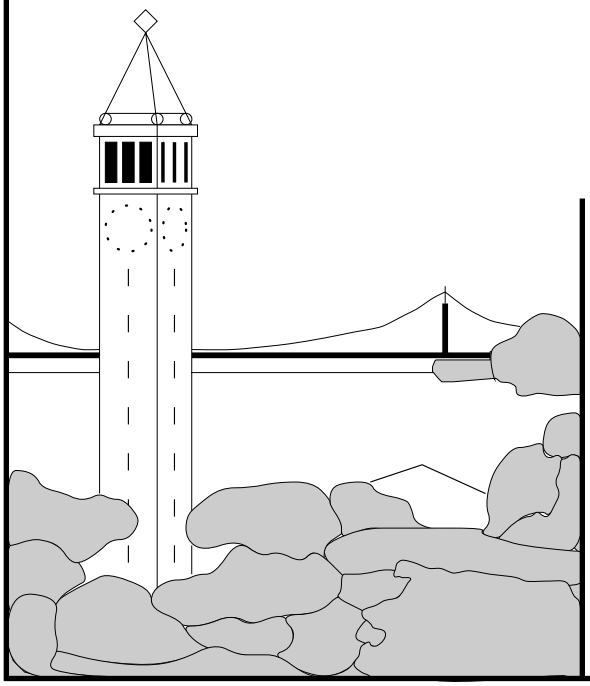
Choosing an Accurate Network Path Model

Almudena Konrad, Anthony D. Joseph



Report No. UCB/CSD-03-CSD-03-1236

May 2003

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Choosing an Accurate Network Path Model

Almudena Konrad, Anthony D. Joseph *

Abstract

*In this paper, we present a novel approach that enables network researchers to quickly select the most accurate modeling and analysis method for a given wired or wireless network path and network characteristic of interest (*e.g.*, delay, loss, or error process). Amongst the network models that our approach includes in its analysis are two data preconditioning models that we have developed as a part of the* Tapas *project, an investigation into new approaches for accurately modeling and analyzing the behavior of various time-varying network path characteristics. 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. We introduce a new approach to the modeling of network characteristics, the data preconditioning methodology, and present the latest application of this methodology, the* Modified hidden Markov Model *(M^3). Our domain analysis methodology defines and classifies binary network traces (*i.e.*, traces which describes the occurrence or the lack of occurrence of a network event over time), and using these classifications, it determines the most accurate model or models from a set of models.*

1 Introduction

Perhaps the most common method for evaluating application and network protocol designs while they are under development is the use of network link simulation and emulation. Simulation and emulation are low-cost techniques that enable networking researchers to quickly, and in a repeatable manner, explore the behavior of a network or application protocol under a variety of network conditions (*e.g.*, varying loss, delay, and error conditions). However, the behavior and thus, the results and performance, of many protocols depends on the characteristics of the network conditions. Designers of such algorithms and protocols make assumptions about the way in which a particular network characteristic varies and encode these assumptions into the algorithms.

For example, a detailed understanding of the packet loss process and burstiness of errors is necessary for the proper design and parameter tuning of error control protocols, such as Automatic Repeat reQuest (ARQ) protocols. Another example is modeling end-to-end delay in the Internet, a process that becomes significantly more complex when the network includes a wireless link. For real-time one-way or two-way audio/video applications, system and human perceptual tolerances dictate maximum acceptable transmission delays. One-way application delays are bounded by the willingness of a human to wait for playback to begin and by system resource limitations on the size of a reasonable playback buffer, while for two-way applications, delays are bounded by human interaction constraints of 200 milliseconds. In both types of applications, data packets with greater delays are discarded since they are no longer useful. The testing of the behavior of multimedia applications under varying delay conditions depends upon an accurate modeling of the delay behavior of the network under test.

Thus, we are led to one of the most important problems in statistics: the choice of an appropriate model for characterizing a given dataset. We encounter this same problem in computer networks, where many design decisions are the results of some chosen simulation parameters and models. Floyd and Kohler [6] argue that the use of inaccurate models in computer networks leads to flaws in Internet research. In analyzing computer networks, researchers are faced with measurements whose characteristics experience non-stationarity (time variability) and complex patterns due to a large number of factors, including both internal network elements and external events. Often, it is difficult to identify and thus, accurately model, the causes of these patterns. However, classical models have often worked surprisingly well in modeling composite events in traditional networks. As we will show in this paper, the characteristics of some of today's wired and wireless networks (*e.g.*, a packet that

*A. Konrad and A. D. Joseph are with the CS Division, U.C. Berkeley, Berkeley, California. Emails: {almudena,adj}@cs.berkeley.edu.

couldn't be transmitted through a noisy wireless channel) or network metrics being examined, has resulted in datasets that cannot be adequately characterized or modeled using classical techniques. In this paper we introduce a new modeling methodology, a data preconditioning technique which takes into consideration the time varying statistical properties of today's networks.

