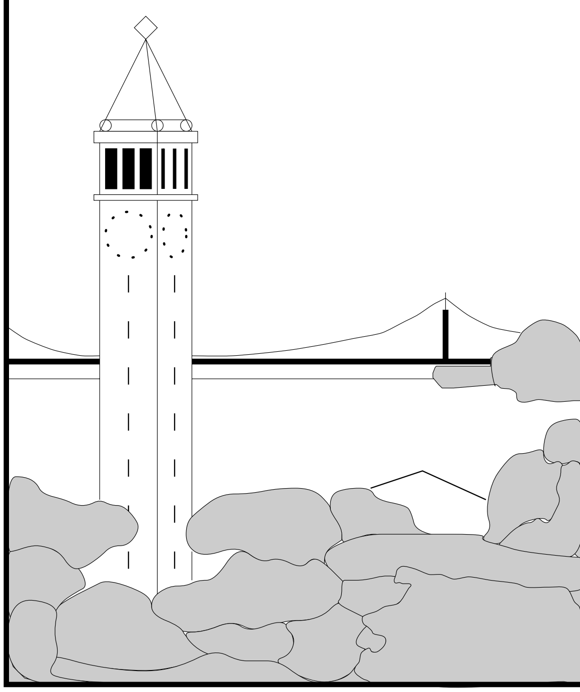
A Markov-Based Channel Model Algorithm for Wireless Networks

Almudena Konrad, Anthony D. Joseph, Reiner Ludwig, Ben Y. Zhao



Report No. UCB/CSD-01-1142

May 2001

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

A Markov-Based Channel Model Algorithm for Wireless Networks

Almudena Konrad, Anthony D. Joseph, Reiner Ludwig, Ben Y. Zhao *

Abstract

Techniques for modeling and simulating channel conditions play an essential role in understanding network protocol and application behavior. We demonstrate that time-varying effects on wireless channels result in non-stationary wireless traces. We present an algorithm that divides traces into stationary components in order to provide analytical channel models that, relative to traditional approaches, more accurately represent characteristics, such as burstiness, statistical distribution of errors, and packet loss processes. In [11], we demonstrated that inaccurate modeling using a traditional analytical model yielded significant errors in error control protocol parameters choices.

Our algorithm also generates artificial traces with the same statistical characteristics as actual collected network traces. Using these traces in a simulator environment enables future protocol and application testing under different controlled and repeatable conditions.

For validation, we develop a channel model for the circuit-switched data service in GSM and show that it: (1) more closely approximates GSM channel characteristics than a traditional Gilbert model and (2) generates artificial traces that closely match collected traces' statistics.

1 Introduction

As communication networks evolve, the design of communication protocols increases in complexity. Evaluating the performance of existing networks provides insights into techniques for optimizing future communication protocols. The most common techniques include simulation, analysis of empirical data, and analytical models (e.g., channel models). Accurate modeling of network events, especially the error behavior, at link layer and above is essential to the understanding of network behavior and to the design of communication protocols. For example, a detailed understanding of the packet loss process and burstiness of the errors is necessary for the proper design and parameter tuning of error control protocols, such as Automatic Repeat reQuest (ARQ) protocols.

Streaming audio and video multimedia applications can also benefit from a better understanding of the underlying network behavior. For example, video and audio codecs can perform real-time predictive rate control by using a model of network traffic characteristics to estimate traffic conditions in real-time.

The traditional network modeling approach to error modeling is to create a Gilbert model [16] (i.e., a two state discrete Markov chain) based upon collected network traffic traces. Using this model, one can then dynamically generate artificial network traces for the network under study and use the traces to simulate, and thus, better understand the performance of existing and new network protocols and applications. These traces provide network protocol and application developers with ease of use and repeatability, two critical characteristics for developing and improving network and application performance. More importantly, for new networks that are under development (or for which there are only limited prototype facilities), it is often difficult to collect a reasonable amount of traces or run experiments. By generating synthetic traces that simulate the network being tested, multiple users can simultaneously gain network access and perform experiments.

Unfortunately, as we will show, the Gilbert model has several significant shortcomings in the accuracy of its error modeling, which directly affects the validity of results based upon traces generated by a Gilbert model. Models based upon Markov processes require that the error statistics remain constant over time. Many networks experience time varying effects, such as congestion-related losses. Wireless channels, in particular, experience effects, such as Raleigh fading, multipath fading, shadowing, etc.

To confirm that wireless traces are by their nature non-stationary, we utilize a previously published, but not widely known algorithm for testing stationarity [2]. Using this algorithm, we tested a sample wireless trace and confirmed its non-stationarity. This implies that traditional stochastic analysis of wireless traces are likely to be less accurate than ideal.

Thus, we propose and evaluate a novel algorithm, the

* A. Konrad, A. D. Joseph and B. Y. Zhao are with the CS Division, U.C. Berkeley, Berkeley, California. Emails: {almudena,adj,ravenben}@cs.berkeley.edu. R. Ludwig is with Ericsson Research, Herzogenrath, Germany. Email: Reiner.Ludwig@eed.ericsson.se

Markov-based Trace Analysis (MTA) algorithm, for the design of channel error models. Our approach is to derive a statistical constant from the wireless trace, and use this constant to divide the previously non-stationary trace into stationary subtraces representing good and bad segments of transmission. By analyzing the length distributions of these segments, we can effectively characterize the transitions between them, and create a model that more accurately represents the original trace.

In practice, this MTA algorithm allows a more accurate analysis of network traces which accounts for their non-stationary behavior. This characteristic makes MTA a general purpose algorithm, meaning that it can be applied to network traces such as wireless traces which experience different error statistics over time. However, the purpose of this work is not to show that the MTA algorithm is general purpose, but to argue that the MTA algorithm generates accurate analytical models for wireless channels.

