Choosing an Accurate Network Model using Domain Analysis

Almudena Konrad
Mills College
akonrad@mills.edu

Ben Y. Zhao
U. C. Santa Barbara
ravenben@cs.ucsb.edu

Anthony D. Joseph
U. C. Berkeley
adj@cs.berkeley.edu

1. Introduction

Network link simulation is perhaps the most common method for evaluating application and network protocol designs. Simulation enables researchers to quickly and repeatably explore the behavior of a protocol under a variety of network conditions (*e.g.*, varying loss, delay, and error). However, accurate results are highly dependent on realistic network conditions being simulated. Previous work [4] 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 [7]. 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 modeling realistic networks, researchers face 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 from some 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 MTA [8], and introduce the Multiple states MTA (MMTA) model.

Given many traditional and new models, researchers face the challenge of choosing the most accurate model for their datasets. 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 we call *domain analysis* 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. We present our data preconditioning technique in Section 4. Next in Section 5, we present domain analysis, apply it to several real and synthetic traces, and introduce *Domain of Applicability Plots* (DAP) as a tool to rapidly visualize model accuracy. We conclude with Section 6.

2. Related Work

Researchers have applied traditional models to characterize the loss process of various network channels from network traces. Bolot *et al.* [3] use a two-state Markov model to characterize the loss process of audio packets to determine the appropriate error control scheme for streaming audio. Yajnik *et al.* [12] characterize packet loss in a multicast network

There is also related work in wireless traffic modeling. Nguyen *et al.* [10] present a two-state Markov wireless error model and develop an improved model based on collected Lucent 900MHz WaveLAN error traces. Balakrishnan and Katz [1] collected similar error traces 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.* [13] also investigate the error characteristics of a wireless channel and compare an Independent and Identically Distributed (IID) model to the Gilbert model. Our work [7] provided a more accurate modeling technique for datasets with regions exhibiting non-stationary behavior.

3. Defining and Classifying Network Traces

We define network traces as binary sequences, where a 1 denotes the occurrence of a specific event in the trace, and a 0 denotes its absence. In this paper, we analyze and model network traces that capture several types of events including 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. In a delay trace, it means that the

Trace	Frames	FER	L_{exp}, EF_{exp}, L_{den}
IP_1	360,385	0.027	0.034, 0.099, 0.82
WLAN_E	288,804	0.063	0.044, 0.099, 0.34
WLAN_D	188,436	0.293	0.046, 0.005, 0.414
GSM_E	616,404	0.055	0.005, 0.056, 0.41

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

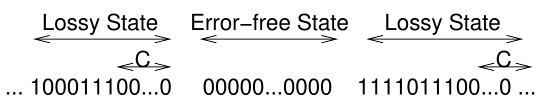

Figure 1. Error trace with lossy and error-free states.

packet or frame arrived with a delay greater than some maximum threshold [1]. We define the Frame Error Rate (FER) as the percentage of all frames (or packets) that have errors (or losses, or delays).

We create models for traces collected from various networks and at different protocol layers (see Table 1). *IP_1* is a loss trace collected by Yajnik [12] on an uncongested IP link from Massachusetts to Sweden. *WLAN_E* was collected under good signal quality conditions from an IEEE 802.11b wireless LAN testbed at T. U. Berlin by Willig [11]. We collected *GSM_E* under poor signal quality conditions at the Circuit-Switched Data (CSD) radio link layer of a GSM wireless cellular network at UC Berkeley. We also collected *WLAN_D* at the transport layer using UDP over a good signal quality 802.11b network at UC Berkeley. This trace was analyzes application delays introduced by a wireless network, with delay threshold threshold set to 20 ms.

We analyze the traces in Table 1 and observe that they 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, as shown in Figure 1. 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 determines the maximum number of consecutive 0's in a lossy state. The optimal value of C is calculated in Section 4, such that the regions of lossy states experience stationary behavior (*i.e.*, the statistics do not change over time for a given window size). See [7] for a formal discussion on stationarity of network traces.

1 The threshold value is application-specific and it indicates the delay value for which the packets will be dropped by the application.

In [7], we observed 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).

