Technical University Berlin

Telecommunication Networks Group

Results of Bit Error Rate Measurements with an IEEE 802.11 compliant PHY

Andreas Willig, Martin Kubisch
Adam Wolisz

{willig,kubisch,wolisz}@ft.ee.tu-berlin.de

Berlin, September 2000

TKN Technical Report TKN-00-008

TKN Technical Reports Series
Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

For designing channel coding schemes, MAC- and data link protocols and for realistic packet level simulations of wireless protocols some knowledge about the channel error behavior is useful. In this report we describe the measurement setup and evaluation methodology that we have developed for performing measurements of bit error behavior. Furthermore we describe different approaches to stochastic modeling of the observed error processes. A major contribution of this report and the companion reports are the results of two measurement campaigns taken in an industrial environment and what they mean for developing stochastic channel models suitable for simulations.

Contents

Chapter 1

Introduction

When designing MAC protocols and framing methods for wireless LANs, it is of vital interest to have insights into the stochastic behaviour of the bit errors occuring on the medium. This knowledge is important for at least three purposes:

- To assess which kinds of error correcting codes are admissible.
- To build accurate and efficient stochastic channel error models as important part of packet level simulations of MAC protocols.
- As input and design constraints for MAC protocols.

It is commonly believed that the wireless channel is a bad one with nonstationary error characteristics and bursty errors. This belief is reflected in the fact that for modeling the wireless channel many researchers resort to simple channel models exhibiting some short range correlation, e.g. the Gilbert-Elliot channel model or models based on hidden markov chains. However, it is hard to find publications where the corresponding model parameters are obtained from real measurements instead of coming from a more or less “educated guess”.

This report and the companion reports [17], [18], [16] describe in detail some bit error rate measurements made in different campaigns and in different environments. Since this work is part of an effort in creating a MAC- and data link layer protocol especially for wireless industrial LANs, considerable effort has been spent to obtain results for appropriate scenarios. A second objective of this report is to determine suitable, measurement based parameters of stochastic channel models usable for packet level simulations.

The remainder of this report is structured as follows: in chapter 2 we describe our basic measurement setup and methodology including some remarks on how the traces are evaluated and the common stochastic models are parameterized, while in chapter 3 the results of the first measurement campaign are described. This campaign was important for evaluating our measurement approach and the setup. The results for the second campaign are described in the companion reports [17], [18], [16]. Our conclusions about the overall project can also be found in [16].

A more thorough description of the stochastic error models and other measurement campaigns can be found in an upcoming report.

Chapter 2

Measurement and Evaluation Methodology

In this chapter we describe our measurement methodology, consisting of the measurement setup and the evaluation methods selected, and furthermore different approaches for stochastic modeling of single traces.

2.1 Measurement Setup

For our measurements we use two stations, a transmitter station and a receiver station, which do not change their roles during a measurement. The basic idea is that the transmitter sends a well-known bit stream over the wireless link, which is captured and stored by the receiver. It is important to note that there is no MAC protocol implemented, nor are there any higher layer protocols involved, which can bias the measurement results. The setup is schematically shown in figure 2.1.

The wireless network interface card (Wireless NIC) is a PCI microcontroller board (Motorola PowerQUICC [9] with Tundra PCI Interface [12] with a core of a PowerPC 603e processor with 50 MHz, 32 MBytes of memory and different devices especially suited for communication purposes, e.g. timers, serial controllers and DMA devices. The coupling to the (Windows NT based) host PC is achieved via a 64 KBytes memory segment located on the microcontroller card, which is via the PCI bus mapped into the host memory and can be "directly" accessed. For signalling purposes interrupts are used. This 64 KByte memory area is denoted as *host interface* and mainly used for passing commands, acknowledgements, status and statistical information and received packets.

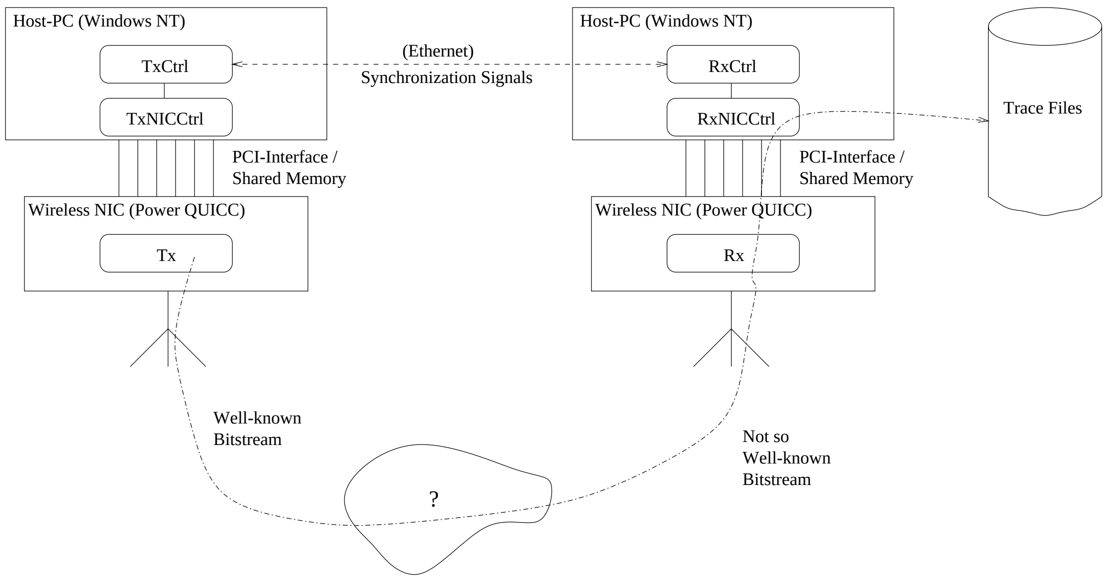


Figure 2.1: Measurement Setup

The wireless NIC carries a MAC-less radiomodem (Harris/Intersil PRISM-I chipset [1]), which is compliant to the IEEE 802.11 DSSS PHY using the 2.4 GHz ISM band and offers four different bit rates and four different modulation types (1 MBit DBPSK, 2 MBit QPSK, 5.5 and 11 MBit CCK, 5.5 and 11 MBit QMBOK). The radiomodem basically comprises of HF circuitry and a baseband processor, which accepts and delivers a serial bitstream from upper layers, performs scrambling and direct sequence processing (chip generation) as well as generation and reception of PHY packets [5]. The PHY packet format is shown in figure 2.2. A packet starts with a preamble (of configurable length), followed by a constant value indicating start of a packet (start frame delimiter, SFD). The signal field indicates the bitrate/modulation type used in the data portion of the packet[1], while the length field indicates the length of the data portion in microseconds (the service field has no significance). The CRC field spans over the three previous values. If the CRC is wrong, the whole packet is discarded by the baseband processor. The PowerPC processor is signalled with this event, however, no further evaluation takes place in the current software version.

In the following we shortly discuss the software modules indicated in figure 2.1.

[1]Please note that while the data part can use different bitrates / modulation types, the PHY header is always transmitted with 1 MBit (D)BPSK modulation. When the data part uses another modulation, both transmitter and receiver must switch the modulation type within a PHY packet, more precisely, after the CRC16 checksum

Preamble	SFD	Signal	Service	Length	CRC16	Data = {chunk}

Figure 2.2: Format of a PHY frame

2.1.1 Tx

The Tx module is located on the wireless NIC on the transmitting computer. Its main responsibilities are to accept control and configuration commands from the TxNICCtrl module (located on the host) via the host interface, and to generate a well known packet stream. The command interface allows the TxNICCtrl module to start and stop the generation of packets, to set transmission parameters (frequency, diversity enable, number of packets to send, and so forth) and to obtain some statistics. The set of adjustable radio parameters is shown in table 2.1.

The main task of Tx, however, is the generation of the *packet stream.* A packet stream consists of a given number of *packets*, which are transmitted at equidistant start times[2]. The necessity for generating packets instead of a single long stream of bits stems from two sources: a packet based transmission allows for re-synchronization (bit synchronization) at the beginning of the next packet, even when synchronization is lost during the preceding packet, and furthermore the PHY only allows for streams of maximum length, due to the bounded size of the length field in the PHY header (which in turn is used by the receiver to determine the end of the frame).

The packet format should allow for unique identification of erroneous bits (thus it must be well known) and for determining the actually expected packet even when all the last received packets show errors. Thus it should contain a high level of redundancy in order to have a good chance of telling for each received packet which packet it could be. The Tx software contains a 32 bit *sequence number*, which is reset to zero at the beginning of a packet stream. A sequence number is coded into a *chunk* and then it is incremented. A chunk has the following structure:

- It starts with two bytes of value 255 (`0xffff`), this is called *anchor*

[2]Since no carrier sensing is carried out, this approach is not friendly to 802.11 networks running on the same frequency, since 802.11 packets are destroyed and the medium is often busy. To point it out clearly: the Tx module contains no MAC functionality!

Parameter	**Description**
ScramblingEnabled	The baseband processor uses a shift register with programmable feedback interconnections, to pass the user data through, and transmits the output of that register. If *ScramblingEnabled* is true, a scrambler tap of `0x48` is used (and thus data is scrambled), otherwise a scrambler tap of `0x00` is used, and the data is transmitted as is.
DiversityEnabled	Determines whether the receiver uses antenna diversity (i.e. selects the antenna with maximum signal level)
Frequency	this number (1 to 12) determines which of the carrier frequencies in 802.11 to use
PreambleLength	determines the number of bits used in the PHY preamble
SFD Threshold	determines how many bit times the baseband processor waits for an SFD after acquiring bit synchronization before giving up
ModulationCode	distinguishes modulation used for data portion: 1 MBit DBPSK, 2 MBit QPSK, 5.5 MBit CCK, 5.5 MBit QMBOK, 11 MBit CCK, 11 MBit QMBOK

Table 2.1: Adjustable radio parameters

Parameter	**Description**
NumPackets	Number of Packets within a trace
GapTime	Time gap between two packets in a trace, given in microseconds (Medium Idle Time)
NumChunks	Number of chunks per packet
CRCUsageEnabled	Determines whether the CRC at the end of each chunk is used or not

Table 2.2: Adjustable trace parameters

- The 32 bit sequence number is 1b8b coded, using the following mapping:

$$\begin{aligned} 0 &\mapsto 00000000 \\ 1 &\mapsto 00111100 \end{aligned}$$

  corresponding to the byte values `0x00` and `0x3c`.

- The last two bytes can be used as a CRC checksum over the whole chunk, allowing for reliable assessment of whether a chunk is correct. However, checksum usage is disabled and zero bytes are transmitted. The reason for this is that it is in case of errors hard to tell whether some bits in the middle of the chunk or within the checksum bytes are wrong.

A chunk thus consists of 36 bytes (*ChunkSize*). A packet now consists of an integral number of chunks.

For our measurements we define a *trace* to be a stream of a finite number of packets, where all parameters are fixed throughout the trace (thus each packet has the same length, the same interpacket time and so forth). The parameters describing a trace are summarized in table 2.2.

Basically the Tx module consists of two parts: in a main loop the packets are computed and stored in a FIFO, while in a periodic timer interrrupt the corresponding interrupt routine gets the first packet out of the FIFO and starts its transfer:

- The baseband processor is signaled to start transmitting, the packet parameters (length, modulation code) are written to baseband processor configuration registers

- A serial controller (SCC, part of the PowerQUICC) is set up to transfer the packets data in a bit serial fashion from the PowerPC's memory to the baseband processor.

- The packet is removed from the FIFO.

The timer period is the *InterpacketTime*, given by

$$InterpacketTime = GapTime + (PreambleLen + 64) + \frac{NumChunks \times 36 \times 8}{BitsPerMicrosecond} \tag{2.1}$$

in microseconds. Since the Preamble and all header information (64 bits) are transmitted with 1 MBit/sec DBPSK modulation, their direct length in bits are counted in the given expression.

Unfortunately the PowerPC processor and the SCC compete for the system bus. If *NumPackets* is large or *GapTime* is small it may happen that the main loop is not fast enough to compute new packets, such that the interrupt routine occasionally finds an empty FIFO. It is easy to determine how often this happens, since for every successfully transmitted packet a counter `txpackets` is incremented, which can, after finishing a trace, be compared with *NumPackets*. However, for $NumPackets = 20000$ and 600 μsec gaptime no losses are observed.

2.1.2 Rx

The Rx module is located on the wireless NIC. Its main responsibilities are to accept control and configuration commands from the RxNICCtrl module, to capture all packets from the wireless link, to deliver the packets including metainformation via the host interface to the RxNICCtrl module and to generate various statistics. The command interface allows the RxNICCtrl module to start and stop the reception of packets (when starting several statistical counters and the internal clock are resetted), to set modem parameters (see table 2.1) and to obtain statistics.

The Rx module generates the following metainformation for every received packet:

- a timestamp (indicating the time of finishing packet reception)
- an RSSI value (Received Signal Strength Indicator, taken at the beginning of a packet).
- a counter value (incremented by Rx for every successfully received and buffered packet)
- the value of the service field in the PHY header.
- the length field `t_length` (in μsec of the PHY header)
- the corresponding number of bits in the packet `bd_length`.

- the `rx_status` indicates whether a received packet has a correct header and furthermore transmission speed with which the packets data is transmitted.
- the `bd_status` indicates, whether the received bit stream could be successfully transferred into the buffer provided by the user.
- the `signal` field is of no further interest.

In the remainder of this section we describe the statistics gathered by Rx while receiving packets.

Before receiving a packet, the Rx software allocates an internal receive buffer (rcvbuf). The packets data is transferred via the SCC from the baseband processor into that buffer. After finishing the packet, the above mentioned metainformation is added. If none such buffer is available, the new packet is written into the last successfully allocated buffer, thus an old packet may be overwritten. Due to an programming error this has happened silently during the first measurement campaign, in further versions a new counter `rcvfrm_dropped` was introduced. This counter is incremented every time no new rcvbuf could be allocated. Furthermore it may happen, that packet reception is truncated e.g. due to loss of bit synchronization. In this case the rcvbuf is only partially overwritten and may contain parts of an old packet. In the first version of our software this was not detected, in further versions this can be checked with the `bd_length` field.

Immediately after successfully receiving a packet a pointer to the corresponding rcvbuf is stored into a list (which is called `recvlist`), however, this does not happen when rcvbuf allocation has failed. If the allocation of a new list element for the `recvlist` failed (the new element is tried to allocate from a special free list `rxfreelist`), the counter `rcvbuf_dropped` is incremented. Otherwise the counter `rxpackets` is incremented.

A separate process within the Rx software is responsible for transferring the received packets from the `recvlist` to the host, via the host interface. After packet transfer the elements from *recvlist* are transferred back into the `rxfreelist`. From this we can conclude that the host must fetch the packets fast enough from the host interface in order to avoid incrementing `rcvbuf_dropped`.

In an older version of the Rx software another counter `hostbuf_dropped` is incremented, whenever a packet to be transferred from the *recvlist* to the host via the host interface is too long for a single buffer in the host interface (the host interface in this respect is organized as a cyclic buffer with fixed element / buffer size). This behaviour was observed sometimes, and we attribute it to incorrectly transferred length fields from the baseband processor to the Rx software.

In our implementation the Tx module and the Rx module are merged in a single program running on the PowerPC.

2.1.3 TxNICCtrl

The TxNICCtrl module is merely a wrapper, which offers to Windows NT users a command line interface to the capabilities of the Tx module, i.e. it allows to set transmission parameters, to start and stop packet generation and so forth. Internally it is a Windows command line application, using a driver, which in turn accesses the host interface and exchanges commands and data with the Tx module.

2.1.4 RxNICCtrl

The RxNICCtrl module serves also as a wrapper, offering a commandline interface to the capabilities of the Rx module (setting parameters, starting and stopping packet reception). However, an important additional functionality is to store all packets, which are received from the Rx module via the host interface into a logfile. The packets data are dumped into the file in a binary format. The format is as follows:

$$\begin{aligned} logfile &::= \{ \quad packet \quad | \quad text \quad \} \\ packet &::= \ 'p' \quad header \quad \{byte\}^{header.len} \\ text &::= \ 't' \quad < string >' 0' \end{aligned}$$

The header contains metainformation about the packet (see section 2.1.2). Thus a logfile can contain both binary packet entries and text entries (genererated by the RxNICCtrl software).

The behaviour of the RxNICCtrl software when receiving packets is as follows: after starting the packet reception the software waits indefinitely for the first packet coming from the host interface (however, console input aborts reception). Then, after every subsequent packet from the host interface a timer is set to a specified timeout value. If the timer expires packet logging is stopped.

We have observed that the RxNICCtrl software should run on a computer with a SCSI harddisk instead of an EIDE harddisk, otherwise there occur packet losses (indicated via the counter `rcvbuf_dropped`). This happens even for comparably low data rates (QPSK modulation). Furthermore in a previous version of our RxNICCtrl software the packets are transformed into an ASCII representation before they are dumped into the logfile. This also results in packet losses.

The question of how to set the timeout value is difficult to answer when the measurement setup is operated in proximity of a running IEEE 802.11 infrastructure wireless LAN, since usually an 802.11 base station sends a beacon packet every second. Thus a timeout setting smaller than one second would be useful. However, unfortunately it is not possible to achieve a tight coupling between start of logging (at the Rx computer) and start of the packet stream (at the Tx computer), since the necessary synchronization signals may experience unpredictable delays. This requires for the Tx computer to wait a little while (it is necessary for the Rx computer to first send the synchronization signal before starting to log packets, since the Tx computer may not be ready, when the Rx computer wants to synchronize). Before the Tx computer actually starts transmitting, the Rx computer may be in listen mode for some unknown time. If the timeout is smaller than one second, it was often observed, that the receiver returns after logging a single beacon. A possible solution of this problem is to start the timer the first time, when two or more packets are received with small timestamp differences (below 100 msec). In all campaigns where no 802.11 WLANs were present, we have set the timeout value to 10 seconds.

In our implementation the RxNICCtrl module and the TxNICCtrl module are merged in a single program.

2.1.5 TxCtrl

The TxCtrl software is a short script, which synchronizes itself with the RxCtrl software for controlling the measurements. For this synchronization a TCP connection over the Ethernet cable is used (see figure 2.1). The TxCtrl software performs the following steps in an infinite loop:

1. Send a connect request to the RxCtrl software over a socket.
2. If the connect request is rejected go back to the first step.
3. Read a string containing the measurement parameters from the socket.
4. Set parameters of the Tx process accordingly (using the TxNICCtrl module). Calculate the time needed for the whole trace (sleeptime).
5. Wait a second, then start transmitting by triggering the TxNICCtrl module appropriately.
6. sleep for the sleeptime.
7. Go back to the first step.

2.1.6 RxCtrl

The RxCtrl software is a short script, which is actually controlling a whole measurement campaign. It loops over all variable parameters, for each parameter set a trace is started (by triggering the TxCtrl software) and logged onto the harddisk. For every point in the product space of the variable parameters the following steps are performed:

1. Code all parameters (variable and fixed ones) into a string (parameter string).
2. Open a socket, accept connection requests from TxCtrl on this socket.
3. On getting a connection request, send the parameter string back to the caller over the socket, then close the socket.
4. Set parameters of the Rx module according to the current parameter set (employing the RxNICCtrl module).
5. Start receiving packets by telling the RxNICCtrl module to do so.
6. After the RxNICCtrl module returns, a trace was recorded. Compress the logfile, determine the next set of parameters and go back to the first step.

2.2 Trace Evaluation and Stochastic Modeling

In this section we describe approaches for constructing error models for the successfully received packets of a *single* trace. It must be clear that this covers only a part of a realistic channel model. Other aspects of such models are discussed later. We describe the methodology for evaluating a single trace and for parameterizing diverse stochastic models from these results. It must be noted that all parameters given in 2.1 and 2.2 have to be known before trace evaluation starts. Besides some commonly used stochastic models, we introduce a special class of markov models, called "bipartite models", to our knowledge not yet covered in the literature. This name stems from the fact that the corresponding markov chain forma a bipartite graph.

2.2.1 Preprocessing

The preprocessing step tries to remove "strange" packets from further evaluation and generates the basic input information for the subsequent steps.

Filtering Missized Packets

The first step in trace evaluation is to filter out missized packets. Filtering out missized packets is straightforward, since the Rx module passes with every packet its size via the host interface. If the packets size is not equal to *ChunkSize* times *NumChunks* the packet is discarded. The number of oversized packets is counted in the *oversized packets* statistics, the number of undersized packets is counted in the *truncated packets* statistics. The rationale for simply filtering out packets instead of further investigating them is as follows:

- In most cases, missized packet rates are low, thus the resulting bias can be neglected.
- When the measurement setup is operated in proximity of an 802.11 WLAN, the Rx software sometimes captures 802.11 packets (e.g. beacons, data packets), which should be removed from the trace. Typically beacons are shorter than the measurement packets, data packets have different sizes.
- A trace has too much packets and there are too much traces to closely inspect every missized packet.

However, in this way we loose information about "real" packet truncations, where reception of measurement packets is aborted, e.g. due to loss of bit synchronization. But if we can process x packets successfully from a trace (this value is given by the `packetsprocessed` statistics), then trivially we have at most *NumPackets* minus `packetsprocessed` truncated packets.

Identification of Lost Packets

The identification of lost packets is based on the packet timestamps and their comparison with the *InterpacketTime* as computed in equation 2.1. If t_1 and t_2 are timestamps of subsequently dumped packets, we calculate

$$g = round\left(\frac{t_2 - t_1}{InterpacketTime}\right)$$

The number of lost packets between t_1 and t_2 is then simply given by $g - 1$. We think that the main reason for loosing packets is the following: the baseband processor requires for each packet that: a) it acquires correctly bit synchronisation during the preamble and detects a proper start frame delimiter (SFD) and b) the header checksum must be correct. If both is given, the baseband processor trusts all the header fields (especially the packet length and the modulation code) and starts to receiving the data. If any of these two fails, the baseband processor does nothing, until the medium goes idle. In this case a packet can be lost. This is not accounted for in our statistics.

Generating the Indicator Sequence

As noted in 2.1.1 the rationale for the choice of the packet stream format was to support unique identification of the received packet within the trace and thus to deduce what its contents should be.

The received packet number ν' of n bits length is represented by the bit sequence $r_1^\nu r_2^\nu \dots r_n^\nu$, the corresponding expected packet ν is represented as $e_1^\nu e_2^\nu \dots e_n^\nu$. The *packet indicator sequence* of packet ν is then defined as $i_1^\nu i_2^\nu \dots i_n^\nu$ with

$$i_l^\nu = r_l^\nu \,\mathrm{XOR}\, e_l^\nu$$

We define the *trace indicator sequence* or simply *indicator sequence* $i_1 i_2 \dots i_m$ to be the concatenation of all packet indicator sequences for all received measurement packets of a single trace (in order of increasing ν). The mean bit error rate of a single trace can then be simply computed as follows:

$$E_k = \frac{\sum_{\nu=1}^{m} i_\nu}{m}$$

We can view the indicator sequence as finite subset of a sample path of a random process $\{B_n\}_{n\in\mathbb{N}}$ where each B_i is a Bernouilli random variable.

The rules for determining which packet is the expected one are simple:

- In general, a chunk is considered correct, when at the corresponding positions the correct flagbytes `0xff` are found, when the checksum bytes are both `0x00` and when all bytes in between have one of the values `0x3c` or `0x00`.
- First we look for the first chunk that looks correct and synchronize on the corresponding sequence number (i.e. we store the sequence number in a variable `seqno`). The rationale for this is that we cannot know how many packets from the start of the trace are lost. Then `seqno`+1 is the next expected sequence number.
- If we detect a lost packet (recognized by comparing the timestamps, see above) we increment `seqno` by the number of chunks in a packet.
- If we read a correct chunk with a sequence number greater than `seqno`+1 we have lost a packet without recognizing an irregular timestamp. We set `seqno` to the sequence number actually read.

- If we read a correct chunk with a sequence number smaller than `seqno`+1 we are irritated, since this should not happen.
- If we read a correct chunk with a sequence number equal to `seqno`+1, we are happy and increment `seqno`.
- If we read an incorrect chunk, we compare it bit for bit to `seqno`+1, then `seqno` is incremented.

Please note that in the indicator sequence any information about packet boundaries, lost packets or packet gap times is completely ignored. However, this sequence is the input for any error detecting or error correcting algorithm and, in the second line, to any MAC scheme. For MAC schemes additionally information about lost packets must be considered.

An interesting intermediate statistic is the fraction of wrong bits in a packet, which can be easily computed by simply counting the ones in a packet indicator sequence.

Packet Filtering Heuristics based on Packet Indicator Sequences

During evaluation of the first set of measurements there occured some packets with an exceedingly high fraction of wrong bits. A closer inspection of these packets showed the following phenomenon: it appears that somewhere in the middle of a packet a few single bits are simply not delivered, and the remaining packet is a "shifted" version of the original packet. A sample packet where this phenomenon occurs is shown below[3]:

```
<H>signal: 0xff0107e0
<H>bd_length: 2016
<H>bd_status: 0x1c10
<H>rx_status: 0x7c
<H>t_length: 8064
<H>service: 0x0
<H>rssi: 0
<H>time: 4392s:951321us
<H>pktcnt: 408
<D>ff ff  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 3c  0 3c 3c  0 3c  0  0 3c  0 3c 3c  0  0  0  0  0
<D>ff ff  0  0  0 80 48  0  0  0  0  0  0  0  0  0  0  0  0 3c  0 3c  c  0 1e  0  0 1e  0 1e 1e  0  0 1e  0 80
<D>ff 7f  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 1e  0 1e 1e  0 1e  0  0 1e  0 1e 1e  0 1e  0  0 80
<D>ff 7f  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 1e  0 1e 1e  0 1e  0  0 1e  0 1e 1e  0 1e 1e  0 80
<D>ff 7f  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 1e  0 1e 1e  0 1e  0  0 1e  0 1e 1e 1e  0  0  0 80
<D>ff 7f  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 1e  0 1e 1e  0 1e  0  0 1e  0 1e 1e 1e  0 1e  0 80
<D>ff 7f  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 1e  0 1e 1e  0 1e  0  0 1e  0 1e 1e 1e 1e  0  0 80
<D>ff 7f  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 1e  0 1e 1e  0 1e  0  0 1e  0 1e 1e 1e 1e 1e  0 80
<D>ff 7f  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 1e  0 1e 1e  0 1e  0  0 1e 1e  0  0  0  0  0  0 80
<D>ff 7f  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 1e  0 1e 1e  0 1e  0  0 1e 1e  0  0  0  0 1e  0 80
```

However, unfortunately it is not so easy to devise a set of rules for detecting "shifted" packets. There are at least two approaches. During our inspections of the first campaign we

[3] All lines starting with `<D>` contain a chunk, all other lines show meta information.

have found that the best indicator seems to be the fraction of erroneous bits per packet and to look for packets with a comparably high fraction. However, we have the problem, that the 1b8b mapping described in section 2.1.1 tends to produce much more `0x00` bytes than `0x3c` bytes at the beginning of a trace, which makes it hard to detect shifts and tends to decrease the fraction of erroneous bits produced by the shifting (The mapping was changed for the second campaign). Another approach was to use character statistics for a single packet, however, this is not yet explored.

2.2.2 Indicator Sequence Evaluation

The basic idea is to subdivide the indicator sequence into *error bursts* and *errorfree bursts*, where intuitively an error burst is a subsequence where many errors occur, while errorfree bursts show a long sequence of zeros in the indicator sequence. More formally we define an *error burst* as follows:

Definition. *An error sequence of length n and of order k_0 is a subsequence $i_l i_{l+1} \dots i_{l+n-1}$ of the indicator sequence $i_1 i_2 \dots i_m$ such that:*

1. $i_l = 1,\ i_{l+n-1} = 1$
2. *for $s, t \in \{l, \dots, l+n-1\}$ with $s < t$ we have the implication:*
$$i_s = 1,\ i_t = 1,\ \forall r \in \{s+1, \dots, t-1\} : i_r = 0 \qquad \Longrightarrow \qquad d(i_s, i_t) \le k_0$$
3. $i_n = 0,\ i_{n+1} = 0\ \dots i_{n+k_0} = 0$
4. $i_{l-1} = 0,\ i_{l-2} = 0,\ \dots i_{l-(k_0+1)} = 0$

where $d(i_s, i_t) := |t - s|$ denotes the distance of i_s and i_t in the indicator sequence.

In this definition an error bursts consists of all subsequences, where between two errors at most k_0 other bits are (regardless of whether they are correct or not) and where all errorfree bursts have a length of at least $k_0 + 1$.

Using this definition the indicator sequence is segmented into alternating error bursts and errorfree bursts. The indicator sequence is transformed into a sequence

$$X_1 Y_1 Z_1 X_2 Y_2 Z_2 \dots X_p Y_p Z_p$$

where X_n is the length of the n-th errorfree burst, Y_n is the length of the n-th error burst and Z_n is the number of erroneous bits within the n-th error burst[4]. Using this sequence we can immediately calculate the following simple statistics:

1. Mean bit error rate:
$$\bar{e} = \frac{\sum_{i=1}^{p} Z_i}{\sum_{i=1}^{p}(X_i + Y_i)}$$
2. Mean error burst length (sample mean):
$$\bar{Y} = \frac{1}{p}\sum_{j=1}^{p} Y_j$$
3. Variance of error burst length (sample variance):
$$\sigma_Y^2 = \frac{1}{p-1}\sum_{j=1}^{p}(Y_j - \bar{Y})^2$$
4. Coefficient of Variation of error burst length:
$$C_Y = \frac{\sqrt{\sigma_Y^2}}{\bar{Y}}$$
5. Mean errorfree burst length (sample mean):
$$\bar{X} = \frac{1}{p}\sum_{j=1}^{p} X_j$$
6. Variance of errorfree burst length (sample variance):
$$\sigma_X^2 = \frac{1}{p-1}\sum_{j=1}^{p}(X_j - \bar{X})^2$$
7. Coefficient of Variation of errorfree burst length:
$$C_X = \frac{\sqrt{\sigma_X^2}}{\bar{X}}$$

Of some interest are also the following conditional probabilities, since these give insigths into the error burstiness:

[4]In our discussion we neglect the fact that an indicator sequence may start with an error burst or may end with an errorfree burst. Furthermore we do not indicate explicitly the dependence on k_0 in the notation.

1. Error correlation: $\Pr[i_{n+s} = 1 | i_n = 1]$ for $(s \geq 1)$. This is approximated as follows:

$$\Pr[i_{n+s} = 1 | i_n = 1] \approx \frac{\sum_{j=1}^{m-s} i_j \cdot i_{j+s}}{\sum_{j=1}^{p} Z_j}$$

2. Errorfree correlation: $\Pr[i_{n+s} = 0 | i_n = 0]$ for $(s \geq 1)$. This is approximated as follows:

$$\Pr[i_{n+s} = 0 | i_n = 0] \approx \frac{\sum_{j=1}^{m-s} (1 - i_j) \cdot (1 - i_{j+s})}{\sum_{j=1}^{p} X_j}$$

and they are directly related to the correlation function of the indicator sequence, since, with the assumption of equally distributed i_k we have:

$$\begin{aligned}
\mathrm{Corr}[i_n, i_{n+k}] &= \frac{\mathrm{Cov}[i_n, i_{n+k}]}{\sqrt{\sigma_{i_n}^2 \sigma_{i_{n+k}}^2}} \\
&= \frac{\mathrm{E}[i_n i_{n+k}] - \mathrm{E}[i_n]\,\mathrm{E}[i_{n+k}]}{\sqrt{\sigma_{i_n}^2 \sigma_{i_{n+k}}^2}} \\
&= \frac{\mathrm{E}[i_n i_{n+k}] - \bar{e}^2}{\sqrt{\bar{e}(1-\bar{e})\bar{e}(1-\bar{e})}} \\
&= \frac{\Pr[i_{n+k} = 1 | i_n = 1]\bar{e} - \bar{e}^2}{\bar{e}(1-\bar{e})} = \frac{\Pr[i_{n+k} = 1 | i_n = 1] - \bar{e}}{(1-\bar{e})} \\
&\approx \Pr[i_{n+k} = 1 | i_n = 1]
\end{aligned}$$

where the approximation holds for small $\bar{e}$ values. Furthermore we have used that

$$\begin{aligned}
\mathrm{E}[i_n i_{n+k}] &= \sum_{x,y \in \{0,1\}} xy \Pr[i_n = x, i_{n+k} = y] \\
&= \Pr[i_n = 1, i_{n+k} = 1] \\
&= \Pr[i_{n+k} = 1 | i_n = 1] \cdot \Pr[i_n = 1] \\
&= \Pr[i_{n+k} = 1 | i_n = 1] \cdot \bar{e}
\end{aligned}$$

2.2.3 Modeling of Burst Lengths as iid Sequence of Random Variables

A very simple and popular class of stochastic models is the class of iid models, i.e. the random variables $X_1, \ldots, X_p$ are assumed to be independent and identically distributed, the same holds for $Y_1, \ldots, Y_p$ (and furthermore that the X_i and Y_i are also independent). The distinction between errorfree bursts and error bursts together with iid modeling give rise to the popular two-state (semi-)markov channel models (e.g. the Gilbert or Gilbert/Elliot channel model), where the channel state alternates between a “good” state (corresponding to error free bursts) and "bad" states (corresponding to error bursts). Beneath the distributions of the good state and bad state durations also the error rates in the bad state must be specified.

The invalidity of this assumption can be checked using the autocovariance function $R_X(k)$ of the sequence $\{X_i\}_{i\in\{1,\ldots,p\}}$ (the same holds for $R_Y(k)$ and the sequence $\{Y_i\}_{i\in\{1,\ldots,p\}}$), which is estimated using the method described in [2]

$$R_X(k) \approx r_k = \frac{c_k}{c_0} \quad (k \in \{1,\ldots,p-1\})$$

where

$$c_k = \frac{1}{p}\sum_{t=1}^{p-k}(X_t - \bar{X})(X_{t+k} - \bar{X})$$

The rule of thumb is that if for some $k \geq 1$ the value of r_k exceeds 0.2 then there is more than weak correlation and thus the X_i cannot be independent.

For modeling there are (at least) two possibilities: a) we use directly the histograms of the X_i and Y_i as distributions of error burst lengths and errorfree burst lengths, or b) we use some simple distribution function matching the sample mean of the X_i (Y_i) and maybe some higher moments, but otherwise neglecting the shape of the distribution. The first method is accurate but requires knowledge of the whole distribution, while the second method is inaccurate and requires only few parameters. In the following we restrict the discussion to the second method, since the simple distributions can easily be used within most simulation tools.

Markovian Two-State Model

For a markovian model the state holding times are necessarily geometrically distributed ($\Pr[X_i = k] = p_X(1-p_X)^{k-1}$, $\Pr[Y_i = k] = p_Y(1-p_Y)^{k-1}$ for $k \geq 1$) and errors occur independently with a fixed rate in every state. The geometric distribution only has a single parameter and thus we can only hope for matching the first moment of the "true" distribution. For estimating the parameters p_X and p_Y we use the fact that the geometric distribution has the mean value $E[X_i] = \frac{1}{p_X}$ and $E[Y_i] = \frac{1}{p_Y}$. Since $\bar{X}$ and $\bar{Y}$ are approximations of $E[X]$ and $E[Y]$ we simply put

$$\begin{aligned} p_X &= (\bar{X})^{-1} \\ p_Y &= (\bar{Y})^{-1} \end{aligned}$$

Furthermore, the bit error rate in the good state e_g is simply zero, while in the bad state it is given by

$$e_b = \frac{\sum_{i=1}^{p} Z_i}{\sum_{i=1}^{p} Y_i}$$

Semi-Markovian Two-State Model approximating the first two Moments

The basic channel model chosen here belongs also to the class of two-state models, however, the distributions of the state holding times are chosen to match the measured mean and variance more closely, thus in most cases loosing the memoryless property of these holding times.

We seek for a simple discrete random variable X taking only positive integer values, for which $E[X] = \bar{X}$ and $V[X] = \sigma_X^2$ holds, which are assumed to be positive, but otherwise arbitrary. Denote $a := \bar{X}$, $b := \sigma_X^2$. Unfortunately none of the well known distributions can be parameterized appropriately:

1. The binomial distribution, hypergeometric distribution and negative binomial distribution require $V[X] \leq E[X]$, which is often not true for errorfree lengths.
2. The geometric distribution and the Poisson distribution have only a single parameter to vary.
3. Two-Point, Three-Point and Four-Point distributions are restricted in their range of joint expectations and variances.

Therefore we employ certain heuristics to assign suitable distributions to the state durations:

1. For distributions with mean values $a \in [1, 3]$ and coefficients of variation $\frac{\sqrt{b}}{a} \leq \sqrt{\frac{2}{3}} \approx 0.81$ we choose a geometric distribution with parameter $p = \frac{1}{\bar{X}}$ ($p = \frac{1}{\bar{Y}}$ respectively). This is reasonable since the distribution has its main mass on 1 and decays rapidly because of its small coefficient of variation.
2. For mean values $a \in [3, 25]$, $b < a$ and coefficients of variation $\frac{\sqrt{b}}{a} \leq \sqrt{\frac{1}{3}} \approx 0.577$ we use a binomial distribution as an approximation[5]. The parameters are calculated as follows:

$$\begin{aligned} n &= \left\lceil \frac{a}{1 - \frac{b}{a}} \right\rceil \\ p &= \frac{a}{n} \end{aligned}$$

[5]The coefficient of variation of a binomial distribution with parameters n, p and $q = 1 - p$ is given by $C_X = \frac{\sqrt{q}}{\sqrt{np}}$. Since $np \geq 3$ we have that $C_X \leq \frac{1}{\sqrt{3}} \approx 0.577$.

The binomial distribution is chosen because in this case the mass of the distribution is shifted to the right, and thus has some similarity to the shape of the binomial distribution.

