Traffic generator using Perlin Noise

Iria Prieto, Mikel Izal, Daniel Morato, Eduardo Magaña

***Abstract*—Study of high speed networks such as optical next generation burst or packet switched networks require large amounts of synthetic traffic to feed simulators. Methods to generate self-similar long range dependent traffic already exist but they usually work by generating large blocks of traffic of fixed time duration. This limits simulated time or require very high amount of data to be stored before simulation. On this work it is shown how self-similar traffic can be generated using Perlin Noise, an algorithm commonly used to generate 2D/3D noise for natural looking graphics. 1-dimension Perlin Noise can be interpreted as network traffic and used to generate long range dependent traffic for network simulation. The algorithm is compared to more classical approach Random Midpoint Displacement showing at traffic generated is similar but can be generated continuously with no fixed block size.**

***Index Terms*—Optical, Perlin noise, traffic, generator**

I. Introduction

THE fast growth of computer networks and the constant development of new services with different requirements, make necessary the design and study of different protocols and technologies through simulation. This is specially important in the case of optical next generation networks, in order to obtain results and compare performance of still-not-implemented services on network architectures.

In order to simulate the behavior of these networks, different traffic models have been used in the literature: Poisson process, Markovian chains, autoregressive models... However, to capture long range dependence (LRD) of real network traffic, self-similar models are the preferred choice (i.e. for Ethernet LAN, [1], [2] and also in variable bit rate (VBR) compressed video, [3], [4], [5], [6], [7], [8]). Fractional Brownian Traffic is a self-similar traffic model which models the cumulative amount of traffic arrived, as a continuous function with gaussian increments. This increments form what is called a Fractional Gaussian Noise (FGN). FGN is characterized by three parameters. Mean and variance are the parameters of the marginal distribution of arrival process per unit time. The third parameter is the Hurst parameter (H) which measures burstiness of the process. $H = 0.5$ corresponds to the pure gaussian noise with independent arrivals (which is self-similar but not long range dependent). Larger values of $H < 1$ give increasing dependent processes being $H = 0.8 - 0.7$ typical values for Internet traces, citeIzal2006.

One Fractional Gaussian Noise, FGN, process is described in [9]. Due to the computational cost of the method, $O(N^2)$, several methods have been proposed after this, [10], [11]. In order to achieve a compromise between computational cost and accuracy, some others methods can be found in the literature, an example is the method showed in [12]. This method is able to generate sample traces with O(n) effort with respect to length. Another example is the approximate method, described in [13], which has been used in the field of computer graphics, as wide as for traffic generation.

This work was supported by the Spanish Ministry of Science and Innovation through the research project INSTINCT (TEC-2010-21178-C02-01). Also, the authors want to thank Spanish thematic network IPoTN (TEC2010-12250-E) and Public University of Navarre for funding through PIF grant

However these methods usually generate fixed sized blocks of FGN samples, for example [9], is able to generate accurate traces with less samples than $5 \cdot 10^5$. This means that size have to be decided in advance and the full process trace has to be generated and stored before actual simulation. This limits our simulation time specially in high speed networks where we need large traces to simulate even very short times. This behavior comes from FGN generators working by generating the desired spectrum of the process and then using Fast Fourrier Transform to get the actual time-domain process or by generating farthest samples first and then refining the samples in between.

Our goal in this work is obtaining a generator to synthesize FGN on-the-fly with no need for pre-generating large blocks of samples. This generator is needed to feed simulations of high speed optical burst or packet switched networks where a large number long range dependent sources have to be provided for different inputs. To get this on-the-fly generation an adaptation of the known Perlin Noise process is proposed. Perlin Noise is usually used to generate natural looking textures and effects in computers graphics but can be reinterpreted as an FGN generator that have not been previously applied, as far as authors knowledge, to traffic generation. The accuracy of the algorithm's adaptation to obtain traces with mean, variance and Hurst parameter is compared with other FGN generators.

On the other hand FGN generates traffic with a gaussian marginal distribution and thus may generate instant peaks larger than channel capacity. The generation methodology to limit FGN traffic to channel capacity while maintaining target average and burstiness is also addressed.

