

# Continuous-time Hidden Markov Models for Network Performance Evaluation<sup>1</sup>

Wei Wei, Bing Wang, Don Towsley  
 Tech. Report 02-15  
 Department of Computer Science  
 University of Massachusetts, Amherst, MA 01003

## Abstract

In this paper, we study the use of continuous-time hidden Markov models for network protocol and application performance evaluation. We develop an algorithm to infer the continuous-time hidden Markov model from a series of end-to-end delay and loss observations of probe packets. This model can then be used to simulate network environments for network performance evaluation. We validate the simulation method by a series of experiments both in *ns* and over the Internet. Our experimental results show that this simulation method can represent a wide range of *real* network scenarios. It is easy to use, accurate, and time efficient.

## I. INTRODUCTION

Simulation is a common approach to evaluate a network protocol or application. An event driven simulator requires the specification of the topology as well as the parameters of every component of the simulated network. The setting of these parameters requires fine-tuning and needs to be representative of a real world scenario, which is a non-trivial task. Furthermore, simulation of very large networks can be very difficult due to excessive memory and CPU time requirements. Consequently, there have been considerable efforts to speed up event-driven simulation. For example, [?] provides a means to distribute an *ns* [?] simulation on several connected workstations. The Scalable Simulation Framework (SSF) aims to transparently utilize parallel processor resources and scale to very large collection of simulated entities [?]. Fluid simulation is another speedup technique that makes simplified assumptions about the system and is studied in [?, ?]. Compared to a simulator, an emulation package has the advantage of using real traffic generators. Dummynet [?] and NIST Net [?] are two such examples. Although still requiring users to specify some parameters such as propagation delay and loss rate, they provide greater transparency: from the user's point of view, the network is a black box simulated by the emulator. The challenge here is how to make the black box a good model of a real network. For instance, Dummynet uses an independent uniform random loss model, which differs from the correlated loss observations made by several studies [?, ?].

In this paper, we propose the use of a continuous-time hidden Markov model (continuous-time HMM) for network performance evaluation. We infer a continuous-time HMM from delay and loss observations seen by

<sup>1</sup>This research was supported in part by the National Science Foundation under NSF grants ANI-0085848, ANI-9973092, EIA-0080119 and by DARPA under contract F30602-00-2-0554. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

a sequence of probes sent from one end host to another end host. This HMM can then be incorporated into a simulator or an emulator to drive the simulation of a network protocol or application by providing losses and delays to packets in the network at arbitrary points of time. By collecting probe traces in a wide variety of network settings, one can construct a library of continuous-time HMMs, with each model representing a particular network setting. A model can then be selected to simulate a network protocol or application under a particular network environment. This simulation method can thus provide users a simulation environment representing a wide range of *real* network scenarios.

We carry out experiments in *ns* and over the Internet to validate this simulation method. The experiments provide us validation results in both controlled and real network environments. We demonstrate that the continuous-time HMM is a good model of the network settings by showing that the behavior of a flow driven by the model is similar to that of a flow in the original network. Here the flow can be governed by a network protocol or an application. We validate for both TCP and a streaming video application. Our experiments show that this simulation method is accurate, and time and space efficient. For a simulation time of 3000 seconds, the running time using this method is around 2 minutes on a Pentium 4 regardless of the complexity of the network (topology, number of flows, etc.) being simulated. A simulation of a similar three-router scenario in *ns* can take over 30 minutes.

Previous work on network modeling focus on the loss behavior and use discrete-time models [?, ?, ?]. Yajnik, etc. use a  $k$ -th order Markov model for the temporal dependence in packet loss [?]. Salamatian and Vaton use a discrete-time HMM to model packet loss in network channels and show that the number of states required is significantly less than using  $k$ -th order Markov model [?]. For the purpose of network protocol or application simulation, the delay characteristics also have to be modeled. Furthermore, it is necessary to assign a delay or a loss to a packet at an arbitrary point of time. This cannot be provided by a discrete-time model.

The rest of the paper is organized as follows. The inference of continuous-time HMM is described in Section II. Section III describes how to obtain the most likely state sequence given a continuous-time HMM and some observations. Section IV provides numerical validation of the model. Section V describes the experimental methodologies. Section VI and VII describe the validation of this simulation method in *ns* and over the Internet, focusing on TCP and a streaming video application respectively. Finally Section VIII concludes the paper and describes future work.

## II. INFERENCE OF THE CONTINUOUS-TIME HIDDEN MARKOV MODEL

In this section, we assume that a network in steady state can be modeled by a continuous-time HMM and describe how the parameters of such a model can be inferred given a sequence of observations. These observations are losses or delays seen by a sequence of probes sent from one end host to another end host in the network.

Because delays can take arbitrary nonnegative real values, we discretize them into a finite set of values. We shall refer to these values along with loss as a set of observation symbols. Let us suppose that the network can be modeled as a continuous-time HMM with  $M$  distinct observation symbols and  $N$  hidden states. Here, a hidden state cannot be observed directly but is reflected by the observations.

Let  $\{Z(t)\}_{t \geq 0}$  be the continuous-time HMM that we wish to learn. Each state  $Z(t)$  contains two components: the hidden state  $X(t) \in \{1, 2, \dots, N\}$  and the observation symbol  $Y(t) \in \{1, 2, \dots, M\}$ . That is  $Z(t) = (X(t), Y(t))$ . Let  $S = \{1, 2, \dots, N\} \times \{1, 2, \dots, M\}$ . The stochastic process  $\{Z(t)\}_{t \geq 0}$  takes values in  $S$  and is governed by a single infinitesimal generator of size  $MN \times MN$ . Denote this infinitesimal generator as  $Q = [q_{ij}]$ , where  $q_{ij} \geq 0$ ,  $q_{ii} = -\sum_{j \neq i} q_{ij}$ ,  $i, j \in S$ .

We develop a mechanism to infer the infinitesimal generator  $Q$ , given the sequence of discrete-time observations  $\{Y_t\}_{t=1}^T$ , where  $T$  is the length of the sequence. The value of  $Y_t$  is either an observation or unknown, and the time difference between two adjacent observations is  $\Delta$ . Since the observations are at discrete times, we start from deriving a probability transition matrix reflected by the observations. Let  $P(\Delta) = [p_{ij}(\Delta)]$  denote the probability transition matrix derived from the continuous-time Markov chain, where  $p_{ij}(\Delta)$  is the probability that the Markov chain, currently in state  $i$ , is in state  $j$  after an additional time  $\Delta$ , where  $i, j \in S$  [?]. Without confusion, we let  $P$  represent  $P(\Delta)$  in the rest of the paper for simplicity. Our first step is to derive the probability transition matrix  $P$  using the observation sequence. Our second step is to calculate the infinitesimal generator  $Q$  from the probability transition matrix  $P$ . We next describe the two steps in detail.

#### A. An EM algorithm to infer the discrete-time model

We consider a discrete-time model derived from the continuous-time Markov chain with the probability transition matrix  $P$  and initial distribution  $\pi$ . For convenience, let  $\lambda = (P, \pi)$ . We next describe the procedure to infer  $\lambda$  from a sequence of  $T$  observations.

When the observation sequence is obtained by periodic probing, our algorithm to infer the parameter  $\lambda$  is derived from the EM algorithm in [?] by representing the hidden state and the observation symbol together as a state. We refer to this as the complete-data EM algorithm. In some situations, it is impossible to obtain a periodic probing sequence. For instance, if one of the hosts is behind a firewall, we may have to use TCP probes, which cannot be guaranteed to follow a specified probing process. For this case, we assume the existence of a minimum time interval  $\Delta$  such that the intervals between observations are multiples of  $\Delta$ . The probe sequence can then be regarded as a periodic sequence with period  $\Delta$  with some missing data points. We develop a missing-data EM algorithm to deal with this situation.

We next describe the missing-data EM algorithm; the complete-data EM algorithm being a special case of it.

We first define some notations conforming to those used in [?]. Let  $(i, j)$  represent a state with hidden component  $i$  and observation symbol  $j$ . An element in the transition matrix  $P$  is denoted as  $p_{(i,j)(k,l)}$  representing the probability of transition from state  $(i, j)$  to state  $(k, l)$ . Define  $\alpha_t(i, j)$  to be the probability of the observation sequence up to time  $t$  and the state being in  $(i, j)$  at time  $t$ , given  $\lambda$ . That is

$$\alpha_t(i, j) = P(Y_1 = y_1, Y_2 = y_2, \dots, Y_t = y_t, Z_t = (i, j) \mid \lambda)$$

Define  $\beta_t(i, j)$  to be the probability of the observation sequence from time  $t + 1$  to  $T$ , given state being in  $(i, j)$  at time  $t$ , given  $\lambda$ . That is

$$\beta_t(i, j) = P(Y_{t+1} = y_{t+1}, \dots, Y_T = y_T \mid Z_t = (i, j), \lambda)$$

Define  $\xi_t(i, j, k, l)$  to be the probability of state being in  $(i, j)$  at time  $t$  and in  $(k, l)$  at time  $t + 1$ , given the observation sequence and  $\lambda$ . Define  $\gamma_t(i, j)$  to be the probability of being in state  $(i, j)$  at time  $t$ , given the observation sequence and  $\lambda$ . That is

$$\xi_t(i, j, k, l) = P(Z_t = (i, j), Z_{t+1} = (k, l) \mid Y_1 = y_1, Y_2 = y_2, \dots, Y_T = y_T, \lambda)$$

$$\gamma_t(i, j) = P(Z_t = (i, j) \mid Y_1 = y_1, Y_2 = y_2, \dots, Y_T = y_T, \lambda)$$

We derive  $\xi_t(i, j, k, l)$  from  $\alpha_t(i, j)$  and  $\beta_{t+1}(k, l)$  as follows

$$\xi_t(i, j, k, l) = \frac{\alpha_t(i, j)p_{(i,j)(k,l)}\beta_{t+1}(k, l)}{\sum_{i=1}^N \sum_{j=1}^M \sum_{k=1}^N \sum_{l=1}^M \alpha_t(i, j)p_{(i,j)(k,l)}\beta_{t+1}(k, l)} \quad (1)$$

Observe that  $\gamma_t$  can be calculated from  $\xi_t$  as

$$\gamma_t(i, j) = \sum_{k=1}^N \sum_{l=1}^M \xi_t(i, j, k, l). \quad (2)$$

The EM algorithm is an iterative algorithm in which each iteration consist of two steps: the expectation step and the maximization step. During the expectation step, we compute the expected number of transitions from state  $(i, j)$  and the expected number of transitions from state  $(i, j)$  to state  $(k, l)$  using the model parameters obtained during the previous iteration. Denote these as  $n(i, j)$  and  $m(i, j, k, l)$  respectively. During the maximization step, we calculate the new model parameters from  $n(i, j)$  and  $m(i, j, k, l)$ . The iteration ends when the difference between the new model and the previous model parameters are less than a certain threshold.

Without loss of generality, we assume the observation values for  $Y_1$  and  $Y_T$  are known. Let  $y_t$  be the observation value for  $Y_t$ . We say  $y_t = u$  if the observation for  $Y_t$  is missing. In the expectation step, we first calculate  $\alpha$

and  $\beta$  using the procedures as follows. The procedure to calculate  $\alpha_t(i, j)$ , where  $1 \leq t \leq T, 1 \leq i \leq N, 1 \leq j \leq M$ , consists of the following steps:

1. Initialization

$$\alpha_1(i, j) = \begin{cases} \pi(i, y_1), & j = y_1. \\ 0, & j \neq y_1. \end{cases}$$

2. Induction

$$\alpha_{t+1}(i, j) = \begin{cases} 0, & y_{t+1} \neq u, j \neq y_{t+1} \\ \sum_{k=1}^N \sum_{l=1}^M \alpha_t(k, l) p_{(k,l)}(i, j), & \text{Otherwise} \end{cases}$$

where  $t = 1, 2, 3, \dots, T - 1$ .

