Sequential Monte Carlo Inference of Internal Delays in Nonstationary Communication Networks *

Mark J. Coates and Robert D. Nowak

Department of Electrical and Computer Engineering
Rice University, MS-380, P.O. Box 1892
Houston, Texas 77251-1892 USA

Fax: (+1 713) 737-6196
Email: {mcoates,nowak}@rice.edu
Web: www.dsp.rice.edu

Submitted to the IEEE TRANSACTIONS ON SIGNAL PROCESSING
Special Issue on Monte Carlo Methods for Statistical Signal Processing
January, 2001

Abstract

On-line, spatially localized information about internal network performance can greatly assist dynamic routing algorithms and traffic transmission protocols. However, it is impractical to measure network traffic at all points in the network. A promising alternative is to measure only at the edge of the network and infer internal behavior from these measurements. In this paper we concentrate on the estimation and localization of internal delays based on end-to-end delay measurements from a source to receivers. We propose a sequential Monte Carlo (SMC) procedure capable of tracking nonstationary network behavior and estimating time-varying, internal delay characteristics. Simulation experiments demonstrate the performance of the SMC approach.

1 Introduction

In large-scale networks, end-systems cannot rely on the network itself to cooperate in characterizing its own behavior. This has prompted several groups to investigate methods for inferring internal network behavior based on "external" end-to-end network measurements [1, 5, 8, 9, 10, 15, 29, 31, 33] or, conversely, estimating source-destination traffic intensities from internal measurements [32, 34, 35]; both problems are often referred to as *network tomography*.

Optimizing communication network routing and service strategies requires knowledge of the queueing delay at different points in the network. However, it is impractical to directly measure packet delays at each and every router for many reasons [29]. Measuring end-to-end (source to

*This work was supported by the National Science Foundation, grant no. MIP–9701692, the Army Research Office, grant no. DAAD19-99-1-0349, the Office of Naval Research, grant no. N00014-00-1-0390.

receivers) delays using timestamps [2, 26] is relatively easy and inexpensive in comparison. Consequently, it is natural to consider the following inverse problem: from end-to-end measurements can we resolve the delay experienced at internal points in the network? This is somewhat analogous to the medical tomography problem, and hence the name *network tomography*. Recently, sequential Monte Carlo methods have received considerable attention in the statistics and signal processing literatures [4, 12, 13, 18, 24, 28]. In this paper, we propose a novel Monte Carlo methodology based on sequential importance sampling [7, 14, 28] that not only addresses the basic (stationary) network tomography problem, but also directly tackles the more challenging and realistic problem of tracking time-varying network delay behavior.

The basic idea is quite straightforward. Consider a network consisting of a single source, sending packets to several receivers. Standard network routing protocols produce a tree-structured topology[1] for the network in this case, with the *source* at the root and the *receivers* at the leaves. A small network with four receivers is depicted in Figure 1, below. The nodes between the source and receivers represent internal *routers*. Connections between the source, routers, and receivers are called *links*. Each link between routers may be a direct connection, or there may be "hidden" routers (where no branching occurs) along the link that are not explicit in our representation.

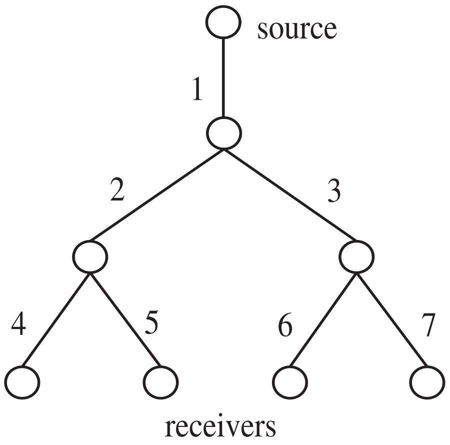


Figure 1: Tree-structured network topology. A binary tree (each parent node has two children nodes) is shown here, but in the general case the tree is non-binary.

Suppose two closely time-spaced (back-to-back) packets are sent from the source to two different receivers. The paths to these receivers traverse a common set of links, but at some point the two paths diverge (as the tree branches). The two packets should experience approximately

[1]We assume that the topology is *known* and *fixed* throughout this paper. In practice routing tables are updated every several minutes. Extensions of our methods that account for changes in topology (over *very* coarse time scales) are possible, but not considered here.

the same delay on each shared link in their path. This facilitates the resolution of the delays on each link. More precisely, the goal of the network tomography problem considered in this paper is to estimate the probability distribution of the delay on each link, based on the end-to-end packet pair measurements. To illustrate the idea in its simplest form, suppose that we send many packet pairs to receivers at the end of links 4 and 5 in Figure 1 and measure the delays experienced by each packet. Each measurement consists of a pair of delays, one being the delay to the receiver at 4 and the other the delay to the receiver at 5. From these measurements, collect events where the delay measured at the receiver at 5 is zero (or, more generally, the minimum possible delay). Now, assuming that the delay is the same for both packets on the common links (1 and 2 in this case), any "additional" delay observed to the receiver at 4 can be attributed to link 4 alone. We can then build a histogram estimate of the delay distribution for link 4. This simple idea can be extended to obtain estimators for the delay distributions on all links [10, 29].

Suppose the network is stationary over the observation period, the delays are identical on shared links, and the true delay distributions are strictly positive. Then, based on the multicast analysis made in [29], one can show that the true distributions can be uniquely identified from such end-to-end measurements (as the number of measurements tends to infinity). A natural estimator in this case is the maximum likelihood estimator (MLE). In previous work, we developed an Expectation-Maximization (EM) algorithm to compute the MLE [10]. More generally, the dynamics of the network may be changing over time, and the delay distributions themselves are no longer static. In this case, we must model the dynamics and track the network behavior. In Section 4, we propose a stochastic model of the network dynamics. The available observations are a highly non-linear function of the system. As a result, the extended Kalman filter is not suitable for the task, and we propose a sequential Monte Carlo algorithm instead. The algorithm is capable of tracking the time-varying delay distributions. We also show, through simulation experiments, that the sequential Monte Carlo method's performance can be significantly better than that of the EM algorithm.

The problem and approach in this paper differ considerably from previous network tomography work in several key respects.

1. The problem considered here is that of inferring internal network behavior characteristics from "external" end-to-end measurements. This is quite different from the source-destination estimation problem [32, 34, 35].

2. The internal delay inference method in [29] is closest in spirit to our problem. However, that method employs multicast probing, which is not supported by many networks due to its scalability limitations. Perhaps a more significant limitation of the multicast approach is that it may not provide an accurate characterization of the normal (unicast) traffic often of most interest, because routers treat multicast packets differently than unicast packets [15]. In contrast, our methods are based on unicast measurements, which can be made on any network and which, of course, directly provide information about unicast traffic[2].

3. Our approach is based on a Bayesian formulation of the network tomography problem, building on our earlier likelihood-based methods [8, 9, 10, 33]. In contrast, the multicast approach in [29] employs an estimator based on empirical probabilities.

