SIMULATION MODELING OF SELF-SIMILARITY IN NETWORK TRAFFIC

Robert Geist
James Westall

Department of Computer Science
Clemson University
Clemson, SC 29634-1906
email: rmg,westall@cs.clemson.edu

Kevin Spicer

Lucent Technologies
Cary, NC 27511 email: spice@lucent.com

A new technique is proposed for synthesizing network packet arrival streams. The technique is a variation on the midpoint subdivision approach to generating sample paths of a stochastic process that exhibits fractional Brownian motion. The new technique and two others are compared in their ability to generate parameterized, stochastically-equivalent synthetic arrival streams. Simulation studies that incorporate arrival time traces captured from production networks are then used to evaluate the capability of the synthetic arrival models to duplicate the congestion levels of the real traffic.

INTRODUCTION

The foundational work of Leland, Taqqu, Willinger, and Wilson [LELA94] has been followed by a large body of research firmly establishing that the Poisson process is a poor model of most network traffic arrival processes. Published analyses of a range of real network traffic studies indicate that the number of packets per unit time arriving at network interfaces that service aggregated traffic sources is both long-range dependent and has a higher variance to mean ratio than the Poisson process.

These results have led to a number of studies proposing new models of the arrival process and its effect on queuing behavior. Most of these models have involved the use of heavy-tailed distributions or mechanisms for synthesizing long-range dependence. Smith [SMIT94] analyzed the performance of a queuing system in which the inter-arrival times were generated by a heavy-tailed distribution. Fiorini, Ding, and Lipsky [FIOR98] proposed a modulated Poisson process in which the modulation was driven by a heavy-tailed distribution.

Other research has focused upon modeling the long-range dependence in the arrival process. A process is said to be long-range dependent if its autocorrelation function is non-summable. Self-similar processes are a subclass of the long-range dependent processes and are characterized by a hyperbolically-decaying autocorrelation function. Finally, fractional Gaussian noise (FGN) processes are a subclass of the self-similar processes. Paxson [PAXS94] and Lau, Erramilli, Wang, and Willinger [ERRA95] have proposed somewhat orthogonal methods for generating approximate FGN.

Taqqu, Willinger, and Sherman [TAQQ97] provided a mathematical result unifying long-range dependence and heavy-tailed modulated processes. They showed that the superposition of an infinite number of packet-train sources in which the duration of the ON and OFF periods was heavy-tailed produced a process that was long-range dependent.

This paper reports a blend of theoretical and experimental results from the domain of self-similar models of network traffic arrival processes. In the next section we present a new technique for approximating fractional Brownian motion (from which fractional Gaussian noise may easily be derived.) Our approach is a refinement of the one proposed by Lau, Erramilli, Wang, and Willinger [ERRA95]. In the third section we experimentally compare aspects of the performance of our FGN generator with the two previously cited. The ability of the three parameter FGN traffic model to predict the queuing behavior of real network traffic is considered in the fourth section. Using arrival time traces captured from production networks, we show that the three parameter FGN model, while clearly better than a Poisson model, has serious shortcomings in modeling real traffic. In the final section we present some conclusions that can be drawn from this work.

ARRIVAL MODEL

Recent studies by Erramilli, Narayan, and Willinger [ERRA96] and by Taqqu, Willinger, and Sherman [TAQQ97] have established both empirical and theoretical foundation for network traffic models in which aggregate packet arrival streams exhibit *self-similarity*. A stochastic process, $X(t)$, is self-similar if there is some constant, $H \in (0,1)$, called the *Hurst parameter*, such that $X(t)$ has a distribution identical to $(1/w^H)X(wt)$ for any t and $w > 0$. This says that the "speeded-up process" is a scaled version of the original.

The classic example of self-similarity is *fractional Brownian motion (FBM)* exhibited by a process, $X(t)$, that has Gaussian increments, specifically:

- $X(0) = 0$
- the increment $X(t_2) - X(t_1)$ has Gaussian distribution, for all $t_1 < t_2$
- the increment mean, $E[X(t_2) - X(t_1)] = 0$
- the increment variance, $V[X(t_2) - X(t_1)] \propto |t_2 - t_1|^{2H}$

If $X(t)$ satisfies these conditions, then $V[w^H X(t)] = w^{2H} V[X(t)] = w^{2H} K t^{2H}$ for some proportionality constant, K, and $V[X(wt)] = K(wt)^{2H}$. Since $w^H X(t)$ and $X(wt)$ are both Gaussian, the self-similarity follows.

