Determining Model Accuracy of Network Traces

Almudena Konrad [a] Ben Y. Zhao [b] Anthony D. Joseph [c]

[a] *Mills College*

[b] *University of California at Santa Barbara*

[c] *University of California at Berkeley*

Abstract

Accurate network modeling is critical to the design of network protocols. Traditional modeling approaches, such as Discrete Time Markov Chains (DTMC) are limited in their ability to model time-varying characteristics. This problem is exacerbated in the wireless domain, where fading events create extreme burstiness of delays, losses, and errors on wireless links. In this paper, we describe the *data preconditioning* modeling technique that is capable of capturing the statistical characteristics of wired and wireless network traces. We revise our previous developed data preconditioning modeling algorithm, the Markov-based Trace Analysis (MTA), and present the Multiple states MTA (MMTA) algorithm. Our main contributions are methodologies created to quantify the accuracy of network models, methodology to choose the most accurate model for a given network and characteristic of interest (*e.g.*, delay, loss, or error process), and the validation of our data preconditioning modeling algorithms.

Key words: model accuracy, network traces, wireless networking, Markov chains, stationarity, data preconditioning

1 Introduction

Simulation of network links is perhaps the most common method for evaluating application and network protocol designs. Simulation enables researchers to accurately and repeatably explore the behavior of a protocol under different

Email addresses: akonrad@mills.edu (Almudena Konrad), ravenben@cs.ucsb.edu (Ben Y. Zhao), adj@cs.berkeley.edu (Anthony D. Joseph).

network conditions (*e.g.* varying loss, delay, and error). However, the validity of results are highly dependent on the accuracy of the network simulation model. Floyd [1] argues that the use of inaccurate models leads to flaws in networking research. We also demonstrated the importance of model accuracy by observing that a naive error model used in simulation during protocol design led to a poor choice of a protocol parameter [2]. For example, a detailed understanding of the packet loss process and burstiness of errors is necessary for the proper design of error control protocols such as Automatic Repeat reQuest (ARQ) protocols.

In realistic networks and especially wireless networks, researchers must model measurements whose characteristics experience non-stationarity (time variability) and complex patterns due to a number of factors, including both internal network elements and external events. While classical models such as Bernoulli, Gilbert, high-order Discrete Time Markov Chain (DTMC), or Hidden Markov Models (HMM) have worked surprisingly well in modeling events in traditional networks, they are ill-suited for handling traces of today's networks (*e.g.*, lossy wireless channels). For example, the Bernoulli model is a memory-less process, where each value is generated statistically independent of previous outputs. Thus, it is unlikely to produce accurate models of networks exhibiting bursty losses such as wireless links. To address this, we introduce a data preconditioning technique that extracts and models the stationary components of non-stationary datasets. We describe the original data preconditioning model the Markov-based Trace Analysis (MTA) [3], and introduce the Multiple states MTA (MMTA) model.

In addition, given the large number of existing traditional and new models, researchers face the challenge of choosing the most accurate model for their datasets. As history has shown, a bad choice can result in inaccurate models that result in misleading simulation results. We show that datasets corresponding to different networks experience different statistical characteristics, underscoring the need to develop a tool that identifies the best model for a given set of network characteristics. In this paper, we introduce a methodology to quantify the accuracy of different models, and show how to use it to choose the best model for a given set of network characteristics.

The paper is structured as follows. We begin with related work in Section 2. In Section 3, we define and classify network traces. In Section 4, we discuss traditional modeling techniques. We present our data preconditioning technique in Section 5. We then present our approach to evaluate model accuracy and our modeling methodology in Section 6. In Section 7, we apply these techniques to network path traces collected from seven different networks. In Section 8, we introduce our *Domain of Applicability Plots* (DAP), a tool to rapidly visualize model accuracy , and use it to evaluate the behavior of various classical and data preconditioning models. Finally, we conclude with Section 9.

2 Related Work

There is significant interest in the area of using network measurements to model network behavior. However, very few researchers address the problem of non-stationarity in network modeling. Zhang and others study stationarity in the Internet and introduce a new notion of stationarity that is more relevant to network properties [4]. They call a dataset *operationally stationary* if the statistics of interest remain within bounds considered operationally equivalent. Their most interesting finding is that stationarity depends on the time scale that is used for evaluation. Others have looked at the stationarity behavior of network traffic, *traffic stationarity*. For example, Molnar *et al.* [5] propose a simple approach for identifying stationary intervals and analyzing them independently. They introduce a new technique for identifying these intervals. Leland *et al.* [6] study the stationarity of self-similar models of network traffic.

Several researchers have applied traditional models to the analysis of non-stationary data collected in computer networks. In particular, they have used traditional models to characterize the loss process of various channels. Bolot *et al.* [7] use a characterization of the loss process of audio packets to determine the appropriate error control scheme for streaming audio. They model the loss process as a two-state Markov chain, and show that the loss burst distribution is approximately geometric. Yajnik *et al.* [8] characterize the packet loss in a multicast network by examining the spatial (across receivers) and temporal (across consecutive packets) correlation in packet loss. Of particular interest is their modeling of temporal loss using a 3^{rd} order Markov chain. Yajnik's work identifies the problem of non-stationarity in their datasets, and they analyze the data by removing these parts of the data that experience non-stationary error behavior.

There is also related work in wireless traffic modeling. Nguyen *et al.* [9] present a two-state Markov wireless error model (*i.e.*, Gilbert model), and develop an improved model based on collected Lucent 900 MHz WaveLAN error traces. Building on this work, Balakrishnan and Katz [10] also collected error traces from a Lucent 900 MHz WaveLAN network and developed a two-state Markov chain error model. Willig *et al.* [11] present a special class of Markov models, called *bipartite models*. Zorzi *et al.* [12] also investigate the error characteristics of a wireless channel and compare an Independent and Identically Distributed (IID) model to the Gilbert model. Their work postulates that higher order models are not necessary.

Trace	**Frames**	**FER**	L_{exp}, EF_{exp}, L_{den}	**C**
IP_1	360,385	0.027	0.034, 0.099, 0.82	1
IP_2	331,021	0.050	0.002, 0.099, 0.11	82
IP_3	155,889	0.064	0.057, 0.099, 0.79	1
WLAN_E	288,804	0.063	0.044, 0.099, 0.34	5
WLAN_D	188,436	0.293	0.046, 0.005, 0.414	41
GSM_E	616,404	0.055	0.005, 0.056, 0.41	23
GSM_D	2,579	0.055	0.002, 0.028, 0.95	31

Table 1
Collected traces and their characteristics: number of frames, Frame Error Rate (FER), the variables $(L_{exp}, EF_{exp}, L_{den})$, and the change of state variable, C.

3 Defining and Classifying Binary Network Path Traces

We define binary network path traces as sequences of 0's and 1's, where a 1 denotes the occurrence of a specific event in the network path, while a 0 denotes the lack of the event. For example, a 1 could represent a lost or dropped packet, while a 0 could represent a correctly received packet. In [2], we used the Runs Test developed by Bendat and Piersol [13] to show that GSM binary error traces are locally stationary binary time series [14], consisting of regions that experience various statistical behaviors. In this paper, we extend that work by analyzing and modeling several types of network path traces. In particular, we analyze traces that capture the following events: IP packet losses, wireless frame errors, and packet delays. A 1 signifies a lost packet in a loss trace and a corrupted frame in an error trace; and in a delay trace, it means that the packet or frame arrived with a delay greater than some maximum threshold [1]. To generalize these cases, we refer to values of 1 in a packet or frame trace as an *error frame*.