4. Our sequential method is specifically designed for tracking time-varying behavior, whereas the method in [29] is only appropriate for stationary cases. The problem of estimating time-varying source-destination traffic intensities from internal measurements was examined in [34], but that task is quite unrelated to inference of internal delays addressed in this paper.

The paper is organized as follows. In Section 2 we detail the measurement process and our observation model. In Section 3 we propose a stochastic dynamical model for nonstationary communication networks. This model underpins our sequential Monte Carlo (SMC) inference algorithm, developed in Section 4. In Section 5, we evaluate the performance of the SMC algorithm with simulated network experiments. Discussion and conclusions are provided in Section 6.

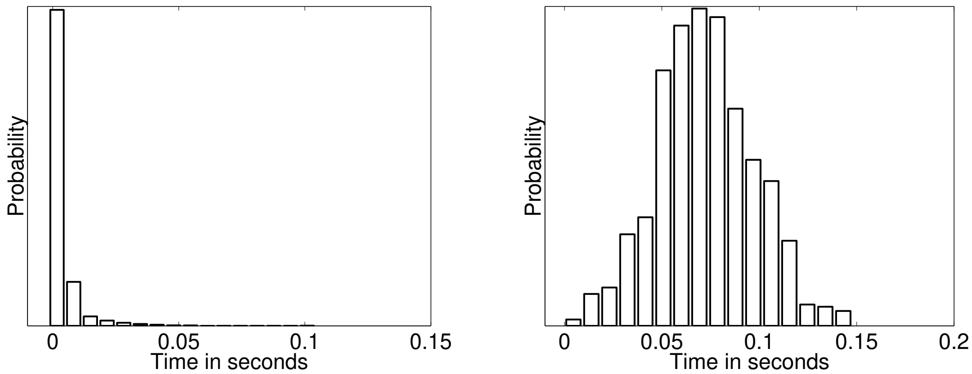
2 Measurements and Observation Model

We collect measurements of the end-to-end delays from source to receivers, and we index the packet pair measurements by $m = 1, \ldots, M$. For the m-th packet pair measurement, let $y_1(m)$ and $y_2(m)$ denote the two end-to-end delays measured. The ordering 1 and 2 is completely arbitrary. In this paper, we do not consider the case in which one or both of the packets is dropped (lost). We simply discard packet pairs in which a loss occurs. However, it is possible to extend our approach to include losses as well. The delays are quantized such that the quantized delay on each link falls in the range $0, 1, \ldots, K$ time units. There are several options for choosing the quantization level, and perhaps the most natural is to quantize the range between the minimum and maximum observed path delay according to a desired level of accuracy. Other possibilities are suggested in [29]. An indication of the nature of commonly encountered delay distributions is provided in Figure 2, which depicts the end-to-end (quantized) delay histograms from recorded measurements of the Internet[3].

To describe our observation model, let us first consider the case of a stationary network in which the delay characteristics are not time-varying. Associated with each individual link/router in the network is a probability mass function (pmf) for the queuing delay. Let

[2] As pointed out in [15], it should be possible to extend the method in [29] to the unicast case.

[3] The measurements shown in Figure 2 were made using a tool called *netdyn* [2]. A large number of packet pairs, with inter-packet spacing of approximately 1 ms, were sent to a remote host. Each packet was 64 bytes in size and the time spacing between packet pair transmissions was approximately 500 ms. The paths involved in these measurements included approximately 10 separate links.



(a) (b)

Figure 2: Delay histograms from two different Internet measurement sessions. Measurements were made using the *netdyn* tool [2].

$p_i = \{p_{i,0} \ldots, p_{i,K}\}$ denote the probabilities of a delay of $0, 1, \ldots, K$ time units, respectively, on link i. Given the packet pair measurements $\mathbf{y} \equiv \{y_1(m), y_2(m)\}$, we are interested in maximum likelihood estimates (MLEs) of $\mathbf{p} \equiv \{p_i\}$, the collection of all delay pmfs. The likelihood of each delay measurement is parameterized by a convolution of the pmfs in the path from the source to receiver. The coupling of the pmfs of each link results in a likelihood function that cannot be maximized analytically. The joint likelihood $l(\mathbf{y}|\mathbf{p})$ of all measurements is equal to a product of the individual likelihoods. The maximization of the joint likelihood function requires numerical optimization, and the EM algorithm is an attractive strategy for this purpose. In previous work, we have developed EM algorithms for network tomography, to estimate both internal losses [8] and internal delays [10].

In nonstationary networks, the queuing behavior varies over time, and the notion of a delay distribution is not well defined. Nonetheless, time functions such as the expected delays across each link are very much of interest. To put such notions on firmer ground, we define the time-varying delay distribution of window size R at measurement m as:

$$p_{i,j}(R,m) = \frac{1}{R} \sum_{l=m-R+1}^{m} \mathbf{1}_{\{z_i(l)=j\}}, \tag{1}$$

with $z_i(l)$ being the (unobserved) delay experienced at queue i by measurement packets l and $\mathbf{1}_{\{z_i(l)=j\}}$ is the indicator function for the event $\{z_i(l) = j\}$. Let $p_{i,R} \equiv \{p_{i,j}(R,m)\}$ denote the time-varying probabilities of a delay on link i. The window size R may be selected on the basis of known or assumed dynamics of the network.

Before moving on, let us comment briefly on the assumption that back-to-back packets are delayed by roughly the same amount on each shared link in their paths. If the delays are identical on shared links, then the difference between the two delay measurements can be attributed solely to the delays experienced on unshared links in the two paths. This is the key to resolving the delays on a link by link basis. However, in practice the two packets may experience slightly different delays on shared links due to the fact that one packet precedes the other in the queues and additional packets may intervene between the two. The nature of this delay differential is exposed in Figure 3, which shows the histogram of the difference between the end-to-end delays of two closely-spaced packets sent to the same Internet receiver. This histogram is constructed from the back-to-back packet pair measurements along the same connection considered in Figure 2 (a). Ideally, the delays should be identical, but we see a small discrepancy between the two. The second packet in the pair typically experiences a slightly greater delay. However, recall that the ordering of the packets was arbitrary in our recording process. In effect then, the discrepancies between the delays on shared links adds a zero mean error to the difference between the two end-to-end measurements. We clearly see the symmetric zero-mean nature in the empirical data shown in Figure 3. This "noise" produces a smoothing (or blurring) in the inferred delay pmfs. Nonetheless, because the errors are zero mean, we can still use the estimated delay pmfs to obtain reasonable estimates of the expected delay on each link. Thus, our methodology can provide important information, even when the delays on shared links are not identical.