An FBM process exhibits *long-range dependence*, which means that the auto-correlation of the increments is a slowly decaying function of increment size. Specifically, the normalized covariance of the increments,

$$\begin{aligned} r(t) &= COV[X(1) - X(0), X(t+1) - X(t)] / \\ &\quad \sqrt{V[X(1) - X(0)] V[X(t+1) - X(t)]} \\ &= (1/2)((t+1)^{2H} - 2t^{2H} + (t-1)^{2H}) \qquad (1) \\ &= H(2H-1)t^{2H-2} + \sum_{j=2}^{\infty} \binom{2H}{2j} t^{2H-2j} \qquad t > 1 \end{aligned}$$

where the last equation follows from the Maclaurin series for $(1+1/t)^{2H}$. Note that for $H > (1/2)$ we have positive, slowing decaying $r(t)$. There is evidence that this long-range dependence is the primary cause of relatively poor queuing performance exhibited by real switches [ERRA96].

Midpoint Displacement

In the network modeling community there is substantial interest in fast techniques for generating synthetic arrival streams that exhibit self-similarity, and a variety of such techniques have been suggested [PAXS94]. We focus here on techniques collectively termed "random midpoint displacement." In these techniques, sample values of a process, $X(t)$, are specified at endpoints of a time interval, $[0, T]$, and then the value at the midpoint, $X(T/2)$, is specified as the average of the values at the endpoints plus some random displacement. This procedure is repeated on subintervals to an arbitrary depth. The displacements are not chosen arbitrarily, but rather chosen to effect Gaussian increments.

In the standard approach, which can be found in both the computer graphics literature [PEIT88] and the computer networks literature [ERRA95], $X(0)$ is assigned a value 0, and a value for $X(T)$ is selected at random from a Gaussian distribution with mean 0 and variance $\sigma^2 T^{2H}$, where σ^2 is a target proportionality constant. The midpoint, $X(T/2)$, is then defined as $X(T/2) = (X(0) + X(T))/2 + D$ where D is Gaussian with mean 0 and variance, $V[D] = \sigma^2 T^{2H}(1 - 2^{2H-2})/2^{2H}$. As a result, the increment, $X(T/2) - X(0)$, is Gaussian with mean 0 and variance

$$\begin{aligned} V[X(T/2) - X(0)] &= V[(X(T) - X(0))/2 + D] \\ &= (1/4)\sigma^2 T^{2H} + \\ &\quad \sigma^2 T^{2H}(1 - 2^{2H-2})/2^{2H} \qquad (2) \\ &= \sigma^2 (T/2)^{2H} \end{aligned}$$

as desired. The same holds for increment $X(T) - X(T/2)$. This procedure is then applied to subintervals $[0, T/2]$ and $[T/2, T]$, but a different displacement variance is required to maintain the proportionality constant. It is straightforward to calculate that the required variance at subdivision level i is

$$V[D_i] = \sigma^2 T^{2H} \left[\frac{1 - 2^{2H-2}}{(2^{2H})^{i+1}} \right] \qquad (3)$$

for $i \geq 0$.

Although this procedure at first appears to achieve the desired goal of Gaussian increments, it does not, unless $H = 1/2$. For example, it is straightforward to check that, after two subdivisions,

$$\begin{aligned} V[X(T) - X(T/4)] &= \sigma^2 T^{2H}((3/4)^2 - \\ &\quad (1/4)^2 + (1/2^{2H})^2) \qquad (4) \\ &\neq \sigma^2 T^{2H}(3/4)^{2H} \end{aligned}$$

unless $H = 1/2$. Thus midpoint displacement techniques produce only approximate Gaussian increments, and this is the primary motivation in seeking other techniques. Nevertheless, the closeness of the approximation and the speed and simplicity of routines for sample path generation have made midpoint displacement techniques popular.

Successive Random Additions

A variation on the standard approach, called *successive random additions*, is more commonly seen in the graphics literature [PEIT88]. It was motivated by the fact that the standard approach can generate undesirable visual artifacts, e.g., synthetic mountain ranges with "creases" located at

midpoints that are fixed early in the construction. In this variation, a Gaussian displacement is added to all points on each iteration, not just to the new midpoints. If $D_i(t)$ is the i.i.d. displacement process for subdivision level i, and $X_i(t)$ is the constructed process at subdivision level i, then

$$\begin{aligned} X_1(0) &= X_0(0) + D_0(0) \\ X_1(T/2) &= (X_0(0) + X_0(T))/2 + D_0(T/2) \\ X_1(T) &= X_0(T) + D_0(T) \end{aligned} \quad (5)$$

and so

$$\begin{aligned} V[X_1(T/2) - X_1(0)] &= V[(1/2)(X_0(T) - X_0(0)) + \\ &\quad D_0(T/2) - D_0(0)] \\ &= (1/4)\sigma^2 T^{2H} + 2V[D_0] \end{aligned} \quad (6)$$

Since the target is $V[X_1(T/2) - X_1(0)] = \sigma^2(T/2)^{2H}$, we must have $V[D_0] = (\sigma^2 T^{2H}/2)((1 - 2^{2H-2})/2^{2H})$. In the general case, $V[D_i]$ is selected to be exactly half of that in (3).

