Available online at www.sciencedirect.com

ScienceDirect

Simulation Modelling Practice and Theory 15 (2007) 328–353

SIMULATION MODELLING PRACTICE AND THEORY

www.elsevier.com/locate/simpat

Suggestions of efficient self-similar generators

Hae-Duck J. Jeong [a,*], Jong-Suk R. Lee [b], Don McNickle [c], Krzysztof Pawlikowski [c]

[a] School of Information Science, Korean Bible University, Seoul, South Korea
[b] Grid Technology Research Department, Supercomputing Centre Korea Institute of Science and Technology Information, Daejeon, South Korea
[c] University of Canterbury, Christchurch, New Zealand

Received 8 January 2006; received in revised form 9 October 2006; accepted 17 October 2006
Available online 28 November 2006

Abstract

The growth of Grid computing and the Internet has been exponential in recent years. These high-speed communication networks have had a tremendous impact on our civilisation. High-speed communication networks offer a wide range of applications, such as multimedia and data intensive applications, which differ significantly in their traffic characteristics and performance requirements. Many analytical studies have shown that self-similar network traffic can have a detrimental impact on network performance, including amplified queueing delays and packet loss rates in broadband wide area networks. Thus, full understanding of the self-similar nature in teletraffic engineering is an important issue.

This paper presents a detailed survey of self-similar generators proposed for generating sequential and fixed-length self-similar pseudo-random sequences for simulation in communication networks. We evaluate and compare the operational properties of the fixed-length and sequential generators of self-similar pseudo-random sequences. The statistical accuracy and time required to produce long sequences are discussed theoretically and studied experimentally. The evaluation of the generators concentrated on two aspects: (i) how accurately self-similar processes can be generated (assuming a given mean, variance and self-similarity parameter H), and (ii) how quickly the generators can generate long self-similar sequences. Overall, our results have revealed that the fastest and most accurate generators of the six sequential and five fixed-length sequence generators considered are the SRP-FGN, FFT and FGN-DW methods.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Self-similar generator; Self-similar process; Hurst parameter; Sequential simulation; Complexity; Communication network

1. Introduction

Many studies of traffic modelling and analysis have shown that *self-similar* (or *fractal*) processes may provide better models for teletraffic in modern communication networks than Poisson processes for the

[*] Corresponding author. Address: 205 Sanggye 7-dong, Nowon-gu, Seoul 139-791, South Korea. Tel.: +82 2 950 5495; fax: +82 2 950 5411.
E-mail address: joshua@bible.ac.kr (Hae-Duck J. Jeong).

1569-190X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2006.10.002

past decade [1–3]. If this is not taken into account, it can lead to inaccurate conclusions about performance of telecommunication networks. Thus, an important requirement for conducting simulation studies of communication networks is the ability to accurately generate long synthetic stochastic self-similar sequences.

A various methods for generating pseudo-random self-similar sequences have been proposed [4,5]. Generators of synthetic self-similar sequences can be divided into two practical classes: sequential generators and fixed-length sequence generators as follows.

1.1. An overview of sequential generators

It is possible to construct a sequential Markovian model that mimics a self-similar sequence [6,7]. However, a disadvantage of this method is that the connection between the model's parameters and its self-similar properties is difficult to understand. Markovian models for self-similar traffic are forced to include several control parameters with a wide range of input values. It is also more complicated to control these values and more sensitive to the accuracy in sequential generators than in generators of fixed-length sequences of self-similar processes with a given Hurst parameter. For example, a method based on superposition of two state Markovian processes was proposed by Andersen and Nielsen [8]. They showed that it can be used to imitate a self-similar process with a certain Hurst parameter, over three to five time scales. However, five control parameters are needed, and the resulted self-similarity can gradually disappear as the time scale increases. Thus, in this respect, their method is unable to adequately model the self-similar counting processes.

Lowen and Teich [9], and Ryu and Lowen [10,11] proposed four generators (i.e., fractal-binomial-noise-driven Poisson process [FBNDP], fractal renewal process [FRP], superposition of fractal renewal processes [SFRP], and fractal-short-noise-driven Poisson process [FSNDP]). We considered only SFRP and FBNDP because they are more flexible, more accurate and faster at generating self-similar processes than FRP and FSNDP, see [11,12].

A sequential generator based on the *renewal reward processes*, has been proposed by Mandelbrot [13] and Taqqu and Levy [14,3]. However, we excluded that generator because it requires O(nM) computations to generate n numbers, where M is an aggregation level. Further, the generator behaves as a fractional Brownian motion when $n \ll M$, for $M \geqslant 16{,}000$; see [14 and 3], for detailed discussions. We considered and investigated the following efficient candidate sequential generators:

- A generator based on the *fractal-binomial-noise-driven Poisson processes* (FBNDP), proposed by Lowen and Teich [9], and Ryu and Lowen [10,11];
- A generator based on the *superposition of fractal renewal processes* (SFRP), proposed by Lowen and Teich [9], and Ryu and Lowen [10,11];
- A generator based on the *$M/G/\infty$* processes (MGIP), proposed by Cox and Isham [15], and Cox [16];
- A generator based on the *Pareto-modulated Poisson processes* (PMPP), proposed by Le-Ngoc and Subramanian [17];
- A generator based on the *spatial renewal processes and fractional Gaussian noise* (SRP-FGN), proposed by Taralp et al. [18]; and
- A generator based on the *superposition of autoregressive processes* (SAP), proposed by Granger [19].

1.2. An overview of fixed-length sequence generators

The most frequently studied models of self-similar traffic in the discrete-time case belong to the class of *fractional autoregressive integrated moving-average* (F-ARIMA) processes and the class of *fractional Gaussian noise* (FGN) processes because they require the Hurst parameter and variance; see [2,5,20]. F-ARIMA (p, d, q) processes were introduced by Hosking [21], where p is the order of autoregression in the ARIMA process, d is the degree of differencing, and q is the order of the moving average. Hosking showed that the F-ARIMA processes are asymptotically self-similar with the Hurst parameter $H = d + \frac{1}{2}$, as long as $0 < d < \frac{1}{2}$.

To describe the FGN process, we first introduce the fractional Brownian motion (FBM) process $\mathbf{B}_H(t)$, $t \geqslant 0$, which has a Hurst parameter H, $0 < H < 1$. The FBM process is a Gaussian process with zero-mean, stationary increments and the autocovariance function

$$\mathrm{Cov}(B_H(t_1), B_H(t_2)) = \frac{1}{2}[t_1^{2H} + t_2^{2H} - (t_1 - t_2)^{2H}]\mathrm{Var}[B_H(1)],$$

where t_1 and t_2 are time. This is statistically self-similar in the sense that $\mathbf{B}_H(at)$, $t \geqslant 0$, has the same finite dimensional distributions as $a^H\mathbf{B}_H(t)$, $t \geqslant 0$, for all $a > 0$. The FGN process $\mathbf{Y}$ is the incremental process of the FBM process. It is defined by $Y_i = B_H(i+1) - B_H(i), i \geqslant 1$, and its properties of stationarity, zero mean and variance $E[Y_i^2] = E[B_H^2(1)] = \sigma_0^2$ are derived from the FBM process. The ACF of the FGN process is given by

$$\rho_k = \frac{1}{2}[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}]. \tag{1}$$

A different approach to generating synthetic self-similar sequences for packet traffic was proposed by Erramilli et al. [22–24], based on deterministic chaotic maps [25]. Chaos is present in a dynamic system if simple, low order, nonlinear deterministic equations can produce behaviour that mimics random processes. In particular, Erramilli and Singh have shown that a simple, two parameter nonlinear chaotic map, referred to as an intermittent map, can capture many of the fractal properties in actual packet traffic measurements. Clearly, the generation of synthetic traffic via nonlinear chaotic maps makes the dynamic system's approach to packet traffic modelling particularly appealing. After an appropriate chaotic map has been derived from a set of traffic measurements, generating a packet stream for an individual source is generally quick and easy. Deriving an appropriate nonlinear chaotic map based on a set of actual traffic measurements, however, currently requires considerable guessing and experimenting. We considered the following fully synthesised fixed-length sequence generators:

- A generator based on the *fractional-autoregressive integrated moving average* (F-ARIMA) process, proposed by Hosking [21];
- A generator based on the *fast Fourier transform* (FFT) algorithm, proposed by Paxson [20];
- A generator based on *fractional Gaussian noise and Daubechies wavelets* (FGN-DW), proposed by Jeong et al. [26,27];
- A generator based on the *random midpoint displacement* (RMD) algorithm and implemented by Lau et al. [28]; and
- A generator based on the *successive random addition* (SRA) algorithm, proposed by Saupe [29], in the version implemented by Jeong et al. [30].

Our comparative evaluation of self-similar pseudo-random teletraffic generators concentrates on two aspects: (i) how accurately self-similar processes can be generated, and (ii) how quickly the methods generate long self-similar sequences.

We describe six sequential generators, based on FBNDP, SFRP, MGIP, PMPP, SRP-FGN and SAP in Section 2, and five fixed-length generators of self-similar sequences, based on the F-ARIMA, FFT, FGN-DW, RMD and SRA methods, in Section 4. Then, in Section 3 and 5 we concentrate on the least biased estimators, the wavelet-based H estimator and Whittle's MLE, as discussed in [31], when presenting the numerical results of a comparative analysis of the generated sequences. In Section 6, the fastest and most accurate sequential generator is compared with the most accurate fixed-length sequence generator; finally, conclusions are presented.

2. Sequential generators

2.1. Method based on fractal-binomial-noise-driven poisson process

For the standard fractal renewal process (FRP), inter-event times are independent random variables. The marginal probability density function (PDF) of such a fractal renewal process assumes the form

$$f(t) = \begin{cases} 0, & t \leqslant A, \\ \delta A^{\delta} t^{-(\delta+1)}, & t > A, \end{cases} \tag{2}$$

where $0 < \delta < 2$ [12].

However, the resulting IDC(t) (see [31]) has a dip near $t = t_0$, caused by the abrupt cutoff in the inter-event time PDF. The time instant t_0, which marks the lower limit for significant scaling behaviour in the IDC(t) and ACF, is also known as the fractal onset time. Furthermore, the power spectral density exhibits excessive oscillations for the same reason.

Selecting δ in the range $1 < \delta < 2$ proves far superior to $0 < \delta < 1$ for the same required value of α, but the form of the inter-event time PDF in Eq. (2) can be further improved. The improved PDF of the FRP decays as a power law given by

$$f(t) = \begin{cases} \delta A^{-1} \mathrm{e}^{-\delta t/A}, & 0 < t \leqslant A, \\ \delta \mathrm{e}^{-\delta} A t^{-(\delta+1)}, & t > A, \end{cases} \tag{3}$$

which is continuous for all t and it produces smoother spectral density function than Eq. (2).

The FRP is recast as a process with real-values that alternates between two values, zero and R, $R > 0$ [11]. This alternating FRP starts at a value of zero ("OFF"), and then switches to a value of R ("ON") at a time corresponding to the first event in the FRP. At the second such event, the alternating FRP switches back to zero, and proceeds to switch back and forth at every successive event in the FRP. Thus, all ON/OFF periods are IID with the same heavy-tailed distribution as in the FRP.

A method based on the fractal-binomial-noise-driven Poisson process (FBNDP) adds M IID alternating FRPs to generate a fractal binomial noise process that serves as the rate function for a Poisson process. The FBNDP requires five input parameters to generate self-similar sequences: A, δ, R, Δt and M. The resulted Hurst parameter H assumes the value $(\alpha + 1)/2$. The algorithm advances by the intervals Δt.

If S is a simulation clock, which advances in time and $S^{(j)}$ is the elapsed time of the jth FRP sequence, then $S^{(j)} = \tau_0^{(j)} + \tau_1^{(j)} + \cdots + \tau_k^{(j)}$ for some k and $j = 1, 2, \ldots, M$, where $\tau_k^{(j)}$ is the inter-arrival time. The sequence of self-similar pseudo-random numbers X_0, X_1, ... is generated by the following steps:

Step 1. For each $j = 1, 2, \ldots, M$, generate $\tau_0^{(j)}$ from

$$\tau_0^{(j)} = \begin{cases} -\delta^{-1} A \log[U(\delta^{(V-1)})(\delta^{(V-U)})^{-1}], & V \geqslant 1, \\ A V^{1/(1-\delta)}, & V < 1, \end{cases} \tag{4}$$

where

$$V \equiv \frac{1 + (\delta - 1)\mathrm{e}^{\delta}}{\delta} U, \tag{5}$$

and U is an IID uniformly distributed random variable over the unit interval $[0, 1)$; set $S^{(j)} = \tau_0^{(j)}$.

Step 2. Find j^* and $S^{(j^*)}$ such that $j^* = \operatorname{argmin}_j\{S^{(j)}\}$.

Step 3. Calculate

$$x = \begin{cases} 0, & \text{if } S^{(j^*)} < A, \\ 1, & \text{if } S^{(j^*)} \geqslant A. \end{cases} \tag{6}$$

Step 4. If $x = 1$, then X_0 should be drawn from a Poisson probability distribution with $\lambda = 1$. If $x = 0$, then $X_0 = 0$.

Step 5. Set $i = 1$, and $y = 0$. Advance the simulation clock, i.e., $S \leftarrow S^{(j^*)}$.