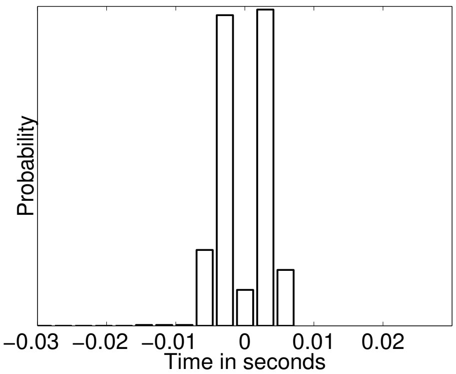


Figure 3: Difference between end-to-end delays of packet pair sent to same receiver. Ideally the difference should be identically zero, since the two packets traverse the same links, but in practice we observe a small error. Measurements were made using the *netdyn* tool [2].

3 A Dynamical Model for Nonstationary Communication Networks

We now consider the problem of estimating time-varying delay distributions as defined in (1). We first formulate a model describing the evolution of the network delay dynamics. Finally, we describe a sequential Monte Carlo procedure for dynamic estimation.

The queuing delay experienced by a measurement packet on each link in the network is due to other packets in the queue(s) of the associated router(s). We assume a network in which each link is a direct connection between two routers and associate the delay on each link with a dedicated output queue at the router from which it emerges (*i.e.,* each outgoing link has its own dedicated queue). Each of these queues has a buffer size K with Markovian services at rate μ. Coupled with a homogeneous (constant rate) Poisson arrival process, this model is the standard M/M/1/K queue model [25]. The extension to heterogeneous networks (differing service rates and queue sizes) is straightforward. We assume that we make measurements (send packet-pairs) at a rate of $C_1 \mu K$ where $C_1 > 1$ is a constant. This ensures that there is sufficient time for the queues to relax between measurements, resulting in approximately statistically independent measurements.

Now, in the nonstationary setting, we model all other packet arrivals at a given queue using a time-varying (inhomogeneous) Poisson arrival process and assume that the bandwidth B of this process is limited such that

$$B < \frac{1}{2C_1 \mu K}. \tag{2}$$

This implies a *quasi*-stationarity; the dynamics of the system are evolving at a rate slow enough that we can discretize at the measurement rate (specifically where the measurements are made) and study the discretized system. Moreover, each measurement essentially encounters a classical M/M/1/K queue. We complete our model by imposing a random walk structure on the log-intensity of the traffic arrivals:

$$\log \lambda_i(m) = \log \lambda_i(m) + \epsilon(m), \tag{3}$$

where m denotes the m-th measurement, and $\epsilon(m)$ is zero-mean Gaussian noise of variance $\sigma_i^2(m)$. The model described thus far induces delay pmfs at each measurement time of the form

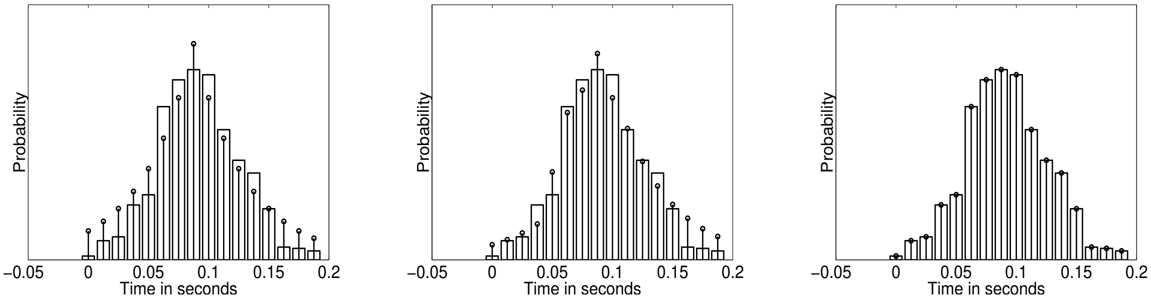
$$p_{i,j}(m) \quad \propto \quad \rho_i^j(m), \tag{4}$$

where $\rho_i(m)$ is the ratio of the arrival rate $\lambda_i(m)$ and service rate on the i-th link. Such pmfs are exponentially increasing or decreasing, for $\rho_i(m) > 1$ and $\rho_i(m) < 1$, respectively. This implies that the mode is either at delay 0 or delay K. Note that this model can provide an excellent fit to the delay histogram depicted in Figure 2 (a).

In real networks, however, the delay pmfs can display modes at other points due to the non-Poissonian nature of traffic and due to the fact that each link may include multiple "hidden" routers. A straightforward extension of the model above can handle these situations. We introduce an additional dynamical (continuous) parameter κ_i for each link and define the delay pmf as

$$p_{i,j} \ \propto \ \rho_i^{|j-\kappa_i|}, \tag{5}$$

which places the mode of the pmf near κ_i. The parameter κ_i evolves according to a continuous random walk (with reflection at 0 and K). In Figure 4 we illustrate the fit between the delay distribution from Figure 2 (b) and mixtures of pmfs of the form $p_{i,j} \propto 0.75^{|j-0.05\,i|}$, $0 \leq i \leq K$. It is not hard to see that if we choose $K+1$ distinct values for i, then the resulting vectors $p_i = [p_{i,0}, \ldots, p_{i,K}]$ are linearly independent, thus forming a basis for $\mathbb{R}^{L+1}$. Therefore, any pmf can be represented as a linear combination of these vectors, as the example in Figure 4(c) shows.



(a) (b) (c)

Figure 4: Fitting a the network delay histogram of Figure 2 (b) (boxes) with mixtures of pmfs of the form $p_{i,j} \propto 0.75^{|j-0.05\,i|}$, $0 \leq i \leq K$ (stems). (a) Fit using a single pmf. (b) Fit using a mixture of four pmfs. (c) Fit using a mixture of 16 pmfs.

4 Sequential Monte Carlo Tracking of Time-Variation

4.1 Basic Problem

We would like to track the internal delay distributions over time. More specifically, based on our measurements we wish to estimate the time-varying delay distribution defined in (1). We will focus on the posterior mean as our estimator. The posterior mean estimate of $p_{i,j}(R,m)$ can be written as:

$$\begin{aligned}
\widehat{p}_{i,j}(R,m) &:= \mathbf{E}_{p(\mathbf{z}_{m-R+1:m}|\mathbf{y}_{1:m})}\left[\sum_{l=m-R+1}^{m} \mathbf{1}_{\{z_i(l)=j\}}\right] \\
&= \frac{1}{R}\sum_{l=m-R+1}^{m} p(z_i(l)=j|\mathbf{y}_{1:m}) \\
&= \frac{1}{R}\sum_{l=m-R+1}^{m} \int p(z_i(l)=j|y(l),\boldsymbol{\lambda}_l)\; p(\boldsymbol{\lambda}_l|\mathbf{y}_{1:m})\, d\boldsymbol{\lambda}_l, \qquad (6)
\end{aligned}$$

where $y(l) \equiv [y_1(l), y_2(l)]$, $\boldsymbol{\lambda}_l$ is a vector composed of the traffic intensities on all links at time l, and $\mathbf{y}_{1:m}$ is a vector composed of the measurements at times $1,\ldots,m$. As before $z_i(l)$ is the (unobserved) delay on link i at time l, and $\mathbf{z}_{m-R+1:m} \equiv [z_i(m-R+1), \ldots, z_i(m)]$.

