Internet Performance Modeling Using Mixture Dynamical System Models

Z. Liu, J. Almhana, V. Choulakian
University of Moncton
Moncton, New Brunswick
Canada E1A 3E9
{liuz,almhanaj,choulav}@umoncton.ca

R. McGorman
Nortel Networks
4001 E. Chapel Hill-Nelson Hwy
North Carolina USA 27709
mcgorman@nortelnetworks.com

Abstract

This paper models Internet traffic input stream and TCP connection durations using dynamical system models. A linear dynamical model with mixture Gaussian output is proposed for the Internet traffic input stream, and a linear dynamical system with mixture lognormal output is developed to model the TCP connection durations. In the proposed models, a sum of independent $AR(1)$ processes is used to approximate the autocorrelation of the real data, and a Gaussian mixture or lognormal mixture is used to fit the marginal distribution. As a result, the output processes can capture the correlation and the marginal distribution simultaneously. Making use of the fact that at each iteration the parameter increment of the EM algorithm has a positive projection on the gradient of the likelihood, a stochastic approximation-based recursive EM algorithm is proposed to fit the traffic marginal distribution. A cross-validation criterion is used for the model selection. To illustrate the usefulness of the proposed models, several experimental results are provided.

Key words: *Internet performance modeling, mixture dynamical system, EM algorithm, cross-validation.*

1 Introduction

In the area of high speed telecommunication network performance modeling, a surprising finding was that the traffic has long memory, namely, their autocorrelation functions decay hyperbolically rather than exponentially (see [13]). This means that the traditional teletraffic models, Poisson processes, are not suitable for high speed telecommunication networks modeling. In addition since long range dependent processes have infinite memory [2], model fitting becomes very hard. For this reason a lot of approximate models have been developed, including the Markov modulated Poisson process models (see [1, 20, 22] and the references therein) and the $M/G/\infty$ model [12]. For more information on high speed network traffic models, we refer the reader to [18].

In the network modeling community, researchers traditionally try to capture the mean, variance, the autocorrelation structure, and the general shape of the marginal distribution [15]. More recently, capturing the tail of the marginal distribution has received a lot of attention since it is of great importance for the queueing performance at a multiplexer (see [6, 9, 15]). This paper proposes dynamical system models that can approximate the autocorrelation and the marginal distribution of real traffic simultaneously.

As pointed out by [18], heavy-tailed distributions of some components in the networks are the internal cause of long memory. However, performance models with heavy-tailed distributions tend to be difficult to analyze. For this reason, Feldmann and Whitt [5] propose using mixtures of exponentials to approximate heavy-tailed distributions, and their experimental results show that mixtures of exponentials can approximate most heavy-tailed distributions with high accuracy. Furthermore, we plot several histograms of real traffic data, which show strong evidence of mixture distributions. For this reason, in the paper, we use Gaussian mixtures to model the marginal distribution of the Internet traffic input stream. Moreover, it is commonly believed that the web file size distributions have a lognormal distribution body and a Pareto tail. The two features are captured by mixture lognormal distribution (see e.g. [7, 10, 16, 19] for more information on file size distribution). The mixture lognormal distribution of the web file sizes motivates us to use a lognormal mixture to model the marginal distribution of the TCP connection duration data.

In this paper, we propose dynamical system models for both the Internet traffic stream and the TCP connection durations. The system state evolves according to a linear system model, and the output process has a mixture distribution. To fit mixture models to the marginal distribution, the EM algorithm is applied. Since the sample sizes are huge and the EM algorithm becomes slow when approaching a local optimum, a recursive EM algorithm is proposed. Fi-

0-7803-8577-2/04/$20.00 ©2004 IEEE

nally, a cross-validation criterion is proposed to select the best model. The rest of the paper is organized as follows: Section 2 presents some properties of linear systems. Section 3 presents the models used in the paper and the model fitting procedure, which consists of two sub-procedures: autocorrelation fitting and marginal distribution fitting, which are addressed in detail by Section 4 and Section 5, respectively. Section 6 gives some experimental results on real data. Section 7 concludes the paper.

2 Dynamical System Model

This section addresses how to use $AR(1)$ processes to capture the autocorrelation of real traffic. To this end, we first give a property of $AR(1)$.

Proposition 2.1 *(see Granger [8]) Let* $\{x_t^i\}, i = 1, 2, \cdots, m$, *be m independent AR(1) processes, that is,*

$$x_{t+1}^i = \alpha_i x_{t-1}^i + w_t^i \tag{1}$$

where $\alpha_i \in (0,1)$, *and* $\{w_t^i\}$ *are m independent processes with* $w_t^i \sim N(0, \sigma_i^2)$. *Then,* $x_t = \sum_{i=1}^m x_t^i \sim ARMA(m, m-1)$.

Evidently, an $AR(1)$ is a first order Markov process, which has one-step dependence. However, Proposition 2.1 tells us that the summation of independent $AR(1)$ processes can increase the dependence between the data. Based on this observation, in the following we use summation of many $AR(1)'s$ to approximate Internet traffic, which has much longer dependence than $AR(1)$. Assume that $\mathbb{E}[x_0] = 0$ and $\text{var}(x_0^i) = \sigma_i^2/(1-\alpha_i^2)$. Then we obtain the autocovariance function

$$\gamma_i(k) = \text{cov}(x_{t-k}^i, x_t^i) = \alpha_i^k \gamma_i(0). \tag{2}$$

From (2), it follows that if $\{x_t^i\}, 1 \le i \le m$ are independent, then the autocovariance function, $\gamma(t)$ of $\sum_{i=1}^m x_t^i$ can be given by

$$\gamma(t) = \sum_{i=1}^m \gamma_i(t) = \sum_{i=1}^m \sigma_{x^i}^2 \alpha_i^t. \tag{3}$$

where $\sigma_{x^i}^2 = \text{var}(x_s^i) = \frac{\sigma_i^2}{1-\alpha_i^2}$.

Equation (3) motivates us to use a summation of many independent AR(1) processes to approximate a long memory process by matching their autocovariance functions. That is, let $\gamma_z(t), t \ge 0$ be the sample autocovariance function of a sample traffic, if we can find a set of positive constants a_i and a set of positive constants b_i such that

$$\gamma_z(t) \approx \sum_{i=1}^m a_i e^{-b_i t}, t \ge 0$$

then, the summation of m AR(1)'s defined in (1) with $\sigma_{x^i}^2 = a_i$ and $\alpha_i = e^{-b_i}$ can capture the autocorrelation of the traffic.