We validate the benefits and accuracy of the MTA algorithm by applying it to 215 minutes of GSM digital wireless cellular network [14] data traces collected at the reliable link layer (Radio Link Protocol layer [5, 7]) to generate a model we call the *MTA GSM channel model*. We then show that, unlike traces generated by the Gilbert model, artificial MTA model network traces have the same statistical properties as traces collected from the actual network. Such traces will provide more accurate simulations of the network being tested, yielding results that more closely match the results obtained on actual networks.

In particular, we generate artificial traces using both the MTA and Gilbert models, and perform *retrace analysis* [11] on these artificial traces. *Retrace analysis* emulates an enhanced RLP layer using a fixed data frame size and fixed per frame overhead (e.g., checksums, sequence numbers, etc.), and calculates the predicted throughput over a range of fixed RLP frames sizes. In our enhanced RLP implementation, frame sizes are multiples of the physical radio block size of 30 bytes[1]. For a given frame size, there is a trade-off between the increased throughput from reducing overhead and the retransmission delay caused when a radio block of an RLP frame is lost and the entire frame is retransmitted. In other words, the greater the frame size, (1) the lower the overhead, and (2) the greater the retransmission delay (the more radio blocks that have to be retransmitted) when a radio block is corrupted. Thus, throughput performance results for each frame size are highly correlated with a collected or synthetic trace's error statistics. In [11], we used *retrace analysis* to show that for bursty error traces (where errors tend to occur in clusters), larger frames yield higher throughput. Furthermore, we showed that incorrectly assuming an even distribution of errors in GSM, leads to the wrong choice of optimal frame size.

These results show that the distribution of errors within traces has a significant influence on models, analysis, and simulations based upon such traces. This conclusion is especially true when the goal is to artificially generate traces for the design, simulation, and analysis of new networking protocols. To replicate and further explore the results from our earlier work, we generate an artificial trace that we call *even trace*, which has the same error rate as collected traces, but has an even error distribution, (i.e., errors are individual events, isolated, and have a constant distance between each other).

The rest of this paper is organized as follows: We start by discussing related work in the next section. Section 3 provides background information about the GSM's Circuit-Switched Data (CSD) service and an overview on Discrete Time Markov Chains. Next, in Section 4, we describe our measurement platform for collecting block level error traces on the GSM wireless link. Then Section 5 shows the development of the MTA algorithm, followed by Section 6, where we develop two analytical models for GSM wireless traffic: the MTA model and the Gilbert model. Finally, in Section 7, we present our algorithm for generating artificial traces and evaluate the MTA algorithm by comparing the traffic statistics of the collected and artificial traces. We conclude and discuss our plans for future work in Section 8.

2 Related Work

Several researchers have explored ways of characterizing the loss process of various channels. Bolot *et al.* [3] use a characterization of the loss process of audio packets to determine an 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.* [19] 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 as a third order Markov chain. Both these efforts analyze the loss process of traces with static error statistics (i.e., the error rates do not vary over time). However, our work addresses the greater challenge of modeling traces with time-varying error statistics.

There is also interesting related work in wireless traffic modeling. Nguyen *et al.* [15] use a trace-based approach for modeling wireless errors. They present a two-state Markov wireless error model, and develop an improved model based on collected WaveLAN error traces. Building on this, Balakrishnan and Katz [1] also collected error traces from a WaveLAN network and developed a two-state Markov chain error model (i.e., Gilbert model). Zorzi *et al.* [20] also investigates the error characteristics in a wireless channel.

[1] Note that the existing GSM RLP implementation uses a frame size of one radio block.

They compare an independent and identically distributed (IID) model to the Gilbert model, and claim that higher order models are not necessary. Their results are drawn by applying these models to artificial traces generated by assigning a fixed-average burst length and a constant bit error rate.

While these previous works confirm that the Gilbert model improves upon the simple IID model, we offer proof in this paper that it has several significant shortcomings in its error modeling accuracy. Furthermore, we argue that there is a need to develop a more accurate model based on real world statistics that better describes and handles time-varying wireless channel error characteristics. Previous work such as that done by Yajnik *et al.* modeled loss processes using higher-order Markov chains for improved accuracy, but was limited to stationary traces. We show that traces on wireless links are non-stationary, and provide an algorithm that successfully models such behaviour.

3 Background

In this section we present a brief background on the technology behind circuit-switched data in GSM networks. We also define Discrete Time Markov Chains (DTMC) and some of their relevant properties.

3.1 Circuit-Switched Data in GSM

The Global System for Mobility (GSM) wireless digital cellular network is a second generation cellular network, providing nearly 700 million subscribers with global roaming capabilities in several hundred countries. GSM implements several error control techniques, including adaptive power control, frequency hopping, Forward Error Correction (FEC), and interleaving. The primary uses of the GSM network are for Circuit-Switched Voice service (CSV) and Short Message Service (SMS). However, an increasing number of subscribers are using GSM's Circuit-Switched Data service (CSD), which provides an optional reliable link layer protocol, the Radio Link Protocol (RLP). We provide a brief summary below; more details about GSM, the CSD service, and RLP can be found in [14].

GSM is a TDMA-based (Time Division Multiple Access) circuit-switched network. At call-setup time, a mobile terminal is assigned a user data channel, defined as the tuple $<$carrier frequency number, slot number$>$. The slot cycle time is 5 milliseconds on average. This timing allows 114 bits to be transmitted in each slot, yielding a gross data rate of 22.8 Kbit/s. The fundamental transmission unit in GSM is a *radio data block*. A Forward Error Correction (FEC) radio data block is 456 bits, representing the payload of 4 time slots. In GSM-CSD, the size of an unencoded data block is 240 bits, resulting in a raw data rate of 12 Kbit/s (240 bits every 20 milliseconds) [6].