The evaluation of this estimator is very difficult. It requires an integration over the density $p(\boldsymbol{\lambda}_l|\mathbf{y}_{1:m})$, which cannot be solved analytically. It is necessary to adopt numerical integration techniques. Moreover, we need to calculate the estimate at each time m. It is important that we form our estimate $\widehat{p}_{i,j}(R,m)$ without redoing all the calculations involved in generating the estimate at time $m-1$. Otherwise we are not only wasting considerable computations, but we render a real-time implementation of our procedure impossible. These considerations necessitate the adoption of a sequential algorithm.

In the dynamic system we defined in Section 3, the available observations $\mathbf{y}_{1:m}$ are a highly non-linear function of the evolving parameters $\boldsymbol{\lambda}_{0:m}$. Standard sequential tracking methods such as the Kalman filter are not applicable; our attempts at linearisation (e.g., the extended Kalman filter) also result in very poor tracking. In previous work [13, 14, 24, 28], it has been observed that sequential Monte Carlo (SMC) procedures can perform well when facing such highly non-linear tracking problems. The estimation algorithm we develop in this section is based on the SMC methodology.

We begin by briefly outlining the Monte Carlo nature of the technique. Because the integral in (6) can not be calculated analytically, we approximate the estimator using Monte Carlo integration. To do this, we must sample from $p(\boldsymbol{\lambda}_l|\mathbf{y}_{1:m})$, which itself is not easily accomplished. An alternative approach is to perform *importance* sampling. Let $\boldsymbol{\lambda}_{0:m}$ denote the trajectories of the traffic intensities on all links over the time interval $0,\ldots,m$. The basic idea here is to generate N draws of $\boldsymbol{\lambda}_{0:m}$ from an *importance distribution* π_m, that has the same support as $p(\boldsymbol{\lambda}_{0:m}|y_{1:m})$ but from which we can sample more easily. We need to sample the entire trajectory $\boldsymbol{\lambda}_{0:m}$ rather than just $\boldsymbol{\lambda}_l$ because the trajectories are highly coupled (evaluating $p(\boldsymbol{\lambda}_l|y_{1:m})$ requires difficult marginalisation). Each draw represents an independent sample path of the network's dynamical evolution and thus independently explores part of the sample space. We use these draws (or *particles*) to compute the desired Monte Carlo integration as follows. We can re-write the integration as,

$$\int p(z_i(l)=j|y(l),\boldsymbol{\lambda}_l)\left[\frac{p(\boldsymbol{\lambda}_{0:m}|y_{1:m})}{\pi_k(\boldsymbol{\lambda}_{0:m}|y_{1:m})}\right]\pi_k(\boldsymbol{\lambda}_{0:m}|\mathbf{y}_{1:m})\,d\boldsymbol{\lambda}_l.$$

Then, the Monte Carlo estimate is

$$\sum_{v=1}^{N} p(z_i(l)=j|y(l),\boldsymbol{\lambda}_l^{(v)})\ \widetilde{w}_m^{(v)}, \tag{7}$$

where $w_m^{(v)} = p(\boldsymbol{\lambda}_{0:m}^{(v)}|\mathbf{y}_{1:m}) \,/\, \pi_m(\boldsymbol{\lambda}_{0:m}^{(v)}|\mathbf{y}_{1:m})$ and $\widetilde{w}_m^{(v)} = w_m^{(v)}\left[\sum_{s=1}^{N} w_m^{(s)}\right]^{-1}$. In order to evaluate this Monte Carlo estimate, we must determine both the weight $w_m^{(v)}$ (up to a proportionality constant) and the value of $p(z_i(l)=j|y(l),\boldsymbol{\lambda}_l^{(v)})$ for each of the N particles. We have $p(\boldsymbol{\lambda}_{0:m}^{(v)}|\mathbf{y}_{1:m}) \propto p(\mathbf{y}_{1:m}|\boldsymbol{\lambda}_{0:m}^{(v)})\,p(\boldsymbol{\lambda}_{0:m}^{(v)})$. As the measurements are independent, the likelihood in this expression can be decomposed as $p(\mathbf{y}_{1:m}|\boldsymbol{\lambda}^{(v)}_{0:m}) = \prod_{l=1:m} p(y(l)|\boldsymbol{\lambda}^{(v)}_l)$, where each factor in the product is a convolution of pmfs that can be evaluated efficiently using FFTs. The $p(\boldsymbol{\lambda}^{(v)}_{0:m})$ term can be determined from the dynamics of the system (3).

Evaluating $p(z_i(l) = j|y(l), \boldsymbol{\lambda}^{(v)}_l)$ involves the application of an upward-downward algorithm [16]. This algorithm propagates the knowledge of (1) the zero delay at the source and (2) the delays $y(l)$ at the two receivers throughout the tree, exploiting the independence of the conditional pmfs to calculate marginal distributions at each node.

4.2 Sequential Importance Sampling

The MC integration approach described above requires us to generate entire trajectories $\boldsymbol{\lambda}_{0:m}$ at each time m, and then to calculate the associated weight. This is computationally demanding and highly wasteful. At time m, we want to perform the integration without redoing calculations made at time $m-1$. This is achieved by forming the trajectory $\boldsymbol{\lambda}^{(v)}_{0:m}$ without modifying the previous trajectory $\boldsymbol{\lambda}^{(v)}_{0:m-1}$, which is possible if the importance sampling distribution has a Markovian structure. At time 0, we sample from the initial distribution $\pi_0(\boldsymbol{\lambda}_0)$. At time m, we sample from $\pi_m(\boldsymbol{\lambda}_m|\boldsymbol{\lambda}^{(v)}_{0:m-1}, \mathbf{y}_{1:m})$, and form the time-m particle v by appending $\boldsymbol{\lambda}^{(v)}_m$ to $\boldsymbol{\lambda}^{(v)}_{0:m-1}$. The weight of particle v at time m can then also be updated recursively:

$$w^{(v)}_m = \widetilde{w}^{(v)}_{m-1} \frac{p(y(m)|\boldsymbol{\lambda}^{(v)}_m)\ p(\boldsymbol{\lambda}^{(v)}_m|\boldsymbol{\lambda}^{(v)}_{m-1})}{\pi_m(\boldsymbol{\lambda}^{(v)}_m|\boldsymbol{\lambda}^{(v)}_{0:m-1}, \mathbf{y}_{1:m})} .$$

We form our approximate estimator, denoted $\widetilde{p}_{i,j}(R, m)$, by replacing the true integrals in (6) by their Monte Carlo approximations (7).