The above approximation problem can be cast into the following optimization problem

$$\min_{\alpha_i>0, b_i>0} \sum_{t=1}^n \left[\gamma_z(t) - \sum_{i=1}^m \gamma_i(t)\right]^2 \tag{4}$$

Therefore, after solving (4) we obtain m independent AR(1) processes whose sum fits the autocorrelation of the traffic. The fitted AR(1) processes have variance a_i and autoregressive parameter $\alpha_i = e^{-b_i}$.

3 System Synthesis

The procedure of synthesizing the model can be divided into three steps: The first step captures the system autocorrelation structure, the second step fits the marginal distribution, and finally the third step connects the first two steps and establishes the model. Step 1 and Step 2 can be implemented simultaneously. The fitting procedure proposed in this paper can be summarized as follows:

Step 1) Fit the autocorrelation function, that is, solve (4) to get $a = (a_1, a_2, \cdots, a_m)$ and $b = (b_1, b_2, \cdots, b_m)$.

The first step generates a stationary process whose autocorrelation captures the time dependence of the real traffic. In fact, the first step generates a dynamical system with system state x_t and output y_t described by

$$\begin{cases} x_t = Ax_{t-1} + w_t \\ y_t = Cx_t \end{cases} \tag{5}$$

where $A = \text{diag}\{\alpha_1, \cdots, \alpha_m\}$, $w_t^\top = (w_t^1, \cdots, w_t^m)$ with $\{w_t^i, t \ge 0\}$ being m sequences of i.i.d Gaussian distributed r.v.s with parameter $\sigma_i^2 = (1-\alpha_i^2)a_i$, and $C = (1, 1, \cdots, 1)$.

Step 2) Let $z = \{z_t, 1 \le t \le n\}$ be a real traffic sample. To fit its marginal distribution, we assume that $\{z_t\}$ has an i.i.d mixture Gaussian distribution with m_1 components and find its distribution $f(z) = \sum_{k=1}^{m_1} \gamma_k \phi(z, \mu_k, \sigma_k^2)$, where $\phi(z, \mu_k, \sigma_k^2)$ designates the density function of Gaussian distribution with mean μ_k and variance σ_k^2.

With the first two steps completed, we can present the model now. To this end, we give some notations. Let $\{r_t\}$ be a sequence of independent and identically distributed r.v.s with distribution γ, where γ is the weight probability vector fitted in Step 2. From step 2, we know that $\text{var}[x_t] \triangleq \sigma_0^2$. Let $D(i) \in \mathbb{R}^{1 \times m_1}$ be defined by

$$D(i) = (\frac{\sigma_i}{\sigma_0}, \frac{\sigma_i}{\sigma_0}, \cdots, \frac{\sigma_i}{\sigma_0}) \tag{6}$$

Step 3) Define

$$x_t = Ax_{t-1} + w_t \tag{7}$$
$$y_t = D(r_t)x_t + \mu_{r_t} \tag{8}$$

where $\mu = (\mu_1, \mu_2, \cdots, \mu_{m_1})$ is the mean vector of the Gaussian mixture fitted in Step 2.

From the above three steps, we obtain the following theorem.

Theorem 3.1 *The autocorrelation function of stochastic process $\{y_t\}$ defined by (8) has the same decreasing rate as that of $\{Cx_t\}$ defined in Step 1, and $\{y_t\}$ has the same marginal distribution fitted in Step 2.*

Proof: Noting $D(i) = d(i)C$, where $d(i) = \frac{\sigma_i}{\sigma_0}$, we obtain that for any given i, j, $1 \le i, j \le m_1$, $\mathbb{E}[(D(i)x_t)(D(j)x_{t-k})] = d(i)d(j)\mathbb{E}[(Cx_t)(Cx_{t-k})]$. From (8) and the assumptions on $\{r_t, t \ge 0\}$, we obtain that when $k \ge 1$, $\mathrm{cov}(y_t, y_{t-k}) = \mathrm{cov}(D(r_t)x_t, D(r_{t-k})x_{t-k}) = C_0\mathrm{cov}(Cx_t, Cx_{t-k})$, where $C_0 = (\sum_{j=1}^{m_1}\gamma_i d(i))^2$ is a positive constant. So, $\mathrm{cov}(Cx_t, Cx_{t-k})$ and $\mathrm{cov}(y_t, y_{t-k})$ have the same rate of decay.

Next, we proceed to prove that y_t has the mixture distribution fitted in Step 2 as marginal distribution. Because $\{r_t, t \ge 0\}$ are i.i.d. and have distribution γ, it suffices to prove that, for each given i, $D(i)x_t + \mu_i \sim N(\mu_i, \sigma_i^2)$. From the definition of (6) and the fact that $Cx_t \sim N(0, \sigma_0^2)$, it follows that $D(i)x_t \sim N(0, \sigma_i^2)$, and thus $D(i)x_t + \mu_i \sim N(\mu_i, \sigma_i^2)$. This completes the proof of Theorem 3.1. Q.E.D.

In some applications, e.g., TCP connection durations, we want to model the marginal distribution of the system by a mixture lognormal distribution. In this case, we define

$$\begin{cases} x_t = Ax_{t-1} + w_t \\ y_t = e^{D(r_t)x_t + \mu_{r_t}} \end{cases} \tag{9}$$

4 Autocorrelation Fitting

To solve the optimization problem (4), the well known Prony Algorithm was invented for this purpose. However, it is well known that the Prony Algorithm (see [11, 17] and the references therein) works very efficiently if the m and n are not large. Even in this case, the Prony Algorithm is very sensitive to the initial values. On the other hand, in Internet traffic modeling studies the data are often of huge sample sizes, for which the Prony Algorithm does not work very well. In the following we propose a heuristic procedure. This procedure is adapted from [5], which uses mixture exponential distribution function to approximate a heavy-tailed distribution.

In the optimization problem (4), we call $a_i, 1 \le i \le m$, the linear parameters and b_i the nonlinear parameters. The fitting procedure is divided into two steps. The first step is to find a set of nonlinear parameter candidates $b_i, 1 \le i \le m$.

We divide the total sample lag into a set of interleaved segments, $T_j = (t_1^j, t_2^j, \cdots, t_{n_j}^j) \subset \{1, 2, \cdots, n\}$. Using least square linear fitting to get b_j such that $-b_j t + c_j$ fits $\log(\gamma_z(t_1^j)), \log(\gamma_z(t_2^j)), \cdots, \log(\gamma_z(t_{n_j}^j))$.

With nonlinear parameter candidates obtained, the second step finds the linear parameters by solving a linear programming problem, namely, finds $a = (a_1, a_2, \cdots, a_m)^\top$ satisfying $a_i \ge 0$ and

$$\mathcal{A}\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} \gamma_z(0) \\ \gamma_z(1) \\ \vdots \\ \gamma_z(n-1) \end{pmatrix} \tag{10}$$

where

$$\mathcal{A} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \beta_1 & \beta_2 & \cdots & \beta_m \\ \beta_1^2 & \beta_2^2 & \cdots & \beta_m^2 \\ \vdots & \vdots & \ddots & \vdots \\ \beta_1^{n-1} & \beta_2^{n-1} & \cdots & \beta_m^{n-1} \end{pmatrix}$$

with $\beta_i = e^{-b_i}$.