Although the approach of successive random additions is successful in removing visual artifacts from the selected sample paths, it is no closer to Gaussian increments than the standard approach. In fact, it can be argued that it is further. In the limit, we have

$$\begin{aligned} V[X_{+\infty}(T) - X_{+\infty}(0)] &= V[X_0(T) - X_0(0) + \\ &\quad \sum_{i=0}^{+\infty} D_i(T) - \sum_{i=0}^{+\infty} D_i(0)] \\ &= \sigma^2 T^{2H} \\ &\quad \left[1 + \frac{1 - 2^{2H-2}}{2^{2H} - 1}\right] \\ &\neq \sigma^2 T^{2H} \end{aligned} \quad (7)$$

For $H = 1/2$, the proportionality "constant" has grown by 50%! Further, neither the new, larger proportionality constant of (7) nor the original one (σ^2) matches the value required for other increments.

Floating Proportionality

We propose here a modification of successive random additions that, we contend, continues to provide artifact-free, visually believable sample paths and yet is more careful to constrain the variances of the most important increments. The key idea is to allow the proportionality constant to change in a controlled way during the subdivisions.

We begin at subdivision level 0 with endpoints $X_0(0)$ and $X_0(T)$ that satisfy $V[X_0(T) - X_0(0)] = K_0 T^{2H}$ for initial constant, $K_0 = 1$, and we define $X_0(T/2) = (X_0(0) + X_0(T))/2$, as before. To move to level 1, we define $X_1(t) = X_0(t) + D_0(t)$ for $t = 0, T/2, T$, where $D_0(t)$ is a Gaussian i.i.d. displacement process, to be specified. The goals for the level 1 increments are:

$$\begin{aligned} V[X_1(T) - X_1(0)] &= K_1 T^{2H} \\ V[X_1(T/2) - X_1(0)] &= K_1 (T/2)^{2H} \end{aligned} \quad (8)$$

for some constant K_1. From the definitions of $X_1(t)$, we know

$$\begin{aligned} V[X_1(T) - X_1(0)] &= K_0 T^{2H} + 2V[D_0] \\ V[X_1(T/2) - X_1(0)] &= (1/4)K_0 T^{2H} + 2V[D_0] \end{aligned} \quad (9)$$

From (8) and (9) we conclude that

$$V[D_0] = (K_0/2)T^{2H}\left[\frac{1 - 2^{2H-2}}{2^{2H} - 1}\right] \quad (10)$$

and that

$$K_1 = K_0 + 2V[D_0]/T^{2H} \quad (11)$$

We continue in this fashion, specifically, for level $i \geq 0$ we define

$$\begin{aligned} V[D_i] &= (K_i/2)T^{2H}\left[\frac{1 - 2^{2H-2}}{(2^{2H})^{i+1} - 1}\right] \\ K_{i+1} &= K_i + 2V[D_i]/T^{2H} \end{aligned} \quad (12)$$

In this fashion, we guarantee that the smallest increments are always correctly related to the largest, i.e.,

$$\begin{aligned} V[X_i(T) - X_i(0)] &= K_i T^{2H} \\ V[X_i(T/2^i) - X_i(0)] &= K_i (T/2^i)^{2H} \end{aligned} \quad (13)$$

Following Norros [NORR94], we then specify an aggregate packet arrival stream as

$$A(t) = mt + \sqrt{ma}X_{i_f}(t) \quad (14)$$

where m is mean arrival rate, a is a variance parameter, and i_f is the final subdivision level. We choose time units so that $T = 2^{i_f}$. Thus the final increment variance, $V[X_{i_f}(1) - X_{i_f}(0)] = K_{i_f}$, and if σ^2 is the target increment variance, we need only choose $a = \sigma^2/(K_{i_f} m)$.

The implementation of this procedure is nearly trivial; in C it is:

```
T_2H = pow(T,2.0*H);
two_2H = pow(2.0,2.0*H);
K = 1.0;
sigmaD = sqrt(K*T_2H);

X[0] = 0.0;
X[SAMPLES] = get_z(sigmaD);
step=SAMPLES;
half_step=step/2;

for(j=0; half_step >= 1;
          step /= 2,half_step /=2,j++)
```

```
{
   for(i = 0; i < SAMPLES; i += step)
      X[i+half_step] = (X[i]+X[i+step])/2.0;

   varD = (K*T_2H/2.0)*((two_2H/4.0 - 1.0)/
          (1.0 - pow(two_2H,j+1)));

   sigmaD = sqrt(varD);

   for(i = 0; i <= SAMPLES; i += half_step)
      X[i] += get_z(sigmaD);

   K += 2.0 * varD/T_2H;
}
```

The routine *get_z()* generates a random deviate from a normal distribution of the indicated standard deviation. Although the procedure is clearly $O(nlog(n))$ in the number of samples, execution time is negligible for relatively large samples, e.g., 128K samples can be generated in less than 1 second user time on a modern cpu (e.g. SGI R12k).