3. For all other cases we resort to a "quantized" version of a continuous distribution[6]. Since we only want to match the first two moments and neglect the shape of the distribution, we can restrict the set of candidate distributions to the well-known ones living on the positive real axis. We choose to use the lognormal distribution function [8], which has the density function

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\log x - \mu)^2}{2\sigma^2}} \quad (x > 0)$$

(for $\mu \in \mathbb{R}$ and $\sigma^2 > 0$) and the expected value and the coefficient of variation are given by

$$E[X] = e^{\mu + \frac{\sigma^2}{2}}$$

and

$$C_X = \sqrt{e^{\sigma^2} - 1}$$

respectively. Then we can simply calculate the parameters of the distribution as follows:

$$\begin{aligned} \sigma^2 &= \log\left(\frac{b}{a^2} + 1\right) \\ \mu &= \log a - \frac{\sigma^2}{2} \end{aligned}$$

The question of how to assign error rates to the bad states is addressed in section 2.2.5.

2.2.4 Hidden Markov Modeling

Beyond the models using markov chains recently error models based on hidden markov chains (hidden markov models, HMM) have found some interest. Some references are [14], [13], [15] and [4]. Especially in [4] a methodology is given for constructing a HMM based on indicator sequences, which, however, has the drawback of using only a single state for the good channel condition. However, this issue is not covered further in this report and remains to be investigated.

[6] The question how close mean and variance of a quantized version of a continuous random variable comes to the true values is not further addressed in this paper.

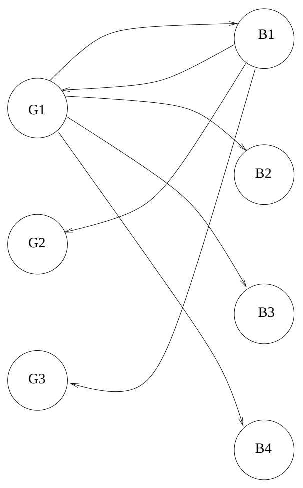


Figure 2.3: A sample bipartite markov chain

2.2.5 Bipartite Markovian Model

A natural generalization of the two state markov model is to employ a number n_1 of "bad" states and n_2 of "good" states and to allow state transitions only from good states to bad states and vice versa (thus forming a bipartite graph). An example model with three good states and four bad states is shown in figure 2.3 with all possible state transitions for the first good state and bad state respectively. If the states are numbered $e_1, \ldots, e_{n_1}, e_{n_1+1}, \ldots e_{n_1+n_2}$ then the transition matrix has the form:

$$\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{Q_1} \\ \mathbf{Q_2} & \mathbf{0} \end{pmatrix}$$

where $\mathbf{Q_1}$ is an $n_1 \times n_2$ stochastic matrix describing the state transitions from the bad states to the good states, while $\mathbf{Q_2}$ is an $n_2 \times n_1$ stochastic matrix for the other direction. A similar model in [7], however, uses two matrices P and Q, not further restricted, where P is used every time the preceding channel symbol was in error, while Q is used otherwise. In general the proposed model occurs as a special subclass of the class of Fritchman channel models [3], where, however, the bad states can have a bit error rate smaller than one. To the best of our knowledge, error models similar to our bipartite model are not discussed in the literature so far.

The operation of this model is as follows: to every state e_i there is assigned a probability distribution $p_{e_i}(k)$ on a subset of the natural numbers (or more precisely an independent random variable having just this distribution). When the system enters a specific good state

e_k, a random number is generated according to the distribution p_{e_k}. This random number is then interpreted as the number of bits where no error occur. When the system enters a specific bad state e_l, again a random number is generated according to p_{e_l}, determining the length of the error burst in bits. Within an error bursts we make the simplistic assumptions that

- at least one bit error occurs within the burst
- The bit errors occur independently with a fixed rate.

In order to build a model from the traces we need to identify the numbers of states n_1 and n_2, the matrices Q_1 and Q_2, the probability distributions p_{e_i} and the bit error rates in the bad states. A rather simple approach, which linearly approximates the specific shape of the involved distribution functions, can be shortly summarized as follows:

- Select a number of good states and bad states.
- Partition the range of possible error burst lengths such that every subinterval has the same probability.
- Construct the transition matrix **P** by simply counting for every state i the number of times it is left towards every possible target state j and divide this by the total number of times the system has left state i.
- Assign to every state a uniform distribution for the corresponding burst lengths (this is the linearization).
- We assume for the error states that errors occur independently with a fixed rate. For this rate two simple approaches exist:
  - All error states have the same mean bit error rate given by
$$\bar{e} = \frac{\sum_{k=1}^{p} Z_k}{\sum_{k=1}^{p} Y_k}$$
  - For every error state i denote $\Gamma_i \subset \{1, \ldots, p\}$ the subset of all error bursts which belong to state i and use
$$\bar{e}_i = \frac{\sum_{k \in \Gamma_i} Z_k}{\sum_{k \in \Gamma_i} Y_k}$$

In some more detail we perform the following steps for constructing the transition matrix **P**:

- Define $h_g \in \mathbb{N}$ and $h_b \in \mathbb{N}$ with $h_g > 1$ and $h_b > 1$. Denote for a single trace F_X and F_Y the distribution function of the X_i and Y_i respectively. This can be precalculated from the measured data.

- Let $x_1, \ldots, x_{h_g}$ be defined via
$$x_i := \max\left\{1, \left\lfloor F_X^{-1}\left(\frac{i}{h_g}\right)\right\rfloor\right\}$$
and $y_1, \ldots, y_{h_b}$ be defined via
$$y_i := \max\left\{1, \left\lfloor F_Y^{-1}\left(\frac{i}{h_b}\right)\right\rfloor\right\}$$
where we use the pseudoinverse of a distribution function defined by
$$F^{-1}(y) = \inf\{x : F(x) \geq y\}$$
In the following we assume that $x_0 = 1 < x_1 < x_2 < \ldots < x_{h_g}$ and $y_0 = 1 < y_1 < y_2 < \ldots < y_{h_b}$ holds, otherwise we assume that duplicates are removed and the numbers are re-indexed appropriately. Define $n_1 = h_g$, $n_2 = h_b$.

- In the next step we count for every interval $I_{X,i} = [x_{i-1}, x_i)$ ($i \in \{1, \ldots, h_g - 1\}$) and for interval $I_{X,h_g} = [x_{h_g-1}, x_{h_g}]$ the number of the values $X_1, \ldots, X_p$ that lie within this interval. Denote this number as $N_{X,i}$. Accordingly for the Y_i values, giving the numbers $N_{Y,i}$.

- Define a $(n_1 + n_2) \times (n_1 + n_2)$ matrix $\mathbf{P}'$ and set all entries to zero. Now we loop over the sequence $X_1 Y_1 Z_1 X_2 Y_2 Z_2 \ldots X_p Y_p Z_p$ the following way: for $i \in \{1, \ldots, p\}$ do:
  - let $I_{X,a}$ be the uniquely determined interval with $X_i \in I_{X,a}$ and let $I_{Y,b}$ be the uniquely determined interval with $Y_i \in I_{Y,b}$. Increment the matrix element $((\mathbf{P}'))_{n_1+a,b}$ by one.
  - if $i < p$: let $I_{Y,b}$ be the uniquely determined interval with $Y_i \in I_{Y,b}$ and let $I_{X,a}$ be the uniquely determined interval with $X_{i+1} \in I_{X,a}$. Increment the matrix element $((\mathbf{P}'))_{b,n_1+a}$ by one.

- For $i \in \{1, \ldots, n_1\}$ divide row i of $\mathbf{P}'$ by $N_{Y,i}$ and for $i + n_1 \in \{n_1 + 1, \ldots, n_2\}$ divide row $i + n_1$ of $\mathbf{P}'$ by $N_{X,i}$. The resulting matrix is then our state transition matrix $\mathbf{P}$.

However, in order to get results of some accuracy, reasonably large values of p are needed, i.e. we need a comparably high bit error rate. For low bit error rates this approach may lead to misleading models.

The number and distributions of the good states and bad states may also be determined by inspection.

Chapter 3

Campaign 1: Produktionstechnisches Zentrum (PTZ), Berlin, June 26, 2000

A set of measurements was performed on June 26, 2000 in the Produktionstechnisches Zentrum (PTZ) in Berlin, Germany, which is a research facility for machinery engineering, driven by industry and academia. The PTZ owns a large factory building, which contains several machines. The measurements were started at $\approx$ 11 AM and stopped at $\approx$ 16 PM. We have investigated a non line-of-sight (NLOS) scenario, with a die sinking electrical discharge (EDM) machine placed between transmitting and receiving station. Both stations have a distance of $\approx$ 6-7 meters. Furthermore it must be noted that the receiving computer was in close proximity to a cabinet containing the power supply for a huge 5 axis milling machine. Both stations are stationary during the measurement campaign, the die sinking machine has worked most of the time, only between 11.53 AM and 12.07 PM it was shut down. A trace number recorded in this interval is 22. In addition, the milling machine worked at the beginning of the campaign and was shut down between 12 PM and 13 PM. To our knowledge, no active interferers (e.g. 802.11 Wireless LANs or cordless telephone sets) were present. A sketch of the setup is shown in figure 3.1.

The main purposes of this campaign is to improve the measurement setup and the measurement methodology and to identify the aspects which are important for channel error modeling.

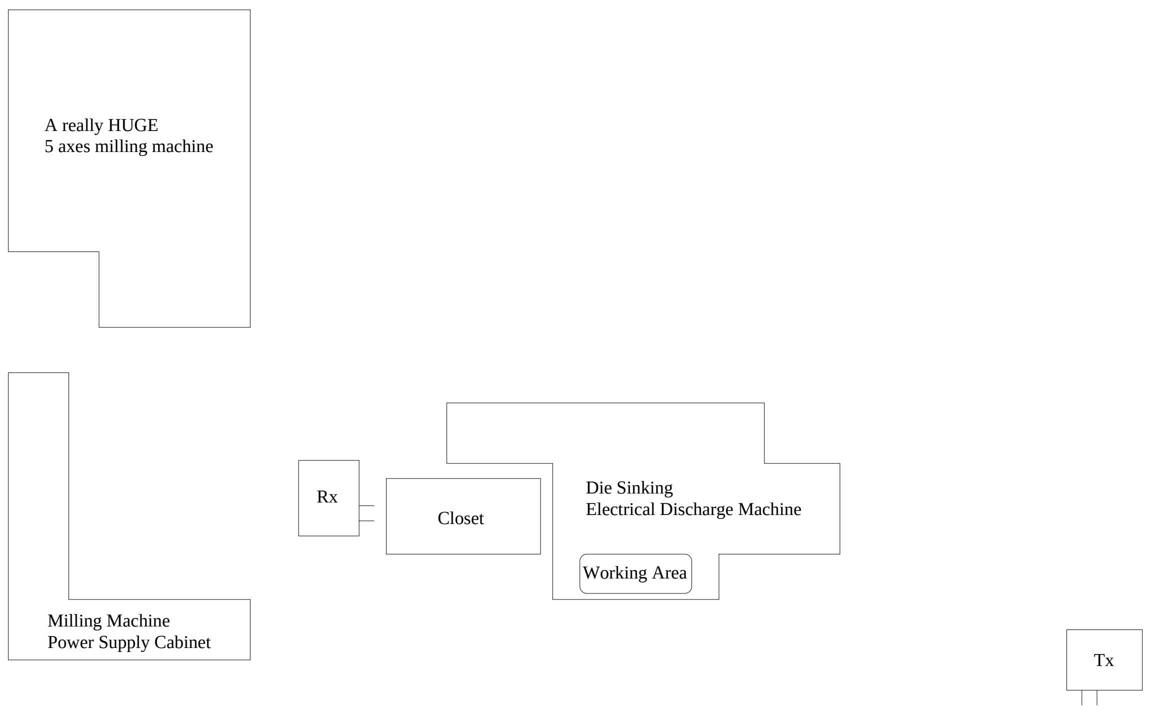


Figure 3.1: Setup of PTZ measurement

3.1 Measurement Parameters

The set of fixed parameters is given in table 3.1. For the remaining parameters we have chosen a factorial design, the values taken for the variable parameters are shown in table 3.2. A single trace consists of 20000 packets and lasts between one and ten minutes, depending on the packet size and gap time chosen. The traces are numbered consecutively, the mapping from the trace numbers to the parameters is given in table A.1. As a brief orientation, traces 1 to 36 are with 2 MBit QPSK modulation, traces 37 to 71 are with 11 MBit CCK modulation and the remaining traces are again with 2 MBit QPSK modulation. Within one type of modulation packet sizes increase, for a single packet size five traces with increasing gap times are made. The choice of 11 MBit CCK modulation is due to the fact that this modulation is also used in the IEEE 802.11 standard [10], [11].

Parameter	**Value**
PreambleLength	128 bits
ScramblingEnabled	True
DiversityEnabled	True
SFD Threshold	152
Frequency	12
NumPackets	20000
Rx-Tx-Distance	$\approx$ 6-7 meter
CRCUsageEnabled	False

Table 3.1: Fixed Parameters for PTZ Campaign

Parameter	**Value**
NumChunks	3, 9, 14, 28, 56, 112, 167
GapTime	600, 1000, 2000, 5000, 10000 μsec
ModulationCode	2 MBit QPSK, 11 MBit CCK

Table 3.2: Variable Parameters for PTZ Campaign

3.2 Measurement Results

In this section we present our measurement results. Since this was the first measurement campaign, we start with some results judging the quality of our measurement equipment, then proceeding with the most interesting first and second order statistics. In the next section then our conclusions regarding stochastic modeling are discussed.

3.2.1 Judging our Measurement Equipment

The key statistics for the quality of our measurement equipment are (see sections 2.1.2 and 2.2.1):

- The Rx counter `rcvbuf_dropped`, indicating the number of frames dropped due to memory shortage or too slow packet transfer to the host. However, this did not happen during this campaign.
- The statistics *ghost packets* counts the number of packets with correct length, where a correctly decoded sequence number is an already consumed one, i.e. where a too old packet is delivered (cf. section 2.2.1). We think that this can happen due to truncated packets, as discussed in section 2.1.2, however, we have not definitely proved this claim.
- The Rx counter `rxpackets` is incremented every time a frame of any (including wrong) size could be successfully received and stored. Accordingly, we define the *lost packets* statistics to be given by *NumPackets* minus `rxpackets`.
- The Rx counter `hostbuf_dropped` counts the number of frames which are too large for a single buffer in the host interface (probably pointing to an error in communicating packet lengths between baseband processor and Rx software)
- The *oversized packets* statistics counts the number of packets larger than allowed (which could happen due to interfering Wireless LANs or to errors in passing the packet length from baseband processor to the Rx software), the *truncated packets* statistics counts the number of packets too small (which can occur for similar reasons or by loss of bit synchronization).
- The value `rcvframe_dropped` was not available in this campaign.

In figure 3.3 we show the rates of ghost packets, oversized packets and truncated packets vs. the trace number, in figure 3.2 we show the *hostbuf dropped* statistics. It is interesting

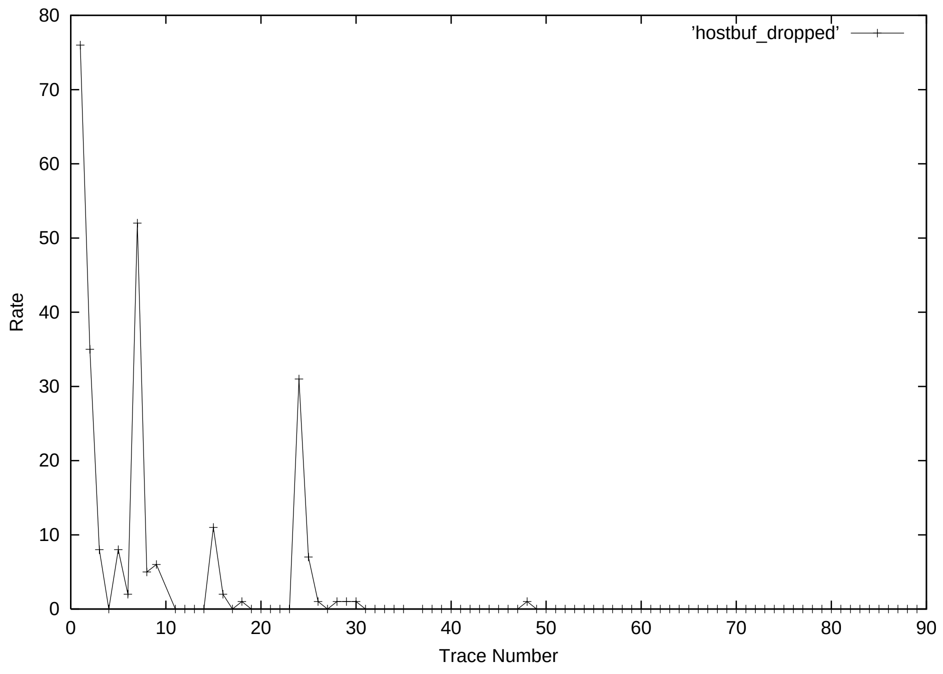


Figure 3.2: Rates of hostbuf dropped packets w.r.t. rxpackets

to note that all curves are practically zero starting with trace number 26. The presumption that the occurence of missized packets is due to a software error can be rejected with high probability, because otherwise the error is likely to occur over the whole campaign. Instead we think that the good behaviour starting with trace 26 coincides with shutting down the milling machine (remember that the receiver was located right near the power supply cabinet) and that the errors are due to magnetic and electrical induction. It is often assumed that truncated packets are mainly a result of loss of bit synchronization, which in turn is often due to multipath fading. However, our results do not support this assumption, since the truncated packet rates are comparably low and occur only up to trace number 26. Furthermore, the rate of missized packets are comparably low and can be neglected in modeling.

In figure 3.4 we show the percentage of lost packets (see section 2.2.1) versus the trace number. This figure shows the same behaviour, i.e. starting with trace 26 this number drops to zero. Since for some traces the fraction of lost packets (presumably due to failing preamble acquisition or PHY header checksum errors) reaches values near 10% this phenomenon should be included in stochastic channel modeling, since lost packets are an important “input” to the MAC protocol, which cannot be handled by lower layer mechanisms, e.g. FEC schemes.

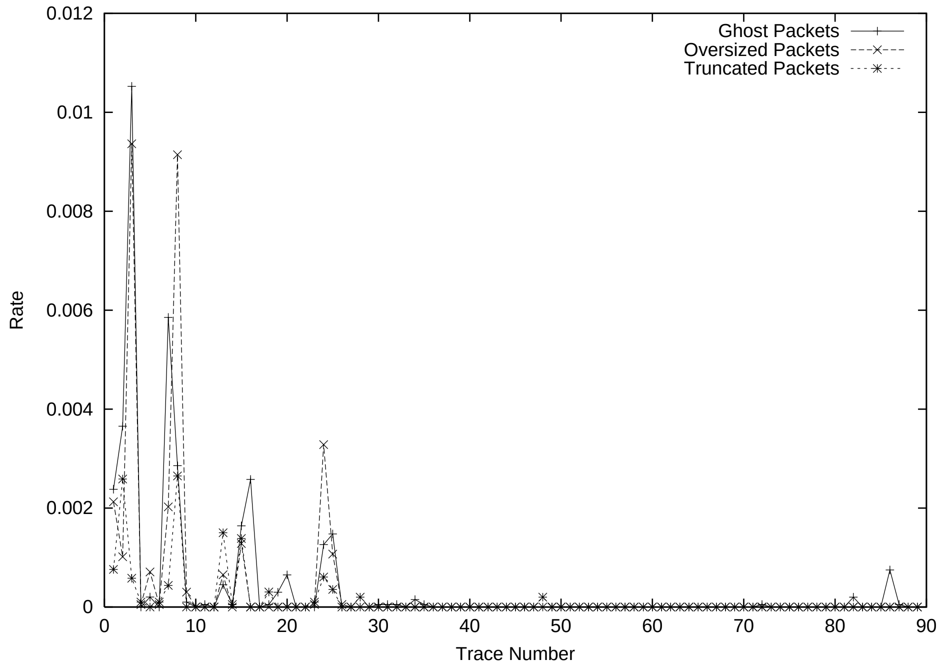


Figure 3.3: Rates of ghost packets, oversized packets and truncated packets w.r.t. rxpackets

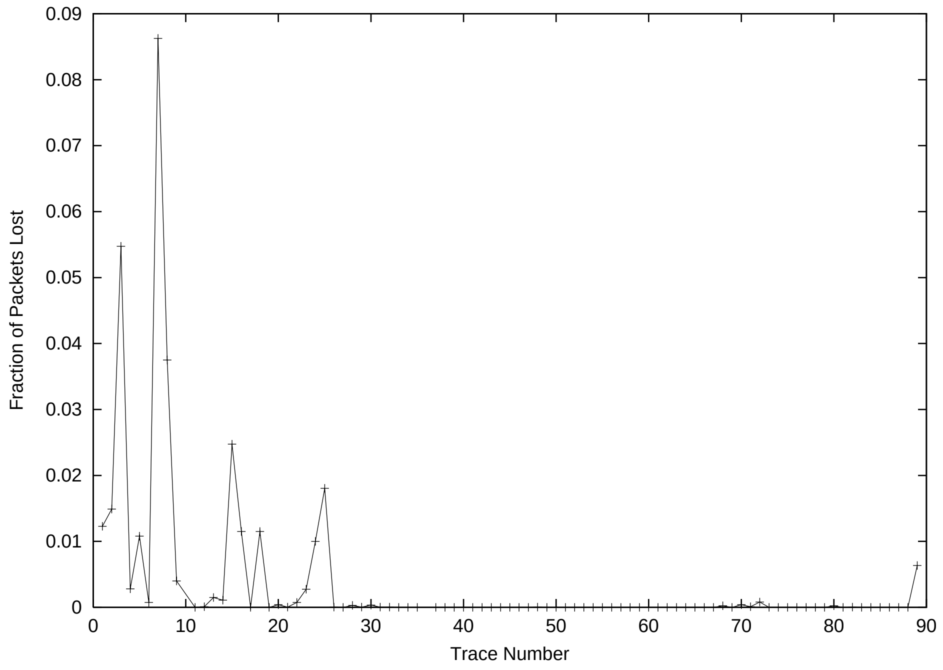


Figure 3.4: Percentage of lost packets vs. Trace Number

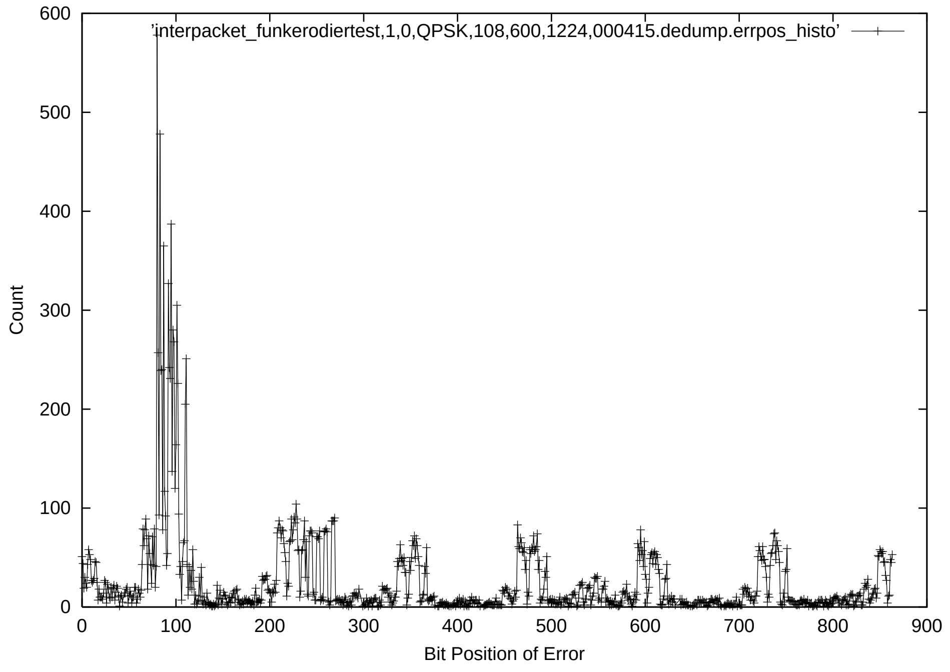


Figure 3.5: Positions of Bit Errors, Trace Number 1

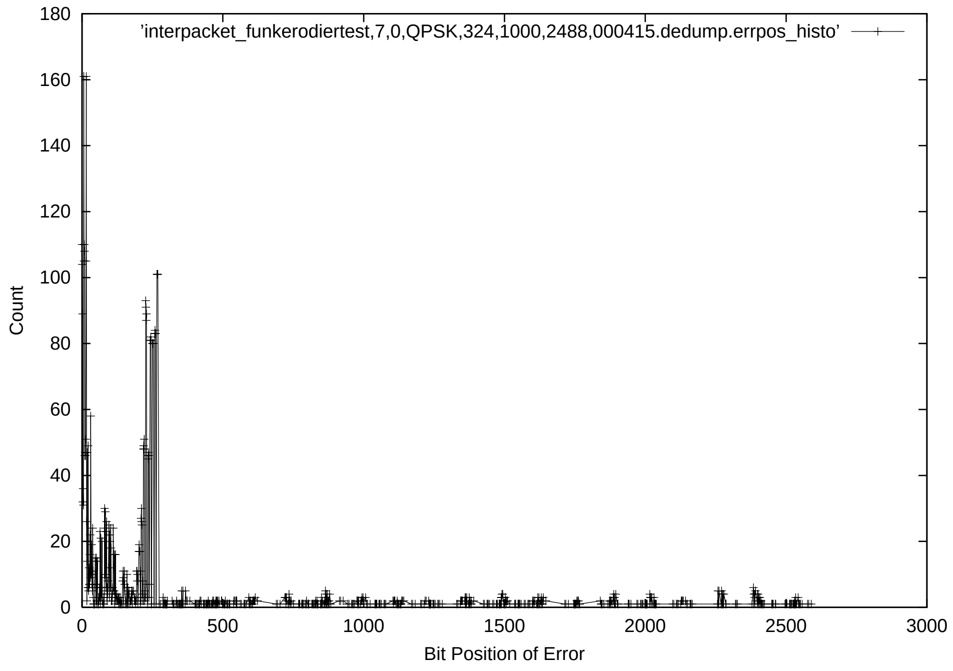


Figure 3.6: Positions of Bit Errors, Trace Number 5

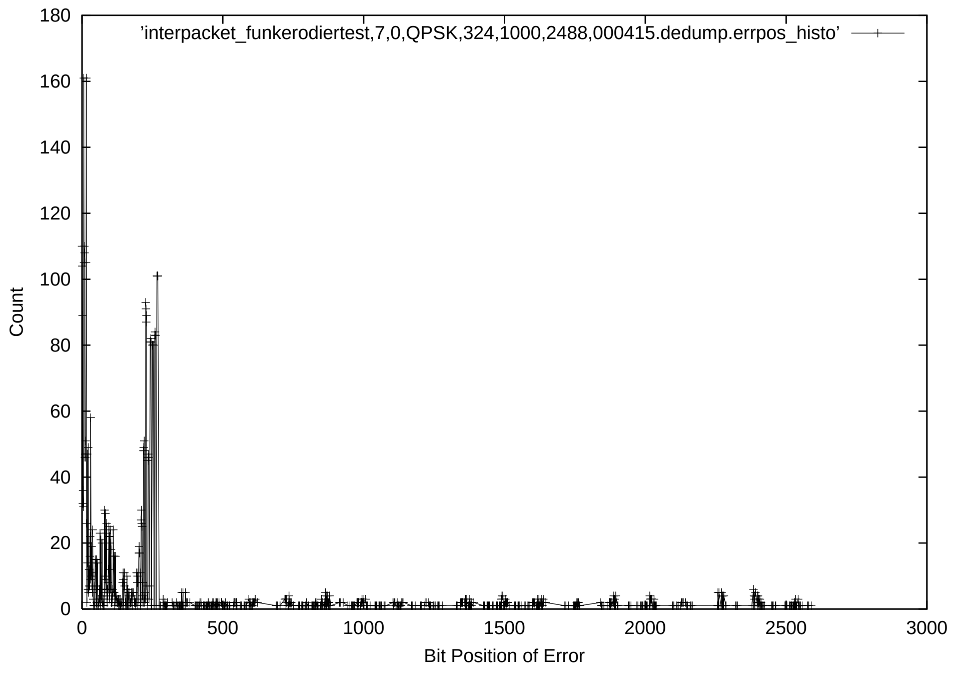


Figure 3.7: Positions of Bit Errors, Trace Number 7

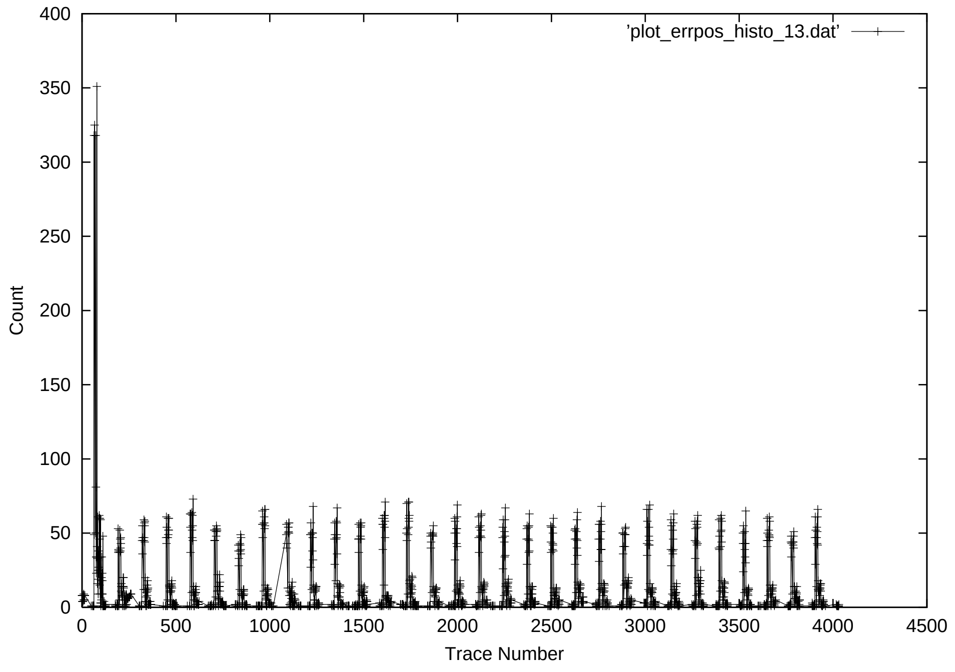


Figure 3.8: Positions of Bit Errors, Trace Number 13

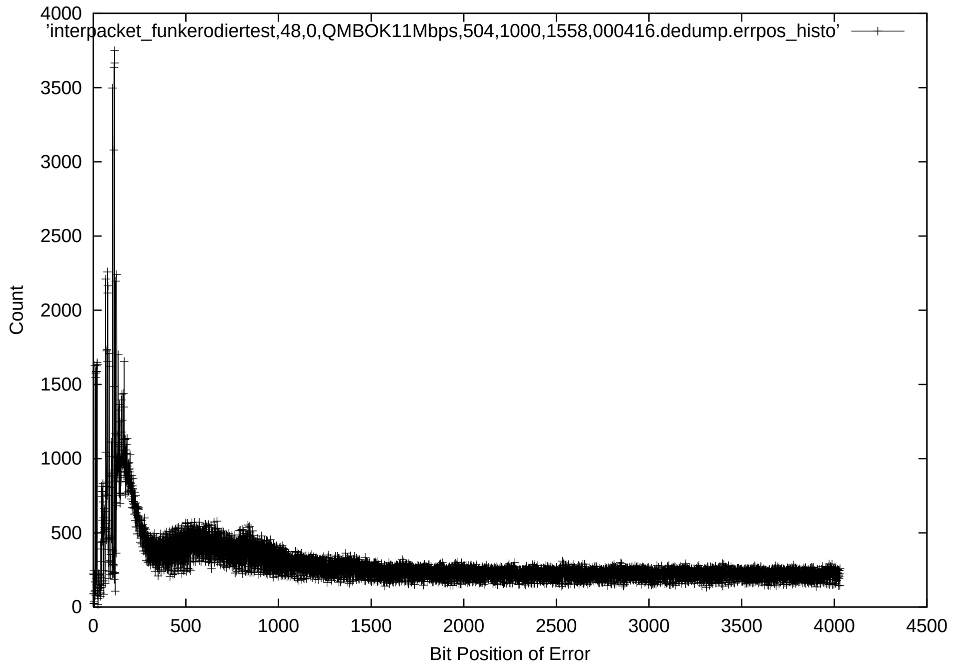


Figure 3.9: Positions of Bit Errors, Trace Number 48

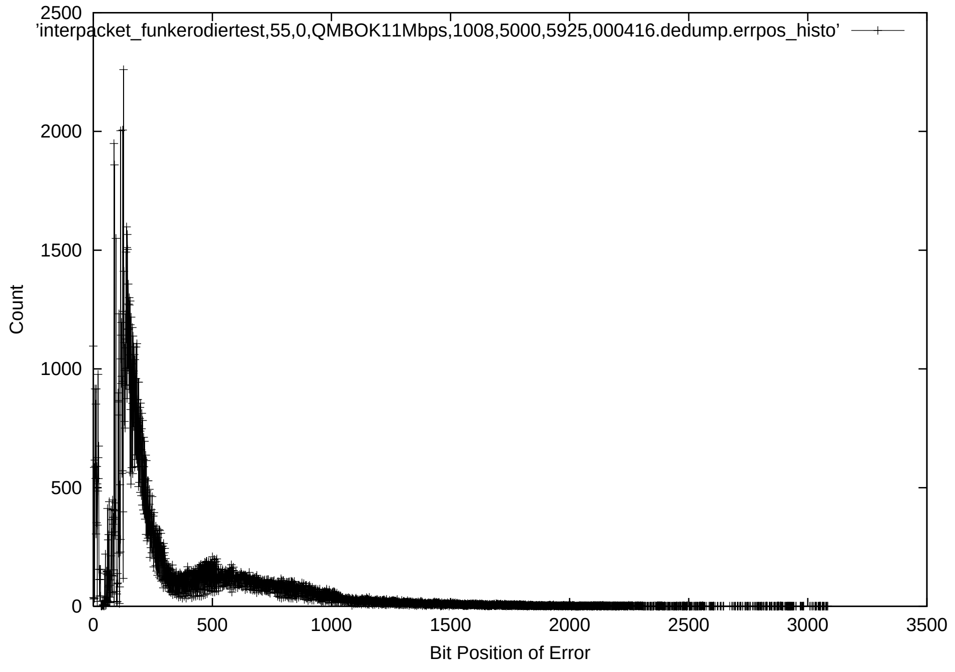


Figure 3.10: Positions of Bit Errors, Trace Number 55

In figures 3.5, 3.6, 3.7, 3.8, 3.9, 3.10 we show the error position histogram for selected traces. That is, within a single trace for every bit error we determine its bit position index within its packet and increment an appropriate counter (which is associated to the index number). Most of the QPSK traces show a pronounced peak around bit position 100 and further peaks with nearly equal distances (from inspection the distance in many traces is approximately 128). Two examples are figures 3.5, 3.8. Other QPSK traces look more irregular, however, the peak at the beginning of a frame occurs also. For CCK modulation the curves shown in figures 3.9, 3.10 are representative. They also show a peak at the beginning of a frame (up to index 200), then the histogram decays. For this modulation we have no periodicity observed. To summarize, we can observe two different effects:

- Bit errors tend to occur most often at the beginning of a frame. This holds for both modulation types. We think that this can be attributed to the need of switching the modulation type from the DBPSK modulated PHY header to the QPSK or CCK modulated data part. However, this should be checked against traces employing DBPSK modulation.
- For QPSK modulation additionally some periodicity can be observed. We currently have no satisfactorily explanation for this, however, we think this can be due to the operation of the scrambler [5]. For this reason in the next campaign we perform a set of measurements with scrambling disabled.

A major problem with the results shown in this section is that it is not immediately clear, whether we have just used a bad pair of modems (due to monday production), or if the described behaviour is typical for this family of modems or even for all 802.11 compliant modems. For this reason in the next campaign we should perform measurements with another pair of modems and compare the results. More in the future it is planned to upgrade to the PRISM II chipset.

NumChunks	Mean BER
3	0.0013868
9	0.0001284
14	0.0000848
28	0.0000084
56	0.0001908
112	0.0000714
167	0.0000242

Table 3.3: Mean BER vs. packet size

3.2.2 First Order Statistics

In this section we show the main first order statistics, some of them are introduced in section 2.2.2. In figure 3.11 we show the mean bit error rate per trace E_k vs. the trace number k (also given in table A.1). In figure 3.12 we show for every trace the fraction of erroneous packets, where an erroneous packet is defined to be a packet with at least one bit error. The following observations are interesting:

- For QPSK modulation traces the mean BERs vary over many orders of magnitude and over longer timescales (remember that each trace lasts at least one minute, up to ten), even for a single packet size. Several QPSK traces show no bit errors at all. However, as a general trend we can observe that the mean bit error rate decreases with increasing packet size (increasing *NumChunks*), as can be seen from table 3.3. An explanation is that the most of the errors tend to occur at the beginning of packets (typically within the first 1000 bits, see also section 3.2.1) while in the middle and end of a packet relatively fewer errors occur. So for longer packets the error clustering at the beginning can be compensated. Furthermore it can be observed that the packet error rates for QPSK are all below 20%, this holds for all packet sizes. This is a further indication that bit errors in the remainder of a packet occur only rarely, since otherwise the packet error rate should increase more than linearly with increasing packet sizes[1]. We have not observed any clear relationships for the *GapTime* parameter.
- For CCK modulation traces the mean BERs have a much higher level, they do not vary so much, and they are decaying with time, while the packet error rates seem to

[1]An interesting experiment would be to cut off the first 1000 bits from all packets and to run all evaluations again.

remain constant (however, on a high level). We think that this behaviour is due to the same reasons as in the QPSK case, however, the CCK modulation seems to be far more susceptible to errors.