The derivation of the induction step is shown below:

$$\begin{aligned} \alpha_{t+1}(i, j) &= P(Y_1 = y_1, \dots, Y_t = y_t, Y_{t+1} = y_{t+1}, X_{t+1} = i, Y_{t+1} = j \mid \lambda) \\ &= \sum_{k=1}^N P(Y_1 = y_1, \dots, Y_t = y_t, Y_{t+1} = y_{t+1}, X_t = k, Y_{t+1} = j, X_{t+1} = i \mid \lambda) \\ &= \sum_{k=1}^N \sum_{l=1}^M P(Y_1 = y_1, \dots, Y_t = l, Y_{t+1} = y_{t+1}, X_t = k, Y_{t+1} = j, X_{t+1} = i \mid \lambda) \\ &= \sum_{k=1}^N \sum_{l=1}^M P(Y_{t+1} = y_{t+1}, Y_{t+1} = j, X_{t+1} = i \mid Y_1 = y_1, \dots, Y_t = l, X_t = k, \lambda) \\ &\quad P(Y_1 = y_1, \dots, Y_t = l, X_t = k \mid \lambda) \\ &= \sum_{k=1}^N \sum_{l=1}^M \alpha_t(k, l) P(Y_{t+1} = y_{t+1}, Y_{t+1} = j, X_{t+1} = i \mid Y_t = l, X_t = k, \lambda) \end{aligned}$$

where the second term is either 0 or  $p_{(i,j)}(k,l)$  depending on  $y_{t+1}$  and  $j$ , hence the induction formula.

The procedure to calculate  $\beta_t(i, j)$ , where  $1 \leq t \leq T, 1 \leq i \leq N, 1 \leq j \leq M$ , contains the following steps:

1. Initialization

$$\beta_T(i, j) = \begin{cases} 1, & j = y_T. \\ 0, & j \neq y_T. \end{cases}$$

2. Induction

$$\beta_t(i, j) = \begin{cases} 0, & y_t \neq u, j \neq y_t \\ \sum_{k=1}^N \sum_{l=1}^M p_{(i,j)}(k,l) \beta_{t+1}(k, l), & \text{Otherwise} \end{cases}$$

where  $t = T - 1, T - 2, \dots, 1$ .

The derivation of the induction step is shown below:

$$\begin{aligned} \beta_t(i, j) &= P(Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, \dots, Y_T = y_T \mid X_t = i, Y_t = y_t, Y_t = j, \lambda) \\ &= \sum_{k=1}^N P(X_{t+1} = k, Y_{t+1} = y_{t+1}, Y_{t+2} = y_{t+2}, \dots, Y_T = y_T \mid X_t = i, Y_t = y_t, Y_t = j, \lambda) \\ &= \sum_{k=1}^N \sum_{l=1}^M P(X_{t+1} = k, Y_{t+1} = l, Y_{t+2} = y_{t+2}, \dots, Y_T = y_T \mid X_t = i, Y_t = y_t, Y_t = j, \lambda) \\ &= \sum_{k=1}^N \sum_{l=1}^M P(Y_{t+2} = y_{t+2}, \dots, Y_T = y_T \mid X_{t+1} = k, Y_{t+1} = l, X_t = i, Y_t = y_t, Y_t = j, \lambda) \\ &\quad P(Y_{t+1} = l, X_{t+1} = k \mid X_t = i, Y_t = y_t, Y_t = j, \lambda) \end{aligned}$$

$$= \sum_{k=1}^N \sum_{l=1}^M \beta_{t+1}(k, l) P(Y_{t+1} = l, X_{t+1} = k \mid X_t = i, Y_t = y_t, Y_t = j, \lambda)$$

where the second term is either 0 or  $p_{(i,j)(k,l)}$  depending on  $y_t$  and  $j$ , hence the induction formula.

Once  $\alpha$  and  $\beta$  are obtained, we calculate  $\xi$  and  $\gamma$  using (1) and (2). Then we calculate  $n(i, j)$  and  $m(i, j, k, l)$  from  $\gamma_t(i, j)$  and  $\xi_t(i, j, k, l)$  as follows

$$n(i, j) = \sum_{t=1}^{T-1} \gamma_t(i, j) \quad (3)$$

$$m(i, j, k, l) = \sum_{t=1}^{T-1} \xi_t(i, j, k, l) \quad (4)$$

The new model parameter estimates are obtained in the maximization step as follows

$$\hat{p}_{(i,j)(k,l)} = \frac{m(i, j, k, l)}{n(i, j)} \quad (5)$$

$$\hat{\pi}_{(i,j)} = \gamma_1(i, j) \quad (6)$$

## B. Obtain the infinitesimal generator $Q$ from the transition matrix $P$

We next describe two methods to obtain the infinitesimal generator  $Q$  from the transition matrix  $P$ . The first method uses infinite second while the second one is an iterative method. We find, in practice, the condition of convergence for the first method is less stringent. However, when convergent, the solution obtained from the two methods are the same since the solution is unique. All the results in the experiments are obtained using the first method.

### B.1 Method 1: using infinite sum to obtain $Q$ from $P$

By solving the Kolmogorov equations, we obtain the following relation between  $P$  and  $Q$

$$P = e^{\Delta Q} \quad (7)$$

Therefore, when  $\ln(P)$  exists,  $Q$  can be calculated as follows

$$Q = \frac{1}{\Delta} \ln(P) \quad (8)$$

We extend the Taylor expansion  $\ln(x) = \sum_{n=1}^{\infty} (-1)^{n-1} (x-1)^n / n$ ,  $0 < x < 2$  to matrices and define

$$\ln(P) = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{(P-I)^n}{n} \quad (9)$$

when the right hand side of (9) converges, where  $I$  is the identity matrix, we have

$$Q = \frac{1}{\Delta} \sum_{n=1}^{\infty} (-1)^{n-1} \frac{(P-I)^n}{n} \quad (10)$$

In this paper, we consider the complete metric space of matrices of size  $MN \times MN$  with the maximum absolute row sum norm as the metric [?]. We next state a theorem describing the convergence condition of (9).

*Theorem 1:* The right hand side of (9) converges if  $\forall i, p_{ii} > \frac{1}{2}$ .

*Proof:* Define  $f_n(P) = \sum_{i=1}^n \frac{(-1)^{i-1}(P-I)^i}{i}$ . We prove that  $f_n(P)$  converges when  $n \rightarrow \infty$  by showing that  $f_n(P)$  is a Cauchy sequence. First, because  $\forall i, p_{ii} > \frac{1}{2}$ , we have

$$\|P - I\| = \max_i (|p_{ii} - 1| + \sum_{j \neq i} p_{ij}) = \max_i (1 - p_{ii} + 1 - p_{ii}) = 2 \max_i (1 - p_{ii}) = 2(1 - \min_i p_{ii}) < 1$$

Without loss of generality, assume  $m \leq n$ , then

$$\begin{aligned} \|f_m(P) - f_n(P)\| &= \left\| \sum_{i=m+1}^n \frac{(-1)^{i-1}(P-I)^i}{i} \right\| \leq \sum_{i=m+1}^n \frac{\|(P-I)^i\|}{i} \leq \sum_{i=m+1}^n \frac{\|P-I\|^i}{i} \leq \sum_{i=m+1}^n \|P-I\|^i \\ &= \|P-I\|^{m+1} \sum_{i=0}^{n-m-1} \|P-I\|^i \leq \|P-I\|^{m+1} \sum_{i=0}^{\infty} \|P-I\|^i = \frac{\|P-I\|^{m+1}}{1 - \|P-I\|} \end{aligned}$$

For any  $\epsilon > 0$ , to satisfy  $\frac{\|P-I\|^{m+1}}{1 - \|P-I\|} < \epsilon$ , we need  $\ln(\|P-I\|^{m+1}) < \ln \epsilon + \ln(1 - \|P-I\|)$ . Therefore,

$$m > \frac{\ln \epsilon + \ln(1 - \|P-I\|)}{\ln \|P-I\|} - 1$$

Hence,  $\forall \epsilon > 0, \exists N = \lceil \frac{\ln \epsilon + \ln(1 - \|P-I\|)}{\ln \|P-I\|} \rceil - 1$ , such that for all  $m, n > N$ ,  $\|f_m(P) - f_n(P)\| < \epsilon$ . Therefore,  $f_n(P)$  is a Cauchy sequence. Thus  $f_n(P)$  converges by the completeness of the metric space.  $\blacksquare$

Note that the convergence condition of Theorem 1 is sufficient but not necessary. Our experiments show that very often convergence holds even when the elements on the diagonal of matrix  $P$  are much less than  $\frac{1}{2}$ . We next state a theorem which provides a much relaxed convergence condition than Theorem 1.

*Theorem 2:* The right hand side of (9) converges if  $\exists c > 0$  and  $\delta > 0$ , such that there exists  $N = N(P, \delta, c)$  satisfying  $\forall n > N, \|(P-I)^n\| \leq \frac{c}{n^\delta}$ .

*Proof:* Since  $\forall n > N, \|(P-I)^n\| \leq c/n^\delta$ , we have

$$\left\| \sum_{n=N+1}^{\infty} (-1)^{n-1} \frac{(P-I)^n}{n} \right\| \leq \sum_{n=N+1}^{\infty} \frac{\|(P-I)^n\|}{n} \leq \sum_{n=N+1}^{\infty} \frac{c}{n^{1+\delta}}$$

$\epsilon$	$\min p_{ii}$				
	0.51	0.6	0.7	0.8	0.9
0.001	536	39	16	9	5
0.0001	650	49	20	11	6
0.00001	764	59	25	14	8
0.000001	878	70	29	16	8

TABLE I

$\epsilon$ ,  $\min p_{ii}$  AND THE NUMBER OF ITEMS REQUIRED.

Since  $\sum_{n=1}^{\infty} \frac{1}{n^{1+\delta}}$  converges,  $\sum_{n=1}^{\infty} (-1)^{n-1} \frac{(P-I)^n}{n}$  converges. ■

In calculation, we use finite sum to approximate the infinite sum in (9). In the following, we state a theorem describing the number of items required in the sum so that the approximation error is less than a threshold  $\epsilon$ .