Packet Dispersion

A difficulty arises in using this model in simulation studies of switch queuing behavior. Given a target Hurst parameter and target mean and variance for small (e.g. 1 second) increments, we can certainly use our model to generate aggregate packet arrival counts, $A(t+1)-A(t)$ for these increments. Nevertheless, to use these counts in a simulation of queuing behavior, we still need a method of generating individual packet arrival instants within the small increments. Using batch arrivals at increment boundaries provides an extremely pessimistic view of queuing behavior, and using uniformly distributed packets provides an extremely optimistic view. Although we can use our model to generate packet arrival counts for much smaller increments, e.g., 1 millisecond, at this level the self-similar model departs from real data [ERRA95].

As a simple compromise between batch and uniformly distributed arrival instants, we distribute according to the density $f(x) = e^{-x}/(1 - e^{-1})$. We will see that this distribution is fairly effective in allowing us to simulate real queuing behavior.

PERFORMANCE EVALUATION

In this section we evaluate the performance of our traffic generator, *fBm*, and two others that have been proposed in the networking research literature [ERRA95, PAXS94]. All three generate a sequence of values whose distribution is an approximation of fractional Gaussian noise (FGN). The individual values of the FGN sequence, N_t, represent the number of packet arrivals per given unit of time. Since fractional Brownian motion results from the integration of FGN, given an FBM process, X_t, the corresponding FGN process, N_t, can be recovered using the relationship, $N_t = X_{t+1} - X_t$. In the remainder of the paper the focus will be upon FGN and the notation N_t is used to distinguish it from FBM.

All three generators are driven by three parameters: the mean number, μ, of arrivals per unit time; the variance, σ^2, of the arrival process; and the Hurst parameter H, $(0.5 <= H < 1.0)$, specifying the degree of self-similarity. We use the notation (FGN, μ, σ^2, H) to specify a specific FGN counting process.

The generator *pax* is Christian Shuler's implementation of a method proposed by Paxson [PAXS94]. Paxson's method is derived from Beran's [BERA86] functional representation $f(\lambda; H)$ of the power spectrum of an FGN process. Paxson observed that the discrete sequence $\{f_1, \ldots, f_{n/2}\}$ where $f_j = f(2\pi j/n; H)$ can be stochastically transformed into the discrete Fourier transform (DFT) of a self-similar time series of n points. The first step in the transformation is to multiply the f_j's by an exponential random variable with mean 1. This operation produces the periodogram of the self-similar time series. Next the complex series, $\{z_1, \ldots z_{n/2}\}$, where $z_i = \sqrt{f_i}e^{i\theta}$, is generated. The phase, θ, is randomly chosen from $[0, 2\pi]$. The second half of this series is completed by setting $f_{n/2+k}$ equal to the complex conjugate of $f_{n/2-k}$ so that it represents the discrete Fourier transform of a real-valued series. The self-similar time series is obtained by applying the inverse DFT.

The generator *rmd* is our implementation of the random midpoint displacement method described by Lau, Erramilli, Wang, and Willinger [ERRA95]. The advantages and disadvantages of this method were addressed in the previous section.

A number of measures might be considered in evaluating the performance of a generator of synthetic network traffic. The speed of generation and degree to which the statistical parameters of the generated traffic can be made to match target values are clearly important. However, the ultimate measure of effectiveness is the degree to which the queuing behavior of the synthetic traffic matches the queuing behavior of "real-world" traffic. In the remainder of the paper we will address each of these issues in turn.

All three generators are so fast that running time is not a major issue. Both *fBm* and *pax* are $O(nlog(n))$ in the number of samples generated and thus are perceptibly slower than *rmd*. As noted earlier *fBm* can generate a series of 128K samples in less than 1 second. The same number can be produced by *pax* in less than 5 seconds.

Any time series of non-constant values can be transformed to any desired variance and mean by performing a multiplicative scaling followed by an additive one, and the Hurst parameter is invariant under linear transformations. Therefore, the only real problem in parameter matching is to match the Hurst parameter.

We use three techniques to obtain estimators of the Hurst parameter: the variance-time plot; R/S analysis; and Whittle's estimator. The first two methods employ statistical analysis of aggregated, adjacent blocks of the FGN series.

For a given sample $\{N_t\}_{t=0,1,...}$, the aggregated samples, ${N_t}^{(m)}, m = 2, 3, ...$, are formed by aggregating the non-overlapping, adjacent blocks as follows:

$$ {N_t}^{(m)} = \frac{1}{m} \sum_{j=tm-m+1}^{tm} N_j \qquad (15) $$

If the process $\{N_t\}$ is self-similar, the aggregated processes have, for all m, the same autocorrelation function as the original process. Furthermore, it can be shown that

$$ V[{N_t}^{(m)}] = m^{-\beta} V[N_t] \qquad (16) $$

where $\beta = 2 - 2H$. Therefore $log(V[{N_t}^{(m)}])$ is a linear function of $log(m)$ has slope $-\beta$, and the value of β may be obtained by linear regression after a collection of values of $log(V[{N_t}^{(m)}])$ have been computed. The goodness-of-fit of the regression is an indicator of the degree of self-similarity in the original process.

