Approximations for system performance under self-similar traffic

N. Gautam
Department of Industrial Engineering,
The Pennsylvania State University,
University Park, PA 16802
ngautam@psu.edu

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

Abstract

In this paper we survey the literature on approximations for system performance measures when the traffic exhibits self-similarity. We propose and test an approximation method for e-commerce servers that can be modeled as multi-stage queueing systems. The two main contributions are: (1) we develop an appropriate design of the e-commerce server to reduce the number of requests in the system, (2) we obtain queueing network approximations that can predict the system performance accurately. Numerical results and suggestions for future work are discussed.

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 [24]. One of the first researchers to have spotted self-similar behavior (or LRD) in networks are Leland et al [13] in ethernet LAN traffic. Ever since, several experiments and measurements have shown that other traffic types such as WWW traffic (Crovella and Bestavros [6]), Wide Area Network traffic (Lucas et al [14]), TCP and UDP traffic (Kushida [12]), all exhibit LRD as well.

One of the greatest concerns due to LRD is how to model the traffic and 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 [32] clearly illustrate the differences between the Internet traffic model against a Poisson model (see Figure 1). Figure 1 and the following quote describing the figure, have been taken from Willinger and Paxson [32]: "The top row shows a randomly selected subset of each trace on a time scale of 100 msec; that is, each point in the plots reflects the number of packets observed during a 100 msec interval, for a total of 6 sec. The second row shows a time scale factor of 10 larger: now, each point reflects the number of packets per 1sec, spanning 60 sec total. The black regions illustrate from where the plots in the row above were made. A key point is that we not only have increased the scale of the X-axis by a factor of 10, but we have also done the same to the Y-axis. With the third row we have again increased the scale in both X and Y by a factor of ten, and in the final row by another factor of size, such that here the plots span the entire hour of the traces."

1.2 Impact of self-similarity on network performance modeling

Grossglauser and Bolot [10] state that there is still considerable debate about how to model traffic processes that exhibit self-similarity, and about their impact on network and application performance. Sahinoglu and Tekinay [24] illustrate that there are several studies that indicate the importance of self-similarity to network performance (such as Park et al [20]) or the irrelevance of the need for capturing self-similarity (such as Heyman and Lakshman [11]). Sahinoglu and Tekinay [24] 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 also state that these large buffers create greater delay in individual streams than were originally anticipated. Willinger et al [33] describe the latest developments and advances in using self-similar traffic for performance modeling of high-speed telecommunication networks.