The paper is organized as follows. First of all, definitions and notation used along the paper are presented. Section II explains the Perlin Noise Traffic Generator and how to obtain desired parameters. Section III presents the method to adapt traces to limited capacity channels and section IV show the results and comparisons. Finally, section IV presents conclusions.

II. Definitions And Notation

Perlin Noise process is defined as a sum of several noises $x_i(t)$ called octaves with increasing spectral components. Each octave is generated from an independent uniform random process. The base octave ($i = 0$) is a random independent noise generated at intervals Δt. For every $t = k\Delta t$ a random

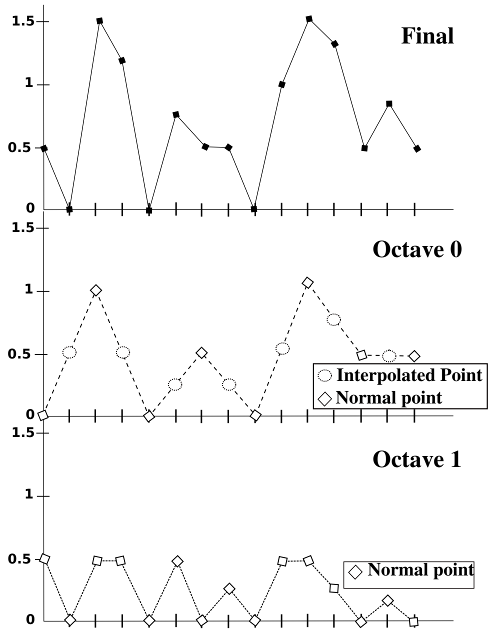

Fig. 1. Example of the Perlin Noise algorithm

independent value is generated with a uniform distribution. For the points in between $t = k\Delta t$ the values are obtained interpolating the previous and next generated value. Interpolation can be linear or any smooth function. Successive octaves ($i = 1...n$) are built in the same way with noise taking values in an increasing number of intermediate points. In octave i the noise take new values in every $k(\Delta t/f^i)$. Each octave considered has a different (usually decreasing) amplitude A_i for the uniform random generation. Amplitude is usually given by a persistence factor p, being $A_i = p^i$.

Given n independent random noise processes $x_i(t)$ interpolated at $k\frac{\Delta t}{f^i}$. The Perlin noise is obtained as

$$n(t) = \sum_{i=0}^{n-1} p^i x_i(t) \tag{1}$$

Figure 1 shows an example with two octaves.

So to summarize, notation:

- n = Number of octaves
- x_i = i-th octave. The first octave will be x_0 and the last one x_{n-1}
- p = Persistence
- A_i = Amplitude for each i-th octave
- f = Factor of interpolation

In such a process note that octave k generates f^k random values for every non-interpolated value of the first octave. In frequency domain octave k has spectral components spreading to f^k times those of first octave. In fact with this generation each octave $x_i(t)$ is approximately a white noise limited to frequency $\frac{2\pi f}{\Delta t}$.

A Perlin Noise with n octaves and interpolating factor f generates f^{n-1} samples of the highest frequency octave for each fresh sample of the first octave. For traffic generation such a process can be interpreted as providing traffic volume arriving in fixed size time slots δt given by the generation time of the high frequency octave. In that case using n octaves means that first octave generates a noise sample and moves slowly during f^{n-1} time slots to its next generated value. Therefore each octave is contributing to generate a process with correlation at least reaching $f^{n-1}\delta t$ time lags.

In this view a Perlin Noise with decaying octave amplitude is generating a long range dependent process with the decay of spectral density controlled by p. This is analogous to a Fractional Gaussian Noise with its decay of power spectral density controlled by the Hurst parameter H. From this idea the objective of this work is to generate FGN traces using Perlin Noise approach instead of RMD.

The usual interpolated factor used for Perlin Noise in graphic applications is $f = 2$. In traffic generation it is very interesting to control the reach of dependence thus larger values of f could be used to get farther correlations.

