5-2009

Fast Generation of Receiver-Based Statistics in Frequency Hop Radios without Decoder Simulation

Michael Masse
Clemson University, massem@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses
Part of the Electrical and Computer Engineering Commons

Recommended Citation
https://tigerprints.clemson.edu/all_theses/606

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.
Fast Generation of Receiver-Based Statistics in Frequency Hop Radios without Decoder Simulation

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Engineering

by
Michael Masse
August 2009

Accepted by:
Dr. Michael Pursley, Committee Chair
Dr. Daniel Noneaker
Dr. Harlan Russell
Abstract

Receiver-based statistics can contain useful information about the quality a wireless channel. We examine methods of generating receiver statistics without real-time simulation of the receiver and the decoder, particularly during a network-level simulation. Attention is restricted to a receiver employing binary orthogonal signaling, noncoherent demodulation, and a soft-decision decoder in a frequency-hop network. Data is gathered from off-line simulations of channel, demodulation, and decoding processes, using a family of turbo product codes. Study and analysis of the data are used to characterize the behavior of the receiver statistics and determine on-line methods of generating the statistics in network-level simulations. To measure the performance of our generation techniques, we compare the results with simulation data. The goal of the generation methods is to provide receiver-based statistics for simulations of higher layer protocols without the need for simulation of the decoding process, which is time consuming. With this goal in mind, we design our generation methods to be simple and fast, favoring approximations over exact replications whenever the resulting simplification is significant. Both demodulator and decoder statistics are considered. Results are given which demonstrate the fidelity of our generation methods.
# Table of Contents

Title Page ................................................................. i

Abstract ................................................................. ii

List of Tables ........................................................... iv

List of Figures ........................................................... v

1 Introduction ............................................................ 1

2 Channel Models ......................................................... 4
   2.1 Additive White Gaussian Noise .................................. 4
   2.2 Partial-Band Interference ....................................... 4
   2.3 Dynamic Channels ............................................... 5

3 System Model .......................................................... 6
   3.1 Modulation, Demodulation, and Soft Decisions ................. 6
   3.2 Adaptive Scaling ................................................. 7
   3.3 Packet Formatting and Decoding ................................ 9
   3.4 Receiver Statistics ............................................. 9
   3.5 Adaptive Coding Protocol ..................................... 10

4 Receiver Statistics and Generation Models ......................... 12
   4.1 Error Count ...................................................... 12
   4.2 Iteration Count ................................................ 22

5 Adaptive Coding Results ............................................. 30
   5.1 Error Count ...................................................... 31
   5.2 Iteration Count ................................................ 33
   5.3 EI Combination ................................................ 35
   5.4 Dynamic Channels ............................................. 37
   5.5 Run Time Comparisons ....................................... 40
   5.6 Data Averaging ................................................ 40

6 Conclusion ............................................................. 42
List of Tables

3.1 Adaptive scaling parameters $w_0$ and $w_1$ for each code rate . . . . . . . . . . . . . . . 8
5.1 Run time comparison for receiver simulation and statistic generation . . . . . . . . . . 39
List of Figures

2.1 Example Markov chain .......................................................... 5
3.1 Receiver diagram ................................................................. 7
4.1 Observed distribution of error counts compared with a binomial mass function, SENR = 3.5 dB .................................................. 13
4.2 Observed distribution of error counts compared with a tiered binomial mass function for several values of \( \rho \), SENR = 15 dB and SEIR = −3 dB .................................................. 14
4.3 Probability of packet error as a function of error count for SENR = 3.2 dB and a code of rate 0.325 ........................................ 15
4.4 Comparison of observed and generated error count distributions, conditioned on correct packet decoding, for SENR = 3.2 dB and a code of rate 0.325 .................................................. 17
4.5 Comparison of bin sizes for error count generation, AWGN channel, SENR = 4.7 dB, \( r = 0.495 \) ................................................................. 18
4.6 Packet error probability for AWGN channel, a code of rate 0.325, and several values of SENR .................................................. 19
4.7 Binomial density functions for probability of binary symbol error given an AWGN channel and several values of SENR .................................................. 19
4.8 Error count distributions for successful decoding, AWGN channel, and a code of rate 0.495; averaging techniques are compared .................................................. 20
4.9 Error count distributions for PBI channel with SENR = 15 dB, SEIR = −3 dB, and a code of rate 0.325; averaging techniques are compared .................................................. 21
4.10 Error count distributions for PBI channel with SENR = 15 dB, \( \rho = 0.5 \), and a code of rate 0.495; averaging techniques are compared .................................................. 22
4.11 Iteration count distributions as a function of error count(\( EC \)) for AWGN channel, SENR = 3.5 dB, and \( r = 0.325 \) .................................................. 23
4.12 Triangle approximation example ............................................ 24
4.13 Comparison of observed and generated iteration count distributions using the unmodified triangle method .................................................. 25
4.14 Comparison of observed and generated iteration count distributions using the improved triangle method .................................................. 26
4.15 Comparison of observed and generated iteration count distributions using the triangle and improved triangle methods .................................................. 27
4.16 Iteration count distributions for AWGN channel and code of rate 0.325; averaging techniques are compared .................................................. 28
4.17 Iteration count distributions for PBI channel with SENR = 15 dB, SEIR = −3 dB, and a code of rate 0.495; averaging techniques are compared .................................................. 28
4.18 Iteration count distributions for PBI channel with SENR = 15 dB, \( \rho = 0.5 \), and a code of rate 0.660; averaging techniques are compared .................................................. 29
5.1 Throughput comparison of generated and simulated error count results for AWGN channel. .................................................. 31
5.2 Throughput comparison of generated error count results for different bin sizes in AWGN channel. ................................................. 32
5.3 Throughput comparison of generated error count results for AWGN channel. ................................................................. 32
5.4 Throughput comparison of generated error count results for strong PBI channel with SENR = 15 dB and SEIR = −3 dB. ........... 33
5.5 Throughput comparison of generated error count results for weak PBI channel with SENR = 15 dB and SEIR = 6 dB. ................. 34
5.6 Throughput comparison of generated iteration count results for AWGN channel. ............................................................... 34
5.7 Throughput comparison of generated iteration count results for strong PBI channel with SENR = 15 dB and SEIR = −3 dB. ........ 35
5.8 Throughput comparison of generated iteration count results for weak PBI channel with SENR = 15 dB and SEIR = 6 dB. ............ 36
5.9 Throughput comparison of generated error and iteration count results for AWGN channel. .................................................... 36
5.10 Throughput comparison of generated error and iteration count results for strong PBI channel with SENR = 15 dB and SEIR = −3 dB. ................................................................. 37
5.11 Throughput comparison of generated error and iteration count results for weak PBI channel with SENR = 15 dB and SEIR = 6 dB. ................................................................. 37
5.12 Adaptive Protocol Performance from simulated and generated statistics for AWGN channel with dynamic excess path loss governed by six state Markov chain with Δ = 2 dB and p = 0.1. ..................................................... 38
5.13 Adaptive Protocol Performance from simulated and generated statistics for PBI channel with SENR = 15 dB and dynamic ρ governed by three state Markov chain with states: ρ₀ = 0, ρ₁ = 0.25, ρ₂ = 0.5 ..................................................... 39
Chapter 1