Step 6. Construct a new inter-event time $\tau_i^{(j)}$ from

$$\tau_i^{(j)} = \begin{cases} -\delta^{-1} A \log[U], & U \geqslant \mathrm{e}^{-\delta}, \\ \mathrm{e}^{-1} A U^{-1/\delta}, & U < \mathrm{e}^{-\delta}, \end{cases} \tag{7}$$

and set $S^{(j^*)} \leftarrow S^{(j^*)} + \tau_i^{(j)}$.

Step 7. Find a new j^* such that $j^* = \text{argmin}_j\{S^{(j)}\}$, and compute $S^{(j^*)} - S$.
Step 8. Repeat Step 6 through Step 8 to obtain x as in Step 3.
Step 9. Advance the simulation clock, i.e., $S \leftarrow S^{(j^*)}$, and set $y = y + x$.
Step 10. Repeat Step 6 through Step 10 within time slot of length Δt.
Step 11. Compute $X_i = \text{POISS}(y)$, set $y = 0$, and $i = i + 1$.
Step 12. Repeat Step 6 through Step 11 until $i = n$, where n is the number of sample points.

An approximate self-similar sequence $\{X_0, X_1, X_2, \ldots\}$ is obtained from these steps. We assume that four input values of the FBNDP method are $A = 9.92$, $R = 200$, $M = 4$ to 14, and $H = 0.6$ to 0.9. Our results show that the appropriate aggregate level M is between 4 and 10. These results show that no aggregation level in this range of M is consistently better than others. Without studying whether the marginal probability distributions of these mixtures of Poisson processes are close enough to normal distributions, we chose the aggregate level of $M = 10$, when comparing this generator with others in Section 3. Thus, the problem of selection of M for securing normality of marginal distributions of the output processes from such a generator remains an open problem.

Note that, for small input values of λ of Poisson processes, only the Poisson approximation can be used, but for large input values of λ we can use either the normal or the Poisson approximation. This implies that for large values of λ it must be possible to approximate the Poisson distribution by the normal distribution; see [32, p.190] for details.

Generation of a sample sequence of 1,048,576 numbers (so, about 524×10^6 inter-event times) took 9 min 38 s on a Pentium II (233 MHz, 512 MB). The FBNDP method requires O(n) computations to generate n numbers. For a more detailed discussion, see [12].

2.2. Method based on superposition of fractal renewal processes

The fractal renewal process (FRP) was described in Section 2.1. This self-similar process results from the superposition of a number of independent and identical FRPs. We now consider a method based on the *superposition of fractal renewal processes* (SFRP), proposed by Lowen and Teich [9] and Ryu and Lowen [10,11]. This method is defined as the superposition of M independent and identical FRPs. The method is characterised by M and the common inter-event PDF in Eq. (3). This method requires three parameters, i.e., α and A from the individual FRPs, and M, the number of FRPs superposed. The resulted Hurst parameter H, and mean μ and variance σ^2 of the marginal output distribution of a related count process in the unit time interval, are given by $H = (\alpha + 1)/2$, $\mu = E[X_n] = \lambda$, $\sigma^2 = \text{Var}[X_n] = (1 + (1/t_0))^\alpha \lambda$, where

$$\lambda = M\delta[1 + (\delta - 1)^{-1}e^{-\delta}]^{-1}A^{-1}$$

is the aggregated arrival rate of events in the unit time interval, and

$$t_0 = (2^{-1}\delta^{-2}(\delta - 1)^{-1}(2 - \delta)(3 - \delta)e^{-\delta}[1 + (\delta - 1)e^{\delta}]^2 A^\alpha)^{1/\alpha},$$

that is the value of time at which the resulting IDC(t) has a dip.

If S is a simulation clock, which advances in time and $S^{(j)}$ is the elapsed time of the jth FRP sequence, then $S^{(j)} = \tau_0^{(j)} + \tau_1^{(j)} + \cdots + \tau_k^{(j)}$ for some k and $j = 1, 2, \ldots, M$. The inter-event times X_i are generated by the following steps:

Step 1. For each $j = 1, 2, \ldots, M$, and $i = 0$, generate $\tau_0^{(j)}$ from Eqs. (4) and (5) in the FBNDP; set $S^{(j)} = \tau_0^{(j)}$.
Step 2. Find j^* such that $j^* = \text{argmin}_j\ \{S^{(j)}\}$, and set $X_0 = S^{(j^*)}$.
Step 3. Advance the simulation clock, i.e., $S \leftarrow S^{(j^*)}$.
Step 4. Set $i = i + 1$. Construct a new inter-event time $\tau_i^{(j)}$ from Eq. (7) in the FBNDP and set $S^{(j^*)} \leftarrow S^{(j^*)} + \tau_i^{(j)}$.
Step 5. Find a new j^* such that $j^* = \text{argmin}_j\{S^{(j)}\}$, and compute $X_i = S^{(j^*)} - S$.
Step 6. Advance the simulation clock, i.e., $S \leftarrow S^{(j^*)}$.
Step 7. Repeat Step 4 through Step 6 until a given $i = n$ is reached, where n is the number of sample points.

Using the previous steps, this method generates an approximate self-similar sequence $\{X_1, X_2, \ldots\}$. This method produces the most accurate result when the aggregation level M is between 4 and 10. It took 22 min, 44 s to generate a sequence of 1,048,576 numbers (so, about 1362×10^6 inter-event times) on a Pentium II (233 MHz, 512 MB). The results were obtained assuming $M = 10$ and $A = 3.8$. The SFRP method requires $O(n)$ computations to generate n numbers. For a more detailed discussion, see [12].

2.3. Method based on $M/G/\infty$ processes

An $M/G/\infty$ is a queueing system in which a server is available immediately for every arriving customer, regardless of how many customers are already being served. Applying this process to generate LRD count sequences, we assume that new arrivals can enter $M/G/\infty$ only at the beginning of time slot of length Δt. Let us call this method as MGIP. The method is based on simulation of customers that arrive at an infinite-server queueing system according to a Poisson process with an arrival rate λ. This method generates asymptotically self-similar sequences obtained from counting the number of customers from unlimited servers in the system, where the service time distribution G satisfies the heavy-tailed condition [16,22,33]. Cox [16] showed that an infinite variance service time distribution results in an asymptotically self-similar process. Likhanov et al. [34] proposed a model for aggregate packet streams, based on combining sequences generated by several ON/OFF sources with a Pareto-distributed ON period. They showed that increasing the number of sources yields a limiting behaviour identical to the $M/G/\infty$ input sequence with a Pareto distribution. To implement their findings we need to assume a given coefficient utilisation of the queueing system ρ, $0 < \rho < 1$, and a Pareto distribution of service times with finite mean service times and infinite variance of service times, i.e., with the shape parameter α, $1 < \alpha < 2$. The simulation will be advanced each time by Δt seconds. Then, the MGIP method consists of the following steps:

Step 1. Given ρ, α, Δt, $i = 1$.

Step 2. Simulate performance of an $M/G/\infty$ queueing system over Δt seconds. This means, generate pseudo-random numbers representing the number of Poisson arrivals to the $M/G/\infty$ queueing system within Δt seconds, and pseudo-random numbers from the Pareto distribution representing service times of these customers. Assume arrival rate $\lambda = \rho(\alpha - 1)/\alpha$, where ρ is traffic intensity and α is the shape parameter of the Pareto distribution, and service rate $(\alpha - 1)/\alpha$.

Step 3. Count the number of customers in the simulated $M/G/\infty$ queueing system at the end of this time slot of length Δt. This is X_i, the ith number of the output LRD self-similar sequence in the scale Δt. LRD sequences in larger time scales, say s, can be obtained by counting number of customers in the system at the end of each s time slots, i.e., by assuming a time lag equal to Δt.

Step 4. Set $i = i + 1$. Repeat Step 2 to Step 4, advancing the simulated time by the next Δt seconds, until $i \leqslant n$, where n is the number of sample points. Otherwise, stop.

A self-similar sequence $\{X_1, X_2, \ldots\}$ is obtained from these steps. We assume the MGIP method with input traffic intensity $\rho = 0.9$ and service rate $\mu = (\alpha - 1)/\alpha$, where $\alpha = 3 - 2H$, for $H = 0.6$, 0.7, 0.8 and 0.9, and time lag $s = 4$ to 14. Our results show that this method is most efficient at time lag s between 4 and 8. Generation of an asymptotic self-similar sequence with 1,048,576 numbers with these time lags took 27 s on a Pentium II (233 MHz, 512 MB). $O(n)$ computations are required to generate a self-similar sequence.

2.4. Method based on Pareto-modulated poisson processes

This method is based on the fact that a Pareto-modulated Poisson process (PMPP), based on a switched Poisson process with two states, with sojourn times governed by an independent and identical Pareto distribution, asymptotically generates a self-similar sequence [17]. Fig. 1 shows a state diagram of the PMPP. The two states of the switched Poisson process can be viewed as intervals with the long and short burst rates of events. This process goes through consecutive cycles of being in State 1 and State 2. The time spent in each cycle is governed by a Pareto distribution characterised by α, $1 < \alpha < 2$. These cycles have the mean length (ML) equal

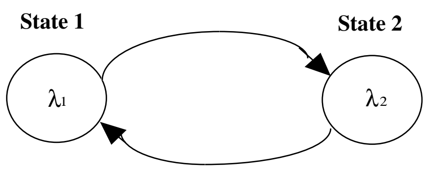

Fig. 1. State diagram of the Pareto-modulated Poisson process [17]. It is a two-state switched Poisson process with the sojourn time in each state following an independent and identical Pareto distribution.

$$\mathrm{ML} = \frac{\alpha}{\alpha-1} + \frac{\alpha}{\alpha-1} = \frac{2\alpha}{\alpha-1} \text{ time units.} \tag{8}$$

Mean numbers of Poisson events (MNPE) generated in state S_1 and state S_2 are:

$$\begin{aligned} &\text{MNPE in state } S_1 = \lambda_1 \frac{\alpha}{\alpha-1}, \quad \text{and} \\ &\text{MNPE in state } S_2 = \lambda_2 \frac{\alpha}{\alpha-1}. \end{aligned} \tag{9}$$

Thus, mean number of Poisson events per cycle is given by

$$\lambda_1 \frac{\alpha}{\alpha-1} + \lambda_2 \frac{\alpha}{\alpha-1} = (\lambda_1+\lambda_2)\frac{\alpha}{\alpha-1}, \tag{10}$$

and using Eqs. (8) and (10), we get the mean number of Poisson events per time unit as

$$\overline{E} = \frac{(\lambda_1+\lambda_2)}{2}. \tag{11}$$

As mentioned the PMPP can be used to generate asymptotically self-similar sequences. The quality generated sequences, in the sense of the closeness to exactly self-similar processes, depends on the size of frames within which one counts numbers of Poisson events occurring in underlining PMPP, see Fig. 2. If frames have length T seconds, then the mean number of Poisson events occurring in a frame can be called the aggregation level of that method, given as

$$\overline{N} = \overline{E} * T, \tag{12}$$

where $\overline{E}$ is the number of events per second. There should exist the minimum acceptable aggregation level $\overline{N}_{\min}$ below which this method would not generate satisfactory self-similar sequences, but we leave this issue for further research. In our investigations we assume $\lambda_1 = 100$, $\lambda_2 = 120$, and $T \geqslant 100$. Thus, we assume $\overline{N}_{\min} \geqslant 11,000$. The PMPP generator follows of the following steps: For a given λ_1, λ_2, α, T and n, let S be the time advance.

Step 1. Set $i = 1$, $k = 1$, $S = 0$, $X_i^{(0)} = 0$.

Step 2. Generate the sojourn times τ_k and τ_{k+1} for state $S_{k \bmod 2}$ and $S_{(k+1) \bmod 2}$ of PMPP using the Pareto distribution with shape parameter $\alpha, 1 < \alpha < 2$.
Produce a sequence of Poisson arrivals within each state S_j with rate λ_j, $j = 1$, or 2.
Calculate $S = S + \tau_k + \tau_{k+1}$ if $S \geqslant iT$ then go to Step 3, otherwise assume $k = k + 1$ and repeat Step 2.

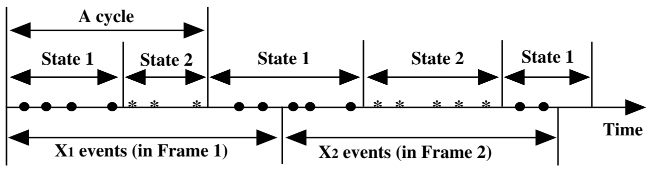

Fig. 2. Graphical explanation of the concept of data aggregation in the generator based on PMPP. Note that ● is an event in Poisson process with rate λ_1, and $*$ is an event in Poisson process with rate λ_2. X_i is the number of Poisson events occurring in Frame i.

Step 3. Count $X_i^{(1)}$, the number of events that occur in the last frame of the length of T seconds.

$$X_i = X_i^{(0)} + X_i^{(1)}. \tag{13}$$

Count the number of events that occur in the remaining time interval $(i\,T, S)$. This is $X_{i+1}^{(0)}$, the initial component of X_{i+1}. If $i < n$ then assume $i = i + 1$, go to Step 2. Otherwise if $i = n$, then stop (the required number of pseudo-random self-similar numbers has been generated).

We compare this generator with others in Section 3, assuming $T = 300$. Generating a sample sequence of 1,048,576 numbers took 7 min, 11 s on a Pentium II (233 MHz, 512 MB). The PMPP method requires O(n) computations to generate n numbers. For a more detailed discussion, see [17].