Of prime importance for the iid modeling approach are the distributions of the error burst lengths and the error free burst lengths. However, before turning to their distributions, we shortly discuss the behaviour of their mean values and variances (or coefficient of variations, respectively). In figure 3.13 the respective mean values and in 3.14 the coefficients of variation are shown. The following points are important:

- The mean error burst lengths do not differ significantly from the burst order value:
  - For $k_0 = 1$ the mean burst lengths in most cases is smaller than two, furthermore the coefficient of variation is in almost every case smaller than one, thus the respective distributions show their main mass between one and three, and often look like geometric distributions (see below).
  - For $k_0 = 8$ all except one mean error burst lengths are smaller than 20, thus not longer as 2.5 times k_0.
  - For $k_0 = 15$ and QPSK modulation the mean values are bounded by 30, most of them are in the proximity of 20.
  - Interestingly, for $k_0 = 50$ and $k_0 = 100$ and QPSK modulation the variation of the mean error burst lengths increases, while for lower burst orders the mean values the variation is smaller.

  Thus in general we can say that error bursts tend to be short. Furthermore, except for $k_0 = 1$ error bursts for CCK tend to be longer than for QPSK, which, however, can be easily explained by the higher mean bit error rates for CCK.

- The mean error burst lengths do not increase linearly with the burst order k_0. This is to be expected if we have longer error free bursts within the trace, which form a "natural barrier" for error bursts.

- For QPSK modulation the coefficient of variation is almost below two (except for $k_0 = 100$), thus the variation is bounded. For CCK modulation the variation can be much larger, except for the burst order $k_0 = 1$.

Since in most cases the range of the error burst length is comparably restricted, we show directly the density plots (normalized histograms). However, the errorfree distributions span

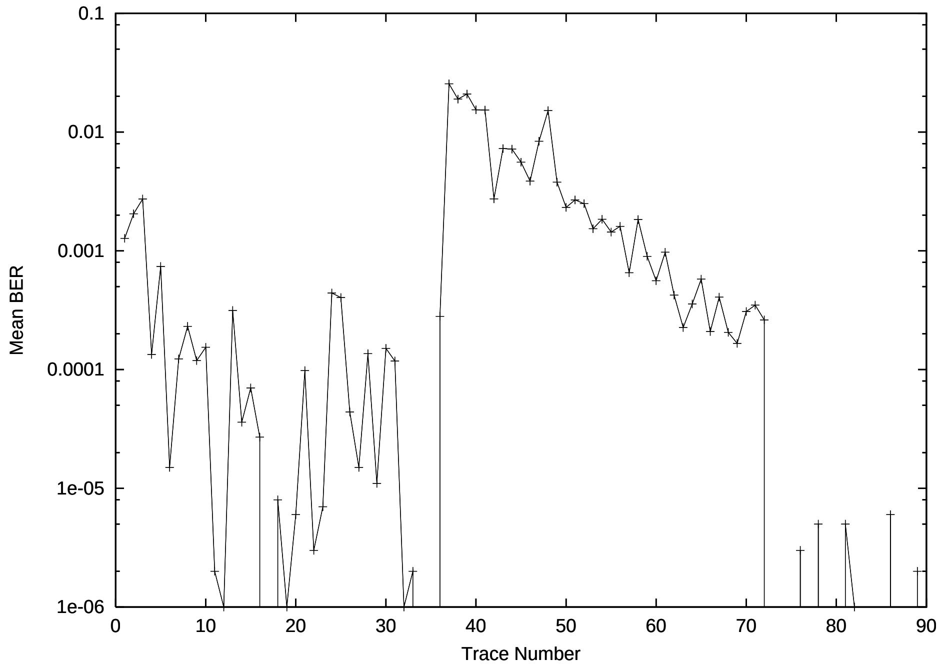


Figure 3.11: Mean BER (over whole trace) vs. trace number

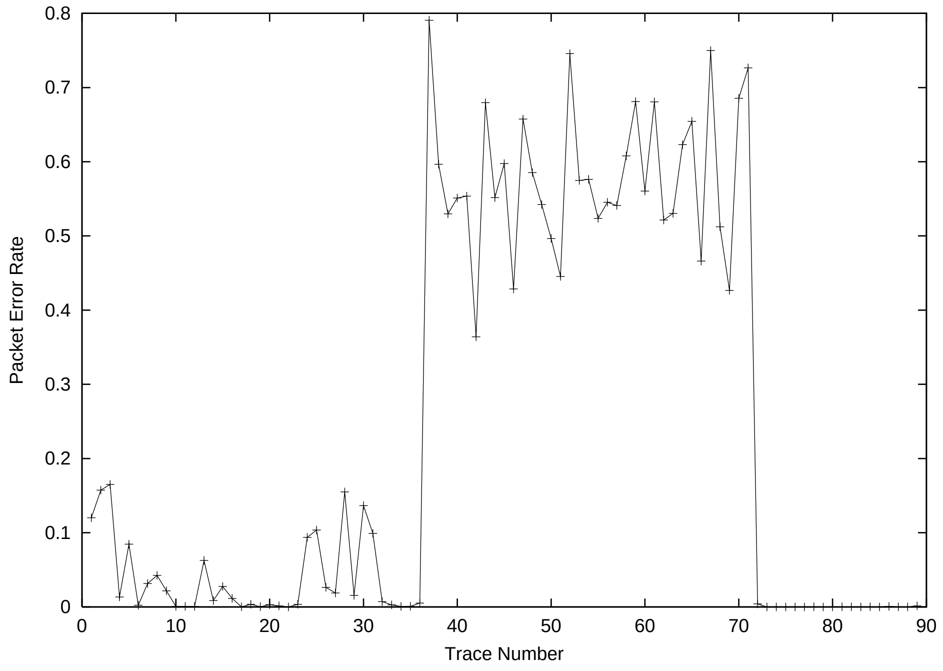


Figure 3.12: Packet Error Rate (over whole trace) vs. trace number

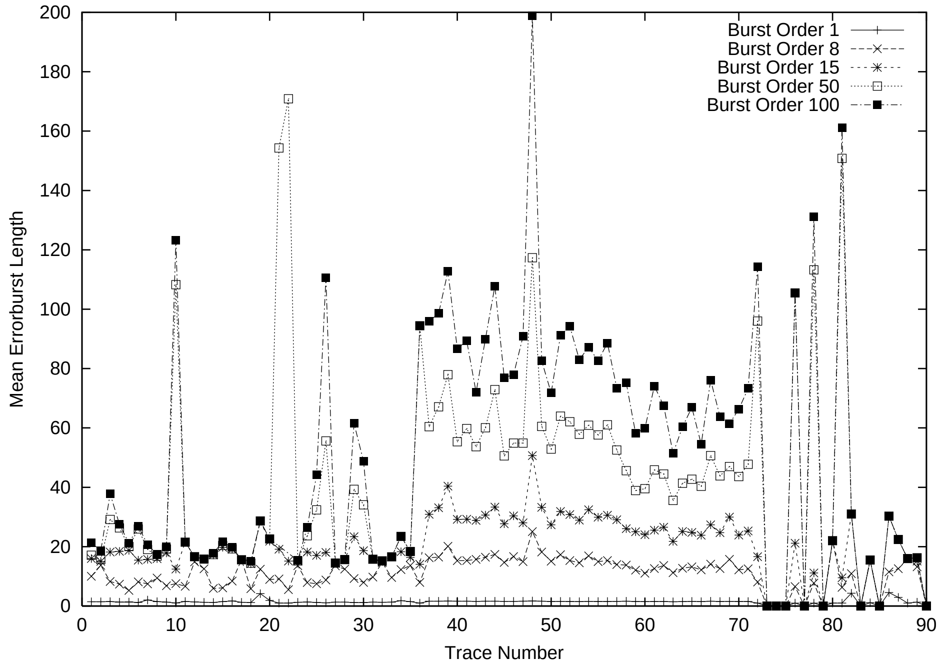


Figure 3.13: Mean Error Burst Lengths vs. trace number

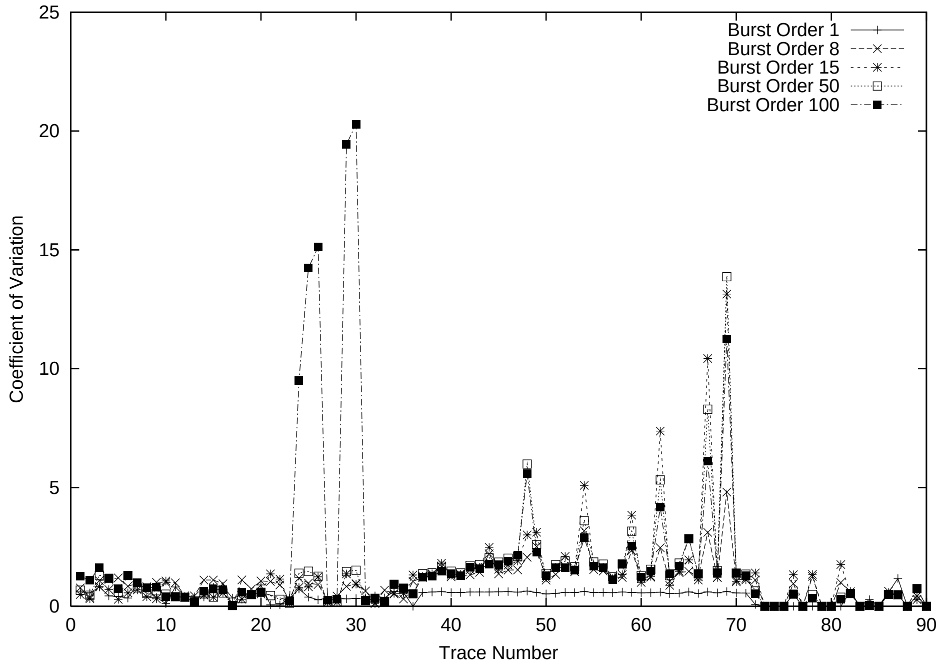


Figure 3.14: Coefficient of Variation of Error Burst Lengths vs. trace number

a much wider range, which is sparsely covered, especially in the tail regions, and shows wild fluctuations in small regions (for a longer trace lengths these fluctuations are assumed to be smaller). Therefore we choose to show a "smoothed" version of the errorfree length distributions, according to the methodology used in [6][2]. The distributions of the lengths of error bursts of order $k_0 = 1$ are shown in tables 3.4, 3.5, 3.6, 3.7, while for $k_0 = 15$ the distributions are shown in tables 3.8, 3.9, 3.10, 3.11. The smoothed curves for the errorfree runlength distributions for $k_0 = 1$ are shown in 3.12, 3.13, 3.14, 3.15, while for $k_0 = 15$ please refer to tables 3.16, 3.17, 3.18 and 3.19. We can make the following important observations:

- For $k_0 = 1$ the error burst lengths in many cases can be well approximated by a geometric distribution. This can be seen from the shape of the distributions as well as from the fact that the mean values are often between one and three and the coefficients of variation are small. This holds for both modulation types. Furthermore, most error bursts are only a single bit long.

- For $k_0 = 1$ the smoothed versions of the density functions for the errorfree length distribution show roughly a similar behaviour: the first part the density decays linearly, the remaining part is nearly constant and has a small nonzero value.

- For $k_0 = 15$ the smoothed versions of the density functions for the errorfree length distribution look irregular, there seems to be no common pattern, except that the tail of the distribution often is close to zero.

In figure 3.15 we show the mean values for the error rates within the bad states (given by $\bar{m} = \frac{1}{p}\sum_{i=1}^{p}\frac{Z_i}{Y_i}$ and the sample variance given by $\frac{1}{p-1}\sum_{i=1}^{p}\left(\frac{Z_i}{Y_i} - \bar{m}\right)^2$) vs. the trace number, while in figure 3.16 we show the corresponding coefficients of variation. The following points are worth discussion:

- In general the error densities (i.e. mean bit error rate during bad states) have comparably high values of more than 0.2. Please not that by our definition of error bursts the density is bounded below by $1/k_0$ for bursts of order k_0. However, even for burst order $k_0 = 100$ the mean density only rarely drops below 0.3.

[2] In short: let $x_1, \ldots, x_n$ be the observations, let $h \in \mathbb{R}^+ \setminus \{0\}$ a parameter and let $K : \mathbb{R} \to \mathbb{R}^+$ be a so-called "estimator kernel" function with $\int_{\mathbb{R}} K(t)dt = 1$. Then the estimated density $f(x)$ at $x \in \mathbb{R}$ is given by $f(x) = \sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-x_i}{h}\right)$. The rationale is to give more weight to places where many of the x_i are clustered. One crucial point is the choice of h, which can be interpreted as the window size. As a heuristic we choose h to be 10 percent of the range of the x_i values. In the paper [6] the authors have chosen K to be a triangular window: $K(t) = (1+t)\cdot\mathbf{1}_{[-1,0)}(t) + (1-t)\cdot\mathbf{1}_{[0,1]}(t).$, as is done in [6]. However, in order to avoid mass on negative values, we have used $K(t) = (2-2t)\cdot\mathbf{1}_{[0,1]}(t)$ instead.

- For all burst orders the coefficient of variations of the error densities are well below 0.7, so the corresponding distributions have their main mass clustered around the mean error density.
- For CCK modulation both the mean values and the coefficients of variation remain nearly constant, with the mean density decreasing with increasing burst orders.
- For QPSK modulation there is more variation in the mean values and coefficients of variations, however, the trend for higher mean values for smaller burst orders can be observed again. The mean values for $k_0 = 50$ and $k_0 = 100$ are often very close to each other.

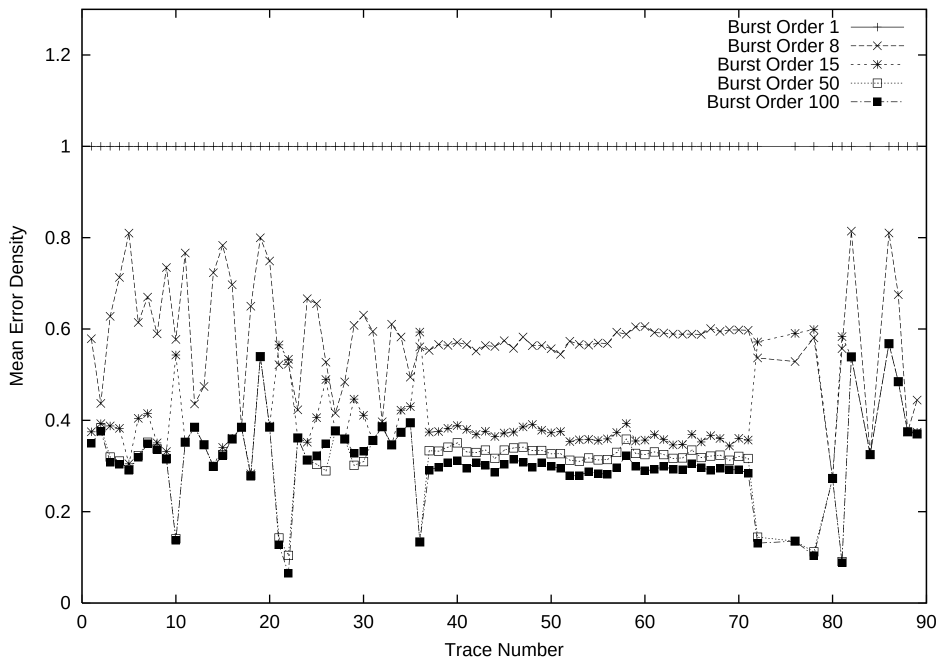


Figure 3.15: Mean Error Density vs. trace number

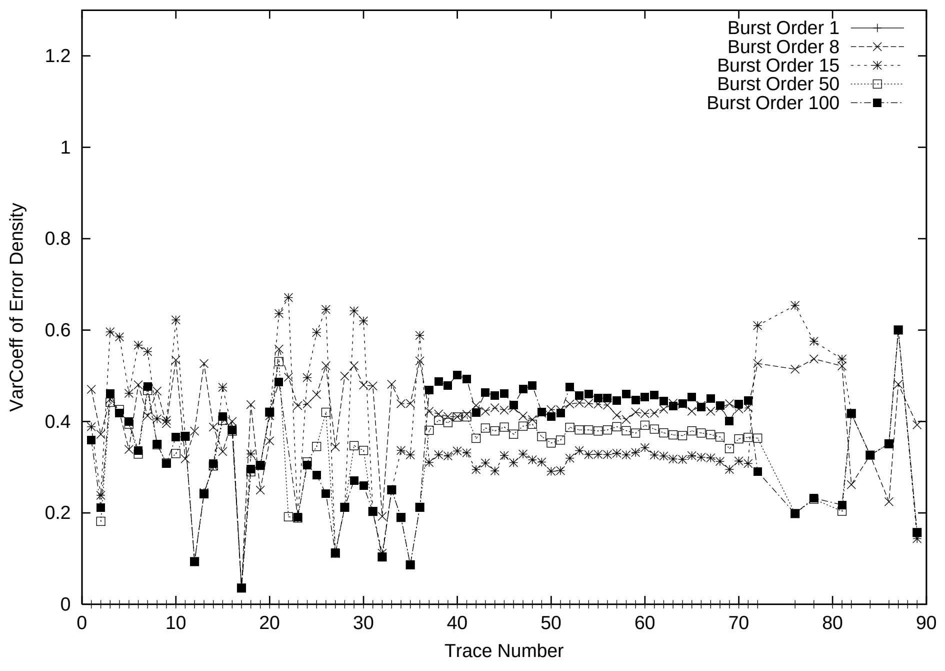
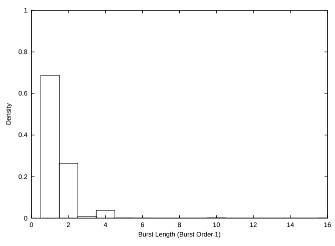
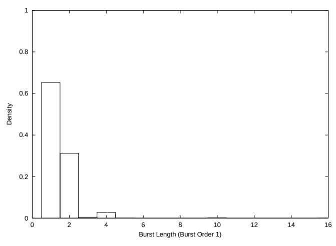


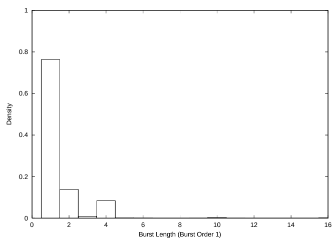
Figure 3.16: Coefficient of Variation of Error Densities vs. trace number



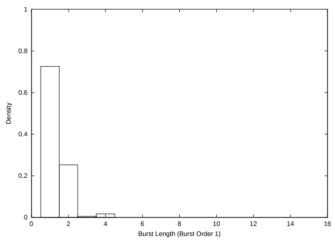
Trace 1, E_k = 0.001276



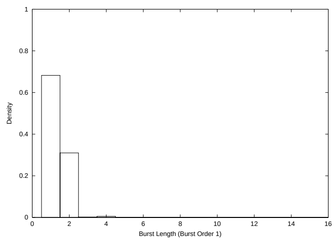
Trace 2, E_k = 0.002051



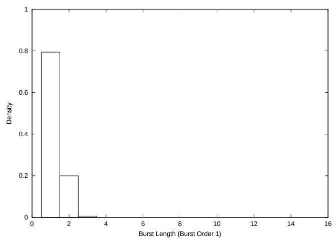
Trace 3, E_k = 0.002734



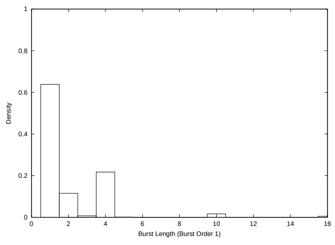
Trace 4, E_k = 0.000134



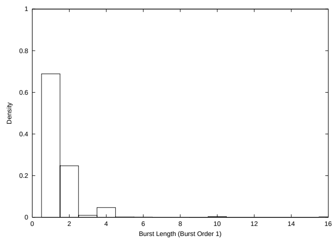
Trace 5, E_k = 0.000739



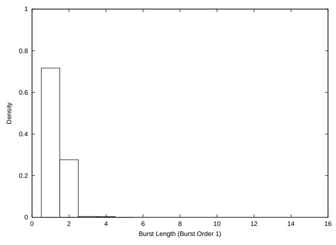
Trace 6, E_k = 0.000015



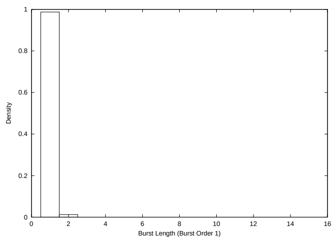
Trace 7, E_k = 0.000123



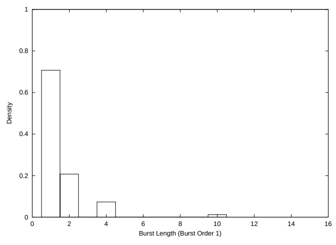
Trace 8, E_k = 0.000231



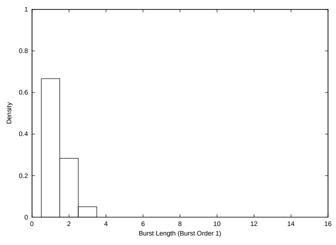
Trace 9, E_k = 0.000119



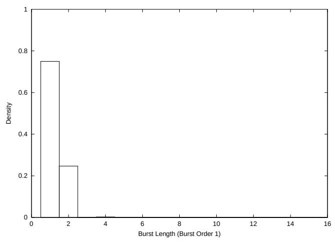
Trace 10, E_k = 0.000154



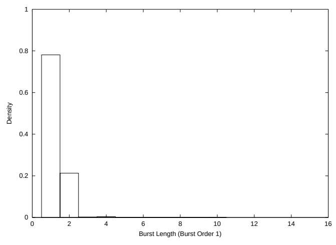
Trace 11, E_k = 0.000002



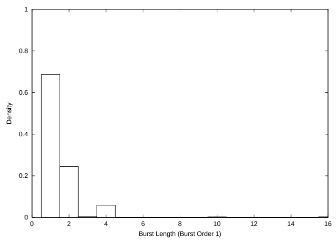
Trace 12, E_k = 0.000001



Trace 13, E_k = 0.000315

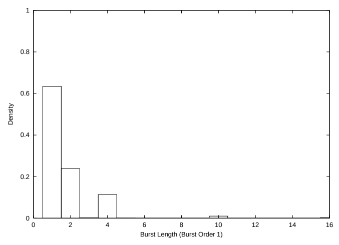


Trace 14, E_k = 0.000036

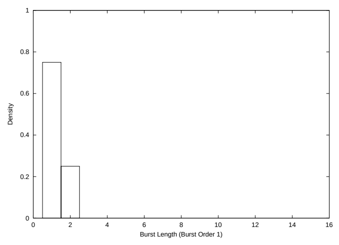


Trace 15, E_k = 0.000070

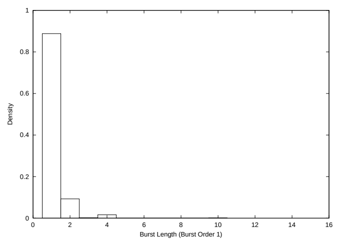
Table 3.4: Density of Error Burst Length (Order 1) for Traces 1 to 15



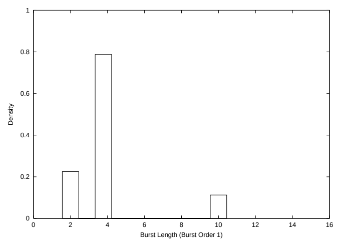
Trace 16, $E_k = 0.000027$



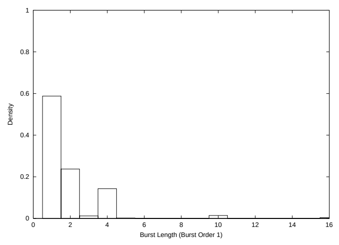
Trace 17, $E_k = 0.000000$



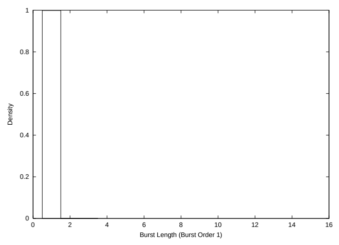
Trace 18, $E_k = 0.000008$



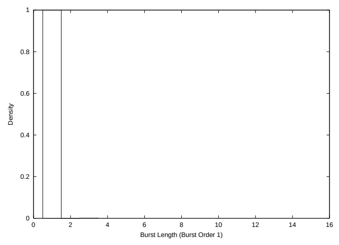
Trace 19, $E_k = 0.000001$



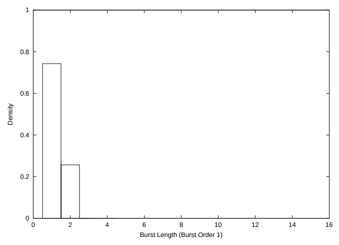
Trace 20, $E_k = 0.000006$



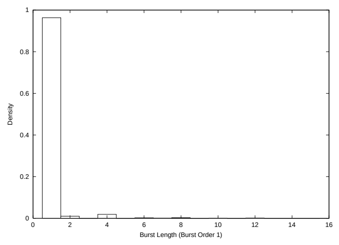
Trace 21, $E_k = 0.000098$



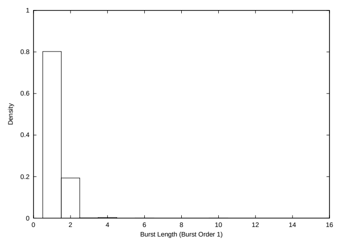
Trace 22, $E_k = 0.000003$



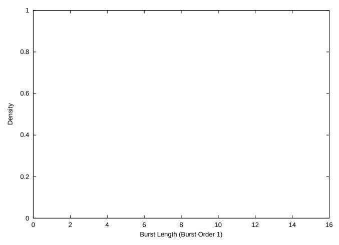
Trace 23, $E_k = 0.000007$



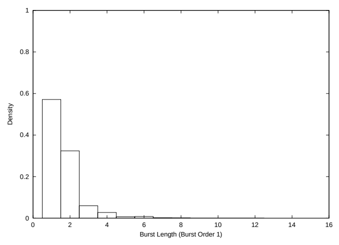
Trace 24, $E_k = 0.000442$



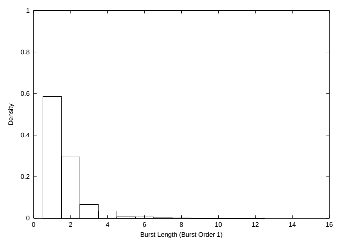
Trace 25, $E_k = 0.000404$



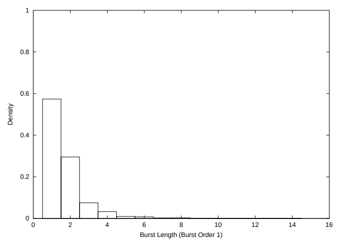
Trace 36, $E_k = 0.000281$



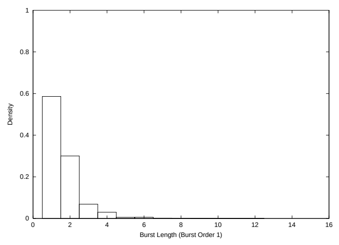
Trace 37, $E_k = 0.025511$



Trace 38, $E_k = 0.018970$

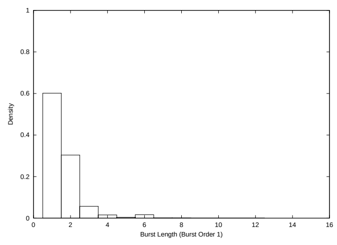


Trace 39, $E_k = 0.020942$

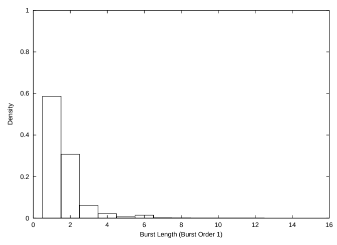


Trace 41, $E_k = 0.015353$

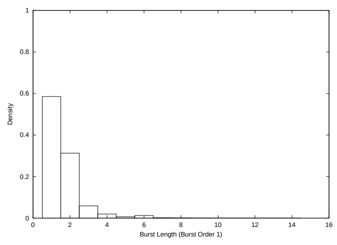
Table 3.5: Density of Error Burst Length (Order 1) for Traces 16 to 41



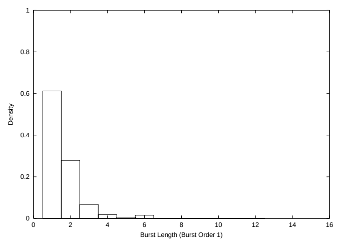
Trace 42, $E_k = 0.002740$



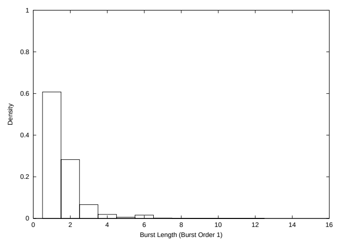
Trace 43, $E_k = 0.007271$



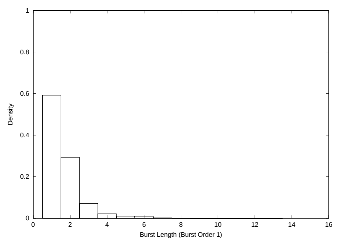
Trace 44, $E_k = 0.007188$



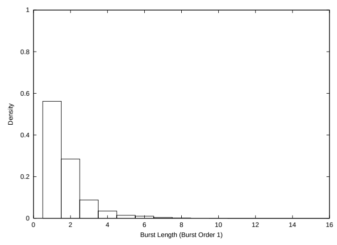
Trace 45, $E_k = 0.005575$



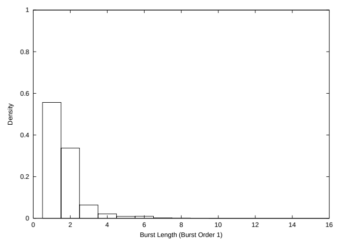
Trace 46, $E_k = 0.003861$



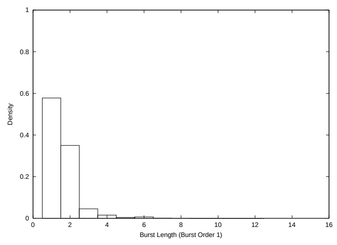
Trace 47, $E_k = 0.008375$



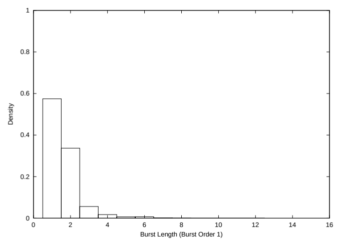
Trace 48, $E_k = 0.015173$



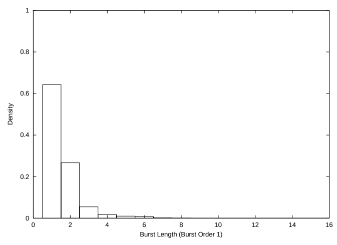
Trace 49, $E_k = 0.003788$



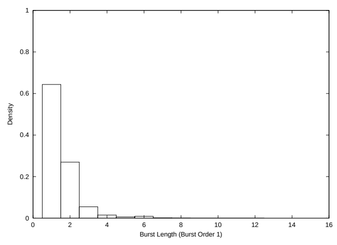
Trace 50, $E_k = 0.002321$



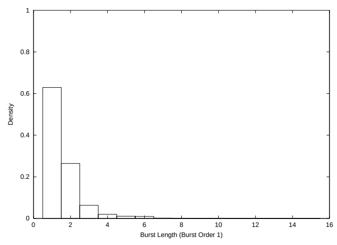
Trace 51, $E_k = 0.002682$



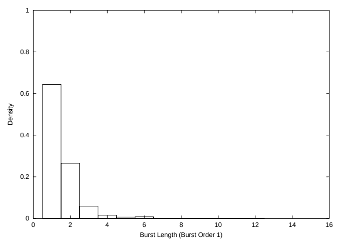
Trace 52, $E_k = 0.002499$



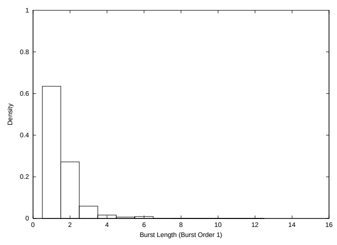
Trace 53, $E_k = 0.001536$



Trace 54, $E_k = 0.001844$

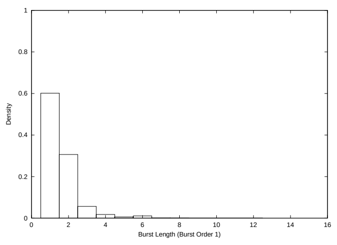


Trace 55, $E_k = 0.001441$

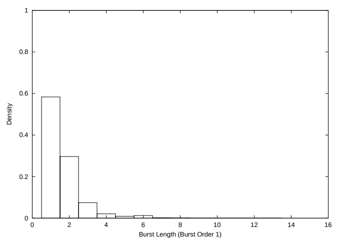


Trace 56, $E_k = 0.001608$

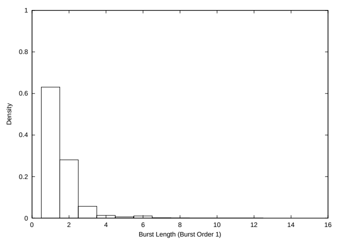
Table 3.6: Density of Error Burst Length (Order 1) for Traces 42 to 56



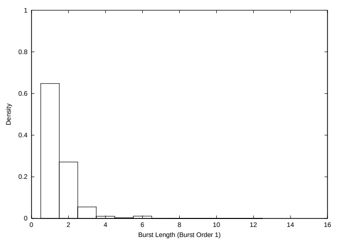
Trace 57, $E_k = 0.000654$



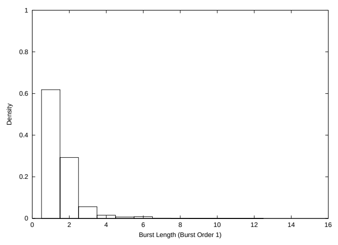
Trace 58, $E_k = 0.001836$



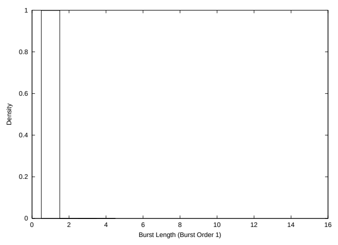
Trace 59, $E_k = 0.000898$



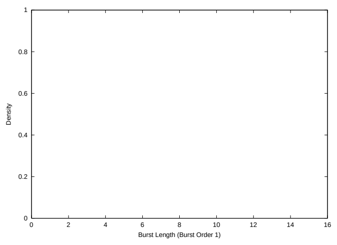
Trace 60, $E_k = 0.000560$



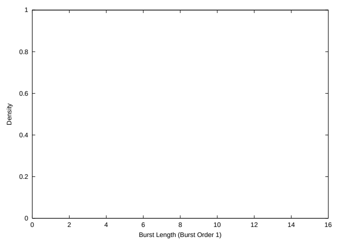
Trace 61, $E_k = 0.000974$



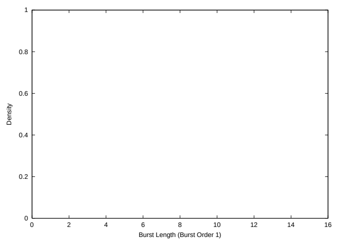
Trace 72, $E_k = 0.000262$



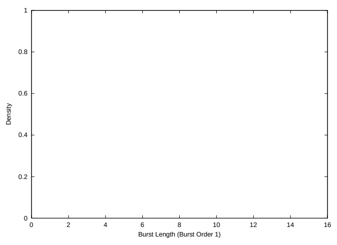
Trace 76, $E_k = 0.000003$



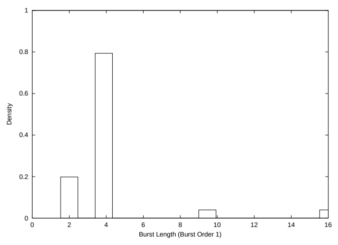
Trace 78, $E_k = 0.000005$



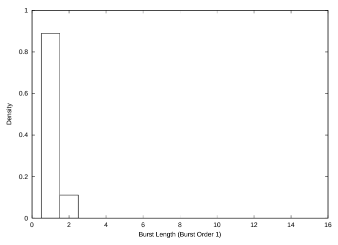
Trace 80, $E_k = 0.000000$



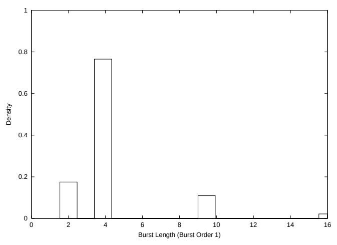
Trace 81, $E_k = 0.000005$



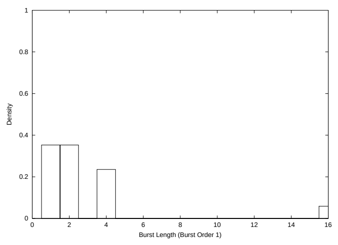
Trace 82, $E_k = 0.000001$



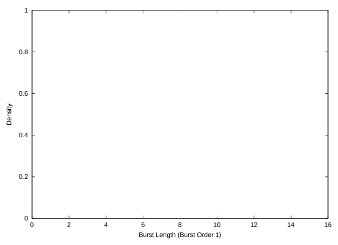
Trace 84, $E_k = 0.000000$



Trace 86, $E_k = 0.000006$

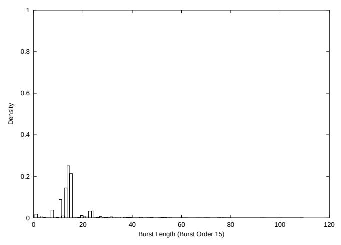


Trace 87, $E_k = 0.000000$

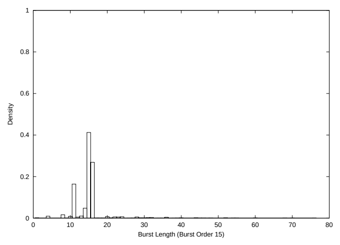


Trace 88, $E_k = 0.000000$

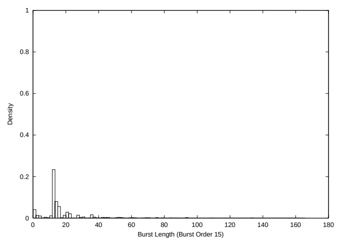
Table 3.7: Density of Error Burst Length (Order 1) for Traces 57 to 88