Introduction

Channel conditions in mobile ad hoc wireless networks can vary significantly from one packet to the next. As a result many protocols are employed to adapt parameters such as the transmission power and code rate in order to maximize throughput while minimizing interference and power consumption in mobile terminals. Some protocols rely on perfect channel state information (e.g. [4] and [14]), or make use of pilot symbols (e.g. [2], [3], and [8]) to make channel estimations. Some protocols rely on a simultaneous feedback link to relay channel information from the receiver to the transmitter (e.g. [6] and [7]). Other protocols only rely on the channel information contained in simple receiver statistics, which require little-to-no extra work to obtain in a practical system. The receiver statistic protocols are the most interesting, since they do not rely on concurrent feedback links, perfect channel state information, or more complicated estimation techniques ([11] and [12]). The information contained in these receiver statistics may also be useful for higher-layer protocols, such as routing. Since physical-layer protocols impact the performance of higher-layer protocols, ignoring their effects in higher-layer simulations may reduce the accuracy of the results. As a result, it is desirable to simulate the workings of the receivers that produce these statistics in network simulations which may not otherwise account for the exact operation of the physical layer.

Simulations of the demodulator and decoder, especially iterative decoders, are time consuming. When only a single link is considered, the time may be acceptable, but when multiple nodes and traffic flows are considered, the time becomes prohibitive. In a multi-hop network, a single packet may require several transmissions to reach its final destination. Thus the run time will not scale linearly with the number of nodes, if each node has the same amount of traffic. Simulations
of the receiver may dominate the computing resources, even though the focus of the research is a higher-layer protocol, such as a routing protocol. Therefore, for each receiver statistic that is used, it is desirable to develop simple models or approximations, which can be simulated in much less time. The benefit is twofold: first, the effect of these adaptive protocols on networks and network protocols can be observed, and second, higher-layer protocols that make use of receiver statistics can be implemented and tested.

In this paper, we introduce models for quickly generating receiver statistics without the need for a precise simulation of the receiver. Our models are developed from analysis and empirical studies of the receiver statistics. Where possible, we have designed our models to reflect the actual behavior of the system, rather than just the results, which are the receiver statistics themselves; however, simplicity and speed were the primary design goals. We have tried to minimize the size of any lookup tables that are required.

We consider frequency-hop (FH) networks in which there are several modulation symbols per hop and several hops per packet. Binary orthogonal modulation is used with noncoherent demodulation. The demodulator provides soft-decisions to an iterative decoder for a family of turbo product codes (TPCs). Receiver statistics are modeled on a per-packet basis, since the block size is not the same for all codes, while the packet length is fixed.

Thermal noise in the receiver is modeled as additive white Gaussian noise (AWGN). Partial-band interference (PBI) is modeled as a block interference channel for which a fraction, $\rho$, of the frequency slots are affected by band-limited white Gaussian noise. The total power in the interference is fixed and independent of $\rho$. The channel parameters affect the distribution of each receiver statistic, adding complexity to our models. We attempt to reduce this complexity through techniques such as averaging. We wish to examine dynamic channels as well as static channels. To model dynamic channels, we employ Markov chains, for which each state of the chain represents a particular value of a channel parameter. Examples include dynamic path-loss, where each state represents some amount of excess path-loss, relative to a base value; and dynamic PBI, where each state represents a different value of $\rho$.

One receiver statistics, the error count, can be obtained by re-encoding the information bits from a successfully decoded packet. The encoded packet is then compared with the hard-decision outputs of the demodulator to count the number of hard-decision errors incurred on the channel. The distribution of the error count for all packets sent over a given channel can be determined easily,
as long as the probability of binary symbol error for hard-decision demodulation is known. However, the error count can be obtained in a real receiver only for correctly decoded packets. Unsurprisingly, the packet error probability is not independent of the error count, which means that the distribution of the error count for packets that can be correctly decoded is not the same as the distribution for all packets. Furthermore, since the inner workings of the decoder are not known, an analytical determination of the packet error probability (PEP) as a function of error count is not possible.

The iteration count, which is the number of iterations required by the decoder to successfully decode a packet, can be obtained directly from any iterative decoder. This statistic can only take on a limited number of values, as the iterative decoder we employ is limited to a maximum number of iterations. The likelihood is small that a packet can be successfully decoded but requires a number of iterations larger than the limit in our examples. The unknown and complex inner workings of the decoder again make an analytical approach impossible. Without a basis in analysis, models for the iteration count must be developed purely from observations of empirical data.

We compare the distributions of our models with the empirically determined correct distributions to gauge their performance. We also compare the results of employing these models in place of actual receiver simulations. In particular, we use an adaptive coding protocol to demonstrate the effectiveness of our models. This protocol uses one or more receiver statistics to select a code rate that will provide both high throughput and adequate error protection for the current channel state. We show that our models are able to reproduce the results obtained from simulation of the demodulator and decoder.
Chapter 2

Channel Models

We examine a FH system in which the frequency band is divided into some number of frequency slots. Each packet transmission is split across several frequency slots, but not all slots are used for every packet. Channel parameters, such as the presence of PBI, are held constant during each use of a frequency slot, but may not be the same between different slots or subsequent uses of the same slot.

2.1 Additive White Gaussian Noise

Thermal noise in the receiver is modeled as AWGN with one-sided spectral density $N_0$. Since it is convenient to discuss the signal-to-noise ratio in dB, we define the symbol-energy-to-noise-ratio (SENR) as $10 \log_{10}(E_s/N_0)$, where $E_s$ is the energy per binary modulation symbol.

2.2 Partial-Band Interference

We use a block interference model \cite{9} to evaluate the effects of PBI. A fraction $\rho$ of the frequency slots of a FH system are affected by PBI, while the remaining fraction $1 - \rho$ slots are not affected. The interference consists of band-limited white Gaussian noise with one-sided power spectral density $N_I/\rho$ in affected slots, which are said to be hit. The total power of the interference is fixed and independent of $\rho$. We define the symbol-energy-to-interference-ratio (SEIR) to be $10 \log_{10}(E_s/N_I)$. If $\rho = 0$, then no slots are affected by PBI, and if $\rho = 1$, all slots are affected.
The frequency slots used in a packet transmission are determined by a pseudo-random hopping pattern. Because the sequence is random, the presence of interference in each slot can be modeled as a random variable, having a Bernoulli distribution with parameter $\rho$. We assume that there are significantly more slots than are needed for a single packet, and that the hopping pattern is long enough that we can ignore the possibility of repeated use of a slot within a packet. Therefore, we assume independence of all frequency slots in a packet.