The traditional approach to modeling networks is the use of a classical network model, such as a Bernoulli, Gilbert, high-order Discrete Time Markov Chain (DTMC), or Hidden Markov Model (HMM). The choice of which model to use is usually an ad hoc one, often without adequate consideration of the statistical properties of each model. For example, the Bernoulli model is a memory-less process, which means that the output value at each iteration is independent of the previous outputs. Thus, output values will be evenly distributed in proportion to the model's probability value. However, a network with bursty behavior would not experience a matching even distribution of outputs. So, it would appear that for such a network, the Bernoulli model would not be an ideal choice, as it would not match the network's actual characteristics.

1.1 The Role of Accurate Modeling

The observation that different models may yield different statistical characteristics from a network and metric under investigation is important only if it is actually the case that the use of an accurate model (*i.e.*, one with the correct distribution) is critical to the correct design of protocols using simulation and emulation. However, we have already demonstrated the correctness of this belief in [3]. In that paper, we observed that a naive assumption about the error model used for simulation during protocol design could lead to a poor choice of value for the protocol's design parameters. In particular, we evaluated alternative logical frame sizes to the fixed data frame size used by the semi-reliable protocol, Radio Link Protocol (RLP), in a Global System for Mobility (GSM) digital cellular data network. More specifically, we were interested in determining the optimal logical frame size in poor signal coverage (*i.e.*, worst case conditions), one that balances the header and checksum overhead associated with each frame, with the time to retransmit an entire corrupted frame, to yield the maximum throughput. Notice that the distribution of errors (*e.g.*, bursty versus more even) will affect the choice of optimal frame size, since bursty errors favor larger frame sizes, while evenly distributed errors favor smaller frame sizes.

To summarize our work in [3], we collected radio link error traces from a commercial GSM network, calculated the throughput for different frame sizes, and determined that 210 bytes is the optimal frame size value for maximum throughput (see Figure 1). We then compared the results when assuming the error process is a Bernoulli process, Gilbert process, and 3rd order DTMC process. Using each error model, we generated an artificial trace of same length as the original GSM trace and computed the optimal frame sizes using each model. Using the Bernoulli model, the optimal frame size was found to be 60 bytes (*i.e.*, a 71 percent decrease in performance relative to the actual optimum value). The Gilbert and 3rd order Markov models yielded optimal frame sizes of 150 and 180 bytes, respectively. We then developed a data preconditioning algorithm, the Markov-based Trace Analysis (MTA) algorithm, which more accurately models the error distribution in the GSM network. Using the MTA error model, we generated an artificial trace that yielded an optimal frame size of 210 bytes (see Figure 1). There are two important observations to derive from this figure: the Gilbert and 3rd order Markov models predicted higher throughput values than expected, and performance of both for larger frame sizes is substantially less than that of the actual trace. The increased performance differences are relatively small in this case, however, they could lead to questions about performance if the system is deployed and peak performance does not match predicted performance. The decreased performance differences are important because the overall protocol design would most likely be a compromise between optimizing performance under both poor and good conditions. A designer who knows that performance under poor conditions would only be slightly reduced by increasing the frame size could choose a larger frame size, and increase overall performance with a small penalty for poor conditions.

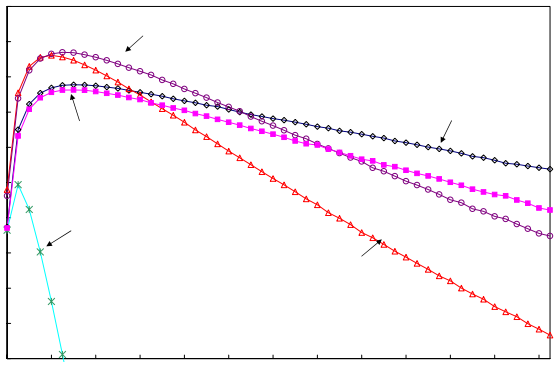


Figure 1. Optimal frame size versus error model.

While the results in [3] were specific to a particular network link and its loss process, in this paper, we generalize and apply our research to a broad spectrum of loss and delay networks path traces collected from three different types of

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.

wired and wireless networks, and the general challenge of domain analysis: *choosing the most appropriate model to use for modeling the behavior of an arbitrary network path and characteristic*. We also revisit our previously developed data preconditioning algorithm, the MTA algorithm, and introduce a new one, the Modified hidden Markov Model (M^3) algorithm. We then show how domain analysis can be used to choose a network model that best represents the characteristics of a given network path, a metric of interest, and scenario.

