

Efficient Real-Time Traffic Generation for 5G RAN

Diarmuid Corcoran

Ericsson AB and Software and Computer Systems

KTH Royal Institute of Technology

Stockholm, Sweden

diarmuid.corcoran@ericsson.com

Per Kreuger

RISE AI

Research Institutes of Sweden

Kista, Sweden

per.kreuger@ri.se

Christian Schulte

Software and Computer Systems

KTH Royal Institute of Technology

Stockholm, Sweden

cschulte@kth.se

Abstract—Modern telecommunication and mobile networks are increasingly complex from a resource management perspective, with diverse combinations of software and infrastructure elements that need to be configured and tuned for efficient operation with high quality of service. Increased real-time automation at all levels and time-frames is a critical tool in controlling this complexity. A key component in automation is practical and accurate simulation methods that can be used in live traffic scenarios. This paper introduces a new method with supporting algorithms for sampling key parameters from live or recorded traffic which can be used to generate large volumes of synthetic traffic with very similar rate distributions and temporal characteristics. Multiple spatial renewal processes are used to generate fractional Gaussian noise, which is scaled and transformed into a log-normal rate distribution with discrete arrival events, fitted to the properties observed in given recorded traces. This approach works well for modelling large user aggregates but is especially useful for medium sized and relatively small aggregates, where existing methods struggle to reproduce the most important properties of recorded traces. The technique is demonstrated through experimental comparisons with data collected from an operational LTE network to be highly useful in supporting self-learning and automation algorithms which can ultimately reduce complexity, increase energy efficiency, and reduce total network operation costs.

Index Terms—Communication system traffic, Fractals, Machine learning, Parametric statistics, System simulation

I. INTRODUCTION

Generation of synthetic traffic data has many applications, but the key motivation for this work is the need to enable real-time and in-loop simulation to facilitate the reinforcement learning (RL) [1] branch of machine learning for resource optimisation in LTE [2] and 5G Radio Access Networks (RAN) [3].

Recent results [4], [5] in game- and simulation-based system have shown ground-breaking results in learning complex tasks with no or minimal human intervention. A key component in these learning systems is the use of simulation-based exploration of new and better action alternatives. In an operational RAN the opportunities for exploration through on-line experimentation are limited, as this would, in many cases, be far too invasive. In addition, the data samples collected from the system may be too sparse, collected at low temporal resolution and be difficult to export for privacy reasons. Part of our solution to these challenges is a novel approach augmenting the RAN system with traffic simulation capabilities that can be

used by machine learning algorithms to complement recorded data samples.

The principle of operation is to collect traffic samples from a running system for the purpose of parameter estimation (Section III) and use these parameters to generate similar synthetic traffic data. This data can then be used in simulations to reason about various kinds of RAN resource optimisation problems, such as RRM (Radio Resource Management) or deployment of workloads onto cloud-based compute infrastructure for the purpose of minimising latency or energy consumption.

Most traffic generators are able to sample arrival rates from a parametric distribution fitted to recorded data, but not to *simultaneously* reproduce the temporal structure of the recording, i.e. the auto-correlation as expressed in e.g. the “burstiness” of the traffic. In contrast, the method described in this paper is suitable to reproduce *both* arrival rate and packet size distribution as well as the most important time domain property, the autocorrelation structure, even of reasonably small aggregates. Small to medium sized aggregates are of special importance for simulation of traffic close to the edge of the network, e.g. for simulating the highly variable demand for a single cellular network node, or its back haul link.

Recorded LTE traffic auto-correlation exhibits long term dependency (LTD) which implies that sample values are correlated over several seconds or longer (see III). It has also been shown [6] that Ethernet and LAN traffic rate time-series exhibit a self-similar structure. This suggests using fractals [7], [8] to model the LTD of observed traffic. A well known and well studied method to generate fractals suitable for traffic modelling involves Fractional Gaussian Noise (FGN) [9], [10]. One efficient way to generate (approximations of) FGN is to use several overlapping spatial renewal processes (SRP) [11], which we exploit in this paper.

Paper Overview: The concept of self-similarity and its application to network traffic is reviewed in Section II, where we also explain the paper’s contributions in detail. Section III introduces the proposed method and includes a detailed description of the algorithms used. In Section IV we discuss empirical results and compare traffic from an LTE network with generated synthetic traffic. Section V concludes the paper with a discussion on the applicability of this work and possible future extensions.

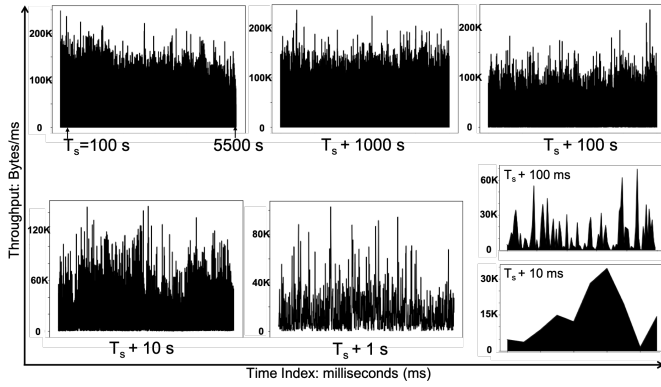


Fig. 1. Self-Similar 848 user aggregate

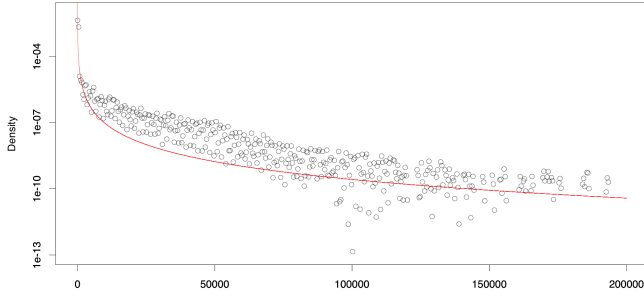


Fig. 2. Log normal fit of B/s rate distribution. Recorded 128 user aggregate.