### 2.3 Dynamic Channels

We wish to model the effects of dynamic channels, in which one or more channel parameters may vary during a session. All channel parameters are fixed over the duration of a packet, but may change between packet transmissions, with changes governed by a discrete-time Markov chain. Each unit of time corresponds to the transmission of one packet. Fig. 2.1 shows an example of such a chain with transition probability $p$. Each state of the Markov chain represents one value of a particular channel parameter, such as $\rho$ or excess path loss, defined as any path loss above some nominal level. Dynamic channels are useful for the study of adaptive protocols, which are designed to adjust to changes in the channel state.
Chapter 3

System Model

3.1 Modulation, Demodulation, and Soft Decisions

We examine a slow FH system with multiple modulation symbols transmitted per hop, or dwell interval; multiple hops per packet; and a hopping pattern determined by a pseudo-random sequence. We employ binary orthogonal signals with noncoherent demodulation. The demodulator consists of a pair of noncoherent correlators, one tuned to each of the orthogonal signals available to the transmitter. Binary frequency-shift keying (BFSK) is an example of a modulation system that matches our model. In the transmitter, data bits are mapped into modulation symbols, which are sent over the channel. Fig. 3.1 shows a block-diagram of the receiver; the output of the channel is represented by $Y(t)$. The demodulator maps the received signal back into binary symbols. For hard-decisions, the value of the received symbol is determined by the correlator with the larger output. The outputs of the correlators for the $k$th symbol in the $j$th dwell interval are denoted by $Z_{0,j,k}$ and $Z_{1,j,k}$. The soft-decision for the received symbol is given by $\Lambda(Z_{0,j,k}, Z_{1,j,k})$.

The log likelihood ratio (LLR) metric, $\Lambda_j$, provides a good soft-decision metric for TPCs for all channels considered [12]. $\Lambda_j$ is defined by

$$\Lambda_j(z_0, z_1) = \ln \left\{ \frac{I_0(z_0/\sigma^2_j)}{I_0(z_1/\sigma^2_j)} \right\}, \quad (3.1)$$

where $I_0$ is the zero-order modified Bessel function of the first kind and $\sigma^2_j$ corresponds to the noise variance, which may include both thermal noise and partial-band interference. The LLR metric is
unattractive for implementation in a real system due to the need for Bessel function computations and knowledge of the noise variance. As a result, we use the log-ratio metric $\Lambda_r$, which is defined by

$$\Lambda_r(z_0, z_1) = \ln\left(\frac{z_0}{z_1}\right).$$

The log-ratio metric does not require knowledge of the noise variance or other channel parameters, and it does not rely on difficult calculations of special functions, making it a good choice for practical implementation. Despite its greater simplicity, the log-ratio metric achieves performance typically within 0.1-0.2 dB of that of the LLR metric [12].

### 3.2 Adaptive Scaling

To improve the performance of the soft-decision decoder, the outputs of the demodulator are scaled by an additional factor determined by the quality of the symbols in each dwell interval. A dwell interval may be of poor quality due to the presence of PBI, for instance. The adaptive scaling subsystem is designed to give a lower weight to dwells with a low reliability. In one extreme, all the symbols in a dwell are erased, while in the other, all soft-decisions are passed to the decoder without any further scaling.
Table 3.1: Adaptive scaling parameters $w_0$ and $w_1$ for each code rate

<table>
<thead>
<tr>
<th>Code rate</th>
<th>0.236</th>
<th>0.325</th>
<th>0.495</th>
<th>0.660</th>
<th>0.793</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_0$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$w_1$</td>
<td>12</td>
<td>11</td>
<td>15</td>
<td>14</td>
<td>11</td>
</tr>
</tbody>
</table>

The demodulator statistic $W_j$ provides a relative measure of the reliability of the symbols in dwell $j$. For noncoherent demodulation, the statistic is

$$W_j = \frac{\sum_k \max\{Z_{0,j,k}, Z_{1,j,k}\}}{\sum_k \min\{Z_{0,j,k}, Z_{1,j,k}\}},$$

where the sums are over all symbol positions $k$ in the $j$th dwell interval. A small value of $W_j$ corresponds to a low reliability of the symbols in dwell $j$. The demodulator statistic is used to determine the scale factor $\lambda_j = g(W_j)$, where $g$ is a nondecreasing function such that $0 \leq g(w) \leq 1$. For this investigation we define

$$g(w) = \begin{cases} 
0, & w < w_0, \\
(w - w_0)/(w_1 - w_0), & w_0 \leq w \leq w_1, \\
1, & w > w_1. 
\end{cases}$$

(3.4)

If $W_j < w_0$, all symbols in dwell $j$ will be erased, and if $W_j > w_1$, all soft-decisions will be passed to the decoder unaltered. It is important that the scaling parameters $w_0$ and $w_1$ be correctly selected, or decoder performance may suffer. A high value of $w_0$ will result in more erasures – useful for PBI channels, for which dwells are either affected by strong interference or no interference. In fading channels, where the noise variance of each dwell varies over a continuum, a large $w_0$ will cause too many erasures and lead to decoder failure. Similarly, if the value of $w_1$ is too large, all soft-decisions will be scaled down too much, decreasing decoder performance. A small value of $w_1$ reduces the impact of the adaptive scaling subsystem, as more of the demodulator outputs will be passed straight to the decoder. Since each code has different error-correcting capacity, the parameters $w_0$ and $w_1$ are selected for each code to give good performance over both fading and PBI channels. The values for $w_0$ and $w_1$ are given in Table 3.1 for the five codes we use. The adaptive scaling subsystem is discussed in further detail in [12].
3.3 Packet Formatting and Decoding

Each packet contains 4096 binary code symbols, which requires multiple code words for codes with a block length less than 4096. All words in a packet must be correctly decoded for the packet to be successfully received, and partial retransmissions are not allowed. When multiple code words are used, they are interleaved such that an equal number of bits from each word is sent in each dwell interval. The result is that errors are spread evenly over all words. To combat burst errors, each packet is interleaved using an S-random interleaver with an S value of 45 [5].

For transmission, each packet is divided into \( N_d \) dwell intervals, each containing \( N \) binary symbols. All binary symbols in a dwell interval are sent within the same frequency slot, and each dwell interval is sent in a new frequency slot. For our simulations, we set \( N = 32 \), which results in \( N_d = 128 \), since \( N_d = 4096 / N \).

We simulate TPCs using a software emulation of an off-the-shelf single-chip encoder/decoder chip [1]. We use five codes with rates 0.236, 0.325, 0.495, 0.660, and 0.793. The codes of rates 0.236 and 0.660 have block sizes of 2048 and 1024, respectively, while the other three have a block size of 4096. The decoder is considered to be a black box, with quantized soft-decisions as inputs and the decoded packet and iteration count as outputs. Nothing else is known about the interior workings of the decoder.