Overall, as we will show in this paper, the results are that classical models perform well for modeling some wired network paths, but surprisingly, not all. However, classical models are insufficient for modeling modern wireless network paths. In part, this is due to the more complex, bursty behavior of these networks. We confirm this property of classical models in a detailed exploration of their behavior in modeling a synthetic network in Section 8. The data preconditioning models are better at capturing the loss and delay behavior of networks, but are still lacking in some areas.

The rest of this paper is structured as follows. In Section 2, we define binary network traces and review the concept of stationarity. In Section 3, we discuss traditional modeling techniques, followed by a discussion of related work in Section 4. In Section 5, we review our data preconditioning technique and two algorithms based on this technique. We present our approach to evaluating model accuracy and our modeling methodology in Section 6, and in Section 7, we apply we apply these techniques to two types of network path traces collected from seven different networks. We introduce our domain of applicability selection technique in Section 8 and use it to evaluate the behavior of the various modeling techniques. Finally, we conclude with Section 9.

2 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 [4], we used the Runs Test developed by Bendat and Piersol [2] to show that GSM binary error traces are locally stationary binary time series [8], 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. In a loss trace, a 1 signifies a lost packet, while in an error trace, a 1 denotes a corrupted frame, and in a delay trace, a 1 means that the packet or frame arrived with a delay greater than some maximum threshold[1]. To generalize all these cases, whenever we encounter a 1 in a packet or frame trace, we will refer to it 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 domain analysis 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.* [15] during an uncongested IP connection from Massachusetts to Sweden. *IP_2* and *IP_3* are IP loss traces collected by Wenyu Jiang at Columbia University (CU). *IP_2* was collected during an uncongested connection from CU to GMD (the German National Research Center for Information Technology), and *IP_3* was collected during 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 [14]. 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, respectively. 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,

[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.

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

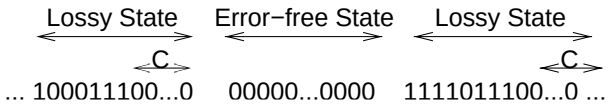


Figure 2. An error trace with lossy and error-free states.

while for *WLAN_D*, we chose a delay threshold of 20 milliseconds. Note that the delay statistics obtained in *GSM_D* and *WLAN_D* are the results of the effects from two links: the delays due to the radio link layer between the sender and the base station in the GSM or access point in the 802.11b network and the delays caused by the IP network. Analyzing each link in isolation might yield different models for each link. However, in this paper, we only analyze the end-to-end effects as a superposition of the effects from the two links. In future work, we plan to explore the differences, if any, between the composition of individual models and the generation of a superposition model. 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 2), by delineating the trace into states (clusters) with lossy states beginning with a first element of 1 and containing bursts of 1's and 0's. A lossy state 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 begins an error-free state, which is then terminated by the 0 preceding the next 1 element in the trace. The value of C is a design decision 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.

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 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.

2.1 Stationarity of Network Path Traces

In the Tapas project, we collect and model network path measurements in the form of binary traces of losses, errors, and delays as described in Section 2. Consider 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 [8]. 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. Thus, 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 traces' statistical properties vary over time. However, these traces exhibit local stationarity (*i.e.*, a non-stationary data set composed of deterministic regions and small stationary regions). In this paper, we show that neglecting the non-stationary aspects of network traces and attempting to fit traditional models onto these traces can lead to inadequate models that do not capture the many possible patterns of the data and their distribution.

We use the Runs Test 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 burst).

2. Divide the trace into segments of equal lengths (window size).

3. Compute the lengths of runs in each segment.

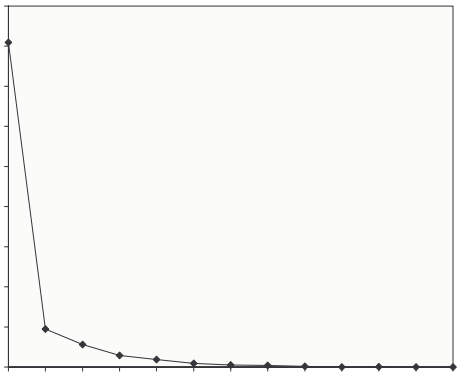


Figure 3. The Runs Test applied to lossy subtrace.

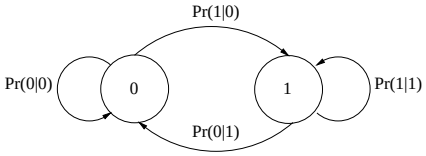


Figure 4. Gilbert model state transition diagram.