II. BACKGROUND AND RELATED WORK

Very simply, self-similarity implies a form of scale invariance, i.e. that a concept or object appears the same regardless of scale. Many concepts in nature exhibit this kind of property, including network traffic in Internet and cellular systems. Seminal work by Leland et al. [6], [12] explores the self-similar nature of Ethernet LAN traffic and concludes that there is evidence for this phenomenon in network traffic. Figure 1 shows rate samples of LTE traffic in a large mobile network for 848 randomly selected active users over a period of 90 minutes. It shows that traffic patterns in modern day mobile networks are indeed similar to those studied by Leland et al., where at increasing time resolution, patterns appear to remain invariant. Peaks start to thin out at the maximum time resolution of the collected data. Aggregating upwards from there shows the same effect as in Leland’s work.

The distribution of rates is another important property of network traffic, and is for small to medium sized aggregates often very far from Poisson or Gaussian in character. Figure 2 shows a log-normal fit to the empirical rate distribution of a medium sized aggregate. Although the fit is not perfect, it is also clear that the rates are neither normally nor exponentially distributed. Pareto, or binomial models may provide fits with similar accuracy, but the log normal model has the advantage of simplicity and, in our experience [13], often fits this type of data very well.

In modelling network traffic, self-similarity refers to the

properties of a rate time-series and manifests primarily through long-tailed auto correlation functions, indicating LTD. I.e. with a memory in the generating process captured by statistical dependence between observations separated by time intervals up to at least several seconds. LTD implies a statistical autocorrelation function (ACF) that decays at a hyperbolic rate, i.e. according to a power law function, for example of the form $f(x) = a/x^k$ (see Figure 4c for an example).

There is also a precise mathematical definition of self-similarity for network modelling. The details of this have been presented in many papers [7], [8], [10], [12], [14] so here we offer only a brief overview to provide context for later sections.

Assume X_t , $t \in \mathbb{N}$ is a covariance stationary time-series and $X_t^{(m)}$ is a new time-series obtained by averaging the original X_t in non-overlapping sub-blocks of size m with mean $\mu = E[X_t]$, variance $\sigma^2 = E[(X_t - \mu)^2]$, and covariance $C(X_t, X_{t+k}) = E[(X_t - \mu)(X_{t+k} - \mu)]$. The normalised ACF is defined as $r(k) = C(X_t, X_{t+k})/\sigma^2$. Given these properties, the time-series X_t and $X_t^{(m)}$ are considered *first order self-similar* if they have the same distribution (μ, σ^2) . They are *exactly second order self-similar* if X_t and $X_t^{(m)}$ have the same autocorrelation structure, that is $r(k) = r^m(k)$. Finally, the time-series is *asymptotically second order self-similar* when $r(k) \sim r^m(k)$ as $m \rightarrow \infty$.

One way to quantify the long-term memory of a time-series is through the Hurst index H ($0 < H < 1$). H captures the degree of self-similarity through the autocorrelation of a time-series and rate at which it decreases as the lag between time samples increases. For $0 < H < 0.5$ the series is anti-persistent and there is no auto-correlation. At $H = 0.5$ the time series exhibits random behaviour and for $0.5 < H < 1$ the series is autocorrelated with a degree of persistent memory determined by H . In modelling and simulating network traffic, H measures the degree of burstiness and overall traffic intensity. There are a number of practical techniques to measuring H in time-series, see [15] for a detailed analysis in relation to network simulation.

There exist many models for exact generation of network traffic based on self-similar models. Most studied among these are Fractional Gaussian Noise (FGN) [10], [16], [17] and Fast Autoregressive Integrated Moving Average (FARIMA) [18] models. Unfortunately, generating exact self-similar time-series is computationally expensive [11], [15], [19], [20], so several approximate models have been developed. Examples of these methods are: fast Fourier transform (FFT), chaotic maps, random midpoint displacement (RMD), successive random addition (SRA), mixed queuing processes, wavelet transformations, and fast Gaussian noise generation (see [7], [21] for an extensive overview and bibliography of these methods).

A FGN model is a stationary Gaussian process with marginal distribution $X \sim \mathcal{N}(0, 1)$ and exactly second order self-similar sub-exponential autocorrelation coefficient function given by:

$$\rho(k) = \frac{1}{2}(|k+1|^{2H} - |2k|^{2H} + |k-1|^{2H}) \quad (1)$$

for $k \geq 1$ where H is the Hurst index.

Paxson [19] proposes a method to generate self-similar sequences using spectral analysis of true FGN and reverse FFT on the obtained parameters. This can produce good and computationally efficient FGN approximations. Unfortunately Paxson’s approach has limited application to our particular problem domain, the generation of network traffic, when we wish to control the number of processes contributing to the traffic aggregate. As such we require a method that gives us a direct and measurable parametric control of the individual contributions to the traffic aggregate. A proposal by Taralp et al. [11] using several parallel spatial renewal processes (SRP) [22] which are convolved to generate approximate FGN sequences offers just such an alternative.

A (single) SRP is described by two parallel and individually i.i.d. point processes $\{T_t\}$ and $\{X_t\}$. $\{T_t\}$ is the temporal and LTD component with renewal times characterised by a heavy-tailed function $F_T(t)$. $\{X_t\}$ is the desired steady state marginal distribution which in the case of FGN must be $\mathcal{N}(0, 1)$. An SRP is thus composed of a chain of renewal periods $\{Y_t\}$ where the n -th period is T_n in duration and has magnitude X_n . $F_T(t)$ is given by Equation 2 which is, in turn, derived from Equation 1. See [11], [22] for a complete derivation.

$$F_T(t) = 1 - H \frac{(t+1)^a - t^a + (t-1)^a}{2^a - 2} \quad (2)$$

for $t \geq 1$ where $a = 2H - 1$.

Taralp [11] uses several convolved SRP processes $\{Y_t\}$ to create FGN approximations. For the purpose of synthesising traffic with many complex LTD dependencies using up to M parallel SRPs, M turns out to be a powerful additional parameter, even though it does not directly map to the number of user processes in recorded traffic. Typically we have seen $M=\{1, \dots, 16\}$ as effective settings.