The $(R/S)_n$ statistic was discovered by Hurst and thus is the original method for computing the Hurst parameter. It is also computed by analyzing adjacent, length n sub-blocks of the original time series. Hurst showed that $(R/S)_n = c \times n^H$. Thus, $\log((R/S)_n)$ is a linear function of $log(n)$ and has slope H. The details of the computation of of $(R/S)_n$ are given by Peters [PETE94].

Whittle's maximum likelihood estimator can also be used to compute the Hurst parameter after it has been determined that a time series is self-similar. It is based upon Beran's formulation of the power spectrum of an FGN process. Intuitively, it seeks to find an H such that the associated power spectrum most closely corresponds to periodogram of the time series being evaluated.

All three traffic generators did an acceptable job of producing time series having Hurst parameters that were consistent and reasonably near the target value. Results for a target H of 0.80 are shown in table 1. Mean and standard deviations are based upon 10 runs, each generating a time series of length 32768. The last two columns contain the mean square errors of the R/S and V/T regressions. Results for other values of H were comparable.

Queuing Behavior of Synthetic Traffic

We now turn our attention to characterizing the queuing behavior of the synthetic workloads. An FCFS deterministic server is simulated, and the primary statistic of interest is the mean population of the server. Since the objective of all three generators is to produce an (FGN, μ, σ^2, H) counting process, the queuing behavior of the synthetic traffic produced by all three generators should be stochastically equivalent.

One factor complicates the implementation of the simulation. The synthetic traffic generators provide packet counts per unit time, but the simulation requires arrival times for individual (or batches of) packets. Therefore, a method for generating arrival times from packet counts must be provided. If all packets are considered to arrive simultaneously at the start of the unit time, queue lengths will be grossly overestimated. Conversely, if the arrival times are evenly distributed throughout the unit time, queue lengths will be seriously underestimated. For the results reported in this section the truncated exponential distribution described previously was used.

To test for differences in the arrival processes of the three generators, synthetic (FGN, 100, 625, 0.8) workloads produced by all three were used to drive simulations. The parameterization was chosen to reflect values consistent with those observed in real traffic. The number of seconds simulated was 32768 so that each simulation processed over 3 million arrivals. Since the generators were identically parameterized, virtually identical queuing behavior should be expected. Table 2 reports mean population as a function of utilization. Variability of the mean population increased with utilization, and, therefore, the simulations were repeated eight times for $\rho = 0.65$. The mean and standard deviations obtained in those eight replications appear in the last two lines of the table. The *fBm* and *pax* workloads provide very close agreement, but all three provide reasonably close agreement across the utilizations evaluated.

The workload labeled *pxe* is the *pax* workload passed through a post-processing exponential filter and then renormalized to have the target mean and variance. Exponentiation is suggested (with caveats) by Paxson as a possible way to eliminate negative packet counts that might arise in the original FGN. However, it is clear from this example that it fundamentally alters the nature of the traffic generated.

The expected populations for a comparable M/D/1 queuing system are also provided as a point of reference. For the M/D/1 system, the arrival counting process is Poisson and the mean and variance are identical. Since the variances of the synthetic workloads are 6.25 times the mean, one would expect worse queuing behavior even in the absence of self-similarity.

Workload	R/S	V/T	Whittle	R/S MSE	V/T MSE
Mean (*fBm*)	0.79	0.76	0.74	0.0003	0.0018
Std Dev	0.019	0.038	0.010	0.0002	0.0009
Mean (*pax*)	0.80	0.79	0.79	0.0004	0.0013
Std Dev	0.010	0.042	0.014	0.0004	0.0012
Mean (*rmd*)	0.79	0.77	0.81	0.0003	0.0016
Std Dev	0.010	0.018	0.005	0.0001	0.0012

Table 1: Hurst parameter matching

Util (ρ)	MD1	fBm	rmd	pax	pxe
.20	0.23	0.23	0.23	0.23	0.29
.25	0.29	0.30	0.30	0.30	0.42
.30	0.36	0.39	0.39	0.39	0.60
.35	0.44	0.48	0.48	0.48	0.85
.40	0.53	0.60	0.60	0.60	1.22
.45	0.63	0.77	0.77	0.77	1.74
.50	0.75	1.00	1.00	1.01	2.46
.55	0.89	1.35	1.35	1.36	3.51
.60	1.05	1.88	1.89	1.90	5.12
.65	1.25	2.77	2.76	2.79	7.62
Mean ($\rho = 0.65$)		2.754	2.788	2.746	
StdDev		0.038	0.020	0.037	