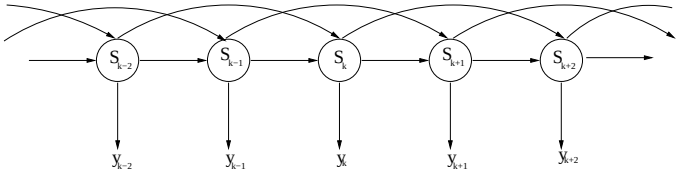


Figure 5. Bayesian Network of a 2nd Order Hidden Markov Model.

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 applied the Runs Test to *GSM_E* with window size of 60. Figure 3 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 a non-stationary process 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.

3 Classical Markov Models

In this section, we present the two types of classical stochastic models for characterizing the statistical properties of network traces that we examine in this paper. One is the Gilbert model, the well known and popular Markov process of memory one, and the second is the Hidden Markov Model (HMM) [10]. We discuss the reasoning behind our choices below.

3.1 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 4). 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 3rd order Markov model, a DTMC of order three (*i.e.*, 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 (*i.e.*, the model can model short and relatively long bursts of an event). However, even with this increase in accuracy, 3rd order Markov models don't always accurately capture real network statistical characteristics (see Figure 1).

3.2 The Hidden Markov Model

For the second model, we choose a HMM model because many statisticians believe that the non-stationary characteristic 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 it only depends on the current state, this is known as the output independence assumption. Figure 5 illustrates the Bayesian network [13] 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 4th or-

der hidden Markov model. The states $\{S_1, S_2\}$ correspond to the lossy and error-free states defined in Section 2, 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, but it significantly increases the computational complexity of the model.

4 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 measurements. Zhang *et al.* [16] study stationarity in the Internet and introduce a new notion of stationarity that is more relevant to network properties. They call a dataset *operational stationary*, if the statistics of interest remain within bounds considered operationally equivalent. Their most interesting finding is the observation 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.* [11] propose a simple approach for identifying stationary intervals and analyzing them independently. They introduce a new technique for identifying these intervals. Leland *et al.* [9] 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.* [5] 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.* [15] 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$^{\text{rd}}$ 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.* [12] 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 [1] also collected error traces from a Lucent 900 MHz WaveLAN network and developed a two-state Markov chain error model. Willig *et al.* [14] present a special class of Markov models, called *bipartite models*. Zorzi *et al.* [17] 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.

In summary, the Tapas project addresses the modeling of similar networks with non-stationary error and delay behaviors, providing a new modeling methodology. Our work is relatively novel in its approach of not only identifying datasets with non-stationary behavior, but also identifying stationary regions and modeling the entire dataset as a sequence of stationary components. We also focus on and demonstrate the importance of accuracy in network modeling. In previous work [3, 4], we have shown how assuming the wrong error distributions has led to incorrect design decisions. For example, as we show in the introduction, choosing an incorrect error model yields a suboptimal wireless frame size. Based on this and other observations, we argue that there is a need to develop methods for choosing the most accurate modeling algorithms that best describe and handle time-varying real world network characteristics and their statistics.

5 Modeling through Data Preconditioning

As we will show, classical modeling approaches are incapable of capturing all of the complexities and characteristics of some datasets. We introduce a new modeling methodology that supports a greater degree of behavior complexity in computer networks. We then describe two instances of this methodology, the Markov-based Trace Analysis (MTA) algorithm and the Modified hidden Markov Model (M^3) algorithm.

5.1 Data Preconditioning

The search for a better method for creating accurate analytical network models that take the non-stationarity behavior of networks into account leads us to propose a new research methodology. This methodology consists of the analysis and preconditioning of data *before* the data is fed into traditional models. Intuitively, we use pattern recognition to break down datasets that experience non-stationarity into subsets that exhibit stationary behavior, and hence are easier to accurately model with traditional models. For a particular network characteristic, we follow the process illustrated in Figure 6. 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 the network traces presented in Section 2, we identified two distinct states: lossy and error-free. Second, we concatenate trace regions with same states

[3]Each network behavior has certain statistical properties.

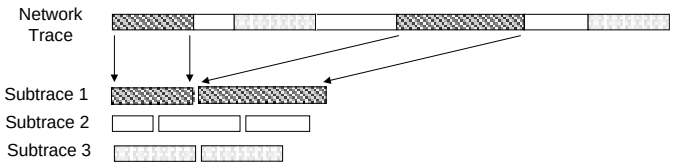


Figure 6. Data Preconditioning: in this example a network trace is decomposed into three subtraces, each consisting of a concatenation of a specific data pattern.

to form stationary subtraces of the original trace, (i.e., lossy and error-free subtraces). These subtraces have the property that they can be 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 particular in this paper, we demonstrate how this research methodology can be applied to significantly improve the accuracy of the modeling of the error and delay processes in wired and wireless networks.