2.5. Method based on spatial renewal processes and fractional gaussian noise

The SRP-FGN generator is a hybrid method that uses a fractional Gaussian noise (FGN) generator based on the spatial renewal process (SRP) developed by Taralp et al. [18]. Before discussing SRP, we first introduce the concept of sub-exponentiality. It means that the ACF of a stationary process decays not exponentially, but hyperbolically, for large lags [35]. For example, Jelenković [36] observed distinctive sub-exponential characteristics of MPEG video traffic in the functional behaviour of its scene length distributions.

The SRP belongs to a class of sub-exponentially time-dependent stochastic processes. The SRP **Z** is composed of a chain of mutually independent renewal periods. For practical reasons, it is assumed that the SRP is a discrete time process and its ith period T_i has length k_i, where k_i is an integer, and the sample of **Z** during the period is represented by a sequence of k_i numbers $Y_1, Y_2, \ldots, Y_{k_i}$, governed by the normal distribution. The consecutive number of the output self-similar time series $X_i \sum_{j=1}^{k_i} Y_j$.

To improve statistical properties of the output sequence (it fit to normal distribution and the required correlation function), it has been proposed to aggregate a number of such sequences [18]. We investigated this suggestion experimentally by considering various levels of aggregation. Our results also show that the SRP-FGN method is most accurate if the level of aggregation $M = 10$, supporting the advice of Taralp et al. [18], who wrote that the aggregation level needs not be high (i.e., ≈10) to obtain accurate results.

The aggregate output sequence $\{X_1, X_2, \ldots\}$ is computed by after having summed the sequences and normalisation of the sample variance to one. The SRP-FGN model has a normal marginal distribution, and is characterised by a sub-exponential ACF. Fig. 3 shows a block diagram of the SRP-FGN generator, which consists of the following steps:

Step 1. Given H, n, M, $i = 0$, $m = 0$, $s = 0$, $l = 1$, $X_0 = 0$.

Step 2. Generate a random length k_i of the renewal cycle T_i governed by the following a cumulative probability distribution function $F_T(k)$.

$$F_T(k) = \begin{cases} 0, & 0 \leqslant k < 1, \\ 1 - \frac{H\{(k+1)^{2H-1} - 2k^{2H-1} + (k-1)^{2H-1}\}}{(2^{2H-1} - 2)}, & 1 \leqslant k. \end{cases} \tag{14}$$

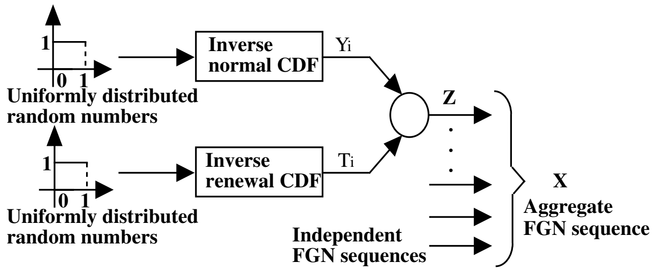
1
0 1
Uniformly distributed random numbers
Inverse normal CDF
Yi
Z
1
0 1
Uniformly distributed random numbers
Inverse renewal CDF
Ti
X
Aggregate FGN sequence
Independent FGN sequences

Fig. 3. Block diagram of the SRP-FGN method [18].

Step 3. The SRP **Z** is composed of a chain of renewal periods where the ith period T_i is k_i in length. Generate k_i random numbers $Y_1, Y_2, \ldots, Y_{k_i}$ governed by the normal distribution. Set $m = m + k_i$. If $m < M$, then $X_i = X_i + \sum_{j=1}^{k_i} Y_j$, set $s = s + m$, and go to Step 2. If $m \geqslant M$, then set $m = m - M$, and go to Step 4.

Step 4. The output value X_i is computed as follows: If $s = 0$, then $X_i = \sum_{j=lM-M+1}^{lM} Y_j$, and set $s = 0$. If $s > 0$, then $X_i = X_i + \sum_{j=1}^{M-s} Y_j$, and set $s = 0$.

Step 5. Set $i = i + 1$, and $X_i = 0$. If $i < n$ and if $m = 0$, then go to Step 2. Otherwise,– if $m < M$, then $X_i = \sum_{j=k_i-m+1}^{k_i} Y_j$, set $s = m$, $l = 1$, and go to Step 2.– if $m \geqslant M$, then set $m = m - M$, $s = 0$, $l = l + 1$, and go to Step 4. If $i = n$, where n is the number of sample points, then stop.

This generator produces approximately self-similar sequences $\{X_0, X_1, X_2, \ldots\}$. We obtained the points of the inverse renewal CDF using Eq. (14). In order to obtain more accurate results of the tail behaviour, we chose a number of intervals, $T = 10{,}000$, for the renewal CDF. The SRP-FGN renewal CDF $F_T(i)$ and complementary CDF are plotted in Figs. 4 and 5. $F_T(i)$ gradually has longer tails as the H value increases. The SRP-FGN method generates sample sequences $\{X_1, X_2, \ldots, X_n\}$ with O(n). It took 26 s to generate a sequence of 1,048,576 numbers on a Pentium II (233 MHz, 512 MB).

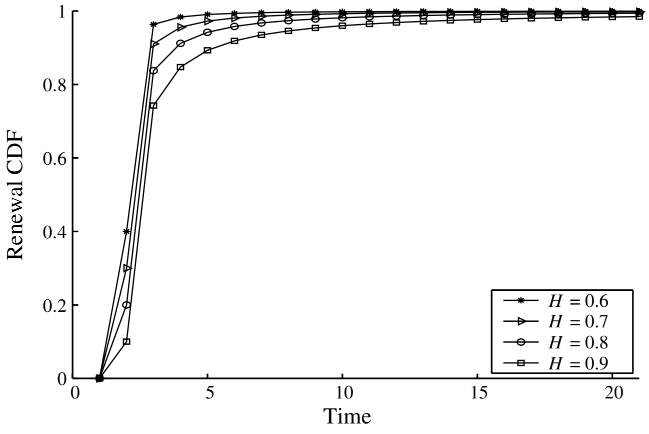

Fig. 4. Cumulative distribution function $F_T(i)$ of the SRP-FGN method for $H = 0.6$, 0.7, 0.8 and 0.9.

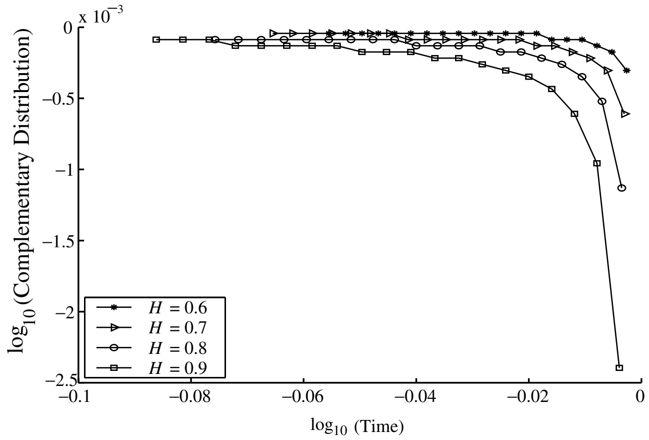

Fig. 5. Complementary cumulative distribution function of the SRP-FGN method for $H = 0.6$, 0.7, 0.8 and 0.9.

2.6. Method based on superposition of autoregressive processes

The method based on the superposition of autoregressive processes (SAP) proposed by Granger [19] generates asymptotically self-similar sequences when aggregating several independent autoregressive processes. In the simplest case this can be the sum of two autoregressive processes of the first order:

$$\begin{aligned} z_{1i} &= A_{1i}z_{1,i-1} + y_{1i}, \\ z_{2i} &= A_{2i}z_{2,i-1} + y_{2i}, \quad i = 1, 2, \ldots \end{aligned} \tag{15}$$

where A_{1i} and A_{2i} are randomly chosen from a beta distribution $B(\alpha_1, \alpha_2)$ on $[0, 1]$ with shape parameters α_1 and α_2, where $\alpha_1 > 0$, $\alpha_2 > 0$. y_{1i} and y_{2i} are a pair of IID sequences of random variables with a mean of zero and variance $\sigma^2 = 1$.

As shown in [19], using the least-square fitting it can be found that $\alpha_2 = 7.7929 * \log(H) + 4.9513$. Thus, the Hurst parameter H is linearly dependent on the shape parameter α_2 of the beta-distribution, while α_1 can be selected arbitrary, for example, $\alpha_1 = 1$ in all cases that we investigated.

The PDF $f(x)$ of the beta distribution is given by

$$f(x) = \begin{cases} \frac{x^{\alpha_1-1}(1-x)^{\alpha_2-1}}{\beta(\alpha_1,\alpha_2)}, & 0 < x < 1, \\ 0, & \text{otherwise}, \end{cases} \tag{16}$$

where $\beta(\alpha_1, \alpha_2)$ is defined by

$$\beta(\alpha_1, \alpha_2) = \int_0^1 x^{\alpha_1-1}(1-x)^{\alpha_2-1}\,\mathrm{d}x = \frac{\Gamma(\alpha_1)\Gamma(\alpha_2)}{\Gamma(\alpha_1 + \alpha_2)}.$$

This method, as based on the superposition of the autoregressive processes, consists of the following steps: Given: α_1, α_2, $i = 0$.

Step 1. Set $i = i + 1$. Determine z_{1i} and z_{2i} using Eq. (15).
Step 2. Calculate the sum,

$$X_i = z_{1i} + z_{2i}, \quad i = 1, 2, \ldots.$$

Step 3. Repeat Step 1 and Step 2 until $i = n$, where n is the number of sample points.

Using the previous steps, the method based on the superposition of the autoregressive process generates an asymptotically self-similar sequence $\{X_1, X_2, \ldots\}$. The CPU time required to generate 1,048,576 numbers was 35 s on a Pentium II (233 MHz, 512 MB). Unlike the other sequential generators, the SAP generator does not require an aggregation level to be assumed as an input parameter, but such input parameters as the shape parameter α_1, instead.

3. Comparison of sequential generators

All six sequential generators based on FBNDP, SFRP, MGIP, PMPP, SRP-FGN and SAP, generate approximately self-similar sequences. We investigate their properties in greater detail in this section. All have been implemented in C on a Pentium II (233 MHz, 512 MB) computer. The mean times required for generating sequences of a given length were obtained using the SunOS 5.7 time command and were averaged over 30 replications, each with sequences of 32,768 (2^{15}), 65,536 (2^{16}), 131,072 (2^{17}), 262,144 (2^{18}), 524,288 (2^{19}) and 1,048,576 (2^{20}) numbers.

We have analysed the accuracy of the six methods. For each of $H = 0.5$, 0.6, 0.7, 0.8 and 0.9, sample sequences generated were analysed as follows. The FBNDP process was analysed with input $M = 10$, $R = 200$, and cutoff parameter $A = 9.92$; the SFRP method required the following three parameters: $M = 10$, $H = 0.6$, 0.7, 0.8 and 0.9, and cutoff parameter $A = 3.8$; the $M/G/\infty$ process (MGIP) with input traffic intensity $\rho = 0.9$, service rate $\mu = \alpha/(\alpha - 1)$, where $\alpha = 3 - 2H$, and time lag $s = 8$; the Pareto-modulated Poisson process (PMPP) with input $T = 300$, $\lambda_1 = 100$ and $\lambda_2 = 120$; the SRP-FGN methods with input

$M = 10$, and the interval number of the renewal CDF (T) = 10,000. The input of the superposition of the autoregressive process (SAP) with a beta-distribution ($B(\alpha_1, \alpha_2)$) on [0, 1] was $B(1, 2.9)$, $B(1, 8.1)$, $B(1, 21.3)$, and $B(1, 71.5)$.

3.1. Accuracy of generated sequences

A "good" estimator is not only one which produces an estimate whose expected value is close to the true parameter (low bias), but also one which has small variance. A comparative analysis of the most frequently used H estimation techniques, the wavelet-based H, Whittle's MLE, periodogram, R/S-statistic, variance-time and IDC(t) estimators, has been done [31]. The results have shown that the wavelet-based H estimator and Whittle's MLE are the least biased of the H estimation techniques. Thus, the least biased estimators were considered when presenting the numerical results of a comparative analysis of the generated sequences. For each of $H = 0.6$, 0.7, 0.8 and 0.9, and for each of $\alpha_2 = 2.9$, 8.1, 21.3 and 71.5, all results are averaged over 30 sequences. The relative inaccuracy ΔH was calculated using

$$\Delta H = \frac{\widehat{H} - H}{H} * 100\%, \qquad (17)$$

where H is the exact value assumed and $\widehat{H}$ is its empirical mean value.

(a) Table 1 shows the results of the six sequential methods using the wavelet-based H estimator with the corresponding 95% confidence intervals $\widehat{H} \pm 1.96\hat{\sigma}_{\widehat{H}}$. For all input H and α_2 values, the SRP-FGN method produced sequences with the least biased H values compared with the other six methods.

For $H = 0.6$, 0.7 and 0.8, the absolute relative error for the FBNDP method was less than 5%, but for $H = 0.9$, it was greater than 5% (i.e., −5.5%). For $H = 0.6$, the estimated H value for the method was positively biased, but for $H = 0.7$, 0.8 and 0.9, were gradually more negatively biased as the H value increased.