The above optimization problem is a nonnegative linear programming problem. Its least square solution contains a lot of zeros. Choose the nonzeros and their corresponding b_i's as the solution of (4). Now, we summarize the above procedure as an algorithm.

Algorithm 4.1 *Step 0) Initialization. Choose $num_1 > 1$ and $num_2 > 1$ satisfying $num_1 > num_2$.*

Step 1) Compute $m_0 = \lfloor \log(n)/log(num_1) \rfloor$, where $\lfloor x \rfloor$ denotes the floor of x.
For $i = 0 : m_0 - 1$
$t = num_1^i : num_1^i + num_2^i$;
Use the least square linear fitting to get b_i such that $-b_i t + a_i$ fits
$\log(\gamma_z(t_1)), \log(\gamma_z(t_2)), \cdots, \log(\gamma_z(t_{num_2^i}))$.
end_For

Step 2) Solve (10) to obtain $a_i, i = 1, 2, \cdots, m_0$. Assume that there are m nonzero elements in a_i's, denoted by $a_{j_s}, s = 1, 2, \cdots, m$

Step 3) Let $\alpha_i = e^{-b_{j_i}}$ and $\sigma_{x^s}^2 = a_{j_s}$. Utilize the assumption of the marginal distribution to determine the marginal distribution parameters.

Remark 4.1 *To emphasize the multiple time scale property, we can aggregate the a_i's corresponding to close b_i.*

To see the usefulness of the above procedure, we give some experimental results.

Example 4.1 *The data considered in this example consists of 24-hour downlink traffic with a one second sampling interval, taken at a high speed Internet provider, denoted as Telco A. Applying Algorithm 4.1 to the downlink traffic yields* $\alpha = (0.9584, 0.9817, 0.9817, 0.9817, 0.9977, 0.9978, 0.9998, 1.0000, 0.9999)$ *and* $\sigma^2 = 10^{11}(0.6108, 0.1128, 0.1128, 0.1128, 0.4600, 0.8929, 3.6572, 0.9149, 0.0015)$. *The sample and fitted autocorrelation functions are plotted in Figure 1.*

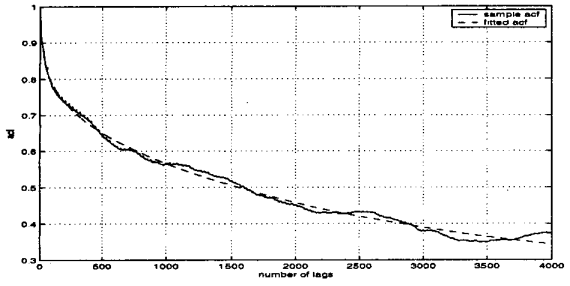

Figure 1. ACF of sample downlink traffic and fitted ACF

The above numerical results show that the proposed procedure works very efficiently to fit the autocorrelation of a given set of traffic.

5 Recursive EM Algorithm for the Marginal Distribution Fitting

To fit the margin distribution of the traffic, we treat the data as i.i.d. To get some intuition about the marginal distribution of the Internet traffic, we plot some histograms. The data are a set of uplink traffic and a set of downlink traffic from 190 high speed Internet subscribers of Telco A. Their histograms are shown in the upper and lower parts of Figure 2, respectively. Figure 2 illustrates that the probability density functions of the traffic have multiple peaks, which motivates us to model the marginal distribution of the Internet traffic by mixture distributions.

Based on the Bernstein's Theorem, Feldmann and Whitt [5] proved that finite mixture distributions can be used to approximate most distributions, especially the heavy-tailed distributions which are very popular in telecommunication network performance modeling but hard to fit. The rest of this section is dedicated to developing a recursive EM algorithm for finite mixtures.

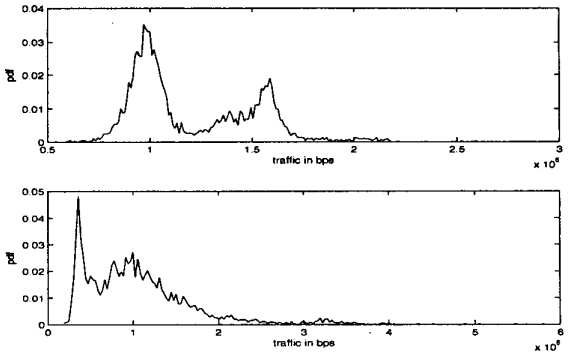

Figure 2. Histogram plots

5.1 EM Algorithm for Gaussian Mixture

Let $\{(r_t, x_t), t \geq 1\}$ be a Gaussian mixture model, where $r_t \in \{1, 2, \cdots, m_1\}$ is the regime-switching variable satisfying $\mathbf{P}(r_t = i) = \gamma_i$. Conditioned on $(r_t = i)$, x_t has distribution

$$\phi(x, \mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}, i = 1, 2, \cdots, m_1 \quad (11)$$

Normally, we only have access to the values of $X = \{x_t, 1 \leq t \leq n\}$, whose likelihood is $\prod_{t=1}^{n} \left(\sum_{i=1}^{m_1} \gamma_i \phi(x_t, \mu_i, \sigma_i^2)\right)$. Obviously, for the Gaussian mixture the maximum likelihood estimation has no closed-form solution. For this reason, we resort to the EM algorithm.

Let $\theta = (\gamma_1, \cdots, \gamma_{m_1}, \mu_1, \cdots, \mu_{m_1}, \sigma_1^2, \cdots, \sigma_{m_1}^2)$, satisfying $\gamma_i > 0, \sum_{i=1}^{m_1} \gamma_i = 1, \lambda_i > 0$, denote the system parameters. Then the likelihood of the complete data $\{(r_t, x_t), 1 \leq t \leq n\}$ is

$$L_n^c(\theta) = \prod_{t=1}^{n} \gamma_{r_t} \phi(x_t, \mu_{r_t}, \sigma_{r_t}^2) \quad (12)$$

Since we have no observations of the values of $\{r_t, 1 \leq t \leq n\}$, the EM algorithm computes and maximizes its conditional expectation, which consists of two steps: E-step and M-step. In the E-step, the conditional mean is computed, namely

$$\mathbb{E}[\log(L_n^c(\theta))|X, \theta^{(k)}] \triangleq \Phi(\theta, \theta^{(k)}) \quad (13)$$

The M-step maximizes $\Phi(\theta, \theta^{(k)})$, that is, solves the optimization problem

$$\theta^{(k+1)} = \text{argmax}_{\theta \in \Theta} \ \Phi(\theta, \theta^{(k)}) \quad (14)$$

where Θ denotes the parameter space. If necessary, the two steps are iterated until the stopping rule is fulfilled.