*Theorem 3:* If for all  $i$ ,  $p_{ii} > \frac{1}{2}$ , then  $\forall \epsilon > 0, \exists N = \left\lceil \frac{\ln \epsilon + \ln(1 - \|P-I\|)}{\ln \|P-I\|} \right\rceil - 1$  such that  $\forall n > N$ ,

$$\left\| \sum_{i=n+1}^{\infty} \frac{(-1)^{i-1} (P-I)^i}{i} \right\| < \epsilon.$$

*Proof:*

$$\begin{aligned} \left\| \sum_{i=n+1}^{\infty} \frac{(-1)^{i-1} (P-I)^i}{i} \right\| &\leq \sum_{i=n+1}^{\infty} \frac{\|(P-I)^i\|}{i} \leq \sum_{i=n+1}^{\infty} \frac{\|P-I\|^i}{i} \\ &\leq \sum_{i=n+1}^{\infty} \|P-I\|^i = \|P-I\|^{n+1} \sum_{i=0}^{\infty} \|P-I\|^i = \frac{\|P-I\|^{n+1}}{1 - \|P-I\|} \end{aligned}$$

When  $n > N = \left\lceil \frac{\ln \epsilon + \ln(1 - \|P-I\|)}{\ln \|P-I\|} \right\rceil - 1$ , we have  $n > \frac{\ln \epsilon + \ln(1 - \|P-I\|)}{\ln \|P-I\|} - 1$ , therefore,  $\ln(\|P-I\|^{n+1}) < \ln \epsilon + \ln(1 - \|P-I\|)$ . This implies  $\frac{\|P-I\|^{n+1}}{1 - \|P-I\|} < \epsilon$ . Hence,  $\left\| \sum_{i=n+1}^{\infty} \frac{(-1)^{i-1} (P-I)^i}{i} \right\| < \epsilon$ . ■

As we have shown in the proof of Theorem 1,  $\|P-I\| = 2(1 - \min p_{ii})$ . Table I shows the number of items required for different settings of  $\epsilon$  and  $\min p_{ii}$ . We observe that the number of items required is an increasing function of  $\min p_{ii}$  and a decreasing function of  $\epsilon$ . We observe that for  $\min p_{ii} \geq 0.6$ , to achieve an error threshold of  $10^{-6}$ , the number of items required is no more than 70. However, when  $\min p_{ii}$  is 0.51, for the same error threshold, the number of items required increases dramatically to 878. We also notice that the number of items required for a fixed  $\min p_{ii}$  does not increase significantly to achieve a higher precision. Therefore, the computation is determined mainly by the matrix instead of the precision required.

When the right hand side of (9) does not converge, we approximate the infinitesimal generator by using

$$Q = \frac{1}{\Delta}(P-I) \tag{11}$$

Our numerical and experimental validations show that the approximation is reasonably good.

### C. Method 2: An Iterative method to obtain $Q$ from $P$

We next describe an iterative method to obtain the infinitesimal generator  $Q$  from the transition matrix  $P$ . This method is motivated by uniformization [?], which is a device to calculate the transition probability matrix  $P$  from the infinitesimal generator  $Q$  as follows:

Let  $v_i = -q_{ii}, i \in S$ , which describes the rate at which the process leaves state  $i$ . Choose  $v$  such that  $v \geq v_i, \forall i \in S$ . Define a  $MN \times MN$  matrix  $P^*$  as follows

$$P_{ij}^* = \begin{cases} 1 - \frac{v_i}{v}, & j = i \\ \frac{q_{ij}}{v}, & j \neq i \end{cases} \quad (12)$$

Then the discrete-time Markov chain transition probability matrix  $P$  can be calculated as

$$P(\Delta) = \sum_{n=0}^{\infty} e^{-v\Delta} \frac{(v\Delta)^n}{n!} P^{*n} \quad (13)$$

where  $P^{*n}$  are the  $n$ -stage transition probability matrix of  $P^*$ .

To achieve the inverse process of uniformization, we develop the following mechanism and refer to it as the inverse of the uniformization. Observe that  $P^*$  can be obtained iteratively from (12) as follows

$$P_{k+1}^* = \frac{1}{V\Delta} (e^{V\Delta} P(\Delta) - I - \sum_{n=2}^{\infty} \frac{(V\Delta)^n}{n!} P_k^{*n}) \quad (14)$$

where the initial matrix  $P_0^*$  is given and  $P_k^*$  represents  $P^*$  at the  $k$ -th iteration. Note that we do not know  $v$  beforehand. In practice, we choose a rate  $V$  and plug it into (14). We describe later how to choose  $V$  to obtain convergent results. The iteration ends when  $\|P_{k+1}^* - P_k^*\| \leq \epsilon$ , where  $\epsilon$  is the threshold for convergence. After  $P^*$  is obtained, the infinitesimal generator  $Q$  can be calculated from  $P^*$  as

$$q_{ij} = \begin{cases} -V(1 - P_{ii}^*), & j = i \\ VP_{ij}^*, & j \neq i \end{cases} \quad (15)$$

We next present a theorem describing the condition under which the iteration of (14) converges. Before that, we first present some lemmas. As in Section II-B.1, we use the maximum absolute row sum norm as matrix norm. Let  $X$  and  $Y$  be matrices. We use the following properties of matrix norm in the proof:

1.  $\|X\| > 0$  when  $X \neq 0$  and  $\|X\| = 0$  iff  $X = 0$ .
2.  $\|kX\| = |k|\|X\|$  for any scalar  $k$ .
3.  $\|X + Y\| \leq \|X\| + \|Y\|$ .
4.  $\|XY\| \leq \|X\|\|Y\|$ .

*Lemma 1:*  $\|X^n\| = 1, n = 1, 2, \dots$ .

*Proof:* For any  $n$ ,  $X^n$  is a transition matrix. sum of each row of the matrix is 1. By definition of maximum absolute row sum norm,  $\|X^n\| = 1$ . ■

*Lemma 2:*  $\|X^n - Y^n\| \leq n\|X - Y\|, n = 1, 2, \dots$ .

*Proof:* We prove the lemma by induction. First, for  $n = 1$ ,  $\|X^n - Y^n\| \leq n\|X - Y\|$  is obvious.

Second, suppose  $\|X^k - Y^k\| \leq k\|X - Y\|, k > 1$ ,

then  $\|X^{k+1} - Y^{k+1}\|$

$$\begin{aligned}
&= \|X^{k+1} - XY^k + XY^k - Y^{k+1}\| \\
&\leq \|X(X^k - Y^k)\| + \|(X - Y)Y^k\| \text{(Property 3)} \\
&\leq \|X\|\|X^k - Y^k\| + \|(X - Y)\| \|Y^k\| \text{(Property 4)} \\
&= \|X^k - Y^k\| + \|X - Y\| \text{(Lemma 1)} \\
&\leq k\|X - Y\| + \|X - Y\| \text{(Assumption)} \\
&= (k + 1)\|X - Y\|.
\end{aligned}$$

By induction,  $\|X^n - Y^n\| \leq n\|X - Y\|, n = 1, 2, \dots$ . ■

We next use contraction mapping theorem to prove the convergence of the inverse of uniformization. Let

$$f(X) = \frac{1}{V\Delta} (e^{V\Delta} P(\Delta) - I - \sum_{n=2}^{\infty} \frac{(V\Delta)^n}{n!} X^n).$$

*Theorem 4:* If  $V(\Delta) < \frac{\ln 2}{\Delta}$ ,  $f(X)$  has a unique fixed point.

*Proof:* We will show when  $V\Delta < \ln 2$ ,  $f(X)$  is a contraction mapping by proving  $\exists \alpha, 0 < \alpha < 1$ , such that  $\forall X, \forall Y, \|f(X) - f(Y)\| < \alpha\|X - Y\|$ .

$$\begin{aligned}
\|f(X) - f(Y)\| &= \frac{1}{V\Delta} \sum_{n=2}^{\infty} \frac{(V\Delta)^n}{n!} \|X^n - Y^n\| \\
&\leq \frac{1}{V\Delta} \sum_{n=2}^{\infty} \frac{(V\Delta)^n}{(n-1)!} \|X - Y\| \text{(Lemma 2)} \\
&= \sum_{n=2}^{\infty} \frac{(V\Delta)^{n-1}}{(n-1)!} \|X - Y\| \\
&= \sum_{n=1}^{\infty} \frac{(V\Delta)^n}{n!} \|X - Y\| \\
&= (\sum_{n=0}^{\infty} \frac{(V\Delta)^n}{n!} - 1) \|X - Y\| \\
&= (e^{V\Delta} - 1) \|X - Y\|
\end{aligned}$$

Since  $V\Delta < \ln 2$ ,  $e^{V\Delta} - 1 < 1$ . Let  $\alpha = e^{V\Delta} - 1$ , we have  $\|f(X) - f(Y)\| < \alpha\|X - Y\|$ , by contraction mapping theorem, there exist a unique fixed point  $X_0$ , such that  $f(X_0) = X_0$ .  $\blacksquare$

Our experiments show that the convergent condition given by Theorem 4 is tight. In the calculation, we set  $V\Delta$  to be 1.99. The maximum rate in the resultant infinitesimal generator is no more than  $V$  from the inverse of the uniformization procedure. Let  $v(Q)$  denote the largest rate in the actual infinitesimal generator  $Q$ . By Theorem 4, if  $v(Q) \leq V$ , the procedure obtains the unique solution; if  $v(Q) > V$ , the procedure provides an approximate solution, with the maximum rate of  $V$ .

### III. MOST LIKELY STATE SEQUENCE

In some situations, after the continuous-time HMM is obtained, we need to obtain the most likely state sequence given the model and some observations. We solve this problem by dynamic programming. Our algorithm differs from the Viterbi Algorithm [?] in that not all the observations are known and our model is a continuous-time instead of a discrete-time model.

Let  $\{\tau_t\}_{t=1}^T$  be a sequence of arrival times, with their corresponding state sequence as  $\{(X_t, Y_t)\}_{t=1}^T$ . At time  $\tau_t$ , the observation symbol  $Y_t$  is either known as  $y_t$  or not known. Our goal is to obtain the Most Likely Estimate (MLE) of the state sequence. Let  $L_t(k, l)$  to be the highest likelihood value along all the paths, at time  $\tau_t$ , which accounts for all the observations up to time  $\tau_t$  and ends at the state  $(k, l)$ . Let  $\Delta_{t-1}$  be the time interval between time  $\tau_t$  and  $\tau_{t-1}$ . That is  $\Delta_{t-1} = \tau_t - \tau_{t-1}$ . Let  $P(\Delta_{t-1})$  be the transition matrix for time interval  $\Delta_{t-1}$ . It is obtained from the continuous-time HMM by uniformization. Then we have

$$L_{t+1}(k, l) = \max_{1 \leq i \leq N, 1 \leq j \leq M} L_t(i, j) P_{(i,j)(k,l)}(\Delta_{t-1}).$$

This formula gives us a recursive way to obtain the most likely state sequence  $\{X_t, Y_t\}_{t=1}^T$  using dynamic programming. To actually retrieve the state sequence, we use  $(\psi_t^X(k, l), \psi_t^Y(k, l))$  to keep track of the argument which maximizes  $L_{t+1}(k, l)$ , for each time point  $\tau_t$  and state  $(k, l)$ . Without loss of generality, we assume that the first observation  $Y_1$  and the last observation  $Y_T$  are known. The complete procedure to find the most likely state sequence is stated as follows:

#### 1. Initialization:

For  $1 \leq i \leq N, 1 \leq j \leq M$ ,

$$L_1(i, j) = \begin{cases} \pi(i, j), & j = y_1 \\ 0, & j \neq y_1 \end{cases}$$

$$(\psi_1^X(i, j), \psi_1^Y(i, j)) = (0, 0).$$

## 2. Recursion:

For  $2 \leq t \leq T$  and  $1 \leq i, k \leq N, 1 \leq j, l \leq M$ , if  $Y_{t-1}$  is known to be  $y_{t-1}$ ,

$$L_t(k, l) = \max_{1 \leq i \leq N} L_{t-1}(i, y_{t-1}) p_{(i, y_{t-1})(k, l)}(\Delta_{t-1})$$

$$\psi_t^X(k, l) = \arg \max_{1 \leq i \leq N} L_{t-1}(i, y_{t-1}) p_{(i, y_{t-1})(k, l)}(\Delta_{t-1})$$