Interleaving is a technique that is used in combination with FEC to combat burst bit errors. Instead of transmitting a data block in four consecutive slots, the block is divided into smaller fragments. Fragments from different data blocks are then interleaved before transmission. The interleaving scheme chosen for GSM-CSD interleaves a single data block over 22 TDMA slots [8]. A few of these smaller fragments can be completely corrupted while the corresponding data block can still be reconstructed by the FEC decoder. The primary disadvantage of this deep interleaving is that it introduces a significant one-way latency of approximately 90 milliseconds[2]. This high latency can have a significant adverse effect on interactive protocols [12].

RLP [5, 7] is a full-duplex logical link layer protocol that uses selective reject and checkpointing for error recovery. The RLP frame size is fixed at 240 bits aligned to the above mentioned radio data block. RLP introduces an overhead of 48 bits per RLP frame, yielding a user data rate of 9.6 Kbit/s in the ideal case (no retransmissions)[3]. RLP transports user data as a transparent byte stream (i.e., RLP does not "know" about IP packets). However, RLP may lose data if a link reset occurs (e.g., after a maximum number of retransmissions of a single frame has been reached).

3.2 Discrete Time Markov Chains

A Discrete Time Markov Chain (DTMC) [16] is a random process $\{X_n \mid n \geq 0\}$ that takes values in a discrete space E. A DTMC is defined by its memory and its transition probabilities and is characterized as follows,

$$\Pr(X_{n+1} = j \mid X_0 = i_0, X_1 = i_1, ..., X_n = i_n) = \Pr(X_{n+1} = j \mid X_{n-z+1}, 1 \leq z \leq K), \quad (1)$$

where $\Pr(X_{n+1} = j \mid X_{n-z+1}, 1 \leq z \leq K)$ are the K_{step} transition probabilities, and K defines the memory.

To calculate the memory of a DTMC, we find the *order* of the Markov chain as first proposed in [13]. To aid in determining the *order* of the Markov chain, we introduce the concept of *conditional entropy*. The *conditional entropy* is an indication of the randomness of the next element of a trace, given the past history. We determine the amount of past history necessary by calculating the i^{th} *order entropy* for $1 \leq i \leq M$, where M is an upper bound on the maximum amount of history we want to record. We choose M to be 6 because maintaining history for 2^6 or 64 states yields

[2]Note that voice is treated differently in GSM. Unencoded voice data blocks have a size of 260 bits and the interleaving depth is 8 slots.

[3]Note that the transparent (without RLP) GSM-CSD service introduces a wasteful overhead for modem control information, reducing the user data rate to 9.6 Kbit/s.

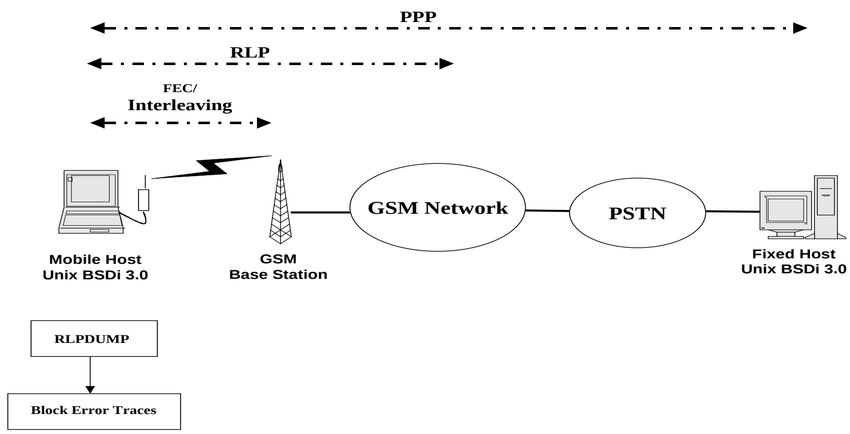


Figure 1. The GSM network and measurement platform.

a reasonable level of implementation and processing complexity. An i^{th} *order entropy* of 0 indicates that knowing the last i elements of the chain totally predicts the next element on the chain. As the *entropy* value increases, there is more randomness in the next element on the chain. We follow the same procedure used by Yajnik *et al* [19] to calculate the conditional entropy for each value of i:

$$H(i) = -\sum_{\vec{x}} \frac{\xi(\vec{x})}{T_{samples}} \sum_{y \in \{0,1\}} \frac{\xi(y,\vec{x})}{\xi(\vec{x})} \log_2 \frac{\xi(y,\vec{x})}{\xi(\vec{x})} \quad (2)$$

In Equation 2, $\vec{x}$ represents the vector $[x_1...x_i]$ which corresponds to one of the 2^i different patterns of i consecutive elements in the chain; $T_{samples}$ represents the total number of samples of length i in the chain; $\xi(\vec{x})$ indicates the number of times the pattern $\vec{x} = [x_1...x_i]$ shows up in the chain; and the term $\xi(y,\vec{x})$ corresponds to the number of times the pattern $\vec{x} = [x_1...x_i]$ appears in the chain followed by y, where $y \in \{0,1\}$.

Given the implicit tradeoff between entropy and complexity of the Markov model, we choose the *order* of the Markov chain K, such that we gain the minimum entropy possible at an acceptable complexity level. As entropy decreases, the order K increases, meaning the number of states (i.e., 2^k) increases exponentialy.

4 Data Collection

In this section, we first introduce the concept of *block error traces*. Then we describe the measurement platform we developed to collect block error traces.