For a Gaussian mixture, direct computation gives

$$\begin{aligned}&\Phi(\theta,\theta^{(k)})\\&=\sum_{t=1}^{n}\mathbb{E}\left[\log(\gamma_{r_t})+\log(\phi(x_t,\mu_{r_t},\sigma_{r_t}^2))|x_t,\theta^{(k)}\right]\\&=\sum_{i=1}^{m_1}\sum_{t=1}^{n}\left[\log(\gamma_i)+\log(\phi(x_t,\mu_i,\sigma_i^2))\right]f(i|x_t,\theta^{(k)})\end{aligned}$$

where

$$f(i|x_t,\theta^{(k)})=\frac{\gamma_i^{(k)}\phi(x_t,\mu_i^{(k)},[\sigma_i^{(k)}]^2)}{\sum_{l=1}^{m_1}\gamma_l^{(k)}\phi(x_t,\mu_l^{(k)},[\sigma_l^{(k)}]^2)} \tag{15}$$

The M-Step is to solve the optimization problem (14), which has a closed-form solution (see [21] for details of the derivation):

$$\gamma_i^{(k+1)}=\frac{1}{n}\sum_{t=1}^{n}f(i|x_t,\theta^{(k)}) \tag{16}$$

$$\mu_i^{(k+1)}=\frac{\sum_{t=1}^{n}x_t f(i|x_t,\theta^{(k)})}{\sum_{t=1}^{n}f(i|x_t,\theta^{(k)})} \tag{17}$$

$$\sigma_i^{(k+1)}=\sqrt{\frac{\sum_{t=1}^{n}[x_t-\mu_i^{(k+1)}]^2 f(i|x_t,\theta^{(k)})}{\sum_{t=1}^{n}f(i|x_t,\theta^{(k)})}} \tag{18}$$

Equations (16)-(18) give the parameter update scheme of the EM algorithm for the Gaussian mixture. The algorithm iterates (16)-(18) until the stopping rule is fulfilled.

5.2 Recursive EM Algorithm

In a more abstract viewpoint, for a given sample path X the parameter updating scheme of the EM algorithm given by (16)-(18) can be viewed as a mapping $\theta^{(k+1)}=\Xi(\theta^{(k)},X)$, where Ξ is a mapping $\Xi(\theta,X):\Theta\to\Theta$. The idea of the recursive EM algorithm proposed below is to replace the fixed large sample path by successive independent small sample paths. More precisely, let $X_1,X_2,\cdots$be a set of observations, where X_i has sample size s_i. Instead of operating on the same X, we let $\theta^{(k+1)}=\Xi(\theta^{(k)},X_{k+1})$. Since in each iteration the operator Ξ operates on different sample paths, the algorithm is a stochastic one.

To consider the convergence of the algorithm, we improve the above parameter updating scheme by introducing a step size to each iteration. Let $\{a_t,t\geq 1\}$ be a sequence of positive numbers, satisfying

$$a_t>0,\ \ a_t\to 0,\ \ \sum_{t=1}^{\infty}a_t=\infty,\sum_{t=1}^{\infty}a_t^{1+\delta}<\infty, \tag{19}$$

where $\delta\in(0,1)$ is a constant. Then, at the k^{th} iteration, the parameter θ is updated by

$$\theta^{(k+1)}=(1-a_k)\theta^{(k)}+a_k\Xi(\theta^{(k)},X_{k+1}) \tag{20}$$

One of the most important properties of the EM algorithm is that it automatically satisfies the parameter constraints, that is, γ must be a probability distribution and $\lambda_i^{(k)}$ must be nonnegative. Since $a_k\to 0$ as $k\to\infty$, we can assume $a_k\in(0,1)$. As a result, the above updating scheme (20) keeps the nice feature of the EM Algorithm. The following is the algorithm.

Algorithm 5.1 *(Recursive EM Algorithm)*

Step 0 Initialization. Set θ_0, and the segment length s_0.

Step 1 Let $k=k+1$, draw sample X_k of size s_0.

Step 2 Compute $\Xi(\theta^{(k-1)},X_k)$ and update θ by (20).

Step 3 If the stop rule is fulfilled, stop; otherwise, go to Step 1.

Remark 5.1 *For analysis of huge data sets, the computation is more sensitive to memory than to the CPU frequency of the computer. Compared to the regular EM algorithm, the above algorithm can save a lot of memory.*

To see the efficiency of the above algorithm, let us give an example.

Example 5.1 *Consider a Gaussian mixture with three components. The system parameters are: $\gamma=(0.4,0.3,0.3)$, $\mu=(1,10,15)$, and $\sigma=(8,5,2)$. With this set of parameters, a sample path of size 20000 is generated.*

Now, we apply Algorithm 5.1 with segment length 1000 and $a_k=\frac{1}{k^{0.6}}$. Choosing initial values $\mu^{(0)}=(2,8,11.5)$, $\sigma^{(0)}=(6,3.5,1.2)$, $\theta^{(0)}=(0.45,0.35,0.2)$ and running the above algorithm 200 iterations yield $\hat{\mu}=(1.8218,11.70769,15.18328)$, $\hat{\sigma}=(7.9997,4.28485,1.790129)$ and $\hat{\gamma}=(0.45667,0.30779,0.23553)$.

Figure 3 plots the sample histogram and the fitted PDFs. The square error of the pdf estimation is $2.4003e-005$. The average likelihood of the computation is plotted in Figure 4. The conventional EM algorithm can make sure that the likelihood keeps strictly increasing with iterations. However, Algorithm 5.1 is a stochastic one and loses this nice characteristic. But, Figure 4 shows the trend of the likelihood increases.

The above numerical results show that the proposed algorithm works very well.