4. Modeling through Data Preconditioning

In this section, we begin by describing two classical stochastic models. We then present our *data preconditioning* methodology for accurately modeling networks exhibiting non-stationary behavior. Data preconditioning preprocesses trace data into stationary subsets, then models them and their relationships with other subsets.

4.1. Classical Models

We describe two classical stochastic models that we use as our basis for comparison. One is the popular Gilbert model, which is a Markov process of memory size one. The other is the Hidden Markov Model (HMM) [9].

The Gilbert model is a popular model for network simulation. It is a DTMC of order one and has two states. In a network trace, the Gilbert model states correspond to the status of each data frame $\{0,1\}$, as previously defined. It 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.

Popular opinion holds that a HMM model is appropriate to model the non-stationary characteristics of empirical 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.

To model our network traces, we choose a two-hidden-state 4^{th} order hidden Markov model. The states correspond to the lossy and error-free states defined in Section 3, while the observation symbols correspond to the status of the data frame $\{0, 1\}$. By using a high order of 4 in HMM to account for correlations between consecutive states, we are comparing our models against the most accurate results of a HMM at the cost of higher computational complexity.

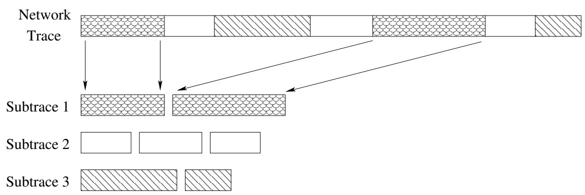

Figure 2. Data preconditioning: a trace decomposed into three subtraces, each is a concatenation of a specific data pattern.

4.2. Data Preconditioning and the MTA Model

Data preconditioning is a way to more accurately model networks with non-stationarity behavior, by analyzing and preconditioning data *before* it 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 model the resulting subsets. We illustrate the process in Figure 2. 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 [2]. Second, we concatenate trace regions with same states to form stationary subtraces of the original trace, which are then modeled using a high-order DTMC. Finally, we use Markov models (or similar techniques) to calculate the transition probabilities between states.

This methodology was first used in the Markov-based Trace Analysis (MTA) model [7]. As an application of data preconditioning, the MTA model identified only two states, the lossy and error-free states described in Section 3.

In decomposing a trace into a subtraces, an important design decision is the choice of the *change-of-state* variable C introduced in Section 3. To determine the optimal C, we generate and test a set of candidate values. For each possible value, we generate lossy subtraces and test their stationarity, finally choosing the largest C value that still generates stationary subtraces.

To test for stationarity in a trace, we used the Runs Test developed by Bendat and Piersol [2]. The Runs Test computes the median run (*i.e.*, error bursts) value of a lossy subtrace, divides this subtrace 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 subtrace is stationary, 90% of runs fall between the range of 0.05 and 0.95. We demonstrate the algorithm using the *GSM_E* trace (see Table 1). We use simple estimation to narrow down the value of C to a small range [7], in this case, between 21 and 25. We generate subtraces for each value in the range, and perform the Runs Test on them. The resulting runs distributions for 21, 22, 23, 24 and 25 are 91.4, 91.5, 90.5, 90.7 and 89.3. Therefore, we choose C to be 24.

Given C, we identify lossy and error-free states and construct the lossy and error-free subtraces. The MTA then models the lossy subtrace as a DTMC and computes the memory and transitions probabilities. The MTA models the length of both states by fitting the states lengths distributions to an exponential distribution function and computes the closest fit parameters. It plots the Cumulative Distribution Function (CDF) of the empirical trace with exponential distributions using parameter values ranging from 0 to 1 in steps of 0.001. MTA then chooses the exponential parameter that yields the CDF closest to the empirical CDF, by calculating the correlation coefficient (cc) [2] between the CDFs. A cc of 1 signifies a perfect match between two CDFs, while 0 indicates no statistical correlation.

4.3. The Multiple States MTA Model

We now introduce our new data preconditioning algorithm, the Multiple states MTA (MMTA). Unlike the MTA, the MMTA is capable of modeling traces with two or more states. The MMTA views each stationary region as a state, and models the transition among states with a higher order DTMC. Using data preconditioning, the MMTA algorithm 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.