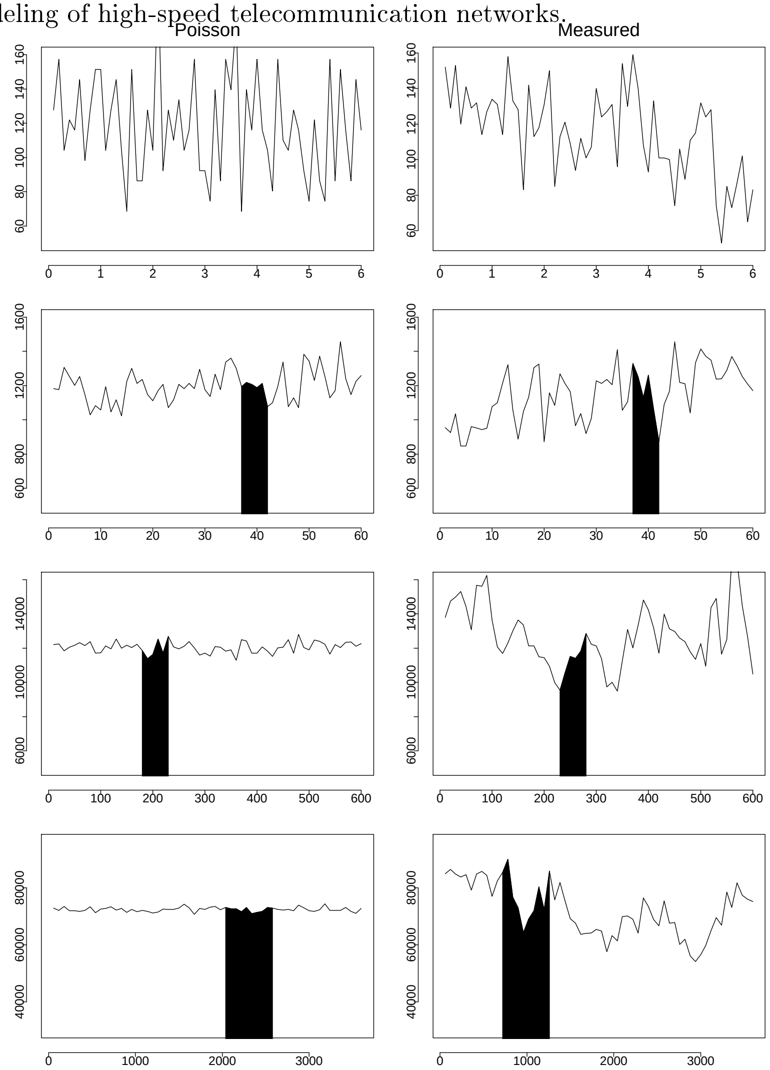

Figure 1: Poisson model vs. Internet traffic

Several researchers have attempted the use of self-similar traffic in a queueing network setting. Shi and Zhu [26] explained 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 [13]); 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. Chan and Li [27] argued that self-similar input traffic (say fractional Brownian motion) to a single queue can be analyzed to obtain standard queue performance metrics. If the system 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.

1.3 Characteristics and causes of self-similarity in network traffic

Park et al [20] attributed LRD or self-similarity in network traffic to heavy-tailed distributions of file sizes. Addie et al [1] state that fractal traffic is characterized by significant long bursts which are caused by down loading large files, or long periods of high levels of VBR video activities, intensive bursts of database activity, and so on. Christensen and Ballingam [5] demonstrate the reduction of self-similarity by application-level traffic shaping.

Whitt [30] explains possible techniques to handle file-sizes with heavy tailed distributions as follows: "Evidently much of the traffic complexity is due to heavy-tailed probability distributions, e.g., of file sizes. Thus there is interest in better understanding the performance impact of heavy-tailed probability distributions. We have been taking three general approaches to queues with heavy-tailed distributions: (i) determining bounds to establish qualitative properties, (ii) exploiting numerical transform inversion to calculate probabilities, and (iii) proving heavy-traffic limit theorems to generate useful approximations. (see Duffield and Whitt [7] as a paper using this approach on: Network Design and Control Using On-Off and Multi-Level Source Traffic Models with Heavy-Tailed Distributions)."

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 [32], 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 $X = (X_k : k \geq 1)$ 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

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

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

A discrete-time, covariance-stationary, zero-mean stochastic process $X = (X_k : k \geq 1)$ 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

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

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

$$VarX^{(m)} = \kappa_1 m^{2H-2}.$$

2 Quality of service (QoS)

The proliferation of users and applications has resulted in severe congestion of the Internet. Internet applications such as video conferencing, Internet telephony (VoIP), telerobotics, mission-critical business information, real-time remote monitoring, remote sensing, real-time planning, scheduling and control of systems, e-business, web-based manufacturing, real-time image transfers, web based education, etc that can tolerate very little packet loss rates and delays are affected during congestion. This has led to users demanding a Quality of Service (QoS) for their applications. QoS is a set of performance measures that the networks or Internet Service Providers (ISPs) guarantee to the users of high-speed networks for their applications. The QoS measures