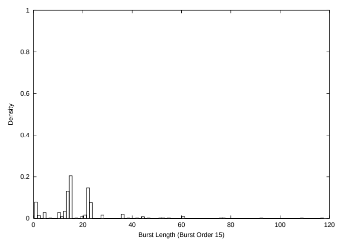
Trace 1, $E_k = 0.001276$



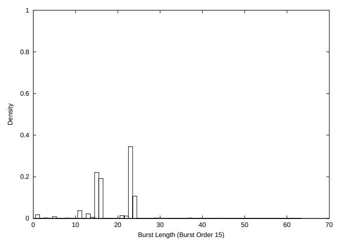
Trace 2, $E_k = 0.002051$



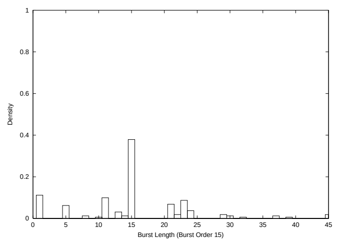
Trace 3, $E_k = 0.002734$



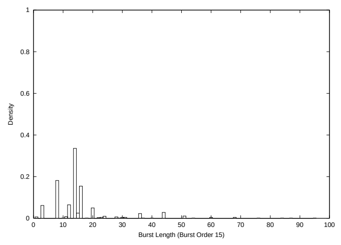
Trace 4, $E_k = 0.000134$



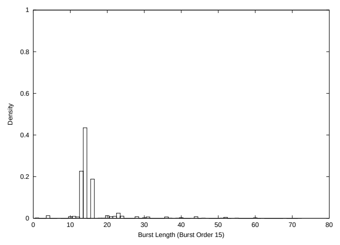
Trace 5, $E_k = 0.000739$



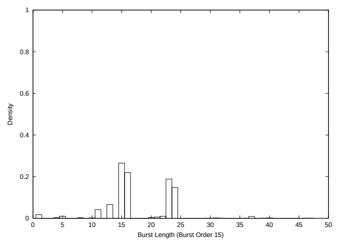
Trace 6, $E_k = 0.000015$



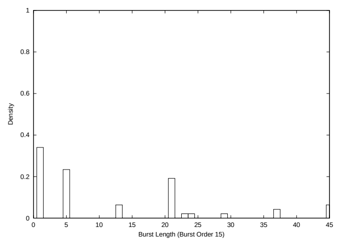
Trace 7, $E_k = 0.000123$



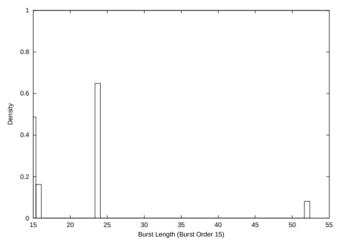
Trace 8, $E_k = 0.000231$



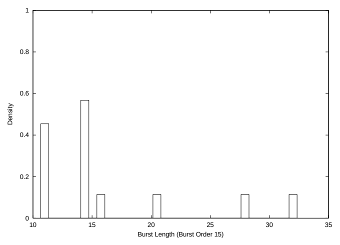
Trace 9, $E_k = 0.000119$



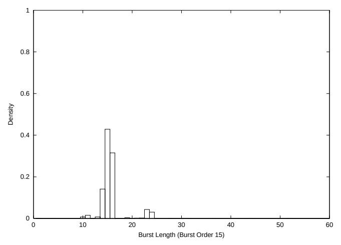
Trace 10, $E_k = 0.000154$



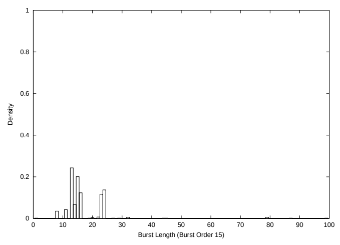
Trace 11, $E_k = 0.000002$



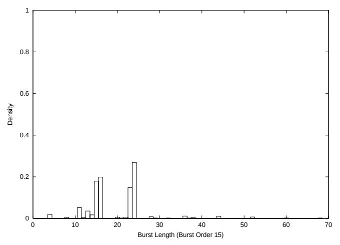
Trace 12, $E_k = 0.000001$



Trace 13, $E_k = 0.000315$

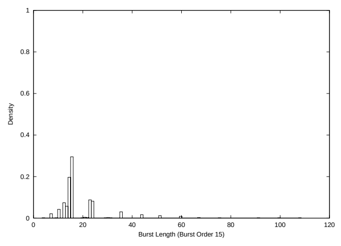


Trace 14, $E_k = 0.000036$

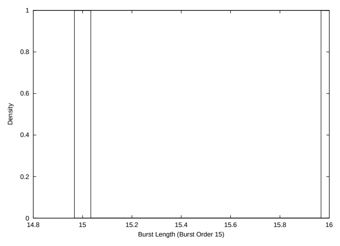


Trace 15, $E_k = 0.000070$

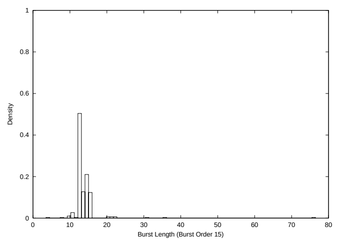
Table 3.8: Density of Error Burst Length (Order 15) for Traces 1 to 15



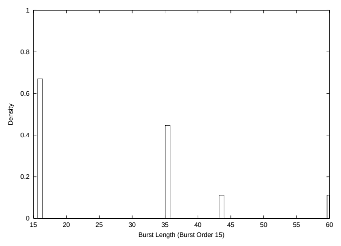
Trace 16, $E_k = 0.000027$



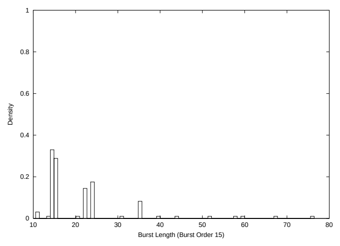
Trace 17, $E_k = 0.000000$



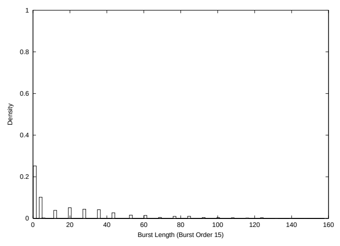
Trace 18, $E_k = 0.000008$



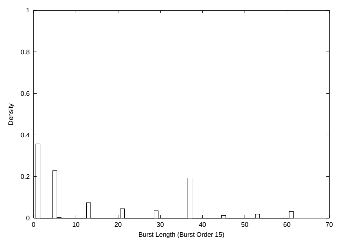
Trace 19, $E_k = 0.000001$



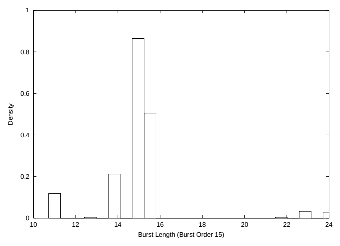
Trace 20, $E_k = 0.000006$



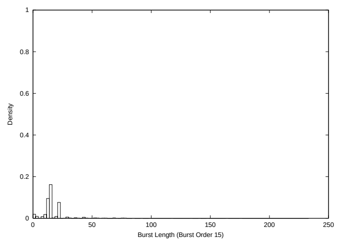
Trace 21, $E_k = 0.000098$



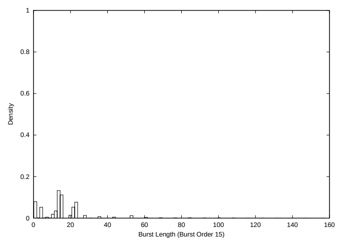
Trace 22, $E_k = 0.000003$



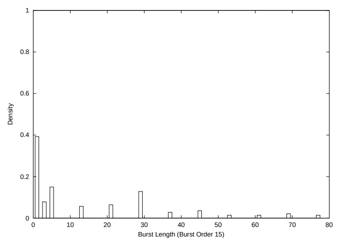
Trace 23, $E_k = 0.000007$



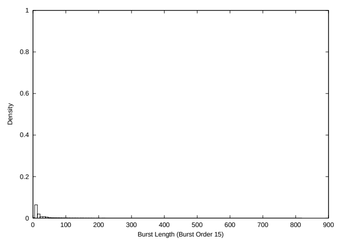
Trace 24, $E_k = 0.000442$



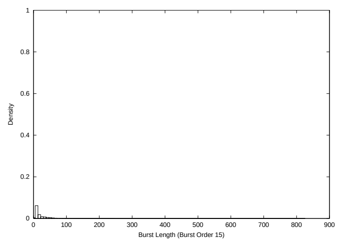
Trace 25, $E_k = 0.000404$



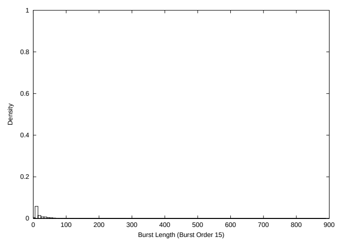
Trace 36, $E_k = 0.000281$



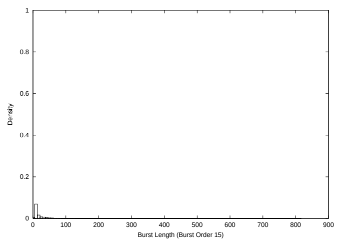
Trace 37, $E_k = 0.025511$



Trace 38, $E_k = 0.018970$

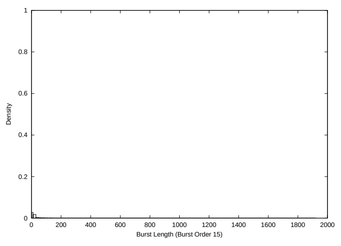


Trace 39, $E_k = 0.020942$

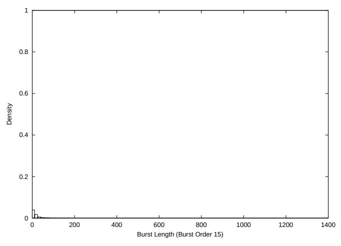


Trace 41, $E_k = 0.015353$

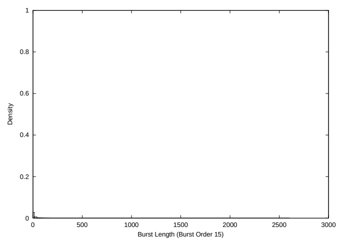
Table 3.9: Density of Error Burst Length (Order 15) for Traces 16 to 41



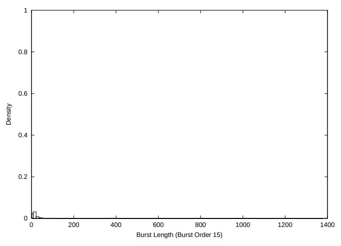
Trace 42, $E_k = 0.002740$



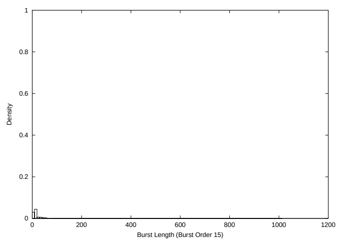
Trace 43, $E_k = 0.007271$



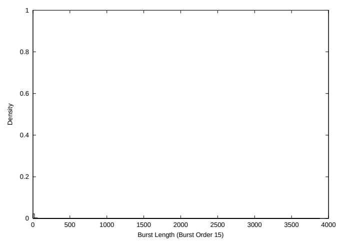
Trace 44, $E_k = 0.007188$



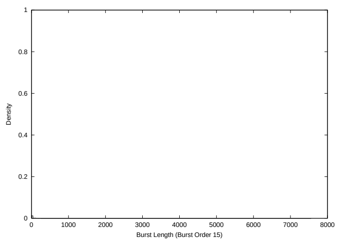
Trace 45, $E_k = 0.005575$



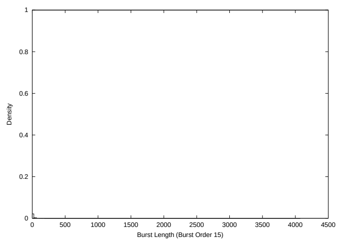
Trace 46, $E_k = 0.003861$



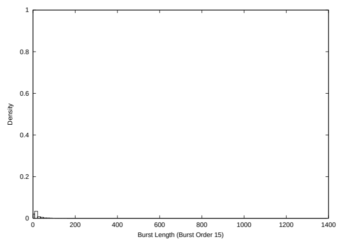
Trace 47, $E_k = 0.008375$



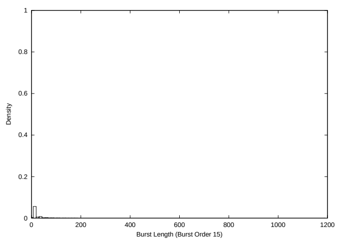
Trace 48, $E_k = 0.015173$



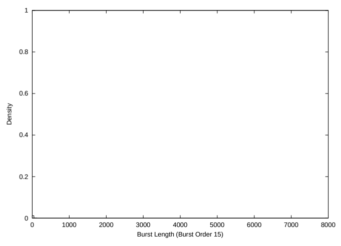
Trace 49, $E_k = 0.003788$



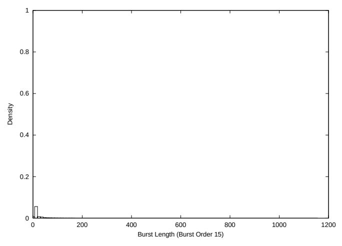
Trace 50, $E_k = 0.002321$



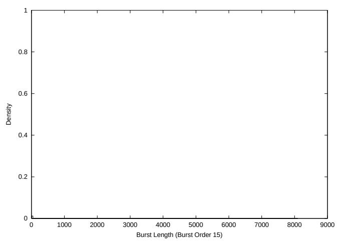
Trace 51, $E_k = 0.002682$



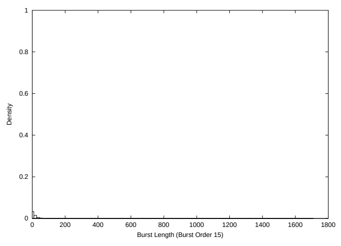
Trace 52, $E_k = 0.002499$



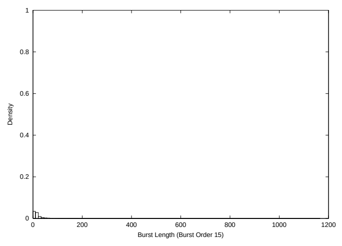
Trace 53, $E_k = 0.001536$



Trace 54, $E_k = 0.001844$

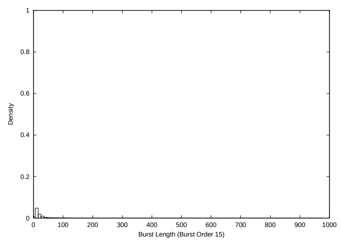


Trace 55, $E_k = 0.001441$

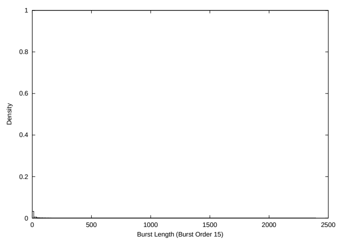


Trace 56, $E_k = 0.001608$

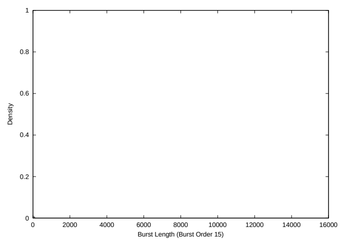
Table 3.10: Density of Error Burst Length (Order 15) for Traces 42 to 56



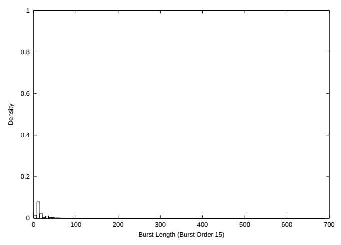
Trace 57, E_k = 0.000654



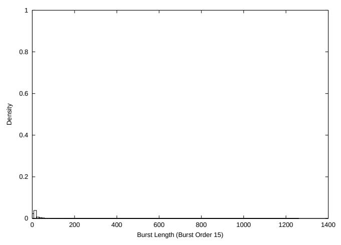
Trace 58, E_k = 0.001836



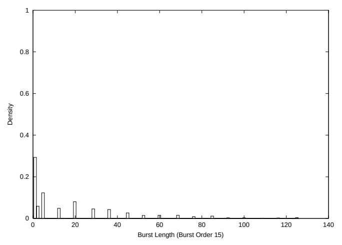
Trace 59, E_k = 0.000898



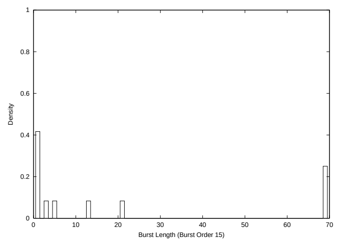
Trace 60, E_k = 0.000560



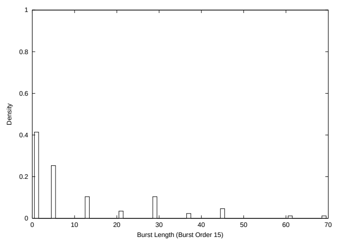
Trace 61, E_k = 0.000974



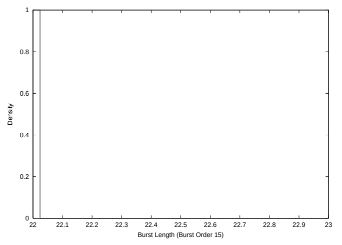
Trace 72, E_k = 0.000262



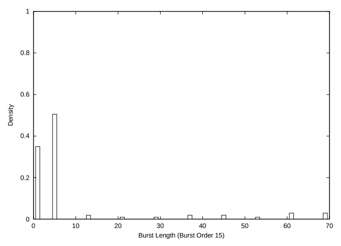
Trace 76, E_k = 0.000003



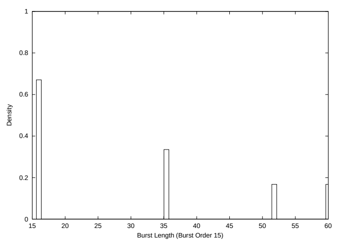
Trace 78, E_k = 0.000005



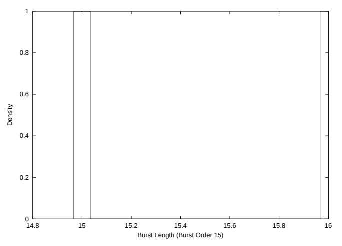
Trace 80, E_k = 0.000000



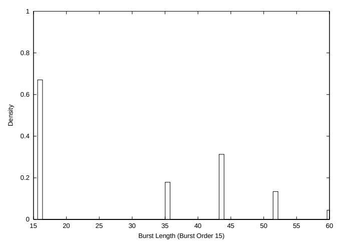
Trace 81, E_k = 0.000005



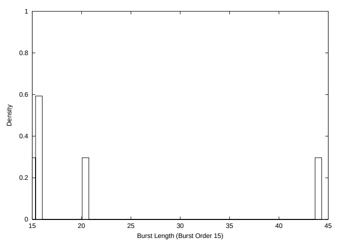
Trace 82, E_k = 0.000001



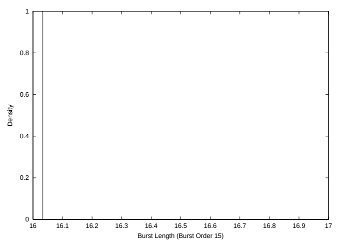
Trace 84, E_k = 0.000000



Trace 86, E_k = 0.000006

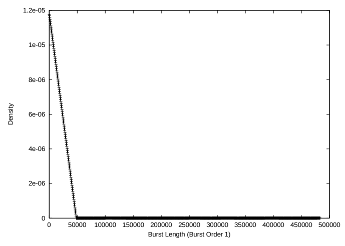


Trace 87, E_k = 0.000000

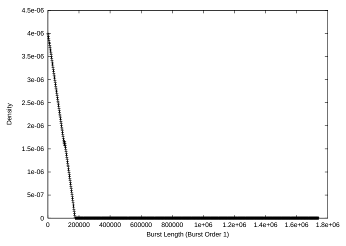


Trace 88, E_k = 0.000000

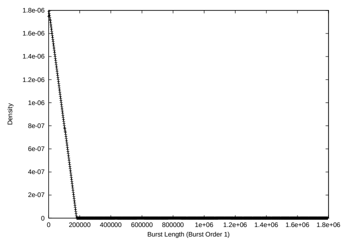
Table 3.11: Density of Error Burst Length (Order 15) for Traces 57 to 88



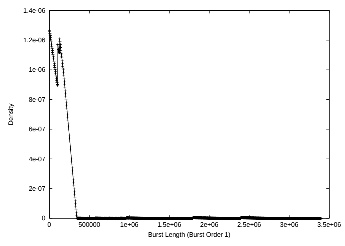
Trace 1, $E_k = 0.001276$



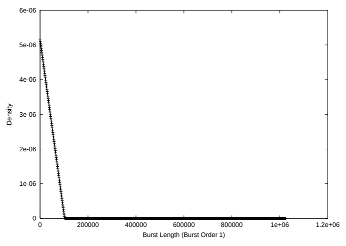
Trace 2, $E_k = 0.002051$



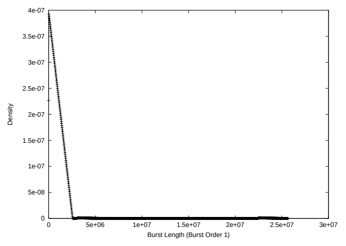
Trace 3, $E_k = 0.002734$



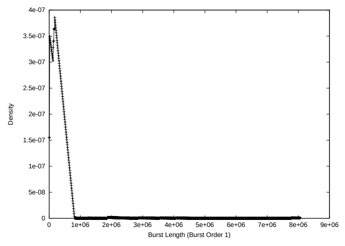
Trace 4, $E_k = 0.000134$



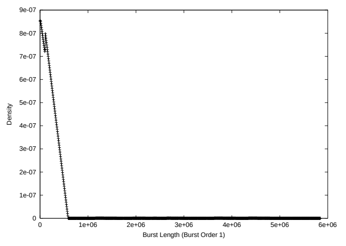
Trace 5, $E_k = 0.000739$



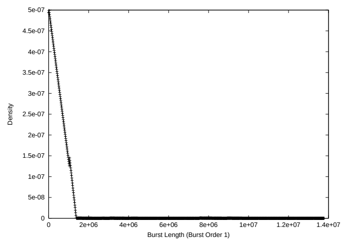
Trace 6, $E_k = 0.000015$



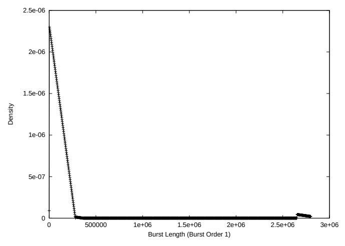
Trace 7, $E_k = 0.000123$



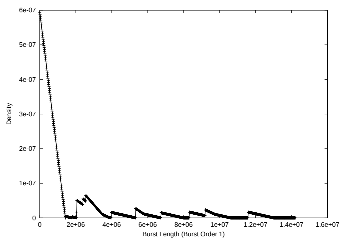
Trace 8, $E_k = 0.000231$



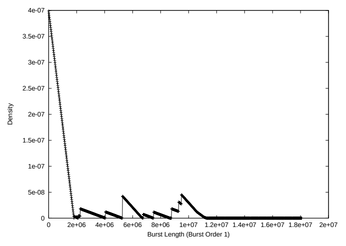
Trace 9, $E_k = 0.000119$



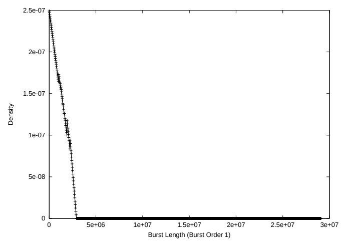
Trace 10, $E_k = 0.000154$



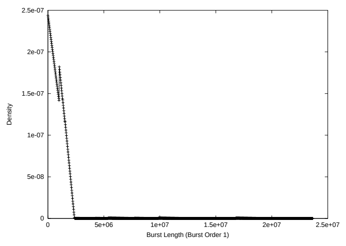
Trace 11, $E_k = 0.000002$



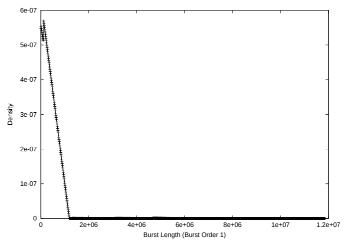
Trace 12, $E_k = 0.000001$



Trace 13, $E_k = 0.000315$

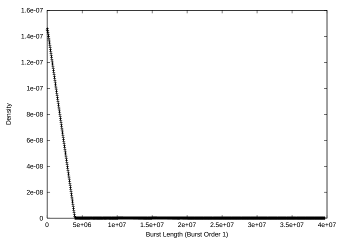


Trace 14, $E_k = 0.000036$

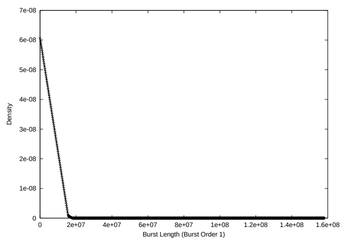


Trace 15, $E_k = 0.000070$

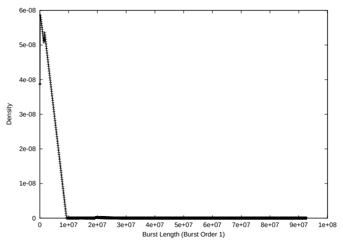
Table 3.12: Smoothed density of Errorfree Burst Length (Order 1) for Traces 1 to 15



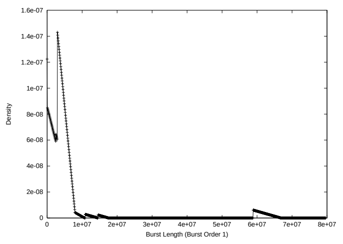
Trace 16, $E_k = 0.000027$



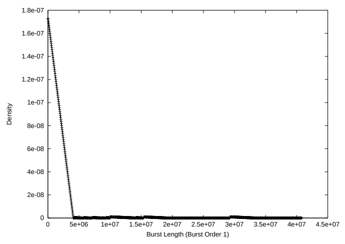
Trace 17, $E_k = 0.000000$



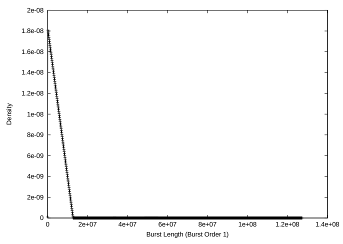
Trace 18, $E_k = 0.000008$



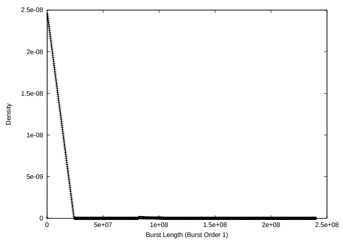
Trace 19, $E_k = 0.000001$



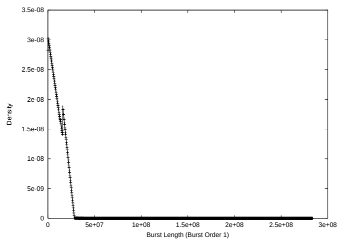
Trace 20, $E_k = 0.000006$



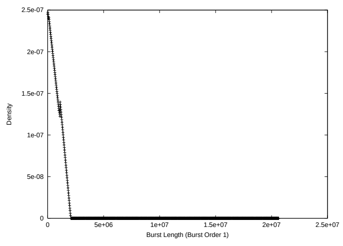
Trace 21, $E_k = 0.000098$



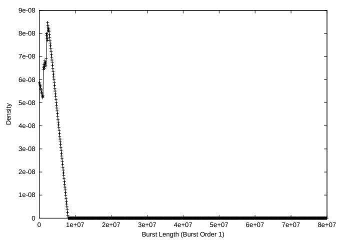
Trace 22, $E_k = 0.000003$



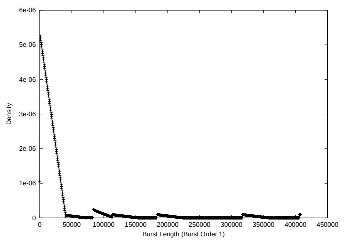
Trace 23, $E_k = 0.000007$



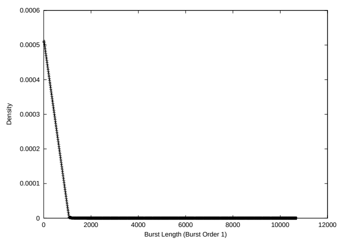
Trace 24, $E_k = 0.000442$



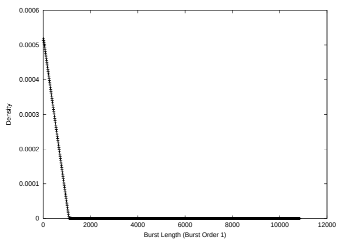
Trace 25, $E_k = 0.000404$



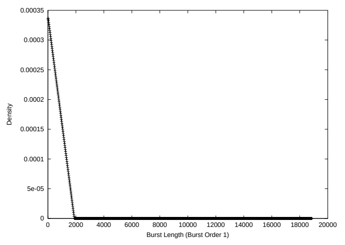
Trace 36, $E_k = 0.000281$



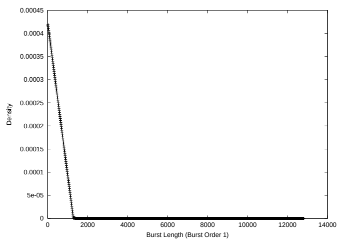
Trace 37, $E_k = 0.025511$



Trace 38, $E_k = 0.018970$

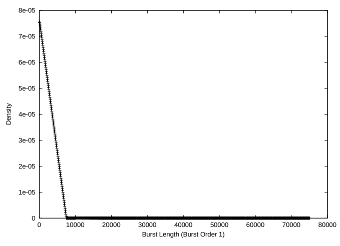


Trace 39, $E_k = 0.020942$

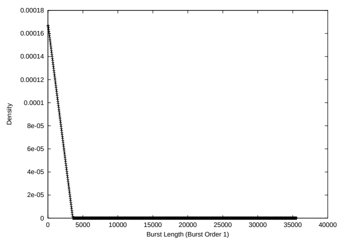


Trace 41, $E_k = 0.015353$

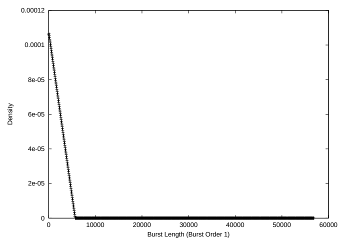
Table 3.13: Smoothed density of Errorfree Burst Length (Order 1) for Traces 16 to 41



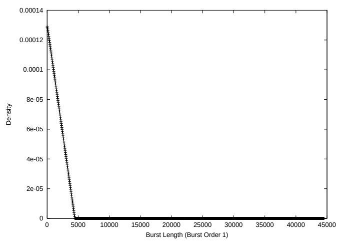
Trace 42, $E_k = 0.002740$



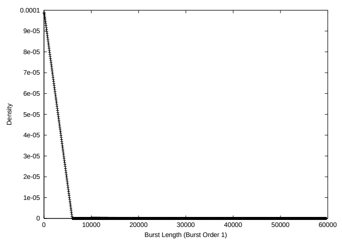
Trace 43, $E_k = 0.007271$



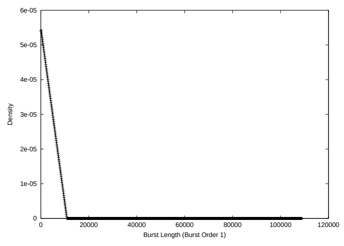
Trace 44, $E_k = 0.007188$



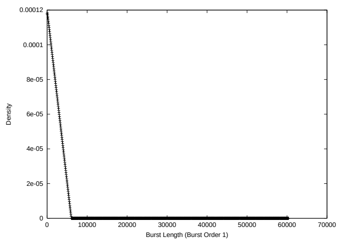
Trace 45, $E_k = 0.005575$



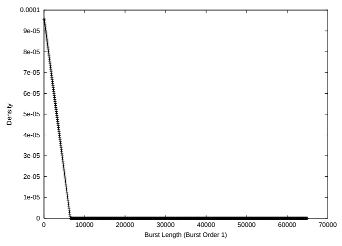
Trace 46, $E_k = 0.003861$



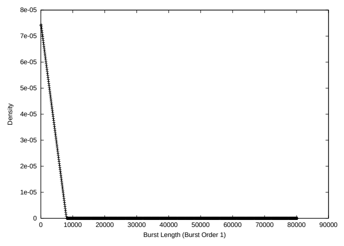
Trace 47, $E_k = 0.008375$



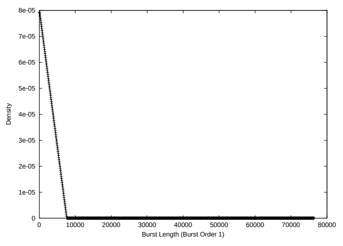
Trace 48, $E_k = 0.015173$



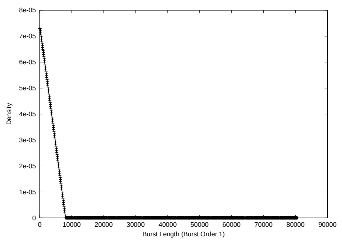
Trace 49, $E_k = 0.003788$



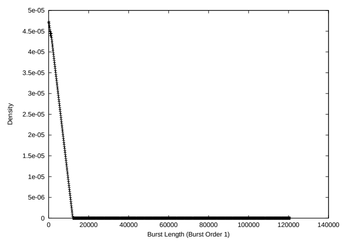
Trace 50, $E_k = 0.002321$



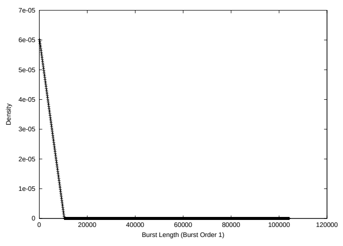
Trace 51, $E_k = 0.002682$



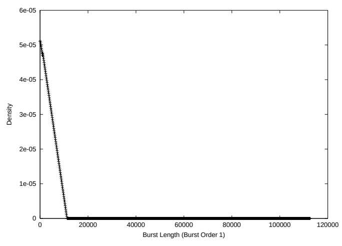
Trace 52, $E_k = 0.002499$



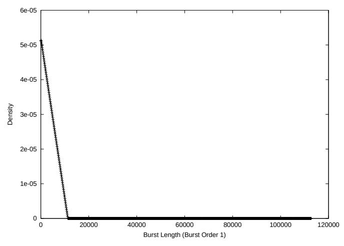
Trace 53, $E_k = 0.001536$



Trace 54, $E_k = 0.001844$

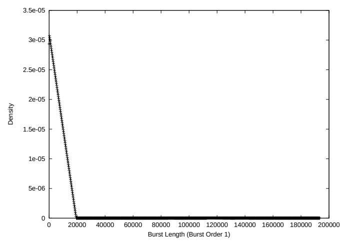


Trace 55, $E_k = 0.001441$

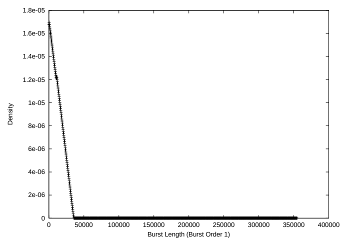


Trace 56, $E_k = 0.001608$

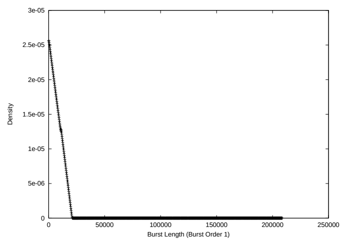
Table 3.14: Smoothed density of Errorfree Burst Length (Order 1) for Traces 42 to 56