5.2 Optimizing the *change-of-state* variable C

An important design desicion in our data preconditioning methodology is the choice of the *change-of-state* variable C. Our goal is to construct subtraces that experience stationarity for a given window size. In Section 2, 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 chooice on the value C, and we will provide an optimization algorithm to find the best possible value for C. We use *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 a C value of 23, we form lossy subtrace by first identifying lossy states, as described in Section 2, and concatenating all lossy states together. To prove that lossy trace is a stationary process we apply the Runs Test described in Section 2.1. Figure 7 shows that 90.5 percent of the runs distribution lie between the 0.05 and 0.95 cut-offs. Therefore, this result proves that lossy subtrace, constructed with a C value of 23, is a stationary process for a window size of 60. Recall in Section 2, *GSM_E* only had 21.2 percent of the runs distribution between the boundary points.

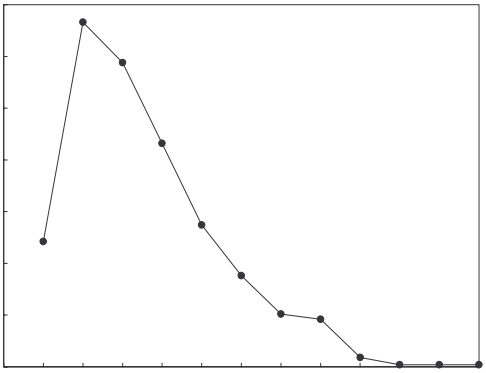


Figure 7. The Runs Test applied to lossy subtrace.

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.

Next, in order to optimize the C value, we developed an algorithm that takes an original non-stationary trace and it executes the Runs Test for a large range of C values. The goal is to find the greater 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 choosing any value smaller that 25 yield a stationary lossy subtrace. Therefore, our intituiticve 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, put 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 [4] is that a trace can be decomposed

into the lossy and error-free states described in Section 2. 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 Modified hidden Markov Model Algorithm

The Modified hidden Markov Model (M^3) modeling algorithm is the most recent application of our data preconditioning methodology. Unlike the MTA algorithm, the M^3 algorithm is capable of modeling traces with two or more data patterns and non-exponential state length distributions. Similar to a HMM, the M^3 views each data pattern as a hidden state, and it models the transition among states with a high order DTMC. Using the data preconditioning approach, the M^3 algorithm concatenates subtraces from each of the same hidden states encountered in the original trace to form subtraces, and then models each subtrace with a high order DTMC. Intuitively, this new algorithm can be viewed as a new type of hidden Markov process [10], where the output independent assumption is *not* taken. Figure 8 shows the Bayesian network representation of a M^3 model of order 2. In this diagram, we assume that $s_{k-2}, s_{k-1}, s_k, s_{k+1}, s_{k+2}$ are the same hidden state and, if we have several hidden states, each hidden state would generate a subtrace.

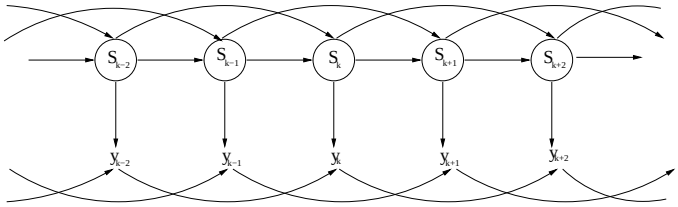


Figure 8. Bayesian Network of a 2nd Order M^3 Model.

In Section 2, 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 M^3 algorithm as follows:

- Similar to the method used for MTA, M^3 first identifies the states in the original trace and it creates subtraces by concatenating same states.
  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, M^3 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-tree 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 M^3 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

In this section, we provide three necessary mechanisms: an approach for evaluating the accuracy of a particular model, a method for determining the minimum size of a collected trace that is necessary to extract model parameters for a specific network, and a process for validating that the created models are representative of a particular network path scenario (*e.g.*, poor signal quality, uncongested, etc.) and metric of interest.