We summarize the steps of the MMTA algorithm:

- Similar to the MTA, MMTA first identifies the states in the original trace and creates subtraces by concatenating similar states together. The MMTA models the lossy subtrace as a DTMC, and calculates its order and transition probabilities. It then models the error-free state as a deterministic process.
- MMTA determines the transitions between error-free and lossy states. It first creates a *state trace* corresponding to the collected dataset (*e.g.*, GSM_E trace), with lossy states replaced by all 1's and error-tree states remaining replaced by all 0's. It then models the state trace as a DTMC, and calculates its order and transition probabilities.

5. Domain Analysis

While data preconditioning models are designed for non-stationary traces such as those from wireless networks, they might not perform as well on traditional network traces, such as those from wired IP. With many available models (Bernoulli, Gilbert, Higher-order Markov, HMM, MTA, MMTA), protocol designers face the challenge of choosing the most accurate model for each trace.

2 Each network behavior has certain statistical properties.

Trace	Gilbert	HMM	MTA	MMTA
IP_1	**0.99**, **0.98**	**0.99**, 0.66	0.72, 0.95	**0.99**, **0.98**
WLAN_E	0.92, 0.74	0.73, 0.51	**0.99**, 0.87	**0.99**, 0.73
WLAN_D	0.93, 0.80	0.29, 0.37	**0.99**, 0.54	**0.98**, 0.95
GSM_E	0.74, 0.92	0.89, 0.92	**0.99**, 0.96	**0.99**, 0.94

Table 2. Artificial traces and their cc's (error burst CDF, error-free burst CDF).

We now describe *domain analysis*, a way to choose the most accurate model for a given trace by quantifying the accuracy of each model applied to it. To choose the best model for trace T, we apply each model to extract a set of defining model parameters, generate artificial traces based on those parameters, and compare the artificial traces against T. To measure their accuracy, we plot the error and error-free burst CDFs for each artificial trace. We then calculate the correlation coefficient (cc) (see Section 4) between the CDF of T and the CDF of the artificially trace. We use the cc as a measure of how closely the distribution of each artificial trace approximates the original trace distribution. We list the cc values for the error and error-free bursts CDF of the traces in Table 2. While we have verified the usefulness of cc as a measure of modeling accuracy, we refer the reader to [6] for more detailed analysis.

5.1. Application to Real Traces

To validate our domain analysis approach, we apply it to our real traces shown in Table 1 to determine the best model for each trace. For our experiment, we choose two classical models (Gilbert and 4^{th} order HMM) and two data preconditioning algorithms (MTA and MMTA). For each real trace, we generate a set of artificial traces, each generated from model parameters for a single model. We then quantify the correlation coefficient (cc) between the CDFs of the original traces and the CDFs of the artificial traces. We show the results in Table 2.

The results show that different models have varying degrees of success in capturing the statistical properties of different metrics for different networks. The Gilbert, the HMM, and the MMTA models perform well when modeling wired IP networks. Classical models accurately capture error bursts in wired networks, but are fairly inaccurate at modeling wireless networks. Data preconditioning models perform well at modeling many of the networks, especially the error burst portions.

5.2. Domain of Applicability Plots

In addition to applying domain analysis on real traces, we want to see how it can be used to choose models for a wide variety of 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 use domain analysis to determine the best model for each trace.

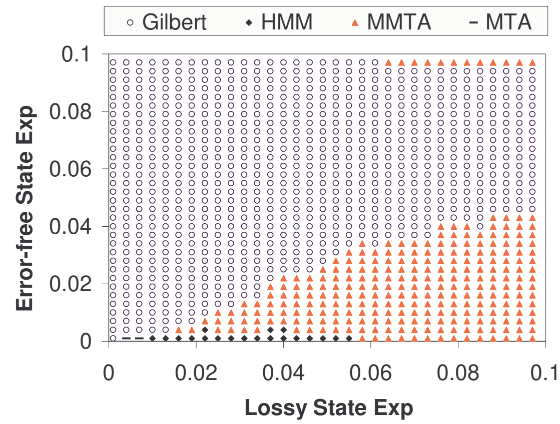

Figure 3. Optimal model for L_{den}=0.2.