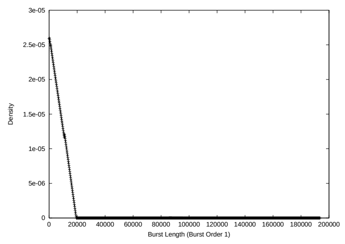
Trace 57, E_k = 0.000654



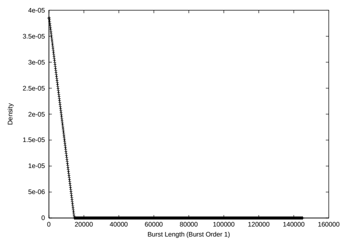
Trace 58, E_k = 0.001836



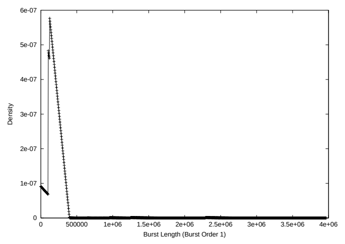
Trace 59, E_k = 0.000898



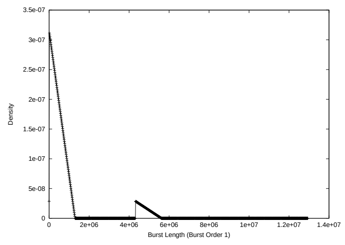
Trace 60, E_k = 0.000560



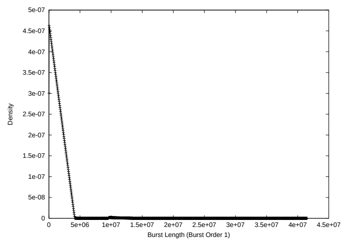
Trace 61, E_k = 0.000974



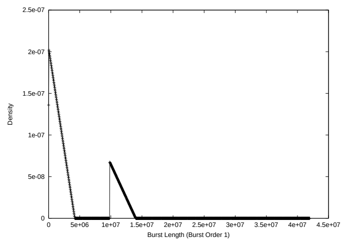
Trace 72, E_k = 0.000262



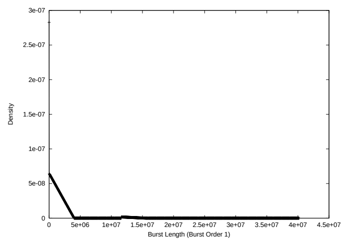
Trace 76, E_k = 0.000003



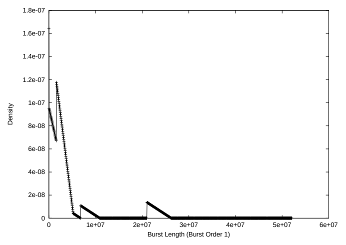
Trace 78, E_k = 0.000005



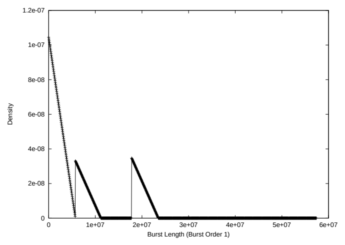
Trace 80, E_k = 0.000000



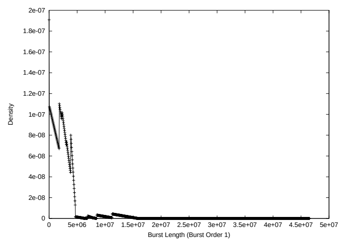
Trace 81, E_k = 0.000005



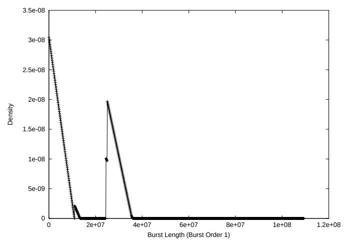
Trace 82, E_k = 0.000001



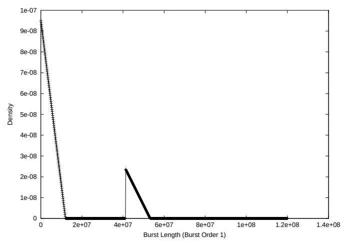
Trace 84, E_k = 0.000000



Trace 86, E_k = 0.000006

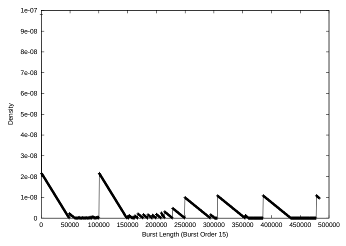


Trace 87, E_k = 0.000000

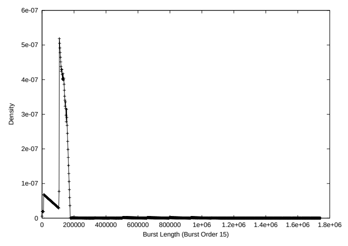


Trace 88, E_k = 0.000000

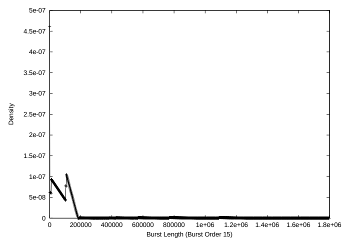
Table 3.15: Smoothed density of Errorfree Burst Length (Order 1) for Traces 57 to 88



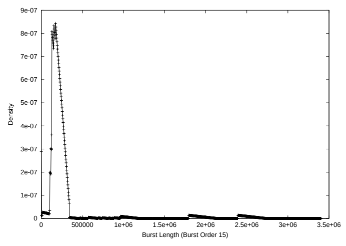
Trace 1, $E_k = 0.001276$



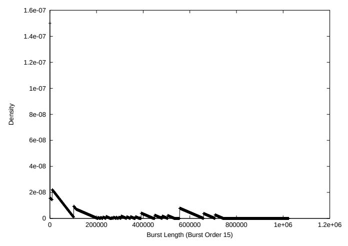
Trace 2, $E_k = 0.002051$



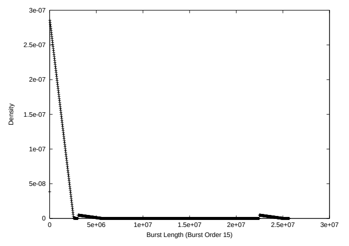
Trace 3, $E_k = 0.002734$



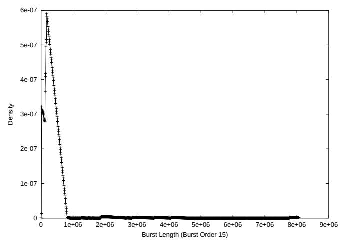
Trace 4, $E_k = 0.000134$



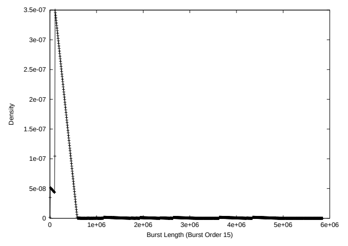
Trace 5, $E_k = 0.000739$



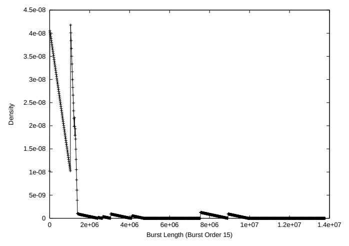
Trace 6, $E_k = 0.000015$



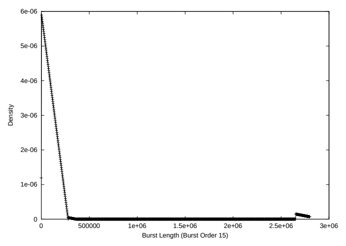
Trace 7, $E_k = 0.000123$



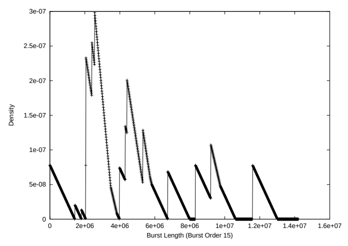
Trace 8, $E_k = 0.000231$



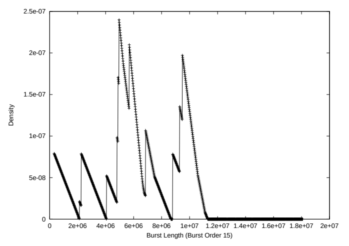
Trace 9, $E_k = 0.000119$



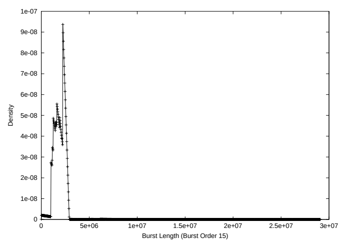
Trace 10, $E_k = 0.000154$



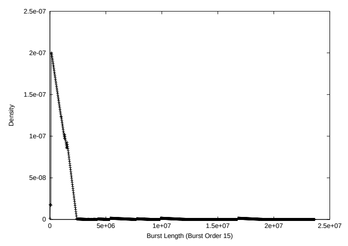
Trace 11, $E_k = 0.000002$



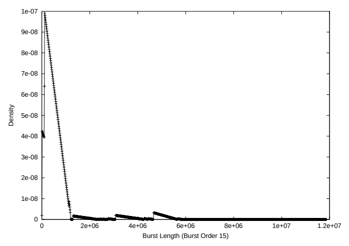
Trace 12, $E_k = 0.000001$



Trace 13, $E_k = 0.000315$

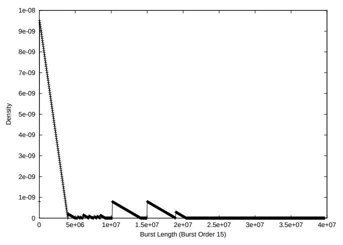


Trace 14, $E_k = 0.000036$

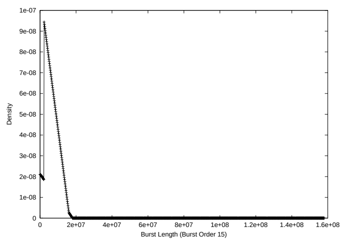


Trace 15, $E_k = 0.000070$

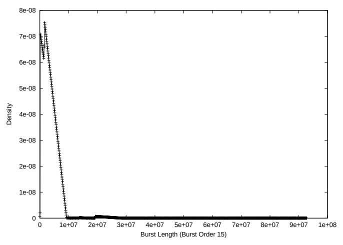
Table 3.16: Smoothed density of Errorfree Burst Length (Order 15) for Traces 1 to 15



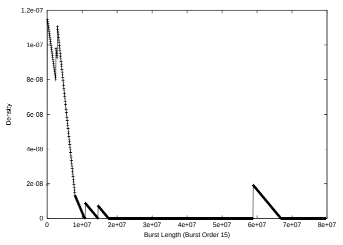
Trace 16, $E_k = 0.000027$



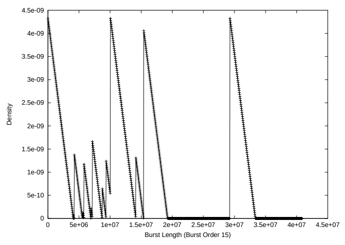
Trace 17, $E_k = 0.000000$



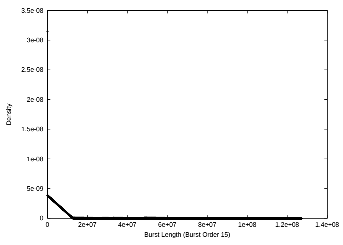
Trace 18, $E_k = 0.000008$



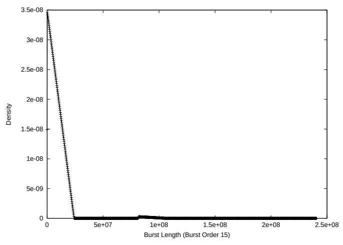
Trace 19, $E_k = 0.000001$



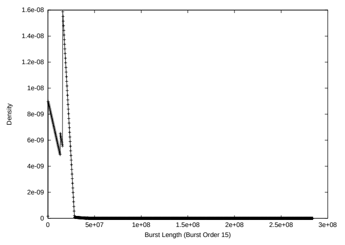
Trace 20, $E_k = 0.000006$



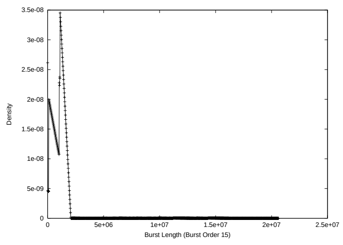
Trace 21, $E_k = 0.000098$



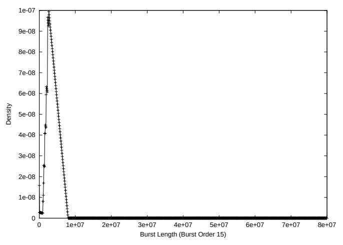
Trace 22, $E_k = 0.000003$



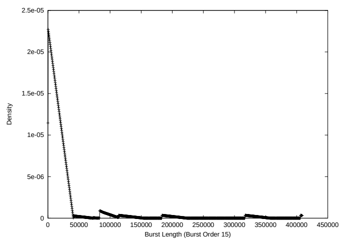
Trace 23, $E_k = 0.000007$



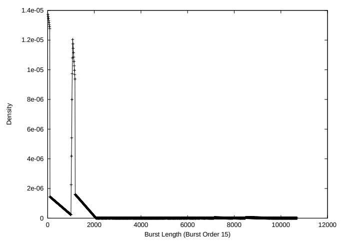
Trace 24, $E_k = 0.000442$



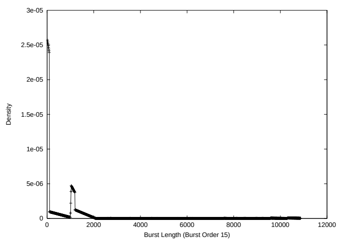
Trace 25, $E_k = 0.000404$



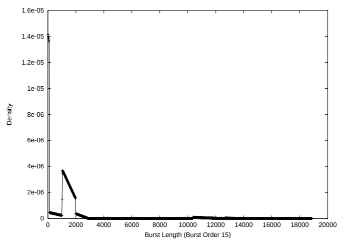
Trace 36, $E_k = 0.000281$



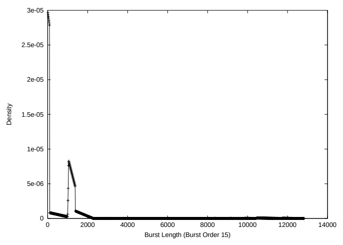
Trace 37, $E_k = 0.025511$



Trace 38, $E_k = 0.018970$

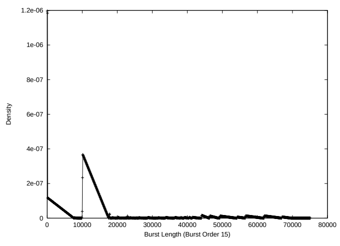


Trace 39, $E_k = 0.020942$

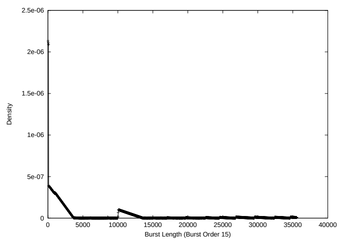


Trace 41, $E_k = 0.015353$

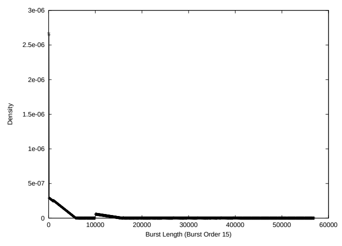
Table 3.17: Smoothed density of Errorfree Burst Length (Order 15) for Traces 16 to 41



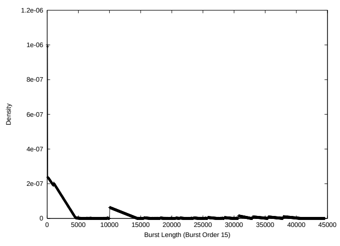
Trace 42, E_k = 0.002740



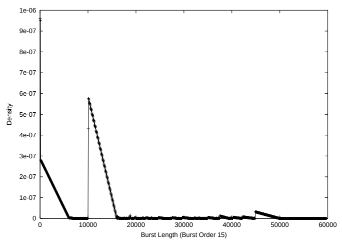
Trace 43, E_k = 0.007271



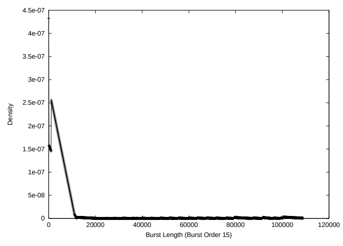
Trace 44, E_k = 0.007188



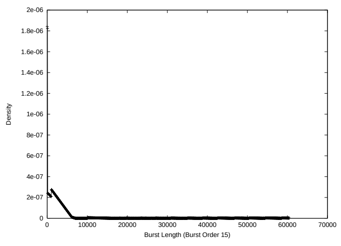
Trace 45, E_k = 0.005575



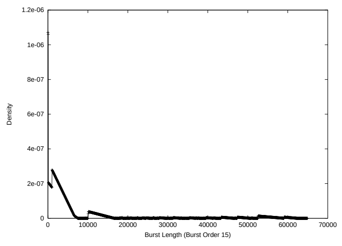
Trace 46, E_k = 0.003861



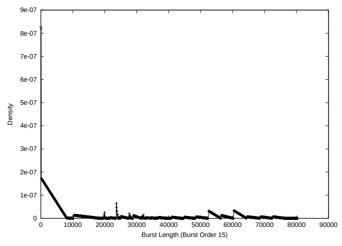
Trace 47, E_k = 0.008375



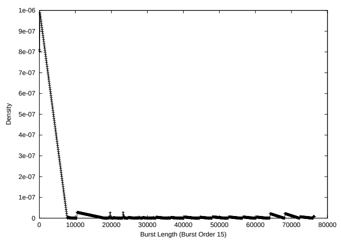
Trace 48, E_k = 0.015173



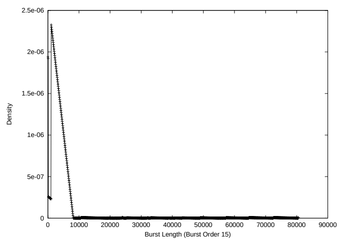
Trace 49, E_k = 0.003788



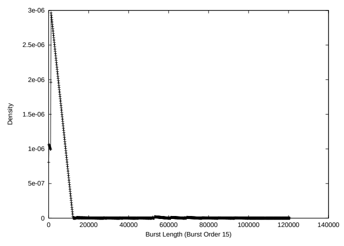
Trace 50, E_k = 0.002321



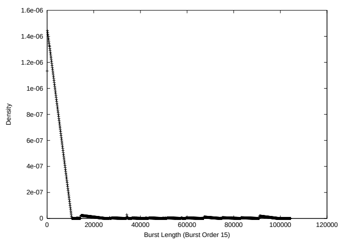
Trace 51, E_k = 0.002682



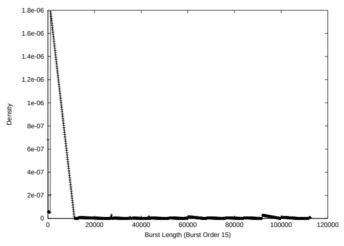
Trace 52, E_k = 0.002499



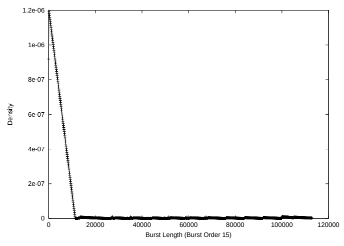
Trace 53, E_k = 0.001536



Trace 54, E_k = 0.001844

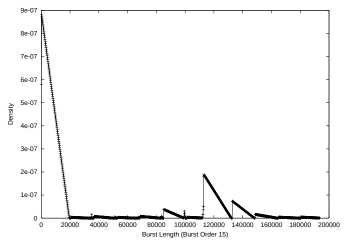


Trace 55, E_k = 0.001441

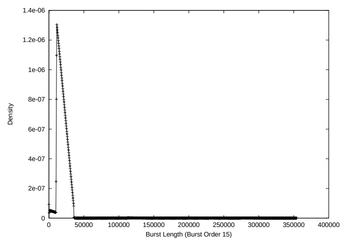


Trace 56, E_k = 0.001608

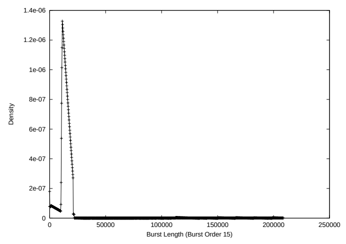
Table 3.18: Smoothed density of Errorfree Burst Length (Order 15) for Traces 42 to 56



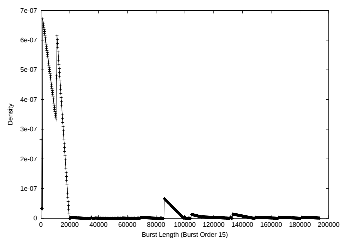
Trace 57, E_k = 0.000654



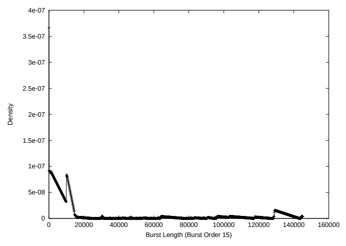
Trace 58, E_k = 0.001836



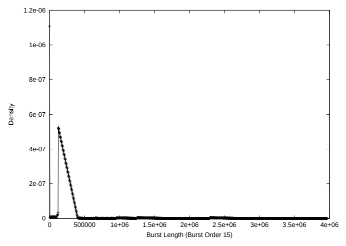
Trace 59, E_k = 0.000898



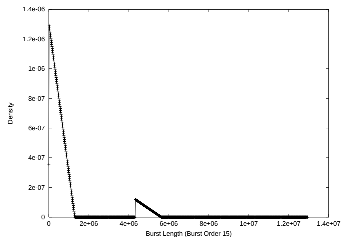
Trace 60, E_k = 0.000560



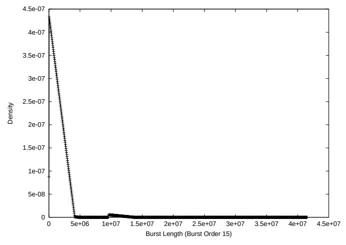
Trace 61, E_k = 0.000974



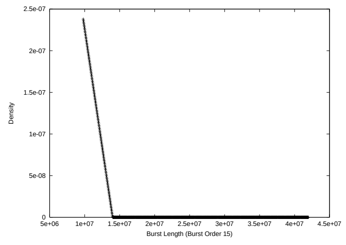
Trace 72, E_k = 0.000262



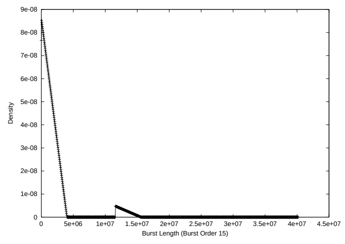
Trace 76, E_k = 0.000003



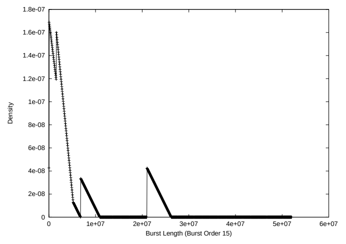
Trace 78, E_k = 0.000005



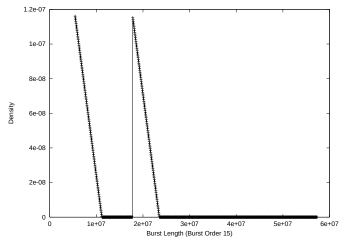
Trace 80, E_k = 0.000000



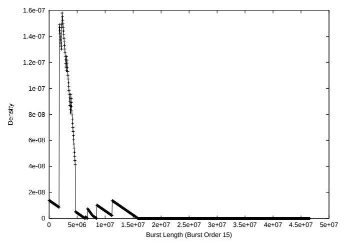
Trace 81, E_k = 0.000005



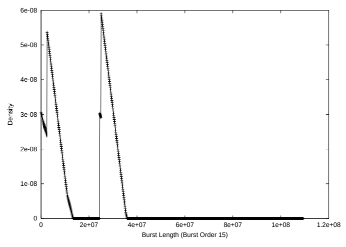
Trace 82, E_k = 0.000001



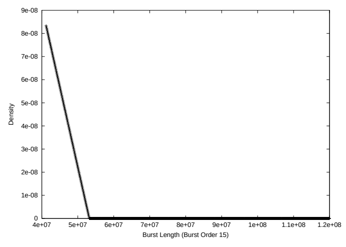
Trace 84, E_k = 0.000000



Trace 86, E_k = 0.000006



Trace 87, E_k = 0.000000



Trace 88, E_k = 0.000000

Table 3.19: Smoothed density of Errorfree Burst Length (Order 15) for Traces 57 to 88

3.2.3 Second Order Statistics

We now turn to some second order statistics of the measured traces. These allow to gain some insights into the correlation structure of the errors occuring on the medium. We look at this at the bit level and at the packet level, where we consider a packet as erroneous, when at least one bit within this packet is erroneous[3].

First we discuss the conditional bit error probability $\Pr[i_{n+s} = 1 | i_n = 1]$ as described in section 2.2.2. The corresponding curves are shown in tables 3.20, 3.21, 3.22, 3.23[4]. Since in all cases the mean BERs are small enough, we can use $\Pr[i_{n+s} = 1 | i_n = 1]$ as a good approximation to the correlation function, as discussed in section 2.2.2. We can make the following observations:

- All traces with CCK modulation (traces 37 to 71) show the same behaviour of slow and monotonic decaying and a large peak for small lags. On very short distances of a few bits (up to 20) the bit errors appear to be strongly correlated, while correlation rapidly drops to levels below 0.2 and thus are weakly correlated, but much above the mean bit error level. In traces 37 to 41 we observe a small peak between lag 800 and 900. The explanation is that these traces correspond to a packet size of 108 bytes (*NumChunks* equals three), giving $108 \cdot 8 = 864$ bits, and that bit errors tend to occur at the start of a packet[5].

- For QPSK traces we can observe, that for small lags (20 to 30 bits) there is strong correlation, however, this decays rapidly. The remaining behaviour is much more diverse than in the CCK case (for small bit error rates the curves do not have much meaning).

- Some QPSK traces show a nice repeated triangular shape, e.g. traces $k = 6$, $k = 10$, $k = 21$, $k = 22$, $k = 25$, $k = 72$, $k = 76$, $k = 78$ and $k = 81$. The period observed is roughly 300 bits, which can be obtained by multiplying the chunk size of 36 bytes with eight (288). After inspection of the corresponding traces we find that these traces contain packets with bit shifts (as explained in section 2.2.1). We have selected two

[3]In future studies one may use different criteria for whether a packet is erroneous, e.g. based on the residual errors of different error correcting codes.

[4]Please note that every figure is annotated with the corresponding mean bit error rate, however, due to presentation with limited precision, occasionally the value zero occurs. In the corresponding traces bit errors show up, but with a rate smaller 10^{-6}

[5]It might be fruitful to evaluate all traces after the first 1000 bits of each packet are removed (see also section 3.2.1), and thus with the influence of error clustering at packets beginning disappeared.

such traces (trace number 6 and trace number 25) and removed all packets with bit shifts. The result was that the triangular shape disapears.

- Most QPSK traces with a sufficiently high error rate show some periodicity of roughly 128 bits (e.g. traces $k = 1$, $k = 2$, $k = 8$ and $k = 9$), often with alternating high and low peaks, and decaying amplitude of the high peaks (such a behaviour cannot be observed for CCK traces). We currently have no satisfactorily explanation for this behaviour, however, we think that this may due to the operation of the scrambler. In a future measurement campaign for this reason we plan to perform measurements with scrambling disabled.

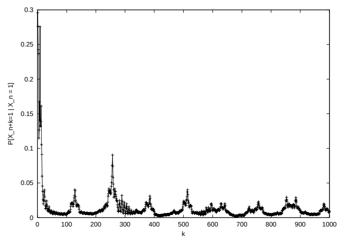
As a conclusion we can state that for CCK modulation the errors show much correlation over comparably long distances, thus errors tend to be clustered. For QPSK things are more diverse: we can observe a strong short term correlation, however, for larger distances correlation often disappears (except the periodicity observed and the effects given by bit shifts).

In tables 3.24, 3.25, 3.26 and 3.27 we show the conditional probability that packet $n + l$ is erroneous, given that packet n is erroneous. The method of estimation is the same as for the bit error probabilities (see section 2.2.2). The following observations are interesting:

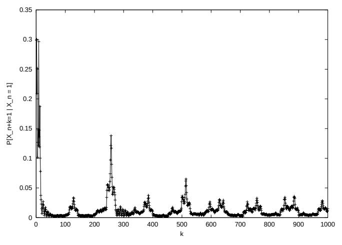
- For CCK traces this probability is nearly constant and close to the overall PER. Thus packet errors seem to occur nearly independent from each other, since

$$\Pr[\text{Packet } n + k \text{ erroneous}] \approx \Pr[\text{Packet } n + k \text{ erroneous}|\text{Packet } n \text{ erroneous}]$$

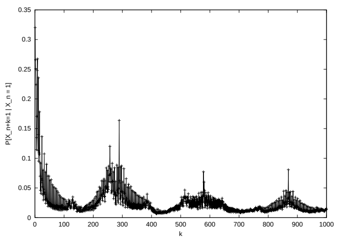
- For QPSK traces this probability is decaying slowly, but remains much beyond the overall PER level. Thus packet errors tend to occur clustered.



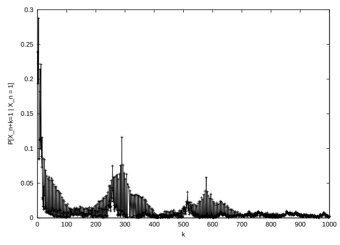
Trace 1, $E_k = 0.001276$



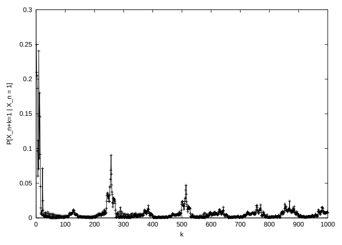
Trace 2, $E_k = 0.002051$



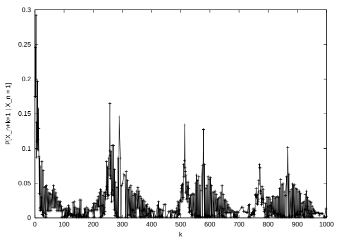
Trace 3, $E_k = 0.002734$



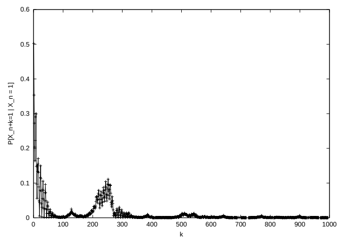
Trace 4, $E_k = 0.000134$



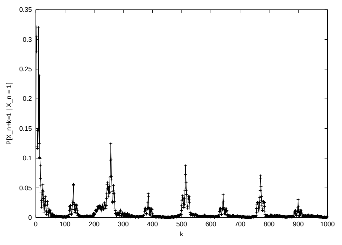
Trace 5, $E_k = 0.000739$



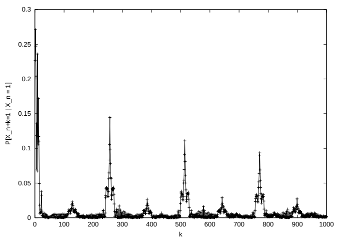
Trace 6, $E_k = 0.000015$



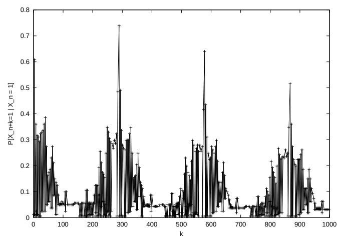
Trace 7, $E_k = 0.000123$



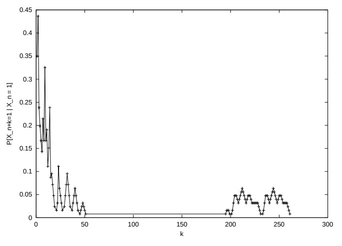
Trace 8, $E_k = 0.000231$



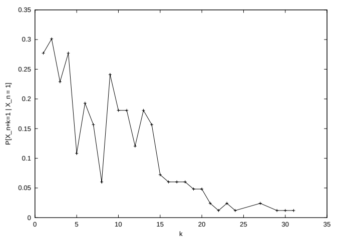
Trace 9, $E_k = 0.000119$



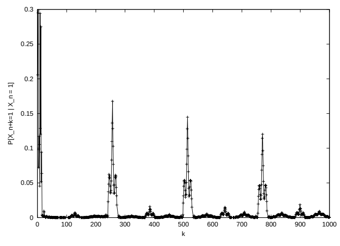
Trace 10, $E_k = 0.000154$



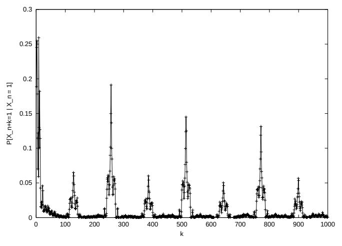
Trace 11, $E_k = 0.000002$



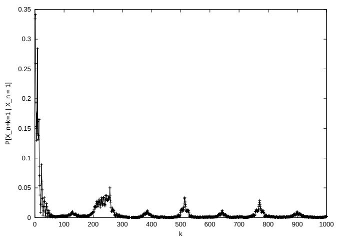
Trace 12, $E_k = 0.000001$



Trace 13, $E_k = 0.000315$

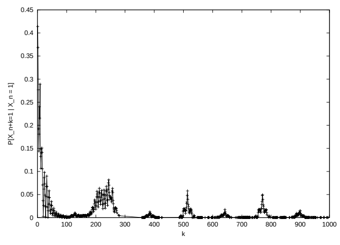


Trace 14, $E_k = 0.000036$

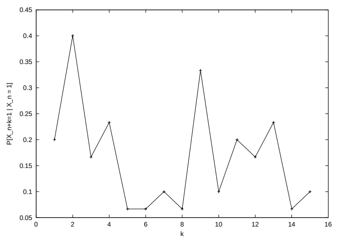


Trace 15, $E_k = 0.000070$

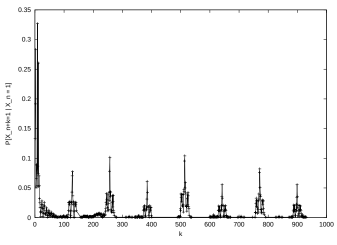
Table 3.20: Conditional Error Probabilities for Traces 1 to 15



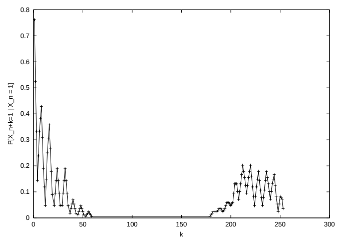
Trace 16, $E_k = 0.000027$



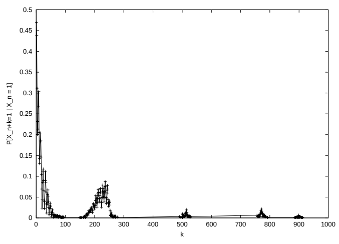
Trace 17, $E_k = 0.000000$



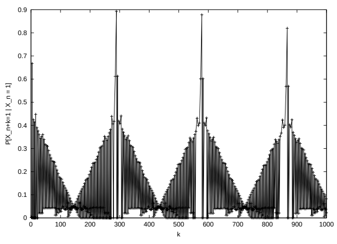
Trace 18, $E_k = 0.000008$



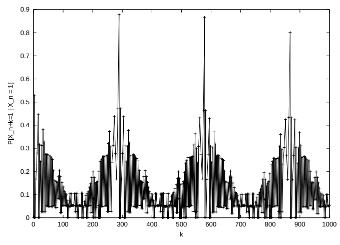
Trace 19, $E_k = 0.000001$



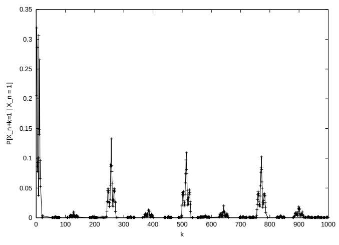
Trace 20, $E_k = 0.000006$



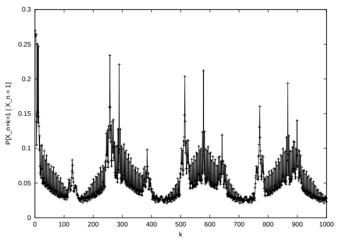
Trace 21, $E_k = 0.000098$



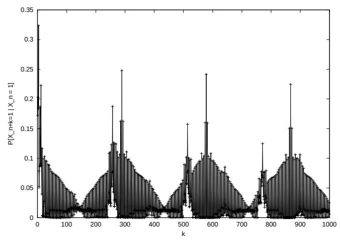
Trace 22, $E_k = 0.000003$



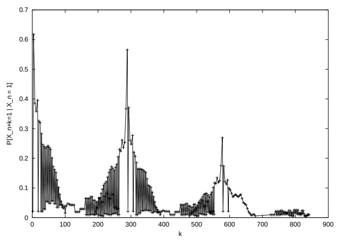
Trace 23, $E_k = 0.000007$



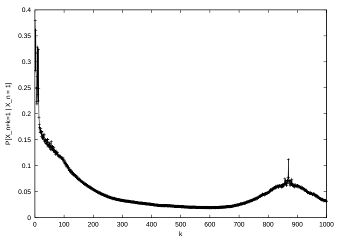
Trace 24, $E_k = 0.000442$



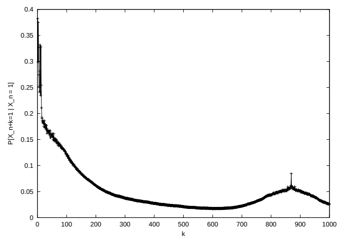
Trace 25, $E_k = 0.000404$



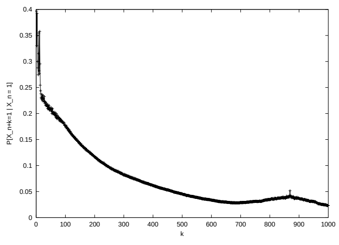
Trace 36, $E_k = 0.000281$



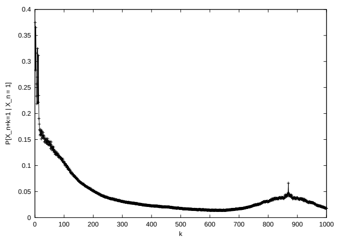
Trace 37, $E_k = 0.025511$



Trace 38, $E_k = 0.018970$

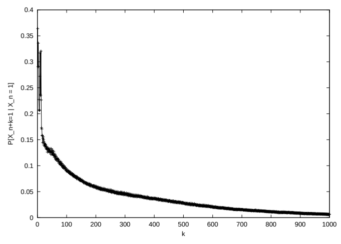


Trace 39, $E_k = 0.020942$

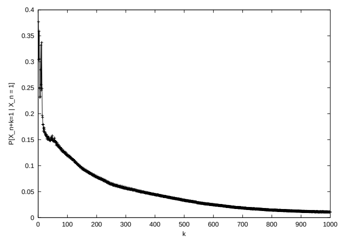


Trace 41, $E_k = 0.015353$

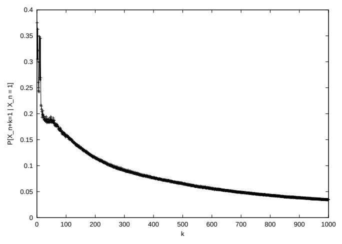
Table 3.21: Conditional Error Probabilities for Traces 16 to 41