The dynamics of the model proposed in Section 3 involve a random walk of $\log \boldsymbol{\lambda}_m$. In this paper, we employ the prior distribution $p(\boldsymbol{\lambda}_m|\boldsymbol{\lambda}_{m-1})$ as the importance function (as adopted in [21] and many subsequent works). In this scenario, we merely need to calculate the likelihood to determine the update in the weights:

$$w^{(v)}_m = \widetilde{w}^{(v)}_{m-1}\ p(y(m)|\boldsymbol{\lambda}^{(v)}_m) . \tag{8}$$

The weight update factor at each time step is the likelihood $p(\mathbf{y}_m|\boldsymbol{\lambda}^{(v)}_m)$. This can be efficiently calculated using n_m FFTs, where n_m is the number of unique links traversed by the two packets involved in the m-th measurement. We discuss the complexity of our algorithm in more detail later in this section. Since we are dealing with discrete distributions, our weight update factor (8) is bounded above by 1, which implies that at any time m, every importance weight is bounded by 1.

The disadvantage of using the prior as the importance function is that the exploration of the state-space has the potential to be inefficient, as knowledge of the current observation is not used to guide the search. In Section 6, we discuss ways in which the sampling strategy can be improved; in practice, we observe that these approaches can produce slightly better performance.

Degeneracy is a major issue in the application of sequential importance sampling. The multiplicative update applied to the weight at each time means that some importance weights may quickly tend to zero, and the number of particles contributing to the estimator is greatly reduced. This effect increases the variability of the estimator (compared to the variance one would have with the full N particles contributing). We will say more on estimator variance in Section 4.4. The procedure of *resampling* aims to generate an unweighted approximation of the weighted particle distribution. When performed at time m, the procedure associates with each particle v a number of offspring $N_m^{(v)}$, such that $\sum_{v=1}^{N} N_m^{(v)} = N$. The procedure thus obtains a new set of particles, each of which has weight $1/N$, and ensures that the number of significant weights remains close to N. There are numerous techniques for performing resampling. The most popular is sampling importance resampling (SIR), introduced in [20]. SIR involves jointly drawing $\left\{ N_m^{(v)} \right\}_{v=1}^{N}$ according to a multinomial distribution of parameters N and $\left\{ \widetilde{w}_m^{(v)} \right\}_{v=1}^{N}$. Other techniques include residual resampling [22, 28], and stratified resampling [6, 24], which is adopted in this paper.

This resampling process does introduce some additional computational overhead in the formation of our approximate estimator at time m. Technically, it necessitates calculating the marginal smoothing distributions $p(\boldsymbol{\lambda}_l|\mathbf{y}_{1:m})$ for $l \in m - R + 1 \ldots m$. This can be done using the two-filter formula of [24], forward filtering–backward smoothing [14, 23], or the backwards simulation procedure [19].

In simulations, we observe that if we use the approximation (replace $\widetilde{w}_m^{(v)}$ by $\widetilde{w}_l^{(v)}$)

$$\sum_{v=1}^{N} p(z_i(l) = j|y(l), \boldsymbol{\lambda}_l^{(v)})\ \widetilde{w}_l^{(v)}, \tag{9}$$

for the summation in (7), then we achieve similar performance. We adopt this approximation in the algorithm we outline below:

Particle Filter for Delay Distribution Estimation

At time m:

Sequential Importance Sampling step

- For $v = 1, ..., N$, sample $\widetilde{\boldsymbol{\lambda}}_m^{(v)} \sim p(\boldsymbol{\lambda}_m|\,\boldsymbol{\lambda}_{m-1}^{(v)})$ and set $\widetilde{\boldsymbol{\lambda}}_{0:m}^{(v)} \triangleq \left(\boldsymbol{\lambda}_{0:m-1}^{(v)}, \widetilde{\boldsymbol{\lambda}}_m^{(v)}\right)$.

- For $v = 1, ..., N$, evaluate the importance weights $\widetilde{w}_m^{(v)}$:

$$w_m^{(v)} \propto p\left(y(l) \mid \widetilde{\boldsymbol{\lambda}}_m^{(v)}\right) \tag{10}$$

$$\widetilde{w}_m^{(v)} = \left[\sum_{s=1}^{N} w_m^{(s)}\right]^{-1} w_m^{(v)} \tag{11}$$

Selection step

- Apply stratified resampling [24] to obtain N new particles $\left(\boldsymbol{\lambda}_{0:m}^{(v)}; v = 1, \ldots, N\right)$, each with weight $1/N$.

Estimation step

- For all i, j :

  1. Evaluate $p(z_i(m) = j|y(m), \boldsymbol{\lambda}_m^{(v)})$ using the upwards-downwards probability propagation algorithm [16].
  2. Estimate $\widetilde{p}_{i,j}(R, m)$ from:

$$\widetilde{p}_{i,j}(R, m) := \frac{1}{R} \sum_{l=m-R+1}^{m} \sum_{v=1}^{N} p(z_i(l) = j|y(l), \boldsymbol{\lambda}_l^{(v)})\ \widetilde{w}_l^{(v)},$$

4.3 Computational Complexity Analysis

The computational complexity of the sequential importance sampling technique is $O(N)$ per measurement. Sampling from the prior distribution is straightforward and is of constant complexity for each particle. Updating the weights at each time step requires n_m FFTs, where n_m is the number of unique links traversed by the two packets in measurement m. The complexity of this procedure is $O(n_m K \log K)$ per particle, where K is the maximum number of delay units per link. The majority of resampling procedures, including the stratified resampling procedure we adopt, can be implemented in $O(N)$ operations. The chief overhead involved in the the algorithm is the application of the up-down algorithm to evaluate $p(z_i(m) = j|y(m), \boldsymbol{\lambda}_m^{(v)})$. For this step, the computational expense is $O(n_m K^2)$ per particle. As this is the dominant expense, the computational overhead of the approximate algorithm outlined above is $O(LK^2N)$ per measurement, where L is the average number of unique links involved in each measurement.

If we avoid the approximation (9) by calculating (at measurement m) the marginal smoothing distributions $p(\boldsymbol{\lambda}_l|\mathbf{y}_{1:m})$ for $l \in m-R+1 \ldots m$, then we introduce a substantial additional computational overhead. Of the available options, using the forward filtering—backward smoothing procedure [14, 23] minimizes the additional computational expense, but even in this case it is $O(RN^2)$ per measurement. For reasonable values of K (~ 40), L (~ 10) and R (~ 300), and a suitable number of particles (N =100–2000), this is by far the dominant computational overhead.

4.4 Convergence Analysis

When we use the prior distribution as the importance function, the importance weights are bounded above by 1 and continuous. The weights at time 0 are bounded by 1, and at any time m, the multiplicative weight update factor (8) involves a product of convolved discrete probabilities; it is easy to check that is bounded above by 1. Since we make use of the stratified resampling scheme and the importance distribution does not depend on the empirical distribution (the distribution of the particles), the assumptions 1–A and 2–A of [11] are satisfied, making Theorem 1 of [11] applicable for our algorithm. The theorem implies that the mean-squared error between the sequential Monte Carlo estimate $\widetilde{p}_{i,j}(R,m)$ and the posterior mean estimate $\widehat{p}_{i,j}(R,m)$ approaches zero as the number of particles increases, i.e.:

$$E\left[(\widetilde{p}_{i,j}(R,m) - \widehat{p}_{i,j}(R,m))^2\right] \leq C_{R,m} \frac{\left\|\sum_{l=m-R+1}^{m} p(z_i(l) = j | y(l), \boldsymbol{\lambda}_l)\right\|^2}{N} ,$$

where $C_{R,m}$ does not depend on N.