If  $Y_{t-1}$  is unknown,

$$L_t(k, l) = \max_{1 \leq i \leq N, 1 \leq j \leq M} L_{t-1}(i, j) p_{(i, j)(k, l)}(\Delta_{t-1})$$

$$(\psi_t^X(k, l), \psi_t^Y(k, l)) = \arg \max_{1 \leq i \leq N, 1 \leq j \leq M} L_{t-1}(i, j) p_{(i, j)(k, l)}(\Delta_{t-1})$$

## 3. Termination:

$$L = \max_{1 \leq i \leq N} L_T(i, y_T)$$

$$X_T = \arg \max_{1 \leq i \leq N} L_T(i, y_T)$$

## 4. Path (state sequence) backtracking:

For  $t = T - 1, T - 2, \dots, 1$ ,

$$X_t = \psi_{t+1}^X(X_{t+1}, Y_{t+1})$$

$$Y_t = \begin{cases} y_t, & Y_t \text{ known} \\ \psi_{t+1}^Y(X_{t+1}, Y_{t+1}), & Y_t \text{ unknown} \end{cases}$$

## IV. NUMERICAL VALIDATION OF THE MODEL

In this section, we apply the model inference procedure to observation traces generated by known continuous-time HMMs. After a model is inferred, we examine if it is close to the original model for validation. We first describe the results from one model in detail. The model  $Q$  we use is shown below:

$$Q = \begin{bmatrix} -4 & 2 & 1 & 1 & 0 & 0 \\ 1 & -5 & 3 & 0 & 1 & 0 \\ 2 & 3 & -6 & 0 & 0 & 1 \\ 2 & 0 & 0 & -7 & 2 & 3 \\ 0 & 2 & 0 & 3 & -8 & 3 \\ 0 & 0 & 2 & 2 & 5 & -9 \end{bmatrix}$$

We use the model to generate a 1000-second long trace. Using sampling intervals of 1ms and 100ms respectively, we obtain observation sequences  $O_1$  (of length 1,000,000) and  $O_{100}$  (of length 10,000) respectively. The sequence  $O_{100}$  is used for model inference. The sequence  $O_1$  is used to check the quality of the inferred continuous-time HMM, as described later. Since we cannot observe the number of hidden states  $N$  directly from the observation sequence  $O_{100}$ , we explore the sensitivity of the models to the choice of  $N$ . In particular we consider  $N$  in  $\{2, 3, \dots, 10\}$  for model inference.

We observe that the inferred continuous-time HMM is not guaranteed to be exactly the same as the original model, since the results of the EM algorithm are local maxima and the inference is from a discrete-time sample of the original model. For example, the inferred infinitesimal generator  $\hat{Q}$  when  $M = 3$  and  $N = 2$  is different from the original model  $Q$  as shown below

$$\hat{Q} = \begin{bmatrix} -3.24 & 1.57 & 0.03 & 0.01 & 0.00 & 1.64 \\ 1.53 & -4.44 & 0.05 & 0.01 & 0.00 & 2.85 \\ 2.07 & 0.00 & -7.67 & 0.00 & 5.53 & 0.07 \\ 0.00 & 1.20 & 1.43 & -4.56 & 1.93 & 0.00 \\ 0.00 & 2.81 & 3.49 & 3.00 & -9.29 & 0.00 \\ 0.06 & 0.00 & 0.33 & 2.32 & 2.97 & -5.67 \end{bmatrix}$$

However, the inferred models are very close to the original model in the following two senses. First, inferred models generate sequences whose autocorrelation functions are very close to that of a sequence generated by the original model. Fig. 1 shows the autocorrelation functions of a sequence of length 10,000 generated from the inferred models for various values of  $N$  and that of the original sequence  $O_{100}$ . The lag is in multiples of 100ms. The models for  $N = 4$  and  $N = 6$  are from (9) while the model for  $N = 9$  is from the approximation (11). We observe that the autocorrelation functions of different models are very similar to each other.

Secondly, symbols lying between observations can be estimated from an inferred model accurately. For instance, we obtain the MLE sequence at the interval of 1ms using the sequence  $O_{100}$  and an inferred model. Our results show that 82% of the MLE symbols match the observation symbols in  $O_1$ . Moreover, most of the MLE sequences for different values of  $N$  are identical, demonstrating the similarity among the inferred models.

We generate another 9 models for further validation. Each model has 2 hidden states and 3 observation symbols. The transition rates of each model are randomly chosen between 1 and 10. For each model, we investigate the range of  $N$  from 2 to 10 for a total of 81 experiments. Among them, (9) converges for 63 cases. In the remaining 18 cases, the sum in (9) diverges mostly for high values of  $N$  ( $N \geq 6$ ). There is only one situation where the divergence occurs for  $N = 4$ . The reason why higher value of  $N$  leads to divergent results can be explained as follows. The number of states in the continuous-time HMM increases linearly with  $N$ . For large values of  $N$ , the probability of the process remaining in a given state after one transition tends to be very small. Although

	$N=2$	$N=3$	$N=4$	$N=5$	$N=6$	$N=7$	$N=8$	$N=9$	$N=10$
test 1	82%	82%	82%	82%	82%	82%	82%	82%*	82%*
test 2	82%	82%	82%	82%	82%*	82%*	82%*	82%*	82%*
test 3	88%	88%	88%	88%	88%*	88%*	88%	88%	88%*
test 4	90%	90%	90%	90%	90%	90%	90%	90%	90%
test 5	90%	90%	90%	90%	90%	90%	90%	90%	90%
test 6	89%	89%	89%	89%	89%	89%	89%	90%*	90%*
test 7	91%	91%	91%	91%	91%	91%	91%	91%	91%*
test 8	89%	89%	89%	89%	89%	89%	89%	89%*	89%
test 9	85%	85%	85%	85%	85%	85%	85%*	85%*	85%*
test 10	86%	86%	86%	86%	86%	86%	86%	86%*	86%*

TABLE II  
NUMERICAL RESULTS

	$N=2$	$N=3$	$N=4$	$N=5$	$N=6$	$N=7$	$N=8$	$N=9$	$N=10$
test 11	90%	90%	90%	90%	90%	90%*	90%*	90%	90%
test 12	87%	87%	87%	87%	87%*	87%	87%	87%	87%*

TABLE III  
NUMERICAL RESULTS FOR SMALLER  $N$

Theorem 1 is not a necessary condition for convergence, very small values of  $p_{ii}$  tends to lead to divergence in (9). For the MLE sequences obtained from the models, 82% to 90% of the MLE symbols match the observation symbols in  $O_1$ . Furthermore, the MLE sequences from the models for different values of  $N$  are mostly identical. Table III shows the ratio of the MLE symbols matching the observation symbols in  $O_1$  for all the models used in the 10 tests. Items marked by \* are obtained by the approximation (11).

To examine the behavior of an inferred model of size smaller than the original model, we carry out the following validation. We generate two models containing 6 hidden states and 3 observation symbols. The transition rates of both models are randomly chosen between 1 and 10. We consider  $N$  in  $\{2, 3, \dots, 10\}$  for model inference. Our algorithm produces convergent results except at  $N = 7, 8$  for one model and at  $N = 7, 10$  for the other. For both models, the MLE sequences from the inferred models for different values of  $N$  are identical and around 90% of the MLE symbols match the observation symbols in  $O_1$ . The results are shown in table II. Items marked by \* are obtained by the approximation (11).

In summary, our numerical validation shows that, for a given model, the models inferred by our algorithm for different values of  $N$  are close to the original model and close to each other.

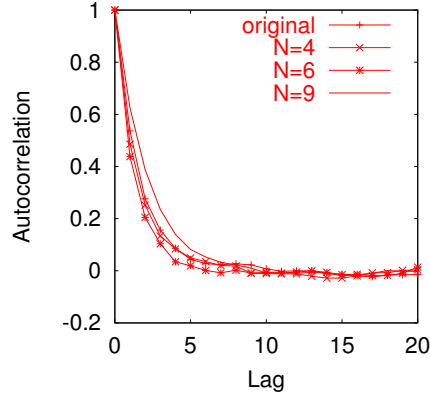


Fig. 1. Autocorrelation functions for the original and the various models,  $M = 3$ .

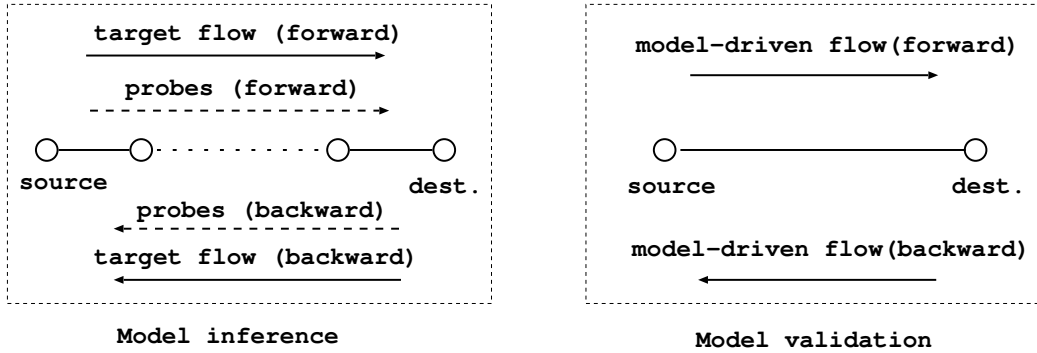


Fig. 2. Model inference and validation

## V. EXPERIMENTAL METHODOLOGIES

In this section, we describe the experimental methodologies used to validate the simulation method. Our validation is based on a series of experiments carried out in *ns* and over the Internet. In *ns*, we investigate in a variety of settings by specifying the traffic and topology parameters. Over the Internet, we examine several settings characterizing different delay and loss characteristics. We believe that validations over these experiments can provide us a comprehensive examination of the model.

As shown in the model inference part of Fig. 2, in each experiment, we send probes from the source to the destination along with a *target* flow. This target flow is governed by a network protocol or an application. When necessary, probe packets are also sent along the backward direction of the target flow. In this way, the propagation delays experienced by the probe packets are the same as the packets in the target flow. The observations of the forward and backward probes are used to infer continuous-time HMMs for the forward and backward paths

respectively. Following that, the models are used to provide packets with end-to-end losses and delays in a flow, as shown in the model validation part of Fig. 2. We validate this simulation method by examining the behavior of the target flow in the original setting with the model-driven flow. Note that the link controlled by the model represents the whole path in the original network. Hence the running time required by our simulation method is independent of the complexity of the network being simulated.

The probes are either sent periodically with the spacing of  $\Delta$  or  $l\Delta$ , where  $l$  is a geometrically distributed random variable. We refer to these as periodic probing and geometric probing respectively. For periodic probing,  $\Delta$  is set to 20ms, 50ms or 100ms. For geometric probing, let  $\tau$  be the average inter-sending time. We set  $\tau$  to be 20ms, 50ms or 100ms and  $\Delta$  to 1ms or 10ms. The random variable  $l$  thus has a geometric distribution with the parameter of  $\Delta/\tau$ . We next describe the experimental methodologies in detail.