For $H = 0.6$, 0.7, 0.8 and 0.9, each relative error for the SFRP method was +2.76%, +1.29%, −0.17% and −3.49%, respectively. As in the FBNDP method, estimated H values for the method ranged from positively biased to negatively biased as the H value increased.

A shortcoming of the MGIP method was that it generated approximately self-similar sequences with strongly biased H values. For $H = 0.6$, 0.7, 0.8 and 0.9, each relative error was −9.25%, −10.84%, +2.42% and +23.5%, respectively. Although these inter-event processes can be used to produce synthetic teletraffic with bursts appearing over a wider range of time scales than Poisson processes, the associated arrival processes do not appear to be self-similar when the aggregation level is low [37,38].

For $H = 0.6$, 0.7, 0.8 and 0.9, the values of the Hurst parameter from the sample sequences of the PMPP method were lower than the desired values. Each relative error was −1.27%, −2.22%, −1.23% and −3.77%, respectively. Furthermore, this method required four control parameters (i.e., two Poisson arrival rates λ_1

Table 1
Mean values of estimated H using the wavelet-based H estimator for the six sequential generators for $H = 0.6$, 0.7, 0.8 and 0.9. We give 95% confidence intervals for the means in parentheses

Methods	Mean values of estimated H and ΔH							
	0.6		0.7		0.8		0.9	
	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)
FBNDP	.6086 (.581, .636)	+1.440	.6875 (.660, .715)	−1.789	.7827 (.755, .810)	−2.157	.8502 (.823, .878)	−5.538
SFRP	.6166 (.589, .644)	+2.759	.7091 (.682, .737)	+1.296	.7986 (.771, .826)	−0.174	.8686 (.841, .896)	−3.491
MGIP	.5445 (.517, .572)	−9.252	.6241 (.597, .652)	−10.84	.8194 (.792, .847)	+2.424	1.1120 (1.084, 1.139)	+23.50
PMPP	.5924 (.565, .620)	−1.266	.6845 (.657, .712)	−2.221	.7902 (.763, .818)	−1.229	.8661 (.839, .894)	−3.766
SRP-FGN	.5942 (.567, .622)	−0.968	.6960 (.669, .724)	−0.569	.8056 (.778, .833)	+0.700	.9031 (.876, .931)	+0.344
SAP	.5989 (.595, .603)	−0.182	.6852 (.680, .690)	−2.112	.7845 (.781, .788)	−1.937	.8971 (.894, .900)	−0.325

and λ_2, an aggregation number M, and the shape parameter α), and generated a self-similar sequence that was positively biased to negatively biased as the shape parameter α approached one.

For $H = 0.6$, 0.7, 0.8 and 0.9, the values of the Hurst parameter from the sample sequences of the SRP-FGN method match the required values well. For $H = 0.6$, 0.7, 0.8 and 0.9, each relative error was −0.97%, −0.57%, +0.70% and +0.34%, respectively.

For $\alpha_2 = 2.9$, 8.1, 21.3 and 71.5, all values of the Hurst parameter from the sample sequences of the SAP method were lower than the required values. Each relative error was −0.18%, −2.11%, −1.94% and −0.33%, respectively.

(b) Table 2 shows the results of the six sequential methods using Whittle's MLE with the corresponding 95% confidence intervals $\widehat{H} \pm 1.96\hat{\sigma}_{\widehat{H}}$. As for the results obtained from the wavelet-based H estimator, the SRP-FGN method produced sequences with the least biased H values compared with the other six methods.

For $H = 0.6$ and 0.7, the absolute relative error for the FBNDP method was less than 3%, while for $H = 0.8$ and 0.9, it was greater than 5% (i.e., −5.53% and −9.29%). For $H = 0.6$, the estimated H value for the method was positively biased; and for $H = 0.7$, 0.8 and 0.9, they gradually became more negatively biased as the H value increased.

For $H = 0.6$, 0.7, 0.8 and 0.9, relative error for the SFRP method was +6.21%, +1.78%, −1.43% and −5.37%, respectively. As in the FBNDP method, estimated H values ranged from positively biased to negatively biased as the H value increased.

A shortcoming of the MGIP method was that it generated approximately self-similar sequences with biased H values for $H = 0.7$, similar to results obtained from the wavelet-based H estimator. For $H = 0.6$, 0.7, 0.8 and 0.9, relative error was −8.0%, −9.6%, −3.2% and +5.5%, respectively.

For $H = 0.8$ and 0.9, the values of the Hurst parameter from the sample sequences of the PMPP method were lower than the required values, but for $H = 0.6$ and 0.7, they were higher. Relative error for $H = 0.6$, 0.7, 0.8 and 0.9 was +7.18%, +2.24%, −1.28% and −5.59%, respectively. This method generated a self-similar sequence that ranged from negatively biased to positively biased as the shape parameter α approached one.

For $H = 0.6$, 0.7, 0.8 and 0.9, the Hurst parameter values from the sample sequences of the SRP-FGN method match the required values well. For $H = 0.6$, 0.7, 0.8 and 0.9, relative error was +0.11%, +1.91%, +2.54% and +2.53%, respectively. The results were consistently overestimated.

For $\alpha_2 = 2.9$ and 8.1, all values of the Hurst parameter from the sample sequences of the SAP method were higher than the required values. Relative error was +24.14% and +10.45%, respectively; thus, these results were overestimated. For $\alpha_2 = 21.3$ and 71.5, relative errors was −0.01% and −8.45%.

Our results show that all six sequential generators produced approximately self-similar sequences, but that relative inaccuracy ($|\Delta H|$) increased as H increased.

Table 2
Mean values of estimated H using Whittle's MLE for the six sequential generators for $H = 0.6$, 0.7, 0.8 and 0.9. We give 95% confidence intervals for the means in parentheses

Methods	Mean values of estimated H and ΔH							
	0.6		0.7		0.8		0.9	
	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)
FBNDP	.6122 (.603, .622)	+2.028	.6828 (.674, .692)	−2.452	.7557 (.747, .765)	−5.531	.8164 (.807, .826)	−9.291
SFRP	.6372 (.628, .647)	+6.205	.7124 (.703, .722)	+1.777	.7886 (.779, .798)	−1.426	.8517 (.843, .861)	−5.366
MGIP	.5520 (.542, .562)	−7.995	.6325 (.623, .642)	−9.641	.7742 (.765, .783)	−3.225	.9499 (.941, .959)	+5.549
PMPP	.6431 (.634, .652)	+7.176	.7157 (.706, .725)	+2.236	.7898 (.781, .799)	−1.281	.8497 (.841, .859)	−5.586
SRP-FGN	.6007 (.591, .610)	+0.110	.7133 (.704, .723)	+1.905	.8203 (.811, .829)	+2.539	.9227 (.914, .932)	+2.526
SAP	.7448 (.736, .754)	+24.14	.7731 (.764, .782)	+10.45	.7999 (.791, .809)	−0.009	.8239 (.815, .833)	−8.451

3.2. Complexity and speed of generation

Performance analysis of a generator can be classified into two types: time complexity and space complexity. Time complexity refers to a function describing how much time it will take the generator to execute, based on the parameters of its input. The exact value of this function is usually ignored in favour of its order, expressed in the so-called Big-O notation (this is based on the limit of the time complexity function as the values of its parameters increase without bound.) Space complexity is the way in which the amount of storage space required by the generator varies with the size of the problem it is solving. The space complexity is normally expressed as an order of magnitude, e.g. $O(n^2)$ means that if the size of the problem (n) doubles then four times as much working storage will be needed. The time complexity was only considered in this paper.

The computational complexities of all six sequential generators of pseudo-random self-similar sequences of a given length n are $O(n)$. However, the number of arithmetic operations per number required by each of them are different, as shown in Table 3. The MGIP, SAP and SRP-FGN methods are the fastest of the six sequential generators requiring 4976, 4481 and 4173 operations per number, respectively. While the SFRP method was the slowest (85,199 operations per number), the FBNDP and PMPP methods require similar amounts of time to generate the same number of self-similar sequences, requiring 11,028 and 19,507 arithmetic operations per number, respectively.

Fig. 6 shows the experimental mean running times of the six sequential generators. All require $O(n)$ computations to generate n numbers.

In summary, our results show that the generator based on the SRP-FGN algorithm is the fastest if long sequences of self-similar pseudo-random numbers are required. The MGIP and SAP methods are nearly as

Table 3
Computational complexities and arithmetic mean operations required for each of the six sequential generators (s: the time lag in the $M/G/\infty$ queueing system, M: aggregation level, T: the number of intervals in the renewal CDF)

Method	Complexity	Operations per number	Mean operations per number when assuming optimum values
FBNDP	$O(n)$	$161M + 9418$	11,028
SFRP	$O(n)$	$1378M + 71,419$	85,199
MGIP	$O(n)$	$130s + 3936$	4976
PMPP	$O(n)$	$130M + 4037$	19,507
SRP-FGN	$O(n)$	$14T + 4159$	4173
SAP	$O(n)$	4481	4481

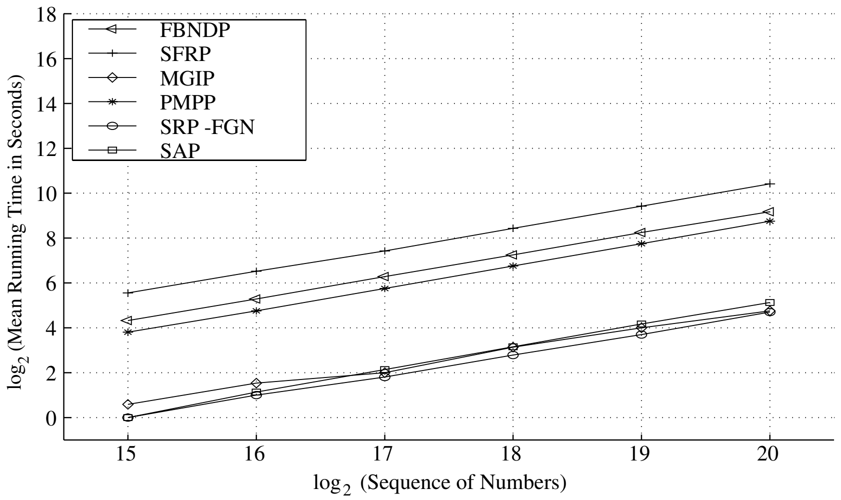

Fig. 6. Mean running times of the six sequential generators. Running times were obtained using the SunOS 5.7 `time` command on a Pentium II (233 MHz, 512 MB); each mean is averaged over 30 iterations, each with sequences of 32,768 (2^{15}), 65,536 (2^{16}), 131,072 (2^{17}), 262,144 (2^{18}), 524,288 (2^{19}) and 1,048,576 (2^{20}) numbers.

fast. The SRP-FGN generator is also the most efficient when evaluated by Whittle's MLE. Therefore, the SRP-FGN generator was compared with the most efficient fixed-length sequence generator, which we discuss in Section 6. However, as pointed out in [39] there are general pitfalls in using generators of LRD processes based on FRPs, since the generated processes fail to capture fully the required autocorrelation structure. For a more detailed discussion, see [39].

4. Fixed-Length sequence generators

The numerical results for the following fixed-length sequence generators are given in Section 5.

4.1. Fast fourier transform method

The fast Fourier transform method generates approximately self-similar sequences based on the fast Fourier transform (FFT) and the fractional Gaussian noise (FGN) process. Its main weakness is in the accuracy of the power spectrum calculation, which involves an infinite summation. Paxson [20] has proposed a solution to this problem by applying a finite approximation. Another possible method to generate self-similar sequences is to run the FFT of white noise through the power spectrum, and then apply the inverse FFT. An overview of the FFT method is given as follows. For a more detailed discussion, see [20,30,40].

Step 1. Given: H. Start for $i = 1$ and continue until $i = n/2$. Calculate a sequence of values $\{f_1, \ldots, f_{\frac{n}{2}}\}$, where $f_i = \hat{f}\left(\frac{2\pi i}{n}, H\right)$, corresponding to the power spectrum of an FGN process for frequencies from $\frac{2\pi}{n}$ to π, $1/2 < H < 1$, and the total length of the sequence generated, n. For an FGN process, the power spectrum $f(\lambda, H)$ is defined by Eqs. (18) and (19).

$$f(\lambda, H) = 2c_f(1 - cos(\lambda))\mathbf{B}(\lambda, H) \tag{18}$$

with $0 < H < 1$ and $-\pi \leqslant \lambda \leqslant \pi$, where

$$c_f = \sigma^2(2\pi)^{-1} sin(\pi H)\Gamma(2H+1),$$
$$\mathbf{B}(\lambda, H) = \sum_{k=-\infty}^{\infty} |2\pi k + \lambda|^{-2H-1}, \tag{19}$$

and $\sigma^2 = \mathrm{Var}[X_k]$ and $\Gamma(\cdot)$ is the gamma function; see [41]. As mentioned, the infinite summation in Eq. (18) for $\mathbf{B}(\lambda, H)$ poses the main difficulty in computing the power spectrum exactly. Paxson [20] proposed to use the approximation given by Eq. (20):

$$\mathbf{B}(\lambda, H) \approx a_1^d + b_1^d + a_2^d + b_2^d + a_3^d + b_3^d + \frac{a_3^{d'} + b_3^{d'} + a_4^{d'} + b_4^{d'}}{8H\pi}, \tag{20}$$

where $d = -2H - 1$, $d' = -2H$, $a_i = 2i\pi + \lambda$, $b_i = 2i\pi - \lambda$.