5 Experiments

To assess the performance of our algorithms we simulate (in Matlab) the four-receiver network depicted in Figure 1. In all experiments shown here we have set the maximum delay on each link $K = 15$ and used 500 particles. However, we have conducted tests in which the number of particles ranged between 200 and 5000, and observed similar performance over the entire range.

Experiment 1: We generate 1000 packet-pair measurements from stationary delay distributions on each link. Figure 5 (a) depicts the true delay distributions on links $2, \ldots, 7$ along with the estimated posterior means computed by the SMC algorithm. For comparison, Figure 5 (a) depicts the results obtained using the EM algorithm proposed in [10]). This same experiment is repeated in 50 independent trials. Figure 6 (a) shows the true expected delay for each link and the expected delay computed from the estimated posterior mean pmfs.

Experiment 2: We perform 50 independent trials of the scenario in Experiment 1, but this time introduce small, random discrepancies (errors of up to 4 time units) between the delays on shared links. Figure 6 (b) depicts the true average delay for each link and the average delay computed from the estimated pmfs (note the agreement with Figure 6 (a), indicative of the robustness of our methods to such errors). The estimated posterior mean pmfs (not shown here) are also very close to the true delay pmfs, similar in quality to those shown in Figure 6 (a).

Experiment 3: We generate 3000 packet-pair measurements from time-varying delay distributions. The temporal dynamics are governed by (3). Figure 7 depicts the true and estimated posterior mean pmfs on links 1, 2, and 7 at two different times. Figure 8 plots the true and estimated expected delay (both based on windowed averages) on links 2, 4, and 7 as a function of time.

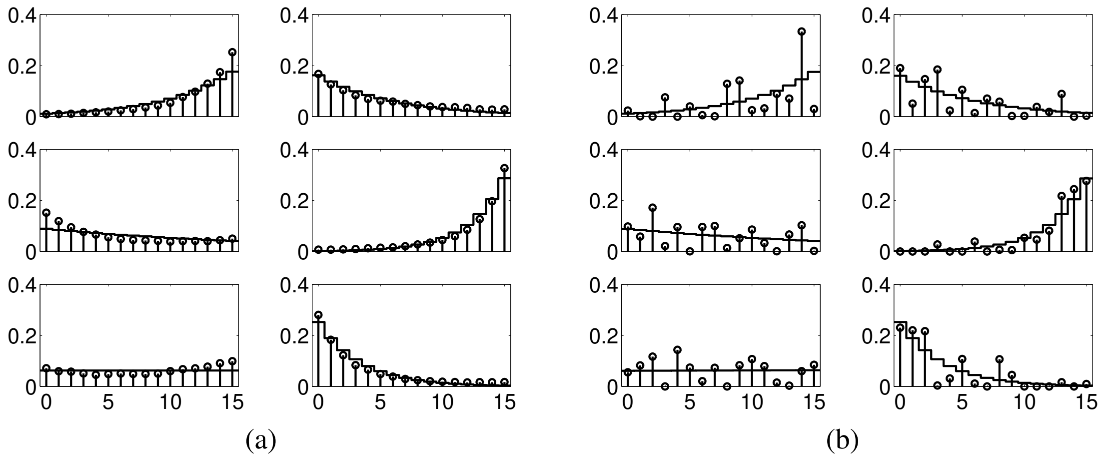


Figure 5: True (solid) and estimated (stem) delay pmfs for links 2 and 3 (row 1), 4 and 5 (row 2), and 6 and 7 (row 3) using the (a) sequential MC method developed in this paper and (b) EM algorithm [10].

6 Discussion

6.1 Experimental Results

Several conclusions can be drawn from the experimental results. First, it is clear from the results of Experiment 1 that the SMC procedure described here can offer significant improvements over the MLE approach [10]. Experiment 2 demonstrates that our estimator is quite robust to significant deviations from the assumption that the delays on shared links are identical, although we do observe a slightly larger variance in our estimates (see Figure 6). This is important in practice since, as evident in Figure 3, real Internet measurements exhibit small delay deviations on shared links. While the basic delay model (4) is quite adequate in some cases (*c.f.* Figure 2 (a)), Experiment 3 demonstrates that the generalized model corresponding to (5) enables the tracking and estimation of a much broader class of delay distributions. This too is very relevant to practice

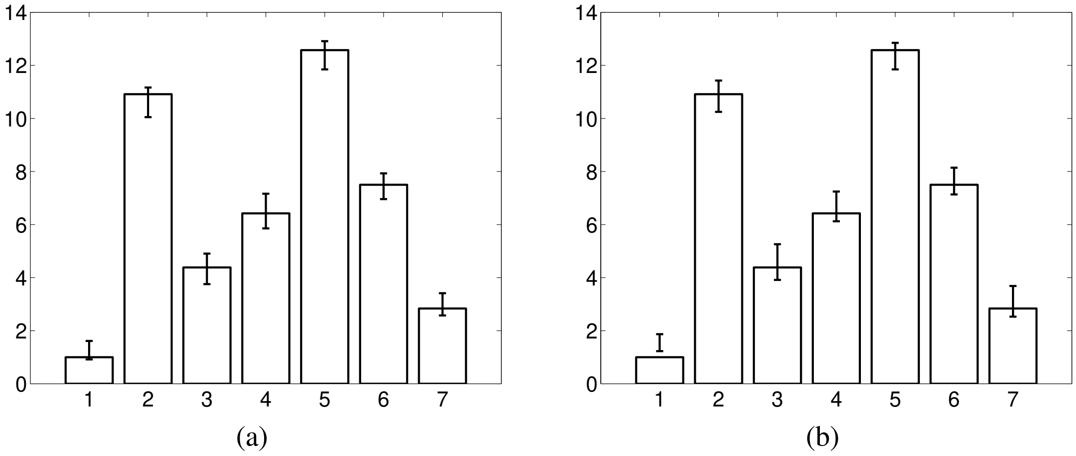


Figure 6: Estimation of average delays on each link for (a) identical delays on shared links and (b) small delay discrepancies on shared links. Boxes indicate the true average delay on each link (1-7). Error bars denote the one-standard-deviation confidence interval of the estimated average delay (using the EM algorithm).

because the delay distributions in the Internet often have more complicated characteristics (*c.f.* Figure 2 (b)).