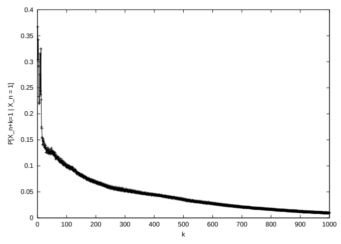
Trace 42, $E_k = 0.002740$



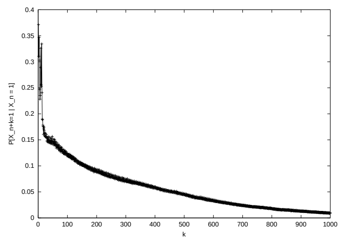
Trace 43, $E_k = 0.007271$



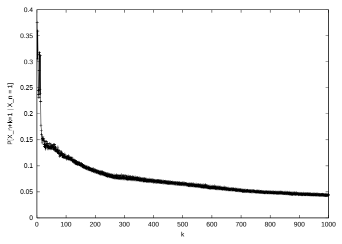
Trace 44, $E_k = 0.007188$



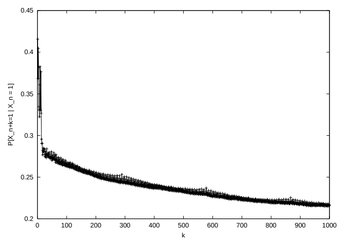
Trace 45, $E_k = 0.005575$



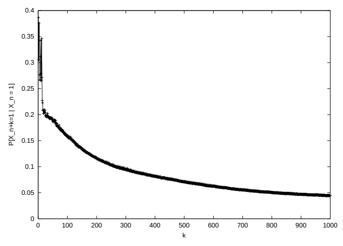
Trace 46, $E_k = 0.003861$



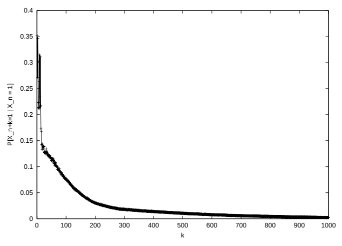
Trace 47, $E_k = 0.008375$



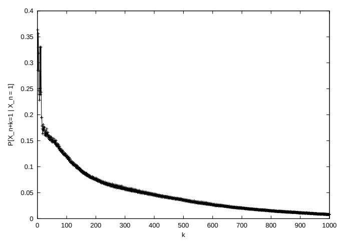
Trace 48, $E_k = 0.015173$



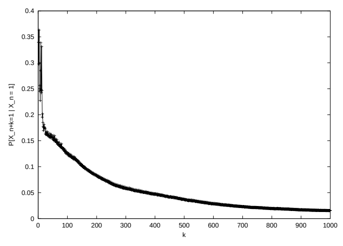
Trace 49, $E_k = 0.003788$



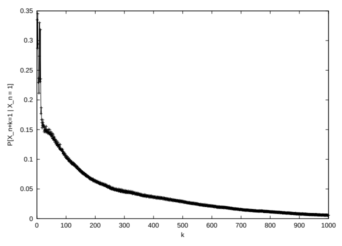
Trace 50, $E_k = 0.002321$



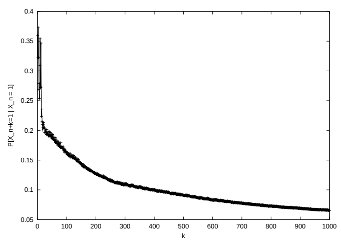
Trace 51, $E_k = 0.002682$



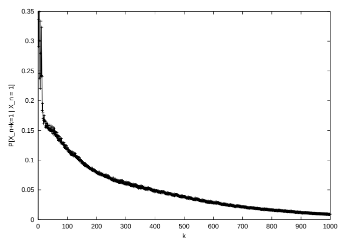
Trace 52, $E_k = 0.002499$



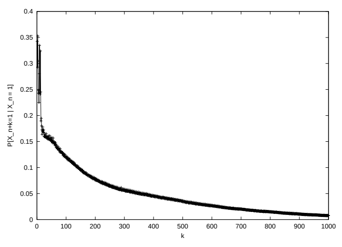
Trace 53, $E_k = 0.001536$



Trace 54, $E_k = 0.001844$

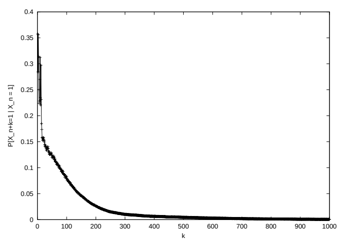


Trace 55, $E_k = 0.001441$

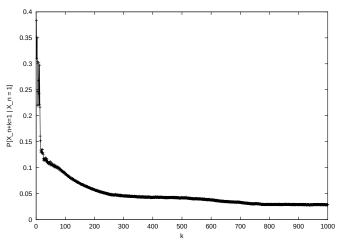


Trace 56, $E_k = 0.001608$

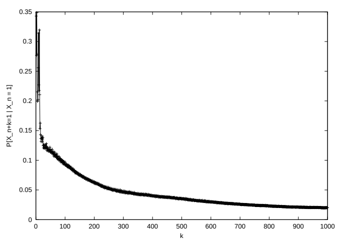
Table 3.22: Conditional Error Probabilities for Traces 42 to 56



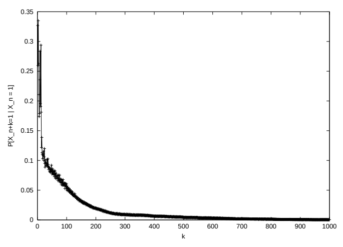
Trace 57, $E_k = 0.000654$



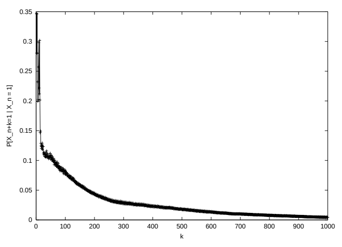
Trace 58, $E_k = 0.001836$



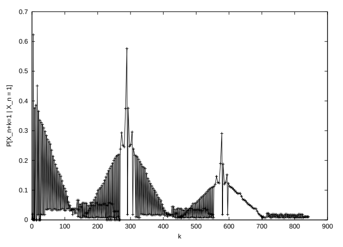
Trace 59, $E_k = 0.000898$



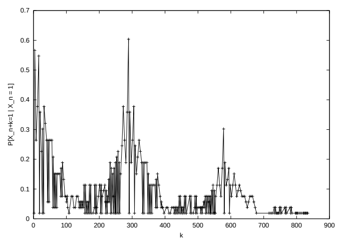
Trace 60, $E_k = 0.000560$



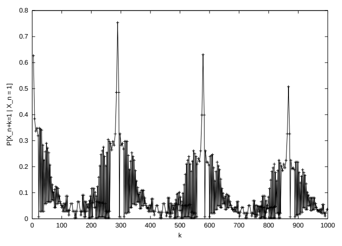
Trace 61, $E_k = 0.000974$



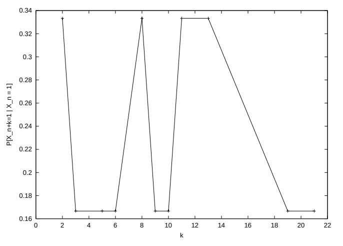
Trace 72, $E_k = 0.000262$



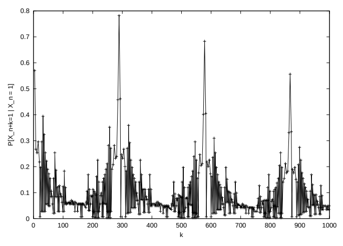
Trace 76, $E_k = 0.000003$



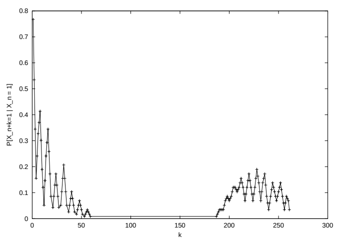
Trace 78, $E_k = 0.000005$



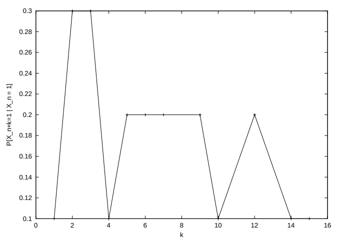
Trace 80, $E_k = 0.000000$



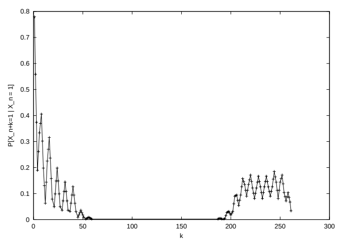
Trace 81, $E_k = 0.000005$



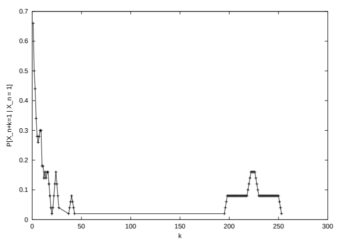
Trace 82, $E_k = 0.000001$



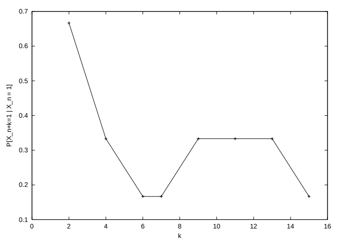
Trace 84, $E_k = 0.000000$



Trace 86, $E_k = 0.000006$

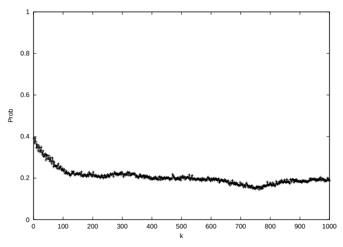


Trace 87, $E_k = 0.000000$

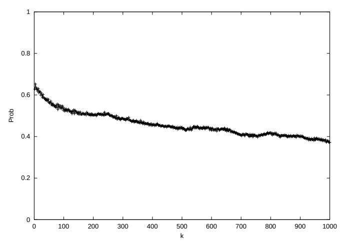


Trace 88, $E_k = 0.000000$

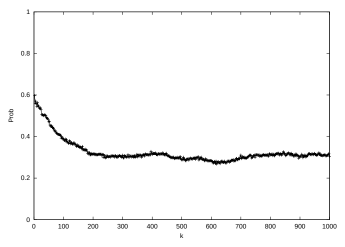
Table 3.23: Conditional Error Probabilities for Traces 57 to 88



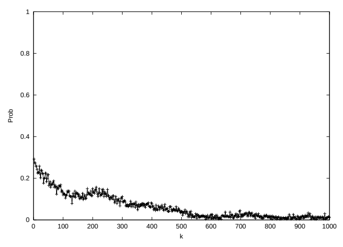
Trace 1, PER = 0.1202



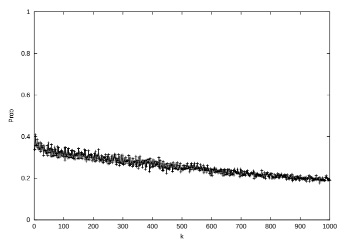
Trace 2, PER = 0.1573



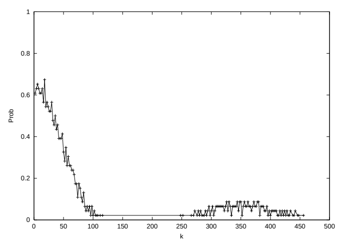
Trace 3, PER = 0.1651



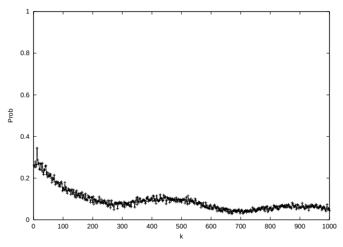
Trace 4, PER =0.0134



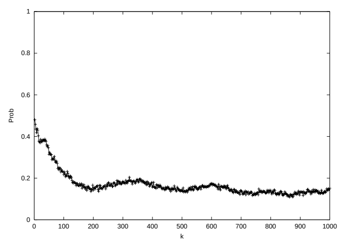
Trace 5, PER = 0.0846



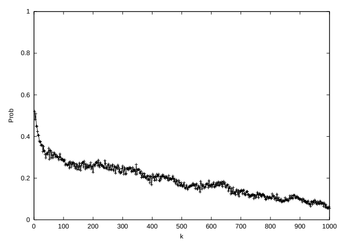
Trace 6, PER = 0.0023



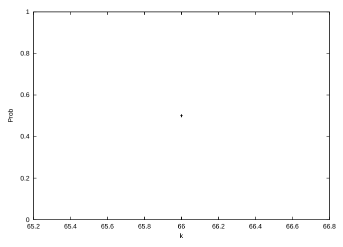
Trace 7, PER = 0.0316



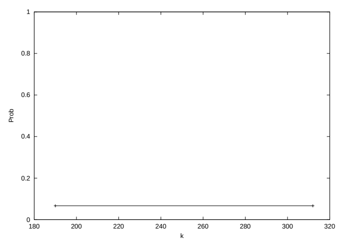
Trace 8, PER = 0.0425



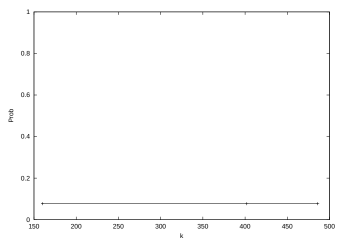
Trace 9, PER = 0.0217



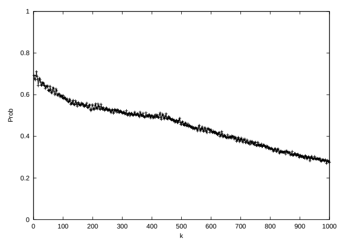
Trace 10, PER = 0.0009



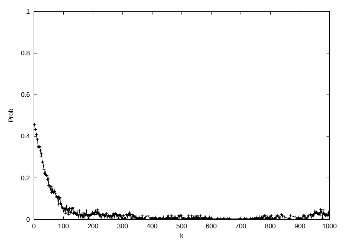
Trace 11, PER = 0.0008



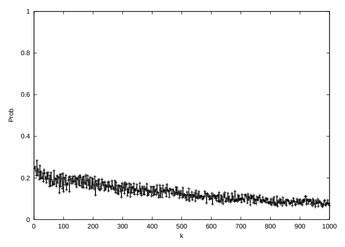
Trace 12, PER = 0.0006



Trace 13, PER = 0.0627

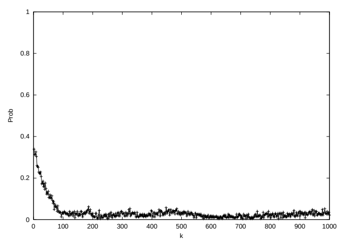


Trace 14, PER = 0.0085

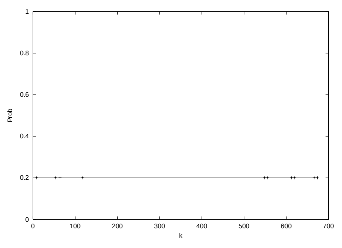


Trace 15, PER = 0.0276

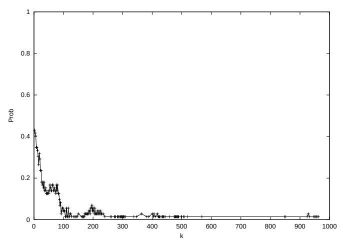
Table 3.24: Conditional Packet Error Probabilities for Traces 1 to 15



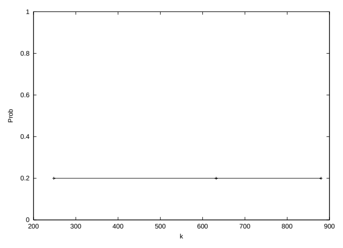
Trace 16, PER = 0.0114



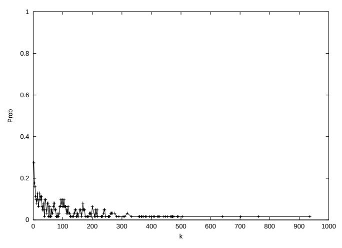
Trace 17, PER = 0.0002



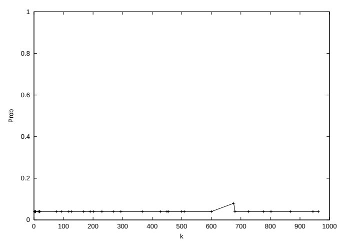
Trace 18, PER = 0.0036



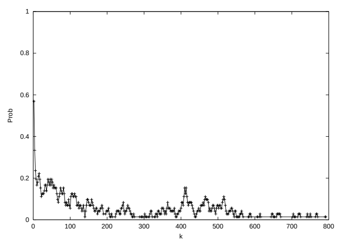
Trace 19, PER = 0.0003



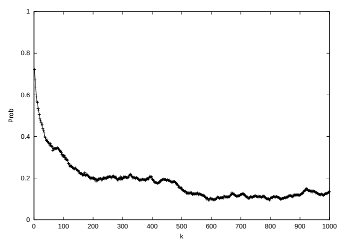
Trace 20, PER = 0.0031



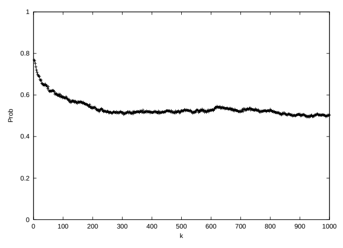
Trace 21, PER = 0.0014



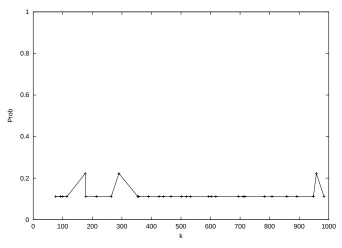
Trace 23, PER = 0.0036



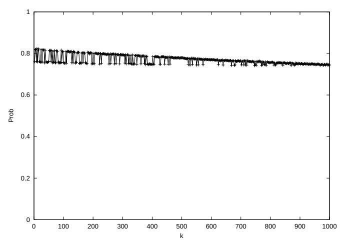
Trace 24, PER = 0.0937



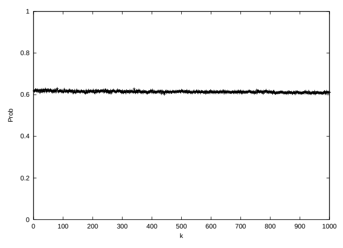
Trace 25, PER = 0.1036



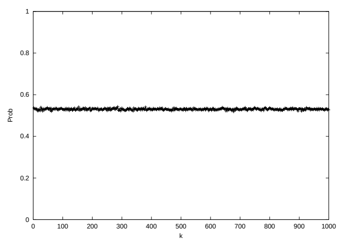
Trace 36, PER = 0.0052



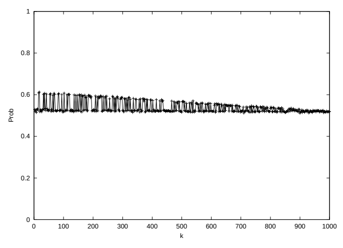
Trace 37, PER = 0.7906



Trace 38, PER = 0.5966

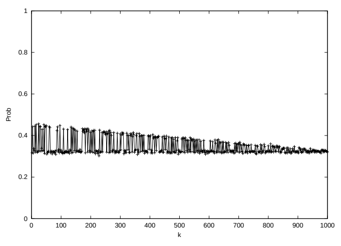


Trace 39, PER = 0.5296

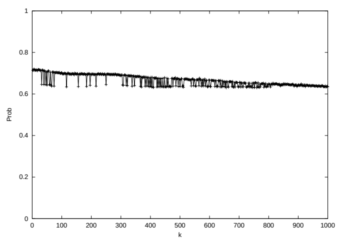


Trace 41, PER = 0.5537

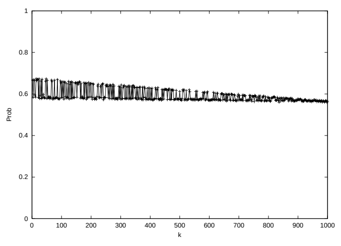
Table 3.25: Conditional Packet Error Probabilities for Traces 16 to 41



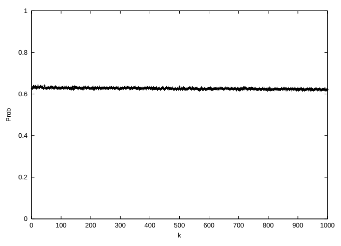
Trace 42, PER = 0.3640



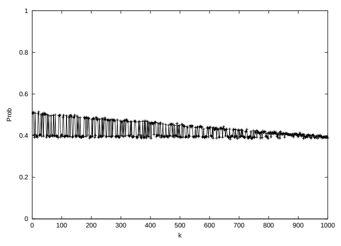
Trace 43, PER = 0.6796



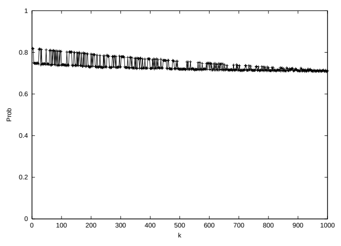
Trace 44, PER = 0.5517



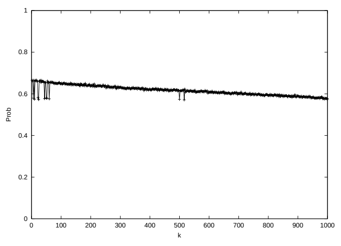
Trace 45, PER = 0.5975



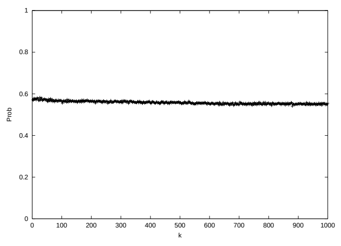
Trace 46, PER = 0.4286



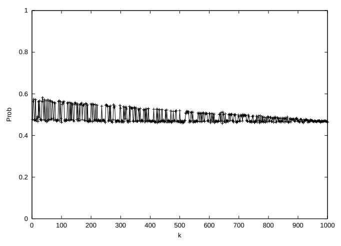
Trace 47, PER = 0.6574



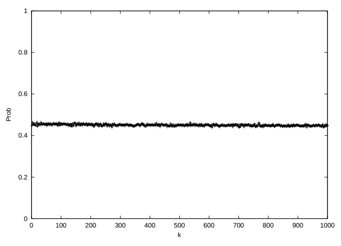
Trace 48, PER = 0.5851



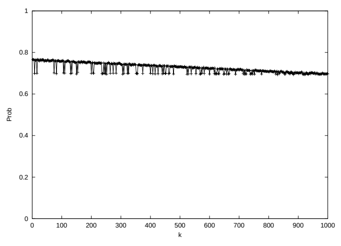
Trace 49, PER = 0.5424



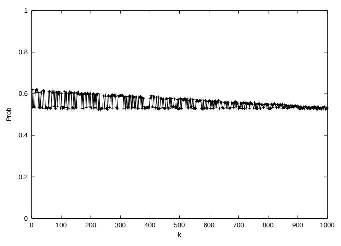
Trace 50, PER = 0.4965



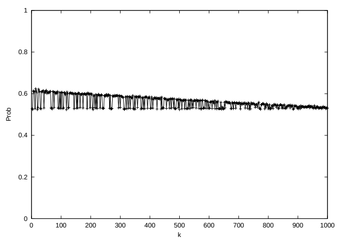
Trace 51, PER = 0.4454



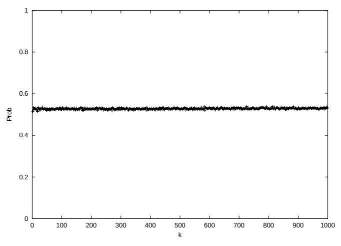
Trace 52, PER = 0.7456



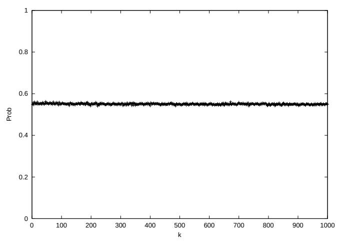
Trace 53, PER = 0.5748



Trace 54, PER = 0.5763

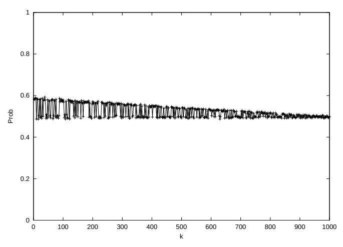


Trace 55, PER = 0.5236

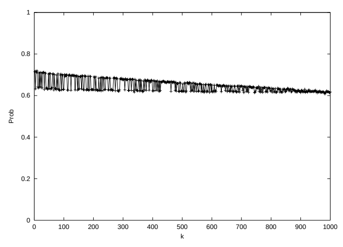


Trace 56, PER = 0.5454

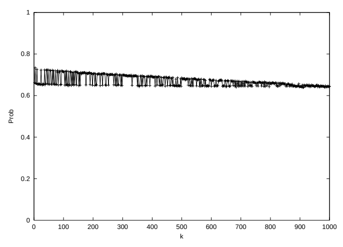
Table 3.26: Conditional Packet Error Probabilities for Traces 42 to 56



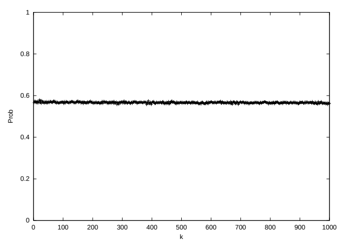
Trace 57, PER = 0.5411



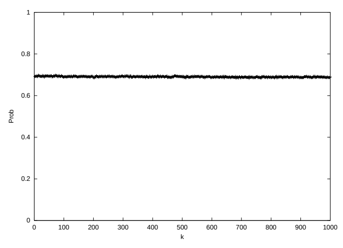
Trace 58, PER = 0.6078



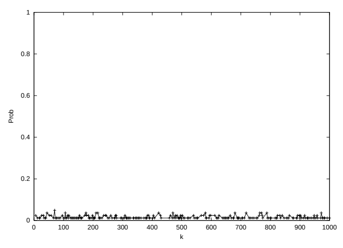
Trace 59, PER = 0.681



Trace 60, PER = 0.5605



Trace 61, PER = 0.6807



Trace 72, PER = 0.0041

Table 3.27: Conditional Packet Error Probabilities for Traces 57 to 72

3.3 Stochastic Modelling

After inspection of the traces we conclude that we have the following aspects to consider when constructing models:

- Loss of whole packets due to errors in the (fixed length and fixed modulation) PHY header[6].
- Occurence of bit shifts (see section 2.2.1).
- Bit error behaviour within a single trace, as given by the indicator sequence. However, to obtain sufficiently accurate results, one needs high mean bit error rates, for comparably low mean bit error rates ($< 10^{-5}$) we run into difficulties.
- Different mean bit error rates for different packet sizes
- Long term variation for a single packet size.

One key issue to resolve is the appropriate choice of timescales in the model. The loss of whole packets and the long term variation for a single packet size are expected to change only slowly, on the scale of one hour or more. However, in this campaign we have not measured the long term variation for a single packet size, so we cannot make any statements on it (especially for the smallest packets we have measured only ten minutes in succession).

At this point in time it is also questionable on whether it makes sense to model the behaviour of different packet sizes. As mentioned before, we think that the differences are mainly due to the fact that in all traces bit errors tend to occur at the beginning of frames, maybe due to the necessary switching of the modulation type between PHY header and data part. For longer packets many errors in the beginning can be compensated by fewer errors in the remaining packets, for short packets this cannot happen[7]. Currently we are not sure whether this behaviour is special to the modem pair actually used or if it is typical for this

[6]In section 3.2.1 we have also discussed the phenomenon of ghost packets and missized packets. The key observation was that all three anomalies seem to be correlated to switching off the milling machine. For the ghost packets and missized packet statistics we assume that these can be attributed to our measurement equipment (e.g. lack of proper magnetic shielding) and thus do not tell anything about the stream of received packets. For this reason they are not included in modeling. For lost packets we assume that they occur due to failure of the receiving radio modem to get bit synchronization or of a wrong checksum in the DBPSK header. Because this is related to the packet stream, it should be modeled.

[7]For selecting a framing method this may lead to the unusual idea to protect only the first few bytes of the data part with a strong FEC code and to use for the remaining part a weaker code (with better coderate) or no code at all.

class of modems. This question will be addressed in a subsequent measurement campaign. If it turns out that this is a typical behaviour, then an interesting problem for modeling occurs: assume that at time t_0 a packet of size s_1 is transmitted and the channel state is governed by model m_1. Assume furthermore that the current channel state w.r.t. m_1 lasts for a time τ much larger than needed for s_1. When finishing the first packet, the next packet is selected, having a size $s_2 \neq s_1$ with another model m_2 associated to it. So the question is: to which state of model m_2 should we switch? We can make the following points:

- Consider the case where all models are two-state markovian models (Gilbert/Elliot-Model). The canonical choice would be to go to the bad state, when we are in the bad state, and to go into the good state, when we were in the good state. If the bit error rates within the states were arbitrary, another possible approach would be to enter the state whose bit error rate is the closest to the current error rate. However, with our approach in the good state no error occurs and in the bad state the bit error rate is comparably high, which justifies our choice. However, please note that the channel acquires memory this way, because the sum of two geometric random variables (state holding times) is not geometric anymore, and thus not memoryless.
- Consider the case where all models are semi-markovian models as generated by the heuristics described in section 2.2.3. The same considerations as above apply.
- Consider the case of bipartite models where the number of good states and bad states can be arbitrarily. We propose to use the following policy:
  - When within a good state, let τ denote the remaining time within the good state in model m_1. From m_2 select the state s_2 from the set of good state such that τ is in the range of state durations generated by s_2.
  - When within a bad state, we can base our decision on at least two criterias: error rate and state duration. However, we assume that for running simulations we restrict ourselves to models of a single burst order k_0, and thus for all traces we have a common lower bound on the error rate. Furthermore from our traces we have observed that the bit error rates are comparable, and thus we resort only to the state duration for determining the next state. For this reason, again we choose a state from the set of bad states, for which τ can be generated within that state.

3.3.1 Modeling of Traces as iid

In this section we present the parameters for the simple stochastic models based on iid sequences of random variables, see section 2.2.3. Every trace is treated with different burst orders (see section 2.2.2).

The (in)validity of the iid assumption can be checked using the autocovariance function of the X_i and Y_i respectively. As mentioned in section 2.2.3, if the value of this function for some lag exceeds the value 0.2 the values can be regarded as correlated (and thus not as independent). Therefore in figure 3.18 we show for every trace the value

$$\max\{R_{X,k_0}(k) | k \in \mathbb{N}, k \geq 1\}$$

accordingly in figure 3.17 we show for every trace the value

$$\max\{R_{Y,k_0}(k) | k \in \mathbb{N}, k \geq 1\}$$

respectively for all burst orders k_0 investigated. While the figures give only an optical impression, in tables B.1, B.2 and B.3 we give numerically for every trace number the same values, however, along with the lag n at which this maximum occurs. We can make the important conclusion that for many traces the independence assumption has to be rejected and that iid models are not adequate.

An important issue to resolve is the choice of the proper burst order. The most important factor are the resulting computation costs when simulating, which in turn depend on two factors:

- the rate of state or model changes, which involve timer expiration, determining the new error rate and setting a new timer based on a random number, and
- the duration of the bad states, since every packet transmitted within a bad state involves random number generation for determining whether a packet is correct (for exceedingly high error rates we can mark all packets as erroneous).

It is clear from definition, that with increasing burst order k_0 the (mean) length of bad states increases, because their number decreases while their size increases. However, for good states this is not immediately clear, since their length decreases, and their number too. Certainly, for k_0 large enough we can represent a whole trace as a single bad period (or as a single good period, when no errors occur), however, this is not meaningful. On the other hand, with $k_0 = 1$ we certainly have the highest rate of state changes. We can think of different heuristics for setting k_0:

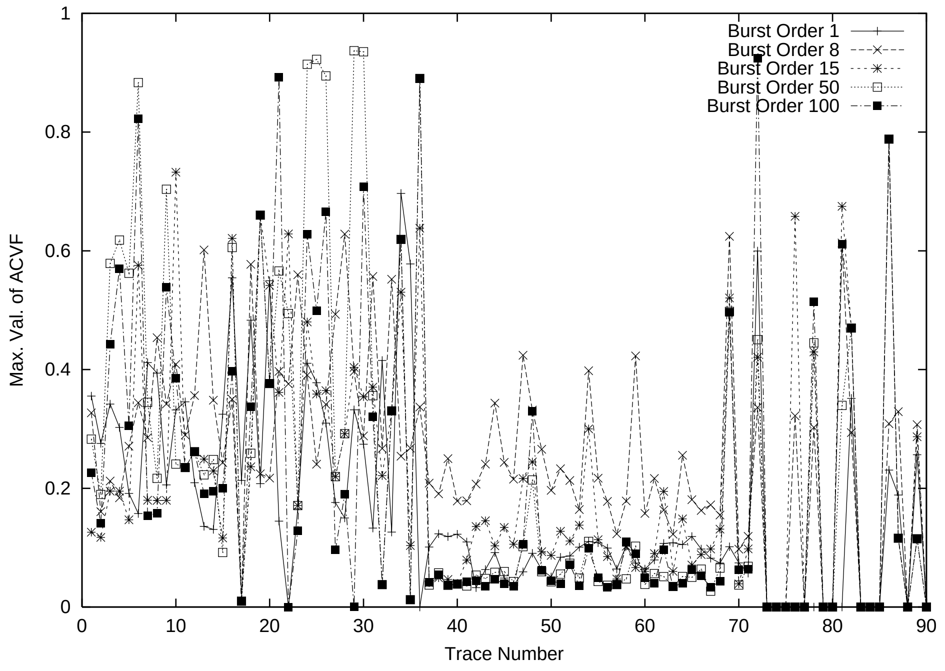


Figure 3.17: Max. value of autocovariance function for error burst lengths vs. Trace Number

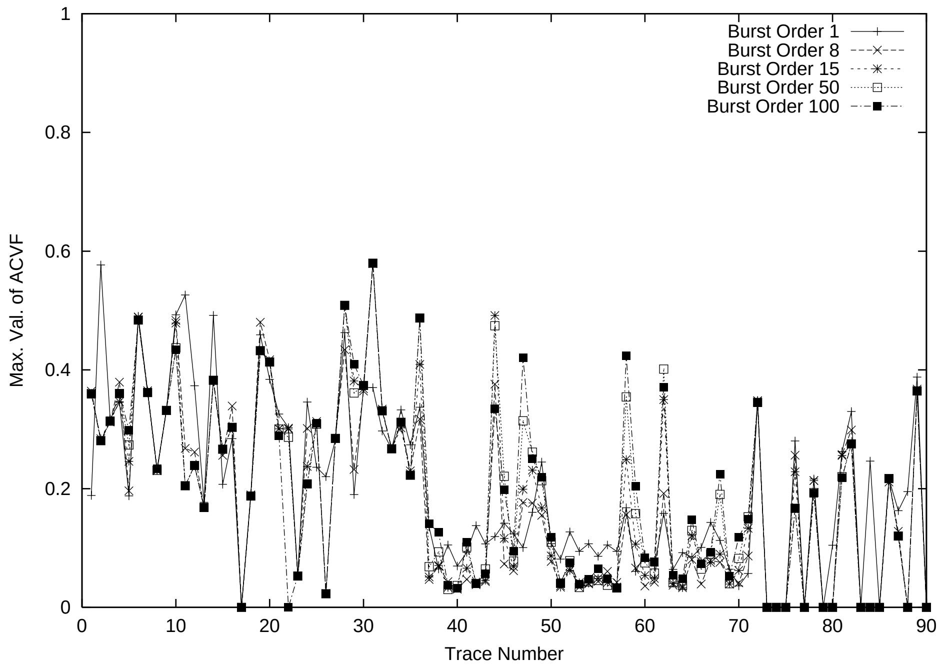


Figure 3.18: Max. value of autocovariance function for errorfree burst lengths vs. Trace Number

- If we want to simulate bit errors with high accuracy (e.g. when bit errors in packet headers have different consequences than those in the packet data) we choose $k_0 = 1$.
- Fix a minimum number of state changes within a trace (say, 100) and determine k_0 such that the sum of the coefficients of variation of good state durations and bad state durations is minimal, thus trying to reduce the overall variability in state durations.

In our evaluations we have chosen $k_0 \in \{1, 8, 15, 50, 100\}$. In figure 3.19 we show for all burst orders, how often the channel was in bad state, thus giving a direct measure of the rate of state changes. It can be seen that for $k_0 = 1$ we have by far the most state changes, while for all other burst orders the rates do not differ so much. If we leave out $k_0 = 1$ as done in figure 3.20 we can see that the reduction from $k_0 = 8$ to $k_0 = 15$ is approximately a factor of two, while the reductions for higher burst orders are much smaller. Thus we conclude that $k_0 = 15$ might be a good compromise between precision and simulation time.

Two-State Markov Model

For the markovian two-state model we give for every trace and for burst orders $k_0 = 1$, $k_0 = 8$, $k_0 = 15$, $k_0 = 50$ and $k_0 = 100$ the values p_X, p_Y and e_b as discussed in section 2.2.3. The corresponding values are shown in tables C.1, C.2 and C.3.

Two-State Semi-Markov Models

For determining the two-state semi-markovian models, we have applied the heuristics described in section 2.2.3. The resulting distributions including their parameters are shown for $k_0 = 1$ in table C.4, for $k_0 = 8$ in table C.5, for $k_0 = 15$ in table C.6, for $k_0 = 50$ in table C.7 and for $k_0 = 100$ in table C.8. The corresponding error densities can be taken from the tables of the two-state markovian model. It is worth noting that in all cases the errorfree burst lengths are modelled with the lognormal distribution, while for error burst lengths the picture is more diverse: For $k_0 = 1$ the geometric distribution frequently occurs, while for all other burst orders it does not occur at all, being dominated by the lognormal distribution. The binomial distribution occurs only occasionally.