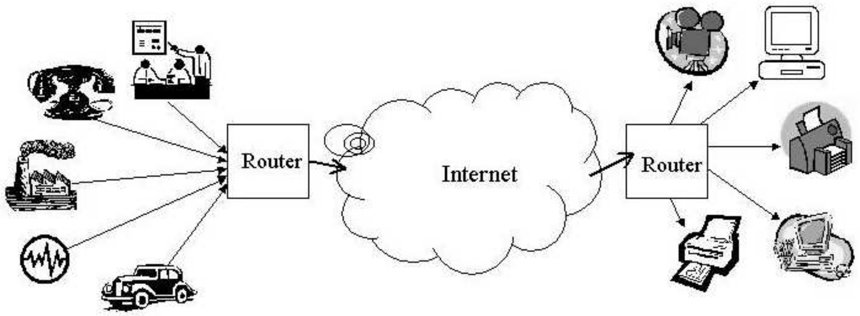

Figure 2: High-speed network with multi-class traffic

typically include bandwidth, delay, delay-jitter, and packet loss. The high-speed network needs to guarantee these QoS measures all the way from the source of information flow to its destination (end-to-end).

The high-speed networks are expected to handle a wide variety of traffic (see Figure 2). Therefore a packet may carry one of the different types of information: voice, video, data, etc. This creates the need to deal with multi-class traffic. For each class of traffic, a different set of QoS requirements, measured by cell-loss probability, delay, delay-jitter, and bandwidth, needs to be assured. For example, real-time traffic has a more stringent delay requirement but can tolerate higher cell-loss; while data traffic can tolerate higher delay but demands much smaller cell-loss.

Sahinoglu and Tekinay [24] state that self-similarity introduces new complexities into optimization of network performance and makes the task of providing QoS together with achieving high-utilization difficult.

3 Performance Analysis Methods

There is an urgent need to develop methods for capacity planning and performance evaluation of networks for the rapidly increasing applications in electronic commerce. In the light of the discussions in the previous sections, 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 [15] and [16]. 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 [29]. 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. (p. 321)" [16]. 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 [4]. In case of software contention they suggest the method of Layered Queueing Networks (LQN), see for example Franks et al [8], Rolia and Sevcik [23], Neilson et al [19] and Woodside et al [34]. Commercial and research software is available to carry out detailed simulation of systems, for example, COMNET (http://www.compuware.com/products/ecosystems/comnet.htm) and ns (http://www-mash.ca.berkeley.edu/ns/).

We restrict our attention to the second class of models described above. The questions we wish to answer are as follows. Can analytical approximations of multi-class queueing network model provide a reasonable first order approximation to the performance of an e-commerce site when the correlation of inter-arrival times is small to moderate? The second question is whether the approximations can provide useful bounds to the average response time for different classes of customers? To answer the first question we describe experiments that we have done to determine how to segment the demand into different classes. The results of these are input to a queueing network model to obtain approximations for the mean delay (or response time) for each class. These are compared with actual observed values. To answer the second question we plan to use recent results from modeling extremal processes to construct approximations for the mean delay, see Toyoizumi, Shanthikumar and Wolff [28].

4 Analysis and Experimental Results

4.1 Scenario: Single server multi-stage system

We consider a typical web-server for an e-business application and model it using a multi-stage queueing network. 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. See Figure 3 for an illustration of the scenario.

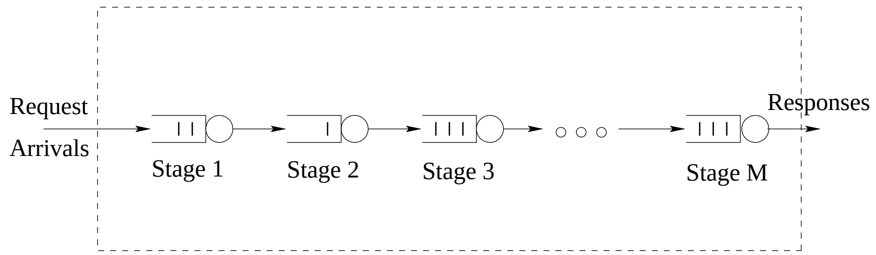

Figure 3: Multi-stage queueing network model of web-server

The request arrival process, is modeled as an autocorrelated process, specifically a fractional brownian motion with a given Hurst parameter. 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 (0.5 in all experiments). Let μ_1, μ_2, $\ldots$, μ_M be the service rates at stages $1, 2, \ldots, M$ respectively. The processing time distribution is exponential at all stages.