We define the Frame Error Rate (FER) as the overall percentage of frames (or packets) that have errors (or losses, or delays) relative to the total number of frames (or packets) in a trace.

To understand the effectiveness of our techniques for a broad set of network types and metrics, we analyzed traces collected under various scenarios from several networks and at different protocol layers (see Table 1). *IP_1* is a loss trace collected by Yajnik *et al.* [8] during an uncongested IP connection from Massachusetts to Sweden. *IP_2* and *IP_3* are IP loss traces collected by Wenyu

[1] The threshold value is dependent upon the particular application of interest and it indicates the delay value for which packets will be dropped by the application.

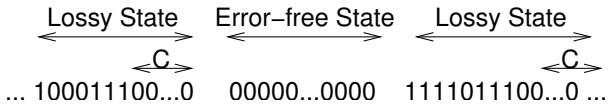

Fig. 1. An error trace with lossy and error-free states.

Jiang at Columbia University (CU). *IP_2* was collected on an uncongested path from CU to GMD (the German National Research Center for Information Technology), and *IP_3* was collected on an uncongested connection from CU to the University of Massachusetts. *WLAN_E* was collected under good signal quality conditions from an IEEE 802.11b wireless LAN testbed at the Technical University of Berlin by Andreas Willig [11]. We collected *GSM_E* under poor signal quality conditions at the Circuit-Switched Data (CSD) radio link layer of a GSM wireless data cellular network at the UC Berkeley campus.

We also collected *GSM_D* and *WLAN_D* at the transport layer using UDP over a poor signal quality GSM CSD link [2] and a good signal quality IEEE 802.11b network at the UC Berkeley campus. These two traces were collected to analyze the delays introduced in applications by various wireless networks. For *GSM_D*, the delay threshold was chosen to be 2 seconds, while for *WLAN_D*, we chose a delay threshold of 20 milliseconds. Note that each of the delay values obtained in *GSM_D* and *WLAN_D* is the sum of delay values across the wireless and wired components of the path. We analyze the end-to-end network delay in this paper, and plan to explore per-link statistics in future work. Finally, we are in the process of collecting and analyzing loss and delay traces in a General Packet Radio Service (GPRS) GSM network and a Code Division Multiple Access (CDMA) 1xRTT wireless data network.

We analyzed the traces in Table 1 and observed that these traces can be decomposed into clusters of 1's and 0's, and long clusters of just 0's. We associate these clusters with lossy states and error-free states (see Figure 1), by dividing the trace into states (clusters). Lossy states begin with an element of 1 and contains bursts of 1's and 0's, and ends with a burst of 0's of length equal to or greater than a *change-of-state* variable C. The next 0 element following the burst of C 0's marks the beginning of an error-free state, which is terminated by the 0 preceding the next 1 element in the trace. The value of C is a design parameter that we have defined as the mean plus one standard deviation of the length of error bursts in a trace. In Section 5.2, we provide an analysis to optimize and justify the parameter C.

In [2], we observe that the length distributions of lossy and error-free states can be approximated with an exponential distribution function, where the smaller the exponential parameter, the larger the average cluster length. Based on this

[2] We are still in the process of collecting additional *GSM_D* traces.

observation, we characterize collected traces using a tuple of three variables $(L_{exp}, EF_{exp}, L_{den})$, where L_{exp} and EF_{exp} are the parameters of the lossy and error-free state length exponential distribution, and L_{den} is the error density in the lossy state (*i.e.*, the probability of getting a 1 inside a lossy state). Note the significant difference between L_{den} and the FER.

3.1 Stationarity of Network Path Traces

We now discuss the notion of statistical stationarity and how we use it to improve accuracy of our models. We define a trace to be the process $\{X_n \mid n \geq 0\}$ with a discrete space $E = \{0, 1\}$. A process X_n is *strictly stationary* if the distribution of $(X_{p+1}, ..., X_{p+k})$ is the same as that of $(X_1, ..., X_k)$ for each p and k. X_n is *second-order stationary* if the mean $m_n = E(X_n)$ is constant (independent of n), and the auto-covariance only depends on the difference k for all n (*i.e.*, $Cov(k, n) = Cov(X_n, Xn - k) = Cov(k)$). Given a second-order stationary binary time series X_n, the process can be modeled as a homogeneous DTMCs, where the value of the chain at time n is determined by the memory of the process [14]. In a homogeneous DTMC, the transition probabilities remain constant over time, (*i.e.*, $Pr(X_{n+1} = j \mid X_n = i) = Pr(X_2 = j \mid X_1 = i)$).

However, checking a binary trace for second-order stationarity is mathematically challenging, and, we believe, not necessary for network modeling. For our purposes, we define a binary trace as *stationary* whenever the statistical properties, such as mean, median or standard deviation do not vary over time for small window sizes (*i.e.*, values of k). The requirement on the window size to be small is necessary to be able to apply high-order DTMCs, where the transitions probabilities do not vary over time.

As mentioned above, we observe that empirical network traces are non-stationary, since the statistical properties of traces vary over time. However, these traces exhibit local stationarity (*i.e.*, a non-stationary data set composed of deterministic regions and small stationary regions). Our work will show that attempting to fit traditional models onto traces with non-stationary properties can lead to inaccurate models.

We use the previously discovered Runs Test [13] to analyze the stationarity of network path traces. The Runs Test computes the median run (*i.e.*, error burst) value of the trace, divides the trace into equal size segments, and plots a histogram of runs not equal to the median value in each segment. Too few or too many runs is a sign of non-stationarity. If a trace is stationary, the number of runs distribution between the 0.05 and 0.95 cut-offs will be close to 90 percent. The Runs Test can be summarized as follows:

(1) Define a run as a number of consecutive ones (also referred to as an error

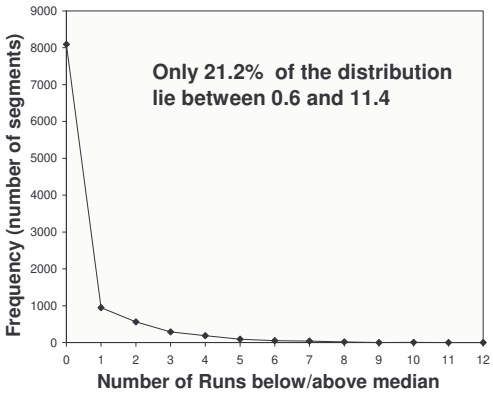

Fig. 2. The Runs Test applied to lossy subtrace.

burst).
(2) Divide the trace into segments of equal lengths (window size).
(3) Compute the lengths of runs in each segment.
(4) Count the number of runs of length above and below the median value for run lengths in the trace.
(5) Plot a histogram for the number of runs.

We apply the Runs Test to *GSM_E* with window size of 60. Figure 2 shows that only 21.2 percent of the runs distribution lie between the 0.05 and 0.95 cut-offs, and 78.8 percent lays outside the left and right cut-offs. Thus, from the Runs Test, we conclude that *GSM_E* is non-stationary for a window size of 60. We also tested several window sizes, and observed that as the window size decreases the percentage of runs distribution between the boundary points also decreases, (i.e., for smaller window sizes,*GSM_E* is non-stationarity). For example, for a window size of 20, only 12.3 percent of the runs distribution lie between the boundary points.