#### A. Loss and delay as symbols

The continuous-time HMM uses  $M$  symbols to represent the observed delay and loss sequence. One way to discretize the delay and loss sequence is to use  $(M - 1)$  symbols to represent delay and a single symbol to represent loss. Another way is to combine loss and large delays into one symbol since large delays and loss are closely related. We find that the first method requires very fine probing interval since the duration of a loss is shorter than the processing time of one packet. We therefore adopt the second approach. More specifically, we divide the range of delay values into  $M$  equal length bins. A delay value falling in the  $i$ th ( $1 \leq i \leq M$ ) bin is represented by symbol  $i$ . More formally, suppose we have a probing observation sequence  $\{d_t\}_{t=1}^T$ . Let the minimum delay and the maximum delay be  $d_{min}$  and  $d_{max}$  respectively. We use  $d_t = \infty$  to denote that the observation at  $t$  is a loss. We say  $d_t = u$  if the observation at  $t$  is missing. The interval  $[d_{min}, d_{max}]$  is divided into  $M$  bins. The length of each bin is denoted as  $b$ . Then  $b = (d_{max} - d_{min})/M$ . We convert the delay and loss observation sequence to a symbol sequence  $\{y_t\}_{t=1}^T$  as follows

$$y_t = \begin{cases} M, & d_t = \infty, \\ \lceil \frac{d_t - d_{min}}{b} \rceil, & d_{min} \leq d_t \leq d_{max}, \\ u, & d_t = u. \end{cases}$$

At the same time, we record the probability of loss in symbol  $M$  since symbol  $M$  represents both loss and delays. Note that, in this way, we regard that loss is only correlated with the largest delay symbol  $M$ . In practice, the correlation of loss and large delays may not be strong in some situations, for instance, when RED is used for queue management in the routers. We extend the EM algorithm in Section II-A so that there is a component of loss associated with each symbol. We next describe the procedure in detail.

When a loss occurs, we denote its corresponding observation symbol to be  $v$  instead of  $M$ , as in the previous

method. We use  $s(j)$  to denote the probability that a loss occurs given that the symbol is  $j$ . We define  $\hat{s}_k(j)$  to be the estimate of  $s(j)$  at the  $k$ -th iteration. The estimate of  $\hat{s}_{k+1}(j)$  can be calculated as follows:

$$\begin{aligned}\hat{s}_{k+1}(j) &= \frac{\text{expected number of times that symbol being } j \text{ and a loss occurs}}{\text{expected number of symbol } j} \\ &= \frac{\sum_{\{t|y_t=u\}} \hat{s}_k(j) \sum_{i=1}^N \gamma_t(i, j) + \sum_{\{t|y_t=v\}} \sum_{i=1}^N \gamma_t(i, j)}{\sum_{t=1}^T \sum_{i=1}^N \gamma_t(i, j)}\end{aligned}$$

The induction of  $\alpha_{t+1}(i, j)$  is modified as follows:

$$\alpha_{t+1}(i, j) = \begin{cases} 0, & y_{t+1} \neq u, y_{t+1} \neq v, j \neq y_{t+1} \\ \frac{\sum_{k=1}^N \sum_{l=1}^M \alpha_t(k, l) p_{(k,l)}(i, j) s(j),}{\sum_{k=1}^N \sum_{l=1}^M \alpha_t(k, l) p_{(k,l)}(i, j)}, & y_{t+1} = v \\ \sum_{k=1}^N \sum_{l=1}^M \alpha_t(k, l) p_{(k,l)}(i, j), & \text{Otherwise} \end{cases}$$

where  $t = 1, 2, 3, \dots, T - 1$ . In our evaluation, unless explicitly stated, we use the method which combines loss and delay into the largest symbol  $M$ .

### B. Mapping symbols to delay and loss values

Once a continuous-time HMM is inferred, we can determine an observation symbol for any point of time from the model. For the purpose of simulation, this symbol has to be mapped back into delay or loss. When mapping a symbol  $i$  to delay or loss, if  $i = M$ , we first decide if it is a loss according to the loss probability in symbol  $M$  (in the method which associates loss to each symbol, this procedure applies to all the symbols). If it is not a loss, we generate a delay for it according to the conditional distribution of delays given that the symbol is  $i$ , which is obtained from the probing trace. In doing so, we occasionally generate out-of-order packets. To avoid this, we modify the delay generated so that packets are in order and use a procedure to maintain the correct distribution at the same time. The details are described as follows.

Suppose a packet is sent from the source at time  $t$ . Denote the delay to be assigned to it as  $d_t$ . Suppose that the symbol for this packet is  $j$ , which is determined by the model. To avoiding generating out-of-order packets, we record the arrival time and the assigned delay of the packet previous to this packet. Let them be  $t'$  and  $d_{t'}$  respectively. The conditional distribution of delays given symbol being  $j$  obtained from the probing trace is referred as the empirical delay distribution and is denoted as  $f_j$ . During the model-driven simulation, all the delays assigned up to time  $t$  for packets given symbol  $j$  form a conditional probability distribution and is denoted as  $g_j(t)$ . Our goal is to make  $g_j(t)$  conform to  $f_j$  when  $t$  is sufficiently large for each  $j$ . To achieve this, we obtain the difference of  $g_j(t)$  and  $f_j$  and normalize it to become a distribution, denoted as  $h_j(t)$ . That is,  $h_j(t)$  is the normalization of  $\max\{f_j - g_j(t), 0\}$ , where  $h_j(0) = f_j$ . The delay value is generated according to  $h_j(t)$ .

Intuitively, if the probability for some delay value is over the empirical value, the probability of choosing this value is 0. Otherwise, the more it is below the empirical value, the more likely it is to be chosen at time  $t$ . Let the delay generated according to  $h_j(t)$  be  $d_t^*$ . If  $t + d_t^* \geq t' + d_{t'}$ , that is, setting  $d_t$  to be  $d_t^*$  does not lead to out-of-order packets, then  $d_t = d_t^*$ . Otherwise,  $d_t$  is modified to be  $t' + d_{t'} - t$  so that this packet arrives back-to-back with the previous packet. When  $d_t$  is determined, this delay value is taken into account to update  $g_j(t)$ . In our experiments, 92% to 99% of the delays after modification are still in the delay range corresponding to the assigned symbol when the number of observation symbols  $M$  is between 2 and 6.

### C. Stationary segments

The model inference is based on the assumption that the network being studied has reached a steady state. We check if a network is in steady state by examining the variance of the loss rate in the probe trace, similar to [?]. We calculate the average loss ratio over an interval of 10 or 50 seconds to form a sequence of loss rates. The cumulative average of the loss rates is then plotted for visual inspection. An abrupt increase (or decrease) or a trend of increase (or decrease) in the cumulative average indicates non-stationarity. A stationary segment is selected to infer the model. In *ns* experiments, the network setting reaches a steady state after a sufficiently long running time. In experiments over the Internet, we divide the probe trace into segments of approximately 1200 seconds and check for stationarity. More rigorous approaches for stationarity test such as runtime tests or reverse placement test can be found in [?].

### D. Usage of virtual probes in ns

For a given setting, we need to investigate the simulation results over a range of probing intervals. In order to save time and space, and to make the results at different probing intervals comparable, we use virtual probes to study a setting with a single link first. Unlike a real probe, a virtual probe does not generate traffic. It records the loss or delay value at a certain point of time by analyzing the *ns* traces. A virtual probe sees a loss if the packet before it is dropped. Otherwise, the delay seen by the probe is calculated by keeping track of the enqueue time and processing time of all of the previous packets. This simulation method allows us to impose “probes” at a range of probing intervals to discover the granularity of the probing required for a network setting. We, for instance, confirmed that very fine probing intervals is required if loss is represented as a single symbol as described in Section V-A.

### E. Measurement of one-way delay over the Internet

In our experiments over the Internet, a probe packet is timestamped when it is sent out from the sender and when it reaches the receiver. If the clocks at the two end hosts are synchronized, the subtraction of the two

timestamps is the one-way delay. However, we do not have an external universal time reference to synchronize the two clocks. Given this, we use the method proposed in [?] to remove clock offset and skew. The resulting delay sequence represents the variable portion of the one-way delays over the network. To obtain the actual delay value, a constant portion has to be added, which contains the one-way transmission delay, propagation delay, and the minimum queuing delay experienced by the probe packets during the chosen time period. We describe the estimation of this value when describing the experiments (in Section VI-B and VII). The data outliers are adjusted before the model inference procedure. That is, we identify an outlier by drawing a boxplot of the data and adjust the outliers to the whisker close to it [?]. In our experiments, the percentage of outlier adjustment is less than 2%.

## VI. EXPERIMENTAL VALIDATIONS FOR TCP

In this section, we validate the simulation method by a series of experiments for TCP carried out in *ns* and over the Internet. That is, in each experiment, we send probes along with a target TCP flow on the forward and backward paths. The forward and backward probes are used to infer the models for the forward and backward paths respectively. The inferred models are then used to provide delay and loss to packets along the forward and backward paths of the model-driven TCP flow. We set the range of  $M$  from 2 to 6 and the range of  $N$  from 2 to 9.

The metrics we use are *average loss rate*, *average throughput*, and *autocorrelation function of the throughput sequence*. The average loss rate records the loss ratio of data packets from the TCP source. The average throughput records the amount of data reaching the TCP sink from the source. The autocorrelation function of the throughput sequence records the temporal dependence of throughput in the TCP flow.

### A. Validation in ns

We next validate our simulation method over experiments under a variety of network settings in *ns*. We study topologies containing a single link and multiple links. The traffic combinations include infinite TCP sources only, infinite TCP sources with on-off UDP sources, and infinite TCP sources with HTTP sources. The target TCP flow uses Reno or SACK. For all the settings we examined, the loss and throughput predictions made by the model-driven simulation falls within 10% of those of the target TCP flow in the original setting if  $M$  and  $N$  are chosen properly. We observe that our inferred model performs better for TCP with the presence of some random traffics (e.g. UDP on-off sources or HTTP sources). We next describe the experimental results.

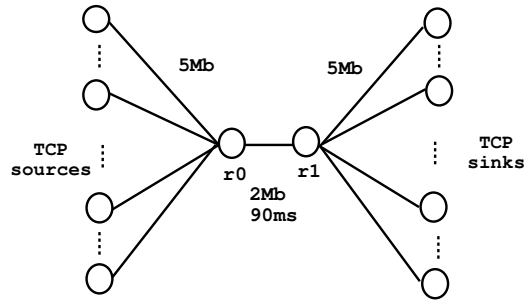


Fig. 3. Infinite TCP flows with a single link in *ns*.

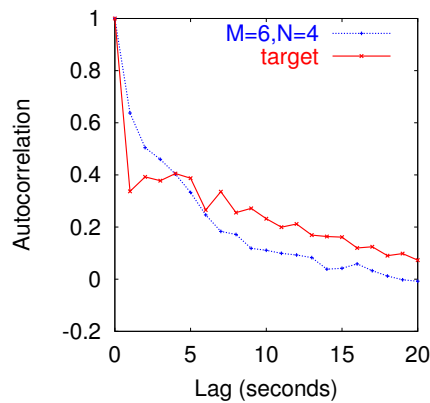


Fig. 4. Autocorrelation function of throughput for infinite TCP flows with a single router in *ns*.