To adjust the output of Perlin Noise process notice that the amplitude of the sum process is given by 2 and the expectation can be calculated from the expectation of the uniform random generator which is a uniform noise generator $x(t)$ modulated by the octave amplitude p^i

$$Amplitude_{max} = \sum_{i=0}^{n-1} p^i = \frac{p^n - 1}{p - 1} \tag{2}$$

$$\bar{n} = \bar{x} * \sum_{i=0}^{n-1} p^i = \bar{x}\frac{p^n - 1}{p - 1} \tag{3}$$

III. Perlin Noise Traffic Generation

This section shows how to use an adapted Perlin Noise process to obtain FGN traces with the same characteristics that can be obtained using RMD method. The target FGN process is defined by mean, variance and Hurst parameter (μ, σ^2, H), given as input arguments to the algorithm. The FGN trace can be transformed to change mean and variance without changing self-similar and LRD correlation structure indicated by H. Therefore the first step to generate FGN with Perlin Noise is to get a process with desired H by choosing a persistence value p as well as the number of octaves and interpolation factor to use. The Hurst parameter depends on p, the number of octaves n and f which define the frequency domain representation on the process.

Figure 2 show this H dependence on p, n and f. Note that for low persistence factors high values of H are obtained. This can be explained because the process obtained is a sum of components with power concentrated in increasing areas. The sum of such octaves with no modification has decaying power spectrum associated with long range dependence. To get an independent arrival process, namely $H = 0.5$, persistence has to be increased in order to give more weight to the highest octave that is already a white noise.

The number of octaves n and interpolating factor f to choose are important because the larger n and f, the larger is the number of correlated samples obtained.

As we can see in the graph, for example, in order to obtain a $H \simeq 0.7$ using 6 octaves we would have to choose a persistence near $p = 1.5$.

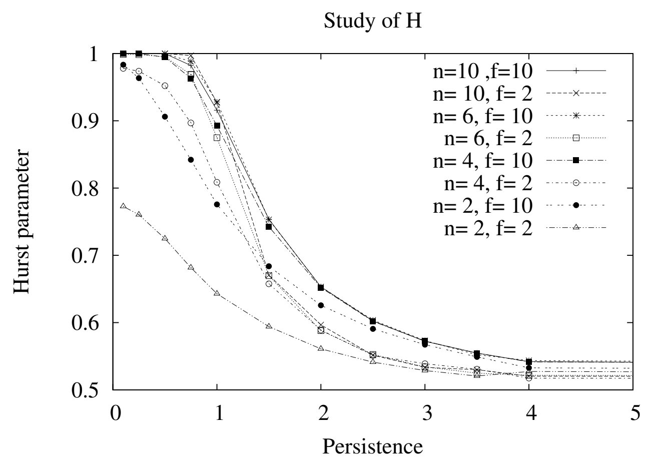

Fig. 2. H parameter depending on number of octaves, persistence and interpolation factor

Therefore n octaves are generated with uniform and independent random generators with $\bar{x_i} = 0$ and weighted with p^i to obtain a Perlin Noise process $n_0(t)$. This process has desired H and mean 0.

The second step is mean and variance adjustment, obtaining a new process $n(t)$ with desired mean μ and variance σ^2.

$$n(t) = \mu + \frac{\sigma}{\sigma_0} n_0(t) \tag{4}$$

Being σ_0^2 the variance of $n_0(t)$. Note that it is easy to generate $n_0(t)$ with 0 mean but not as easy to get a normalized process with variance 1. In the next section we derive the theoretical formula of variance of an $n_0(t)$ built as described.

After rescaling the process to get mean and variance, H parameter keeps unchanged, thus a FGN trace is obtained with target mean, variance and H and correlation reaching as far as f^{n-1}. It should be noted that the value of σ_0 could also be obtained from the actual synthesized $n_0(t)$ once it has been generated. However this work's objective is to generate continuously the process $n(t)$ with no need to generate fixed size block traces in advance. The theoretical value of σ_0 allows the generation of samples of $n(t)$ as soon as samples from $n_0(t)$ are generated.