In this paper we modify the algorithm of [11] to use a discrete time rejection sampling approach to compute the autocorrelation coefficient, rather than a piece-wise linear approximation of a continuous time function. This gives us a more precise and computationally very efficient algorithm for generation of FGN using overlapping SRP’s as a starting point for generating synthetic LTD traffic (section III).

To our knowledge, none of the methods used to produce LTD time series have the ability to *also* ensure that the rate distributions follow those observed in recorded data. We therefore propose to extend Taralp’s approach, as summarised above, and make the following novel contributions:

- 1) A method to generate scaling parameter values used to synthesise accurate¹ cellular or network traffic. This is generally applicable for use in data-driven machine learning algorithms. See Figure 3.
- 2) An online variant of the algorithm that can be used to generate samples on demand, with a very compact process state representation.
- 3) A very fast batch algorithm based on Equation 2, that can generate hours of synthetic traffic in seconds.

¹in terms of rate distribution similarity and burstiness

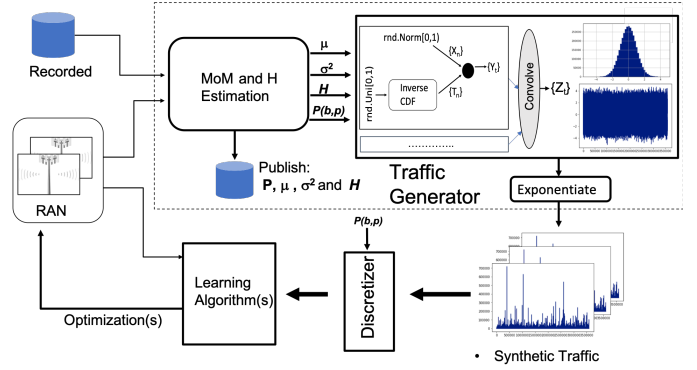


Fig. 3. Method Overview

- 4) Empirical results that show that our synthetic traffic matches data from an LTE network well, according to our similarity criteria.
- 5) Additional empirical data to verify that Leland et al. observations on self-similarity of LAN traffic applies also to modern mobile data networks.

III. METHOD

A. Overview

Studies of recorded arrival events from live LTE installations indicate that not only are the time series highly self similar [6], [12] (i.e. features long range dependency), but also that the rate distributions measured over a large range of time scales and all but the largest aggregates have heavy tails [13], [23], [24]. In order to replicate this type of behaviour we need to sample rates from heavy-tailed distributions and ensure that consecutive samples have the required long term dependency. Furthermore, if we need to reproduce the traffic in small aggregates and/or at high temporal resolution, which typically contain long stretches of zero arrivals, interspersed with the typical high rate bursts, we either need to identify a discrete distribution with the required properties, or find a way to discretise an otherwise suitable continuous distribution.

One straightforward and direct method to achieve this end, is to exponentiate generated and scaled fractal Gaussian noise, and then discretise the resulting continuous rate samples in some way. We propose, describe and evaluate a method that can be outlined as follows (also captured in fig 3):

- 1) Generate Fractional Gaussian Noise (FGN) using a small number M of overlapping Spatial Renewal Processes (SRP) according to a variant of the method in [11].
- 2) Scale the obtained noise to have mean μ and standard deviation σ corresponding to the log of those observed in recorded traces.
- 3) Exponentiate the FGN to obtain log-normally distributed rate samples.
- 4) Sample arrival packet sizes from the observed empirical distribution $P(b, p)$ and aggregate the obtained (continuous) rates into discrete arrival events.

The advantages of this method is that it is easy to extract approximate scaling parameters using method-of-moments

(MoM)² estimation from recorded traces, and that exponentiation and discretisation preserves most of the long term dependencies (LTD) and the auto correlation structure (ACF) from the FGN, at least for traffic aggregates covering more than few users. The loss of autocorrelation (from exponentiation and discretisation) as measured by the Hurst index is modest in most practical cases. Figure 4 plots the sample path (4a), probability density function (4b) and auto correlation function (4c) for the raw FGN generated by the SRPs, and the corresponding properties of the exponentiated noise (4d-4f).

B. Algorithm

Since we use a modified version of Taralp's method, we will present it in some detail, and in the context of a specific method to scale and exponentiate the obtained noise into viable high temporal resolution rate samples, which are then aggregated and discretised into packet arrivals.

The version of the method presented in Algorithm 1 is suitable for generating consecutive samples of FGN on demand and one at the time. It uses a discrete time version of the auto-correlation co-efficient function, rather than (as in Taralp's case), a piece-wise linear approximation of a continuous one. This implementation is based on rejection sampling, which amounts to a linear search for a coefficient when sampling new inter-renewal times. This may seem inefficient, but it is exact up to the discretisation of the time dimension, and in practice works very well, due to the relative rareness of very large coefficients. Irregardless of the method to generate inter-renewal times, for large sample batches even more efficient methods can be devised and Algorithm 4, is an example of a very compute efficient solution.

Algorithm 1 maintains a record of the state of each contributing renewal processes that amounts to the current rate X_m , and next renewal time R_m for each $m \in M$. In addition it keeps track of the "current" time t , and the latest generated FGN sample Y . An initial state can be obtained by a call to `INITSTATE()`, and samples of the FGN extracted from the states produced by consecutive calls to `EVOLVE()`.

At each invocation, the current time t is incremented, then each process is checked to see if it should be renewed. On renewal we update its rate with a value sampled from a Gaussian distribution, and a new renewal time produced by adding the previous renewal time to the result of a call to `DT()`, which implements the rejection sampling of the autocorrelation coefficient function `FT()`. Once this process is complete for all contributing processes, an FGN sample y is produced by adding current rates of the M processes, and normalising it to ensure that the resulting noise samples will have variance of 1. The sample is stored in the state variable Y from where it can be extracted from the state returned by each call to the `EVOLVE()` function.