Step 2. Multiply the sequence of values $\{f_1, \ldots, f_{\frac{n}{2}}\}$ by an exponential random variable with a mean of one. Paxson [20] used this step since, when estimating the power spectrum of a process using the periodogram, the power spectrum estimated for a given frequency is distributed asymptotically as an exponential random variable with a mean equal to the actual power (see also Beran [42, (p. 409)]).

Step 3. Generate $\{Z_1, \ldots, Z_{\frac{n}{2}}\}$, a sequence of complex values such that $|Z_i| = \sqrt{\hat{f}_i}$ and the phase of Z_i is uniformly distributed between 0 and 2π. This random phase technique, taken from Schiff [43], preserves the spectral density corresponding to $\{\hat{f}_i\}$. It also makes the marginal distribution of the final sequence normal, as proved by Lindeberg ([44, p. 256]), and defines the requirements for FGN.

Step 4. Start for $i = 0$ and continue until $i < n$. Construct $\{Z'_0, \ldots, Z'_{n-1}\}$, an *expanded* version of $\{Z_1, \ldots, Z_{\frac{n}{2}}\}$:

$$Z'_i = \begin{cases} 0, & \text{if } i = 0, \\ Z_i, & \text{if } 0 < i \leqslant \frac{n}{2}, \quad \text{and} \\ \overline{Z_{n-i}}, & \text{if } \frac{n}{2} < i < n. \end{cases} \tag{21}$$

where $\overline{Z_{n-i}}$ denotes the complex conjugate of Z_{n-i}. $\{Z'_i\}$ retains properties of the power spectrum used to construct $\{Z_i\}$, but because $\{Z'_i\}$ is symmetric about $Z'_{\frac{n}{2}}$, it now corresponds to the FFT of a real-valued signal.

Step 5. Calculate the inverse FFT $\{Z'_i\}$ to obtain the approximate FGN sequence $\{X_i\}$ with a mean of zero and variance of one. Then form the final sequence $X_1, X_2, \ldots, X_n$ by assigning $X_i \leftarrow Z'_i$, $i = 1,2,\ldots,n$.

This method generates an approximately self-similar sequence $\{X_1, X_2, \ldots, X_n\}$ with the exact values. Generating a sample sequence of 1,048,576 numbers took 33 s on a Pentium II (233 MHz, 512 MB). The FFT method requires O($n\log n$) computations to generate n numbers because of the Fourier transform algorithm [20,45]. The Danielson-Lanczos' FFT[1] algorithm was used [46]. For a more detailed discussion, see [20].

4.2. Fractional-Autoregressive integrated moving average method

Hosking [21,47] states that the F-ARIMA method[2] is used to generate an approximately self-similar process with a Hurst parameter of $H = d + \frac{1}{2}$. We used the F-ARIMA$(0, d, 0)$ method for generating self-similar sequences, where d is the fractional differencing parameter, $0 < d < \frac{1}{2}$. Hosking's algorithm is used to generate the process $\mathbf{X} = \{X_i : i = 0, 1, 2, \ldots, n\}$ with a normal marginal distribution, a mean of zero and variance σ_0^2, and an autocorrelation function (ACF), $\{\rho_k\}(k = 0, \pm 1, \ldots)$ defined as

$$\rho_k = \gamma_k/\gamma_0 = \frac{\Gamma(1-d)\Gamma(k+d)}{\Gamma(d)\Gamma(k+1-d)}, \text{ where} \tag{22}$$
$$\gamma_k = \sigma_0^2 \frac{(-1)^k \Gamma(1-2d)}{\Gamma(k-d+1)\Gamma(1-k-d)}; \text{ see[47] and page 63 on[41].}$$

Step 0. Set $N_0 = 0$ and $D_0 = 1$. X_0, the first pseudo-random element in the output self-similar sequence, is generated from the normal distribution $N(0, \sigma_0^2)$, where σ_0^2 is the required variance of the X_i.

Step i ($i = 1, \ldots, n-1$.) Compute mean$_i$ and Var$_i$ of X_i recursively, using the following equations:$N_i = \rho_i - \sum_{j=1}^{i-1} \phi_{i-1,j}\rho_{i-j}$, $D_i = D_{i-1} - N_{i-1}^2/D_{i-1}$, $\phi_{ii} = N_i/D_i$, $\phi_{ij} = \phi_{i-1,j} - \phi_{ii}\phi_{i-1,i-j}, j = 1, \ldots, i-1$, where ϕ_{ij}, $i = 0, j = 0, \ldots, n-1$, is given by

$$\phi_{ij} = -\binom{i}{j}\frac{(j-d-1)!(i-d-j)!}{(-d-1)!(i-d)!},$$

mean$_i = \sum_{j=1}^{i} \phi_{ij}X_{i-j}$, $var_i = (1-\phi_{ii}^2)$var$_{i-1}$. Generate X_i from N(mean$_i$, var$_i$). Increase i by 1. If $i = n$, then stop.

A self-similar sequence $\{X_1, X_2, \ldots, X_n\}$ is obtained in n steps. The F-ARIMA method is too computationally intensive to generate long sample sequences. Generation of an F-ARIMA traffic sample sequence with 1,048,576 numbers took 41 h, 0 min and 22 s on a Pentium II (233 MHz, 512 MB). This method requires O(n^2) computation time.

4.3. Random midpoint displacement method

The *random midpoint displacement* (RMD) method generates an approximately self-similar sequence in the time interval $[0, T]$. The RMD algorithm is an approximate fractional Brownian motion (FBM) generation method. The basic concept of the RMD algorithm is to interpolate the interval $[0, T]$ recursively and calculate the values of the process at the midpoints from the values at the endpoints.

[1] Available at http://ita.ee.lbl.gov/.

[2] The autocorrelation functions of two other simple processes, F-ARIMA$(1, d, 0)$ and F-ARIMA$(0, d, 1)$, behave similarly at high lags, but the F-ARIMA$(0, d, 1)$ autocorrelation function drops more sharply between lags 1 and 2. For a more detailed discussion, see [47].

Fig. 7 illustrates the first three steps of the process. This method leads to the generation of the sequence $(d_{3,1}, d_{3,2}, d_{3,3}, d_{3,4})$. The interval between 0 and 1 is subdivided to construct the increments governed by a normal distribution. Adding offsets to the midpoints makes the marginal distribution of the final result normal. For more detailed discussions of the RMD method, see [28,30,48].

Step 1. If the process **Y** is to be computed for any time instance t between 0 and 1, then begin by setting $Y_0 = 0$ and selecting Y_1 as a pseudo-random number from a normal distribution with mean 0 and variance $\mathrm{Var}[Y_1] = \sigma_0^2$. Then $\mathrm{Var}[Y_1 - Y_0] = \sigma_0^2$.

Step 2. Next, $Y_{\frac{1}{2}}$ is constructed as the average of Y_0 and Y_1, that is, $Y_{\frac{1}{2}} = \frac{1}{2}(Y_0 + Y_1) + d_1$. The offset d_1 is a normal random number (NRN), which is multiplied by a scaling factor $\frac{1}{2}$, with mean 0 and variance S_1^2 of d_1. Compare the visualisation of this step and the next one in Fig. 7. For $\mathrm{Var}[Y_{t_2} - Y_{t_1}] = |t_2 - t_1|^{2H}\sigma_0^2$ to be true for $0 \leqslant t_1 \leqslant t_2 \leqslant 1$, then

$$\mathrm{Var}\left[Y_{\frac{1}{2}} - Y_0\right] = \frac{1}{4}\mathrm{Var}[Y_1 - Y_0] + S_1^2, \quad \left(\frac{1}{2}\right)^{2H}\sigma_0^2 = \frac{1}{4}\sigma_0^2 + S_1^2.$$

Thus $S_1^2 = \left(\frac{1}{2^1}\right)^{2H}(1 - 2^{2H-2})\sigma_0^2$.

Step 3. Reduce the scaling factor by $\sqrt{2}$; that is, now assume $\frac{1}{\sqrt{8}}$, and divide the two intervals from 0 and $\frac{1}{2}$ and from $\frac{1}{2}$ to 1 again. $Y_{\frac{1}{4}}$ is set as the average $\frac{1}{2}\left(Y_0 + Y_{\frac{1}{2}}\right)$ plus an offset $d_{2,1}$, which is an NRN multiplied by the current scaling factor $\frac{1}{\sqrt{8}}$. The corresponding formula holds for $Y_{\frac{3}{4}}$, that is,

$$Y_{\frac{3}{4}} = \frac{1}{2}(Y_{\frac{1}{2}} + Y_1) + d_{2,2},$$

where $d_{2,2}$ is a random offset computed as before. Therefore, the variance S_2^2 of $d_{2,*}$ must be chosen such that

$$\mathrm{Var}\left[Y_{\frac{1}{4}} - Y_0\right] = \frac{1}{4}\mathrm{Var}\left[Y_{\frac{1}{2}} - Y_0\right] + S_2^2, \quad \left(\frac{1}{2^2}\right)^{2H}\sigma_0^2 = \frac{1}{4}\left(\frac{1}{2}\right)^{2H}\sigma_0^2 + S_2^2.$$

Thus $S_2^2 = \left(\frac{1}{2^2}\right)^{2H}(1 - 2^{2H-2})\sigma_0^2$.

Step 4. Proceed in the same manner: reduce the scaling factor by $\sqrt{2}$, that is, scale by $\frac{1}{\sqrt{16}}$. Then set

$$Y_{\frac{1}{8}} = \frac{1}{2}\left(Y_0 + Y_{\frac{1}{4}}\right) + d_{3,1}, \quad Y_{\frac{3}{8}} = \frac{1}{2}\left(Y_{\frac{1}{4}} + Y_{\frac{1}{2}}\right) + d_{3,2},$$
$$Y_{\frac{5}{8}} = \frac{1}{2}\left(Y_{\frac{1}{2}} + Y_{\frac{3}{4}}\right) + d_{3,3}, \quad Y_{\frac{7}{8}} = \frac{1}{2}\left(Y_{\frac{3}{4}} + Y_1\right) + d_{3,4}.$$

In each formula, $d_{3,*}$ is computed as a different NRN multiplied by the current scaling factor $\frac{1}{\sqrt{16}}$. The following step computes **Y** at $t = \frac{1}{16}, \frac{3}{16}, \ldots, \frac{15}{16}$ using a scaling factor again reduced by $\sqrt{2}$, and continues as indicated above. The variance S_3^2 of $d_{3,*}$ is chosen such that

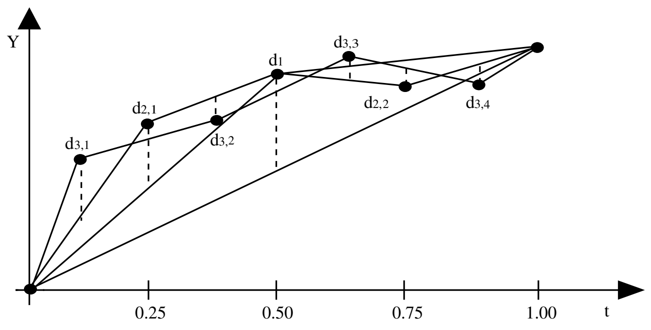

Fig. 7. The first three stages in the RMD method.

$$\mathrm{Var}[Y_{\frac{1}{8}} - Y_0] = \frac{1}{4}\mathrm{Var}\left[Y_{\frac{1}{4}} - Y_0\right] + S_3^2, \quad \left(\frac{1}{2^3}\right)^{2H}\sigma_0^2 = \frac{1}{4}\left(\frac{1}{2^2}\right)^{2H}\sigma_0^2 + S_3^2,$$

that is, $S_3^2 = \left(\frac{1}{2^3}\right)^{2H}(1 - 2^{2H-2})\sigma_0^2$. The variance S_n^2 of $d_{n,*}$, therefore, yields $\left(\frac{1}{2^n}\right)^{2H}(1 - 2^{2H-2})\sigma_0^2$.

Step 5. Calculate the values at the midpoints in the previous same manner until the given n is equal to $2^{No\ of\ Steps}$. Then form the final sequence $X_0, X_1, X_2, \ldots$ by assigning $X_i \leftarrow Y_{i/2^{NoofSteps}}$, $i = 0,1,2,\ldots$.

A self-similar sequence $\{X_0, X_1, \ldots, X_n\}$ is obtained from the previous steps. Generation of an approximately self-similar sequence with 1,048,576 numbers took 17 s on a Pentium II (233 MHz, 512 MB). The theoretical algorithmic complexity is O(n) [49].

4.4. Successive random addition method

An alternative method for the direct generation of an FBM process is based on the *successive random addition* (SRA) algorithm [29,31]. The SRA method uses the midpoints as the RMD method does, but adds a displacement of a suitable variance to all the points [49]. Adding offsets to all points should make the resultant sequence self-similar and of normal distribution [49]. The SRA method consists of the following steps:

Step 1. If the process $\mathbf{Y}$ is to be computed for time instances t between 0 and 1, then begin by setting $Y_0 = 0$ and selecting Y_1 as a pseudo-random number from a normal distribution with mean 0 and variance $\mathrm{Var}[Y_1] = \sigma_0^2$. Then $\mathrm{Var}[Y_1 - Y_0] = \sigma_0^2$.