3.4 Receiver Statistics

3.4.1 Error Count

The error count refers to the number of binary symbol errors incurred as a result of hard-decision demodulation of a received packet. One way to obtain the error count is to re-encode a correctly decoded packet and compare the results with the outputs of the demodulator. This method has two requirements: the receiver must contain an encoder, and the packet must be decoded correctly. The first requirement is reasonable, as an encoder is likely to be included if the device is intended for transmission and reception, and there are single-chip encoder and decoder solutions available [1]. While the second requirement means that an error count cannot be obtained for incorrect packets, a packet error event carries a certain amount of information about the quality or state of the channel. The error count is a good metric of channel quality because there is a
correlation between the number of errors in a packet and the probability that it is decodable. A high error count suggests that a lower code rate is needed, while a low error count implies that a higher code rate will suffice.

3.4.2 Iteration Count

The iteration count refers to the number of iterations required by the decoder to correctly decode a packet, making it only available for iterative decoding methods, which we do not consider. The iteration count can be obtained for all packets, but a failed packet will have an iteration count of the maximum allowed number of iterations. Once again, the event of decoder failure carries significant information. The iteration count is also used in an adaptive coding protocol, providing information relative to the ability of a particular code rate to correct the effects of the channel. Since the iteration count is a good indicator of the difficulty in decoding a packet, it is a good metric for code selection. A low iteration count implies that a higher code rate might also be successful, thus increasing throughput. A high iteration count implies that the current code rate is too high and that a lower rate would give better performance.

3.5 Adaptive Coding Protocol

An adaptive coding protocol which makes use of receiver statistics is a good test case for the generation methods. We consider an adaptive coding protocol which uses the error count and iteration count from the previous packet to determine which code rate should be used for the next packet. The protocols are discussed in more detail in [11] and [12]. If the generation methods that we develop do not produce the proper distributions, then the adaptive protocols will make decisions based on incorrect data and possibly select a different code as a result. Since the protocol performance is close to theoretical maximums, it is likely that any erroneous choice will result in a lower average throughput. Producing skewed or erroneous performance results would be undesirable, as it would cast doubt on any research that employed the generation methods.

The protocol can use either statistic by itself to select the proper code rate. The parameters for the error count protocol consist of a series of thresholds $\xi_0 - \xi_{n_c}$, where $n_c$ is the number of code rates available to the adaptive coding system. Code $i$ is selected for the next packet transmission when the error count $\xi$ for the previous packet is between $\xi_i$ and $\xi_{i-1}$. By convention, we set $\xi_0 = n$,
where \( n \) is the packet length, and \( \xi_{ne} = 0 \). For all \( i < j \), \( \xi_i > \xi_j \). Using only the error count, the protocol is able to select any code rate, regardless of the rate used in the previous packet. In the event of a packet error, no error count is available, and the protocol automatically chooses the next lower rate code.

The iteration count uses two parameters, \( \upsilon_i \) and \( \delta_i \), for each code rate \( i \), with the requirement that \( \upsilon_i < \delta_i \). If the iteration count for a packet using code \( i \) is \( \upsilon \) and \( \upsilon < \upsilon_i \), then the next packet will use code rate \( i + 1 \), if available. Similarly, if \( \upsilon > \delta_i \), then code \( i - 1 \), if available, will be used for the next packet. Otherwise, the system will continue to use code \( i \). Note the disadvantage that the protocol can only select the next higher or lower rate code.

When both statistics are employed together, better performance can be obtained [12]. The code selection method for the combination relies on the selection methods for the individual statistics. Suppose the current code is \( i \) and the error count and iteration count imply that codes \( j_e \) and \( j_i \), respectively, should be used for the next packet. If \( j_e < i \) and \( j_i < i \), then code \( i - 1 \) is selected, but if \( j_e > i \) and \( j_i > i \), then code \( i + 1 \) is selected. In all other cases, code \( i \) is used. In other words, the combination only changes codes when both statistics agree on the direction of the change. As with the iteration count, only the next higher or lower rate codes, in addition to the current code, may be selected.
Chapter 4

Receiver Statistics and Generation Models

4.1 Error Count

4.1.1 Analysis

The error count is the number of binary hard-decision errors incurred over the channel. If we let $Z_i$ be an indicator variable for the symbol error event in the $i$th binary code symbol, then $Z_i$ will have a Bernoulli distribution with parameter $p$, where $p$ is the probability of hard-decision binary symbol error for a given modulation scheme and channel state. For binary orthogonal signaling with noncoherent demodulation, such as BFSK, the symbol error probability for an AWGN channel is given in [10] by

$$p = \frac{1}{2} \exp \left( \frac{-E_s}{2N_0} \right).$$  \hspace{1cm} (4.1)

Since we model noise as independent samples of a Gaussian random process with mean 0 and variance $\sigma^2$, we can treat the symbol error indicators as independent random variables. Thus, the error count $\xi$ can be obtained by taking the sum of of $Z_i$ for all $i$,

$$\xi = \sum_i Z_i,$$  \hspace{1cm} (4.2)
Figure 4.1: Observed distribution of error counts compared with a binomial mass function, SENR = 3.5 dB

and ξ can therefore be represented as a random variable with a binomial distribution with parameters $n$ and $p$, where $n$ is the number of binary symbols in a packet. Fig. 4.1 shows that empirical data confirms analytical expectations, and the error count can be modeled with a binomial random variable. The parameter $p$ can easily be calculated from the given channel state, which would be known to the simulation.

For channels with PBI, the error count distribution is no longer binomial, as some fraction of the symbols is also affected by the interference. Since a fraction $\rho$ of the frequency slots are affected by PBI, the probability that any randomly chosen frequency slot is affected by PBI is $\rho$. The error count can be broken down into the number of errors in dwells that are hit by PBI and the number of errors in dwells that are not hit by PBI. The number of affected dwells in a packet can be represented by a binomial random variable $H$ with parameters $(N_d, \rho)$. If $H$ dwells are hit with PBI, then the number of binary symbol errors in dwells affected by PBI has a binomial distribution with parameters $(NH, p_1)$. The number of binary symbol errors in dwells not affected by PBI has a binomial distribution with parameters $(N(N_d - H), p_0)$. The parameters $p_0$ and $p_1$ are the probabilities of binary symbol error in dwells that are not hit and hit by interference, respectively;
Figure 4.2: Observed distribution of error counts compared with a tiered binomial mass function for several values of $\rho$, $\text{SENR} = 15 \, \text{dB}$ and $\text{SEIR} = -3 \, \text{dB}$ for BFSK, they are given by

$$p_0 = \frac{1}{2} \exp \left( \frac{-\mathcal{E}_s}{2N_0} \right)$$  

$$p_1 = \frac{1}{2} \exp \left( \frac{-\mathcal{E}_s}{2(N_0 + N_i/\rho)} \right).$$  

Let the random variables $H$, $\xi_0$, and $\xi_1$ be defined as follows:

$$H \sim \text{binomial}(N_d, \rho)$$  

$$\xi_0 \sim \text{binomial}(N(N_d - H), p_0)$$  

$$\xi_1 \sim \text{binomial}(NH, p_1).$$

The total error count for a packet is the sum of $\xi_0$ and $\xi_1$. We refer to this new distribution as a \textit{tiered} binomial distribution. Fig. 4.2 shows some sample error count distributions for select values of $\rho$. 