In Algorithm 5.1, the length of the segment s_0 is chosen by experience. The smaller the segment length used, the more memory is saved, but the algorithm become less robust. On the other hand, the larger the segment length chosen, the more robust the algorithm is, but more memory is consumed. For a set of simulated data, s_0 can be chosen very small. Experimental results show that to deal with the

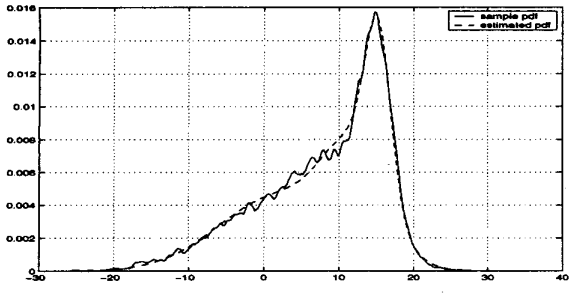

Figure 3. Comparison between sample pdf and fitted pdf

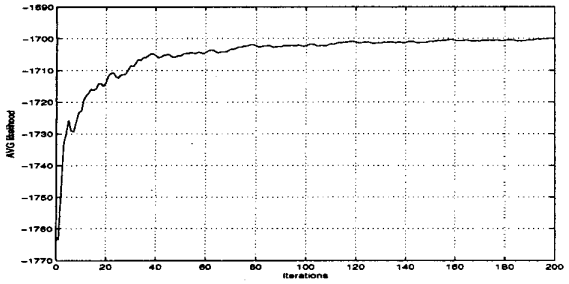

Figure 4. Average likelihood versus iterations

above simulated data of 3-component, the segment length can be chosen as short as 20. But for real traffic, since they are not so homogeneous as the simulated data and contain outliers, the algorithm cannot work properly if s_0 is chosen too small.

5.3 Convergence Analysis

If we know a closed bounded set $\Theta_0 \subset \Theta$, which contains the optimal solution, we can apply the traditional projection $\theta^{(k+1)} = \prod_{\Theta_0}((1-a_k)\theta^{(k)} + a_k \Xi(\theta^{(k)}, X_{k+1}))$, where $\prod_{\Theta_0}$ is the nearest projection on Θ_0. However, for real traffic, we have no idea about the optimal solution before starting the algorithm. Since the optimization problem has constraints, $\lambda_i > 0, \gamma_i > 0, \sum_{i=1}^{m_1} \gamma_i = 1$, and the parameter space is an open set, to establish the convergence of the above Algorithm 5.1 in large range, we change the presentation of the above algorithm using the *random varying truncations* (RVT) approach proposed by Chen et. al [3]. The key idea of RVT is to replace the fixed projection set Θ_0 by a sequence of increasing closed sets M_s which approximate the whole parameter space Θ. If at some step s, the updated parameter $\theta^{(s+1)}$ given by (20) crosses the boundary of the current truncation set, namely $\theta^{(k+1)} \notin M_s$, restart the algorithm and change the truncation set to M_{s+1}. Under some conditions, it can be proven that with probability one there exist at most finite times of truncations and the truncation set contains the optimal solution. After the truncations stop, the algorithm becomes a regular projection-based stochastic approximation algorithm. Chen [4] extends the RVT method to deal with stochastic approximation problem with state-dependent noise, which is defined as a measurable function of the system state, and the convergence conditions on the noise are given with the system state restricted to a fixed closed set. These extensions make it much easier to verify the convergence conditions.

To apply Theorem A.1 of [14], which is adapted from Theorem 1 of [4], to establish the convergence of the recursive EM algorithm, we utilize the fact that at each iteration, the parameter increments have a positive projection on the gradient of the likelihood function, which is established by Xu [21] for a Gaussian mixture. To see this, for a given sample $X = \{x_t, 1 \le t \le n\}$ and a given $\theta \in \Theta$, we define

$$L_n(X,\theta) = \sum_{t=1}^{n} \log\left(\sum_{i=1}^{m_1} \gamma_i f(x_t, \lambda_i)\right) \tag{21}$$

Proposition 5.1 *The EM algorithm can be represented by*

$$\gamma^{(k+1)} = \gamma^{(k)} + S_\gamma(\theta^{(k)}) \frac{\partial L_n(X,\theta)}{n\partial\gamma}\bigg|_{\gamma=\gamma^{(k)}} \tag{22}$$

$$\mu^{(k+1)} = \mu^{(k)} + S_\mu(\theta^{(k)}) \frac{\partial L_n(X,\theta)}{n\partial\lambda}\bigg|_{\theta=\theta^{(k)}} \tag{23}$$

$$\left[\sigma^{(k+1)}\right]^2 = \left[\sigma^{(k)}\right]^2 + S_\sigma(\theta^{(k)}) \frac{\partial L_n(X,\theta)}{n\partial(\sigma^2)}\bigg|_{\theta=\theta^{(k)}} \tag{24}$$

where

$$S_\gamma(\theta^{(k)}) = \begin{pmatrix} \gamma_1^{(k)} & & \\ & \ddots & \\ & & \gamma_{m_1}^{(k)} \end{pmatrix} - \gamma^{(k)}[\gamma^{(k)}]^\top$$

$$S_\mu(\theta^{(k)}) = \begin{pmatrix} \frac{[\sigma_1^{(k)}]^2}{\gamma_1^{(k+1)}} & & \\ & \ddots & \\ & & \frac{[\sigma_{m_1}^{(k)}]^2}{\gamma_{m_1}^{(k+1)}} \end{pmatrix}$$

$$S_\sigma(\theta^{(k)}) = \begin{pmatrix} \frac{[\sigma_1^{(k)}]^4}{\gamma_1^{(k+1)}} & & \\ & \ddots & \\ & & \frac{[\sigma_{m_1}^{(k)}]^4}{\gamma_{m_1}^{(k+1)}} \end{pmatrix}$$

Remark 5.2 *With a little abuse of notation, the above $S_\mu(\theta^{(k)})$ and $S_\sigma(\theta^{(k)})$ contain $\gamma^{(k+1)}$ instead of $\gamma^{(k)}$. At each iteration, γ is updated first; so when calculating $\mu^{(k+1)}$ and $\sigma^{(k+1)}$, $\gamma^{(k+1)}$ is available for computing $S_\mu(\theta^{(k)})$ and $S_\sigma(\theta^{(k)})$.*

Proof: The proof of (22) is a special case of Theorem 1 of [21]. From Theorem 1 of [21], we obtain

$$\mu^{(k+1)} = \mu^{(k)} + T_\mu(\theta^{(k)}) \frac{\partial L_n(X,\theta)}{n\partial\lambda}\bigg|_{\theta=\theta^{(k)}} \quad (25)$$

$$\left[\sigma^{(k+1)}\right]^2 = \left[\sigma^{(k)}\right]^2 + T_\sigma(\theta^{(k)}) \frac{\partial L_n(X,\theta)}{n\partial(\sigma^2)}\bigg|_{\theta=\theta^{(k)}} \quad (26)$$

where

$$T_\mu(\theta^{(k)}) = \begin{pmatrix} \frac{n[\sigma_1^{(k)}]^2}{\sum_{t=1}^n f(1|x_t,\theta^{(k)})} & & \\ & \ddots & \\ & & \frac{n[\sigma_{m_1}^{(k)}]^2}{\sum_{t=1}^n f(m_1|x_t,\theta^{(k)})} \end{pmatrix}$$

$$T_\sigma(\theta^{(k)}) = \begin{pmatrix} \frac{n[\sigma_1^{(k)}]^4}{\sum_{t=1}^n f(1|x_t,\theta^{(k)})} & & \\ & \ddots & \\ & & \frac{n[\sigma_{m_1}^{(k)}]^4}{\sum_{t=1}^n f(m_1|x_t,\theta^{(k)})} \end{pmatrix}$$

Noting expression (16), we have $S_\mu(\theta^{(k)}) = T_\mu(\theta^{(k)})$ and $S_\sigma(\theta^{(k)}) = T_\sigma(\theta^{(k)})$. This completes the proof of Proposition 5.1. Q.E.D.