Algorithm 1 A discrete implementation of the spatial renewal process algorithm as described in [11]. State representation, sampling procedures and momentary evolution.

```

function INITSTATE( $H, M$ )                                ▷ Initialise SRP state
     $srp \leftarrow \{H : H, M : M, t : 0,$                     ▷ SRP parameters
                     $X : array(M),$                         ▷ Component process states
                     $R : array(M),$                         ▷ and renewal times
                     $Y : 0\}$                                 ▷ Storage for latest sample
    for  $m \leftarrow 1, M$  do
         $srp[X][m] \leftarrow rnd.Normal()$ 
         $srp[R][m] \leftarrow dT(rnd.uniform(), H)$ 
    end for
    return  $srp$                                             ▷ Return the state
end function

function EVOLVE( $srp$ )                                    ▷ Evolve process state
     $srp[t] \leftarrow srp[t] + 1$                             ▷ Increment time
     $y \leftarrow 0$                                           ▷ Initialise new sample
    for  $m \leftarrow 1, srp[M]$  do                          ▷ For each process  $m$ 
        if  $srp[t] > srp[R][m]$  then                        ▷ If deadline passed
             $srp[X][m] \leftarrow rnd.Normal()$              ▷ Renew process
             $srp[R][m] \leftarrow srp[R][m] + DT(rnd.unif())$ 
        end if
         $y \leftarrow y + srp[X][m]$                           ▷ Convolve into sample  $y$ 
    end for
     $srp[Y] \leftarrow y/\sqrt{M}$                             ▷ Store normalised sample
    return  $srp$                                             ▷ Return updated process state
end function

function DT( $U, H$ )                                        ▷ Sample renewal time  $\delta$ 
     $t \leftarrow 1$ 
    while  $FT(t, H) < U$  do                                ▷ by rejection sampling
         $t \leftarrow t + 1$ 
    end while                                              ▷ Linear search
    return  $t$ 
end function

function FT( $t, H$ )                                        ▷ Discrete autocorrelation
     $a \leftarrow 2H - 1$                                     ▷ co-efficient function
    return  $1 - H \frac{(t+1)^a - t^a + (t-1)^a}{2^a - 2}$ 
end function

```

Algorithm 2 Scaling and exponentiation of fractal Gaussian noise samples into continuous rate samples

```

function RATE( $\mu, \sigma, srp$ )
     $srp \leftarrow EVOLVE(srp)$ 
     $srp[Y] \leftarrow e^{\mu + \sigma * srp[Y]}$ 
    return  $srp$ 
end function

```

Algorithm 2 scales the the samples produced by the SRP procedure to give a log-normal response with parameters μ and σ . These are in turn derived by means of MoM estimation:

$$\begin{cases} \hat{\sigma}^2 = \ln(1 + \hat{v}c^2) \\ \hat{\mu} = \ln \hat{m} - \frac{1}{2}\hat{\sigma}^2 \end{cases} \quad (3)$$

²MoM also works when the recorded data contains zeros, and produces continuous models suitable to subsequent discretisation.

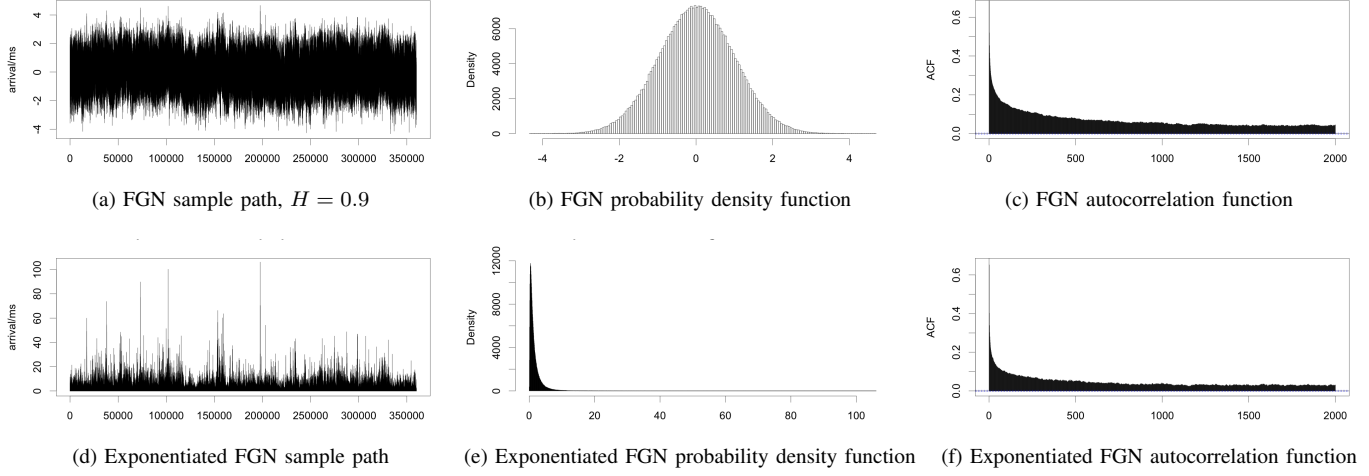


Fig. 4. Unscaled raw (top) and exponentiated (bottom) FGN sample path, PDF and ACF plots. The exponentiated sample path is interpreted as momentary arrival rates.

where $\hat{m}$ and $\hat{v}$ are the sample mean and variance coefficient respectively of the recorded data we would like to emulate.

Algorithm 3 shows how we generate N consecutive rate samples by repeatedly calling `EVOLVE()`, on the previous state representation srp , starting from a single call to `INITSTATE()`, and in addition, uses a packet size distribution to discretise the continuous rate samples into discrete arrivals. This is very useful for generating sparse arrival processes with high time resolution where the continuous rate, generated by the rate process, may very well be far below 1 bit per time frame.