Figure 4.3: Probability of packet error as a function of error count for SENR = 3.2 dB and a code of rate 0.325

4.1.2 Modeling

A real system will be able to determine an error count only for packets that can be correctly decoded. The conditional distribution of the error count given that the packet decoded successfully will be different from a binomial distribution if the PEP depends on the error count. The correlation between the error count and PEP is shown in Fig. 4.3. The conditional distribution of the error count given successful packet decoding must be determined through simulation. Examples of the distribution of the error count for all packets and the distribution of the error count for correct packets only are given in Fig. 4.4 and are labeled as “Empirical Data.”

To solve this problem, we propose a method of generating the conditional distribution of the error count given correct packets from the unconditional distribution. Let the conditional PEP for error count $x$ and a given channel state be written $\text{PEP}(x)$. Let $g(x)$ represent the probability mass function of a binomial density with parameters $(NN_d, p)$ for an AWGN channel, or a tiered binomial density with parameters $(N_d, \rho, N, p_0, p_1)$ for a PBI channel, and let $f(x)$ represent the probability mass function of the desired conditional error count distribution, given a packet decoded successfully. The following steps illustrate how to generate a random variable with probability mass

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
Error Count & 650 & 700 & 750 & 800 \\
\hline
Packet Error Probability & 0.000 & 0.200 & 0.400 & 0.600 & 0.800 & 1.000 \\
\hline
\end{tabular}
\end{center}
function \( f \):

1. Draw a discrete random variable \( X \) according to the distribution with probability mass function \( g \).

2. Draw a continuous random variable \( U \) from a uniform distribution over \([0,1] \).

3. Compare \( U \) with \( \text{PEP}(X) \).

   - If \( U > \text{PEP}(X) \), the error count for the packet is \( X \), and the packet was successfully decoded.
   
   - If \( U \leq \text{PEP}(X) \), the packet failed to decode, and the error count \( X \) is discarded.

The design of this method is meant to reflect the two-step process of simulating the demodulator and decoder. In the first step, the packet is corrupted by the channel, represented by a random number of binary symbol errors. In the second step, the corrupted packet is passed through the decoder, which either decodes or fails to decode the packet. The second part is represented by steps 2 and 3 of the above method, wherein a Bernoulli random variable simulates the success or failure of the decoder. The parameter of the Bernoulli distribution is the conditional PEP, given the randomly selected number of binary symbol errors. The PEPs are determined empirically through simulation of the receiver and decoder. Fig. 4.4 shows that this method accurately recreates the conditional distribution of the error count, given the packet decoded successfully. Note that as the average PEP for a channel state approaches zero, the conditional distribution of the error count converges to the unconditional binomial distribution, as expected.

### 4.1.3 Data Set Reduction

The amount of space required to store a PEP for each possible error count, code rate, and channel state can be large. A large data set requires more storage space, more time to read, more memory, and more time to search – any of which can be detrimental to the run time of a network-level simulation. We wish to exploit redundancy in the data to reduce the overall amount of data that must be stored. It is likely that the accuracy of our generated statistics will be affected by this compression, so we examine any negative impact. Another thing to note is that our models require data for every channel state for which statistics will be generated. If we can use averaged data to generate statistics, then we can possibly reduce the number of simulations required to obtain
Figure 4.4: Comparison of observed and generated error count distributions, conditioned on correct packet decoding, for SENR = 3.2 dB and a code of rate 0.325.

the generation data. Rather than gathering data for every single channel state, we can gather data for a representative subset, and we can use the average data for all channel states in the set. The constituency of a representative subset can be determined from analysis of the binomial distribution of the error count for each channel state. The subset should be selected such that the channel states are evenly spaced along the channel parameters and such that a statistically significant sample of packets is collected at each error count. In this way, the subset of channel states will provide accurate PEP data for all values of the error count. Alternatively, a representative subset can be determined from the results of off-line simulation. In this case, there is no reduction in the amount of off-line simulation required, but there is still a reduction in the amount of off-line data that must be stored. Since our goal is to reduce the amount of on-line simulation, rather than the amount of off-line simulation, we do not investigate this further or propose a robust method of determining a representative subset.

We first seek to reduce the resolution of the error count data that we store. For a packet of 4096 binary symbols, the error count can take on any integer value in the range [0,4096], though most of the larger values can be disregarded due to low probability of occurrence and high probability of
Experience with the error count protocol has shown it to be insensitive to parameter changes with a difference smaller than 10, so a one-bit resolution may be finer resolution than is necessary. It is possible that we can reduce the required data by grouping the error counts into bins of coarser resolution. The PEPs are averaged over all error counts in a bin, and only the average is stored. Error counts are still generated to a one-bit resolution, but the PEP used to accept or reject each draw is the average for the appropriate bin. In Fig. 4.5 we compare the generated densities that result from using several bin sizes. Note from the graph that the average PEP is biased high for error counts in the low end of a bin and biased low for error counts in the high end of a bin. From this graph, we see that a bin size of 10 provides a good working point. Anything less gives performance near to a one-bit resolution, and anything higher seems to stray too far from the actual density. All further results that employ averaging use a bin size of 10.

Next, we look at the channel parameters for further data reduction. Fig. 4.6 shows the packet error probability as a function of the error count for several values of SENR in an AWGN channel. Observe that the PEP increases as a function of SENR for fixed values of the error count. The similarity of the curves suggests that averaging will have only a small negative impact on the
Figure 4.6: Packet error probability for AWGN channel, a code of rate 0.325, and several values of SENR

Figure 4.7: Binomial density functions for probability of binary symbol error given an AWGN channel and several values of SENR
Figure 4.8: Error count distributions for successful decoding, AWGN channel, and a code of rate 0.495; averaging techniques are compared.

accuracy of our generated results. Comparing with Fig. 4.7, we see that the curves that are farthest apart have only a small overlap in the appropriate error count density functions, while the curves whose density functions agree also agree in PEP. The implication is that some form of weighted averaging will produce accurate results. To average the data, we simulate an equal number of packets for each channel condition, counting the total number of packet successes and failures for each error count bin. We can then compute the average packet error rate for each bin. This averaging method gives appropriate weight to each channel state based on the probability that a given error count was produced by that particular channel state. Thus, the data is most accurate for events of high probability. Inaccuracy for events of low probability has only a small negative effect.