4.1 Block Error Traces

An accurate representation of a wireless channel's error characteristics for a given time period can be captured by a bit error trace. A bit error trace contains information about whether a particular bit was transmitted correctly (i.e., a "1" represents a corrupted bit, while a "0" represents a correctly transmitted bit). The average Bit Error Rate (BER) is the first-order metric commonly used to describe such a trace. The same approach can be applied on a block level instead of on a bit level. A block error trace consists of a binary sequence where each element represents the transmission state of a data block. There are two block states, a "1" represents a corrupted data block, while a "0" represents a correct data block. Corrupted blocks are detected using an error detection code (e.g., Cyclic Redundancy Check). In this paper, we refer to block error traces simply as traces. We also use the *BLock Error Rate* (BLER) of a trace to define the average rate of corrupted data blocks.

For a trace, we define an *error burst* to be a run of consecutive 1's, and an *error-free burst* as a run of consecutive 0's. A trace is *stationary* whenever the error statistics remain relatively constant over time. We identify trace sections that exhibit stationary properties by finding error-free bursts of length equal to or greater than the *change-of-state* constant C. The value of C is a design decision that we define as the mean plus one standard deviation of the length of error bursts of a trace. By removing trace sections consisting of error-free bursts of length equal to or greater than C, we guarantee that the resulting trace will have stationarity or constant error statistic properties. We explain the reasoning behind our choice in more detail in Section 6.1.

We have collected traces under several different scenarios. As shown in Figure 1, we vary the movement of the mobile host (fixed, walking, and driving) while keeping the

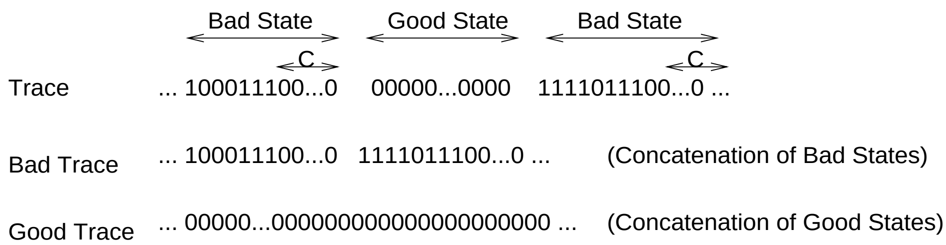


Figure 2. The separation of an error trace into two stationary traces.

other endpoint fixed. We collected 215 minutes of traces in a fixed environment, where the mobile host's signal strength was below 4 on a scale of 1 to 5. In the following sections, we refer to this trace as the *real trace.* In Section 6, we use the *real trace* to develop an analytical traffic model for RLP. Note that the error characteristics we have measured in these traces are only valid for the particular FEC and interleaving scheme implemented in GSM's Circuit Switched Data network (see Section 3.1). To analyze other types of networks, the first step is to collect block or packet level traces and then to apply the analysis described below.

4.2 Measurement Platform

We depict our measurement platform in Figure 1. A single-hop network running the Point-to-Point Protocol (PPP) [17] connects the mobile host to a fixed host that terminates the circuit-switched GSM connection. We used the *sock* tool [18] to generate traffic on the link. To collect traffic traces at the RLP layer, we ported the RLP protocol implementation of a commercial available GSM data PC-Card to BSDi3.0 UNIX. In addition, we developed RLPDUMP, a protocol monitor tool for RLP. RLPDUMP logs whether or not a received block could be correctly recovered by the FEC decoder. This determination is possible because every RLP frame corresponds to an FEC encoded data block (see Section 3.1). Thus, a received block suffers an error whenever the corresponding RLP frame has a frame checksum error. We used sock to generate bulk data traffic and used RLPDUMP to capture block error traces.

5 The MTA Algorithm

The basic concept behind the MTA algorithm is the assumption that a trace with non-stationary properties can be decomposed into a set of piecewise stationary traces consisting of what we define as "good" and "bad" states. The MTA algorithm defines these states, and parameterizes transitions between them as a function of a preset parameter, the *change-of-state* constant.

Good states contain only correctly transmitted blocks, while bad states exhibit stationarity, and a sequence of bad states can be modeled by a traditional DTMC. The MTA algorithm computes the distribution of lengths for both good and bad states, along with the memory and parameters for the DTMC used on the sequence of bad states.

In this section, we first discuss stationarity properties and how to test a trace for stationarity. We then present the MTA algorithm and show how it is applied to a trace.

5.1 Stationarity

We consider a network traffic trace to be a random process $\{X_n \mid n \geq 0\}$ with a discrete space $E = \{0, 1\}$ where a 1 denotes a corrupted frame, and a 0 denotes a correct transmitted frame. If $X_n = i$, then the process is said to have value i at time n. A process X_n that takes values on the discrete space $E = \{0, 1\}$ can also be viewed as a binary time series [4]. One major challenge in the analysis of time series is the concept of *stationarity*. 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 autocovariance only depends on the difference k for all n ($Cov(k, n) = Cov(X_n, Xn - k) = Cov(k)$). Given a binary time series X_n that is second-order stationary, the process can be modeled as a DTMC where the value of the chain at time n is determined by the memory of the process [10].

Checking a trace for stationarity is mathematically challenging. We use the test for stationarity introduced by Bendat and Piersol called the Runs Test [2], 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. The segment length can be arbitrarily chosen, as long as all the segments are of equal length.

3. Compute the lengths of runs in each segment.

4. Count the number of runs of length above and below the median value for run lengths in the trace.

5. Plot a histogram for the number of runs.

For a stationary trace, the number of runs distribution between the 0.05 and 0.95 cut-offs will be close to 90 percent [2].