Table 2: Mean population: (FGN, 100, 625, 0.8)

Util (ρ)	MD1	fBm	rmd	pax
.20	0.23	0.23	0.23	0.23
.25	0.29	0.30	0.30	0.30
.30	0.36	0.39	0.38	0.39
.35	0.44	0.48	0.48	0.48
.40	0.53	0.61	0.61	0.60
.45	0.63	0.77	0.77	0.77
.50	0.75	1.00	1.00	1.00
.55	0.89	1.35	1.35	1.35
.60	1.05	1.87	1.88	1.87
.65	1.25	2.65	2.67	2.65
Mean ($\rho = 0.65$)		2.661	2.659	2.660
StdDev		0.005	0.009	0.014

Table 3: Mean population: (FGN, 100, 625, 0.55)

We now turn to the issue of sensitivity of the server population to variations in the Hurst parameter. The simulation just described was rerun with the Hurst parameter reduced from 0.80 to 0.55. The results are shown in table 3. The mean population values obtained using traffic from the three generators are now virtually identical. A statistically significant reduction in the mean populations is evident at the higher utilizations, but the $H = 0.55$ results are still far closer to the $H = 0.80$ results than to the M/D/1 results. This indicates that H is significant but secondary to the variance in determining system performance. For example, if the variance is increased from 625.0 to (an experimentally determined) 666.0 the mean population will return to 2.75 at utilization $\rho = 0.65$.

We now consider the effects of changing the mean number of packets arriving per unit time. If we aggregate our original (FGN, 100, 625, 0.80) synthetic trace by summing pairs of adjacent elements, the aggregated series will have mean 200.0. If we re-compute the variance of the aggregated trace we will find it approximately $625.0 \times 2^{2H} = 1894.0$. Thus, an (FGN, 200, 1894, 0.80) series is stochastically equivalent to the aggregation of the first series. However, if we construct such a series and process it with the simulator we obtain a mean population of 3.30 at $\rho = 0.65$, significantly larger than expected.

This effect is a consequence of the problematic interactions of the FGN process with the packet dispersion procedure that generates inter-arrival times. It is clearly the case that the larger the mean packet count, the more effect that procedure has on the operation of the simulated system.

This important point may be restated as follows. If we simply change the arrival rate from 100.0 to 200.0 and do not adjust the variance, we still have a self-similar, FGN time series. However, it is stochastically different from the original, and we have no reason to expect invariant queuing behavior. If we *do* make the appropriate variance adjustment, then the new series is stochastically equivalent to the original when viewed at a coarser or finer time scale. In this case we should be able to expect invariance in queuing behavior, and the nature of any change in behavior is a reflection of the way in which the inter-arrival time generator is altering the behavior the system.

For another example of this effect, we change our inter-arrival time generator from the truncated exponential to one that uniformly distributes arrivals across the unit time interval. One would expect the smoother arrival process to lead to smaller mean populations, and that is the case. We obtain a mean population of 1.78 for the (FGN, 100, 625, 0.80) workload at $\rho = 0.65$, but a mean population of 1.66 for the (FGN, 200, 1864, 0.80) workload. Thus, the inter-arrival time generator can also determine whether mean queue populations grow or shrink at different levels of aggregation.

MODELING REAL TRAFFIC

We now turn to the problem of modeling real-world data. We first consider two publicly available traffic archives. The first, BCR, is the BC-pAug89 archive captured at Bellcore's Morristown Research and Engineering Center. It is available at http://ita.ee.lbl.gov/html/contrib/BC.html. The second, DEC, is the TCP traffic portion of the dec-pkt-1 trace, captured at DEC's Western Research Lab. It is available at http://ita.ee.lbl.gov/html/contrib/DEC-PKT.html. Both of these traces contain the arrival time of each packet at millisecond or better precision. Thus, arrival counts for any desired unit of time can be readily obtained.

By using traces from which true inter-arrival times can be deduced, it is possible to investigate two questions. The first is: "How closely does the series of aggregated arrival counts resemble an FGN counting process?" If the queuing behavior of a simulation driven by the aggregated arrival counts is markedly different from that of synthetic FGN workloads having identical parameterization, then the answer is clearly "not closely." If similar queuing behavior is observed *at several levels of aggregation*, then the answer is likely to be "closely."

The second question is: "How closely do the inter-arrival times generated by the simulator resemble the inter-arrival times in the original trace?" If the queuing behavior of a simulation driven by the actual aggregated arrival counts is markedly different from the queuing behavior obtained by driving the simulation with the original inter-arrival times, then the answer is again "not closely."