Proposition 5.1 tells us that, in essence, the EM Algorithm is a gradient algorithm

$$\theta^{(k+1)} - \theta^{(k)} = S(\theta^{(k)}) \frac{L_n'(X_{k+1},\theta^{(k)})}{n},$$

where $S(\theta) = \begin{pmatrix} S_\gamma(\theta) & 0 & 0 \\ 0 & S_\lambda(\theta) & 0 \\ 0 & 0 & S_\sigma(\theta) \end{pmatrix}$ and

$$L_n'(X_{k+1},\theta^{(k)}) = \frac{\partial L_n(X_{k+1},\theta)}{\partial\theta}|_{\theta=\theta^{(k)}}.$$

So, the proposed parameter update scheme (20) can be represented by

$$\begin{aligned} \theta^{(k+1)} &= (1-a_k)\theta^{(k)} + a_k \Xi(\theta^{(k)}, X_{k+1}) \\ &= (1-a_k)\theta^{(k)} \\ &\quad + a_k \left[\theta^{(k)} + S(\theta^{(k)}) \frac{L_n'(X_{k+1},\theta^{(k)})}{n}\right] \\ &\triangleq \theta^{(k)} + a_k y_{k+1} \end{aligned} \quad (27)$$

where

$$\begin{aligned} y_{k+1} &= S(\theta_k) l'(\theta^{(k)}) \\ &\quad + S(\theta^{(k)}) \left[\frac{L_n'(X_{k+1},\theta^{(k)})}{n} - l'(\theta^{(k)})\right] \end{aligned} \quad (28)$$

and $l(\theta) = \mathbb{E}[L_n(\theta, X)/n]$. If we denote $h(\theta) = S(\theta)l'(\theta)$ and

$$\varepsilon_{k+1} = S(\theta^{(k)}) \left[\frac{L_n'(X_{k+1},\theta^{(k)})}{n} - l'(\theta^{(k)})\right], \quad (29)$$

then (27) becomes a standard stochastic approximation algorithm:

$$\theta^{(k+1)} = \theta^{(k)} + a_k y_{k+1} \quad (30)$$

$$y_{k+1} = h(\theta^{(k)}) + \varepsilon_{k+1} \quad (31)$$

Chen [4] considers an unconstrained optimization problem, but ours is a constrained one. To apply his results, we have to deal with the boundary of the parameter space. To this end, let us introduce some notations:

$$M_s = \{\theta : \|\theta\| \le \rho_s, d(\theta, \partial\Theta) \ge 1/\rho_s\} \quad (32)$$

$$Q_s = \{\theta : \|\theta\| \le s, d(\theta, \partial\Theta) \ge 1/s\} \quad (33)$$

where $d(\theta, \partial\Theta)$ denotes the distance between θ and $\partial\Theta$, and $\{\rho_s, s \ge 1\}$ is a sequence of positive numbers satisfying $\rho_s \to \infty, s \to \infty$.

Remark 5.3 *The randomly varying truncation technique is used only to establish the convergence of the recursive EM algorithm. Actually, since the EM algorithm always satisfies the constraints, when the first truncation set is chosen large enough, no truncation occurs at all.*

To establish the convergence of $\theta^{(k)}$, we redefine it using the RVT technique as follows:

$$\theta^{(k+1)} = \tilde{\theta}^{(k)} I_{[\tilde{\theta}^{(k)} \in M_{\tau^{(k)}}]} + \theta^* I_{[\tilde{\theta}^{(k)} \notin M_{\tau^{(k)}}]} \quad (34)$$

where

$$\tilde{\theta}^{(k)} = (1-a_k)\theta^{(k)} + a_k \Xi(\theta^{(k)}, X_{k+1})$$

$$\tau^{(k)} = \sum_{i=1}^{k-1} I_{[\tilde{\theta}^{(i)} \notin M_{\tau^{(i)}}]}, \ \tau^0 = 0$$

and θ^* is a given point.

The following theorem establishes the convergence of the proposed algorithm.

Theorem 5.1 *Assume that there exists a bounded closed set Q_{c_0} and $\theta^* \in Q_{c_0}$ satisfying*

$$\sup_{\theta \in \partial Q_{c_0}} l(\theta) < l(\theta^*) \quad (35)$$

and that $l(\theta)$ has finite number of stationary points. Let $\theta^{(k)}$ be given by (34). Then,

$$\theta^{(k)} \to \theta^0, \ a.s. \quad (36)$$

where $l'(\theta^0) = 0$.

Proof: The proof consists of two steps. Step 1 establishes that the noise process $\{\varepsilon_t, t \ge 1\}$ is a martingale difference sequence. The second step proves that the conditions (A1)-(A4) of Theorem A.1 of [14] are fulfilled. The details can be given following the same line as Theorem 2 of [14]. Q.E.D.

Remark 5.4 *From the above derivation, we find that the recursive EM algorithm is a stochastic approximation algorithm with noise given by (29). If the sample size s_k of X_k is increasing with k, then using the strong law of large numbers, it is easy to prove that* $\lim_{k\to\infty} \varepsilon_k = 0, a.s.$ *In this case,* $\theta^{(k+1)} = \Xi(X_{k+1}, \theta^{(k)})$ *converges without averaging. This property has been illustrated for a Poisson Mixture by Example 5.1 of [14].*

5.4 Real Traffic Marginal Distributions Fitting

This subsection applies the recursive EM algorithm to fit a set of real traffic. The data used in the following experiment are a set of uplink traffic data from Telco A with 190 subscribers. The histogram of the traffic has been plotted in Figure 2, which motivates us to use a Gaussian mixture to model its marginal distribution.