#### A.1 Infinite TCP flows only with a single link

The only form of traffic in this setting is infinite TCP flows, as shown in Fig. 3. There are 15 TCP sources connected to router  $r_0$  with their corresponding sinks connected to  $r_1$ . Each TCP source uses TCP SACK, with a maximum window size of 20. All packets are 1000 bytes in size. The link between each TCP source/sink and the corresponding router has a bandwidth of 5Mb and a propagation delay of 10ms. The link between router  $r_0$  and  $r_1$  forms a bottleneck link with the bandwidth of 2Mb, the maximum queue length of 200 packets, and the propagation delay of 90ms. The TCP flows are started randomly in the range of  $[0.1, 0.5]$ s. We choose a TCP flow as the target flow and send a series of probe packets at regular intervals of  $\Delta$  along with the target TCP flow. Here  $\Delta$  is 20ms.

The average loss rate and throughput of the target TCP flow is 0.776% and 0.126Mbps. The errors of both throughput and loss rate of the model-driven TCP flow relative to the target TCP are shown in Table IV. We observe that the relative errors of throughput and loss are within 10% and 6% respectively.

Fig. 4 shows the autocorrelation function of throughput for the model-driven TCP and the target TCP, where

$M$	$N$	throughput error	loss error
2	2	-9.1%	0.7%
3	2	-10.1%	5.6%
4	2	-2.9%	-5.6%
5	2	-3.1%	-5.4%
6	2	-3.4%	-1.0%
2	3	-9.5%	1.1%
3	3	-7.4	-0.2%
4	3	-5.2%	-1.2%
6	3	-4.1%	-0.2%
2	4	-8.9%	0.5%
3	4	-6.9	-0.1%
4	4	-6.2%	-0.8%
6	4	-5.0%	2.7%
2	5	-9.4%	1.0%

TABLE IV

RELATIVE ERROR FOR INFINITE TCP FLOWS WITH A SINGLE ROUTER IN  $ns$ .

the lag is measured in seconds. The model used in the graph is inferred by setting  $M = 6$  and  $N = 4$ . The two curves do not match very well. We conjecture the reason is that, for this simple setting, the behavior of the TCP flows become deterministic and is not described well by the model.

### A.2 Infinite TCP flows and on-off UDP sources with a single link

This setting differs the previous one in that 5 on-off UDP sources are connected to router  $r_0$  with their corresponding sinks connected to router  $r_1$ . The average on and off times are both set to be 1 second. A on-off source sends at the rate of 40Kb, with each packet of 1000 bytes. As in the previous setting, the probes are sent along with the chosen target TCP flow at a regular interval of 20ms.

The average loss rate and throughput of the target TCP flow is 0.918% and 0.127Mbps. The errors of throughput and loss rate of the model-driven TCP flow relative to the target TCP are shown in Table V. We observe that the relative errors of throughput and loss are within 15% and 16% respectively.

Fig. 5 shows the autocorrelation function of throughput for the model-driven TCP and the target TCP, where the lag is measured in seconds. The model used in the graph is inferred by setting  $M = 4$  and  $N = 5$ . We observe a good match of the two functions except at lag of 1 to 3 seconds.

### A.3 TCP and HTTP sources with a single link

In this setting, there is a single link with TCP and HTTP sources. As shown in Fig. 6, TCP sources are connected to router  $r_0$  ( $r_1$ ) with the corresponding TCP sinks connected to router  $r_1$  ( $r_0$ ). The number of TCP

$M$	$N$	throughput error	loss error
2	2	-13.1%	1.2%
3	2	-9.9%	-2.8%
4	2	-0.6%	-15.9%
2	3	-14.9%	3.3%
3	3	-12.0	2.4%
4	3	-6.4%	-8.3%
2	4	-14.8%	3.2%
3	4	-11.3	-0.8%
4	4	-4.2%	-8.1%
2	5	-13.3%	2.1%
3	5	-10.7%	-0.2%
4	5	-8.1%	-2.3%

TABLE V

RELATIVE ERROR FOR INFINITE TCP AND ON-OFF UDP SOURCES WITH A SINGLE ROUTER IN *ns*.

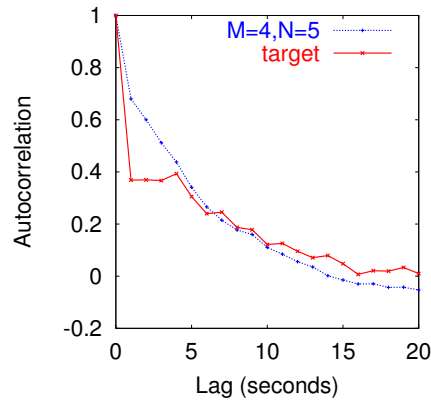


Fig. 5. Autocorrelation function of throughput for TCP and On-Off sources with a single router in *ns*.

flows from  $r_0$  to  $r_1$  and from  $r_1$  to  $r_0$  are set to be 25 and 20 respectively. Each TCP source uses TCP Reno, with a maximum window size of 20. All packets are 1000 bytes in size. The link between each TCP source and the corresponding router has a bandwidth of 10Mb. The propagation delay between a TCP source or sink to its corresponding router is uniformly distributed in  $[10, 20]$ ms. There are 60 HTTP clients connected to router  $r_0$ , each client has 16 HTTP sources. Their corresponding HTTP servers are connected to router  $r_1$ . The propagation delay from an HTTP server or client to its corresponding router is set to be uniformly distributed in  $[10, 20]$ ms. The HTTP traffic follows the empirical data provided by *ns*. Router  $r_0$  and  $r_1$  form a bottleneck link, with the bandwidth of 5Mb, propagation delay of 90ms and the maximum queue length of 200 packets. We choose a TCP flow from  $r_0$  to  $r_1$  as the target flow. A series of probe packets at regular intervals of  $\Delta$  are sent along the forward path of the target TCP flow and called forward probes. At the same time, a series of probe packets at regular

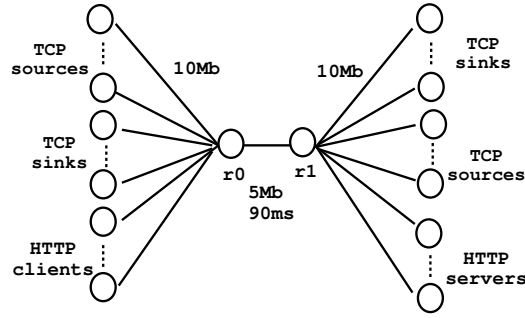


Fig. 6. TCP and HTTP traffic with a single link in *ns*.

$M_f(=M_b)$	$N_f(=N_b)$	throughput error	loss error
2	2	-15.1%	-5.2%
3	2	14.2%	-43.6%
4	2	-19.2%	14.9%
5	2	-12.6%	12.0%
6	2	-0.9%	0.9%
5	3	7.4%	-7.9%
6	3	8.3%	-10.7%

TABLE VI

RELATIVE ERROR FOR TCP AND HTTP SOURCES WITH A SINGLE ROUTER IN *ns*.

intervals of  $\Delta$  are sent along the backward path (ACKs) of the target TCP flow and called backward probes. Here  $\Delta$  is 20ms.

We ran a simulation for 3400 seconds and focus on the segment of [2000, 3200]. The comparison of the TCP flow predicted by our approach with the target TCP flow in the original setting is shown in Table VI. The average loss rate of the target TCP flow along the path of  $r_0$  to  $r_1$  is 1.61% and the average throughput is 0.179Mbps. The errors of both throughput and loss rate of the model-driven TCP flow relative to the target TCP are shown in Table VI. In the table,  $M_f$  and  $M_b$  represent the number of observation symbols for the forward and backward models respectively, and  $N_f$  and  $N_b$  represent the number of hidden states for the forward and backward models respectively. For this experiment, we set the sizes of the forward and backward models to be the same. As shown in the table, small value of  $M$  (e.g.  $M = 2, 3, 4$ ) leads to high relative errors. The reason is that, for this experiment, the delay distribution from a model with small values of  $M$  deviates from that of the target TCP flow because the description of the delay is too coarse. Our algorithm cannot produce convergent results for models with a large number of states (e.g.  $M \geq 5$  and  $N \geq 4$ ), for the reason given in Section IV.

Fig. 7 shows the autocorrelation function of throughput for the model-driven TCP and the target TCP, where the lag is measured in seconds. The model used in the graph is inferred by setting  $M = 4$  and  $N = 2$ . We

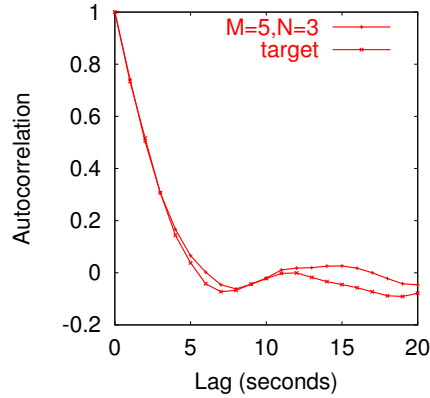


Fig. 7. Autocorrelation function of throughput for TCP and HTTP sources with a single router in *ns*.

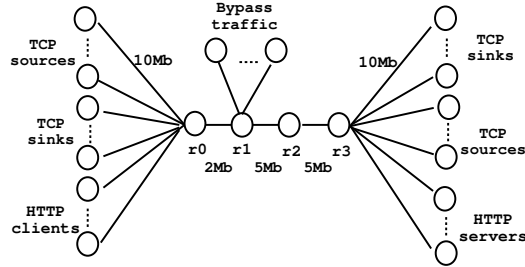


Fig. 8. TCP and HTTP sources with multiple links in *ns*.

observe a good match of the two functions. The results for other settings of  $M$  and  $N$  values in Table VI are similar.

#### A.4 TCP and HTTP sources with multiple links

We next examine the performance of the model in a setting with multiple links as shown in Fig. 8. We connect four routers  $r_0, r_1, r_2$  and  $r_3$ . The bandwidth between router  $r_0$  and  $r_1$  is set to be 2Mb. The bandwidth between the other consecutive routers (router  $r_1$  to  $r_2$  and router  $r_2$  to  $r_3$ ) is set to be 5Mb. The maximum queue length and propagation delay between two consecutive routers are set to be 100 packets and 30ms respectively. There are two types of flows in this setting: flows traversing all the routers and flows pass part of the routers. we call the first type of flows *traversing* flows and the second type of flows *bypass* flows. The numbers of traversing TCP flows from router  $r_0$  to  $r_3$  and from  $r_3$  to  $r_0$  are set to be 3 and 2 respectively. Each traversing flow starts from a TCP source connected to router  $r_0$  ( $r_3$ ), with their corresponding TCP sink connected to router  $r_3$  ( $r_0$ ). There are 10 HTTP clients connected to router  $r_0$ , with their corresponding HTTP servers connected to router  $r_3$ . The settings for the traversing TCP flows and the HTTP flows are the same as in Section VI-A.3. Each bypass flow

from $\rightarrow$ to	$r_0 \rightarrow r_1$	$r_0 \rightarrow r_2$	$r_1 \rightarrow r_2$	$r_1 \rightarrow r_3$	$r_2 \rightarrow r_3$
number	3	1	2	2	4
from $\rightarrow$ to	$r_1 \rightarrow r_0$	$r_2 \rightarrow r_0$	$r_2 \rightarrow r_1$	$r_3 \rightarrow r_1$	$r_3 \rightarrow r_2$
number	1	2	3	3	2

TABLE VII  
NUMBER OF BYPASS FLOWS IN THE MULTIPLE ROUTER SETTING.