Fig. 4.8 compares the observed distribution and generated distributions of the error count for two different values of SENR in an AWGN channel. Both generated curves are produced from data with an error count bin size of 10, but the data used to produce the second curve is the average PEP for all AWGN channel states in the range $0 \leq \text{SENR} \leq 8 \text{ dB}$. In one of the sets of curves, the PEP is low enough (less than 10%) that the averaging has almost no impact on accuracy, but in the other set of curves, which represent a PEP of approximately 98%, the averaging has a much
larger impact. However, we can see that most of the reduced accuracy is due to the grouping of error counts into bins rather than averaging over channel states. Due to the high PEP, the errors in the second set of curves will have only a small impact on the overall performance, as few correct packets will be produced from that data.

When we apply the same technique to PBI channels, we do not see the same performance. Two example distributions are shown in Fig. 4.9 for a PBI channel with SENR = 15 dB, SEIR = −3 dB, and select values of ρ. The negative impact due to the error counts bins is smaller here, due to the lower PEP, but averaging over channel states has a more significant impact. In the curve labeled Avg1, the data used is averaged over a set of values of ρ in the range [0, 1], and there is no significant loss in accuracy due to this averaging. However, when we use data that is averaged over a set of values of ρ in the range [0, 1] and a set of values of SEIR in the range [−6, 0] dB, we see a shift in the curve with higher PEP, as shown in the curve labeled Avg2. The shift shown here is probably not large enough to cause a significant change in adaptive coding protocol performance, but the accuracy is not as good as it was for the AWGN channel. Fig. 4.10 shows a worse situation. As before, the curve labeled Avg1 corresponds to the use of data averaged over ρ only, while the
Figure 4.10: Error count distributions for PBI channel with SENR = 15 dB, \( \rho = 0.5 \), and a code of rate 0.495; averaging techniques are compared.

curve labeled Avg2 is the result of data averaged over \( \rho \) and SEIR in the range \([0, 6]\) dB. For the set of curves corresponding to SEIR = 3 dB, we see a departure from observed results in Avg1 and a significant departure in Avg2. What these results seem to imply is that channel state averaging will only work for certain ranges of channel states, such as AWGN or strong PBI, while some ranges, such as weak PBI, will not work as well. A possible explanation for this behavior is the fixed total power PBI model employed. Since the power in hit dwells decreases as \( \rho \) increases, the average error count is a nonmonotonic function of \( \rho \) for some values of SEIR and is monotonic for others. The result is a relation between the error count and PEP that is not one-to-one; the weighted average is significantly different from correct values for a significant portion of the probability mass.

### 4.2 Iteration Count

#### 4.2.1 Analysis

The iteration count refers to the number of iterations required by an iterative decoder to successfully decode a packet, and its distribution is much harder to characterize than that of
Figure 4.11: Iteration count distributions as a function of error count (EC) for AWGN channel, SENR = 3.5 dB, and $r = 0.325$

the error count. The number of iterations required depends on many factors, such as the decoding algorithm used, the number of code symbol errors, the number of code symbol erasures, etc. Without knowledge of the specific decoding algorithm used and its parameters, the exact distribution cannot be determined analytically. An attempt to describe the distribution of the iteration count for low density parity check (LDPC) codes is made in [13], and the observed distribution is similar.

4.2.2 Modeling

We can infer some relationships from general knowledge of the decoder and from empirical data. For instance, we would expect the iteration count to be correlated with the number of errors and erasures in a packet, and this is borne out in observed data, shown in Fig. 4.11. With stopping conditions of successfully decoding the packet or reaching the maximum number of iterations allowed, we know that the iteration count for an unsuccessful packet will be the maximum number of iterations. The decoder we use is limited to a maximum of 32 iterations to decode a code word. Since the iteration count is deterministic for incorrect packets, we can restrict our attention to the conditional distribution of the iteration count given a correctly decoded packet. Some example
observed distributions are shown in Fig. 4.11 for a code of rate 0.325 in an AWGN channel with SENR = 3.5 dB and an error count bin size of 10. Note the triangular shape that dominates these curves. We exploit this regular shape to create a simple model for generating iteration counts. If we approximate the entire curve with one triangle, we can represent it by storing three points: the iteration counts which represent the beginning, peak, and end of the triangle. The height of the peak can be calculated from the width of the base of the triangle by limiting the total area to a value of one, while the mass values of any other iteration count between the beginning and ending values can be determined by a linear function connecting the peak mass at the peak value and zero mass at either the starting or ending value. We can generate iteration counts using this distribution in multiple ways. One method is the well-known acceptance-rejection method, which uses an easily produced probability distribution and a uniform random process on [0,1] to generate a desired probability distribution. Another method, which we describe here, is to use an interval test. We divide the range [0,1] into several intervals, one for each iteration count in the set of possible values. The length of each interval is the probability mass of the corresponding iteration count. Since the total mass of the distribution is 1, the entire range is covered. To generate an iteration count, we start by drawing a uniform random variable over [0,1]. The iteration count is determined by which
interval the random value occupies. Note that the order of the intervals does not matter, as long as the same ordering is used every time. The following steps illustrate one way this method could be implemented:

1. Draw a continuous random variable $U$ from a uniform distribution over $[0,1]$. Set $i = 1$ as a starting condition.

2. Compute $m = \sum_{x=1}^{i} P(X = x)$. $P(X = x)$ is the probability mass function for the triangular distribution.

3. Compare $U$ and $m$.

   - If $U \leq m$, the randomly drawn iteration count is $i$.
   - If $U > m$, increase $i$ by 1 and repeat step 2.

Fig. 4.12 illustrates the triangle approximation of an observed distribution taken from Fig. 4.11. The triangle is not an exact match for the observed distribution, but it gives a close approximation with low complexity, and it is most accurate for distributions with a low average PEP.
The three points are easily determined from the observed distributions; the starting and ending values are decremented and incremented by one, respectively, to be zero-intercept values. Fig. 4.13 shows the generated iteration count distributions, compared with observed data, in an AWGN channel with SENR = 3.5 dB and a code of rate 0.325. The distributions produced by the triangle method seem to represent a lower peak mass and a larger variance than the observed distributions. The problem appears to be that the linear approximation breaks down for points near the endpoints, and including those values skews the slope of the triangle approximation. We disregard these points in an attempt to gain better accuracy. Since these points constitute only a small probability mass, the lost accuracy due to their exclusion should be outweighed by the increased accuracy gained at more massive points. Using simulation data, we determine where the endpoints should fall for our triangles to be more accurate. From the probability mass at these points, we see that we should disregard values whose mass is more than one order of magnitude less than the peak value. We, therefore, select a cutoff value of 1/20th of the peak probability mass; any smaller value is treated as a zero. The value of 1/20th is chosen to include slightly more than one order of magnitude. Fig. 4.14 demonstrates the performance of the improved triangle method.
Figure 4.15: Comparison of observed and generated iteration count distributions using the triangle and improved triangle methods.