Algorithm 3 On-line generation of packet arrival events from continuous rate samples

```

function ARRIVALS( $\hat{m}, \hat{v}, N, H, M, pkSzDstr$ )
   $\sigma^2 \leftarrow \log(1 + \hat{v}^2)$ ;  $\mu \leftarrow \log \hat{m} - \sigma^2/2$ 
   $srp \leftarrow \text{INITSTATE}(H, M)$ 
   $pZ \leftarrow pkSzDstr()$   $\triangleright$  Sample packet size distribution
   $acc \leftarrow 0$ ;  $arr \leftarrow []$   $\triangleright$  Accumulators
  for  $n \leftarrow 1, N$  do
     $srp \leftarrow \text{RATE}(\mu, \sqrt{\sigma^2}, srp)$ 
     $acc \leftarrow acc + srp[Y]$   $\triangleright$  Add sample to accumulator
    while  $acc > pZ$  do  $\triangleright acc > \text{expected packet size}$ 
       $arr \leftarrow \text{append}(arr, [\{srp[t], pZ\}])$   $\triangleright$  Arrival
       $acc \leftarrow acc - pZ$   $\triangleright$  Decrement accumulator
       $pZ \leftarrow pkSzDstr()$   $\triangleright$  Re-sample pkt size
    end while
  end for
  return  $arr$ 
end function

```

Algorithm 4 is a vectorised alternative to algorithm 1 and can produce a batch sample path of SRP generated FGN with given H and then scaled, exponentiated and discretise as described previously. A table of starting t values as required by rejection sampling linear search is built in advance for selected r cutoffs, and then used to accelerate the starting value of t

given a uniformly random r (see lines 2,12 and 13). It also uses a slightly altered variant of `DT()` where t can be selected to support a fast linear search. This is a valuable optimisation given the nature of F_T 's CDF where most r lead to a t of between two and ten, depending on H . Another optimisation is to pre-initialise a matrix of $n \times M$ from $\mathcal{N}(0, 1)$ to avoid this in a loop body (line 3). The implementation of `vSRP()` uses Python's NumPy [25] and Numba [26] vectorisation support and Table I shows key aggregated statistics from generating 100 execution runs of one hour of sample path data at ms granularity for different values of M and H . It can generate a 1 hour ms sample path in a couple of seconds depending on M and H , for example $M = 5$ and $H = 0.75$ is in the range of 3 seconds.

TABLE I
100 BATCHES OF 1 HOUR SYNTHETIC SAMPLES AT ms RESOLUTION

	M	2	5	10	20
$H=0.75$	Std	1.0005	0.9997	0.9993	0.9991
	Mean	0.0030	0.00057	0.0037	-0.0010
	Time (s)	131.5562	320.7621	599.4399	1153.15
	H-SRP mean	0.7404	0.7402	0.7406	0.7405
$H=0.95$	Std	1.0005	0.9960	0.9988	1.0030
	Mean	0.0092	0.0156	-0.0233	0.0037
	Time (s)	164.9317	358.9244	659.7713	1275.6013
	H-SRP mean	0.9323	0.9323	0.9324	0.9325

A rudimentary comparison against Algorithm 1 shows `vSRP()` to be 10-15 times faster in terms of execution time (Xeon E3-1245, 3.5Ghz, 32G RAM). Other key metrics such as mean FGN close to 0 and standard deviation close to 1 in addition to mean H of the generated sample path (H-SRP) for specified H (i.e 0.75) show that the method is stable. A detailed analysis of the mean and worst case execution time of Algorithm 4 was also carried out but not presented here due to space limitations. In summary the mean execution complexity is determined by $n \cdot M \cdot \varrho(F_T H)$, where n is the number of samples and $\varrho(F_T H)$ is the mean number of calls to F_T

Algorithm 4 A fast vectorized and discrete implementation of the spatial renewal process

```

1: function vSRP( $n, H, M$ )
2:    $tableT \leftarrow CDF(H)$   $\triangleright$  Build threshold table for  $t$ 
3:    $X \leftarrow rnd.Normal(n, M)$   $\triangleright$  Pre-initialize  $X$ 
4:    $K \leftarrow zeros(M)$ 
5:   for  $m \leftarrow 1, M$  do
6:      $K[m] \leftarrow K[m] + DT(rnd.uniform(), H)$ 
7:   end for
8:   for  $i \leftarrow 1, n$  do  $\triangleright X$  is  $n \times M$  matrix
9:     for  $m \leftarrow 1, M$  do
10:      if  $K[m] \leq i$  then  $\triangleright$  Deadline passed
11:         $r \leftarrow rnd.uniform()$  ;  $t \leftarrow 1$ 
12:        if  $r > THRESHOLD(tableT)$  then
13:           $t \leftarrow tableT[r]$   $\triangleright$  accelerate  $t$  search
14:        end if
15:         $K[m] \leftarrow K[m] + DT_t(r, H, t)$   $\triangleright$  Update
16:      else
17:         $X[i][m] \leftarrow X[i-1][m]$   $\triangleright$  Retain previous
18:      end if
19:    end for
20:  end for
21:  return  $\frac{X.sum(axis=m)}{\sqrt{M}}$   $\triangleright$  normalised samples
22: end function

```

given H (in range 2-10). The worst case situation is more complex, especially with $H > 0.9$ where calls to F_T can be in the order of millions for a very small minority of r . This can be mitigated by the table look-up presented in our algorithm and also by determining a maximum bound on the number of iterations for a given sample path length. A full performance comparison between FGN generation methods is not within scope for this paper but a variant of Paxson's [19] FFT was also implemented using Numpy and Numba and a basic performance comparison shows Paxson's algorithm can also generate a batch sample of 1 hour in 2-3 seconds. Accuracy of the algorithm is similar to the SRP variant but without the advantage of having M parallel processes to tune traffic generation as discussed in section IV.

IV. EMPIRICAL RESULTS

In order to evaluate that the proposed mechanism can indeed produce synthetic traffic which is both self-similar, and has a long tailed distribution with properties comparable to those seen in recorded traces, we have studied the properties of various recorded aggregates, and compared these with generated samples. All of the examples reported on here are random selections of different sizes from a 529 user subset of an even larger set from a major urban environment. The data contains packet and byte arrival rates at 1ms resolution over a four hour peak traffic period. Each studied data set was chosen among several randomly selected candidates (of the same size) to represent typical Hurst index values, rate mean

TABLE II
STATISTICS OF RECORDED AND GENERATED BYTE AND PACKET ARRIVAL RATES, 128 USERS

Byte arrivals/ms	Recorded	Generated
Mean	79.038	65.206
Variation coefficient	17.479	13.622
Zero ratio	97.102 %	83.408 %
99th %-ile	466.00	1440.0
Hurst index	0.76153	0.76315
ACF MAE, 2 s	0.02116	

Packet arrivals/ms	Recorded	Generated
Mean	0.082135	0.032367
Variation coefficient	12.235	3.5963
Zero ratio	97.102 %	83.408 %
99th %-ile	1	3
Hurst index	0.75382	0.71824
ACF MAE, 2 s	0.00785	

and variance. None of the selected sets show signs of heavy congestion.³

We compare moments, key quantiles, and Hurst index of the recorded and generated data, and also compare the rate distributions and auto correlation functions of the generated data against the recorded ones. For estimating the H index we used the R `hurstSpec` [27] function which produces its estimates by linear regression of the log(spectrum) versus log(frequency) with frequency points accumulated into boxes of equal width on a logarithmic scale and spectrum values averaged over each box. Unless explicitly stated otherwise, we have used $M = 16$ and $H = 0.96$ in these experiments. The ACF mean absolute error (MAE) is a quantitative measure of the difference between the recorded and generated ACFs.