$M_f(= M_b)$	$N_f(= N_b)$	throughput error	loss error
3	2	-7.7%	15.6%
4	2	8.1%	-0.07%
5	2	17.0%	1.9%
2	3	-9.9%	8.5%
3	3	5.6%	-10.7%
4	3	9.3%	-3.2%
2	4	-3.9%	1.7%
3	4	-3.5%	7.2%

TABLE VIII  
RELATIVE ERROR OF LOSS AND THROUGHPUT OF TCP AND HTTP SOURCES WITH MULTIPLE ROUTERS IN *ns*.

starts from a source connected to router  $r_i$  and ends at another router  $r_j$ , where  $0 \leq i, j \leq 3$ . The propagation delay and bandwidth from the source of a bypass flow to the connected router are 10ms and 10Mb respectively. The number of bypass flows along the path from router  $r_i$  to  $r_j$  are listed in Table VII.

We choose a traversing TCP flow from  $r_0$  to  $r_3$  as the target flow and send probe packets with the TCP flow along both the forward and backward directions at the probing interval of 20ms. The target TCP flow encounters a loss rate of 1.49% along the path of  $r_0$  to  $r_3$  and its average throughput is 0.144Mbps. The errors of both throughput and loss rate of the model-driven TCP flow relative to the target TCP flow are shown in Table VIII. We observe that in general the performance of a model improves as the number of states increases. The relative errors of the throughput and the loss rate fall within 17% for various settings.

Fig. 9 compares the autocorrelation functions of the throughput of a model-driven TCP and the target TCP flow, where lags are measured in seconds. The model used in the graph is inferred using  $M = 3$  and  $N = 2$ . The figure demonstrates a good match of the two functions.

### B. Validation over the Internet

We next describe our validation of the simulation method over the Internet. Two traces  $T_1$  and  $T_2$  are shown in Table VI-B. Each trace is collected by a 1-hour run and contains two-way (along the forward and the backward paths) traces. For both traces, the sender of the target TCP is at our site (UMass) and the receiver is at University

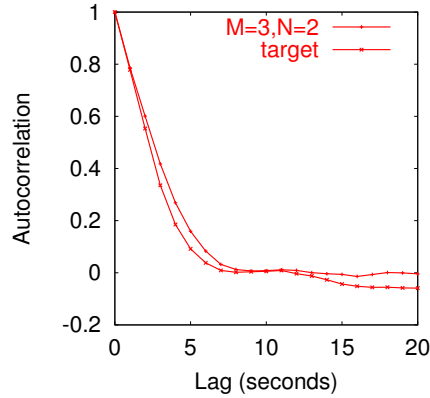


Fig. 9. Autocorrelation function of throughput for TCP and HTTP sources with multiple routers in *ns*.

Trace	Time	Probing	Loss rate	Throughput
$T_1$	03/08/02 22:24	Periodic, $\Delta=20\text{ms}$	1.51%	0.214Mbps
$T_2$	03/16/02 14:06	Geometric, $\Delta=10\text{ms}$ , $\tau=50\text{ms}$	0.41%	1.578Mbps

TABLE IX

TWO VALIDATION EXPERIMENTS FOR TCP OVER THE INTERNET

of Southern California (USC). The number of hops from UMass to USC is 20 and the number of hops from USC to UMass is 14. The measurement of one-way delays is described in Section V-E. We take half of the minimum round-trip time of the target TCP flows as the constant portion to obtain the actual delays. To incorporate the models into *ns* and drive a TCP flow, we need to obtain some key parameters used in the target TCP flow. We find that the target TCP flow uses TCP SACK by the method described in [?]. Most of the packets in the target TCP flow (over 99%) are 1448 bytes. The receiver window size changes over the session while it is fixed in *ns*. We obtain a rough estimate of the window size from [?] and fine-tune it during the validation process. We next present the results for traces  $T_1$  and  $T_2$ .

In trace  $T_1$ , probes are sent at regular intervals  $\Delta$  of 20ms on the directions from UMass to USC and from USC to UMass. We focus on a stationary segment of 300 seconds. On the route from UMass to USC, the average loss rate and throughput of the target TCP flow are 1.51% and 0.214Mbps respectively, as shown in Table VI-B. In *ns*, we set the one-way delay to be 100ms and the window size to be 8. Table X summarizes the errors of throughput and loss rate of the model-driven TCP flows relative to the target TCP flow for different settings of the models, where entries marked by \* are obtained by the approximation (11). We observe that, for this trace, the number of observation symbols as 2 to 3 and the number of hidden states as 2 to 5 are sufficient to achieve good results. The comparison of the ACF of the throughput for the model-driven TCP and the target

$M_f$	$N_f$	$M_b$	$N_b$	throughput	loss	$M_f$	$N_f$	$M_b$	$N_b$	throughput	loss
				error	error					error	error
2	3	2	2	-0.4%	-12.0%	2	3	2	3	-0.3%	10.1%
2	4*	2	2	-4.2%	8.0%	2	4*	2	3	-4.7%	-6.7%
2	5*	2	2	-5.6%	9.4%	2	5*	2	3	-5.5%	10.4%
3	3	3	2	-1.0%	16.0%	3	3	3	3	-0.6%	11.0%
3	4*	3	2	2.9%	-3.3%	3	4*	3	3	1.1%	-9.0%
3	5*	3	2	3.1%	-0.6%	3	5*	3	3	2.6%	2.3%

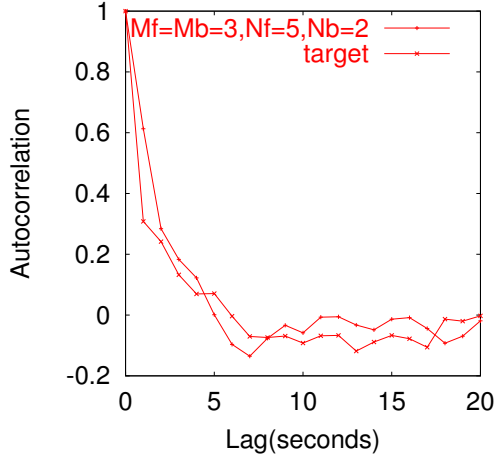
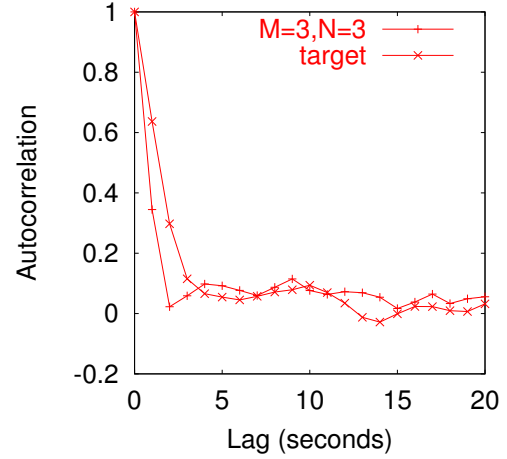
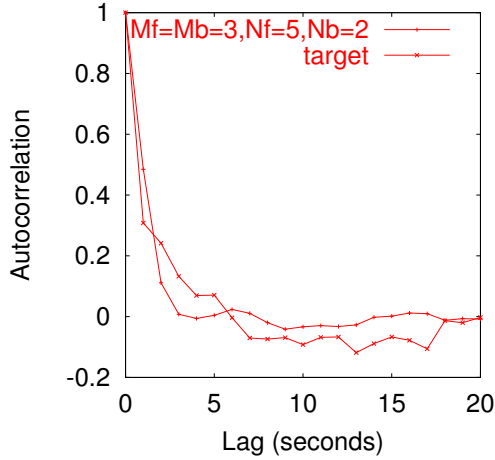
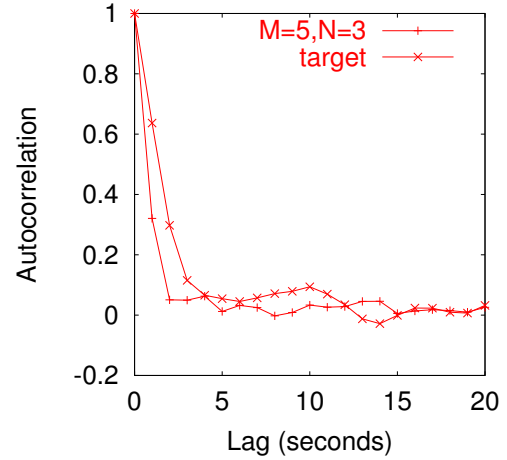
TABLE X  
RELATIVE ERRORS OF LOSS RATE AND THROUGHPUT FOR TRACE  $T_1$

$M_f$	$N_f$	throughput	error	loss	error	$M_f$	$N_f$	throughput	error	loss	error
2	2		-0.1%		-3.0%	2	6		-0.3%		-1.1%
2	3		2.8%		-12.4%	2	7		-1.4%		4.0%
2	4		2.4%		-10.4%	2	8		-1.5%		3.6%
2	5		-0.6%		0.3%	2	9		1.4%		-6.8%

TABLE XI  
RELATIVE ERRORS OF LOSS RATE AND THROUGHPUT FOR TRACE  $T_2$

TCP flow is shown in Fig. 10(a), where lags are measured in seconds and the model is obtained by setting  $M_f = 3, N_f = 5, M_b = 3, N_b = 2$ .

In trace  $T_2$  (see Table VI-B), we use geometric probing on the forward and the backward paths with the average probing interval  $\tau$  of 50ms. The minimum probing interval  $\Delta$  is set to be 10ms. We focus on a stationary segment of 1200 seconds and use the missing-data EM algorithm in the model inference process. As shown in Table VI-B, the average loss rate in this trace is 0.41%, significantly lower than in trace  $T_1$ . The average throughput is 1.578Mbps. In *ns*, we set the one-way delay to be 43ms and the window size to be 18. We cannot find a convergent solution for the model on the backward path even for  $M_b = 2$  and  $N_b = 2$ . The reason is that there are no losses on the backward path and the delay variations are in a very short range ( $\leq 2$ ms). We therefore use a constant delay (the average delay on the backward path) for the backward direction. The settings of the model for the forward path and the relative error of the model-driven TCP flow relative to the target TCP are shown in Table XI. The relative errors of throughput and loss rate of all the settings in the table fall within 3% and 13% respectively. We also observe good conformance of the model-driven TCP flow and the target TCP from the ACFs of the throughput as shown in Fig. 10(b), where lags are measured in seconds.

(a) ACF of throughput for  $T_1$ .(b) ACF of throughput for  $T_2$ .Fig. 10. ACF of throughput for  $T_1$  and  $T_2$ .(a) ACF of throughput for  $T_1$ .(b) ACF of throughput for  $T_2$ .Fig. 11. ACF of throughput for  $T_1$  and  $T_2$  when associating loss to each symbol.

### C. Validation over the Internet Revisited

We next describe the validation results for trace  $T_1$  and  $T_2$  when loss is associated with each delay symbol.

Table XII shows the loss rate and throughput errors for different  $M$  and  $N$  settings for trace  $T_1$ . We observe that the maximum error of loss is decreased from 16% (see Table X) to less than 8%. The loss predictions are also more consistent for various models. Table XIII shows the loss rate and throughput errors for different  $M$  and  $N$  settings for trace  $T_2$ . We also observe more consistent loss predictions for various models. Furthermore, in general, we observe that the loss error decreases as  $N_f$  increases. Fig. 11 shows ACF of the throughput for  $T_1$  and  $T_2$ . It is similar to when combining loss to the largest symbol (Fig. 10).