3.3.2 Bipartite Modeling

We do not discuss the results of bipartite modelling here for two reasons:

- They are hard to communicate, since the number of parameters is large, and

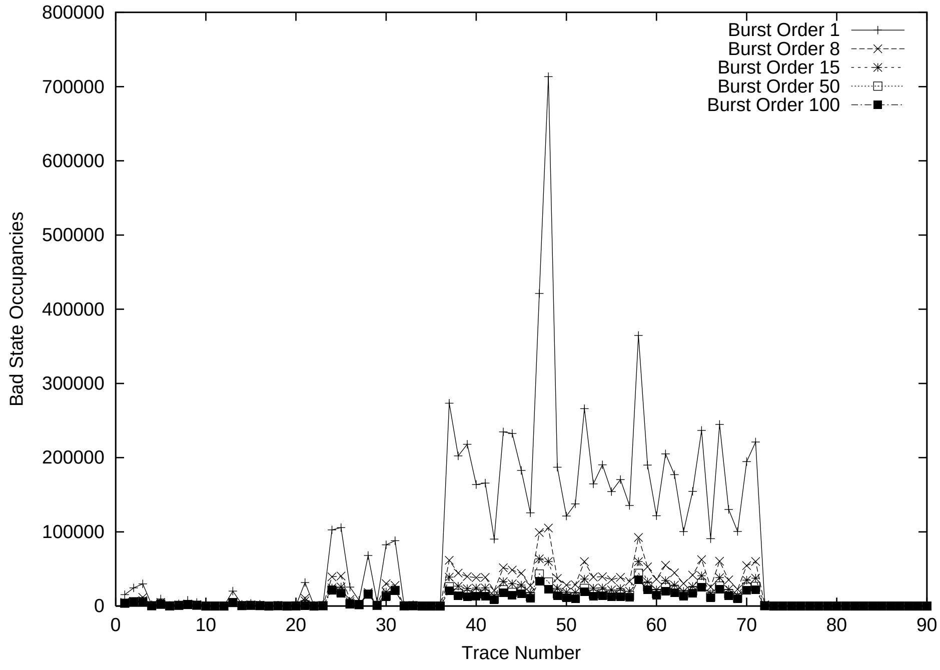


Figure 3.19: Number of times being in bad state vs. Trace Number

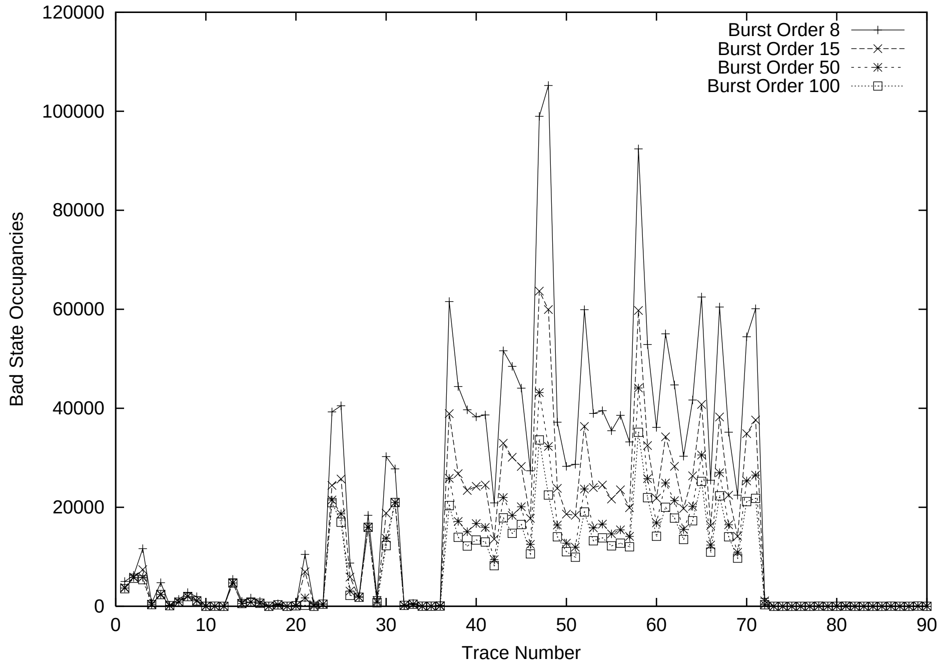


Figure 3.20: Number of times being in bad state vs. Trace Number without $k_0 = 1$

- there is no appealing physical meaning associated with the data, only a density plot of the matrix **P** would give insights.

The bit error rates in the bad state are for all burst orders well above 10%, often ranging to 60%-70%.

3.4 Conclusions and Lessons Learned

In the following, we shortly discuss our findings and impressions from this first measurement campaign. Regarding our measurement setup and choice of parameters, we see the following improvements:

- The 1b8b coding used in the Tx module (see section 2.1.1) has the drawback that specifically at the beginning of a trace significantly more zero bits than one bits are transmitted. However, with zero bits it is harder to obtain a high bit error rate when bit shifts occur. But a high bit error rate in case of bit shifting can serve as a nice criterion for filtering out these packets. For this reason we switch to the 1b8b coding:

$$\begin{aligned} 0 &\mapsto 11000011 \\ 1 &\mapsto 00111100 \end{aligned}$$

  This method allows also to investigate whether errors are independent from the value of the data bits.

- We should repeat the measurements with another pair of modems, because it is not clear in the moment whether the findings discussed in section 3.2.1 belong only to the actually used pair of modems or are typical for this class of modems. Some test measurements within our institute have produced too few errors in order to make any statement.

- If strong electromagnetic fields are present, our setup produces packets with incorrect lengths or delivers old packets. A more appropriate shielding should be investigated.

- Since we cannot observe any dependency on the *GapTime* parameter, we keep it fixed in further measurement campaigns.

- In one of the next campaigns we should investigate LOS scenarions as well.

- The periodicity of 128 bits found for QPSK modulation in the error correlation structure (see section 3.2.3) cannot be explained satisfactorily. However, one guess is that it has something to do with the operation of the scrambler. In a following measurement campaign we will perform measurements with and without scrambling and compare the results.
- The question of whether the presence of scrambling influences error rates should be addressed in a following campaign. Some preliminary measurements taken at our institute showed a significantly reduced error rate for operation without scrambling[8].
- For further measurement campaigns an upgrade to the more recent PRISM II chipset is interesting.

For characterizing the error behaviour of the channel, we can make the following statements:

- Errors tend to occur at the beginning of packets, where the error rate is much higher than in the remaining packet. This implies different mean bit error rates for different packet sizes. In the moment it is not clear whether this is due to the pair of modems actually used or whether this is a more general property. In the latter case, this must be considered when building stochastic models (e.g. differentiating against packet sizes). A possible approach would be to use two submodels: one for, say, the first 1000 bits, and a second for the remaining bits of a packet.
- We can clearly identify the need to incorporate different aspects into a channel model, namely the stochastic bit error behaviour of the received packets and the packet loss process.
- In our measurements for QPSK the mean bit error rates even for a single packet size can differ over two or three orders of magnitude.
- Regarding bit errors, there is a considerable degree of correlation, especially over short distances. Over longer distances the correlation shows up for the CCK modulation, while for QPSK the picture is more diverse. Regarding packet errors, for CCK the packet losses can be regarded as independent, while for QPSK they seem to be correlated / clustered.

[8]Since the scrambler operation incorporates usage of an eight-bit shift register with feedback, one erroneous bit may have influence on other bits currently present in the shift register.

- Within error bursts the error density is very high (ranging from 0.1 to 0.5), however, the number of consecutively wrong bits is in most cases one or two, longer bursts of consecutively wrong bits are rare.
- Since the channel is instationary on longer timescales (switching on and off machines), we need measurements of a single packet size over longer times, in order to capture the long term channel behaviour.
- Using iid models for the length of good states and bad states is often bad or not adequate, as can be seen from high autocorrelation values.
- For iid models the geometric distribution (as necessary for markovian models) is often not adequate, because of the large coefficients of variation, especially for errorfree burst lengths.
- In general the errorfree burst lengths show much more variability than the error burst lengths. Error bursts tend to be short, in most cases not more than two or three times the burst order. For errorfree bursts it is often the case that after an error burst the following errorfree burst is a short one.
- Its not worth to use models based on 11 MBit CCK modulation, since this modulation type shows so much errors, that it is ruled out for use in industrial environments.
- The packet loss process is of equal importance as the bit error process. It can be investigated with the same methods as described in section 2.2.2 by simply forming an indicator sequence (received packets are marked as '0' and lost packets as '1').

For building MAC- and data link protocols for industrial wireless LANs, we can draw the following conclusions:

- Since even for higher burst orders the error densities are comparably high (typically 0.2 and more) the effectiveness of FEC schemes is questionable. However, this needs to be investigated.
- Any MAC-scheme should incorporate both FEC (for which effectiveness can be evaluated using the indicator sequences) and suitable retransmission schemes (since there are losses of whole packets). Especially the latter calls for good repolling schemes in case of polling-based MAC protocols.

- Since we are investigating and modeling the behaviour in industrial environments, it makes sense to use channel models based on worst-case-traces (at least for QPSK, CCK seems to be ruled out).

Some interesting questions for further research and measurements are the following:

- Evaluate the effectiveness of different FEC schemes when confronted with our traces.
- Evaluate the effectiveness of different retransmission schemes when confronted with our traces. How can these schemes effectively combined with FEC?
- The occurence and the reasons for bit shifts should be investigated more deeply.
- Evaluate how good bipartite models can approximate the measured process.

Chapter 4

Acknowledgements

We would like to gratefully acknowledge the help of the people from the PTZ Berlin, namely of Dipl.-Ing. Ulrich Doll and Dr.-Ing. Hendrik Engel, who give us access to the PTZ facilities and to Dipl.-Ing. Sascha Piltz and Dipl.-Ing. Dirk Oberschmidt who helped us in setting up the measurement environment. Especially Sascha Piltz showed much patience during the second campaign.

Furthermore we want to thank Christian Hoene from the Telecommunication Networks Group, who has developed and set up the first versions of the wireless NIC.

Bibliography

[1] Richard L. Abrahams. *2.4GHz 11Mbps MACless DSSS Radio HWB1151 Users Guide - AN9835.1.* Intersil, 1999.

[2] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis – forecasting and control.* Holden-Day, San Francisco, 3 edition, 1994.

[3] B. D. Fritchman. A binary channel characterisation using partitioned markov chains. *IEEE Transactions on Information Theory*, 13(2):221–227, April 1967.

[4] J. Garcia-Frias and P. M. Crespo. Hidden Markov Models for burst error characterization in indoor radio channels. *IEEE Transactions on Vehicular Technology*, 46:1006–1020, November 1997.

[5] Intersil. *HFA3860B Data Sheet, File Number 4594.1*, 1999.

[6] Kevin Lai and Mary Baker. Measuring bandwidth. In *Proc. of IEEE Infocom'99*, 1999.

[7] R. H. McCullough. The binary regenerative channel. *Bell Systems Technical Journal*, 47:1713–1735, October 1968.

[8] P. Heinz Müller, editor. *Wahrscheinlichkeitsrechnung und Mathematische Statistik.* Akademie Verlag, Berlin, third edition, 1980.

[9] Motorola Inc. *MPC 860 PowerQUICC Users's Manual*, 1998.

[10] The Editors of IEEE 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Draft Standard.* IEEE, 1997.

[11] The Editors of IEEE 802.11. *Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer Specifications: High Speed Physical Layer in the 5 GHz Band*, 1999.

[12] Tundra Corporation. *Reference Manual Tundra PCI Interconnect*, 1998.

[13] W. Turin and R. van Nobelen. Hidden markov modeling of fading channels. In *Proc. IEEE Vehicular Technology Conference*, pages 1234–1238, May 1998.

[14] William Turin. *Digital Transmission Systems – Performance Analysis and Modeling.* McGraw-Hill Telecommunications. McGraw-Hill, New York, 1998.

[15] William Turin and Robert van Nobelen. Hidden Markov Modeling of Flat Fading Channels. *IEEE Journal on Selected Areas in Communications*, 16(9):1809–1817, December 1998.

[16] Andreas Willig, Martin Kubisch, and Adam Wolisz. Bit Error Rate Measurements – Second Campaign, factorial measurement. TKN Technical Report Series TKN-00-011, Telecommunication Networks Group, Technical University Berlin, November 2000. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[17] Andreas Willig, Martin Kubisch, and Adam Wolisz. Bit Error Rate Measurements – Second Campaign, longterm1 measurement. TKN Technical Report Series TKN-00-009, Telecommunication Networks Group, Technical University Berlin, November 2000. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

[18] Andreas Willig, Martin Kubisch, and Adam Wolisz. Bit Error Rate Measurements – Second Campaign, longterm2 measurement. TKN Technical Report Series TKN-00-010, Telecommunication Networks Group, Technical University Berlin, November 2000. http://www-tkn.ee.tu-berlin.de/publications/tknrreports.html.

Appendix A

Mapping of Trace Numbers to Parameters for PTZ Traces (Campaign 1)

The table gives the mapping from trace numbers to the variable parameters of the corresponding traces. Three values are given: the modulation type, the packet size in bytes (equals *NumChunks* times *ChunkSize*) and the interpacket gap (*GapTime*). Furthermore, for each trace we give the mean bit error rate and the fraction of erroneous packets w.r.t. the number of received packets.

Number	**Parameters**	**Mean BER**	**PER**
1	QPSK,108,600	0.001276	0.120228814417333
2	QPSK,108,1000	0.002051	0.157344432037357
3	QPSK,108,2000	0.002734	0.165141496958477
4	QPSK,108,5000	0.000134	0.0134877657440834
5	QPSK,108,10000	0.000739	0.0846138293570562
6	QPSK,324,600	0.000015	0.00230172629472104
7	QPSK,324,1000	0.000123	0.0316279069767442
8	QPSK,324,2000	0.000231	0.0425974025974026
9	QPSK,324,5000	0.000119	0.0217871485943775
10	QPSK,324,10000	0.000154	0.000935453695042095
11	QPSK,504,600	0.000002	0.0008
12	QPSK,504,1000	0.000001	0.00065006500650065
13	QPSK,504,2000	0.000315	0.0627441161742614
14	QPSK,504,5000	0.000036	0.00855941535689258
15	QPSK,504,10000	0.000070	0.0276339400153807
16	QPSK,1008,600	0.000027	0.0114820435002529
17	QPSK,1008,1000	0.000000	0.00025
18	QPSK,1008,2000	0.000008	0.00364188163884674

Number	Parameters	Mean BER	PER
19	QPSK,1008,5000	0.000001	0.0003
20	QPSK,1008,10000	0.000006	0.00310124049619848
21	QPSK,2016,600	0.000098	0.00145007250362518
22	QPSK,2016,1000	0.000003	5.00375281461096e-05
23	QPSK,2016,2000	0.000007	0.00360992730007521
24	QPSK,2016,5000	0.000442	0.0937373737373737
25	QPSK,2016,10000	0.000404	0.103620347268191
26	QPSK,4032,600	0.000044	0.0265013250662533
27	QPSK,4032,1000	0.000015	0.0189
28	QPSK,4032,2000	0.000136	0.155046513954186
29	QPSK,4032,5000	0.000011	0.015650782539127
30	QPSK,4032,10000	0.000151	0.136647826739359
31	QPSK,6012,600	0.000118	0.0992049602480124
32	QPSK,6012,1000	0.000001	0.00710035501775089
33	QPSK,6012,2000	0.000002	0.00285
34	QPSK,6012,5000	0.000000	0.0007
35	QPSK,6012,10000	0.000000	0.00085
36	QPSK,108,600	0.000281	0.00520340586565752
37	CCK11Mbps,108,600	0.025511	0.7906
38	CCK11Mbps,108,1000	0.018970	0.5966
39	CCK11Mbps,108,2000	0.020942	0.52965
40	CCK11Mbps,108,5000	0.015398	0.55115
41	CCK11Mbps,108,10000	0.015353	0.5537
42	CCK11Mbps,324,600	0.002740	0.36405
43	CCK11Mbps,324,1000	0.007271	0.67965
44	CCK11Mbps,324,2000	0.007188	0.55175
45	CCK11Mbps,324,5000	0.005575	0.59755
46	CCK11Mbps,324,10000	0.003861	0.42865
47	CCK11Mbps,504,600	0.008375	0.65745
48	CCK11Mbps,504,1000	0.015173	0.585179258962948
49	CCK11Mbps,504,2000	0.003788	0.5424
50	CCK11Mbps,504,5000	0.002321	0.49655
51	CCK11Mbps,504,10000	0.002682	0.44545
52	CCK11Mbps,1008,600	0.002499	0.74565
53	CCK11Mbps,1008,1000	0.001536	0.57485
54	CCK11Mbps,1008,2000	0.001844	0.57635
55	CCK11Mbps,1008,5000	0.001441	0.5236
56	CCK11Mbps,1008,10000	0.001608	0.5454
57	CCK11Mbps,2016,600	0.000654	0.5411
58	CCK11Mbps,2016,1000	0.001836	0.60785
59	CCK11Mbps,2016,2000	0.000898	0.681
60	CCK11Mbps,2016,5000	0.000560	0.56055
61	CCK11Mbps,2016,10000	0.000974	0.6807
62	CCK11Mbps,4032,600	0.000424	0.52155
63	CCK11Mbps,4032,1000	0.000226	0.5305
64	CCK11Mbps,4032,2000	0.000357	0.623
65	CCK11Mbps,4032,5000	0.000577	0.65445
66	CCK11Mbps,4032,10000	0.000209	0.4662

Number	**Parameters**	**Mean BER**	**PER**
67	CCK11Mbps,6012,600	0.000407	0.74995
68	CCK11Mbps,6012,1000	0.000206	0.512328082020505
69	CCK11Mbps,6012,2000	0.000166	0.4266
70	CCK11Mbps,6012,5000	0.000308	0.68547418967587
71	CCK11Mbps,6012,10000	0.000349	0.726422642264226
72	QPSK,108,600	0.000262	0.0041533226581265
73	QPSK,108,1000	0.000000	0
74	QPSK,108,2000	0.000000	0
75	QPSK,108,5000	0.000000	0
76	QPSK,108,10000	0.000003	5e-05
77	QPSK,324,600	0.000000	0
78	QPSK,324,1000	0.000005	0.0001
79	QPSK,324,2000	0.000000	0
80	QPSK,324,5000	0.000000	5.00125031257814e-05
81	QPSK,324,10000	0.000005	0.0001
82	QPSK,504,600	0.000001	0.000200010000500025
83	QPSK,504,1000	0.000000	0
84	QPSK,504,2000	0.000000	0.0001
85	QPSK,504,5000	0.000000	0
86	QPSK,504,10000	0.000006	0.00075
87	QPSK,1008,600	0.000000	0.0002
88	QPSK,1008,1000	0.000000	5e-05
89	QPSK,1008,2000	0.000002	0.00150958587027625

Table A.1: Mapping of Trace numbers to measurement parameters

Appendix B

Maximum Values of Autocovariance Functions for Error Bursts and Errorfree Bursts for PTZ Trace (Campaign 1)

In the following tables we give for the burst orders $k_0 \in \{1, 8, 15, 50, 100\}$ the maximum values of the autocovariance functions $R_{X,k_0}(n)$ and $R_{Y,k_0}(n)$ along with the lag at which the maximum occurs.

Number	$(\mathbf{n}, \mathbf{R_{X,1}(n)})$	$(\mathbf{n}, \mathbf{R_{Y,1}(n)})$	$(\mathbf{n}, \mathbf{R_{X,8}(n)})$	$(\mathbf{n}, \mathbf{R_{Y,8}(n)})$
1	(1517, 0.188476)	(1, 0.355329)	(2, 0.364152)	(1, 0.326969)
2	(3, 0.576872)	(1, 0.275673)	(1, 0.281042)	(1, 0.160558)
3	(44, 0.317083)	(1, 0.341952)	(24, 0.312294)	(6, 0.212329)
4	(978, 0.345473)	(1, 0.302541)	(389, 0.379306)	(1, 0.184038)
5	(121, 0.187709)	(3, 0.191350)	(58, 0.196288)	(1, 0.270865)
6	(497, 0.490600)	(40, 0.157503)	(166, 0.489302)	(1, 0.344042)
7	(256, 0.364594)	(1, 0.411813)	(135, 0.365051)	(1, 0.285437)
8	(482, 0.230897)	(1, 0.394028)	(123, 0.229900)	(1, 0.453673)
9	(4733, 0.332474)	(4, 0.205795)	(1938, 0.332128)	(1, 0.342109)
10	(159, 0.492886)	(5, 0.331876)	(60, 0.483437)	(6, 0.408792)
11	(5, 0.526302)	(2, 0.345663)	(10, 0.267806)	(1, 0.289674)
12	(8, 0.373427)	(26, 0.209290)	(2, 0.261637)	(9, 0.356355)
13	(17, 0.177035)	(5, 0.135784)	(5425, 0.168409)	(1, 0.601614)
14	(6, 0.491948)	(3, 0.130838)	(1, 0.382522)	(1, 0.348281)
15	(10, 0.207235)	(1, 0.324747)	(4, 0.256168)	(1, 0.243275)
16	(10, 0.284205)	(5, 0.554525)	(2, 0.339065)	(1, 0.350002)
17	(0, 0)	(8, 0.213333)	(0, 0)	(1, 0.009804)

Number	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},1}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},1}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},8}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},8}(\mathbf{n}))$
18	(1078, 0.191425)	(1, 0.483173)	(493, 0.189582)	(2, 0.577241)
19	(9, 0.459079)	(19, 0.207814)	(3, 0.480267)	(7, 0.224320)
20	(30, 0.384136)	(5, 0.549628)	(12, 0.417007)	(1, 0.217184)
21	(5629, 0.325594)	(16, 0.144555)	(1964, 0.301001)	(7, 0.396884)
22	(1049, 0.302983)	(0, 0)	(492, 0.301786)	(9, 0.375896)
23	(1411, 0.054416)	(5, 0.165522)	(347, 0.053035)	(1, 0.559670)
24	(7027, 0.345937)	(5, 0.410550)	(2540, 0.301545)	(5, 0.390586)
25	(7332, 0.236089)	(1, 0.377808)	(2625, 0.313188)	(7, 0.240581)
26	(6, 0.220078)	(1, 0.309689)	(8737, 0.023199)	(7, 0.341481)
27	(24, 0.281950)	(1, 0.176093)	(8, 0.284717)	(1, 0.492954)
28	(5, 0.462636)	(1, 0.150083)	(1, 0.433976)	(1, 0.628358)
29	(39, 0.189951)	(4, 0.332225)	(8, 0.232166)	(1, 0.396785)
30	(4, 0.370249)	(4, 0.274331)	(1, 0.364274)	(8, 0.288225)
31	(6, 0.370150)	(1, 0.133040)	(1, 0.579042)	(1, 0.556828)
32	(147, 0.297118)	(1, 0.415273)	(4, 0.334051)	(1, 0.267063)
33	(42, 0.267090)	(5, 0.126292)	(10, 0.267595)	(1, 0.552098)
34	(35, 0.332709)	(2, 0.696860)	(14, 0.308828)	(8, 0.254088)
35	(80, 0.273870)	(1, 0.577985)	(20, 0.229963)	(16, 0.267639)
36	(39, 0.337197)	(0, 0)	(16, 0.317137)	(6, 0.336848)
37	(3, 0.135526)	(1, 0.101039)	(651, 0.046973)	(1, 0.208606)
38	(2001, 0.064682)	(1, 0.123320)	(25, 0.069481)	(1, 0.190638)
39	(3, 0.105129)	(1, 0.118660)	(245, 0.043449)	(1, 0.250007)
40	(3, 0.069571)	(1, 0.122592)	(388, 0.030569)	(1, 0.178471)
41	(3, 0.092534)	(1, 0.109642)	(365, 0.047201)	(1, 0.179030)
42	(3, 0.137779)	(1503, 0.032671)	(6, 0.038219)	(1, 0.207065)
43	(3, 0.106882)	(1, 0.063147)	(921, 0.043345)	(1, 0.240854)
44	(3, 0.119263)	(1, 0.091690)	(5, 0.376095)	(1, 0.343384)
45	(3, 0.141741)	(1, 0.048708)	(3, 0.072958)	(1, 0.243602)
46	(3, 0.123837)	(1, 0.036587)	(6, 0.061748)	(1, 0.215372)
47	(5, 0.100662)	(3, 0.059253)	(230, 0.176803)	(1, 0.424185)
48	(3, 0.160180)	(2, 0.090409)	(2, 0.175755)	(1, 0.331744)
49	(452, 0.244868)	(6, 0.062870)	(2, 0.154972)	(1, 0.265747)
50	(3, 0.107476)	(1, 0.050872)	(11, 0.077190)	(1, 0.196609)
51	(3, 0.081317)	(1, 0.083840)	(925, 0.036390)	(1, 0.232894)
52	(3, 0.127174)	(1, 0.086098)	(253, 0.065594)	(1, 0.213141)
53	(3, 0.094240)	(1, 0.100910)	(208, 0.033967)	(1, 0.164183)
54	(3, 0.107240)	(1, 0.110289)	(1570, 0.039514)	(1, 0.397975)
55	(3, 0.085923)	(3, 0.108267)	(3, 0.043761)	(1, 0.217066)
56	(3, 0.105151)	(1, 0.099185)	(2, 0.060730)	(1, 0.177934)
57	(3, 0.094317)	(1, 0.064022)	(656, 0.042023)	(1, 0.123373)
58	(3, 0.167872)	(1, 0.103878)	(9, 0.155962)	(1, 0.178637)
59	(3, 0.060741)	(1, 0.074212)	(7, 0.066357)	(5, 0.422960)
60	(3, 0.089261)	(1, 0.063914)	(8, 0.036013)	(1, 0.157638)
61	(3, 0.069440)	(1, 0.079326)	(83, 0.042564)	(1, 0.216535)
62	(3, 0.158420)	(3, 0.107219)	(2, 0.191828)	(1, 0.163408)
63	(3, 0.064052)	(1, 0.108273)	(2385, 0.037334)	(1, 0.122621)
64	(3, 0.092166)	(1, 0.105159)	(641, 0.035217)	(1, 0.255688)
65	(3, 0.084569)	(1, 0.118741)	(10, 0.080645)	(1, 0.180769)

Number	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},1}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},1}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},8}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},8}(\mathbf{n}))$
66	(3, 0.101010)	(3, 0.097772)	(258, 0.039197)	(1, 0.162669)
67	(3, 0.143098)	(1, 0.082202)	(23, 0.079227)	(1, 0.172013)
68	(3, 0.112713)	(1, 0.074086)	(237, 0.075912)	(1, 0.154969)
69	(3, 0.064237)	(2, 0.101475)	(27, 0.048738)	(1, 0.624431)
70	(562, 0.036058)	(2, 0.074291)	(117, 0.041421)	(1, 0.097509)
71	(3, 0.057132)	(1, 0.057118)	(10, 0.086563)	(1, 0.119101)
72	(4192, 0.348697)	(1, 0.599705)	(1515, 0.348121)	(6, 0.336322)
73	(0, 0)	(0, 0)	(0, 0)	(0, 0)
74	(0, 0)	(0, 0)	(0, 0)	(0, 0)
75	(0, 0)	(0, 0)	(0, 0)	(0, 0)
76	(53, 0.280340)	(0, 0)	(22, 0.256178)	(15, 0.322061)
77	(0, 0)	(0, 0)	(0, 0)	(0, 0)
78	(164, 0.216404)	(0, 0)	(58, 0.213625)	(13, 0.302611)
79	(0, 0)	(0, 0)	(0, 0)	(0, 0)
80	(6, 0.104731)	(0, 0)	(0, 0)	(0, 0)
81	(284, 0.262434)	(0, 0)	(122, 0.257366)	(7, 0.613316)
82	(15, 0.330028)	(14, 0.351459)	(7, 0.298800)	(4, 0.293805)
83	(0, 0)	(0, 0)	(0, 0)	(0, 0)
84	(5, 0.246408)	(0, 0)	(0, 0)	(0, 0)
85	(0, 0)	(0, 0)	(0, 0)	(0, 0)
86	(7, 0.216166)	(25, 0.230739)	(3, 0.211205)	(15, 0.308860)
87	(12, 0.162901)	(1, 0.188713)	(3, 0.128060)	(6, 0.329016)
88	(6, 0.195011)	(0, 0)	(0, 0)	(0, 0)
89	(5, 0.388001)	(1, 0.256566)	(1, 0.367798)	(1, 0.307202)
90	(0, 0) (0, 0)	(0, 0)	(0, 0)	

Table B.1: Trace numbers and maximum values of autocovariance function for error bursts and errorfree bursts

Number	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},15}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},15}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},50}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},50}(\mathbf{n}))$
1	(2, 0.360660)	(2, 0.126107)	(2, 0.359884)	(1, 0.282755)
2	(246, 0.284178)	(2, 0.117206)	(1, 0.280821)	(2, 0.189569)
3	(20, 0.313839)	(4, 0.195267)	(16, 0.313501)	(1, 0.578934)
4	(238, 0.346268)	(4, 0.195214)	(175, 0.360196)	(1, 0.618106)
5	(83, 0.245998)	(1, 0.146826)	(4, 0.273628)	(1, 0.562310)
6	(115, 0.488576)	(5, 0.575516)	(115, 0.484106)	(1, 0.883419)
7	(82, 0.362664)	(2, 0.179826)	(71, 0.361757)	(2, 0.344701)
8	(113, 0.231509)	(2, 0.179368)	(111, 0.231007)	(2, 0.216799)
9	(1135, 0.331720)	(5, 0.179625)	(1101, 0.331691)	(1, 0.703720)
10	(47, 0.478820)	(5, 0.732388)	(12, 0.437217)	(1, 0.240804)
11	(6, 0.204829)	(2, 0.235053)	(6, 0.204829)	(2, 0.235053)
12	(11, 0.238892)	(6, 0.261661)	(11, 0.238892)	(6, 0.261661)
13	(4715, 0.168315)	(2, 0.249066)	(4713, 0.168315)	(2, 0.222660)
14	(1, 0.382507)	(6, 0.229072)	(1, 0.382513)	(1, 0.248355)
15	(3, 0.266586)	(2, 0.116340)	(3, 0.266376)	(19, 0.091836)
16	(2, 0.303263)	(4, 0.620974)	(2, 0.303244)	(4, 0.605529)

Number	$(\mathbf{n}, \mathbf{R_{X,15}(n)})$	$(\mathbf{n}, \mathbf{R_{Y,15}(n)})$	$(\mathbf{n}, \mathbf{R_{X,50}(n)})$	$(\mathbf{n}, \mathbf{R_{Y,50}(n)})$
17	(0, 0)	(1, 0.009804)	(0, 0)	(1, 0.009804)
18	(304, 0.187585)	(101, 0.236399)	(302, 0.187552)	(83, 0.259045)
19	(2, 0.432336)	(2, 0.660091)	(2, 0.432336)	(2, 0.660091)
20	(8, 0.414077)	(4, 0.541674)	(8, 0.413373)	(4, 0.376389)
21	(1419, 0.300734)	(4, 0.361483)	(306, 0.300293)	(2, 0.566064)
22	(311, 0.300485)	(5, 0.628507)	(61, 0.286648)	(17, 0.494407)
23	(326, 0.052890)	(1, 0.170905)	(326, 0.052890)	(1, 0.170905)
24	(2, 0.237347)	(4, 0.480152)	(2, 0.207994)	(1, 0.914086)
25	(2061, 0.309221)	(5, 0.358609)	(408, 0.308673)	(1, 0.922403)
26	(5914, 0.023124)	(4, 0.364537)	(3071, 0.022909)	(2, 0.894617)
27	(8, 0.284635)	(1, 0.219580)	(8, 0.284635)	(1, 0.219580)
28	(1, 0.508682)	(1, 0.292089)	(1, 0.508682)	(1, 0.292069)
29	(2, 0.381454)	(3, 0.403721)	(2, 0.361178)	(1, 0.937086)
30	(7278, 0.368198)	(4, 0.354289)	(2219, 0.373805)	(1, 0.935331)
31	(1, 0.579712)	(1, 0.370433)	(1, 0.579711)	(1, 0.355986)
32	(4, 0.331172)	(75, 0.221328)	(4, 0.331040)	(20, 0.037780)
33	(10, 0.267037)	(2, 0.330475)	(10, 0.267037)	(2, 0.330475)
34	(10, 0.302322)	(3, 0.530271)	(7, 0.311745)	(2, 0.619289)
35	(18, 0.225126)	(17, 0.103593)	(17, 0.222502)	(3, 0.012260)
36	(12, 0.409254)	(4, 0.638222)	(4, 0.487427)	(4, 0.890383)
37	(2, 0.051233)	(965, 0.040907)	(2, 0.068571)	(211, 0.037146)
38	(70, 0.070525)	(154, 0.049147)	(1, 0.093362)	(27, 0.057536)
39	(173, 0.033506)	(1, 0.046573)	(807, 0.030095)	(172, 0.036526)
40	(1624, 0.032374)	(1, 0.040021)	(3094, 0.036923)	(522, 0.038714)
41	(19, 0.066056)	(1, 0.079171)	(81, 0.101465)	(810, 0.035471)
42	(2, 0.041055)	(1, 0.135709)	(2, 0.040221)	(2673, 0.053761)
43	(3, 0.045517)	(1, 0.145282)	(2, 0.064694)	(3362, 0.048175)
44	(4, 0.491737)	(6494, 0.103158)	(5, 0.474295)	(1568, 0.058320)
45	(3, 0.116019)	(1, 0.134048)	(1, 0.220844)	(2342, 0.059853)
46	(3, 0.068636)	(1, 0.105743)	(1, 0.078002)	(410, 0.042873)
47	(7, 0.199011)	(539, 0.216634)	(6, 0.314213)	(1052, 0.101677)
48	(1, 0.231190)	(1, 0.244314)	(1, 0.261550)	(306, 0.214159)
49	(2, 0.167635)	(412, 0.093247)	(2, 0.213444)	(48, 0.059604)
50	(30, 0.086780)	(8846, 0.086929)	(3, 0.109757)	(3168, 0.042476)
51	(2232, 0.033820)	(1, 0.127492)	(1158, 0.040496)	(1241, 0.055437)
52	(167, 0.062143)	(703, 0.111285)	(36, 0.078608)	(455, 0.073565)
53	(599, 0.041468)	(1, 0.138022)	(69, 0.033408)	(335, 0.048736)
54	(146, 0.040721)	(4885, 0.300179)	(18, 0.042968)	(2870, 0.110460)
55	(2, 0.048651)	(1, 0.113355)	(1, 0.045348)	(5728, 0.043102)
56	(2, 0.041774)	(1, 0.084882)	(2, 0.037378)	(1187, 0.036558)
57	(113, 0.034313)	(1676, 0.047128)	(21, 0.032886)	(999, 0.037806)
58	(6, 0.249095)	(1, 0.102457)	(2, 0.354413)	(3636, 0.047472)
59	(49, 0.106537)	(2383, 0.066519)	(3, 0.157921)	(1984, 0.102499)
60	(81, 0.053764)	(421, 0.060101)	(5, 0.074982)	(7, 0.038267)
61	(248, 0.048672)	(1, 0.089975)	(109, 0.057814)	(4973, 0.056363)
62	(4, 0.350057)	(1, 0.194656)	(18, 0.401281)	(183, 0.051243)
63	(359, 0.040130)	(1, 0.059900)	(2, 0.041953)	(323, 0.034493)
64	(80, 0.033084)	(1, 0.148446)	(129, 0.037368)	(1915, 0.051508)

Number	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},\mathbf{15}}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},\mathbf{15}}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},\mathbf{50}}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},\mathbf{50}}(\mathbf{n}))$
65	(11, 0.120676)	(1, 0.070757)	(14, 0.129607)	(458, 0.049979)
66	(3, 0.077016)	(3409, 0.088215)	(3, 0.064366)	(662, 0.063680)
67	(14, 0.075250)	(1, 0.097794)	(11, 0.089830)	(6714, 0.026676)
68	(11, 0.089645)	(372, 0.131078)	(1, 0.190215)	(3, 0.065573)
69	(20, 0.039242)	(1, 0.520622)	(12, 0.040011)	(5171, 0.497904)
70	(6, 0.062234)	(1, 0.039104)	(65, 0.082347)	(128, 0.037078)
71	(4, 0.133308)	(1, 0.097915)	(3, 0.152495)	(1607, 0.068491)
72	(1052, 0.347710)	(4, 0.420603)	(348, 0.345159)	(4, 0.450469)
73	(0, 0)	(0, 0)	(0, 0)	(0, 0)
74	(0, 0)	(0, 0)	(0, 0)	(0, 0)
75	(0, 0)	(0, 0)	(0, 0)	(0, 0)
76	(12, 0.228339)	(4, 0.658217)	(4, 0.166918)	(0, 0)
77	(0, 0)	(0, 0)	(0, 0)	(0, 0)
78	(45, 0.213543)	(5, 0.429804)	(10, 0.193305)	(1, 0.444954)
79	(0, 0)	(0, 0)	(0, 0)	(0, 0)
80	(0, 0)	(0, 0)	(0, 0)	(0, 0)
81	(103, 0.255759)	(5, 0.675010)	(21, 0.220150)	(10, 0.339535)
82	(4, 0.275489)	(4, 0.469790)	(4, 0.275489)	(4, 0.469790)
83	(0, 0)	(0, 0)	(0, 0)	(0, 0)
84	(0, 0)	(0, 0)	(0, 0)	(0, 0)
85	(0, 0)	(0, 0)	(0, 0)	(0, 0)
86	(2, 0.216929)	(2, 0.788147)	(2, 0.216929)	(2, 0.788147)
87	(3, 0.120178)	(3, 0.115961)	(3, 0.120178)	(3, 0.115961)
88	(0, 0)	(0, 0)	(0, 0)	(0, 0)
89	(1, 0.365193)	(12, 0.286758)	(1, 0.364639)	(12, 0.115116)
90	(0, 0)	(0, 0)	(0, 0)	(0, 0)

Table B.2: Trace numbers and maximum values of autocovariance function for error bursts and errorfree bursts

Number	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},\mathbf{100}}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},\mathbf{100}}(\mathbf{n}))$
1	(2, 0.359076)	(1, 0.226269)
2	(1, 0.280688)	(1, 0.141291)
3	(16, 0.314800)	(1, 0.442401)
4	(174, 0.360077)	(1, 0.569652)
5	(69, 0.298183)	(1, 0.305256)
6	(114, 0.484059)	(1, 0.822723)
7	(69, 0.361627)	(2, 0.153837)
8	(110, 0.233475)	(2, 0.158028)
9	(1093, 0.331683)	(1, 0.538659)
10	(11, 0.433638)	(1, 0.385269)
11	(6, 0.204829)	(2, 0.235053)
12	(11, 0.238892)	(6, 0.261661)
13	(4712, 0.168315)	(2, 0.191055)
14	(1, 0.382476)	(98, 0.195360)
15	(3, 0.266186)	(4, 0.200075)

Number	$(\mathbf{n}, \mathbf{R_{X,100}}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R_{Y,100}}(\mathbf{n}))$
16	(2, 0.303217)	(10, 0.397451)
17	(0, 0)	(1, 0.009804)
18	(300, 0.187519)	(70, 0.337115)
19	(2, 0.432336)	(2, 0.660091)
20	(8, 0.413373)	(4, 0.376389)
21	(173, 0.289320)	(1, 0.892735)
22	(0, 0)	(0, 0)
23	(326, 0.052880)	(298, 0.128326)
24	(2, 0.208404)	(2, 0.627812)
25	(37, 0.310368)	(1, 0.499281)
26	(2221, 0.022739)	(1, 0.665786)
27	(8, 0.284634)	(1, 0.096184)
28	(1, 0.508681)	(1, 0.189588)
29	(2, 0.409782)	(638, 0.000159)
30	(747, 0.373590)	(1, 0.707789)
31	(1, 0.579711)	(1, 0.320241)
32	(4, 0.331040)	(20, 0.037780)
33	(10, 0.267037)	(2, 0.330475)
34	(7, 0.311745)	(2, 0.619289)
35	(17, 0.222502)	(3, 0.012260)
36	(4, 0.487427)	(4, 0.890383)
37	(1, 0.140811)	(10, 0.041899)
38	(1, 0.126203)	(3, 0.054314)
39	(488, 0.037057)	(381, 0.036600)
40	(248, 0.032282)	(790, 0.038502)
41	(1, 0.109831)	(158, 0.042446)
42	(15, 0.040374)	(3079, 0.044278)
43	(90, 0.056684)	(226, 0.034699)
44	(10, 0.334494)	(5357, 0.046759)
45	(1, 0.197862)	(1589, 0.038817)
46	(19, 0.094127)	(141, 0.034742)
47	(3, 0.420609)	(242, 0.105914)
48	(1, 0.250063)	(243, 0.329697)
49	(1, 0.219332)	(28, 0.062149)
50	(3, 0.118246)	(1733, 0.044546)
51	(41, 0.041381)	(689, 0.039422)
52	(49, 0.075199)	(368, 0.071487)
53	(2901, 0.038534)	(1190, 0.035856)
54	(8, 0.047644)	(1738, 0.098559)
55	(1, 0.064807)	(466, 0.049209)
56	(19, 0.048028)	(3424, 0.033707)
57	(1786, 0.032418)	(1279, 0.038054)
58	(2, 0.423960)	(2, 0.109638)
59	(25, 0.204193)	(4333, 0.089834)
60	(5, 0.083771)	(1, 0.048994)
61	(10, 0.077074)	(197, 0.040481)
62	(1, 0.370857)	(2409, 0.096799)
63	(30, 0.054442)	(1742, 0.034141)

Number	$(\mathbf{n}, \mathbf{R}_{\mathbf{X},\mathbf{100}}(\mathbf{n}))$	$(\mathbf{n}, \mathbf{R}_{\mathbf{Y},\mathbf{100}}(\mathbf{n}))$
64	(103, 0.047951)	(52, 0.040515)
65	(4, 0.147516)	(106, 0.062990)
66	(4, 0.073511)	(218, 0.052788)
67	(16, 0.093153)	(524, 0.033181)
68	(1, 0.224645)	(6, 0.043535)
69	(6, 0.052665)	(4614, 0.496145)
70	(9, 0.118291)	(5, 0.062833)
71	(4, 0.148599)	(13, 0.063345)
72	(311, 0.344852)	(4, 0.924206)
73	(0, 0)	(0, 0)
74	(0, 0)	(0, 0)
75	(0, 0)	(0, 0)
76	(4, 0.166918)	(0, 0)
77	(0, 0)	(0, 0)
78	(10, 0.192062)	(10, 0.514564)
79	(0, 0)	(0, 0)
80	(0, 0)	(0, 0)
81	(20, 0.218189)	(10, 0.610857)
82	(4, 0.275489)	(4, 0.469790)
83	(0, 0)	(0, 0)
84	(0, 0)	(0, 0)
85	(0, 0)	(0, 0)
86	(2, 0.216929)	(2, 0.788147)
87	(3, 0.120178)	(3, 0.115961)
88	(0, 0)	(0, 0)
89	(1, 0.364639)	(12, 0.115116)
90	(0, 0)	(0, 0)

Table B.3: Trace numbers and maximum values of autocovariance function for error bursts and errorfree bursts

Appendix C

Model Parameters for Markovian and Semi-Markovian Trace Models for Campaign 1

In the following tables we give for the burst orders $k_0 \in \{1, 8, 15, 50, 100\}$ the parameters of two-state markovian and two-state semi-markovian models for every trace. For the markovian model, as discussed in section 2.2.3 for every trace the parameters of two geometric random variables p_X and p_Y are given, determining the state holding times in the good state and bad state, respectively, and also the bit error rate e_b in the bad state. For the semi-markovian models we give for the good states and the bad states the respective distributions and the according parameters, i.e. the success probability p in case of a geometric distribution (geom(p)), the numbers n and success probability p in case of a binomial distribution (binomial(n,p)) and the parameters μ and σ^2 in case of a lognormal distribution (lognormal(μ, σ^2)).