A comparison of the overall iteration count distributions is given for the two generation methods in Fig. 4.15. Figs. 4.14 and 4.15 give results for an AWGN channel with SENR = 3.5 dB and a code of rate 0.325. As shown, the improved triangle method provides a close approximation to the observed distribution.

4.2.3 Data Set Reduction

We also wish to reduce the data required to reproduce the iteration count. Since the iteration count data depends on the error count data, we take a similar approach as with the error count. We use an error count bin size of 10 for consistency with the error count data, but an initial investigation showed that the iteration count was not as sensitive to this grouping as the error count; bin sizes are large as 50 did not adversely affect the accuracy of our generated iteration counts. For the AWGN channel, we average distribution data over SENR values in the range [0, 8] dB. Fig. 4.16 shows the effect of averaging. Again, the approximation is most accurate for channels with a low average PEP.

In Fig. 4.17, we see the effects of averaging in a PBI channel with strong interference. Here we see almost no difference due to averaging. As before, the curve labeled Avg1 corresponds to data.
Figure 4.16: Iteration count distributions for AWGN channel and code of rate 0.325; averaging techniques are compared.

Figure 4.17: Iteration count distributions for PBI channel with $\text{SENR} = 15$ dB, $\text{SEIR} = -3$ dB, and a code of rate 0.495; averaging techniques are compared.
Figure 4.18: Iteration count distributions for PBI channel with SENR = 15 dB, $\rho = 0.5$, and a code of rate 0.660; averaging techniques are compared.

averaged over a set of values of $\rho$ in the range $[0, 1]$, while the curve labeled Avg2 corresponds to data averaged over values of $\rho$ in the range $[0, 1]$ and SEIR in the range $[-6, 0]$ dB. We also examine weak PBI, shown in Fig. 4.18. In this graph, Avg2 represents iteration counts generated from data averaged over values of $\rho$ in the range $[0, 1]$ and values of SEIR in the range $[6, 12]$ dB. As with the error count, the extra averaging introduces greater error into the approximation for weak PBI than for strong PBI. It is possible that the inaccuracy in this case is due to dependence on the error count data, which has been shown to be less accurate for these channels. Note that iteration counts higher than 32 are produced. The number of iterations given is the total number of iterations for a packet, which may contain several blocks. The code of rate 0.660 uses four blocks per packet, resulting in a maximum iteration count of 128. The number of iterations is given on a per packet basis, rather than on a per block basis, because the systems we model operate on a packet-by-packet basis. The total can be divided by the number of blocks to obtain the average iteration count of the packet. The same techniques could easily be applied to produce an iteration count per block, but we do not consider any such cases. As with the error count, there is a tradeoff between data set size and accuracy for generating iteration counts.
Chapter 5

Adaptive Coding Results

To demonstrate the effectiveness of these rapid generation techniques, we use the models to reproduce the performance of the adaptive coding protocol that can make use of the error count and iteration count. The protocol is discussed in further detail in [11] and [12]. We also use the PEPs stored for each error count to randomly determine which packets are successful and which are not, allowing us to reproduce the throughput of the individual codes as well as the adaptive protocol. The protocols make a good performance metric, as incorrect statistics from the receiver can result in a different code being chosen, resulting in either an increase or decrease in throughput. Since the adaptive protocol performs very close to upper bounds in most cases, erroneous receiver statistics would most likely result in reduced throughput. We define throughput as the number of information bits in correctly decoded packets divided by the total number of packet transmission attempts. The unit of throughput is therefore received information bits per packet transmission attempt.

Three example channels are used to demonstrate the statistic generation performance. The first is a channel with only AWGN. When data averaging is applied to this channel, samples are collected for SENR values in the range of $[0,8]$ dB and averaged into one data set. The second channel is an example of strong PBI, for which SENR = 15 dB and SEIR = $-3$ dB. Two levels of data averaging are applied in this channel; the first, $Avg1$, consists of data averaged over values of $\rho$ in the range $[0,1]$, which is the entire range of $\rho$. The second average set, $Avg2$, refers to data from $Avg1$ that is also averaged over SEIR in the range $[-6,0]$ dB. The third channel is an example of weak PBI, with SENR = 15 dB and SEIR = 6 dB. $Avg1$ refers to the same data set as before, but $Avg2$ is changed to be an average of SEIR values in the range $[0,6]$ dB. To provide a
5.1 Error Count

Examining the AWGN channel, we find that the modified binomial distribution produces results nearly identical to simulation in the absence of data averaging. The error count protocol is run with parameters $\xi_0 = 4096$, $\xi_1 = 740$, $\xi_2 = 480$, $\xi_3 = 260$, $\xi_4 = 160$, and $\xi_5 = 0$. Fig. 5.1 compares the generated results with simulation results for both static codes and the error count protocol. We next turn our attention to the effects of grouping the error counts into bins, shown in Fig. 5.2. Interestingly, the error count protocol is insensitive to the bin sizes tested, even sizes as large as 50, which produced a very poor reproduction of the actual error count distribution.

The error count protocol is only slightly more sensitive, if at all, to the effects of channel state data averaging. Fig. 5.3 shows the performance of the error count protocol using generated statistics. The differences between the two curves from generated statistics are negligible; the simulation is able
Figure 5.2: Throughput comparison of generated error count results for different bin sizes in AWGN channel.

Figure 5.3: Throughput comparison of generated error count results for AWGN channel.
Figure 5.4: Throughput comparison of generated error count results for strong PBI channel with SENR = 15 dB and SEIR = −3 dB.

to accurately reproduce the protocol performance using receiver statistics generated from a reduced data set.

Results for strong and weak PBI channels are given in Figs. 5.4 and 5.5, respectively. In Fig. 5.4, we see that the generated statistics reproduce the actual protocol performance nearly perfectly, even when using averaged data. There is a slight departure from the correct results for the curves generated from averaged data in Fig. 5.5. The departure is consistent with the inaccuracy of generated error count distributions produced from averaged data in channels with weak PBI, though the impact here is small.

5.2 Iteration Count

The previously defined parameters for the iteration count protocol are $v_1 = 3.5$, $v_2 = 3$, $v_3 = 3$, $v_4 = 1.5$, $v_5 = 0$, $\delta_1 = 32$, $\delta_2 = 13$, $\delta_3 = 13$, $\delta_4 = 9$, and $\delta_5 = 9$. While the iteration count for a code block will always be an integer, the protocol uses the average iteration count for packets containing multiple blocks, which allows for some packets to have a fractional iteration count. The iteration counts produced by the improved triangle method are divided by the number of blocks per
Figure 5.5: Throughput comparison of generated error count results for weak PBI channel with SENR = 15 dB and SEIR = 6 dB.

Figure 5.6: Throughput comparison of generated iteration count results for AWGN channel.
Figure 5.7: Throughput comparison of generated iteration count results for strong PBI channel with SENR = 15 dB and SEIR = −3 dB.