A. Perlin Noise Variance

In order to calculate the variance of $n_0(t)$ needed for on-the-fly variance scaling a theoretical formula for this variance is derived here. The process $n_0(t)$ is generated as a sum of n independent processes $x_i(t)$ with mean $E[x_i(t)] = 0$. Therefore the sum process will also have $E[n_0(t)] = 0$ and the variance will be the weighted sum of the variances of individual octaves.

$$\mathrm{Var}[n_0(t)] = \sum_{i=0}^{n-1} \mathrm{Var}[p^i x_i(t)] = \sum_{i=0}^{n-1} p^{2i} \mathrm{Var}[x_i(t)] \tag{5}$$

The variance of every octave depends on the density function of the random generator. Every octave is generated from samples of a uniform random variable x with density function $f(x)$ in equation 6, taking values in $(-1, 1)$ with 0 mean.

$$f(x) = \begin{cases} \frac{1}{2} & when \quad -1 \leq x \leq 1 \\ 0 & Otherwise \end{cases} \tag{6}$$

In the highest frequency octave every sample is a realization of the generator but lower frequency octaves have interpolated points in-between fresh random samples. In octave i there are $k = f^{n-1-i}$ interpolated samples. Intermediate points in octave i between a sample with vale a and the next sample with value b can be written as

$$a + (b - a)\frac{i}{k} \qquad \text{for} \quad i = 0...k \tag{7}$$

The variance of the highest frequency octave x_{n-1} is calculated from the density function of the generation

$$\mathrm{Var}[x_{n-1}(t)] = \int x^2 f(x)\,\mathrm{d}x = \int_{-1}^{1} x^2 \frac{1}{2}\,\mathrm{d}x = \frac{1}{3} \tag{8}$$

Variance of other octaves can be derived taking into account that intermediate points are fixed once the values of the random samples are known. Thus we can write the integral conditioned to the first sample being a and integrate just over the right sample of the interval.

$$V_a = \int_{-1}^{1} \frac{1}{k} \sum_{i=1}^{k} \left(\frac{i\ (x-a)}{k} + a \right)^2 \frac{1}{2}\,\mathrm{d}x = \\ = \frac{\left(6\,a^2 + 2\right)\,k^2 + \left(3 - 9\,a^2\right)\,k + 3\,a^2}{18\,k^2} \tag{9}$$

Being a the left random sample and k the number of interpolation intervals at octave i, $k = f^{n-1-i}$. The variance is obtained by integrating again over the left sample density function depending just on k for octave i.

$$V[x_i(t)] = \int_{-1}^{1} V_a f(a)\,\mathrm{d}a = \frac{2\,k^2 + 1}{9\,k^2} \tag{10}$$

With the weighted sum of variances of every $x_i(t)$ the variance of $n_0(t)$ is obtained using equation 5

$$\mathrm{V}[n_0(t)] = \sum_{i=0}^{n-1} p^{2i} V[x_i(t)] = \\ = \frac{\sum_{i=0}^{n-1} \left(2\,f^{2\,n} + f^{2\,i+2}\right)\,p^{2\,i}}{9\,f^{2\,n}} \tag{11}$$

This variance can be calculated just from p, n and f thus allows to rescale $n_0(t)$ to an $n(t)$ with the needed mean and variance. An example of the result of this process is shown in the figure 3. It has been generated using 6 octaves, factor of interpolation $f = 2$ and persistence $p = 1.4$ in order to obtain

TABLE I
ANALYSIS OF VALUES OBTAINED THROUGH STANDARDISATION PROCESS

$\mu_{Desired}$	$\mu_{Obtained}$	$\sigma^2_{Desired}$	$\sigma^2_{Obtained}$	$H_{Desired}$	$H_{Obtained}$
60,000	60,000.82949	15,000	15,047.568	0.7	0.701

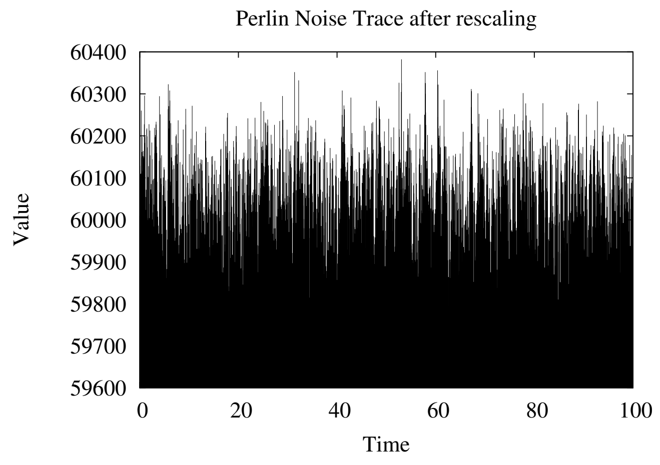

Fig. 3. $n(t)$ with targets $\mu = 60,000$, $\sigma^2 = 15,000$, $H = 0.7$ ($f = 2$, $n = 6$, $p = 1.4$)