4 Classical Markov Models

Before we discuss our algorithms for modeling non-stationary datasets, we present as background the two types of classical stochastic models for characterizing the statistical properties of network traces that we examine in this paper. One is the well known and popular Gilbert model, which is a Markov process of memory size one. The other is the Hidden Markov Model (HMM) [15]. We discuss the reasoning behind our choices below.

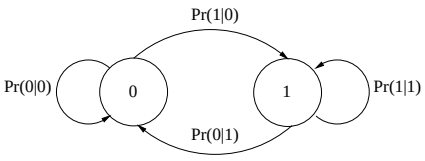

Fig. 3. Gilbert model state transition diagram.

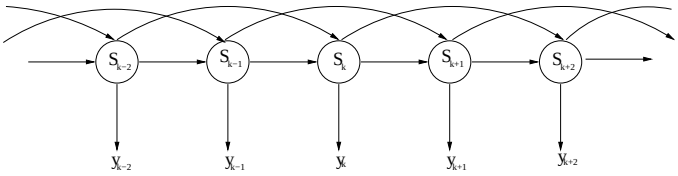

Fig. 4. Bayesian Network of a 2nd Order Hidden Markov Model.

4.1 The Gilbert Model

We choose the Gilbert model because it is one of the most common models used for network simulation. The model is a DTMC of order one and has two states (see Figure 3). In a network trace, the Gilbert model states correspond to the status of each data frame {0,1}, as defined previously. The Gilbert model predicts the state of the next frame by only considering the previously received frame. As a result, the Gilbert model can only model relatively short bursts of an event.

An alternative to the Gilbert model is a 3^{rd} order Markov model, a DTMC of order three with eight states. Compared to the Gilbert model, this model keeps track of the status of the previous three frames, increasing its prediction accuracy at the cost of additional complexity. However, even with this increase in accuracy, 3^{rd} order Markov models do not always accurately capture real network statistical characteristics (see [2]).

4.2 The Hidden Markov Model

For the second model, we choose a HMM model because many statisticians believe that the non-stationary characteristics of empirical network traces makes Hidden Markov Models (HMM) a good potential candidate to model network traces. In a HMM, each data pattern is associated with a hidden state, giving the HMM its main advantage: the ability to model non-stationary processes. The model parameters in a HMM are the transition probabilities between hidden states, the memory of the process, and the conditional probabilities of the observations given the current state. In a HMM, the current observation is statistically independent of the previous observations and only depends on the current state. This is known as the output independence assumption. Figure 4 illustrates the Bayesian network [16] for the graphical representation of a HMM of order 2, where $s_1, ..., s_k, ...$ represents the sequence of states and $y_1, ..., y_k, ...$ represents the sequence of observation.

We model network traces with a two-hidden-state 4^{th} order hidden Markov model. The states $\{S_1, S_2\}$ correspond to the lossy and error-free states de-

fined in Section 3, while the observation symbols $\{Y_1, Y_2\}$ correspond to the status of the data frame $\{0, 1\}$. We choose a high order of 4 to account for possible correlations between consecutive states. Using an order greater than 4 improves accuracy slightly while significantly increasing the computational complexity of the model.

5 Modeling through Data Preconditioning

In this section, we introduce *data preconditioning*, a new modeling methodology that supports a greater degree of behavior complexity in computer networks. We first describe the concepts behind the methodology, then discuss a way to optimize the change-of-state variable C. Finally, we further illustrate the concept by describing two instances of this methodology, the Markov-based Trace Analysis (MTA) algorithm and the Multiple-states MTA (MMTA) algorithm.

5.1 Data Preconditioning

The search to create accurate network models for datasets exhibiting non-stationarity led us to a new methodology that calls for analysis and preconditioning of data *before* it is fed into traditional models. Intuitively, we use pattern recognition to break down non-stationary datasets into stationary subsets which can be accurately modeled using traditional models. For a particular network characteristic, we follow the process illustrated in Figure 5. First, we identify data patterns that exhibit stationarity and suggest an underlying process consisting of some number of "states." Each state is associated with a specific data pattern corresponding to a particular network behavior [3]. For example, for network traces presented in Section 3, we identified two distinct states: lossy and error-free. Second, we concatenate trace regions with same states to form stationary subtraces, (i.e., lossy and error-free subtraces). Because of their stationarity, these subtraces can be accurately modeled using a high-order DTMC. Note that there will be as many subtraces as states. Finally, we use Markov models (or other similar modeling techniques) to calculate the transition probabilities between states.

This approach can be used to model very different characteristics of datasets from collected network measurements, including packet loss, end-to-end latency, or throughput. In this paper, we demonstrate how this research method-

[3] Each network behavior has certain statistical properties.

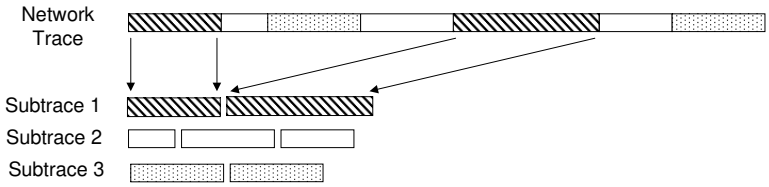

Fig. 5. Data Preconditioning: in this example a network trace is decomposed into three subtraces, each consisting of a concatenation of a specific data pattern.

ology can significantly improve the modeling accuracy of error and delay processes in wired and wireless networks.

5.2 Optimizing the change-of-state variable C

An important design decision in our data preconditioning methodology is how to locate the appropriate transitions between different states. As the model scans the input trace, it transitions from its current state S to a new state S' if it observes a sequence of C events corresponding to state S'. We call C the *change-of-state* variable.

In Section 3, we defined C as the mean plus one standard deviation of the length of error bursts in the trace. In this section we will analyze our choice on the value C, and provide an algorithm to optimize the value of C. We use the *GSM_E* trace for our analysis.

We first calculate the mean and standard deviation for the error burst length in *GSM_E*. For this trace, the mean value was found to be 6 frames and the standard deviation was 17 frames, yielding a *state-of-change* constant value C of 23 $(6 + 17)$ frames. With $C = 23$, we first identify lossy states as described in Section 3, and then concatenate all lossy states together to form the lossy subtrace. To prove that the resulting lossy trace is a stationary process, we apply the Runs Test described in Section 3.1. Figure 6 shows that 90.5 percent of the runs distribution lie between the 0.05 and 0.95 cut-offs. Therefore, this result proves that the lossy subtrace, constructed with a C value of 23, is a stationary process for a window size of 60. Recall from Section 3 that *GSM_E* only had 21.2 percent of the runs distribution between the boundary points.

Next, in order to optimize the C value, we developed an algorithm that takes an original non-stationary trace and executes the Runs Test for a large range of C values. The goal is to find the largest C value that yields a stationary lossy subtrace.