There are two key research issues:

1. **Can queueing analysis and approximations be used to predict the large queues?** Note that Sahinoglu and Tekinay [24] 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 present an innovative technique that combines traditional queueing analysis with newly developed approximations to predict the queue sizes accurately.

2. **Can the system be modified suitably in order to reduce the queue sizes?**. From a customer-satisfaction standpoint, it is very im-

portant for the e-business that the queues be kept small. In the sections that follow, we consider suitable enhancements to the current design/architecture and demonstrate their benefits.

Traditional approximation methods to analyze performance of queueing networks such as Whitt [31] and Buzacott and Shanthikumar [3] would severely underestimate the queue sizes. The bounds for tandem queues developed in Seshadri and Pinedo [25] would also be inadequate. However, we 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 key idea is to decompose the queueing behavior into two parts: one due to the burstiness 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 below. The rest of the approximation proceeds by using a queueing network analyzer. 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.

We performed several experiments to demonstrate how our approximation technique predicts the queue sizes accurately. We use Paxson [21], in particular we used FFT-FGN, the S version for Hurst parameter values of 0.5, 0.7, and 0.9 for simulating 1 Million the self-similar arrivals into the system for each experiment. All experiments assume the number of stages $M = 8$. The analytical approximations based on queueing networks are carried out using the software packages in Moses et al [17] and [18]. The total number of customers waiting in the 8 queues (L) are obtained by adding the correction term for traffic burstiness and analytical approximations. A summary of the input parameters and results are described (see Table 1) below.

Note that the approximations are reasonably accurate based on the above results considering that traditional queueing approximations end up in errors ranging from 60% to 80%. However a correction factor to the above methodology (details in a forthcoming paper) results in much better approximations.

4.2 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 4.1 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 4.1. 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 results in Table 2 are similar to the last row of results in Table 1. The number in the queue has indeed reduced but not enough. Also note that the analytical results match the experimental findings reasonably accurately.

Even with 100 servers, the total number in the queue can be brought down to only about 375. In fact, even having 100 servers in the first stage does not affect subsequent queue lengths (see Table 3 below).

4.3 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. In other words the self-similar's traffic's coefficient of variation (ratio of standard deviation to mean) is significantly reduced, see table below:

	Mean	Variance
Before Shaper	1.0356	0.4356
After Shaper	1.0356	0.0911

An experiment which is an enhancement to that in Section 4.2 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 4.1. 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 $N = (N_1 ... N_8)$ is (57321353), and the service rate $\mu = (\mu_1 ... \mu_8)$ is (0.20330.15450.33390.5101.050.36330.2060.4667).

Once again, note that in terms of mean utilizations, the above 2 results are similar to the last row of results in Table 1. The number in the system has

Index	Hurst Parameter	Arrival rate (λ)	Service rate $\mu = (\mu_1 ... \mu_8)$
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)

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

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	65.7	203.7	14.44
2	379.69	65.3	458.8	20.83
3	242.33	65.5	251.0	3.58
4	430.92	65.3	458.8	6.47
5	390.74	65.1	464.9	18.99

Table 1: Experimental results and predicted estimates

# of servers $N = (N_1 ... N_8)$	Service rate $\mu = (\mu_1 ... \mu_8)$	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

Table 2: Results for the multi-server case

Index	# of servers $N = (N_1 ... N_8)$	Service rate $\mu = (\mu_1 ... \mu_8)$
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

Table 3: Results for large number of servers

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

Table 4: Results when system uses a shaper

again reduced but not enough. Also note that the analytical results match the experimental findings reasonably accurately.

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. This leads to extremely large queues in the downstream buffer. Figure 4 represents the inter arrival times of the traffic before the shaper and Figure 5 represents the inter departure times after the shaper. Note the large bursts (corresponding to the troughs) in Figure 5. The low points in the graphs suggest extremely high arrival rates.