6.2 Improved Importance Sampling

In our basic formulation developed in Section 4, we used the prior distribution in our importance sampling. The disadvantage of using the prior as the importance distribution is that the exploration of the state-space has the potential to be inefficient, as knowledge of the current observation is not used to guide the search. Improvements in our SMC procedure may be obtained by using the *optimal* importance distribution [14, 37]. Unfortunately, in the internal delay tracking problem, it is extremely difficult to sample from the optimal importance distribution. We can, however, achieve a slight improvement over the use of the prior distribution by considering a local linearization of the optimal distribution [36].

We consider the function $l(\boldsymbol{\lambda}_m) = \log\, p(\boldsymbol{\lambda}_m|\boldsymbol{\lambda}_{m-1}, y(m))$, and reparameterise using $\mathbf{r}_m = \log \boldsymbol{\lambda}_m$.We have:

$$
\begin{aligned}
l(\mathbf{r}_m) &= \ln p(y(m)\,|\mathbf{r}_m) + \ln p(\mathbf{r}_m|\mathbf{r}_{m-1}) - \ln p(y(m)) \\
&= \ln p(y(m)\,|\mathbf{r}_m) - \frac{1}{2\sigma^2}\left(\mathbf{r}_m - \mathbf{r}_{m-1}\right)^T\left(\mathbf{r}_m - \mathbf{r}_{m-1}\right) - \ln p(y_m).
\end{aligned}
$$

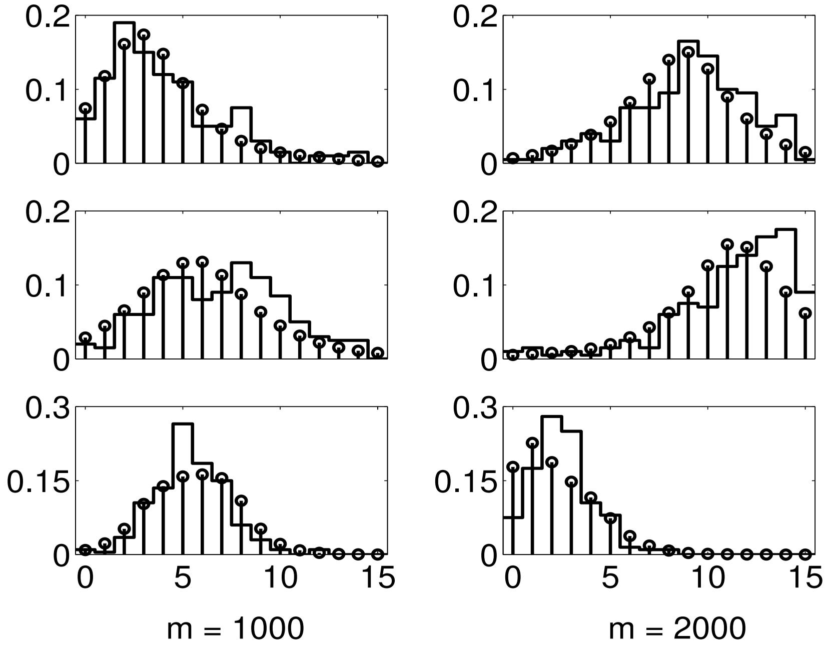


Figure 7: Delay pmf estimates in nonstationary scenario using sequential Monte Carlo procedure. True (solid) and estimated (stem) pmfs on links 1, 2, and 7 at measurements $m = 1000$ and $m = 2000$ for a window size $R = 200$.

Using a first-order Taylor expansion about $\mathbf{r}$, we have

$$\ln p(y(m)\,|\mathbf{r}_m) \;\simeq\; \ln p(y(m)\,|\mathbf{r}) + \left.\frac{\partial \ln p(y(m)\,|\mathbf{r}_m)}{\partial \mathbf{r}_m}\right|_{\mathbf{r}_m=\mathbf{r}} (\mathbf{r}_m - \mathbf{r})$$

If we choose $\mathbf{r} = \mathbf{r}_{m-1}$, then we can write

$$\begin{aligned} l(\mathbf{r}_m) \;&=\; \ln p(y(m)\,|\mathbf{r}_m = \mathbf{r}_{m-1}) \;+\; \left.\frac{\partial \ln p(y(m)\,|\mathbf{r}_m)}{\partial \mathbf{r}_m}\right|_{\mathbf{r}_m=\mathbf{r}_{m-1}} (\mathbf{r}_m - \mathbf{r}_{m-1}) \\ &\qquad -\frac{1}{2\sigma^2}\left(\mathbf{r}_m - \mathbf{r}_{m-1}\right)^T \left(\mathbf{r}_m - \mathbf{r}_{m-1}\right) + \text{constant} \\ &=\; \ln p(y(m)\,|\mathbf{r}_m = \mathbf{r}_{m-1}) + \text{constant} \\ &\qquad -\frac{1}{2\sigma^2}\left(\mathbf{r}_m - \mathbf{r}_{m-1} - \mu(\mathbf{r}_{m-1})\right)^T \left(\mathbf{r}_m - \mathbf{r}_{m-1} - \mu(\mathbf{r}_{m-1})\right), \end{aligned}$$

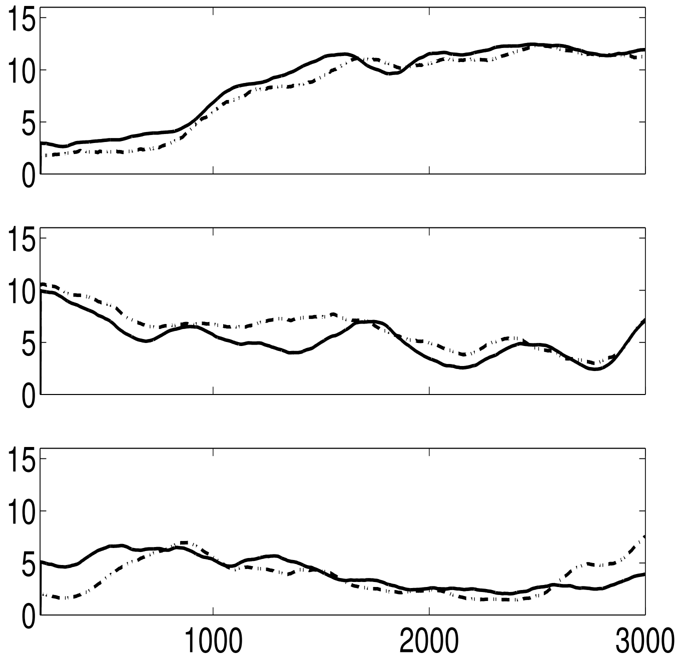


Figure 8: Tracking of average delay on links 2, 4, and 7 over measurement period. True (solid) and estimated (dashed) expected delay versus time. Both the true and estimated expected delays were calculated based on an $R = 200$ windowed average.

where

$$\mu(\mathbf{r}_{m-1}) = \sigma^2 \left. \frac{\partial \ln p(y(m)|\mathbf{r}_m)}{\partial \mathbf{r}_m} \right|_{\mathbf{r}_m = \mathbf{r}_{m-1}}.$$

This functional form suggests the adoption of the importance function:

$$\pi(\mathbf{r}_m|\mathbf{r}_{m-1}, y(m)) = \mathcal{N}\left(\mathbf{r}_{m-1} + \mu(\mathbf{r}_{m-1}), \sigma^2\right).$$

Sampling from this importance function, as opposed to the prior, involves the additional overhead of calculating $\mu(\mathbf{r}_{m-1}^{(v)})$ for each particle v. This is a similar computational task to the calculation of the likelihood, the chief expense being the convolution of the distributions (involving n_m FFTs, where n_m is the number of unique links traversed by the two packets in measurement m). As the probability propagation technique requires substantially more computation, this additional overhead has little effect on the speed of the algorithm.