Table 2, shows the the percentage of runs distribution between the boundary points for various C values between 21 and 25. We are interested in obtaining the largest C value that gives 90 percent distribution. Table 2 illustrates that

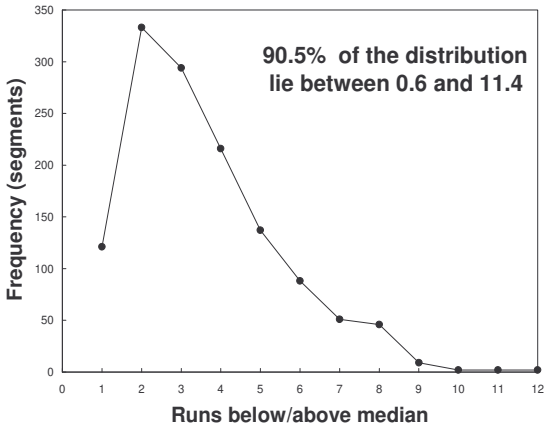

Fig. 6. The Runs Test applied to lossy subtrace.

Variable C	**Percentage**
25	89.3
24	90.7
23	90.5
22	91.5
21	91.4

Table 2
Percentage of runs distribution between boundary points for a range of C values.

choosing any value smaller that 25 yield a stationary lossy subtrace. Therefore, our intuitive choice of 23 was inside this optimal range of values. In fact, choosing any C value close to 23 will yield stationarity.

Decreasing the window size in the Runs test puts more restriction in the stationary behavior. The smaller the window size, the smaller the C value would have to be to obtain stationary subtraces.

5.3 The Markov-based Trace Analysis Algorithm

The basic concept behind the Markov-based Trace Analysis (MTA) algorithm [2] is that a trace can be decomposed into the lossy and error-free states described in Section 3. The lossy states are concatenated to form the lossy subtrace, while the error-free states are concatenated to form the error-free subtrace. Lossy subtrace exhibits stationarity and it can be modeled using a high-order DTMC. Next, the MTA algorithm models lossy subtrace as a DTMC and computes the memory and transitions probabilities.

The last step of the MTA algorithm is to determine the best fitting distribution

for the lengths of both lossy and error-free states. MTA approximates the states' lengths distribution using an exponential distribution function and computes the exponential function's parameters using a fitting function. The Cumulative Distribution Function (CDF) of the empirical trace is plotted along with exponential distributions with parameter values ranging from 0 to 1 in steps of 0.001. MTA then chooses the exponential parameter that yields a CDF curve that is the best approximation to the empirical CDF curve. The best approximation is determined by calculating the correlation coefficient, as explained in Section 6, between the original CDF curve and the exponential approximations.

We define two random processes with a discrete space $E = \{0, 1, 2, ...\}$:

- The *lossy state length* process $\{B_n \mid n \geq 0\}$, where B_n represents the number of elements in the n^{th} *lossy state*, (*i.e.*, the length of the state).
- The *error-free state length* process $\{G_n \mid n \geq 0\}$, where G_n represents the n^{th} *error-free state* length.

The application of the MTA algorithm to an input trace can be summarized as follows:

(1) Calculate the mean (m_e) and standard deviation (sd_e) values for error burst lengths in the trace.
(2) Set C, the *change-of-state* variable, equal to $m_e + sd_e$.
(3) Partition the trace into *lossy state* and *error-free state* portions using the following definitions:
   - *Lossy state*: runs of 1's and 0's, with the first element being a 1, and with runs of only 0's that have length less than or equal to the C.
   - *Error-free state*: runs of only 0's that have length greater than C.
(4) Create *lossy sub-trace* by concatenating the lossy state portions of the error trace.
(5) Model *lossy sub-trace* as a DTMC, and calculate its order and transition probabilities.
(6) Determine the best fitting exponential distributions for the length processes B_n and G_n.

5.4 The Multiple States MTA

The Multiple states MTA (MMTA) modeling algorithm is the most recent application of our data preconditioning methodology. Unlike the MTA algorithm, the MMTA algorithm is capable of modeling traces with two or more data patterns and non-exponential state length distributions. The MMTA views each data pattern as a state, and it models the transition among states with a high order DTMC. Using the data preconditioning approach, the MMTA al-

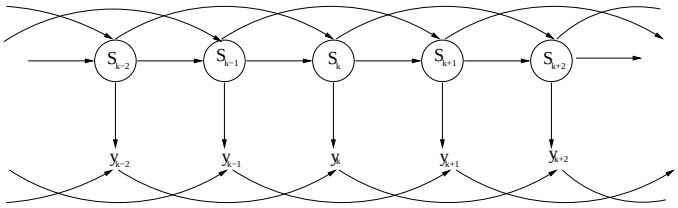

Fig. 7. Bayesian Network of a 2nd Order MMTA Model.

gorithm concatenates subtraces from each of the same states encountered in the original trace to form subtraces, and then models each subtrace with a higher order DTMC. Figure 7 shows the Bayesian network representation of a MMTA model of order 2.

In Section 3, we identified two hidden states in our network traces (*i.e.*, the error-free and lossy states). Using this observation, we summarize the steps of the MMTA algorithm as follows:

- Similar to the method used for MTA, MMTA first identifies the states in the original trace and creates subtraces by concatenating states of the same type.

(1) Create lossy subtrace from the lossy state portions of the error trace.
(2) Model lossy subtrace as a DTMC, and calculate its order and transition probabilities.
(3) Model the error-free state as a deterministic process, where each element is 0.

- Next, MMTA determines the transitions between error-free and lossy states:

(1) Create *state trace*. This trace corresponds to the collected dataset (*e.g.*, GSM_E trace), with lossy states (as defined by the first step) replaced by all 1's and error-free states (as defined by the first step) remaining all 0's.
(2) Model state trace as a DTMC, and calculate its order and transition probabilities.

In summary, the MMTA algorithm applies traditional Markov process properties to local stationary data by identifying stationary regions and modeling these regions and the transition between them using DTMCs.

6 Model Accuracy and Validation

The MTA and MMTA models add to an already long list of existing models. Each model has an associated computational cost and complexity, and its own level of accuracy. Given a dataset, researchers want to choose the more accurate and least complex model, but lack a clear metric of model accuracy.

In this section, we present three mechanisms to solve this dilemma. First, we

describe an approach for evaluating the accuracy of a particular model. Next, we describe a way to determine the minimum size of a collected trace necessary to extract model parameters for a specific network. Finally, we provide a process that determines whether created models are representative of a particular network path scenario and metric of interest.

6.1 Measuring Model Accuracy

We illustrate our model accuracy metric by comparing the model accuracy of two classical models (*i.e.*, Gilbert and 4^{th} order HMM) and two data preconditioning algorithms (*i.e.*, MTA and MMTA). Using each model with the collected traces in Table 1, we can generate artificial traces and compare each their resulting statistics with those of the original trace. We then quantify the accuracy of each model, by first plotting the error and error-free burst Cumulative Distribution Functions (CDF) for each artificial trace. We then calculate for each trace the correlation coefficient (*cc*) [13] between the CDFs of original trace and the CDFs of the artificial trace from the model. We use the *cc* as a measure of how closely each artificial trace approximates the original trace. A *cc* of 1 signifies that the two traces experience the same error or error-free statistics, while a *cc* of 0 indicates no statistical correlation between the traces.