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. (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.

4.4 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 an 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 (1) single servers at all stages without shaper, (2) multiple servers at all stages without shaper, and, (3) 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 4.1. 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 $\mu = (\mu_1 ... \mu_8)$ is (1.01671.08131.001671.021.051.091.031.4).

For the multiple server cases the number of servers $N = (N_1 ... N_8)$ is

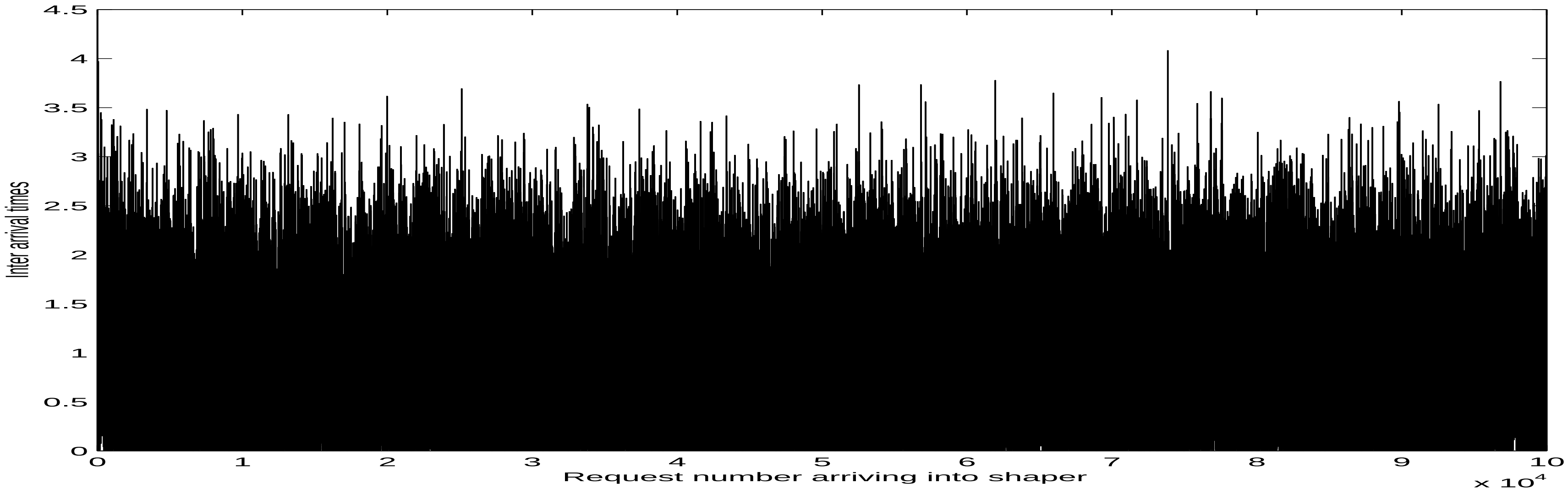

Figure 4: Traffic before shaper

# of servers	Shaper present?	K value	loss rate	Observed # in queue	Estimated # in queue	Error (%)
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
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
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

Table 5: Results with admission control

(57321353), and the service rate
$\mu = (\mu_1 ... \mu_8)$ is (0.20330.15450.33390.5101.050.36330.2060.4667). When there is a shaper, its service rate is 1.00167 units.

Once again, note that in terms of mean utilizations, the above 2 results are similar to the last row of results in Table 1. The number in the system has significantly reduced. Note that the analytical results 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.