Step 2. Next, $Y_{\frac{1}{2}}$ is constructed by the interpolation of the midpoint, that is, $Y_{\frac{1}{2}} = \frac{1}{2}(Y_0 + Y_1)$.

Step 3. Add a displacement of a suitable variance to all points, i.e., $Y_0 = Y_0 + d_{1,1}$, $Y_{\frac{1}{2}} = Y_{\frac{1}{2}} + d_{1,2}$, $Y_1 = Y_1 + d_{1,3}$. The offsets $d_{1,*}$ are governed by a normal random number. For $\mathrm{Var}[Y_{t_2} - Y_{t_1}] = |t_2 - t_1|^{2H}\sigma_0^2$ to be true, for any t_1, t_2, $0 \leqslant t_1 \leqslant t_2 \leqslant 1$, it is required that$\mathrm{Var}\left[Y_{\frac{1}{2}} - Y_0\right] = \frac{1}{4}\mathrm{Var}[Y_1 - Y_0] + 2S_1^2$, $\left(\frac{1}{2}\right)^{2H}\sigma_0^2 = \frac{1}{4}\sigma_0^2 + 2S_1^2$, that is, $S_1^2 = \frac{1}{2}\left(\frac{1}{2^1}\right)^{2H}(1 - 2^{2H-2})\sigma_0^2$.

Step 4. Step 2 and Step 3 are repeated. Therefore, $S_n^2 = \frac{1}{2}\left(\frac{1}{2^n}\right)^{2H}(1 - 2^{2H-2})\sigma_0^2$, where σ_0^2 is the initial variance and $0 < H < 1$.

Step 5. Calculate the values at the midpoints as noted previously until the given n is equal to $2^{No\ of\ Steps}$. Then form the final sequence $X_0, X_1, X_2, \ldots$ by assigning $X_i \leftarrow Y_{i/2^{NoofSteps}}$, $i = 0, 1, 2, \ldots$.

Using these steps, the SRA method generates an approximately self-similar sequence $\{X_0, X_1, \ldots, X_n\}$. It took 15 s to generate a sequence of 1,048,576 numbers on a Pentium II (233 MHz, 512 MB). The theoretical algorithmic complexity is O(n) [49].

4.5. Fractional gaussian noise and daubechies wavelets method

We present a new generator of pseudo-random self-similar sequences based on fractional Gaussian noise (FGN) and Daubechies wavelets (DW), called the FGN-DW method [26,27]. A pseudo-random generator of self-similar teletraffic based on Haar wavelet transforms has been proposed in [50,51 and 52]. We used Daubechies wavelets because the generator based on Daubechies wavelets produces more accurate self-similar sequences than one based on Haar wavelets. In other words, not only estimates of H obtained from the Daubechies wavelets are closer to the true values than those from the Haar wavelets, but also variances obtained from the Daubechies wavelets are lower. The reason behind is that the Daubechies wavelets produce smoother coefficients of wavelets that are used in the discrete wavelet transform than the Haar wavelets [53–55]. Haar wavelets are discontinuous, and they do not have good time–frequency localisation properties, since their Fourier transforms decay as $|\lambda|^{-1}$, for $\lambda \to \infty$, meaning that the resulting decomposition has a poor scale. Therefore, Daubechies wavelets produce more accurate coefficients than Haar wavelets; for a more detailed discussion, see [53,55].

Our method for generating synthetic self-similar FGN sequences in a time domain is based on a discrete wavelet transform (DWT). Wavelets can provide compact representations for a class of FGN processes [56,57,54], because the structure of wavelets naturally matches the self-similar structure of long-range dependent processes [53,55,58].

We claim that the FGN-DW method is sufficiently fast for the practical generation of synthetic self-similar sequences that can be used as simulation input data. The general strategy behind our method is similar to Paxson's, who used the Fourier transform [20].

Fig. 8 graphically illustrates a discrete Fourier and a discrete wavelet transform. Wavelet analysis transforms a sequence onto a time-scale grid, where the term *scale* is used instead of *frequency*, because the mapping is not directly related to frequency as in the Fourier transform. The wavelet transform delivers good resolution in both time and scale, as compared to the Fourier transform, which provides only good frequency resolution. The algorithm consists of the following steps:

Step 1. Given: H. Start for $i = 1$ and continue until $i = n$. Calculate a sequence of values $\{f_1, f_2, \ldots, f_n\}$ using Eq. (23) (following), where $f_i = \hat{f}\left(\frac{\pi i}{n}; H\right)$, corresponding to the spectral density of an FGN process for frequencies f_i ranging between $\frac{\pi}{n}$ and π.The main difficulty with using Eq. (18) when computing the spectral density is that it requires to execute the infinite summation. The approximation of $f(\lambda,H)$ is given in [41] as

$$f(\lambda, H) = c_f|\lambda|^{1-2H} + \mathrm{O}(|\lambda|^{min(3-2H,2)}), \tag{23}$$

where c_f is Eq. (19) and $\mathrm{O}(\cdot)$ represents the residual error.This formula was used in the generation of self-similar sequences proposed in this paper. Another generator of self-similar sequences based on FGN was also proposed by Paxson [20], but his method was based on a more complicated approximation of $f(\lambda,H)$ as shown in Eq. (20). Eq. (23) can be used to determine $f(\lambda, H)$ for $\lambda \to \infty$, or for $n \to \infty$ at $\lambda = \frac{\pi}{n}$. For a large value of $\lambda, f(\lambda,H)$ can be calculated by Eq. (18).

Step 2. Multiply $\{f_i\}$ by realisations of an independent exponential random variable with a mean of one to obtain $\{\hat{f}_i\}$, because the spectral density estimated for a given frequency is distributed asymptotically as an independent exponential random variable with mean $f(\lambda, H)$ [42].

Step 3. Generate a sequence $\{Y_1, Y_2, \ldots, Y_p\}$ of complex numbers such that $|Y_i| = \sqrt{\hat{f}_i}$ and the phase of Y_i is uniformly distributed between 0 and 2π. This random phase technique, taken from Schiff [43], preserves the spectral density corresponding to $\{\hat{f}_i\}$. It also makes the marginal distribution of the final sequence normal and produces the requirements for FGN.

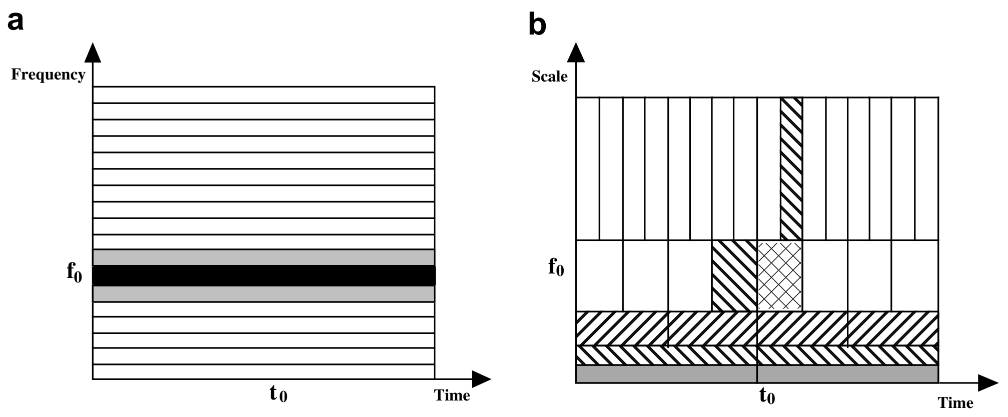

Fig. 8. A graphical representation of a discrete Fourier transform and a discrete wavelet transform: (a) a discrete Fourier transform and (b) a discrete wavelet transform.

Step 4. Calculate the two synthetic coefficients of orthonormal Daubechies wavelets that are used in the inverse DWT (IDWT). The output sequence $\{X_1, X_2, \ldots, X_n\}$ representing approximately self-similar FGN process (in time domain) is obtained by applying the IDWT operation to the sequence $\{Y_1, Y_2, \ldots, Y_n\}$.

Using the previous steps, the proposed FGN-DW method generates a fast and sufficiently accurate self-similar FGN process $\{X_1, X_2, \ldots, X_n\}$. It took 16 s to generate a sequence of 1,048,576 numbers on a Pentium II (233 MHz, 512 MB). Its theoretical algorithmic complexity is O(n). Moreover, the accuracy of Daubechies wavelets is slightly better than Haar wavelets, but there is no difference in the time taken to obtain the same number of coefficients. For more detailed discussion, see also [31].

5. Comparison of fixed-length generators

Paxson [20] and Lau et al. [28] suggest that the FFT- and RMD-based methods are sufficiently fast in the generation of simulation input data for practical applications. In this paper, we report on the properties of these two methods and the F-ARIMA-based method, and compare them with SRA and FGN-DW, two recently proposed alternative methods for the generation of pseudo-random self-similar sequences [26,30]. These five fixed-length sequence generators are comparable because all of them have the same statistical properties, such as normal marginal distributions, means and variances. They were implemented in C on a Pentium II (233 MHz, 512 MB) computer. The mean times required for generating sequences of a given length were obtained using the SunOS 5.7 `time` command and were averaged over 30 replications, each with sequences of 32,768 (2^{15}), 65,536 (2^{16}), 131,072 (2^{17}), 262,144 (2^{18}), 524,288 (2^{19}) and 1,048,576 (2^{20}) numbers.

We have analysed the accuracy with which five considered generators generate normal pseudo-random sequences with the required value of H. For $H = 0.6$, 0.7, 0.8 and 0.9, each method was used to generate 30 sample sequences of 32,768 (2^{15}) numbers starting from different random seeds. Self-similarity and marginal distributions of the sequences generated were assessed by the same techniques as those used in Section 3.

5.1. Accuracy of generated sequences

A summary of the results of our analysis follows:

The estimates of the Hurst parameter for the wavelet-based H estimator and Whittle's MLE are shown in Table 4. All results were averaged over 30 sequences.

(a) The results for the wavelet-based H estimator with the corresponding 95% confidence intervals $\widehat{H} \pm 1.96\hat{\sigma}_{\widehat{H}}$, (see Table 4), show that for all input H values, the F-ARIMA, the FFT and the FGN-DW methods produced sequences with less biased H values than other methods.Using the FFT method, for $H = 0.6$, 0.7 and 0.8, the values of the Hurst parameter from the sample sequences match the required values well, but for $H = 0.9$, the accuracy of the match is lower. Relative error was +0.08%,

Table 4
Mean values of estimated H using the wavelet-based H estimator for the five fixed-length sequence generators for $H = 0.6$, 0.7, 0.8 and 0.9. We give 95% confidence intervals for the means in parentheses

Methods	Mean values of estimated H and ΔH							
	0.6		0.7		0.8		0.9	
	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)
F-ARIMA	.5974 (.593, .601)	−0.427	.6990 (.693, .704)	−0.142	.7947 (.787, .801)	−0.663	.8900 (.880, .899)	−1.115
FFT	.6005 (.596, .604)	+0.083	.6967 (.692, .700)	−0.469	.7862 (.782, .790)	−1.719	.8639 (.859, .867)	−4.012
FGN-DW	.6013 (.574, .629)	+0.214	.6987 (.671, .726)	−0.185	.7962 (.769, .824)	−0.474	.8938 (.866, .921)	−0.694
RMD	.5963 (.591, .601)	−0.613	.6907 (.684, .696)	−1.332	.7805 (.773, .787)	−2.443	.859 2 (.852, .866)	−4.536
SRA	.5848 (.579, .589)	−2.528	.6797 (.674, .685)	−2.899	.7700 (.763, .776)	−3.744	.8499 (.842, .856)	−5.568

−0.47%, −1.75% and −4.11%, respectively.The estimated values of the F-ARIMA method were similar to the FFT method. For $H = 0.6$, 0.7, 0.8 and 0.9, all values of the Hurst parameter from the sample sequences were lower than the required values. Relative error was −0.43%, −0.14%, −0.66% and −1.12%, respectively.The FGN-DW method demonstrated a high level of accuracy and was fast. For $H = 0.6$, 0.7, 0.8 and 0.9, the relative error was +0.21%, −0.19%, −0.47% and −0.69%, respectively.The RMD method generated approximately self-similar sequences [28,30]. For $H = 0.6$, 0.7, 0.8 and 0.9, the Hurst parameter tended to be lower than the required value. Relative error was −0.61%, −1.33%, −2.44% and −4.54%, respectively.The SRA method results were similar to the RMD results. SRA generated self-similar sequences with the most biased H values. For a more detailed discussion, see [30].

(b) The results for Whittle's MLE with the corresponding 95% confidence intervals $\widehat{H} \pm 1.96\hat{\sigma}_{\widehat{H}}$, (see Table 5), show that for all input H values, the FFT and the FGN-DW methods produced sequences with less biased H values than other methods.The FFT method demonstrated a high level of accuracy. For $H = 0.6$, 0.7, 0.8 and 0.9, the values of the Hurst parameter from the sample sequences match the required values very well. Relative error was +0.03%, +0.03%, +0.03% and +0.02%, respectively.Using the F-ARIMA method, for $H = 0.6$, 0.7, 0.8 and 0.9, all values of the Hurst parameter from the sample sequences were lower than the required values. Relative error was −3.28%, −5.31%, −6.64% and −7.51%, respectively.The FGN-DW method is more accurate than the F-ARIMA, RMD and SRA methods, but not the FFT method. For $H = 0.6$, 0.7, 0.8 and 0.9, the relative error was −2.52%, −3.92%, −4.75% and −5.22%, respectively.The RMD method generated approximately self-similar sequences [28,30]. For $H = 0.6$, 0.7, 0.8 and 0.9, the Hurst parameter tended to be lower than the required value. Relative error was −3.91%, −6.18%, −7.48% and −8.21%, respectively.The SRA method results were similar to the RMD results. It generated self-similar sequences with the most biased H values. For a more detailed discussion, see [31].