5.2 Algorithm

The MTA algorithm views a trace as a process with two types of states: *bad* and *good*. The algorithm divides the trace into a *bad trace* consisting of a concatenation of *bad states*, and a *good trace* consisting of a concatenation of *good states* (see Figure 2). A *bad state* is defined as a sequence of zeros and ones (always started by a one), where each run of zeros is not greater than the *change-of-state* constant C (defined in Section 4.1 as the mean plus one standard deviation of the length of error bursts). If a run of zeros is equal to or greater than C, then the trace enters the *good state*.

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

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

- The *good burst length process* $\{G_n \mid n \geq 0\}$, where G_n represents the length of the n^{th} good state.

The distributions of B_n and G_n are found by plotting the cumulative density function (CDF) and finding the "best" fitting distributions. We provide an example of how to determine these distributions in Section 7.1.

The *good trace* is a deterministic process, where each value is zero. The *bad trace* is an stationary random process, therefore it can be modeled as a DTMC with a certain memory. The MTA algorithm calculates the memory of the *bad trace*, and determines its transition probabilities.

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* constant, equal to (m_e + sd_e).

3. Partition the trace into *bad state* and *good state* portions using the following definitions:

    - Bad state: runs of 1's and 0's, with the first element being a 1, and with runs of 0's that have length less than or equal to the C.

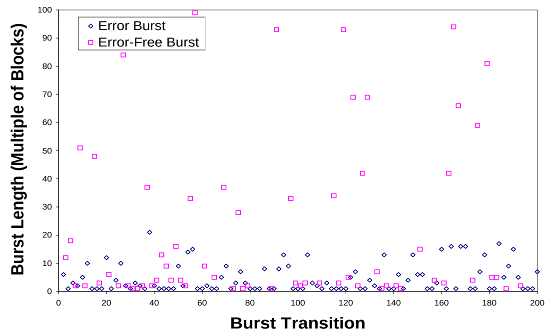


Figure 3. Burst length in *real trace*.

    - Good state: runs of 0's that have length greater than C.

4. Create *bad trace* and *good trace* stationary traces from the bad and good state portions of the trace.

    - Bad trace: concatenate the bad state portions of the trace.

    - Good trace: concatenate the good state portions of the trace.

5. Model *bad trace* as a DTMC, and calculate its order and transition probabilities.

6. Determine the best fitting distributions of the bad and good burst length processes B_n and G_n.

In summary, to take advantage of the Markov Process properties in non-stationary traces, we have used a novel approach to traffic modeling: a *Markov-based Trace Analysis* (MTA) algorithm that divides a trace into subset traces that have stationary properties.

6 Modeling GSM Wireless Channel

In this section, we demonstrate the process of extracting characteric statistics from a given trace using both the MTA and Gilbert models [16]. We apply both algorithms to *real trace* to generate the statistics which we will later use to generate artificial traces based on each model.

6.1 MTA GSM Channel Model

This section presents an application of the steps of the MTA algorithm (as described in section 5) to *real trace*. First, the MTA algorithm analyzes the *error free* and *error* burstiness experienced by *real trace* (see Figure 3), and calculates the mean and standard deviation for the error burst

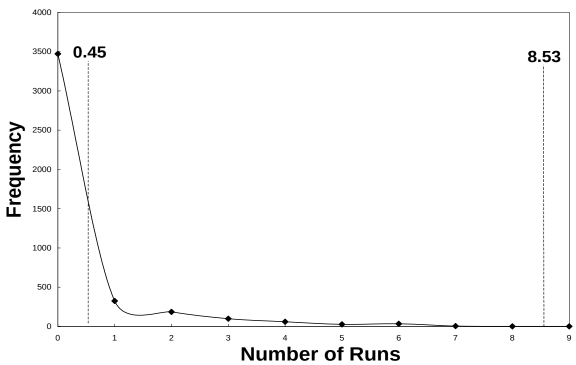


Figure 4. The Runs Test applied to *real trace***.**

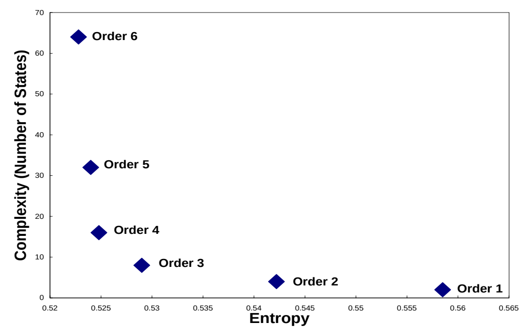


Figure 6. Complexity versus Entropy in *bad trace***.**

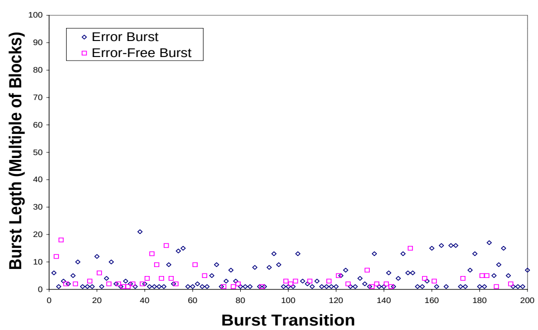


Figure 5. Burst length in *bad trace***.**

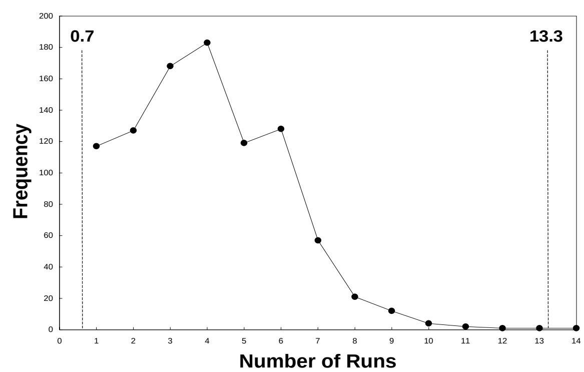


Figure 7. The Runs Test applied to *bad trace***.**

length. In this case, the mean value was found to be 6 blocks and the standard deviation was 14 blocks, yielding a *state-of-change* constant value C of 20 $(6 + 14)$ blocks. Figure 3 shows that the length of error-free bursts are significantly longer than the length of error bursts. These long error free bursts make the trace non-stationary. In particular, examining the trace using a window size value that is smaller than the maximum error free burst makes the channel appear error free. More importantly, as the window size decreases towards the size of error burst clusters, the channel exhibits significantly different error characteristics.

