (1998). "The Traffic Measurement and the Empirical Studies for the Internet." In *GLOBECOM-98*, Vol. 2, pp. 1142–1147.

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

[18] Lucas, M.T., D.E. Wrege, B.J. Dempsey, and A.C. Weaver. (1997). "Statistical Characterization of Wide-Area IP Traffic." In *Proc. of Sixth International Conference on Computer Communications and Networks*, pp. 442–447.

[19] Menasce, D.A. and V.A.F. Almeida. (1998). *Capacity Planning for Web Performance*. Prentice-Hall.

[20] Menasce, D.A. and V.A.F. Almeida. (2000). *Scaling for E-Business*. Prentice-Hall.

[21] Menasce, D.A., F. Ribeiro, V.A.F. Almeida, R. Fonseca, R. Riedi, and W. Meira Jr. (2000). "In Search of Invariants for E-Business Workloads." In *Proceedings of EC'00*, Inst. Math. Appl., October, Minneapolis, MN.

[22] Moses, M. and S. Seshadri. (2000). "Using Modeling Software to Improve Operations." *International Journal of Operations and Quantitative Management* 6(3), 1–25 (available at `http://www.stern.nyu.edu/HOM`).

[23] Moses, M., S. Seshadri, and M. Yakirevich. (1999). *Gaining Competitive Advantage through Business Process Improvement using the HOM Software System.* Irwin-McGraw-Hill (software available for download at `http://www.stern.nyu.edu/HOM`).

[24] Neilson, J.E., C.M. Woodside, D.C. Petriu, and S. Majumdar. (1995). "Software Bottlenecking in Client–Server Systems and Rendezvous Networks." *IEEE Transactions on Software Engineering* 21(9), 776–782.

[25] ns. `http://www-mash.ca.berkeley.edu/ns/`.

[26] Park, K., G. Kim, and M.E. Crovella. (1996). "On the Relationship between File Sizes, Transport Protocols, and Self-Similar Network Traffic." In *Proc. of Intl. Conf. on Network Protocols*, pp. 171–180.

[27] Paxson, V. (1995). "Fast Approximation of Self-Similar Network Traffic." Technical report LBL-36750/UC-405, April.

[28] Rolia, J.A. and K.C. Sevcik. (1995). "The Method of Layers." *IEEE Transactions on Software Engineering* 21(8), 689–700.

[29] Sahinoglu, Z. and S. Tekinay. (1999). "On Multimedia Networks: Self-Similar Traffic and Network Performance." *IEEE Communications Magazine* 37(1), 48–52.

[30] Seshadri, S. and M. Pinedo. (1998). "Bounds for the Delay in Multiclass Open Queueing Networks under Shortfall Based Priority Rules." *Probability in the Information and Engineering Sciences* 12(3), 329–350.

[31] Shi, J. and H. Zhu. (1999). "Merging and Splitting Self-Similar Traffic." In *Proc. of Fifth Asia-Pacific Conference on Communications, APCC/OECC-99*, Vol. 1, pp. 110–114.

[32] Stallings, W. (1998). *High-Speed Networks TCP/IP and ATM Design Principles*. Upper Saddle River, NJ: Prentice-Hall.

[33] Chan, Tat-Keung and V.O.K. Li. (1998). "Decomposition of Network of Queues with Self-Similar Traffic." In *GLOBECOM-98*, Vol. 5, pp. 3001–3006.

[34] Toyoizumi, H., J.G. Shanthikumar, and R.W. Wolff. (1997). "Two Extremal Autocorrelated Arrival Processes." *Probability in the Engineering and Informational Sciences* 11, 441–450.

[35] Transaction Processing Council. TPC-W. `http://www.tpc.org`.

[36] Whitt, W. (1983). "The Queueing Network Analyzer." *The Bell System Technical Journal* 62(9), 2779–2815.

[37] Willinger, W. and V. Paxson. (1998). "Where Mathematics Meets the Internet." *Notices of the American Mathematical Society* 45(8), 961–970.

[38] Willinger, W., M.S. Taqqu, and A. Erramilli. (1996). "A Bibliographical Guide to Self-Similar Traffic and Performance Modeling for Modern Figh-Speed Networks." In F.P. Kelly, S. Zachary, and I. Ziedins (eds.), *Stochastic Networks: Theory and Applications*, Royal Statistical Lecture Note Series, Vol. 4, pp. 339–366, Oxford, UK: Clarendon Press.

[39] Woodside, C.M., J.E. Neilson, D.C. Petriu, and S. Majumdar. (1995). "The Stochastic Rendez-Vous Network Model for Performance of Synchronous Client–Server-Like Distributed Software." *IEEE Transactions on Computers* 44(1).