Our results show that all five generators produced approximately self-similar sequences, with the relative inaccuracy ΔH increasing with H, but always remaining below 9%.

5.2. Complexity and speed of generation

The computational complexities of the five fixed-length sequence generators for generating pseudo-random self-similar sequences of a given length n are shown in Table 6. The number of arithmetic operations per number required by each of them are also shown in Table 6. The F-ARIMA method was the slowest, requiring $178n^2 + 3{,}936$ time operations per number. FFT was also slower than the FGN-DW, RMD and SRA methods.

The results of our experimental analysis of the mean times required by each generator are shown in Fig. 9. Our main conclusions are:

(a) The F-ARIMA method was the slowest of the five methods, as expected.

(b) On average, the FFT method was faster than F-ARIMA, but slower than the other three. This was caused by the relatively high complexity of the inverse FFT algorithm. FFT requires $O(n \log n)$ computations to generate n numbers [45] and $49 \log n + 4175$ time operations per number.

Table 5
Mean values of estimated H using Whittle's MLE for the five fixed-length sequence generators for $H = 0.6$, 0.7, 0.8 and 0.9. We give 95% confidence intervals for the means in parentheses

Methods	Mean values of estimated H and ΔH							
	0.6		0.7		0.8		0.9	
	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)	$\widehat{H}$	ΔH(%)
F-ARIMA	.5803 (.571, .590)	−3.281	.6628 (.654, .672)	−5.308	.7469 (.738, .756)	−6.642	.8324 (.823, .842)	−7.507
FFT	.6002 (.591, .610)	+0.027	.7002 (.691, .710)	+0.033	.8003 (.791, .809)	+0.033	.9002 (.891, .909)	+0.024
FGN-DW	.5849 (.575, .594)	−2.521	.6725 (.663, .682)	−3.924	.762 (.753, .771)	−4.745	.853 (.844, .862)	−5.223
RMD	.5765 (.567, .586)	−3.910	.6567 (.647, .666)	−6.180	.7401 (.731, .749)	−7.482	.8261 (.817, .835)	−8.214
SRA	.5762 (.567, .586)	−3.965	.6563 (.647, .666)	−6.249	.7395 (.730, .749)	−7.567	.8252 (.816, .834)	−8.311

Table 6
Computational complexities and arithmetic operations required by each of the five fixed-length sequence generators

Method	Complexity	Operations per number
F-ARIMA	$O(n^2)$	$178n^2 + 3936$
FFT	$O(n \log n)$	$49 \log n + 4175$
FGN-DW	$O(n)$	4091
RMD	$O(n)$	4103
SRA	$O(n)$	4167

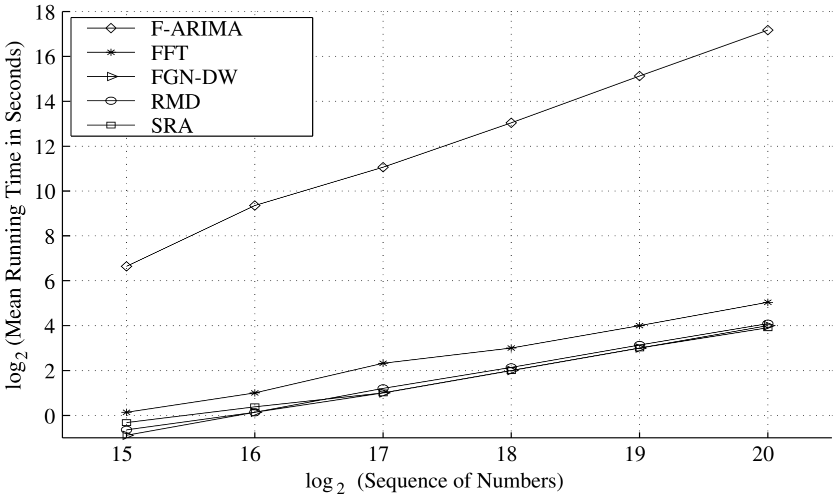

Fig. 9. Mean running times of the five fixed-length sequence generators. Running times were obtained using the SunOS 5.7 `time` command on a Pentium II (233 MHz, 512 MB); each mean is averaged over 30 iterations, each with sequences of 32,768 (2^{15}), 65,536 (2^{16}), 131,072 (2^{17}), 262,144 (2^{18}), 524,288 (2^{19}) and 1,048,576 (2^{20}) numbers.

(c) The FGN-DW, RMD and SRA methods were equally fast. The theoretical complexity of forming a spectral density, and constructing normally distributed complex numbers, is O(1), while the inverse DWT is $O(n)$ [20,55]. Thus, the time complexity of FGN-DW is also $O(n)$ and this method requires 4091 operations per number. The theoretical algorithmic complexity of the RMD and SRA methods is also $O(n)$ [49] and they require 4103 and 4167 operations per number, respectively.

Overall, our results showed that the generator based on FGN-DW is the fastest of the five generators if long sequences of self-similar pseudo-random numbers are required.

6. Sequential generators versus fixed-length sequence generators

We have presented the results of a comparative analysis of six sequential generators of (long) pseudo-random self-similar sequences. All six sequential generators, based on the FBNDP, SFRP, MGIP, PMPP, SRP-FGN and SAP methods, generated approximately self-similar sequences; SRP-FGN was the most accurate. However, our results show that for most input H values, the MGIP and SAP-based generators were strongly biased. The FBNDP method was biased for $H = 0.9$.

The analysis of mean times required to generate sequences of a given length demonstrates that all six sequential generators are more attractive than the F-ARIMA-based generator for practical simulation studies of communication networks because they are much faster. However, these generators require more input parameters, and selecting appropriate values is a problem that remains.

Furthermore, in the case of SAP, the question of how to define the relationship between the Hurst parameter and the two shape parameters (i.e., $\alpha_1 > 0$ and $\alpha_2 > 0$) of a beta-distribution remains.

We have also presented the results of a comparative analysis of five fixed-length generators of self-similar sequences. All five, based on the F-ARIMA, FFT, FGN-DW, RMD and SRA methods, generated approximately self-similar sequences, with the relative inaccuracy of the resultant $\widehat{H}$ below 9% if $0.6 \leqslant H \leqslant 0.9$. However, the analysis of mean times required to generate sequences of a given length shows that the FFT, FGN-DW, RMD, and SRA generators are more attractive for practical simulation studies of communication networks because they generate sequences much faster. When the wavelet-based H estimator and Whittle's MLE (the least biased of the H estimation techniques), are applied (see Chapter 3 in [31]), FFT produces the most accurate results, with the FGN-DW results almost as accurate. Thus, FFT and FGN-DW are the most practical in both accuracy and speed for simulation studies with self-similar input.

Table 7 and Fig. 10 show a comparison of the three fastest and most accurate generators of the six sequential and five fixed-length sequence generators: the SRP-FGN, FFT and FGN-DW generators. While estimated H values for the FGN-DW method obtained using the wavelet-based H estimator were more accurate than those for the SRP-FGN and FFT methods, those for the FFT method obtained using Whittle's MLE were the most accurate.

Even though the SRP-FGN and FGN-DW methods have the same computational complexity, O(n), the FGN-DW method was faster than the SRP-FGN and FFT methods, as shown in Fig. 10. Furthermore, while the SRP-FGN method required three input parameters (i.e., H, M and T), the FFT and FGN-DW methods required only the Hurst parameter H. Thus, the FFT method was more accurate than the SRP-FGN and FGN-DW methods, and the FGN-DW method was faster than the other two. Overall, all three methods

Table 7
Comparison of the most efficient SRP-FGN, FFT and FGN-DW methods. Relative inaccuracies of mean values of estimated H obtained using the wavelet-based H estimator and Whittle's MLE

Estimator	Method	ΔH			
		0.6	0.7	0.8	0.9
Wavelet-based	SRP-FGN	−0.968	−0.569	+0.700	+0.344
	FFT	+0.083	−0.469	−1.719	−4.012
	FGN-DW	+0.214	−0.185	−0.474	−0.694
Whittle's MLE	SRP-FGN	+0.110	+1.905	+2.539	+2.526
	FFT	+0.027	+0.033	+0.033	+0.024
	FGN-DW	−2.521	−3.924	−4.745	−5.223

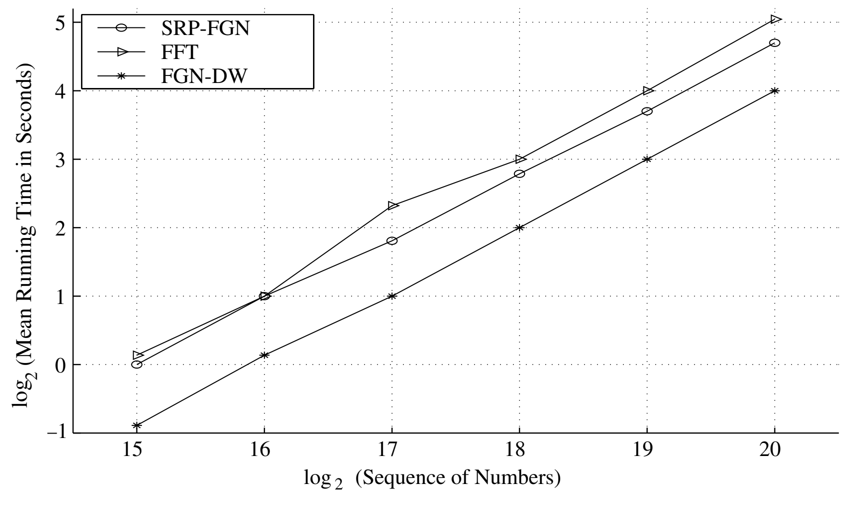

Fig. 10. Mean running times of the SRP-FGN, FFT and FGN-DW generators. Running times were obtained using the SunOS 5.7 `time` command on a Pentium II (233 MHz, 512 MB); each mean is averaged over 30 iterations, each with sequences of 32,768 (2^{15}), 65,536 (2^{16}), 131,072 (2^{17}), 262,144 (2^{18}), 524,288 (2^{19}) and 1,048,576 (2^{20}) numbers.

are more attractive for practical simulation studies of telecommunication networks than the other nine generators.

7. Conclusions

One of the problems that communication network researchers face when conducting simulation studies is how to generate long synthetic sequential self-similar sequences. Three aspects must be considered: (i) how accurately self-similar processes can be generated, (ii) how quickly the methods generate long self-similar sequences, and (iii) how appropriately self-similar processes can be used in sequential simulations.

Most of the existing synthetic methods for generating self-similar sequences require large amounts of CPU time. Some current methods must store either part, or all, of the sequence in memory before generating numbers of a sequence. In addition, they are often inaccurate or inappropriate in simulation studies of communication networks. Sequential generators of self-similar sequences depend on the level of approximation, and need several input parameters to be assumed, while fixed-length sequence generators need only to assume the Hurst parameter to generate self-similar sequences.

Certainly, more efficient and accurate generators of self-similar sequences of pseudo-random numbers are needed. A comparative study of self-similar pseudo-random teletraffic generators was undertaken. Overall, our results, obtained using the wavelet-based H estimator and Whittle's MLE, which are the least biased of the H estimation techniques considered in [31], have revealed that the fastest and most accurate generators of the six sequential and five fixed-length sequence generators considered are the SRP-FGN, FFT and FGN-DW methods. However, these methods have both strengths and weaknesses. The FFT and FGN-DW methods are more attractive for non-sequential simulations, because they can generate the required number of sequences more accurately and quickly than the SRP-FGN method. If the FFT and FGN-DW methods are used for sequential simulations, sufficient numbers of sequences must be generated before the simulation begins. However, the required number of sequences is not easy to predict in practical simulations. On the other hand, the SRP-FGN method is more attractive for sequential simulations, because it does not need to "know" the required number of sequences beforehand. Unfortunately, this method is less accurate and requires more generating time than the FFT method.

Network layer traffic generators that can generate packet arrival processes have been addressed in this paper. In practical simulation studies of communication networks, application level traffic generators are more approaching to the real network traffic. While these network layer traffic generators are more flexible [31], those application-layer traffic generators [59–61] are only fit to a specific process. On the other hand, for practical simulation studies of communication networks, self-similar processes with arbitrary marginal distributions are needed. In order to obtain these, it is needed to transform a given sequence from the self-similar processes into a sequence that represents a realisation of a specific process. For example, the FFT and FGN-DW methods can be used to synthesise VBR video traffic, and the SRP-FGN method can be used to investigate the queueing behaviour in (steady-state) simulation studies of queueing systems fed by self-similar input.

Acknowledgements

We thank anonymous referees for their constructive remarks and valuable comments. This work was partially supported by the Korean Bible University's Research Grant, and Korea Institute of Science and Technology Information, Korea.