6.1 Measuring Model Accuracy

We are interested in evaluating model accuracy for two classical models (*i.e.*, Gilbert and 4^{th} order HMM) and two data preconditioning algorithms (*i.e.*, MTA and M^3). Using each model with the collected traces in Table 1, we can generate artificial traces and compare each trace's resulting statistics with the original traces' statistics. We then need to quantify the accuracy of each model. We do this by first plotting the error and error-free burst Cumulative Distribution Functions (CDF) for each artificial trace. Then, for each trace, we calculate the correlation coefficient (cc) [2] between the error and error-free CDFs of artificial trace and the CDFs of the original trace. 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 generated traces and computed their cc values for a given reference trace. First, we generated a reference trace with fixed set of $(L_{exp}, EF_{exp}, L_{den})$ values of (0.006, 0.1, 1.0). Next, we generated 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 we computed 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 plotted the mean error burst and its percentage reduction (relative to the reference trace's error burst size) for each observed cc value (see Figure 9). Thus, the percentage 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 9 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.

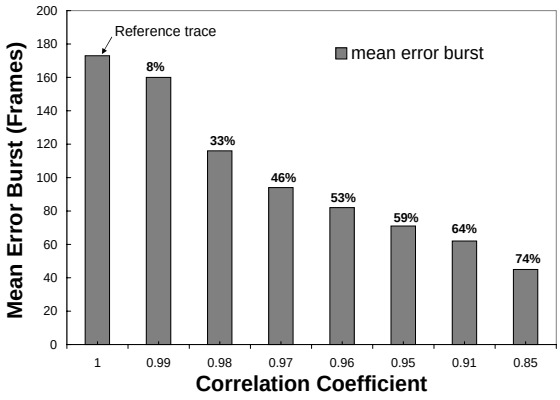


Figure 9. Mean error burst and percentage reduction for different correlation coefficient values.

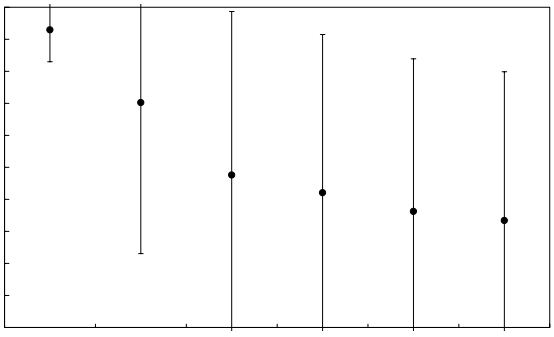


Figure 10. *WLAN_E* **error path modeling: mean and standard deviation correlation coefficient values for different subtrace lengths.**

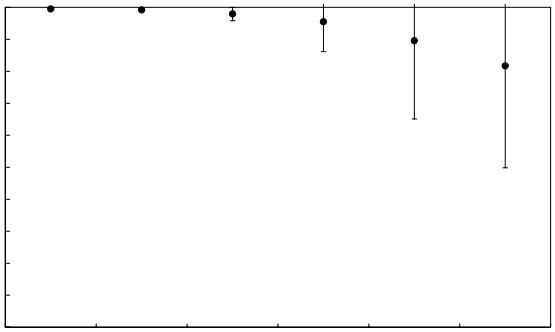


Figure 11. *GSM_E* **error path modeling: mean and standard deviation correlation coefficient values for different subtrace lengths.**

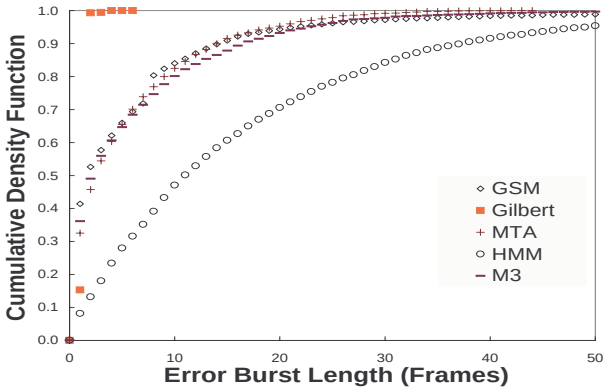


Figure 12. Error burst distribution for *GSM_E* **model.**

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, and 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 the reference traces *ref_WLAN_E* and *ref_GSM_E*, Figures 10 and 11 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 is an accurate representation of the network path, and metric of interest for a given scenario (*i.e.*, will the model accurately describe the characteristics of additional traces collected from the network path for the same scenario?). To verify that the answer is yes, we perform an experiment that determines the best model using a subsection of a reference trace, and then uses this model to create an artificial trace. We then compare artificial trace's statistics to those of a different reference trace subsection

and to the entire reference trace itself.

We extracted 200,000 frames from *GSM_E* trace, and called this reference trace *AB*. We divided *AB* into two subtraces of 100,000 frames each, and called these subtraces *A* and *B*. Next, we calculated the best model for subtrace *A* using the cc metric to determine model accuracy (see Section 6.1). The M^3 model yielded the highest cc value, therefore we chose this model to create a 100,000 frame artificial trace M^3_A.