To better understand the relationship between *cc* values and model accuracy, we calculated the error burst statistics of several artificial traces and computed their *cc* values for a given reference trace. First, we generate a reference trace with a fixed set of $(L_{exp}, EF_{exp}, L_{den})$ values of (0.006, 0.1, 1.0). Next, we generate artificial traces by changing the value L_{exp} from 0.0065 to 0.02 in steps of 0.0005, while keeping EF_{exp} and L_{den} constant (*i.e.*, $(EF_{exp}, L_{den}) = (0.1, 1)$), and computing the associated *cc* value for each artificial trace. Finally, using the reference trace's mean error burst size as a reference point (*i.e.*, 173 frames), we plot the mean error burst and its percentage reduction (relative to the reference trace's error burst size) for each observed *cc* value (see Figure 8). Thus the proportional reduction indicates the decrease in size of the mean error burst of an artificial trace relative to the mean error burst of the reference trace. Figure 8 shows that an artificial trace with a *cc* of 0.99 yields a mean error burst of 160 frames or only an 8 percent reduction. As the *cc* decreases, the percentage of reduction increases, and *cc* values smaller than or equal to 0.96 will yield percentages greater than or close to 50 percent. Based on these observations, we choose to associate *cc* values smaller than or equal to 0.96 (*i.e.*, mean percentage reduction greater that 50 percent) with inaccurate models.

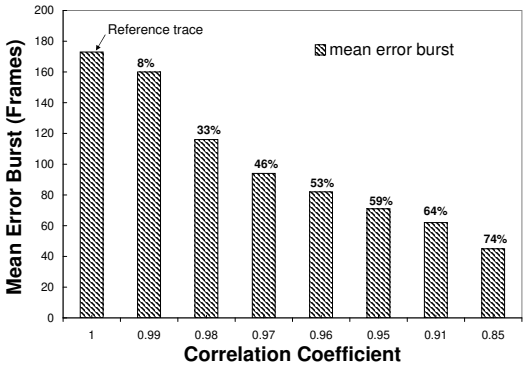

Fig. 8. Mean error burst and percentage reduction for different correlation coefficient values.

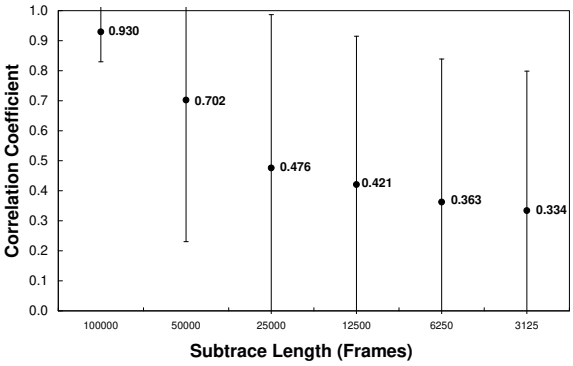

Fig. 9. *WLAN_E* error path modeling: mean and standard deviation correlation coefficient values for different subtrace lengths.

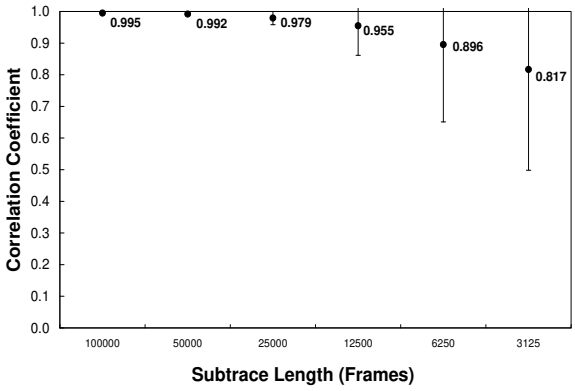

Fig. 10. *GSM_E* error path modeling: mean and standard deviation correlation coefficient values for different subtrace lengths.

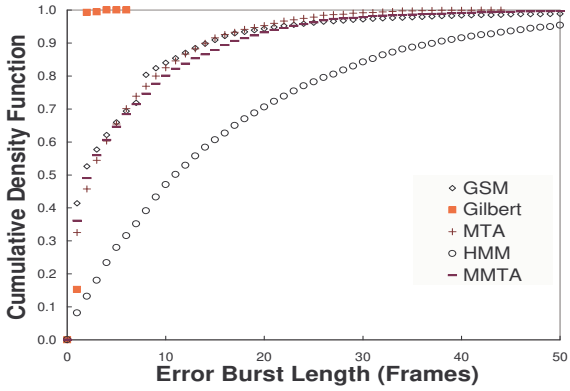

Fig. 11. Error burst distribution for *GSM_E* model.

6.2 Minimum Trace Length for Accurate Modeling

Another important aspect in the generation of accurate models is determining the minimum trace length required to precisely capture model parameters. To address this issue, we provide the following analysis method. Given a specific network path, scenario, and metric of interest, we collect a very large trace, (*e.g.*, a 200,000 frame trace representing over an hour's worth of data), we call this trace the *reference trace*. Next, we calculate the maximum error-free burst (*max_EFB*) encountered in this trace. If *max_EFB* is close to the size of the collected trace (*i.e.*, 200,000 in this case), then a larger trace must be collected. Once we have the typical *max_EFB* and a reference trace of length *ref_len*, we divide this trace into subtraces of sizes $\frac{ref_len}{2^j}$, where $j = 1, 2, 3, ..., m$. The maximum value of j (*i.e.*, m) is chosen such that $\frac{ref_len}{2^j} > 1,000$ frames. For example, a reference trace of 200,000 frames will generate 2 subtraces of

100,000 frames, 4 subtraces of 50,000 frames, 8 subtraces of 25,000 frames, 16 subtraces of 12,500 frames, 32 subtraces of 6,250 frames, 64 subtraces of 3,125 frames, and 128 subtraces of 1,562 frames (*i.e.*, $m = 7$ is the maximum value that yields a subtrace length greater than 1,000 frames). Then, we calculate the *cc* value of each subtrace to the reference trace. The *cc* value indicates the degree of statistical correlation between the subtraces and the reference trace. As previously discussed, a *cc* of 0.96 or less signifies an inaccurate model, therefore a subtrace with such a *cc* value should not be used to obtain a model's parameters.

As an example, we perform this analysis on *WLAN_E* and *GSM_E*. First, we calculate their *max_EFB* values to be 81,493 and 20,447, respectively. We then take the first 200,000 frames of each trace to construct the reference traces, *ref_WLAN_E* and *ref_GSM_E*. We choose $m = 6$, which generates a total of 126 subtraces of similar and different lengths. For reference traces *ref_WLAN_E* and *ref_GSM_E*, Figures 9 and 10 illustrate the mean and standard *cc* values for each subtrace length. For *GSM_E*, subtraces of sizes as small as 25,000 frames yield *cc* values greater than 0.96. Subtraces of size equal or smaller than 12,500 frames can give *cc* values greater that 0.96, but there is a greater chance that the *cc* value will be smaller than 0.96. For *WLAN_E*, any trace smaller than 100,000 frames will have a high probability of having a *cc* value smaller than 0.96, and even the 100,000 length subtraces have some likelihood of having *cc* values of 0.96 or less. From this analysis, we conclude that given a particular path, the minimum length required to extract the model parameters is a somewhat arbitrary choice that depends on the path's typical *max_EFB*. A reasonable, safe length would be to use a trace of length equal to or greater than the double of the *max_EFB*. For *WLAN_E*, the doubled *max_EFB* is 162,986, which is greater than 100,000 frames, the maximum subtrace length that we found in our earlier analysis. For *GSM*, the doubled *max_EFB* is 40,894, and our analysis shows than any length equal to or greater than 25,000 will lead to accurate model parameters.