Results for the AWGN channel are given in Fig. 5.6, showing that the improved triangle method can reproduce the performance of the iteration count protocol, losing only a little accuracy due to averaged data. The generated statistic performs equally well in strong PBI channels, shown in Fig. 5.7. The performance of the protocol using generated statistics closely mimics the performance of the protocol using decoder simulations, regardless of averaging.

Weak PBI channels present the same difficulties in generating the iteration count. Fig. 5.8 shows significant divergence from the expected result when averaged data is used. However, the adaptation protocol using generated statistics performs nearly identically to simulation results when channel-specific (i.e. not averaged) data is used.

5.3 EI Combination

The error count and iteration count can also be used in combination to better adapt the code rate as the channel changes; we refer to this combination as the EI combination protocol. The EI combination protocol uses the same parameters as the individual statistic protocols, with the
Figure 5.8: Throughput comparison of generated iteration count results for weak PBI channel with SENR = 15 dB and SEIR = 6 dB.

Figure 5.9: Throughput comparison of generated error and iteration count results for AWGN channel.
exception that $\xi_3 = 250$. The results are similar to the individual statistic protocols. The adaptive protocol using generated statistics gives the same performance as the protocol using statistics derived from receiver simulations. Results for the AWGN channel are given in Fig. 5.9, showing little adverse effect from the data averaging. Strong and weak PBI results are given in Figs. 5.10 & 5.11, and the results are similar to the individual statistic results, as expected. Data averaging works fine for strong PBI, but not for weak PBI.

5.4 Dynamic Channels

We also examine the performance of our generation techniques for dynamic channels. The first such example is an AWGN channel with some nominal SENR and some excess path loss, governed by a six-state Markov chain. State zero represents no additional path loss, while each state beyond zero represents an additional 2 dB of path loss. The transition probability is $p = 0.1$. Performance results for the three protocols are given in Fig. 5.12, which shows a nearly exact match between the simulated protocol performance and generated statistic protocol performance. Additionally, some of the differences between the curves can be attributed to simulation error, which
Figure 5.11: Throughput comparison of generated error and iteration count results for weak PBI channel with SENR = 15 dB and SEIR = 6 dB.

Figure 5.12: Adaptive Protocol Performance from simulated and generated statistics for AWGN channel with dynamic excess path loss governed by six state Markov chain with $\Delta = 2$ dB and $p = 0.1$. 
Figure 5.13: Adaptive Protocol Performance from simulated and generated statistics for PBI channel with SENR = 15 dB and dynamic \( \rho \) governed by three state Markov chain with states: \( \rho_0 = 0, \rho_1 = 0.25, \rho_2 = 0.5 \).

<table>
<thead>
<tr>
<th></th>
<th>Receiver Simulation</th>
<th>Statistic Generation</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>10K packets</td>
<td>min 31s</td>
<td>1.6s</td>
<td>19 times</td>
</tr>
<tr>
<td></td>
<td>max 266s</td>
<td>1.9s</td>
<td>140 times</td>
</tr>
<tr>
<td>100K packets</td>
<td>min 294s</td>
<td>16.0s</td>
<td>18 times</td>
</tr>
<tr>
<td></td>
<td>max 2625s</td>
<td>18.5s</td>
<td>142 times</td>
</tr>
<tr>
<td>1M packets</td>
<td>min 3046s</td>
<td>160.0s</td>
<td>19 times</td>
</tr>
<tr>
<td></td>
<td>max 26635s</td>
<td>185.3s</td>
<td>144 times</td>
</tr>
</tbody>
</table>

Table 5.1: Run time comparison for receiver simulation and statistic generation

becomes more prevalent for Markov chain simulations, as more simulation time is needed to reach a steady state. The second dynamic example is shown in Fig. 5.13, in which the value of \( \rho \) for the PBI is governed by a three-state Markov chain, whose states correspond to \( \rho = 0, \rho = 0.25, \) and \( \rho = 0.5 \). The transition probability of this chain is also \( p = 0.1 \). All three protocols are able to achieve approximately the same performance using generated statistics as using receiver simulations.
5.5 Run Time Comparisons

The primary benefit of using these models is a significant reduction in computation time required to simulate packet transmissions, allowing longer and more complex network-level simulations to be run in reasonable amounts of time. Some sample run time results are given in Table 5.1. The actual run time of a decoder simulation depends on the number of iterations: a high average number of iterations takes longer to run than a low average number. The decoding time also varies by code, with the codes of rates 0.660 and 0.793 requiring less time than the codes of rates 0.236, 0.325, and 0.495. The minimum and maximum times given in Table 5.1 are over all codes. The time required to directly generate the receiver statistics from random variables is fairly constant. From the data in the table, we see a speedup between one and two orders of magnitude. The actual value will depend on the packet error probabilities of the simulated channels, as packet errors require 32 iterations, lengthening the decoder simulation time considerably. The run times given in Table 5.1 show the potential benefit of replacing real-time simulation of the receiver and the decoder in a network-level simulation with direct generation methods. Using directly generated receiver statistics, a network-level simulation can be run in up to two orders of magnitude less time. By reducing the time required to simulate a packet, we increase the number of packets that can be simulated within a particular amount of time and thereby increase the overall complexity of the networks we can simulate.

5.6 Data Averaging

The largest potential downside of these models is that they require data derived from off-line channel simulation. The time savings is still large, as each channel need only be simulated a set amount, rather than each time a channel is used, but it requires that all potential channel conditions be known and simulated in advance. As previously discussed, this problem can be reduced or eliminated by data averaging. If it can be determined without simulation, a representative subset of channels can be simulated for off-line data gathering, and the average result of that data can be used to generate statistics for any channel within the range. If a representative subset cannot be determined easily, then all channel states can be simulated and the data averaged. The primary and desired result is a smaller data set. There is a potential side benefit of having a reduced amount of off-line simulation required for data gathering. However, as this side benefit is not related to the
primary goal of reducing on-line simulation time, we do not pursue it further.
Chapter 6

Conclusion

We have shown methods of generating both the error count and iteration count, two important receiver statistics, without the need for real-time simulation of the operation of the receiver. Using the techniques outlined here, we can generate statistics with the correct distribution in a fraction of the time required for simulation of the receiver and the decoder. The methods we outline have the downside of requiring a large amount of data which must be gathered from off-line simulations of the receiver and decoder. We have shown how to reduce the amount of data which must be stored while having a minimal impact on accuracy for many channels. For channels where the impact is significant, such as weak PBI, a tradeoff exists between accuracy and data set size.

The largest benefit of these models is the speedup. We have shown that our models reduce the simulation time by up to two orders of magnitude, allowing for an increase in network complexity by an equal factor without increasing simulation time. This speedup is both useful and important for the investigation of large networks with cross-layer protocols that exploit receiver statistics.
Bibliography