To determine the accuracy of the statistics of artificial trace, M^3_A, we calculated the cc of the error burst and error-free burst CDFs (see Section 6.1) between M^3_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 M^3_A and *A* and between M^3_A and *B* are relatively close in value (especially for for error bursts), which indicates that the artificial trace generated by M^3 reasonably accurately models other regions of a 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 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 a network path, metric of interest, and scenario. Then, we 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 M^3 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 0.74, we plot the error burst CDF for the *GSM_E* trace models in Figure 12. 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 the networks, metric of interest, and scenario, and, as shown by the modeling of *IP_2* and *GSM_D*, we still need better models for capturing network path behaviors. The Gilbert model performs well when modeling wired IP networks, however, surprisingly, 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 M^3. Note that the M^3 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 M^3 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 M^3 is the highest cost model.

Thus, the choice of model may also depend on the type of simulation being done. If a trace can be generated in

Trace	Gilbert	HMM	MTA	M^3	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 M^3	**0.99** or **0.99**
IP_2	0.92, 0.81	0.19, 0.68	0.95, 0.62	**0.98**, 0.94	M^3	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	M^3	**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	M^3
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.

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 2) 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?*

8.1 Generating Artificial Traces

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 M^3 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 error; and thus, the bias switches towards higher order models being better choices. 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 [7]. Given a random variable X with a CDF $F(x)$, the variable u is

Trace	Original	Gilbert	HMM	MTA	M^3
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.

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.

We construct Domain Applicability Plots (DAP) for each L_{den} value, where each point in the DAP diagram indicates the best model for each (L_{exp}, EF_{exp}) pair. The best model is defined as the model with a corresponding maximum average cc value. Note that, on both the x and y axes, as the exponential distribution parameter increases, the state length decreases (see Figure 13).

8.2 Model Statistical Accuracy

In this section, we explore the statistical accuracy of analytical models for describing a network characteristics. In particular, we evaluate two well-known classical models and our two data preconditioning models by analyzing Domain Applicability Plot (DAP) diagrams.

Figures 14, 15, and 16 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 14), 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 M^3 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 M^3 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 M^3 is optimal (the "optimal-M^3" region), the mean cc value for the M^3 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. Therefore, we can conclude that, for this network, an L_{den} value of 0.2, using the M^3 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 15). 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.

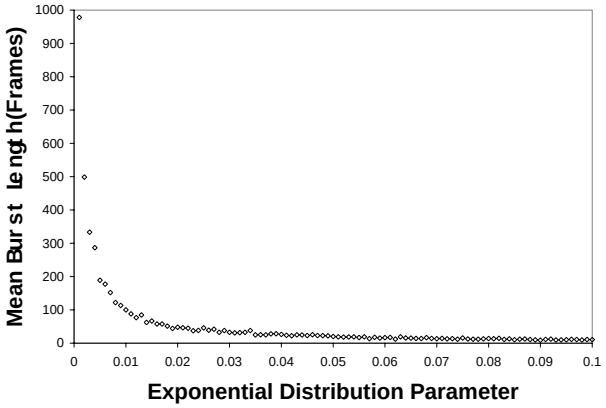


Figure 13. Mean burst length versus exponential distribution function parameter.

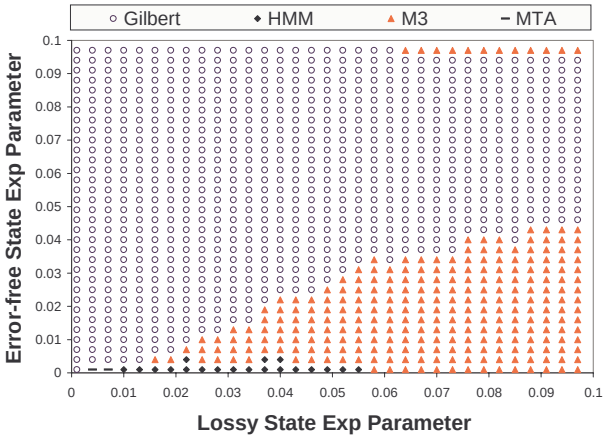


Figure 14. Optimal model for L_{den}=0.2.

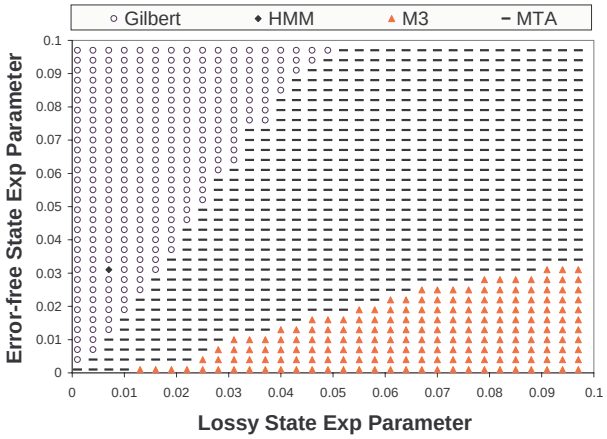


Figure 15. Optimal model for L_{den}=0.4.