A. 128 user aggregate

Figure 5 shows the sample paths, (log-log) rate distribution and auto correlation function of 128 recorded (5a-5c) and generated (5d-5f) traces. We see a similar "burstiness" of the two sample paths, with a slightly higher occurrence of very high rates in the generated trace, and what appears to be a truncation of the rate distribution for the recorded trace. The auto-correlation functions of the generated trace is overall slightly lower than that of the recorded one, despite the fact that we used a input Hurst index of .96 to compensate for down scaling of the auto correlation of the raw FGN caused by exponentiation and discretisation. The ACF MAE is otherwise very good in this case.

Looking at Table II we see the corresponding statistics for both byte and packet rates. It turns out that it is difficult to reproduce the very high variation coefficients observed in the recorded traces. This is also reflected in the slightly lower zero ratios, and larger high percentiles of the generated traces.

Overall this result is still very encouraging and we conclude that at least for this type of traffic aggregate, the proposed

³We have refrained from including examples of aggregates with more than 128 users, since for these, the rate distributions become more Gaussian due to mixing of flows and the law of large numbers. The novelty of our approach is the ability to reproduce the properties of medium to small aggregates.

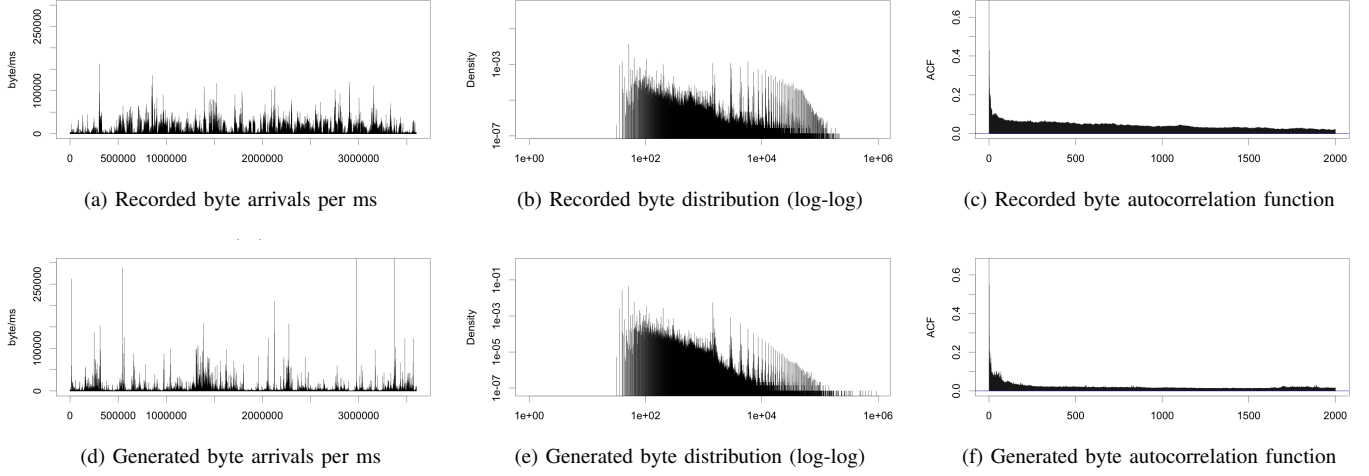


Fig. 5. Comparison between recorded (top) and generated (bottom) byte arrival traces, 128 user aggregates

method can reproduce the most important characteristics of observed traces. We note however that generated traces with the same input parameters can vary somewhat between runs, especially for smaller sample sets. This is likely also an effect of the high variance requirement. Note that the mean and variation coefficient listed for the byte rates can be used to directly reproduce these results.

For larger UE aggregates, the zero-ratio, Hurst index and variation coefficient typically goes down, and it thus becomes easier to reproduce the properties of observed traces. The same would hold for longer recorded traces as well, provided that their properties remain roughly stationary. That is rarely the case however, and for different times of day, and different weekly patterns, separate parameters should probably be used. The method can easily be adapted to gradual changes of the input parameters. For smaller aggregates on the other hand, the variation coefficient, and zero ratio increases. The following two sections will explore two such cases.

B. 16 users aggregate

Figure 6 and Table III show the corresponding plots and statistics for a 16 UE aggregate from the same data set. First of all, we note that both the variation coefficient, and the zero rate goes up significantly in the recorded traces, and that our method struggles with reproducing these. We note also that the autocorrelation of this particular recorded trace is comparatively low, and that our method have no problem in reproducing that (using the same .96 input Hurst index). Inspecting the log-log plots of the rate distributions, it is also clear that the recorded distribution is no longer clearly uni-modal, and probably consists of a number of users (or time periods) with higher rates than the rest, again driving up the variance. The best our method can achieve in this type of situation is an approximation, which it manages quite reasonably in this case.