6.3 Conclusions

Our experiments demonstrate the potential of sequential Monte Carlo algorithms for network delay tomography. We find that very good estimates of the delay pmfs can be obtained from a small number of measurements, and estimates of expected delays are very robust, even in the presence of non-ideal delay discrepancies on shared links. The sequential Monte Carlo algorithm appears to track slowly varying network behavior reasonably well.

Ongoing work is aimed at deeper theoretical analyses of our methods. There are several improvements and extensions possible for our framework including tracking of hyperparameters [27] (*e.g.,* variance in random walk underlying traffic intensities) and the more sophisticated importance sampling strategies such as the linearized resampling scheme proposed in Section 6.2, the auxiliary particle filter [30], and shemes incorporating local MCMC moves [17]. We are also conducting more realistic network simulation experiments with the *ns*-2 package [3].

More generally, our sequential modeling and inference framework could be adapted to dynamical problems arising in wireless and peer-to-peer commnication networks.

References

[1] Multicast-based inference of network-internal characteristics. gaia.cs.umass.edu/minc.

[2] *Netdyn probing tool.* http://www.cs.umd.edu/projects/netcalliper/NetDyn.html.

[3] The network simulator-2. http://www.isi.edu/nsnam/ns/.

[4] E. R. Beadle and P. M. Djuric. A fast weighted bayesian bootstrap filter for nonlinear model state estimation. *IEEE Transactions on Aerospace and Electronic Systems*, 33:338–343, 1997.

[5] R. Cáceres, N. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Trans. Info. Theory*, 45(7):2462–2480, November 1999.

[6] J. Carpenter, P. Clifford, and P. Fearnhead. Building robust simulation-based filters for evolving data sets. Technical report, University of Oxford, Dept. of Statistics, 1999.

[7] T.C. Clapp and S.J. Godsill. Fixed-lag smoothing using sequential importance sampling. *Bayesian Statistics VI*, pages 743–752, 1999.

[8] M. Coates and R. Nowak. Network loss inference using unicast end-to-end measurement. In *Proc. ITC Seminar on IP Traffic, Measurement and Modelling*, pages 28:1–9, Monterey, CA, Sep. 2000.

[9] M. Coates and R. Nowak. Networks for networks: Internet analysis using Bayesian graphical models. In *Proc. IEEE Neural Network for Signal Processing Workshop*, Sydney, Aust., Dec. 2000.

[10] M. Coates and R. Nowak. Network tomography for internal delay estimation. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Salt Lake City, UT, May 2001.

[11] D. Crisan and A. Doucet. Convergence of sequential Monte Carlo methods. Technical report, Cambridge University Engineering Department, Cambridge, England, 2000. Technical Report CUED/F-INFENG/TR381.

[12] P. Djuric. Sequential estimation of signals under model uncertainty. In *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[13] A. Doucet, N. de Freitas, and N.J. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Series Statistics for Engineering and Information Science. Springer-Verlag, New York, 2001.

[14] A. Doucet, S.J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statist. Computing*, 10:197–208, 2000.

[15] N.G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. to appear, *Proc. IEEE Infocom'01*, April 2001. Available as http://www.research.att.com/projects/minc/dlpt00.ps.

[16] B. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, Cambridge, Massachusetts, 1998.

[17] W. R. Gilks and C. Berzuini. Following a moving target – Monte Carlo inference for dynamic Bayesian models. Technical report, Univ. Pavia, Italy, 1998.

[18] S. J. Godsill, A. Doucet, and M. West. Methodology for Monte Carlo smoothing with application to time-varying autoregressions. In *Symposium on Frontiers of Time Series Modelling*, Tokyo, 2000. Institute of Statistical Mathematics.

[19] S.J. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. Technical report, Institute of Statistics and Decision Sciences, Duke University, 2000.

[20] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F*, 140:107–113, 1993.

[21] J.E. Handschin and D.Q. Mayne. Monte Carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *Int. J. Cont.*, 9(5):547–559, 1969.

[22] T. Higuchi. Monte Carlo filter using the genetic algorithm operators. *J. Statist. Comput. Simul.*, 59:1–23, 1997.

[23] M. Hürzeler and H.R. Künsch. Monte Carlo approximation for general state space models. *J. Comp. Graph. Statist.*, 7:175–193, 1998.

[24] G. Kitagawa. Monte Carlo filter and smoother for nonlinear non-Gaussian state space models. *J. Comp. Graph. Statist.*, 5:1–25, 1996.

[25] L. Kleinrock. *Queueing Systems. Volume I: Theory*. Wiley & Sons, New York, 1975.

[26] J. Kurose and K. Ross. *Computer Networking*. Addison Wesley, 2001.

[27] J. Liu and M. West. Combined parameter and state estimation in simulation-based filtering. In *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.

[28] J.S. Liu and R. Chen. Sequential Monte Carlo methods for dynamic systems. *J. Am. Statist. Ass.*, 93:1032–1044, 1998.

[29] F. LoPresti, N.G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal delay distributions. Technical report, UMass CMPSCI 99-55, 1999.

[30] M. K. Pitt and N. Shephard. Filtering via simulation: Auxiliary particle filters. *J. Am. Statist. Assoc.*, 94:590–599, 1999.

[31] S. Ratnasamy and S. McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *Proceedings of INFOCOM '99*, New York, NY, March.

[32] C. Tebaldi and M. West. Bayesian inference on network traffic using link count data (with discussion). *J. Amer. Stat. Assoc.*, pages 557–576, June 1998.

[33] Y. Tsang, M. Coates, and R. Nowak. Passive network tomography using EM algorithms. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Salt Lake City, UT, May 2001.

[34] S. Vander Wiel, J. Cao, D. Davis, and B. Yu. Time-varying network tomography: router link data. In *Proc. Symposium on the Interface: Computing Science and Statistics*, Schaumburg, IL, June 1999.

[35] Y. Vardi. Network tomography: estimating source-destination traffic intensities from link data. *J. Amer. Stat. Assoc.*, pages 365–377, 1996.

[36] M. West and J. F. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer-Verlag, 1997.

[37] V. S. Zaritskii, V. B. Svetnik, and L. I. Shimelevich. Monte Carlo technique in problems of optimal data processing. *Auto. Remo. Cont.*, 12:95–103, 1975.