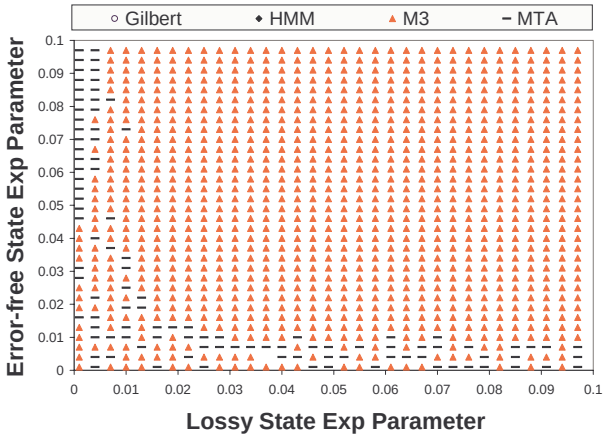


Figure 16. Optimal model for L_{den}=0.7.

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-M^3", 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-M^3" 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

As we have shown in this paper, accurate network path modeling can enable the creation of both models and artificial traces that are statistically indistinguishable from traces from real networks. We believe that such models can provide both predictive and descriptive power and can yield a better understanding of network's and their characteristics. These models can also be used in network path simulators and emulators to optimize both new and existing protocols. We have developed two data preconditioning approaches to network modeling that are better able to model some network paths and metrics of interest than classic models.

The main contribution of this paper is to aid network and application protocol developers in developing and choosing appropriate models for simulation of network conditions. As such, in this paper, we have proposed the characterization of network conditions using a triplet of values to ex-

	Optimal Model Region						
	L_{den} = 0.2		L_{den} = 0.4			L_{den} = 0.7	
Model	**Gilbert**	**M^3**	**Gilbert**	**MTA**	**M^3**	**MTA**	**M^3**
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**
M^3	**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.

press the lengths of error-free and lossy regions and the error rate in the lossy region. We also propose a simple methodology for evaluating model accuracy and choosing the best models for characterizing a network.

The primary conclusion from our analyses is that classic modeling techniques work well for some, but importantly, not all wired networks. However, when modeling delay and losses in wireless networks, the data preconditioning approaches are more accurate. Another important conclusion is that more work remains to be done in the search for accurate models, as our evaluation shows that all models have accuracy limitations depending on the characteristics of the network under measurement.

References

[1] Hari Balakrishnan and Randy Katz. Explicit loss notification and wireless web performance. In *Proceedings of the IEEE Globecom Internet Mini-Conference*, November 1998.

[2] J.S. Bendat and A.G. Piersol. *Random Data: Analysis and Measurement Procedures.* John Wiley and Sons, 1986.

[3] Blind. Citation removed for blind review.

[4] Blind. Citation removed for blind review.

[5] J. Bolot, S. Fosse-Parisis, and D.Towsley. Adaptive FEC-based error control for internet telephony. In *Proceedings of Infocom.* ACM, March 1999.

[6] S. Floyd and E. Kohler. Internet research needs better models. In *Hotnets-I*, October 2002.

[7] Raj Jain. *The Art of Computer Systems Performance Analysis.* John Wiley and Sons, 1991.

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

[9] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. In *IEEE/ACM Transaction on Networking*, Feb 1994.

[10] Iain L. MacDonald and Walter Zucchini. *Hidden Markov and Other Models for Discrete-valued Time Series.* Chapman & Hall/CRC, first edition, 1997.

[11] S Molnar and A. Gefferth. On the scaling and burst structure of data traffic. In *8th International Conference on Telecommunication Systems, Modelling and Analysis*, May 2000.

[12] Giao T. Nguyen, Randy Katz, and Brian Noble. A trace-based approach for modeling wireless channel behavior. In *Proceedings of the Winter Simulation Conference*, pages 597–604, December 1996.

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

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

[15] M. Yajnik, J. Kurose, and D. Towsley. Packet loss correlation in the MBone multicast network: Experimental measurements and Markov chain models. *UMASS COMPSCI Technical Report 95-115*, 1996.

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

[17] Michele Zorzi and Ramesh R. Rao. On the statistics of block errors in bursty channels. In *IEEE Transactions on Communications*, 1997.