$M_f$	$N_f$	$M_b$	$N_b$	throughput error	loss error
2	2	2	2	-3.3%	-0.8%
2	3	2	2	-3.0%	-0.5%
2	4	2	2	-3.8%	2.6%
2	5	2	2	-2.6%	0.7%
3	2	2	2	-3.7%	4.3%
3	3	2	2	-3.2%	7.3%
2	2	2	3	-2.8%	-1.2%
2	3	2	3	-2.7%	-2.0%
2	4	2	3	-3.6%	1.1%
2	5	2	3	-0.9%	-5.6%
3	2	2	3	-3.8%	3.4%
3	3	2	3	-1.6%	-1.1%
2	2	2	4	-2.4%	-1.6%
2	3	2	4	-2.8%	-0.4%
2	4	2	4	-3.3%	1.4%
2	5	2	4	-0.6%	-5.6%
3	2	2	4	-2.5%	-0.3%
3	3	2	4	-1.7%	-1.2%
2	2	3	2	-2.8%	-0.7%
2	3	3	2	-2.6%	-1.5%
2	4	3	2	-2.6%	1.0%
2	5	3	2	-2.2%	-1.3%
3	2	3	2	-3.6%	3.2%
3	3	3	2	-1.8%	0.4%
2	2	3	3	-2.7%	-0.3%
2	3	3	3	-2.5%	-2.4%
2	4	3	3	-2.6%	0.2%
2	5	3	3	-2.6%	0.6%
3	2	3	3	-2.6%	0.2%
3	3	3	3	-2.9%	2.9%
2	2	3	4	-3.0%	-0.5%
2	3	3	4	-2.2%	-1.6%
2	4	3	4	-1.2%	-6.3%
2	5	3	4	-3.1%	2.5%
3	2	3	4	-3.0%	1.3%
3	3	3	4	-2.3%	3.5%

TABLE XII

RELATIVE ERROR AND THROUGHPUT AND LOSS RATE FOR  $T_1$  (MULTIPLE LOSSES).

$M_f$	$N_f$	throughput error	loss error
2	2	-2.0%	9.1%
2	3	-2.6%	9.9%
2	4	-2.1%	9.8%
2	5	-2.4%	9.1%
2	6	-2.2%	9.3%
2	7	-2.3%	8.5%
2	8	-2.3%	8.9%
2	9	-1.8%	10.1%
2	10	-2.1%	9.9%
3	2	-1.9%	9.3%
3	3	-1.9%	10.4%
3	4	-1.9%	5.8%
3	5	-1.6%	5.5%
3	6	-1.5%	5.3%
3	7	-2.2%	11.2%
3	8	-1.7%	5.9%
3	9	-1.5%	6.6%
4	2	-1.4%	7.6%
4	3	-1.7%	7.5%
4	4	-0.0%	-0.1%
4	5	-0.2%	-0.4%
4	6	-1.0%	4.5%
4	7	-1.3%	5.7%
5	2	-1.3%	8.4%
5	3	-0.7%	5.8%

TABLE XIII

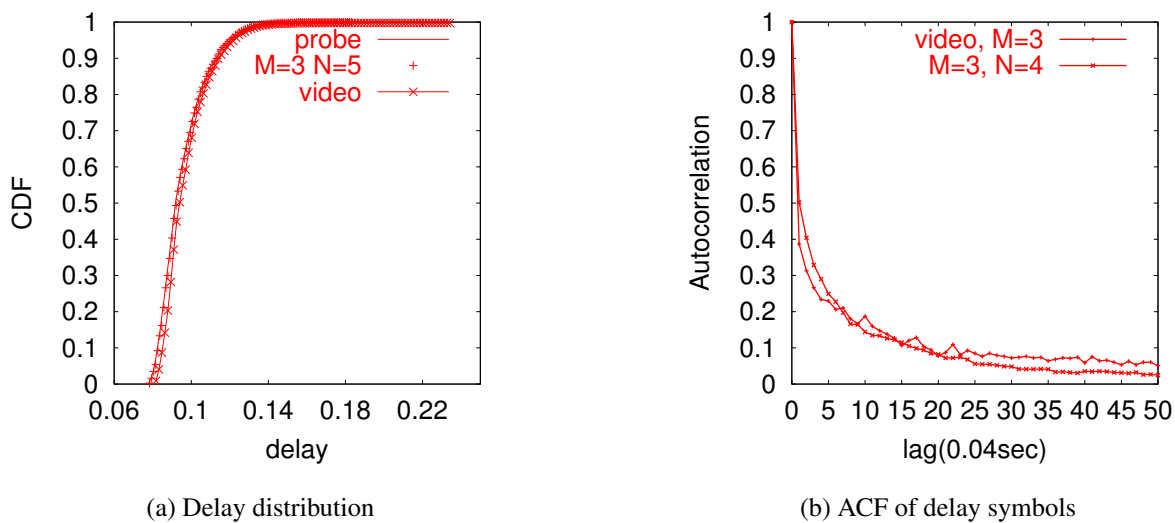
RELATIVE ERROR OF LOSS AND THROUGHPUT FOR TRACE  $T_2$ (MULTIPLE-LOSSES).

Fig. 12. Results for a streaming video application.

TABLE XIV  
RELATIVE ERROR FOR A STREAMING VIDEO APPLICATION

$M$	$N$	delay error	CV error	$M$	$N$	delay error	CV error
2	2	-2.1%	-3.5%	3	2	-2.3%	0.4%
2	3	-2.0%	1.9%	3	3	-2.4%	-0.6%
2	4	-2.1%	0.9%	3	4	-2.4%	-0.3%
2	5	-2.0%	1.8%	3	5	-2.2%	-0.6%

## VII. VALIDATION FOR A STREAMING VIDEO APPLICATION

In this section, we validate the simulation method using a streaming video application. We send probe packets along with a streaming video application from one end host to another end host. The streaming video application uses UDP and there are no acknowledgments along the backward path. Therefore we only need to infer the model along the forward path. Since streaming media applications are sensitive to delays and losses, we use the following metrics:

- average loss rate, average delay and CV (coefficient of variation) of the delays
- distribution of the delays
- ACF of the delay and loss symbols

The one-way delays from the sender to the receiver are obtained as described in Section V-E. To calculate the ACF of the delay symbols, the variable portion of the one-way delay is sufficient. To calculate the other delay metrics, we need to obtain the actual delay by adding a constant portion to the variable portion. However, unlike with TCP, this constant portion does not affect the behavior of the application. We estimate the constant portion roughly from the round-trip time reported by *ping* packets. We send *ping* packets lasting for 30 minutes before the experiments and take half of the minimum round-trip time to be the constant portion.

We next describe an experiment from our site (UMass) to a site in Universidade Federal do Rio de Janeiro (UFRJ), Brazil, which was carried out at 10:25am on April 1, 2002 and lasted for 1 hour. The number of hops from UMass to UFRJ is 14. We use a 200Kbps CBR (Constant Bit Rate) video. The frame rate of the video is 25 frames per second. For this experiment, we use geometric probing with the average probing interval  $\tau$  as 50ms and the minimum time interval  $\Delta$  as 10ms. The constant portion of the delay is estimated to be 78ms.

We choose a 1200-second segment from the video stream as the target stream and use the missing data EM algorithm in the model inference process. For the target stream, the average loss rate is 0.2%, the average delay is 99ms and the CV of the delays is 0.12. The relative error of the results from the inferred models are shown in Table XIV. The delay errors are within 3% for various settings of  $M$  and  $N$ , where  $M$  is from 2 to 3 and  $N$  is from 2 to 5. The errors of CV of the delays are within 4%. The result loss rates for various settings are around

0.2%.

The cumulative delay distributions of the target video stream, probes and a model-driven video stream are shown in Fig. 12 (a). We observe that the delay distribution of the model-driven stream is almost identical to that of the probes. The difference in the delay distribution of the model-driven stream and the target stream is caused by the different observations seen by the probes and the target stream.

We compare the ACFs of the delays of the target stream and model-driven streams for different settings of  $M$  and  $N$ . We observe that, for the same  $M$ , the ACF becomes closer to that of the target stream as  $N$  increases. Fig. 12 (b) shows the ACFs of the delay symbols of the target video stream and that of model-driven streams. Each lag is 40ms. The model is inferred by setting  $M = 3$ ,  $N = 4$ .

## VIII. CONCLUSION AND FUTURE WORK

In this paper, we explored the use of continuous-time HMMs for network protocol and application evaluation. We develop a mechanism to infer a continuous-time HMM from a sequence of observations of probe packets. Our validations of this simulation method in *ns* and over the Internet demonstrate that this method has the potential to represent a wide range of *real* network scenarios. It is accurate and time and space efficient. We observe that for real network settings we examined, models of small size ( $M = 2, 3$ , or 4 and  $N = 2, 3$ ) are sufficient to achieve good results. As future work, we are pursuing the following directions: (i) develop methods to convert observation symbols into delay values smoothly, (ii) examine the use of the model in more real network scenarios, (iii) investigate how to use the probing trace in combination with the continuous-time HMM to model transient behavior in the network.

## REFERENCES

- [1] G. Riley, R. Fujimoto, and M. Ammar, "Parallel/Distributed NS." <http://www.cc.gatech.edu/computing/compass/pdns/index.html>.
- [2] S. McCanne and S. Floyd, "ns-LBNL network simulator," <http://www-nrg.ee.lbl.gov/ns/>.
- [3] J. H. Cowie, "Scalable simulation framework API reference manual," March 1999.
- [4] Y. Guo, "Time-stepped hybrid simulation (TSHS) for large scale networks," in *Proc. IEEE INFOCOM*, March 2000.
- [5] B. Liu, D. R. Figueiredo, Y. Guo, J. Kurose, and D. Towsley, "A study of networks simulation efficiency: fluid simulation vs. packet-level simulation," in *INFOCOM*, 2001.
- [6] L. Rizzo, "Dummysnet: A simple approach to the evaluation of network protocols," in *Computer Communication Review*, p. 27(2), February 1997.
- [7] M. Carson, "NIST Net," <http://snad.ncsl.nist.gov/itg/nistnet>.
- [8] V. Paxson, "End-to-end Internet packet dynamics," in *IEEE/ACM Transactions on Networking*, p. 7(3), June 1999.
- [9] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement," in *Proc. ACM SIGMETRICS*, June 2000.
- [10] M. Yajnik, J. Kurose, and D. Towsley, "Packet loss correlation in the Mbone multicast network," in *IEEE Global Internet*, November 1996.
- [11] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *Proc. IEEE INFOCOM*, March 1999.
- [12] K. Salamatian and S. Vaton, "Hidden Markov modelling for network communication channels," in *Proc. ACM SIGMETRICS*, June 2001.
- [13] S. Ross, *Stochastic Process*. John Wiley & Sons, 1996.
- [14] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, vol. 77, pp. 257–285, February 1989.

- [15] I. Ryzhik, I. S. Gradshteyn, and A. Jeffrey, *Table of Integrals, Series, and Products*. Academic Press, 6th ed., 2000.
- [16] J. S. Bendat and A. G. Peirsol, *Random Data Analysis and Measurement Procedures*. John Wiley & Sons, 1986.
- [17] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *Proc. IEEE INFOCOM*, March 1999.
- [18] J. W. Tukey, *Explanatory Data Analysis*. Addison-Wesley, 1977.
- [19] J. Padhye and S. Floy, "On inferring TCP behavior," in *Proc. ACM SIGCOMM*, August 2001.
- [20] M. Mathis, J. Semke, and J. Mahdavi, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Communications Review*, vol. 27, no. 3, 1997.