5 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

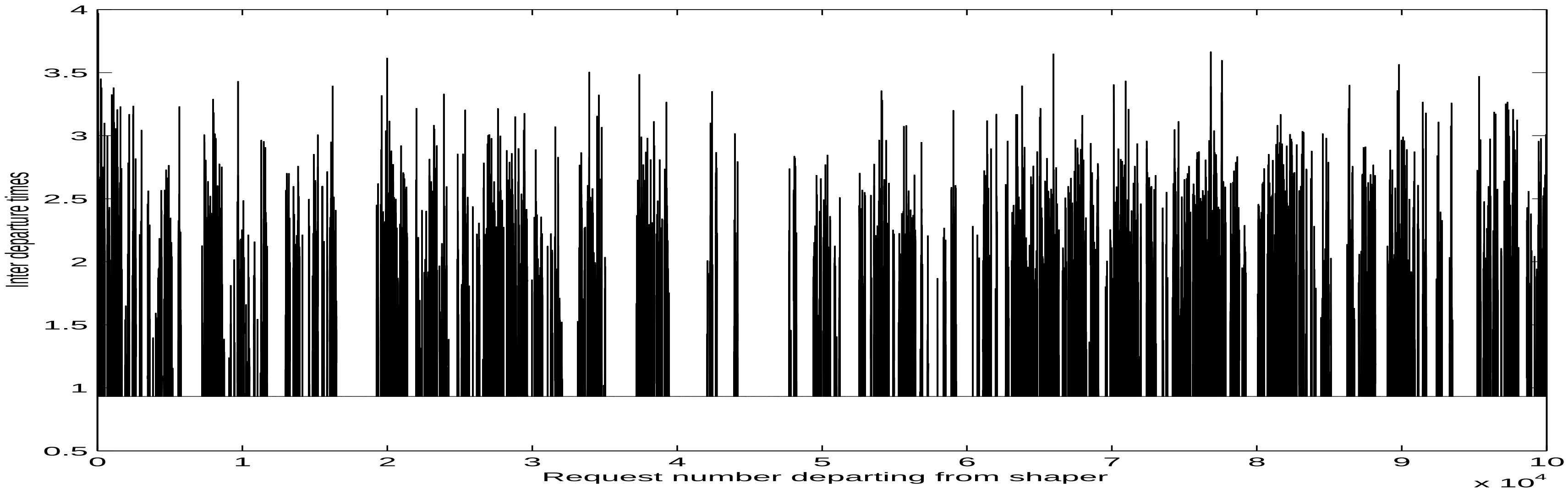

Figure 5: Traffic after shaper

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.

References

[1] R.G. Addie, M. Zukerman, and T.D. Neame. *Broadband traffic modeling: simple solutions to hard problems.* IEEE Communications Magazine, pp88–95, Aug. 1998.

[2] P. Barford and M. Crovella. Generating Representative Web Workloads. Proc. 1998 ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems, Madison, WI, June 22-26., pp151-160.

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

[4] J. P. Buzen. Operational Analysis: An Alternative to Stochastic Modeling. in Performance of Computer Installations, North-Holland, 1978, pp175-194.

[5] K.J. Christensen and V. Ballingam *Reduction of self-similarity by application-level traffic shaping.* Proc. 22nd Annual Conf. on Local Computer Networks, pp 511–518, 1997.

[6] M.E. Crovella and A. Bestavros. *Self-similarity in World Wide Web traffic: evidence and possible causes.* IEEE/ACM Trans. on Networking Vol. 5 No. 6, pp 835–846, Dec. 1997

[7] N.G. Duffield and W. Whitt. *Network Design and Control Using On-Off and Multi-Level Source Traffic Models with Heavy-Tailed Distributions.* Chapter in Self-Similar Network Traffic and Performance Evaluation, Kihong Park and Walter Willinger, eds., 2000.

[8] G. Franks, A. Hubbard, S. Majumdar, D. Petriu, J. Rolia, and C. M. Woodside. A Toolset for Performance Engineering and Software Design of Client-Server Systems. Performance Evaluation Journal, vol. 24, no. 1-2, 1996, pp117-135.

[9] B. V. Gnedenko and I. V. Kovalenko. Introduction to Queueing Theory. Birkhauser Press. Boston, MA, 1989.