To fit a Gaussian mixture with 4 components to this traffic, we choose the segment length $s_0 = 1311$ and initial values $\mu^{(0)} = 10^5[1.8, 3.8, 2.8, 3.3]$, $\sigma^{(0)} = 10^5[1.3, 1.5, 0.7, 1.7]$ and $\theta^{(0)} = [1/4, 1/4, 1/4, 1/4]$. With the set of initial values, running the algorithm 1000 iterations yields $\hat{\mu} = [631062.8925, 1175071.3591, 579609.9346, 1355064.7578]$, $\hat{\sigma} = [67495.9475, 185313.863, 300684.8207, 510676.314]$ and $\hat{\gamma} = [0.0946, 0.193, 0.1014, 0.6108]$. The square error between the sample PDF and the fitted PDF is $sqerr = 2.2003e-004$. Figure 5 plots the sample and fitted PDFs. The above experimental results show that the recursive EM algorithm works very efficiently to capture the marginal distribution of real traffic.

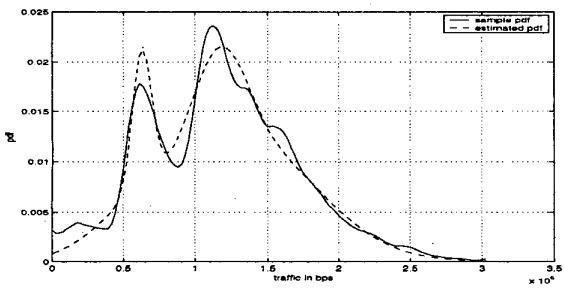

Figure 5. Comparison between the sample PDF and the fitted PDF

6 Experimental Results on Real Traffic

This section applies model (9) to a set of TCP connection durations and uses the proposed procedure in the previous sections to fit the model.

The TCP connection durations data used in this paper were collected by the University of North Carolina at Chapel Hill, available at http://www-dirt.cs.unc.edu/NetDepend/data. First, we plot a small sub-sample of the data, which is shown in Figure 6. This figure shows that the TCP connection duration sample is comprised of a few very long flows (elephants) and a large amount of short flows (mice). The mice comprise the major stream of the Internet traffic stream, and a small amount of the elephants results in the long memory of the Internet traffic.

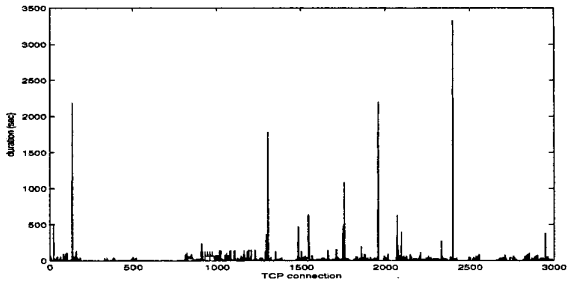

Figure 6. A TCP connection durations sample

In the Internet modeling field, it is commonly accepted that the sizes of files in the Internet are heavy-tailed distributed. Researchers have found that the web file size distributions have a body similar to lognormal distribution, while their tails are close to Pareto distribution. To capture these characteristics, the double Pareto-lognormal distribution was proposed by [19]. In essence, the double Pareto-lognormal distribution is a mixture lognormal distribution. If we assume that each file is transferred by one TCP connection, from the file size models we are tempted to assume the TCP connection durations have mixture lognormal distribution. To verify this assumption, we denote by $\{d_1, d_2, \cdots, d_n\}$ the original sample TCP connection durations and let $x_t = \log_{1.2}(d_t), 1 \le t \le n$. Their histograms are plotted in Figure 7. The upper part of Figure 7 plots the histogram of $\{d_1, d_2, \cdots, d_n\}$, and the lower part is the histogram of $\{x_1, x_2, \cdots, x_n\}$. Compared to the mice, the elephants are huge but their quantity is so small that the histogram cannot be well displayed. For this reason, we will deal with their logarithm values.

To apply the model (9) to describe the TCP connection durations, it suffices to fit their logarithm values $\{x_1, x_2, \cdots, x_n\}$ to a mixture dynamical system described by (7) and (8). We first fit the autocorrelation of $\{x_1, x_2, \cdots, x_n\}$. Applying Algorithm 4.1 to the data yields $b = [0.25029, 0.03199, 0.02716, 0.003479, 0.002425, 0.000294, 0.000011, 0.0001806]$ and $a = [46.6249, 7.3866, 37.5739, 2.6990, 21.6165, 17.6530,$

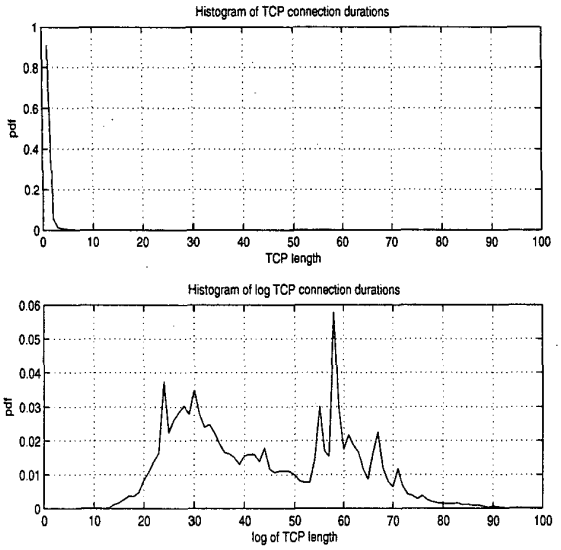

Figure 7. PDF of TCP connection duration

3.7468, 16.0481]. The result of the fitting is plotted in Figure 8.

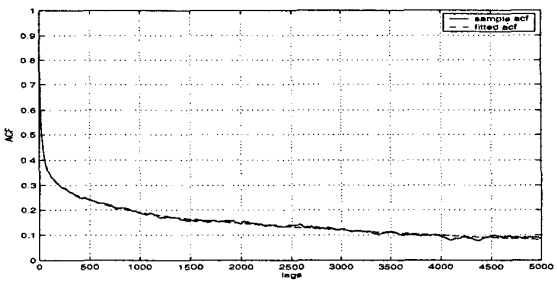

Figure 8. TCP connection duration autocorrelation fitting result

Second, we apply the recursive EM algorithm to fit $\{x_1, x_2, \cdots, x_n\}$ to a Gaussian mixture. In the step of autocorrelation fitting, the dimension of x_t is uniquely determined by fitting the autocorrelation of the real traffic. But, we are free to choose the number of components of the mixture in the marginal distribution fitting step. A lot of experiments have been conducted on this set of data, which show that Gaussian mixtures with 2-7 components can all fit the marginal distribution with acceptable precision. Obviously, the more components used, the higher the accuracy that can be achieved. If more than 7 components are used, some components in the fitted model are almost identical. For the purpose of model selection, we divide the sample into two parts: The first part is used to fit the model, and the second part is used for model selection by using the cross-validation procedure, which works as follows: Divide the original data into two parts $\{x_1, \cdots, x_N\}$ and $\{x_{N+1}, \cdots, x_n\}$. We fit $\{x_1, \cdots, x_N\}$ to Gaussian mixtures with $m_1 = 2, 3, \cdots, 7$ components and simulate a sample path $y = \{\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_{n-N}\}$ for each fitted model. Define the performance index