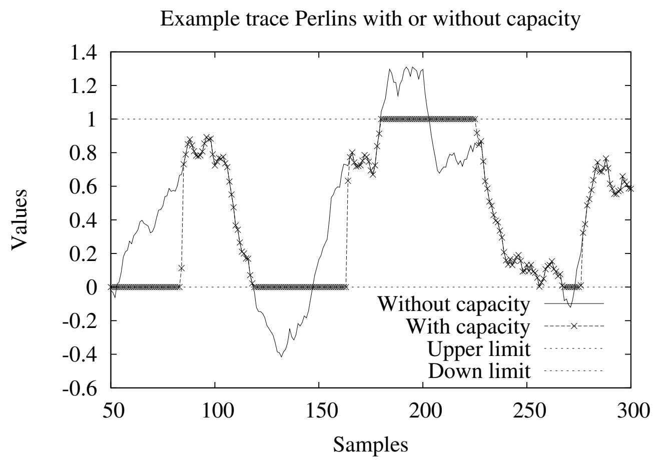

Fig. 4. How really works the algorithm to preserve the mean for real system with capacity showing the acumulative bytes of a Perlin trace

a target $H \simeq 0.7$. In the table I the trace characteristics are shown. In the following sections we analyze the results of the process comparing to other FGN generation methods.

IV. GENERATING LIMITED CAPACITY CHANNEL TRAFFIC

In the previous section a method is provided to generate a FGN with desired mean and variance $n(t)$ by scaling a Perlin Noise process $n_0(t)$. Usually this is interesting to generate traffic on a given channel with limited capacity C in order to simulate network behavior. The problem with this output process is that once mean and variance are chosen the maximum values of traffic volume per sample time may be higher than channel capacity limit or even lower than 0. The FGN process has been generated without those limits in mind.

To fix that, an algorithm is needed that shape the output $n(t)$ to keep it within limits of channel capacity and over 0. The algorithm needs to preserve the mean and as much as possible also variance and Hurst parameter (i.e. correlation structure) in the capacity limited process $n_s(t)$

In order to keep the mean in the system every time generated traffic is over capacity limit, the volume exceeding the capacity is stored to be added to the traffic generated in the next interval. Therefore in periods where traffic generated is over capacity limit for extended time, the cumulative stored traffic is generated after the high traffic epoch ends. This keeps the same expectation of $\bar{n_s}(t) = \bar{n}(t)$ provided that $\bar{n}(t) < C$. On the opposite side when the generated traffic volume in an interval is negative it is also added to the traffic generated in the next interval thus reducing the generated traffic after that and keeping the mean of $n_s(t)$ equal to that of $n(t)$.

An example is shown in figure 4. It shows the bytes for one Perlin Noise trace with the parameter values of: $H \simeq 0.994$, $\mu = 0.3$ and $\sigma^2 \simeq 0.2$. It can be seen how a initial process is followed by the shaped version. The shaped one has never a slope greater that the one given by capacity ($C = 1$ in this case) or lower than 0. However due to the build method it has the same overall growing rate than the original one $\mu = 0.3$.

Shaping the process to obtain desired capacity limit is easy to do with previous algorithm. The problem is the effect of that shaping in the output process variance and correlation structure given by Hurst parameter. It is clear that a process with limited range can not have arbitrary large variance. In the section that follows the limits of variance and H that can be expected from the limited capacity algorithm are shown. Without loss of generality we normalize the capacity limit as $C = 1$ and scale all parameters to fit.

A. Absolute error of Variance

Although the shaping process does not modify the mean of the output traffic process. Variance may be clearly affected. The channel limits will not allow variance to grow too high even if the original process can have arbitrary large variance. Also note the maximum variance may depend on the mean value of the generated traffic. A process with mean $\mu = 0.5$ can be easily expected to have $\sigma \simeq 0.25$. But the same value of variance seems too high for a process with $\mu = 0.01$ since values lower than 0 will be cut.