6.3 Modeling Technique Validation

The final step in validating our modeling methodology is to guarantee that a generated model accurately capture the statistical properties for the metric of interest on the given network path. The model should accurately describe the statistical properties of additional traces collected from the same network. To verify this, we run our algorithm on a subsection of a reference trace and use the model to create an artificial trace. We then compare the statistics of the artificial trace to those of other subsections of the reference trace and the entire reference trace as a whole.

We extracted 200,000 frames from *GSM_E* trace, and call this reference trace *AB*. We divided *AB* into two subtraces of 100,000 frames each, and called these subtraces *A* and *B*. Next, we calculate the best model for subtrace *A* using the *cc* metric to determine model accuracy (see Section 6.1). The MMTA model yielded the highest *cc* value, therefore we chose this model to create a 100,000 frame artificial trace $MMTA_A$.

To determine the accuracy of the statistics of artificial trace, $MMTA_A$, we calculated the *cc* of the error burst and error-free burst CDFs (see Section 6.1) between $MMTA_A$ and traces *A* (**0.98** and 0.90), *B* (**0.98** and 0.95), and *AB* (**0.99** and 0.93). The computed *cc* values between $MMTA_A$ and *A* and between $MMTA_A$ and *B* are relatively close in value (especially for for error bursts), which indicates that the artificial trace generated by MMTA accurately models other regions of the reference trace.

This analysis shows that our model generation technique is not biased by a particular section of a trace we are analyzing, but rather it demonstrates that a captured trace can be used to accurately model the statistics of a particular network characteristic over a long period of time.

7 Choosing the Best Network Path Model

In this paper we have presented two classical models and two data preconditioning models that capture the error and error-free statistics of network traces. In this section, we apply the model validation methods described in the previous section to the collected traces listed in Table 1. We show that the various models yield differing degrees of accuracy when used to emulate different metrics on different network paths. We then compare the computational complexity and performance of the various models.

7.1 Choosing Accurate Models for Collected Traces

For each of the collected traces in Table 1, we determined the model parameters for the two classical and two data preconditioning models. We list the *cc* values for the error and error-free bursts CDF of the traces, the best model choice, and the associated best average *cc* value in Table 3. Examining the error burst CDF *cc* values for the *GSM_E* trace shows values for the Gilbert, HMM, MTA, and MMTA models of 0.74, 0.89, 0.99, and 0.99 respectively. As we discussed in the previous section, *cc* values less than or equal to 0.96 indicate models that poorly capture the statistics of the network and metric being investigated. To better clarify the differences between a *cc* of 0.99 and a *cc* of

Trace	Gilbert	HMM	MTA	MMTA	Best Model	Best Average *cc*
IP_1	**0.99**, **0.98**	**0.99**, 0.66	0.72, 0.95	**0.99**, **0.98**	Gilbert or MMTA	**0.99** or **0.99**
IP_2	0.92, 0.81	0.19, 0.68	0.95, 0.62	**0.98**, 0.94	MMTA	0.96
IP_3	**0.99**, **0.99**	**0.98**, 0.75	0.76, 0.96	**0.99**, **0.98**	Gilbert	**0.99**
WLAN_E	0.92, 0.74	0.73, 0.51	**0.99**, 0.87	**0.99**, 0.73	MTA	0.93
WLAN_D	0.93, 0.80	0.29, 0.37	**0.99**, 0.54	**0.98**, 0.95	MMTA	**0.97**
GSM_E	0.74, 0.92	0.89, 0.92	**0.99**, 0.96	**0.99**, 0.94	MTA	**0.98**
GSM_D	0.27, 0.74	0.71, 0.96	0.91, 0.84	0.82, 0.82	MTA	0.88

Table 3
Artificial traces, their correlation coefficient (error burst CDF, error-free burst CDF), best model(s), and average correlation coefficient for best model(s).

Trace	Original	Gilbert	HMM	MTA	MMTA
IP_1	23, 1, 0	5, 1, 0	7, 1, 0	62, 4, 4	13, 1, 0
IP_2	6374, 2, 80	4, 1, 0	594, 102, 103	37, 2, 4	169, 2, 9
IP_3	13, 1, 0	5, 1, 0	7, 1, 0	34, 3, 3	10, 1, 1
WLAN_E	42, 2, 3	4, 1.67, 0.54	140, 13, 15	23, 2, 2	28, 2.68, 2.68
WLAN_D	2212, 4, 37	8, 1, 1	1448, 194, 206	61, 4, 6	122, 4, 8
GSM_E	626, 6, 17	6, 1.86, 0.40	124, 16, 16	44, 5, 6	72, 6.37, 8.21
GSM_D	38,20,11	2,1.5,0.87	36,12,12	7,3,3	52,26,18

Table 4
Original and artificial traces' *error burst* statistics: maximum, mean, and standard deviation.

0.74, we plot the error burst CDF for the *GSM_E* trace models in Figure 11. Examining this figure, we can see that the CDFs for the Gilbert and the HMM model are not good approximations to the real distribution, therefore we may conclude that *cc* values of 0.74 and 0.89 indicate poor correlations between the artificial traces and the actual trace. On the other hand, a *cc* value of 0.99 yields a very good approximation.

Tables 4 and 5 show the maximum, mean, and standard deviation values of the error and error-free bursts for the original and artificial traces for each of the models. Note that those models with mean values that are similar to the reference traces' mean values in general have higher *cc* values.

Overall, the results show two important observations: different models have varying degrees of success in capturing the statistical properties of different metrics for different networks, and as shown by the modeling of *IP_2* and

Trace	Original	Gilbert	HMM	MTA	MMTA
IP_1	977,40,70	383,121,90	3500,1033,791	260,55,46	486,156,118
IP_2	3079,50,193	404,239,195	5973,1400,1220	15205,3743,3251	5769,325,489
IP_3	607,17,27	146,81,66	678,254,193	149,45,38	240,68,51
WLAN_E	81493,42.00,1306	393,195,159	1799,415,356	331,63,53	2689,219,258
WLAN_D	5893, 11, 132	148, 50, 40	2295, 1724, 1558	2094, 294, 305	1830, 42, 90
GSM_E	20447,114,550	888,535,438	3258,654,563	2927,477,420	3453,574,550
GSM_D	907,347,253	1107,2805,2270	523,674,488	2160,4516,3528	864,1688,1194

Table 5
Original and artificial traces' *error free burst* statistics: maximum, mean, and standard deviation.

GSM_D, we still need better models for capturing network path behaviors. The Gilbert model performs well when modeling wired IP networks. Surprisingly, however, it is not always accurate for IP networks (*e.g.*, *IP_2*). The HMM model accurately captures error bursts in some wired networks, but is fairly inaccurate at modeling wireless networks. The data preconditioning models perform well at modeling many of the networks, especially the error burst portions. However, in general, as shown in Table 5, they are not as accurate in modeling the error free bursts. Note that the same observation is true for both the Gilbert and HMM models. We believe that future research should focus on optimizing the modeling of *both* error burst and error free burst behavior.