TABLE III
STATISTICS OF RECORDED AND GENERATED BYTE AND PACKET ARRIVAL RATES, 16 USERS

Byte arrivals/ms	Recorded	Generated
Mean	6.3522	4.0938
Variation coefficient	56.592	23.413
Zero ratio	99.675 %	97.160 %
99th %-ile	0	58.000
Maximum	98 180	34 602
Hurst index	0.65834	0.67300
ACF MAE, 2 s	0.00361	
Packet arrivals/ms	Recorded	Generated
Mean	0.007446	0.094990
Variation coefficient	35.680	6.5228
Zero ratio	99.675 %	97.160 %
99th %-ile	0	1
Maximum	68.181	24.800
Hurst index	0.65344	0.64795
ACF MAE, 2 s	0.02029	

C. Smaller aggregates

For even smaller aggregates, the ACF of the recorded traces becomes more diverse, often with clear peaks exposing regular events in the data streams due to e.g. protocol time-outs, polling and user client behaviours. Very long completely silent periods also become common. Since we do not model repeating events explicitly, we cannot expect to reproduce these properties in a recorded data set exactly.

However, here is where we believe M , the number of contributing SRPs can come into play. When using low values of M we come across examples that do have this type of noisy ACF structure, especially when also using high input H values. We currently have no method to reproduce a particular type of structure of this type, but is clear that it does sometimes appear spontaneously as a combined effect of low M and mean, with high H and variance. A thorough study of the spread and distribution of such behaviours for a given set of input parameters remains as future work, but Figure 7 gives a

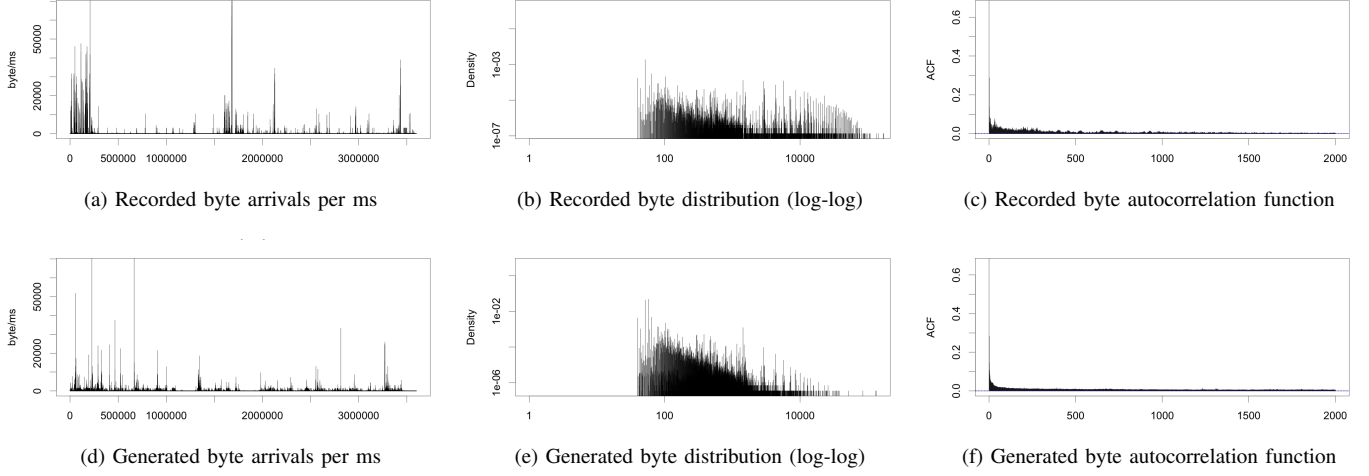


Fig. 6. Comparison between recorded (top) and generated (bottom) byte arrival traces, 16 user aggregates

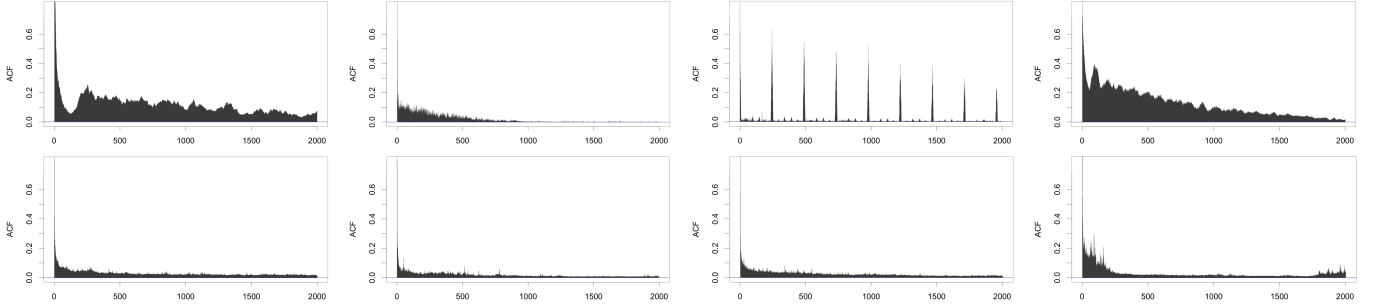


Fig. 7. A set of byte rate ACFs of randomly selected recorded 4 user aggregates (top) and corresponding generated ones with $M = 4$ and $H = 0.99$, (bottom).

small set of examples of the type of ACFs seen for recorded 4 user aggregate, and the corresponding ACF generated by the proposed method with $M = 4$ and $H = 0.99$. The diversity of the generated traffic is not as great as in the recorded cases, but at least has more apparently random elements than those with higher M s. The contribution of each involved SRP may also be separated, use different and dynamically varying H values and perhaps even be individually discretised, all of which would potentially allow for greater control over processes aimed at emulating smaller aggregates.

V. CONCLUSION

This paper describes a new and fast method to generate realistic network traffic data for simulations to aid and accelerate real-time resource management algorithms. The method is evaluated by analysing key statistics of recorded LTE traffic traces which shows that the method generates synthetic traffic data with very similar temporal and rate statistical properties. We believe this approach to be unique in facilitating traffic modelling of scenarios not currently possible through standard parameterised approaches. The contributions include the general method as outlined in Section III, an on-line discrete time version of the SRP method without the deficiencies observed for the approximations used by earlier

work [28]. A very fast vectorised version of the algorithm with performance comparable to or better (depending on number of parallel SRPs) than FFT based methods for producing FGN, is also introduced.