To evaluate the reasonable values of target μ, σ^2 that can be reasonably reached with the algorithm traces with different μ,σ^2 are generated and the actual variance obtained after shaping is compared to the target variance. Figure 5 shows the absolute errors of variance obtained depending on μ and σ using different Hurst parameters. As can be observed, the bigger error appear when $H \simeq 0.5$. Therefore some limits can be established attending to this errors.

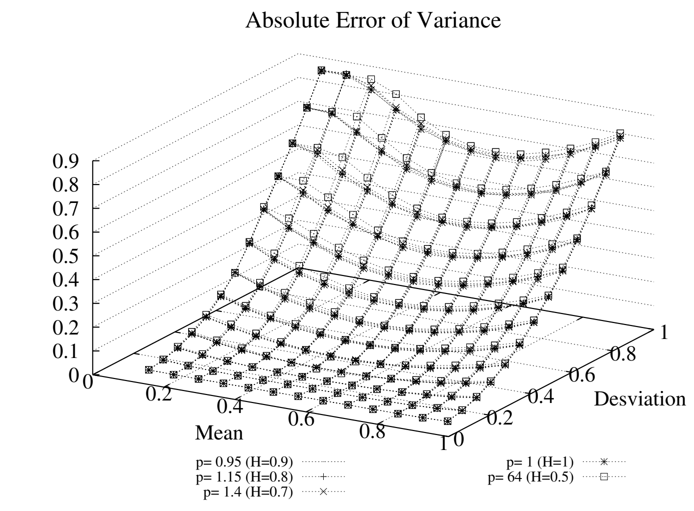

Fig. 5. Absolute error of Variance for traces with different persistences

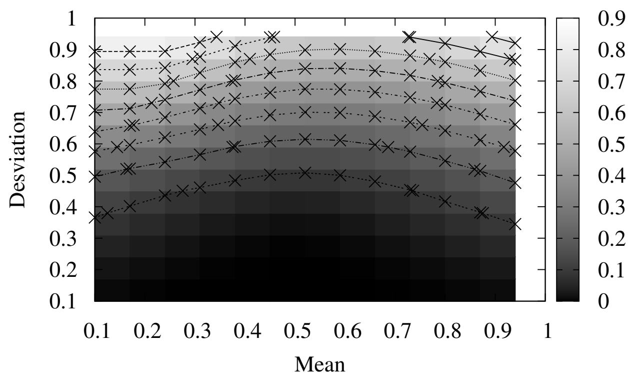

Fig. 6. Absolute error for Traces with H=0.5

Figure 6 shows (μ, σ) plane indicating the area where variance error is less than a given one i.e. 0.1 for $H = 0.5$. For the rest of $H > 0.5$ the variance error is even lower so the same area is safe to generate for every H value.

The zone of safe generation can be summarized as.

- $\mu \leq 0.15 \Rightarrow \sigma^2_{max} = 0.150 \quad (\sigma = 0.35)$
- $0.150 < \mu \leq 0.2 \Rightarrow \sigma^2_{max} = 0.16 \quad (\sigma = 0.4)$
- $0.2 < \mu \leq 0.7 \Rightarrow \sigma^2_{max} = 0.2025 \quad (\sigma = 0.45)$
- $0.7 < \mu \leq 0.9 \Rightarrow \sigma^2_{max} = 0.16 \quad (\sigma = 0.4)$
- $\mu > 0.9 \Rightarrow \sigma^2_{max} = 0.1225 \quad (\sigma = 0.35)$

V. Comparing RMD to Perlin Noise

In this section results for Perlin Noise-based generator are compared to those obtained using RMD algorithm [13]. This algorithm has been chosen since it is an approximate method known by his accuracy to obtain FGN traces in a efficient way. Maximum channel capacity is normalized to $C = 1$.

Table II show target values of μ, σ^2 and H for an example of comparison. These values are within safe zone as seen in previous section and generated trace obtain close actual values as seen on table. A slightly lower variance is expected because of limited channel capacity that reduce the range of possible values. H value is obtained with variance-aggregation estimator. For Perlin Noise generator the number of octaves used was $n = 6$ which gave a persistence value $p = 1.4$ to achieve a Hurst parameter $H \simeq 0.7$.