We apply the Runs Test (as described in Section 5) to test *real trace* for stationarity. Figure 4 shows that only 17 percent of the runs distribution lie between the 0.05 and 0.95 cut-offs, and 83 percent lays outside the left and right cut-offs. Thus, from the Runs Test, we conclude that *real trace* is a non-stationary process.

Since our goal is to isolate and analyze sections that experience stationarity, we use the MTA algorithm to create two new traces, called *bad trace* and *good trace*, each consisting of stationary trace sections. The MTA algorithm creates *bad trace* and *good trace* (as described in Section 5) by first identifying *good* and *bad* states and then concatenating *good* states to form *good trace* and *bad* states to form *bad trace*. Figure 5 shows the error-free bursts and error burstiness experienced by *bad trace*. In this plot, the average error free burst is 3.26 blocks, with a maximum value of 20 blocks (recall that the *change-of-state* constant C was defined to be 20). The error free burst mean and maximum values in *bad trace* are much smaller than the error burst mean and maximum value in *real trace*. Thus, our choice of C guarantees that *bad trace* will experience constant error statistic properties and therefore stationarity. To prove that *bad trace* is an stationary process we apply the Runs Test. Figure 7 shows that 87 percent of the runs distribution lie between the 0.05 and 0.95 cut-offs. Therefore, this result proves that *bad trace* is a stationary process and can thus be modeled as a DTMC.

Next, the MTA algorithm models *bad trace* as a DTMC with memory K. To determine the memory K of the DTMC, the MTA algorithm first calculates the conditional entropy values. Table 1 shows the conditional entropy calculated for different K values. Figure 6 illustrates how the complexity of the DTMC measured in number of states increases exponentially as entropy decreases. For this trace we chose K to be 4 (i.e., 16 number of states), which corresponds to only 0.38 percent increase in entropy from the chosen upper bound of $K = 6$. We could have chosen K to be larger than 4, but we did not want to significantly increase the complexity of the Markov model.

Table 2 shows the probabilities of the trace being in each state and the associated transition probabilities. The transition probabilities were also calculated by frequency counting.

Order K	*Entropy*
6	0.5228
5	0.5240
4	0.5248
3	0.5290
2	0.5422
1	0.5585

Table 1. Entropy for the *bad trace***.**

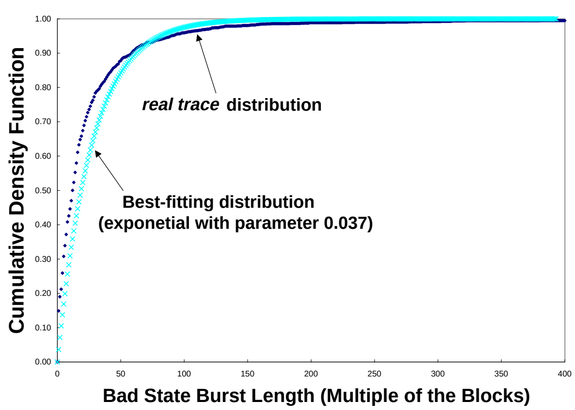


Figure 8. Bad state burst length distribution.

The last step of the MTA algorithm is to determine the best fitting distribution for the bad burst length process B_n and good burst length process G_n. Figures 8 and 9 show the CDF for the processes B_n and G_n. Each figure shows two plots, one plot is the CDF as calculated from the empirical data, (i.e, the distribution of *real trace*), and the other plot corresponds to the CDF of an exponential distribution with parameter α. We assume that the distributions of B_n

State i	$\Pr(i)$	$\Pr(1 \mid i)$	$\Pr(0 \mid i)$
0000	0.1254	0.1699	0.8301
0001	0.0305	0.6414	0.3586
0010	0.0172	0.1832	0.8168
0011	0.0344	0.8009	0.1991
0100	0.0166	0.3073	0.6927
0101	0.0033	0.8129	0.1871
0110	0.0087	0.2683	0.7317
0111	0.0415	0.8889	0.1111
1000	0.0305	0.3022	0.6978
1001	0.0210	0.7037	0.2963
1010	0.0027	0.0547	0.9453
1011	0.0159	0.8820	0.1180
1100	0.0350	0.4556	0.5444
1101	0.0153	0.8623	0.1377
1110	0.0415	0.3118	0.6882
1111	0.5604	0.9341	0.0659

Table 2. Fourth order Markov model statistics.