We use the BCR trace to illustrate these issues. It contains 1 million packet arrival times spanning 3142.8 seconds. Table 4 shows sample means and variances obtained when the trace is aggregated over 6 orders of magnitude. The predicted variance is obtained by multiplying the preceding variance by 2^{2H} where H was computed to be 0.845. Thus, the data does appear to be self-similar over several orders of aggregation.

Three simulation studies were conducted and the results are reported in table 5. The column labeled *Inter-arrival*

Parameter	BCR	DEC
μ	318	598
σ	114	193
σ/μ	0.36	0.32
H	0.845	0.84

Table 6: BCR and DEC parameterization

contains the results that were obtained when the true inter-arrival times were used to drive the simulator. Since deterministic service times are used in all cases, this simulation is deterministic. In the other two cases, eight runs were made for each value of ρ, and means and standard deviations are reported. For the results reported in the column labeled *Synthetic*, a unique (FGN, 318.17, 13112, 0.845) arrival process was generated and used to drive each simulation. For the results reported in the column labeled *Aggregated*, a file of 3142 packet counts representing the true number of arrivals in each 1 second interval was used to drive the simulation.

For the *Synthetic* and *Aggregated* simulations, packet inter-arrival times were stochastically generated from identical distributions. Therefore, differences in these columns *must* indicate that the actual arrival counting process is not well modeled by the FGN synthetic counting process. Conversely, differences in the *Aggregated* and *Inter-arrival* simulations are attributable to the synthesis of inter-arrival times. It is clear from the table that in this case the major source of inaccuracy is the assumption that the arrival process is well-modeled by an (FGN, 318.17, 13112, 0.845) distribution. Nevertheless, the behavior of the system driven by the synthetic traffic is far closer to the behavior of the actual system than it is to the behavior of the M/D/1 system.

We then replicated the same experiments using the DEC trace. It contains 2,153,461 arrival times spanning one hour of real time. Its parameterization is compared to the BCR trace in table 6. In addition to the similarity of the Hurst parameters and coefficients of variation, DEC actually looks quite similar to a fractionally aggregated version of BCR. The ratio of the means gives the fractional aggregation factor of 1.881. The variance of BCR is 12296 and $12296 \times 1.881^{2\times 0.84}$ is 37769, quite close to DEC variance of 37249.

The results of the simulations are shown in table 7. As expected the DEC based synthetic workload produced queuing behavior very similar to its BCR based counterpart. However, the actual DEC workload produced queuing behavior far worse than BCR. This indicates that the actual counting process is significantly worse, in terms of generating queuing delay, than the actual BCR counting process, which is, in turn, significantly worse than FGN.

As a final experiment in this series, we exponentiated the FGN counting process and renormalized it to the target

Resolution	Mean	True Var	Pred Var	Pred::True
0.125	39.77	477		
0.25	79.55	1526	1537	1.00
0.5	159.08	4348	4925	1.13
1.0	318.17	13112	14029	1.06
2.0	636.34	42341	42308	0.99
4.0	1272.71	135645	136617	1.00

Table 4: Bellcore trace aggregation statistics.

	Synthetic		Aggregated		Inter-arrival
Util	Mean	Std Dev	Mean	Std Dev	Mean
0.20	0.235	0.000	0.236	0.000	0.221
0.25	0.312	0.000	0.315	0.000	0.291
0.30	0.406	0.002	0.422	0.002	0.395
0.35	0.537	0.006	0.612	0.008	0.607
0.40	0.785	0.029	1.019	0.012	1.047
0.45	1.288	0.050	1.996	0.016	2.039
0.50	2.275	0.090	4.258	0.022	4.367
0.55	4.299	0.119	9.784	0.034	10.001
0.60	8.738	0.877	21.522	0.043	21.587
0.65	17.213	1.561	45.798	0.046	45.446

Table 5: BCR trace: Mean population

mean and variance (as was done in the pxe column of table 2). While studies have shown that exponentiated FGN remains self-similar [PAXS94], it is clearly no longer FGN. As can be seen in table 8 the performance of the synthetic workload is now worse than the actual workload.

These results demonstrate that FGN traffic models, while clearly better than Poisson models, have signficant limitations in their ability to predict the behavior of real traffic. To better understand the nature of the limitations of FGN we turn to the higher central moments of the distributions in question. The *coefficient of skewness* is a dimensionless quantity used to characterize the third moment of a distribution sample. Its value is $\frac{1}{n\sigma^3}\sum_{j=1}^{n}(x_j-\mu)^3$ [JAIN91]. Since the Gaussian distribution is symmetric with respect to its mean, all FGN distributions have a coefficient of skewness of 0.0. However, the BCR packet count data, the DEC packet count data, and one instance of the exponentiated FGN data cited in table 8 have coefficients of skewness of 0.69, 1.41, and 7.15 respectively.