7.2 Model Computational Complexity

Another important feature to consider when choosing a network model is the model's computational complexity. One measure of the complexity of a model is its execution time. For example, on a 1.8GHz Intel Pentium 4 processor, the modeling of the *IP_1* trace took 8 seconds using the Gilbert model, 57 seconds using the HMM model, 7 seconds using the MTA algorithm, and 59 seconds using MMTA. Note that the MMTA uses two 4^{th} order DTMCs, resulting in a total of 32 states. The HMM model consists of a single 4^{th} order DTMC, and it calculates the output according to the state. The cost of the HMM is similar to the MMTA model. The MTA model consists of one small 4^{th} order DTMC for modeling the lossy subtrace portion of the trace, while the Gilbert model uses one large 1^{st} order DTMC for modeling the original trace. The MTA model has a lower computation cost than the Gilbert because it only needs to calculate the transition probability for the lossy subtrace, which is a much smaller trace than the original trace. Overall, we observe that the MMTA is the highest cost model.

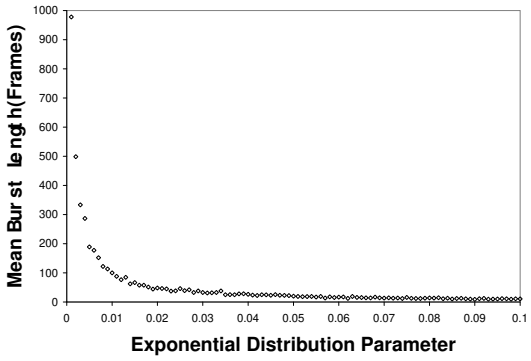

Fig. 12. Mean burst length versus exponential distribution function parameter.

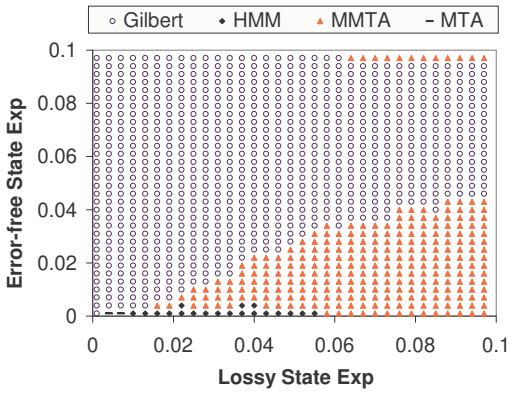

Fig. 13. Optimal model for L_{den}=0.2.

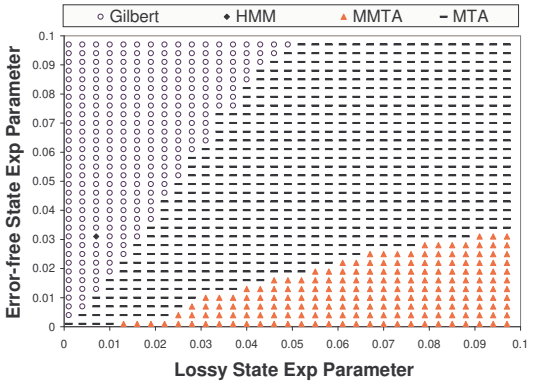

Fig. 14. Optimal model for L_{den}=0.4.

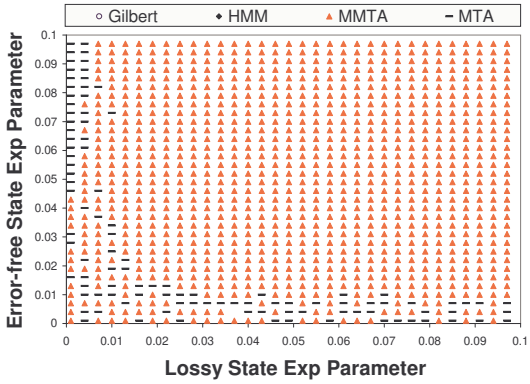

Fig. 15. Optimal model for L_{den}=0.7.

Thus, the choice of model may also depend on the type of simulation being done. If a trace can be generated in advance, model complexity will be less of an issue. However, for real-time trace generation, developers may need to consider both the complexity and the accuracy of a model.

8 Determining the Domain of Applicability

In this section, to better understand the behavior of each of the four models, we observe them while they attempt to capture the properties of a synthetic network. We first use the three parameters for classifying traces (L_{exp}, EF_{exp}, L_{den}, defined in Section 3) to capture the properties of a synthetic network and network characteristic of interest, and then identify the domain of applicability for each model: *for a given characteristic of a trace, which model performs best at modeling that characteristic?*

We answer this question with the following process. We begin by generating artificial traces (using a method described below) for various values of L_{exp}, EF_{exp}, and L_{den}. Next, for each model and each trace, we calculate the *cc* for the error and error-free burst CDFs, and the average value of these two *cc* values. Note that the accuracy of the *cc* for the error bursts CDF is equally as important as the accuracy of the error-free burst CDF. However, one could add a weight to either one depending on the importance of obtaining the correct distribution accuracy for each burst type. For example, in Table 3 for the *IP_1* trace, the Gilbert, the HMM, and the MMTA models give a *cc* for the error burst distribution of 0.99, however, the *cc* for the error-free burst distribution in the HMM is only 0.66.

To generate artificial traces for our exploration of domain analysis, we first choose three fixed values for the parameter L_{den} of 0.2, 0.4, and 0.7, while for the L_{exp} and EF_{exp} parameters, we vary the values of each from 0.001 to 0.1 in steps of 0.001. We use the fixed L_{den} values to generate Bernoulli process-based random errors inside the lossy state. Note that this means that inside a lossy state the occurrence of errors are memoryless (*i.e.*, the next frame's value doesn't depend on the previous frame's value). The effect of using a Bernoulli process to generate errors is, for small values of L_{den}, that it biases the domain analysis results towards the simpler Gilbert model, instead of more complex higher order models. However, as the value L_{den} increases, so does the likelihood of occurrence of multiple consecutive errors; and thus, the bias switches towards higher order models. Since most real network traces will experience some degree of memory, using them for domain analysis would yield results that were almost always biased towards memory process-based models. Thus, we choose an artificial trace generation method that will allow us to explore the full range of domain analysis and results.

We determine the lossy and error-free bursts lengths by using the inverse transformation method [17]. Given a random variable X with a CDF $F(x)$, the variable u is uniformly distributed between 0 and 1. We can generate a sample value of X by generating u and calculating $x = F^{-1}(u)$. For an exponential function with parameter α, $u = F(x) = 1 - e^{-\alpha x}$. Thus, we can determine x from $x = -ln(u)/\alpha$.

We summarize the algorithm for generating an artificial trace as follows:

(1) Choose the number of frames, N, to generate in the artificial trace.
(2) The algorithm repeats the following steps until all N frames have been generated:
    (a) Determine g_{len}, the error-free state length from the error-free state

length distribution (*i.e.*, exponential distribution function with parameter EF_{exp}).

(b) Determine b_{len}, the lossy state length from the *lossy state length* distribution (*i.e.*, exponential distribution function with parameter L_{exp}).

(c) Generate g_{len} error-free frames (*i.e.*, a sequence of "0" of length g_{len}).

(d) Generate b_{len} frames, where each frame is an error frame with probability L_{den}.

In examining the artificial trace generator's results, it is important to consider that some of the parameter values explored by the trace generator are not found in real networks. As a point of reference, Table 1 shows the parameter values for several sample traces of real networks.