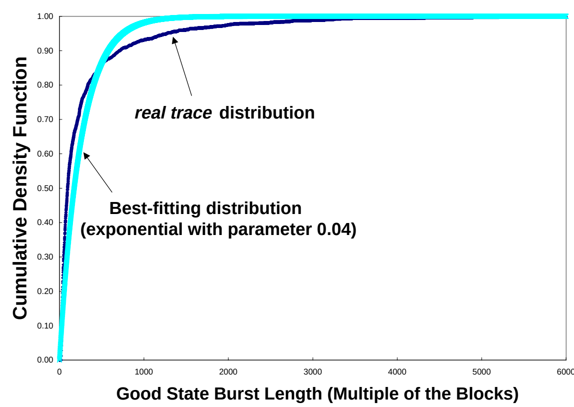


Figure 9. Good state burst length distribution.

and G_n are exponential with parameter α, (i.e. the CDF $F(x) = 1 - e^{-\alpha x}$, where x is the good or bad state burst length). For each distribution, B_n and G_n, the MTA algorithm plots the CDF of the exponential distribution with α ranging from 0 to 1 in steps of 0.001, and then chooses a value of α that provides the best approximation to the empirical data's CDF, (i.e., the distribution for *real trace*). We denote $\vec{x}$ as the vector with the CDF values based on the empirical data, and $\vec{y}$ as the vector with the CDF values based on the predicted exponential distribution. We use the *standard error* as a measure of the error between plots, and choose the distribution with smallest standard error. The equation for the standard error of the predicted $\vec{y}$ is

$$S_{error}(\vec{y},\vec{x}) = \sqrt{[\frac{1}{n(n-2)}][f(y) - \frac{[n\sum xy - \sum x \sum y]^2}{f(x)}]} \quad (3)$$

where $f(a) = n\sum a^2 - (\sum a)^2$, and n is the dimension of the vectors $\vec{y}$ and $\vec{x}$.

The predicted distributions for the bad and good bursts are exponential distributions with parameters $\alpha_b = 0.037$ and $\alpha_g = 0.04$, respectively. The standard error values for the predicted distributions of B_n and G_n are 0.013 and 0.025 respectively. Note that a lower standard error value indicates a more accurate prediction.

6.2 The Gilbert GSM Model

To study the performance and accuracy of the MTA algorithm, we compared the MTA model to the traditional Gilbert model. The Gilbert model is a DTMC of *order* one (i.e., with two states). In our traces, the Gilbert model states correspond to the states of the data block $\{0, 1\}$, where a 1 denotes a corrupted frame and a 0 denotes a correct frame. The Gilbert model predicts the state of the next block by just looking at the previous received block. Figure 10 shows the Gilbert model state transition diagram. Finally, Table 3 shows the results of the Gilbert model transition probability calculations for *real trace*.

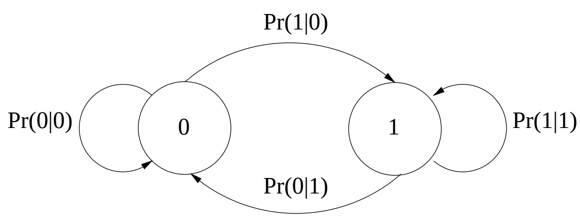


Figure 10. Gilbert model state transition diagram.

State i	$\Pr(i)$	$\Pr(1 \mid i)$	$\Pr(0 \mid i)$
0	0.9449	0.0087	0.9913
1	0.0551	0.8509	0.1491

Table 3. Gilbert model statistics.

7 Trace Generation and Evaluation

A key capability of the MTA algorithm is the ability to generate artificial traces (of any duration) with the same statistical characteristics as traces collected from any given network. In this section, we demonstrate how to generate an artificial trace given characteristic statistics from the MTA model. We also generate an artificial trace based on the Gilbert model, and compare both artificial traces against the *real trace*. We show that with respect to key characteristics such as error burst length distribution and throughput vs frame size, the MTA artificial trace provides a much improved approximation of the original *real trace*.

7.1 MTA Artificial Trace Generation

The algorithm for trace generation from an MTA model is 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 good state burst length from the good state burst distribution G_n.

   (b) Determine b_{len}, the bad state burst length from the bad state burst distribution B_n.

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

   (d) Generate b_{len} frames that are either error or error free frames depending on the transition probabilities calculated for the bad state trace in the MTA model.

Recall that in the MTA model, we observed that the good and bad state distributions, G_n and B_n, fit exponential distributions. Thus, to calculate g_{len} and b_{len} we can use the inverse transformation method from [9]. 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 $x = F^{-1}(u)$. In the exponential case with parameter α, $F(x) = 1 - e^{-\alpha x} = u$, x can be determined from $x = -ln(u)/\alpha$. In this case, x corresponds to g_{len} and b_{len}, respectively.

It should be clear by inspection that an artificial trace created by the above algorithm is guaranteed to have the same characteristics as those extracted by the MTA algorithm.

7.2 Trace Comparison

Here we evaluate the MTA algorithm by comparing the error statistics of the *real trace* against the two artificial traces. Figure 11 plots each CDF for the error burst lengths of the three traces. The mean, standard deviation, and maximum values are summarized in Table 4. Note that *real trace*

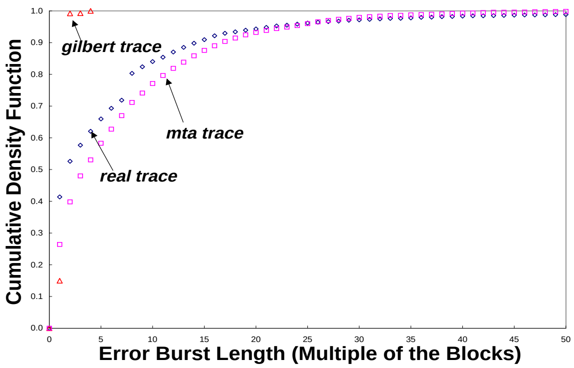


Figure 11. Corrupted burst length distribution.