Based upon analysis of traffic at our own site, we suspect that positive coefficients of skewness are pervasive. For a period of two weeks, we captured input and output packet counts each second at the interface of a departmental router to the external internet. Approximately 200 local IP addresses are serviced by this router. The per-second input packet counts were then analyzed in hourly batches of 3600. Only two of the 336 hourly batches had negative coefficients of skewness. Mean skewness was 3.19. In a comparable analysis of an enterprise router serving several thousand IP addresses, we found mean skewness closer to that of the BCR packet counts, but only 15 of 336 samples with negative skewness.

As a final data point on the effect of skewness, we compared the simulated queuing behavior of one of the hourly batches that had negative skewness and a synthetic FGN workload of identical mean, variance, and H. In contrast to the BCR and DEC studies, the FGN workload yielded *higher* queue populations than did the captured data.

CONCLUSIONS

We have proposed a new method for generating approximate FGN and demonstrated its use in synthesizing network traffic. This new method, employing a variation on the technique of random midpoint displacement, was shown to correct some technical shortcomings of related methods. However, simulation studies showed the queuing behavior of the synthetic traffic generated by all three methods analyzed to be quite similar.

Analyis of FGN-based synthetic traffic indicates that the Hurst factor, while statistically significant, is clearly secondary to the variance in determining its queuing behavior. Analysis of multiple collections of real network traffic demonstrates that FGN traffic models, while clearly better than Poisson models, have significant limitations in their ability to predict the queuing behavior of real traffic. However, it would be incorrect to conclude that all long-range dependent or even all self-similar traffic models share this deficiency. We conjecture that a family of self-similar or

	Synthetic		Aggregated		Inter-arrival
Util	Mean	Std Dev	Mean	Std Dev	Mean
0.200	0.233	0.000	0.236	0.000	0.289
0.250	0.309	0.000	0.343	0.005	0.391
0.300	0.399	0.001	0.652	0.017	0.568
0.350	0.519	0.003	3.236	0.018	3.477
0.400	0.731	0.018	11.946	0.010	12.582
0.450	1.271	0.062	22.197	0.012	23.163
0.500	2.445	0.078	34.708	0.016	35.168
0.550	5.072	0.358	50.452	0.039	50.320
0.600	9.830	0.632	73.838	0.050	73.143
0.650	20.786	1.762	114.535	0.062	112.584

Table 7: DEC trace: Mean population

	Synthetic		Inter-arrival
Util	Mean	Std Dev	Mean
0.200	1.367	1.394	0.289
0.250	1.604	0.894	0.391
0.300	3.887	1.748	0.568
0.350	5.773	1.432	3.477
0.400	10.227	2.468	12.582
0.450	20.238	7.033	23.163
0.500	28.565	9.996	35.168
0.550	50.826	15.256	50.320
0.600	84.370	17.011	73.143
0.650	143.451	32.116	112.584

Table 8: Exponentiated FGN

long-range dependent distributions in which it is possible to specify the mean, variance, and skewness independently would provide significantly improved performance prediction.

REFERENCES

[BERA86]J. Beran, *Estimation, Testing, and Prediction for Self-Similar and Related Processes*, Ph.D. dissertation, ETH, Zurich, 1986.

[ERRA96]A. Erramilli, O. Narayan, and W. Willinger, "Experimental Queueing Analysis with Long-Range Dependent Packet Traffic," *IEEE/ACM Trans. on Networking*, **4**(2), April, 1996, pp. 209-223.

[FIOR98]P. Fiorini, Y. Ding, and L. Lipsky, "On Modeling and Characterizing Self-Similar Data Traffic", *Proc CMG '98*.

[LAU95]W-C. Lau, A. Erramilli, J. Wang, and W. Willinger, "Self-Similar Traffic Generation: The Random Midpoint Displacement Algorithm and Its Properties," *Proc. ICC '95*.

[LELA94]W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", *IEEE/ACM Trans. on Networking*, **2**(1), Feb, 1994, pp. 1-15

[JAIN91]R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley and Sons, New York, 1991.

[NORR94]I. Norros, "A Storage Model with Self-Similar Input," *Queueing Systems*, **16**(1994), pp. 387-396.

[PAXS94]V. Paxson, "Fast Approximation of Self-Similar Network Traffic," *ACM SIGCOMM Computer Communication Review*, **27**(5), October, 1997, pp. 5 - 18.

[PEIT88]H. Peitgen and D. Saupe (eds), *The Science of Fractal Images*, Springer-Verlag, New York, 1988.

[PETE94]Peters, E., *Fractal Market Analysis*, John Wiley and Sons, New York, 1994.

[SMIT94]K. Smith, "The Effects of Self-Similar Traffic on Queueing Systems", *Proc CMG '94*.

[TAQQ97]M. Taqqu, W. Willinger, and R. Sherman, "Proof of a Fundamental Result in Self-Similar Traffic Modeling," *ACM SIGCOMM Computer Communication Review*, **27**(2), April, 1997, pp. 5 - 23.