Currently we are working on a system implementation of our method at Ericsson, we also plan to release our reference code as open source. Potential future work include further investigation of useful synthetic series for user aggregates less than 16. Early experiments indicate that careful selection of M can be a viable route. Other areas of potential study are parameter selection to capture various types of traffic such as video or web page browsing, and methods to dynamically adopt the method to non-stationary processes. One area of huge potential is publication of parameter values. This could facilitate new kinds of research baselines for comparison in the networking community where publication of actual data sets can be very sensitive due to complex privacy concerns.

VI. ACKNOWLEDGEMENTS

We gratefully acknowledge the help from many colleagues at Ericsson for their valuable comments. Kreuger is funded by Ericsson. Corcoran and Schulte are partially funded by the WASP (Wallenberg Autonomous Systems and Software Program) research program.

REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
- [2] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (EUTRAN); Overall description; Stage 2," ETSI, Tech. Rep. ETSI TS 36.300 V12.5.0, Apr. 2015.
- [3] —, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; NR; NR and NG-RAN Overall Description; Stage 2 (Release 15)," 3GPP, Tech. Rep. 3GPP TS 38.300 V15.1.0, 2018.
- [4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.
- [5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, p. 354, oct 2017.
- [6] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic," in *Conference Proceedings on Communications Architectures, Protocols and Applications*, ser. SIGCOMM '93. New York, NY, USA: ACM, 1993, pp. 183–193. [Online]. Available: <http://doi.acm.org/10.1145/166237.166255>
- [7] F. I. Radu Dobrescu, *Large Scale Networks: Modeling and Simulation*. CRC Press, 2016, ch. chapter3. [Online]. Available: <https://www.routledgehandbooks.com/doi/10.1201/9781315368368-4>
- [8] B. Tsybakov and N. D. Georganas, "Self-similar processes in communications networks," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 1713–1725, Sep. 1998.
- [9] B. B. Mandelbrot and J. R. Wallis, "Computer experiments with fractional gaussian noises: Part 1, averages and variances," *Water Resources Research*, vol. 5, no. 1, pp. 228–241, 1969.
- [10] J. Beran, "Statistical methods for data with long-range dependence," *Statist. Sci.*, vol. 7, no. 4, pp. 404–416, 11 1992. [Online]. Available: <https://doi.org/10.1214/ss/1177011122>
- [11] T. Taralp, M. Devetsikiotis, and I. Lambadaris, "Efficient fractional gaussian noise generation using the spatial renewal process," in *1998 IEEE International Conference on Communications, ICC 98*, vol. 3. IEEE, 07 1998, pp. 1456 – 1460 vol.3.
- [12] K. Park and W. Willinger, *Self-Similar Network Traffic: An Overview*. John Wiley & Sons, Inc., 2002, ch. 1, pp. 1–38.
- [13] P. Kreuger and R. Steinert, "Scalable in-network rate monitoring," in *IFIP/IEEE Integrated Network Management — IM'15*, IFIP/IEEE. Ottawa, Canada.: IEEE, May 11–15 2015.
- [14] J. C. López-Ardao, C. López-García, A. Suárez-González, M. Fernández-Veiga, and R. Rodríguez-Rubio, "On the use of self-similar processes in network simulation," *ACM Trans. Model. Comput. Simul.*, vol. 10, no. 2, pp. 125–151, Apr. 2000. [Online]. Available: <http://doi.acm.org/10.1145/364996.365004>
- [15] H.-D. J. Jeong, "Modelling of self-similar teletraffic for simulation," PhD, Department of Computer Science and Management, University of Canterbury, Christchurch, New Zealand, 2002.
- [16] B. B. Mandelbrot, "A Fast Fractional Gaussian Noise Generator," *Water Resources Research*, vol. 7, pp. 543–553, Jun. 1971.
- [17] Changcheng Huang, M. Devetsikiotis, I. Lambadaris, and A. Kaye, "Fast simulation for self-similar traffic in ATM networks," in *Proceedings IEEE International Conference on Communications ICC '95*, vol. 1. IEEE, 1995, pp. 438–444. [Online]. Available: <http://ieeexplore.ieee.org/document/525208/>
- [18] Hosking, "Modeling Persistence In Hydrological Time Series," *Water Resources*, vol. 20, no. 12, pp. 1898–1908, 1984.
- [19] V. Paxson, "Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 5, pp. 5–18, Oct. 1997. [Online]. Available: <http://doi.acm.org/10.1145/269790.269792>
- [20] H.-D. J. Jeong, D. McNickle, and K. Pawlikowski, "A Comparative Study of Generators of Synthetic Self-Similar Teletraffic," Department of Computer Science and Management, University of Canterbury, Christchurch, New Zealand, Tech. Rep. TR-COSC 03/98, 1998.
- [21] A. E. Walter Willinger, Murad S. Taqqu and A. E. Walter Willinger, Murad S. Taqqu, "A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks," *Stochastic networks: ...*, pp. 339–366, 1996.
- [22] P. R. Jelenković, A. A. Lazar, and N. Semret, "Multiple Time Scales and Subexponentiality in MPEG Video Streams," in *International Conference on Broadband Communications BC 1996: Broadband Communications*, 1996, pp. 64–75.
- [23] A. B. Downey, "Lognormal and pareto distributions in the internet," *Computer Communications*, vol. 28, no. 7, pp. 790–801, 2005.
- [24] K. Fukuda, "Towards modeling of traffic demand of nodes in large scale network," in *Communications, 2008. ICC'08. IEEE International Conference on*. IEEE, 2008, pp. 214–218.
- [25] T. Oliphant, "NumPy: A guide to NumPy," USA: Trelgol Publishing, 2006–. [Online]. Available: <http://www.numpy.org/>
- [26] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, ser. LLVM '15. New York, NY, USA: ACM, 2015, pp. 7:1–7:6. [Online]. Available: <http://doi.acm.org/10.1145/2833157.2833162>
- [27] W. Constantine and D. Percival, "A fractal time series modeling and analysis package," CRAN R library, Accessed May 2019. [Online]. Available: {<https://CRAN.R-project.org/package=fractal>}
- [28] S. Ledesma and D. Liu, "A fast method for generating self-similar network traffic," *International Conference on Communication Technology Proceedings, ICCT*, vol. 1, no. June, pp. 54–61, 2000.