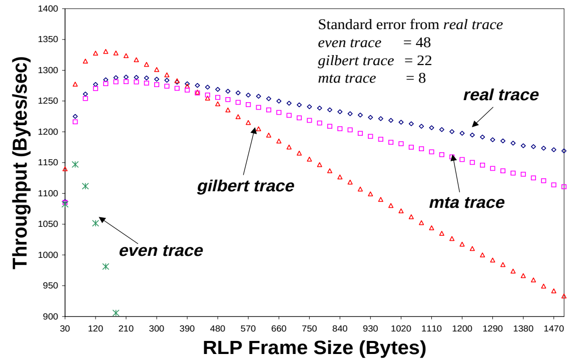


Figure 12. Retrace analysis of four traces.

and the MTA artificial trace experience similar bursty characteristics with 95 percent of the corrupted burst lengths being smaller than 22 frames long, while in the Gilbert trace 95 percent of the corrupted burst lengths are of size one. These results show that the error burst distribution of the MTA trace represents a much closer approximation to the collected trace, *real trace*.

Trace	Mean	St Deviation	Maximum
real_trace	6	14	126
mta_trace	7.0	8.1	82
gilbert_trace	1.8	0.4	4

Table 4. Error Length Statistics

To demonstrate the importance of an accurate model for setting system parameters, we cite an example where a naive assumption about the channel statistics can lead to poor performance. In [11], we showed how an inaccurate channel model can lead to poor decision on the optimal RLP frame size of an enhanced multiple radio block implementation (see Section 1). We repeat this demonstration using the *real trace*, artificial traces from MTA and Gilbert, and an artificial trace based on trivial assumptions we call *even trace*. We artificially generated *even trace* with the same BLER as *real trace*, but with an even error distribution. We then perform retrace analysis on the four traces, yielding the results shown in Figure 12. Note that the throughput for *even trace* decreases dramatically as frame size increases, yielding an optimal frame size of only 60 bytes or 2 radio blocks. The Gilbert trace experiences higher throughput values for small frame sizes, but throughput decreases rapidly as the frame size increases. Its optimal frame size is 150 bytes (5 radio blocks). In contrast, the throughput plots for *real trace* and the MTA trace follow similar paths. Furthermore, they both yield an optimal frame size of 210 bytes (7 radio blocks). In this particular case, retrace analysis shows that the improved accuracy of the MTA artificial trace over the Gilbert artificial trace leads to a more optimal design decision.

We used the standard error equation (see Equation 3) to measure how closely each artificial trace approximates the *real trace*. The standard error for *even trace* was 48, for *gilbert trace* was 22, and for *mta trace* was 8. Small standard error values signify that the traces experience similar error statistics.

In summary, we used the characteristics from the MTA and Gilbert models to generate artificial traces, and used these traces to measure how accurately both algorithms model real traces. Both CDF and retrace analysis show that the artificial trace from the MTA model more accurately portrays the original *real trace*. Thus, we conclude that the MTA model provides a more accurate approximation technique than the traditional Gilbert model.

8 Conclusion

In this paper, we present a novel algorithm for modeling networks channels that experience time varying error statistics. The time varying nature of wirelss and some wired channels has been a limiting factor in the analysis or modeling using Discrete Time Markov Chains. However, our Markov-based Trace Analysis algorithm and techniques allow us to separate a non-stationary network trace into stationary traces and to accurately model the traces using DTMCs.

We compare the application of the MTA model and the traditional Gilbert model to traces collected in the GSM wireless digital cellular networks and show that MTA model synthetic traces have burst error distributions that are closer to the real distributions of collected traces than the distribution of traces generated from the Gilbert model.

We further show that when using retrace analysis to calculate the throughput for different frame sizes, our MTA model yields the correct optimal frame size decision, whereas less accurate models including the Gilbert model and an even error distribution model yield incorrect and non-optimal frame sizes. The results of the retrace analysis gives an example where a less accurate traffic model leads to the wrong design decision.

We are in the process of applying the MTA model to the problem of modeling next-generation 2.5 generation and 3rd generation GSM networks, including the General Packet Radio Service (GPRS). Both networks currently have limited prototype deployment, making experimentation difficult. However, by creating MTA models for each network, we will enable easy, rapid experimentation and prototyping.

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] J. Bolot, S. Fosse-Parisis, and D.Towsley. Adaptive FEC-based error control for internet telephony. *Proc. Infocom '99*, March 1999.

[4] G. Box. *Time Series Analysis: Forecasting and Control*. Prentice hall, 1994.

[5] ETSI GSM Technical Specification 04.22. *GSM Radio Link Protocol for data and telematic services, Version 5*, December 1995.

[6] ETSI GSM Technical Specification 04.22. *Digital cellular communications system (Phase 2+); Radio Link Protocol for data and telematic services on the Mobile Station-Base Station Systems(MS-BSS) Interface and the Base Station System-Mobile Switching Center (BSS-MSC) interface, Version 6.1.0*, November 1998.

[7] ETSI GSM Technical Specification 04.22. *GSM Radio Link Protocol for data and telematic services, Version 6.1*, November 1998.

[8] ETSI GSM Technical Specification 05.03. *Digital cellular communications system (GSM Radio Access Phase 3); Channel coding, Version 6.0.0*, January 1998.

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

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

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

[12] R. Ludwig and B. Rathonyi. Link layer enhancements for TCP/IP over GSM. In *Proceedings of IEEE INFOCOM*, 1999.

[13] N. Merhav, M. Gutman, and J. Ziv. On the estimation of the order of a Markov chain and universal data compression. *IEEE Trans. on Information Technology*, 35(5):1014–1019, September 1989.

[14] M. Mouly and M. B. Pautet. *The GSM System for Mobile Communications*. Cell and Sys, France, 1992.

[15] 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.

[16] Sheldom M. Ross. *Stochastic Processes*. John Wiley and Sons, 1996.

[17] W. Simpson. The point-to-point protocol. *RFC 1661*, Jul 1994.

[18] W.R. Stevens. *TCP/IP Illustrated, Volume I (The Protocols)*. Addison Wesley, 1994.

[19] 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.

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