$$ER = \frac{1}{n-N} \sum_{t=1}^{n-N} [\tilde{x}_t - x_{N+t}]^2$$

and choose the one with minimal ER as the selected model. In this example, the sample size of the data is 358230. The first 300000 entries are used for model fitting and the remaining 58230 entries are used for model selection. The fitted model results with $2-7$ components are given in Table 1, which shows that the ER attains a minimum at $m_1 = 3$. So, we conclude that the model with 3-components should be chosen. When $m_1 = 3$, the fitted parameters are $\hat{\mu} = [-8.9621, 7.6148, 14.5972]$, $\hat{\sigma} = [4.1920, 11.34026, 3.8576]$ and $\hat{\gamma} = [0.37618, 0.4299, 0.1944]$. The sample PDF and the fitted PDF are plotted in Figure 9.

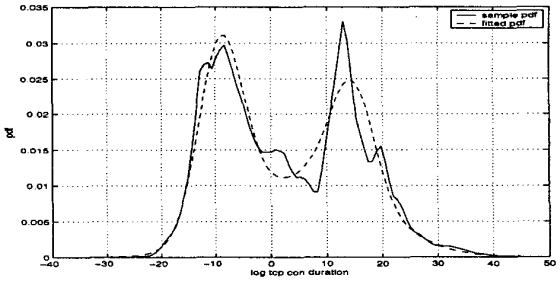

Figure 9. Sample PDF and fitted PDF with 3-components

7 Concluding Remarks

This paper proposed a mixture linear dynamical system model for an Internet packet input stream and a nonlinear mixture dynamical model for TCP connection durations. The proposed models can capture the autocorrelation and marginal distribution of real data simultaneously. To fit the marginal distribution, a recursive EM algorithm is proposed. Compared to the conventional EM algorithm, the proposed recursive algorithm requires much less memory, which is critical in Internet performance modeling. More importantly, it can be implemented online. Finally, a cross-validation criterion is applied for model selection.

Table 1. Computation results for various models

# of Comp	2	3	4	5	6
ER	160.9707	143.3078	169.9892	182.1334	197.7227

A lot of experimental results reveal that Internet traffic shows periodicity. The model proposed in this paper captures the average decay rate, but it cannot model the periodicity. As we know, the TES processes [15] can capture the autocorrelation very accurately. To integrate TES into the proposed model may lead to good results.

Acknowledgments

This work was funded by Atlantic Canada Opportunity Agency (ACOA) and through grant from the National Science and Engineering Research Council of Canada (NSERC) to Dr. V. Choulakian.

References

[1] A. Andersen and B. Nielsen. A Markovian approach for modeling packet traffic with long-range dependence. *IEEE J. on Selected Areas in Communications*, 16(5):719–732, 1998.

[2] J. Beran. *Statistics for long-memory processes. Monographs on statistics and Applied Probability*. Chapman and Hall, New York, 1994.

[3] H. F. Chen, L. Guo, and A. J. Gao. Convergence and robustness of the Robbins-Monro algorithm truncated at randomly varying bounds. *Stochastic Processes and Their Applications*, 27:217–231, 1988.

[4] H. F. Chen. Stochastic approximation with state-dependent noise. *Science in China (series E)*, 43(5):531–541, 2000.

[5] A. Feldmann and W. Whitt. Fitting mixture of exponentials to long-tailed distributions to analyze network performance models. *Performance Evaluation*, 31:245–279, 1998.

[6] M. W. Garrett and W. Willinger. Analysis, modeling, and generation of self-similar vbr video traffic. In *Proc. of the SIGCOMM'94 Conference*, pages 269–280, 1994.

[7] W. Gong, Y. Liu, V. Misra, and D. Towsley. On the tails of web file size distributions. In *Proc. of 39-th Allerton Conference on Communication, Control and Computing*, 2001.

[8] C. W. J. Granger. Long memory relationships and the aggregation of dynamic models. *Journal of Econometrics*, 14:227–238, 1980.

[9] M. Grossglauser and J. C. Bolot. On the relevance of long-range dependence in network traffic. In *Proc. of the ACM SIGCOMM'96 Conference*, 1996.

[10] F. Hernandex-Campos, J. Marron, G. Samorodnitsky, and F. Smith. Variable heavy tailed durations in internet traffic, part i: understanding heavy tails. *http://www.cs.unc.edu/Research/dirt/proj/marron /VarHeavyTails*, 2002.

[11] M. Kahn, M. S. Mackisack, M. R. Osborne, and G. K. Smyth. On the consistency of prony's method and related algorithms. *J. Comput. Graph. Statist.*, 1:329–349, 1992.

[12] M. M. Krunz and A. M. Makowski. Modeling video traffic using $m/g/\infty$ input processes: A compromise between Markovian and lrd models. *IEEE J. Select. Areas Commun*, 16(5):733–748, 1998.

[13] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. In *Proc. ACM SIGCOMM'93*, pages 183–193, 1993.

[14] Z. Liu, J. Almhana, V. Choulakian, and R. McGorman. Integer-valued mixture dynamical systems with application to internet traffic modeling. *submitted for publication*, 2003.

[15] B. Melamed. *An overview of TES processes and modeling methodology*, pages 359–393. Springer-Verlag, Lecture Notes in Computer Science, 1993.

[16] M. Mitzenmacher. Dynamic models for file sizes and double pareto distributions. *http://www.eecs.harvard.edu/ michaelm/NEWWORK /postscripts/filesize.pdf*, 2003.

[17] M. R. Osborne and G. K. Smyth. A modified prony algorithm for fitting sums of exponential functions. *SIAM J. Sci. Statist. Comput.*, 16:119–138, 1995.

[18] K. Park and W. Willinger. *Self-similar network traffic and performance evaluation*. John Wiley and Sons Inc., New York, 2001.

[19] W. J. Reed. The double pareto-lognormal distribution-a new parametric model for size distributions. *http://www.math.uvic.ca/faculty/reed*, 2001.

[20] P. Salvador and R. Valadas. Multiscale fitting procedure-using Markov modulated poisson processes. *Telecommunications Systems*, 23(1):123–148, 2003.

[21] L. Xu and M. I. Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural Computation*, 8(1), 1996.

[22] T. Yoshihara, S. Kasahara, and Y. Takahashi. Practical time-scale fitting of self-similar traffic with Markov-modulated poisson process. *Telecommunication Systems*, 17(1):185–211, 2001.