[10] M. Grossglauser and J.-C. Bolot. *On the relevance of long-range dependence in network traffic.* IEEE/ACM Trans. on Networking Vol.7, No. 5, pp 629–640, Oct. 1999

[11] D.P. Heyman and T.V. Lakshman. *What are the Implications of Long-Range Dependence for VBR-Video Traffic Engineering?* IEEE/ACM Trans. on Networking, Vol. 4, No. 3, June 1996.

[12] T. Kushida. *The traffic measurement and the empirical studies for the Internet.* GLOBECOM–98, pp 1142-1147, Vol. 2, 1998.

[13] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson. *On the self-similar nature of Ethernet traffic (Extended Version).* IEEE/ACM Trans. on Networking Vol.2, No. 1, pp 1–15, Feb. 1994

[14] M.T. Lucas, D.E. Wrege, B.J. Dempsey, and A.C. Weaver. *Statistical characterization of wide-area IP traffic.* Proc. Sixth International Conference on Computer Communications and Networks, pp 442–447, 1997.

[15] D. A. Menasce and V. A. F. Almeida. *Capacity Planning for Web Performance.* Prentice Hall, PTR, 1998.

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

[17] Moses, M. and S. Seshadri. *Using Modeling Software to Improve Operations.* To appear in International Journal of Operations and Quantitative Management (available at http://www.stern.nyu.edu/HOM)

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

[19] J. E. Neilson, C. M. Woodside, D. C. Petriu, S. Majumdar, Software bottlenecking in client-server systems and Rendezvous Networks, IEEE Transactions on Software Engineering, Sep 1995; Vol. 21, Iss. 9; pg. 776-782.

[20] K. Park, G. Kim, and M.E. Crovella. *On the relationship between file sizes, transport protocols, and self-similar network traffic.* Proc. Intl. Conf. on Network Protocols, pp 171–180, 1996.

[21] V. Paxson. *Fast Approximation of Self-Similar Network Traffic.* technical report LBL-36750/UC-405, April 1995.

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

[23] J. A. Rolia and K. C. Sevcik. The Method of Layers. IEEE Tr. Software Eng., vol. 21, no. 8, 1995, pp689-700.

[24] Z. Sahinoglu and S. Tekinay. *On multimedia networks: self-similar traffic and network performance.* IEEE Communications Magazine, Vol. 37, No. 1, pp 48–52, Jan. 1999

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

[26] J. Shi and H. Zhu. *Merging and splitting self-similar traffic.* Fifth Asia-Pacific Conference on Communications, APCC/OECC–99, pp. 110–114, vol.1, 1999.

[27] Tat-Keung Chan and V.O.K. Li. *Decomposition of network of queues with self-similar traffic.* GLOBECOM–98, pp 3001-3006, Vol. 5, 1998.

[28] H. Toyoizumi, J. G. Shanthikumar and R. W. Wolff. Two Extremal Autocorrelated Arrival Processes. Probability in the Engineering and Informational Sciences, 11, 1997, pp441-450.

[29] Transaction Processing Council. TPC-W, http://www.tpc.org..

[30] W. Whitt. *Performance Models Consistent With Traffic Measurements.* http://www.research.att.com/~wow/performance.html, 2000.

[31] W. Whitt. *The Queueing Network Analyzer.* The Bell System Technical Journal, 62, 9 (1983), p. 2779-2815.

[32] W. Willinger and V. Paxson. *Where Mathematics meets the Internet.* Notices of the American Mathematical Society, 45(8), pp. 961–970, Sept. 1998.

[33] W. Willinger, M.S. Taqqu and A. Erramilli. *A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks.* Stochastic Networks: Theory and Applications, F.P. Kelly, S. Zachary and I. Ziedins (Eds.), Royal Statistical Lecture Note Series, Vol. 4, pp. 339–366, Clarendon Press, Oxford, UK, 1996.

[34] C. M. Woodside, J. E. Neilson, D. C. Petriu, and S. Majumdar. The Stochastic Rendez-vous Network Model for Performance of Synchronous Client-Server-like Distributed Software. IEEE Transactions Computers. vol. 44, no. 1, 1995.