In this section, we introduce a methodology that will allow us to choose the most accurate models for a wide variety of network traces. To this end, we generate a large number of synthetic traces by varying the three parameters $(L_{exp}, EF_{exp}, L_{den})$, defined in Section 3. We then generate Domain Applicability Plots (DAP) to show the most accurate model for each combination of L_{exp}, EF_{exp}, and L_{den}, where the best model is defined as the model with a corresponding maximum average *cc* value for the error and error-free bursts. Since we cannot show three-dimensional plots, we choose two representative values for L_{den} (0.2, 0.4 and 0.7), and perform experiments that vary across the L_{exp} and EF_{exp} parameters, both varying from 0.001 to 0.1 in steps of 0.001.

Figures 13, 14, and 15 show the DAPs for L_{den} values of 0.2, 0.4, and 0.7, respectively. Observe that, for $L_{den} = 0.2$ (see Figure 13), the Gilbert model is best for a large portion of the graph. The result is as we expected because of the use of a Bernoulli process to generate losses in the lossy state. Here, the error burst length is relatively small. As a result, for a large portion of points in this plot, the Gilbert model is the optimal choice. However, as the probability of error in the lossy state L_{den} increases, the error burst length increases and thus, the region occupied by the Gilbert model decreases and the MMTA and MTA become better choices.

Further examination of the results shows that the mean *cc* value in this area for the Gilbert model is 0.99, while for this same region the mean *cc* value for the MMTA model is 0.98 (see Table 6). Thus, while the Gilbert model yields the best results, the M^3 also performs very well for this "optimal-Gilbert" region (see Section 6 for an explanation of the relationship between *cc* values and a model's accuracy). For the region where the MMTA is optimal (the "optimal-MMTA" region), the mean *cc* value for the MMTA model is 0.97, while the mean *cc* for the Gilbert model in this region is 0.96. In Section 6, we showed that *cc* values smaller than or equal to 0.96 yield inaccurate models.

	Optimal Model Region						
	$L_{den} = 0.2$		$L_{den} = 0.4$			$L_{den} = 0.7$	
Model	**Gilbert**	**MMTA**	**Gilbert**	**MTA**	**M^3**	**MTA**	**MMTA**
Gilbert	**0.99**	0.96	**0.99**	0.96	0.91	0.90	0.92
HMM	0.90	0.92	0.89	0.91	0.92	0.64	0.77
MTA	0.89	0.82	**0.97**	**0.98**	0.91	**0.99**	**0.97**
MMTA	**0.98**	**0.97**	0.96	0.96	**0.97**	**0.99**	**0.98**

Table 6
Correlation coefficient for each L_{den} value (0.2, 0.4, 0.7) and each optimal region.

Therefore, we can conclude that, for this network, an L_{den} value of 0.2, using the MMTA model always yields highly accurate models, while the Gilbert model only performs best for a subset of the network parameter space.

Next, we examine the model choices for an L_{den} value of 0.4 (see Figure 14). In this DAP diagram, there are three optimal regions. In the "optimal-Gilbert" region, the mean *cc* value for the Gilbert model is 0.99. Table 6 shows the mean *cc* values for the other models in this "optimal-Gilbert" region. The MTA model performs the best over the largest region of the plot, with a mean *cc* value for the model of 0.98. The other models in this "optimal-MTA" region have mean *cc* values of less than or equal to 0.96, which indicates that they are inaccurate representations for these regions. For the "optimal-MMTA", the mean *cc* value for the model is 0.97, while the other models for this region have mean *cc* values of less than 0.93 (*i.e.*, they are inaccurate models for this region).

Finally, we examine the model choices for a high value of L_{den}, 0.7. For this high value, almost the entire DAP diagram consists of an "optimal-MMTA" region with a mean *cc* value for the model of 0.98. In this region, the MTA model's mean *cc* value was 0.97, which is also very good, while both the Gilbert and HMM perform very poorly. We believe that this result can be explained as the inability of traditional models to capture the long error bursts inside lossy states. In contrast, the data preconditioning models are capable of accurately capturing both low and high error densities inside lossy states.

9 Conclusion

Our work seeks to aid network and application protocol developers in developing and choosing appropriate models for network simulation. We introduce our data preconditioning methodology for modeling non-stationary datasets, and

present the new Multiple states MTA model (MMTA). We show that MMTA is better in capturing error burst statistics than classical models and more consistently accurate across different networks than our previous MTA model. The primary conclusion from our analysis of existing models is that classic modeling techniques work well for some, but not all wired networks. However, when modeling delay and losses in wireless networks, the data preconditioning approaches are more accurate.

The main contribution of this paper are methodologies to evaluate the accuracy of models, to choose the best models for a given network, and to evaluate modeling techniques. Using our methodology and Domain of Applicability Plots (DAP), researchers can quickly evaluate any analytical model for a given network characteristic.

References

[1] S. Floyd, E. Kohler, Internet research needs better models, in: Hotnets-I, October 2002.

[2] A. Konrad, B. Zhao, A. Joseph, R. Ludwig, A Markov-based channel model algorithm for wireless networks, in: Proceedings of the Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), 2001.

[3] R. Ludwig, A. Konrad, A. Joseph, Optimizing the end-to-end performance of reliable flows over wireless link, in: Proceedings of ACM/IEEE MobiCom, 1999.

[4] Y. Zhang, V. Paxson, S. Shenker, The stationarity of internet path properties: Routing, loss, and throughput, in: ACIRI Technical Report, 2000.

[5] S. Molnar, A. Gefferth, On the scaling and burst structure of data traffic, in: 8th International Conference on Telecommunication Systems, Modelling and Analysis, 2000.

[6] W. Leland, M. Taqqu, W. Willinger, D. Wilson, On the self-similar nature of ethernet traffic, in: IEEE/ACM Transaction on Networking, 1994.

[7] J. Bolot, S. Fosse-Parisis, D.Towsley, Adaptive FEC-based error control for internet telephony, in: Proceedings of Infocom, ACM, 1999.

[8] M. Yajnik, J. Kurose, D. Towsley, Packet loss correlation in the MBone multicast network: Experimental measurements and Markov chain models, UMASS COMPSCI Technical Report 95-115.

[9] G. T. Nguyen, R. Katz, B. Noble, A trace-based approach for modeling wireless channel behavior, in: Proceedings of the Winter Simulation Conference, 1996, pp. 597–604.

[10] H. Balakrishnan, R. Katz, Explicit loss notification and wireless web performance, in: Proceedings of the IEEE Globecom Internet Mini-Conference, 1998.

[11] A. Willig, M. Kubisch, A. Wolisz, Measurements and stochastic modeling of a wireless link in an industrial environment, in: Technical Report TKN-01-00, Technical University Berlin, 2001.

[12] M. Zorzi, R. R. Rao, On the statistics of block errors in bursty channels, in: IEEE Transactions on Communications, 1997.

[13] J. Bendat, A. Piersol, Random Data: Analysis and Measurement Procedures, John Wiley and Sons, 1986.

[14] B. Kedem, Binary Time Series, New York and Basel, 1980.

[15] I. L. MacDonald, W. Zucchini, Hidden Markov and Other Models for Discrete-valued Time Series, 1st Edition, Chapman & Hall/CRC, 1997.

[16] J. Pearl, Probabilistic Reasoning in Intellegent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.

[17] R. Jain, The Art of Computer Systems Performance Analysis, John Wiley and Sons, 1991.