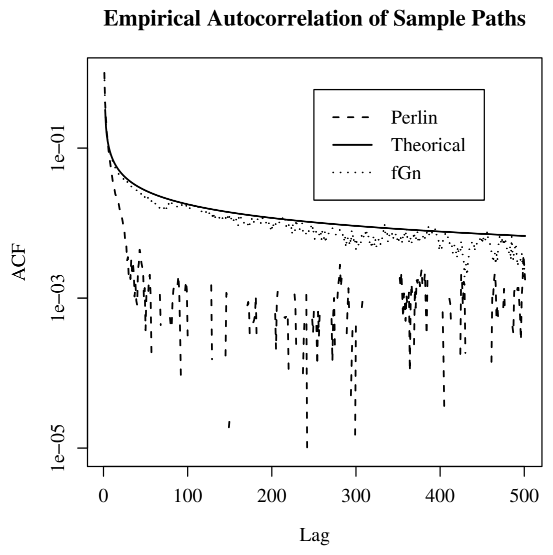

Fig. 7. Analytical Autocorrelation Curve with H=0.7

Both methods achieve accurate enough results to target values with relative errors below 3.5% as seen in the table III.

TABLE II
COMPARATIVE OF OBTAINED PARAMETERS THROUGH PERLIN NOISE AND RMD GENERATORS TAKING INTO ACCOUNT LIMITED CAPACITY $C = 1$

	Target	Perlin generated	RMD generated
μ	0.380	0.380	0.381
σ^2	0.096	0.0832	0.0927
H	0.7	0.732	0.724

TABLE III
ABSOLUTE ERRORS OF THE TRACES OBTAINED USING PERLIN NOISE AND FGN TAKING INTO ACCOUNT LIMITED CAPACITY

	Perlin Trace	FGN Trace
μ	0%	0.1%
σ^2	1.2%	0.33%
H	3.2%	2.4%

In order to check Perlin trace self-similar and long range dependence properties, its autocorrelation function and spectral density are compared to those of the RMD generated traces.

The autocorrelation of Perlin and RMD generated traces are plotted in figure 7. Theoretical autocorrelation for FGN is also plotted. It is given by $\rho(k) = 1/2[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}]$, [14]. Autocorrelation functions decay slowly with lag k showing $1/2 < H < 1$. Perlin trace deviate from that of RMD generated but its slope as $k \to \infty$ shows long range dependence.

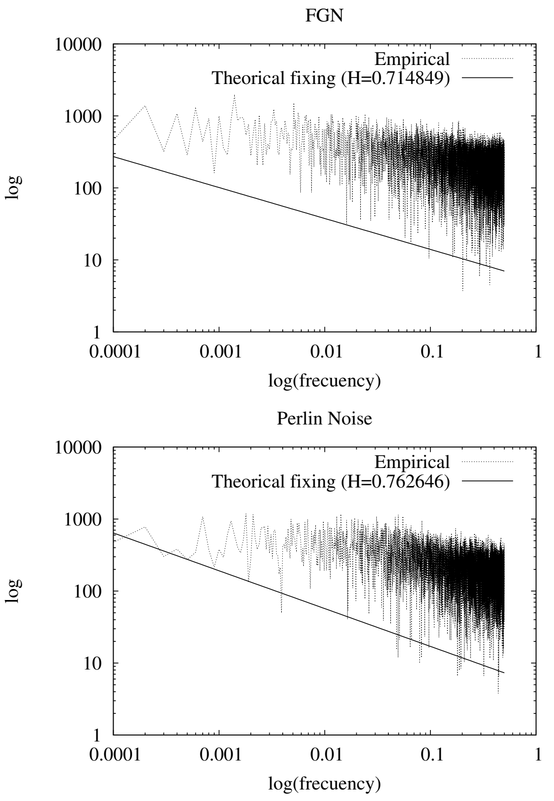

Fig. 8. Spectral density of FGN and Perlin traces with H=0.7 and approximation by straight lines

Spectral density for both traces is shown in figure 8. The best fit to the spectral density of a self-similar process [14] is also plotted showing similar H values to the ones obtained with variance-aggregation estimator. Thus spectral density of the trace generated with Perlin Noise algorithm is that of a self-similar process with the target H value.

Therefore the algorithm generate traces with correct parameters comparable to the ones generated by RMD and it can be operated to generate a continuous series of values without fixed size blocks.