References

[1] T. Karagiannis, M. Molle, M. Faloutsos, Long-range dependence: ten years of internet traffic modelling, IEEE Internet Computing 8 (5) (2004) 57–64.

[2] W. Leland, M. Taqqu, W. Willinger, D. Wilson, On the self-similar nature of ethernet traffic (extended version), IEEE ACM Transactions on Networking 2 (1) (1994) 1–15.

[3] W. Willinger, M. Taqqu, R. Sherman, D. Wilson, Self-similarity through high-variability: statistical analysis of ethernet LAN traffic at the source level, IEEE ACM Transactions on Networking 5 (1) (1997) 71–86.

[4] R.E.A. Khayari, R. Sadre, B.R. Haverkort, A. Ost, The pseudo-self-similar traffic model: application and validation, Performance Evaluation 56 (1–4) (2004) 3–22.

[5] J. López-Ardao, C. López-García, A. Suárez-González, M. Fernández-Veiga, R. Rodríguez-Rubio, On the use of self-similar processes in network simulation, ACM Transactions on Modeling and Computer Simulation 10 (2) (2000) 125–151.

[6] S. Robert, J.-Y. LeBoudec, On a Markov modulated chain exhibiting self-similarities over finite timescale, Performance Evaluation 27–28 (1996) 159–173.

[7] S. Robert, J.-Y. LeBoudec, New models for pseudo self-similar traffic, Performance Evaluation 30 (1–2) (1997) 57–68.

[8] A. Andersen, B. Nielsen, An application of superpositions of two-state Markovian sources to the modelling of self-similar behaviour, in: Proceedings of IEEE INFOCOM'97, Kobe, Japan, 1997, pp. 196–204.

[9] S. Lowen, M. Teich, Estimation and simulation of fractal stochastic point processes, Fractals 3 (1) (1995) 183–210.

[10] B. Ryu, S. Lowen, Point process approaches to the modeling and analysis of self-similar traffic – Part i: model construction, in: Proceedings of IEEE INFOCOM'96, San Francisco, CA, 1996, pp. 1468–1475.

[11] B. Ryu, S. Lowen, Point process models for self-similar network traffic, with applications, Communications in Statistics-Stochastic Models 14 (3) (1998) 735–761.

[12] B. Ryu, Fractal Network Traffic: from Understanding to Implications, Ph.D. thesis, Graduate School of Arts and Sciences, Columbia University, 1996.

[13] B. Mandelbrot, Long-run linearity, locally gaussian processes, H-spectra and infinite variances, International Economic Review 10 (1969) 82–113.

[14] M. Taqqu, J. Levy, Using renewal processes to generate long-range dependence and high variability, Dependence in Probability and Statistics 11 (1986) 73–89.

[15] D. Cox, V. Isham, Point Processes, Chapman and Hall, New York, 1980.

[16] D. Cox, Long-range dependence: a review, in: H.A. David, H.T. David (Eds.), Statistics: An Appraisal, Iowa State Statistical Library, The Iowa State University Press, 1984, pp. 55–74.

[17] T. Le-Ngoc, S. Subramanian, A pareto-modulated poisson process (PMPP) model for long-range dependent traffic, Computer Communications 23 (2000) 123–132.

[18] T. Taralp, M. Devetsikiotis, I. Lambadaris, A. Bose, Efficient fractional Gaussian noise generation using the spatial renewal process, in: Proceedings of IEEE International Conference on Communications (ICC'98), Atlanta, GA, USA, 1998, pp. 7–11.

[19] C. Granger, Long memory relationships and the aggregation of dynamic models, Journal of Econometrics 14 (1980) 227–238.

[20] V. Paxson, Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic, Computer Communication Review, ACM SIGCOMM 27 (5) (1997) 5–18.

[21] J. Hosking, Modeling persistence in hydrological time series using fractional differencing, Water Resources Research 20 (12) (1984) 1898–1908.

[22] A. Erramilli, P. Pruthi, W. Willinger, Fast and physically-based generation of self-similar network traffic with applications to ATM performance evaluation, in: S. Andradottir, K.J. Healy, D.H. Withers, B.L. Nelson (Eds.), Proceedings of the 1997 Winter Simulation Conference, Atlanta, Georgia, 1997, pp. 997–1004.

[23] A. Erramilli, R. Singh, P. Pruthi, Chaotic maps as models of packet traffic, in: J. Labetoulle, J.W. Roberts (Eds.), The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks: Proceedings of the 14th International Teletraffic Congress-ITC 14, 1a, Elsevier, Amsterdam, Antibes Juan-les-Pins, France, 1994, pp. 329–338.

[24] W. Leland, W. Willinger, M. Taqqu, D. Wilson, Statistical analysis and stochastic modeling of self-similar data traffic, in: J. Labetoulle, J.W. Roberts (Eds.), The Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks: Proceedings of the 14th International Teletraffic Congress-ITC 14, 1a, Elsevier, Amsterdam, Antibes Juan-les-Pins, France, 1994, pp. 319–328.

[25] E. Costamagna, L. Favalli, P. Gamba, G. Iacovoni, A simple model for VBR video traffic based on chaotic maps: validation through evaluation of ATM multiplexers QoS parameters, in: Proceedings of IEEE International Conference on Communications (ICC'98), Atlanta, GA, USA, 1998, pp. S16_6.1–S16_6.5.

[26] H.-D. Jeong, D. McNickle, K. Pawlikowski, Fast self-similar teletraffic generation based on FGN and wavelets, in: Proceedings of the IEEE International Conference on Networks, ICON'99, Brisbane, Australia, 1999, pp. 75–82.

[27] H.-D. Jeong, J.-S. Lee, D. McNickle, K. Pawlikowski, Distributed steady-state simulation of telecommunication networks with self-similar teletraffic, Simulation Practice and Theory 13 (3) (2005) 233–256.

[28] W.-C. Lau, A. Erramilli, J. Wang, W. Willinger, Self-similar traffic generation: the random midpoint displacement algorithm and its properties, in: Proceedings of IEEE International Conference on Communications (ICC'95), Seattle, WA, 1995, pp. 466–472.

[29] A. Crilly, R. Earnshaw, H. Jones, Fractals and Chaos, Springer-Verlag, New York, 1991.

[30] H.-D. Jeong, D. McNickle, K. Pawlikowski, A comparative study of three self-similar teletraffic generators, in: Proceedings of 13th European Simulation Multiconference, ESM'99, vol. 1, Warsaw, Poland, 1999, pp. 356–362.

[31] H.-D. Jeong, Modelling of Self-Similar Teletraffic for Simulation, Ph.D. thesis, Department of Computer Science, University of Canterbury, 2002.

[32] W. Feller, An Introduction to Probability Theory and Its ApplicationsThird edition, I, John Wiley & Sons, Inc., New York, 1968.

[33] M. Parulekar, A. Makowski, $M/G/\infty$ input processes: a versatile class of models for network traffic, in: Proceedings of IEEE INFOCOM'97, Kobe, Japan, 1997, pp. 419–426.

[34] N. Likhanov, B. Tsybakov, N. Georganas, Analysis of an ATM buffer with self-similar ("fractal") input traffic, in: Proceedings of IEEE INFOCOM'95, Boston, Massachusetts, 1995, pp. 985–992.

[35] C. Klüppelberg, Subexponential distributions and integrated tails, Journal of Applied Probability 25 (1988) 132–141.

[36] P. Jelenković, The Effect of Multiple Time Scales and Subexponentiality on the Behavior of a Broadband Network Multiplexer, Ph.D. thesis, Graduate School of Arts and Sciences, Columbia University, 1996.
[37] H. Leroux, M. Hassan, R. Egudo, Are inter-arrival times of internet traffic also self-similar?, in: Proceedings of IEEE International Conference on Telecommunications (ICT'99), vol. 1, Cheju, South Korea, 1999, pp. 549–553.
[38] V. Paxson, S. Floyd, Wide-area traffic: the failure of poisson modeling, IEEE ACM Transactions on Networking 3 (3) (1995) 226–244.
[39] M. Roughan, J. Yates, D. Veitch, The mystery of the missing scales: pitfalls in the use of fractal renewal processes to simulate LRD processesApplications of Heavy Tailed Distributions in Economics, Engineering and Statistics, American University, Washington, DC, 1999.
[40] S. Ledesma, D. Liu, Synthesis of fractional Gaussian noise using linear approximation for generating self-similar network traffic, Computer Communication Review, ACM SIGCOMM 30 (2) (2000) 4–17.
[41] J. Beran, Statistics for Long-Memory Processes, Chapman and Hall, New York, 1994.
[42] J. Beran, Statistical methods for data with long range dependence, Statistical Science 7 (4) (1992) 404–427.
[43] S. Schiff, Resolving time-series structure with a controlled wavelet transform, Optical Engineering 31 (11) (1992) 2492–2495.
[44] W. Feller, Second ed.An Introduction to Probability Theory and Its Applications, II, John Wiley & Sons Inc., New York, 1966.
[45] W. Press, B. Flannery, S. Teukolsky, W. Vetterling, Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, Cambridge, 1986.
[46] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C, Cambridge University Press, Cambridge, 1999.
[47] J. Hosking, Fractional dfferencing, Biometrika 68 (1) (1981) 165–176.
[48] H.-O. Peitgen, H. Jurgens, D. Saupe, Chaos and Fractals: New Frontiers of Science, Springer-Verlag, New York, 1992.
[49] H.-O. Peitgen, D. Saupe, The Science of Fractal Images, Springer-Verlag, New York, 1988.
[50] V. Ribeiro, R. Riedi, M. Crouse, R. Baraniuk, Simulation of nonGaussian long-range-dependent traffic using wavelets, in: Performance Evaluation Review, Proceedings of ACM SIGMETRICS'99, vol. 27(1), 1999, pp. 1–12.
[51] R. Riedi, M. Crouse, V. Ribeiro, R. Baraniuk, A multifractal wavelet model with application to network traffic, IEEE Transactions on Information Theory 45 (3) (1999) 992–1018.
[52] D. Veitch, J.-A. Bäckar, J. Wall, J. Yates, M. Roughan, On-line generation of fractal and multifractal traffic, in: The PAM2000 Passive and Active Measurement Workshop, Hamilton, New Zealand, 2000.
[53] I. Daubechies, in: Ten Lectures on WaveletsCBMS-NSF Regional Conference Series in Applied Mathematics, 61, SIAM Press, Philadelphia, Pennsylvania, 1992.
[54] M. Roughan, D. Veitch, P. Abry, On-line estimation of the parameters of long-range dependence, in: Proceedings of GLOBECOM'98, Sydney, Australia, 1998, pp. 3716–3721.
[55] M. Wickerhauser, Adapted Wavelet Analysis from Theory to Software, A.K. Peters, Ltd., Wellesley, Massachusetts, 1994.+++.
[56] P. Flandrin, Wavelet analysis and synthesis of fractional Brownian motion, IEEE Transactions on Information Theory 38 (2) (1992) 910–917.
[57] L. Kaplan, C.-C. Kuo, Fractal estimation from noisy data via discrete fractional Gaussian noise (DFGN) and the Haar Basis, IEEE Transactions on Signal Processing 41 (12) (1993) 3554–3562.
[58] P. Abry, D. Veitch, Wavelet analysis of long-range-dependent traffic, IEEE Transactions on Information Theory 44 (1) (1998) 2–15. <http://www.emulab.ee.mu.oz.au/~darryl/>.
[59] J. Cao, W.S. Cleveland, Y. Gao, K. Jeffay, F.D. Smith, M. Weigle, Stochastic models for generating synthetic HTTP source traffic, in: Proceedings of IEEE INFOCOM'2004, Hong Kong, 2004, pp. 1547–1558.
[60] J. Wallerich, Design and Implementation of a WWW Workload Generator for the NS-2 Network Simulator, 2006. <http://www.net.in.tum.de/~jw/nsweb/>.
[61] M. Weigle, P. Adurthi, F. Hernández-Campos, K. Jeffay, F. Smith, Tmix: a tool for generating realistic TCP application workloads in NS-2, Computer Communication Review, ACM SIGCOMM 36 (3) (2006) 65–76, Jul.

A Glossary of Terms and Acronyms

ACF: autocorrelation function
CDF: cumulative distribution function
CPU: central processing unit
DW: Daubechies wavelets
DWT: discrete wavelet transform
F-ARIMA: fractional autoregressive integrated moving-average
FBM: fractional Brownian motion
FBNDP: fractal-binomial-noise-driven Poisson process
FFT: fast Fourier transform
FGN: fractional Gaussian noise
FGN-DW: fractional Gaussian noise and Daubechies wavelets
FRP: fractal renewal process
FSNDP: fractal-short-noise-driven Poisson process
IDC: index of dispersion for counts

IDWT: inverse discrete wavelet transform
IID: independent, identically distributed
LRD: long-range dependence
MB: mega byte
MGIP: $\boldsymbol{M/G/\infty}$ *processes*
ML: mean length
MLE: maximum likelihood estimator
MNPE: mean numbers of Poisson events
MPEG: moving picture experts group
NRN: normal random number
PDF: probability density function
PMPP: Pareto-modulated Poisson processes
RMD: random midpoint displacement
SAP: superposition of autoregressive processes
SFRP: superposition of fractal renewal processes
SRA: successive random addition
SRP-FGN: spatial renewal processes and fractional Gaussian noise