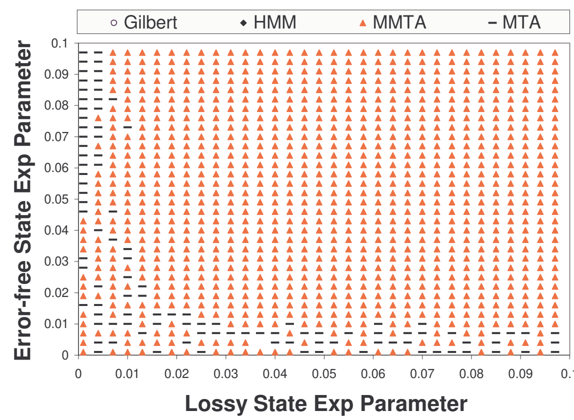

Figure 4. Optimal model for L_{den}=0.7.

We want to generate Domain Applicability Plots (DAP) that 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 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.

We use the fixed L_{den} values to generate Bernoulli process-based random errors inside the lossy state. This means that inside a lossy state the occurrence of errors are memoryless (*i.e.*, the next frame's value does not depend on the previous frame). Using a Bernoulli process to generate errors, for small values of L_{den}, biases the domain analysis results towards the simpler Gilbert model and against complex higher order models. However, as the value L_{den}

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

Table 3. Correlation coefficient for each L_{den} value (0.2,0.7) and each optimal region.

increases, so does the likelihood of occurrence of multiple consecutive errors. Since most real network traces experience some degree of memory, their domain analysis would yield results that were strongly biased towards memory process-based models. Thus we generate a wide range of traces to explore the full capacityof domain analysis.

We determine the lossy and error-free bursts lengths by using the inverse transformation method [5]. 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 $x = -ln(u)/\alpha$.

Figures 3 and 4 show the DAPs for L_{den} values of 0.2 and 0.7, respectively. Note that for $L_{den} = 0.2$, the Gilbert model is best for a large portion of the graph. The result is as expected, because of the use of a Bernoulli process to generate losses in the lossy state. The error burst length is relatively small and the Gilbert model is the best choice for many points. However, as the probability of error in the lossy state L_{den} increases, the error burst length increases and the MMTA and MTA become better choices.

Further examination shows that the mean cc value in this area for the Gilbert model is 0.99 and 0.98 for the MMTA model (see Table 3). Thus, even where the Gilbert yields the best results, the M^3 also performs very well. For the region where the MMTA is optimal, the mean cc value for the M^3 model is 0.97, while the mean cc for the Gilbert is 0.96. We have showed that cc values smaller than or equal to 0.96 yield inaccurate models [6]. Therefore, for this network with $L_{den} = 0.2$, 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 a high value of L_{den}, 0.7. For this value, almost the entire DAP diagram is dominated by MMTA, with a mean cc value in this region of 0.98. While the MTA's mean cc was a high 0.97, both the Gilbert and HMM perform very poorly. We believe that this result demonstrates the inability of traditional models to capture the long error bursts inside lossy states. In contrast, data preconditioning models can accurately capture both low and high error densities inside lossy states.

6. 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 it is better in capturing error burst statistics than classical models and more consistently accurate across different networks than our previous MTA model.

In addition, we propose a domain analysis methodology to evaluate the accuracy of models and to choose the best models for a given network. The primary conclusion from our analyses 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.

References

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

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

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

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

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

[6] A. Konrad and A. D. Joseph. Choosing an accurate network path model. Technical Report UCB/CSD-03-1236, U. C. Berkeley, May 2003.

[7] A. Konrad, B. Y. Zhao, A. Joseph, and R. Ludwig. A Markov-based channel model algorithm for wireless networks. In *Proc. of ACM MSWiM*, 2001.

[8] R. Ludwig, A. Konrad, and A. Joseph. Optimizing the end-to-end performance of reliable flows over wireless link. In *Proc. of ACM/IEEE MobiCom*, 1999.

[9] I. L. MacDonald and W. Zucchini. *Hidden Markov and Other Models for Discrete-valued Time Series*. 1997.

[10] G. T. Nguyen, R. Katz, and B. Noble. A trace-based approach for modeling wireless channel behavior. In *Proc. of the Winter Simulation Conference*, Dec. 1996.

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

[12] 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.

[13] M. Zorzi and R. R. Rao. On the statistics of block errors in bursty channels. *IEEE Trans. on Comm.*, 1997.