VI. Conclusion

Self-similar traffic models are commonly used in order to feed simulations of high speed optical burst or packet switched networks with real-like traffic. Perhaps the most used of these is the Fractional Gaussian Noise (FGN) due to its simplicity. Several methods have been described in the literature to achieve a compromise between accurate and efficiency in the generation of FGN traces. However the main disadvantage of them is that trace length has to be decided in advance. In this paper it has been shown that an FGN generator can be built as an adaptation of the well known Perlin Noise process which has been often used in computer graphics. This adaptation has been reinterpreted as an FGN modeling network traffic in a fixed capacity channel.

The main advantage of this Perlin noise based generator is that it can generate traffic on-the-fly with no need to store the self-similar traffic trace in advance.

The accuracy of the generator has been compared to that of Random Midpoint Displacement algorithm, the classic FGN generator. RMD is known to be fast, simple and efficient. Perlin noise algorithm has been shown to obtain similar results generating random traffic with the same parameters than a RMD generator.

The long range dependency and spectral properties of Perlin generated traffic have been shown to be equivalent to those of a process generated with RMD and that they fit theoretical characteristics of FGN.

Therefore Perlin noise based traffic generator can be used as a long range dependent traffic source of FGN that can be generated continuously removing the need of large size trace generation before simulation.

References

[1] S. M. Klivansky, C. Song, and A. Mukherjee, "On long-range dependence in nsfnet traffic," 1994.

[2] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of ethernet traffic (extended version)," *Networking, IEEE/ACM Transactions on*, vol. 2, no. 1, pp. 1 –15, feb 1994.

[3] J. Beran, R. Sherman, M. Taqqu, and W. Willinger, "Long-range dependence in variable-bit-rate video traffic," *Communications, IEEE Transactions on*, vol. 43, no. 234, pp. 1566 –1579, feb/mar/apr 1995.

[4] V. Frost and B. Melamed, "Traffic modeling for telecommunications networks," *Communications Magazine, IEEE*, vol. 32, no. 3, pp. 70 – 81, march 1994.

[5] J. Gordon, "Long range correlation in multiplexed pareto traffic," in *Broadband Communications, 1996. Global Infrastructure for the Information Age. Proceedings of the International IFIP-IEEE Conference on*, 1996, pp. 28 –39.

[6] D. Heyman and T. Lakshman, "Source models for vbr broadcast-video traffic," *Networking, IEEE/ACM Transactions on*, vol. 4, no. 1, pp. 40 –48, feb 1996.

[7] P. Jelenkovic, A. Lazar, and N. Semret, "The effect of multiple time scales and subexponentiality in mpeg video streams on queueing behavior," *Selected Areas in Communications, IEEE Journal on*, vol. 15, no. 6, pp. 1052 –1071, aug 1997.

[8] R. Narasimha and R. Rao, "Modeling variable bit rate video on wired and wireless networks using discrete-time self-similar systems," in *Personal Wireless Communications, 2002 IEEE International Conference on*, dec. 2002, pp. 290 – 294.

[9] C. Huang, M. Devetsikiotis, I. Lambadaris, and A. Kaye, "Fast simulation for self-similar traffic in atm networks," in *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on*, vol. 1, jun 1995, pp. 438 –444 vol.1.

[10] D. Ostry, "Synthesis of accurate fractional gaussian noise by filtering," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1609 –1623, april 2006.

[11] E. Perrin, R. Harba, R. Jennane, and I. Iribarren, "Fast and exact synthesis for 1-d fractional brownian motion and fractional gaussian noises," *Signal Processing Letters, IEEE*, vol. 9, no. 11, pp. 382 –384, nov. 2002.

[12] T. Taralp, M. Devetsikiotis, and I. Lambadaris, "Efficient fractional gaussian noise generation using the spatial renewal process," in *Communications, 1998. ICC 98. Conference Record.1998 IEEE International Conference on*, vol. 3, jun 1998, pp. 1456 –1460 vol.3.

[13] W.-C. Lau, A. Erramilli, J. Wang, and W. Willinger, "Self-similar traffic generation: the random midpoint displacement algorithm and its properties," in *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on*, vol. 1, jun 1995, pp. 466 –472 vol.1.

[14] J. Beran, *Statistics for long-memory processes*, ser. Monographs on statistics and applied probability, 61. Chapman & Hall, Oct. 1994.