Number	$\mathbf{k_0 = 1 : (p_X, p_Y, e_b)}$	$\mathbf{k_0 = 8 : (p_X, p_Y, e_b)}$
1	(0.00089, 0.70432, 1)	(0.00029, 0.09967, 0.43140)
2	(0.00144, 0.70106, 1)	(0.00037, 0.07304, 0.40540)
3	(0.00186, 0.67958, 1)	(0.00072, 0.12108, 0.45672)
4	(0.00010, 0.76103, 1)	(4.11639e-05, 0.13537, 0.44168)
5	(0.00055, 0.74997, 1)	(0.00028, 0.19029, 0.50274)
6	(1.23956e-05, 0.82496, 1)	(4.42157e-06, 0.12317, 0.41977)
7	(6.14167e-05, 0.49767, 1)	(2.96369e-05, 0.13387, 0.55770)
8	(0.00015, 0.67896, 1)	(5.62397e-05, 0.10611, 0.43543)
9	(9.18537e-05, 0.77308, 1)	(3.75993e-05, 0.14662, 0.46355)
10	(2.88863e-05, 0.98757, 1)	(1.10135e-05, 0.13245, 0.35540)

Number	$\mathbf{k_0 = 1 : (p_X, p_Y, e_b)}$	$\mathbf{k_0 = 8 : (p_X, p_Y, e_b)}$
11	(1.02936e-06, 0.65079, 1)	(4.34070e-07, 0.15044, 0.55752)
12	(7.56562e-07, 0.72289, 1)	(1.86040e-07, 0.06730, 0.39903)
13	(0.00025, 0.79434, 1)	(6.76710e-05, 0.07967, 0.37080)
14	(2.88564e-05, 0.81138, 1)	(1.26282e-05, 0.16922, 0.47684)
15	(4.65521e-05, 0.66587, 1)	(2.12983e-05, 0.16413, 0.53899)
16	(1.60046e-05, 0.58604, 1)	(6.37058e-06, 0.11896, 0.51028)
17	(1.55017e-07, 0.8, 1)	(3.72042e-08, 0.06410, 0.38461)
18	(6.96431e-06, 0.86708, 1)	(3.14965e-06, 0.17134, 0.43741)
19	(2.54302e-07, 0.23809, 1)	(1.30252e-07, 0.08064, 0.67741)
20	(3.09114e-06, 0.53041, 1)	(1.19177e-06, 0.11182, 0.54859)
21	(9.82895e-05, 0.99804, 1)	(3.25029e-05, 0.10987, 0.33301)
22	(3.25782e-06, 0.99809, 1)	(1.52963e-06, 0.18081, 0.38625)
23	(5.82049e-06, 0.79481, 1)	(1.44892e-06, 0.07350, 0.37211)
24	(0.00032, 0.72991, 1)	(0.00012, 0.12642, 0.45237)
25	(0.00033, 0.82779, 1)	(0.00012, 0.13348, 0.42067)
26	(3.96793e-05, 0.90867, 1)	(1.35515e-05, 0.11393, 0.36719)
27	(1.19225e-05, 0.77263, 1)	(2.96095e-06, 0.07393, 0.38546)
28	(0.00010, 0.77731, 1)	(2.85059e-05, 0.08049, 0.38393)
29	(9.19465e-06, 0.86975, 1)	(3.04478e-06, 0.10814, 0.37560)
30	(0.00012, 0.84529, 1)	(4.69026e-05, 0.12599, 0.40664)
31	(9.16340e-05, 0.77757, 1)	(2.88859e-05, 0.10276, 0.41932)
32	(9.46167e-07, 0.80017, 1)	(2.06909e-07, 0.06818, 0.39118)
33	(1.89214e-06, 0.77569, 1)	(6.11311e-07, 0.10398, 0.41541)
34	(1.28934e-07, 0.55405, 1)	(4.05520e-08, 0.08154, 0.47639)
35	(9.04531e-08, 0.68253, 1)	(2.28732e-08, 0.07317, 0.43902)
36	(0.00028, 1, 1)	(0.00010, 0.12542, 0.34779)
37	(0.01623, 0.62003, 1)	(0.00378, 0.06145, 0.44002)
38	(0.01193, 0.61739, 1)	(0.00268, 0.06070, 0.44836)
39	(0.01288, 0.60260, 1)	(0.00240, 0.04964, 0.45242)
40	(0.00962, 0.61541, 1)	(0.00229, 0.06555, 0.45568)
41	(0.00973, 0.62464, 1)	(0.00231, 0.06525, 0.44794)
42	(0.00174, 0.63561, 1)	(0.00040, 0.06352, 0.43205)
43	(0.00455, 0.62244, 1)	(0.00101, 0.06074, 0.44366)
44	(0.00451, 0.62402, 1)	(0.00095, 0.05745, 0.44163)
45	(0.00354, 0.63241, 1)	(0.00086, 0.06798, 0.44587)
46	(0.00243, 0.62786, 1)	(0.00053, 0.06001, 0.43869)
47	(0.00526, 0.62360, 1)	(0.00125, 0.06683, 0.45602)
48	(0.00898, 0.58314, 1)	(0.00134, 0.03991, 0.46414)
49	(0.00232, 0.61260, 1)	(0.00046, 0.05491, 0.45098)
50	(0.00150, 0.64813, 1)	(0.00035, 0.06624, 0.43838)
51	(0.00171, 0.63610, 1)	(0.00035, 0.05785, 0.43651)
52	(0.00165, 0.65973, 1)	(0.00037, 0.06521, 0.43871)
53	(0.00102, 0.66456, 1)	(0.00024, 0.06873, 0.43694)
54	(0.00118, 0.63972, 1)	(0.00024, 0.05921, 0.44560)
55	(0.00095, 0.66331, 1)	(0.00022, 0.06685, 0.43810)
56	(0.00105, 0.65676, 1)	(0.00024, 0.06533, 0.43930)
57	(0.00042, 0.64216, 1)	(0.00010, 0.07167, 0.45522)
58	(0.00113, 0.61557, 1)	(0.00028, 0.07256, 0.46519)

Number	$\mathbf{k_0 = 1 : (p_X, p_Y, e_b)}$	$\mathbf{k_0 = 8 : (p_X, p_Y, e_b)}$
59	(0.00058, 0.65637, 1)	(0.00016, 0.08286, 0.45370)
60	(0.00037, 0.67260, 1)	(0.00011, 0.09043, 0.45217)
61	(0.00063, 0.65248, 1)	(0.00017, 0.07924, 0.45249)
62	(0.00027, 0.64662, 1)	(6.93771e-05, 0.07319, 0.44796)
63	(0.00015, 0.68693, 1)	(4.70245e-05, 0.08919, 0.42982)
64	(0.00023, 0.67037, 1)	(6.46474e-05, 0.07851, 0.43447)
65	(0.00036, 0.63567, 1)	(9.69821e-05, 0.07573, 0.45115)
66	(0.00014, 0.67311, 1)	(3.94988e-05, 0.08213, 0.43494)
67	(0.00025, 0.62601, 1)	(6.29061e-05, 0.07086, 0.45835)
68	(0.00013, 0.65777, 1)	(3.65912e-05, 0.07914, 0.44484)
69	(0.00010, 0.62895, 1)	(2.33391e-05, 0.06364, 0.45321)
70	(0.00020, 0.65814, 1)	(5.66804e-05, 0.08225, 0.44672)
71	(0.00022, 0.65894, 1)	(6.25334e-05, 0.07964, 0.44440)
72	(0.00026, 0.99757, 1)	(9.36721e-05, 0.12354, 0.34638)
73	(5.78732e-08, nan, n/a)	(5.78732e-08, nan, n/a)
74	(5.78732e-08, nan, n/a)	(5.78732e-08, nan, n/a)
75	(5.78732e-08, nan, n/a)	(5.78732e-08, nan, n/a)
76	(3.12516e-06, 1, 1)	(1.33109e-06, 0.15277, 0.36805)
77	(1.92910e-08, nan, n/a)	(1.92910e-08, nan, n/a)
78	(5.34365e-06, 1, 1)	(2.00630e-06, 0.12955, 0.34716)
79	(1.92910e-08, nan, n/a)	(1.92910e-08, nan, n/a)
80	(1.35071e-07, 1, 1)	(3.85918e-08, 0.04545, 0.27272)
81	(5.49799e-06, 1, 1)	(2.37283e-06, 0.15844, 0.36883)
82	(3.47321e-07, 0.23275, 1)	(1.98469e-07, 0.09146, 0.70731)
83	(1.24014e-08, nan, n/a)	(1.24014e-08, nan, n/a)
84	(1.24014e-07, 0.90000, 1)	(3.72042e-08, 0.06451, 0.32258)
85	(1.24014e-08, nan, n/a)	(1.24014e-08, nan, n/a)
86	(1.22860e-06, 0.22072, 1)	(6.82558e-07, 0.08709, 0.71612)
87	(1.11618e-07, 0.34000, 1)	(4.96080e-08, 0.07954, 0.56818)
88	(4.34049e-08, 1, 1)	(1.24014e-08, 0.0625, 0.375)
89	(1.53512e-06, 0.75617, 1)	(3.93143e-07, 0.07579, 0.39608)
90	(n/a, nan, n/a)	(n/a, nan, n/a)

Table C.1: Parameters for Two-State Markovian Models for $k_0 = 1$ and $k_0 = 8$

Number	$\mathbf{k_0 = 15 : (p_X, p_Y, e_b)}$	$\mathbf{k_0 = 50 : (p_X, p_Y, e_b)}$
1	(0.00022, 0.06247, 0.35653)	(0.00021, 0.05841, 0.34198)
2	(0.00035, 0.06641, 0.39082)	(0.00034, 0.06382, 0.38119)
3	(0.00046, 0.05491, 0.32532)	(0.00036, 0.03427, 0.25708)
4	(2.47369e-05, 0.05429, 0.29513)	(2.09064e-05, 0.03799, 0.24449)
5	(0.00014, 0.05338, 0.27291)	(0.00014, 0.04985, 0.26311)
6	(3.12796e-06, 0.06465, 0.31204)	(2.43289e-06, 0.03848, 0.23922)
7	(1.97242e-05, 0.06357, 0.39811)	(1.81503e-05, 0.05249, 0.35730)
8	(4.12504e-05, 0.06273, 0.35103)	(4.03774e-05, 0.05957, 0.34061)
9	(2.20393e-05, 0.05589, 0.30159)	(2.13806e-05, 0.05197, 0.28913)
10	(8.66657e-06, 0.08047, 0.27568)	(2.34750e-06, 0.00923, 0.12394)

Number	$\mathbf{k_0 = 15 : (p_X, p_Y, e_b)}$	$\mathbf{k_0 = 50 : (p_X, p_Y, e_b)}$
11	(2.23236e-07, 0.04644, 0.34426)	(2.23236e-07, 0.04644, 0.34426)
12	(1.73637e-07, 0.06018, 0.38425)	(1.73637e-07, 0.06018, 0.38425)
13	(5.88338e-05, 0.06335, 0.33915)	(5.88090e-05, 0.06328, 0.33894)
14	(7.06571e-06, 0.05805, 0.29265)	(7.04088e-06, 0.05763, 0.29151)
15	(1.10715e-05, 0.05058, 0.31972)	(1.09055e-05, 0.04901, 0.31456)
16	(3.96204e-06, 0.05221, 0.36036)	(3.94947e-06, 0.05190, 0.35935)
17	(3.72042e-08, 0.06410, 0.38461)	(3.72042e-08, 0.06410, 0.38461)
18	(1.96385e-06, 0.07012, 0.28748)	(1.95130e-06, 0.06910, 0.28510)
19	(8.06326e-08, 0.03488, 0.48837)	(8.06326e-08, 0.03488, 0.48837)
20	(7.01410e-07, 0.04584, 0.38354)	(6.88996e-07, 0.04422, 0.37675)
21	(2.16660e-05, 0.05219, 0.23734)	(5.28446e-06, 0.00648, 0.12092)
22	(9.68050e-07, 0.06600, 0.22304)	(1.92372e-07, 0.00585, 0.10082)
23	(1.35253e-06, 0.06577, 0.35677)	(1.35253e-06, 0.06577, 0.35677)
24	(7.69426e-05, 0.05500, 0.31670)	(6.77287e-05, 0.04221, 0.27616)
25	(8.15443e-05, 0.05865, 0.29104)	(5.92786e-05, 0.03087, 0.21085)
26	(9.17483e-06, 0.05540, 0.26375)	(4.76698e-06, 0.01798, 0.16480)
27	(2.84004e-06, 0.06905, 0.37538)	(2.84004e-06, 0.06905, 0.37538)
28	(2.47928e-05, 0.06403, 0.35118)	(2.47897e-05, 0.06401, 0.35107)
29	(1.76271e-06, 0.04277, 0.25674)	(1.31158e-06, 0.02542, 0.20516)
30	(2.91785e-05, 0.05362, 0.27827)	(2.13327e-05, 0.02933, 0.20822)
31	(2.18077e-05, 0.06335, 0.34246)	(2.18067e-05, 0.06334, 0.34241)
32	(2.04830e-07, 0.06703, 0.38850)	(2.03790e-07, 0.06570, 0.38274)
33	(4.44968e-07, 0.06040, 0.33173)	(4.44968e-07, 0.06040, 0.33173)
34	(3.22336e-08, 0.05464, 0.40437)	(2.91142e-08, 0.04265, 0.35071)
35	(2.07938e-08, 0.06109, 0.40514)	(1.97541e-08, 0.05438, 0.38066)
36	(7.73169e-05, 0.07106, 0.26040)	(2.47051e-05, 0.01059, 0.12349)
37	(0.00242, 0.03237, 0.36669)	(0.00164, 0.01655, 0.28243)
38	(0.00163, 0.03021, 0.36981)	(0.00106, 0.01490, 0.28474)
39	(0.00143, 0.02475, 0.38298)	(0.00093, 0.01282, 0.30818)
40	(0.00146, 0.03422, 0.37583)	(0.00102, 0.01806, 0.28772)
41	(0.00147, 0.03418, 0.37021)	(0.00097, 0.01672, 0.27843)
42	(0.00026, 0.03466, 0.36276)	(0.00018, 0.01863, 0.28013)
43	(0.00064, 0.03263, 0.37334)	(0.00043, 0.01664, 0.28550)
44	(0.00059, 0.03001, 0.37166)	(0.00036, 0.01371, 0.27849)
45	(0.00055, 0.03609, 0.36922)	(0.00039, 0.01977, 0.28421)
46	(0.00034, 0.03293, 0.37134)	(0.00024, 0.01820, 0.29114)
47	(0.00080, 0.03566, 0.37831)	(0.00055, 0.01822, 0.28495)
48	(0.00077, 0.01973, 0.40287)	(0.00042, 0.00852, 0.32293)
49	(0.00029, 0.03006, 0.38538)	(0.00020, 0.01651, 0.30758)
50	(0.00023, 0.03652, 0.36680)	(0.00015, 0.01891, 0.27776)
51	(0.00022, 0.03139, 0.36961)	(0.00014, 0.01563, 0.28329)
52	(0.00022, 0.03243, 0.35973)	(0.00014, 0.01610, 0.27406)
53	(0.00014, 0.03456, 0.35671)	(9.87931e-05, 0.01727, 0.27012)
54	(0.00015, 0.03081, 0.37388)	(0.00010, 0.01642, 0.29388)
55	(0.00013, 0.03341, 0.35847)	(9.12183e-05, 0.01734, 0.27546)
56	(0.00014, 0.03270, 0.36059)	(9.63204e-05, 0.01638, 0.27512)
57	(6.19105e-05, 0.03438, 0.36382)	(4.38561e-05, 0.01902, 0.28421)
58	(0.00018, 0.03836, 0.38022)	(0.00013, 0.02193, 0.29475)

Number	$\mathbf{k_0 = 15 : (p_X, p_Y, e_b)}$	$\mathbf{k_0 = 50 : (p_X, p_Y, e_b)}$
59	(0.00010, 0.03995, 0.35610)	(8.02668e-05, 0.02571, 0.28840)
60	(6.77723e-05, 0.04158, 0.34440)	(5.23490e-05, 0.02529, 0.27137)
61	(0.00010, 0.03913, 0.35921)	(7.75074e-05, 0.02180, 0.27504)
62	(4.38854e-05, 0.03766, 0.36449)	(3.31466e-05, 0.02247, 0.28809)
63	(3.06410e-05, 0.04591, 0.33962)	(2.40762e-05, 0.02809, 0.26454)
64	(4.08983e-05, 0.03983, 0.34852)	(3.13479e-05, 0.02416, 0.27593)
65	(6.33387e-05, 0.04035, 0.36817)	(4.74787e-05, 0.02341, 0.28516)
66	(2.52877e-05, 0.04193, 0.34692)	(1.91843e-05, 0.02477, 0.27022)
67	(3.97959e-05, 0.03653, 0.37356)	(2.80841e-05, 0.01973, 0.28610)
68	(2.33706e-05, 0.04046, 0.35617)	(1.71504e-05, 0.02279, 0.27347)
69	(1.47215e-05, 0.03335, 0.37661)	(1.13610e-05, 0.02128, 0.31146)
70	(3.63044e-05, 0.04185, 0.35498)	(2.63811e-05, 0.02293, 0.26773)
71	(3.91518e-05, 0.03953, 0.35240)	(2.75894e-05, 0.02094, 0.26507)
72	(6.47679e-05, 0.06012, 0.24395)	(2.14750e-05, 0.01041, 0.12784)
73	(5.78732e-08, nan, n/a)	(5.78732e-08, nan, n/a)
74	(5.78732e-08, nan, n/a)	(5.78732e-08, nan, n/a)
75	(5.78732e-08, nan, n/a)	(5.78732e-08, nan, n/a)
76	(7.52363e-07, 0.04724, 0.20866)	(2.89373e-07, 0.00947, 0.12559)
77	(1.92910e-08, nan, n/a)	(1.92910e-08, nan, n/a)
78	(1.69764e-06, 0.08959, 0.28424)	(4.43716e-07, 0.00882, 0.11075)
79	(1.92910e-08, nan, n/a)	(1.92910e-08, nan, n/a)
80	(3.85918e-08, 0.04545, 0.27272)	(3.85918e-08, 0.04545, 0.27272)
81	(2.00631e-06, 0.10520, 0.29009)	(4.24429e-07, 0.00663, 0.08967)
82	(1.11639e-07, 0.03225, 0.46774)	(1.11639e-07, 0.03225, 0.46774)
83	(1.24014e-08, nan, n/a)	(1.24014e-08, nan, n/a)
84	(3.72042e-08, 0.06451, 0.32258)	(3.72042e-08, 0.06451, 0.32258)
85	(1.24014e-08, nan, n/a)	(1.24014e-08, nan, n/a)
86	(3.84716e-07, 0.03303, 0.48898)	(3.84716e-07, 0.03303, 0.48898)
87	(3.72060e-08, 0.04464, 0.44642)	(3.72060e-08, 0.04464, 0.44642)
88	(1.24014e-08, 0.0625, 0.375)	(1.24014e-08, 0.0625, 0.375)
89	(3.61941e-07, 0.06612, 0.37587)	(3.55701e-07, 0.06147, 0.35565)
90	(n/a, nan, n/a)	(n/a, nan, n/a)

Table C.2: Parameters for Two-State Markovian Models for $k_0 = 15$ and $k_0 = 50$

Number	$\mathbf{k_0 = 15 : (p_X, p_Y, e_b)}$
1	(0.00020, 0.04695, 0.28669)
2	(0.00033, 0.05365, 0.33001)
3	(0.00034, 0.02645, 0.21526)
4	(2.06745e-05, 0.03630, 0.23622)
5	(0.00013, 0.04731, 0.25232)
6	(2.41359e-06, 0.03715, 0.23284)
7	(1.78741e-05, 0.04847, 0.33506)
8	(4.00736e-05, 0.05705, 0.32864)
9	(2.12258e-05, 0.04997, 0.28003)

Number	$\mathbf{k_0 = 15 : (p_X, p_Y, e_b)}$
10	(2.16694e-06, 0.00811, 0.11873)
11	(2.23236e-07, 0.04644, 0.34426)
12	(1.73637e-07, 0.06018, 0.38425)
13	(5.87966e-05, 0.06321, 0.33864)
14	(7.00364e-06, 0.05603, 0.28496)
15	(1.07651e-05, 0.04624, 0.30065)
16	(3.93061e-06, 0.05049, 0.35127)
17	(3.72042e-08, 0.06410, 0.38461)
18	(1.93875e-06, 0.06633, 0.27546)
19	(8.06326e-08, 0.03488, 0.48837)
20	(6.88996e-07, 0.04422, 0.37675)
21	(6.45676e-07, 0.00053, 0.08250)
22	(6.20567e-09, 6.20809e-05, 0.06524)
23	(1.34942e-06, 0.06513, 0.35409)
24	(6.60077e-05, 0.03771, 0.25323)
25	(5.38569e-05, 0.02264, 0.17027)
26	(3.44925e-06, 0.00904, 0.11454)
27	(2.83849e-06, 0.06877, 0.37403)
28	(2.47681e-05, 0.06362, 0.34927)
29	(1.12556e-06, 0.01625, 0.15286)
30	(1.90443e-05, 0.02050, 0.16308)
31	(2.17994e-05, 0.06320, 0.34178)
32	(2.03790e-07, 0.06570, 0.38274)
33	(4.44968e-07, 0.06040, 0.33173)
34	(2.91142e-08, 0.04265, 0.35071)
35	(1.97541e-08, 0.05438, 0.38066)
36	(2.47051e-05, 0.01059, 0.12349)
37	(0.00132, 0.01042, 0.22592)
38	(0.00087, 0.01014, 0.23871)
39	(0.00076, 0.00887, 0.26343)
40	(0.00083, 0.01153, 0.22929)
41	(0.00080, 0.01119, 0.22877)
42	(0.00016, 0.01388, 0.24008)
43	(0.00035, 0.01111, 0.23435)
44	(0.00029, 0.00927, 0.23416)
45	(0.00032, 0.01299, 0.22722)
46	(0.00020, 0.01283, 0.24274)
47	(0.00043, 0.01101, 0.22128)
48	(0.00029, 0.00502, 0.27342)
49	(0.00017, 0.01210, 0.26347)
50	(0.00013, 0.01392, 0.23630)
51	(0.00012, 0.01095, 0.23924)
52	(0.00011, 0.01061, 0.22459)
53	(8.25664e-05, 0.01204, 0.22555)
54	(8.63578e-05, 0.01146, 0.24666)
55	(7.62751e-05, 0.01210, 0.23007)
56	(7.96607e-05, 0.01128, 0.22944)
57	(3.73654e-05, 0.01361, 0.23881)

Number	$\mathbf{k_0 = 15 : (p_X, p_Y, e_b)}$
58	(0.00010, 0.01330, 0.22419)
59	(6.83078e-05, 0.01717, 0.22659)
60	(4.40322e-05, 0.01670, 0.21313)
61	(6.23245e-05, 0.01349, 0.21198)
62	(2.76772e-05, 0.01483, 0.22772)
63	(2.09654e-05, 0.01944, 0.21028)
64	(2.68397e-05, 0.01656, 0.22093)
65	(3.92041e-05, 0.01495, 0.22066)
66	(1.69932e-05, 0.01832, 0.22572)
67	(2.32520e-05, 0.01313, 0.23008)
68	(1.46054e-05, 0.01566, 0.22064)
69	(1.01157e-05, 0.01627, 0.26749)
70	(2.20355e-05, 0.01507, 0.21082)
71	(2.27114e-05, 0.01363, 0.20969)
72	(1.92721e-05, 0.00874, 0.11963)
73	(5.78732e-08, nan, n/a)
74	(5.78732e-08, nan, n/a)
75	(5.78732e-08, nan, n/a)
76	(2.89373e-07, 0.00947, 0.12559)
77	(1.92910e-08, nan, n/a)
78	(4.05133e-07, 0.00762, 0.10518)
79	(1.92910e-08, nan, n/a)
80	(3.85918e-08, 0.04545, 0.27272)
81	(4.05138e-07, 0.00620, 0.08808)
82	(1.11639e-07, 0.03225, 0.46774)
83	(1.24014e-08, nan, n/a)
84	(3.72042e-08, 0.06451, 0.32258)
85	(1.24014e-08, nan, n/a)
86	(3.84716e-07, 0.03303, 0.48898)
87	(3.72060e-08, 0.04464, 0.44642)
88	(1.24014e-08, 0.0625, 0.375)
89	(3.55701e-07, 0.06147, 0.35565)
90	(n/a, nan, n/a)

Table C.3: Parameters for Two-State Markovian Models for $k_0 = 100$

Number	**Error Bursts**	**Errorfree Bursts**
1	geom(0.70432)	lognormal(4.81406, 4.39900)
2	geom(0.70106)	lognormal(3.35844, 6.36769)
3	geom(0.67958)	lognormal(2.97831, 6.61420)
4	geom(0.76103)	lognormal(6.72093, 4.93707)
5	geom(0.74997)	lognormal(5.06832, 4.85714)
6	geom(0.82496)	lognormal(8.47806, 5.64020)
7	lognormal(0.38011, 0.63540)	lognormal(6.98874, 5.41815)
8	geom(0.67896)	lognormal(5.83091, 5.86175)
9	geom(0.77308)	lognormal(6.07189, 6.44683)

Number	Error Bursts	Errorfree Bursts
10	geom(0.98757)	lognormal(8.27341, 4.35745)
11	geom(0.65079)	lognormal(12.73038, 2.11236)
12	geom(0.72289)	lognormal(13.11192, 1.96510)
13	geom(0.79434)	lognormal(4.30151, 7.98504)
14	geom(0.81138)	lognormal(7.49564, 5.91507)
15	geom(0.66587)	lognormal(7.42952, 5.09082)
16	lognormal(0.26311, 0.54251)	lognormal(8.31454, 5.45616)
17	geom(0.8)	lognormal(14.06954, 3.22037)
18	geom(0.86708)	lognormal(8.87207, 6.00527)
19	lognormal(1.32464, 0.22087)	lognormal(13.79547, 2.77852)
20	lognormal(0.33843, 0.59131)	lognormal(10.61722, 4.13949)
21	geom(0.99804)	lognormal(4.85912, 8.73693)
22	geom(0.99809)	lognormal(9.39327, 6.48235)
23	geom(0.79481)	lognormal(8.40814, 7.29196)
24	geom(0.72991)	lognormal(4.34393, 7.38769)
25	geom(0.82779)	lognormal(3.40659, 9.19182)
26	geom(0.90867)	lognormal(5.38203, 9.50528)
27	geom(0.77263)	lognormal(8.00835, 6.65744)
28	geom(0.77731)	lognormal(5.19296, 7.92474)
29	geom(0.86975)	lognormal(9.55538, 4.08299)
30	geom(0.84529)	lognormal(5.39516, 7.13772)
31	geom(0.77757)	lognormal(5.60512, 7.38516)
32	geom(0.80017)	lognormal(11.60478, 4.53211)
33	geom(0.77569)	lognormal(10.22632, 5.90294)
34	geom(0.55405)	lognormal(14.18321, 3.36148)
35	geom(0.68253)	lognormal(14.71177, 3.01331)
36	geom(1)	lognormal(5.99841, 4.35303)
37	geom(0.62003)	lognormal(2.77479, 2.69197)
38	geom(0.61739)	lognormal(3.01685, 2.82226)
39	geom(0.60260)	lognormal(2.67375, 3.35517)
40	geom(0.61541)	lognormal(3.14605, 2.99474)
41	geom(0.62464)	lognormal(3.34878, 2.56550)
42	geom(0.63561)	lognormal(4.84056, 3.01908)
43	geom(0.62244)	lognormal(3.82989, 3.12162)
44	geom(0.62402)	lognormal(4.54784, 1.70362)
45	geom(0.63241)	lognormal(4.29879, 2.68645)
46	geom(0.62786)	lognormal(4.39712, 3.24230)
47	geom(0.62360)	lognormal(4.11708, 2.25843)
48	geom(0.58314)	lognormal(1.65620, 6.11207)
49	geom(0.61260)	lognormal(4.68805, 2.74815)
50	geom(0.64813)	lognormal(4.99367, 3.00720)
51	geom(0.63610)	lognormal(4.76431, 3.21293)
52	geom(0.65973)	lognormal(4.85020, 3.11029)
53	geom(0.66456)	lognormal(5.40467, 2.96253)
54	geom(0.63972)	lognormal(5.20424, 3.07320)
55	geom(0.66331)	lognormal(5.38275, 3.13733)
56	geom(0.65676)	lognormal(5.32314, 3.05688)
57	geom(0.64216)	lognormal(6.30270, 2.94460)

Number	Error Bursts	Errorfree Bursts
58	geom(0.61557)	lognormal(4.42009, 4.72644)
59	geom(0.65637)	lognormal(6.08591, 2.70009)
60	geom(0.67260)	lognormal(6.50470, 2.75638)
61	geom(0.65248)	lognormal(5.76750, 3.18441)
62	geom(0.64662)	lognormal(6.69431, 3.01326)
63	geom(0.68693)	lognormal(7.44071, 2.65488)
64	geom(0.67037)	lognormal(6.89308, 2.88618)
65	geom(0.63567)	lognormal(6.04884, 3.72263)
66	geom(0.67311)	lognormal(7.50266, 2.73172)
67	geom(0.62601)	lognormal(6.94094, 2.66989)
68	geom(0.65777)	lognormal(7.24839, 3.31999)
69	geom(0.62895)	lognormal(7.68554, 2.96151)
70	geom(0.65814)	lognormal(6.99823, 3.01285)
71	geom(0.65894)	lognormal(7.13873, 2.47895)
72	geom(0.99757)	lognormal(5.25932, 5.97753)
73	lognormal(nan, nan)	lognormal(16.66501, 0)
74	lognormal(nan, nan)	lognormal(16.66501, 0)
75	lognormal(nan, nan)	lognormal(16.66501, 0)
76	geom(1)	lognormal(10.90764, 3.53675)
77	lognormal(nan, nan)	lognormal(17.76362, 0)
78	geom(1)	lognormal(9.52053, 5.23812)
79	lognormal(nan, nan)	lognormal(17.76362, 0)
80	geom(1)	lognormal(14.96541, 1.70408)
81	geom(1)	lognormal(9.49779, 5.22666)
82	lognormal(1.29075, 0.33400)	lognormal(13.54575, 2.65452)
83	lognormal(nan, nan)	lognormal(18.20545, 0)
84	geom(0.90000)	lognormal(14.99774, 1.81024)
85	lognormal(nan, nan)	lognormal(18.20545, 0)
86	lognormal(1.36443, 0.29284)	lognormal(11.80708, 3.60509)
87	lognormal(0.64525, 0.86710)	lognormal(14.87683, 2.26269)
88	geom(1)	lognormal(16.15875, 1.58788)
89	geom(0.75617)	lognormal(11.31093, 4.15192)

Table C.4: Distributions and Parameters for Semimarkovian Models for $k_0 = 1$

Number	Error Bursts	Errorfree Bursts
1	lognormal(2.07113, 0.46935)	lognormal(6.48231, 3.28804)
2	lognormal(2.54992, 0.13344)	lognormal(5.39008, 5.01535)
3	lognormal(1.72624, 0.76997)	lognormal(4.38362, 5.67954)
4	lognormal(1.58796, 0.82352)	lognormal(8.08315, 4.02959)
5	lognormal(1.21514, 0.88803)	lognormal(6.09234, 4.17501)
6	lognormal(1.82697, 0.53433)	lognormal(10.02293, 4.61216)
7	lognormal(1.67837, 0.66499)	lognormal(8.08150, 4.68996)
8	lognormal(1.98386, 0.51868)	lognormal(7.36679, 4.83818)
9	lognormal(1.56755, 0.70460)	lognormal(7.41149, 5.55406)
10	lognormal(1.67273, 0.69762)	lognormal(9.71486, 3.40305)

Number	Error Bursts	Errorfree Bursts
11	lognormal(1.55404, 0.68026)	lognormal(14.02060, 1.25890)
12	lognormal(2.60952, 0.17790)	lognormal(15.20841, 0.57778)
13	lognormal(2.44275, 0.17409)	lognormal(6.26140, 6.67890)
14	lognormal(1.37658, 0.79993)	lognormal(8.73493, 5.08926)
15	lognormal(1.40686, 0.80035)	lognormal(8.60225, 4.30926)
16	lognormal(1.80824, 0.64142)	lognormal(9.69602, 4.53558)
17	binomial(16,0.975)	lognormal(16.15326, 1.90714)
18	lognormal(1.36147, 0.80525)	lognormal(10.06178, 5.21286)
19	lognormal(2.30517, 0.42504)	lognormal(14.78919, 2.12919)
20	lognormal(1.81186, 0.75787)	lognormal(12.04535, 3.18942)
21	lognormal(1.83890, 0.73895)	lognormal(6.51887, 7.63060)
22	lognormal(1.41461, 0.59131)	lognormal(10.52678, 5.72740)
23	lognormal(2.56773, 0.08530)	lognormal(10.49318, 5.90301)
24	lognormal(1.63168, 0.87283)	lognormal(5.68454, 6.62542)
25	lognormal(1.67809, 0.67130)	lognormal(4.87114, 8.17930)
26	lognormal(1.86680, 0.61062)	lognormal(6.99345, 8.43110)
27	binomial(287,0.04712)	lognormal(10.09753, 5.26493)
28	lognormal(2.43349, 0.17201)	lognormal(7.15792, 6.61493)
29	lognormal(1.95969, 0.52923)	lognormal(11.21301, 2.97814)
30	lognormal(1.74795, 0.64715)	lognormal(6.90016, 6.13454)
31	lognormal(2.10162, 0.34741)	lognormal(7.33670, 6.23090)
32	binomial(21,0.69841)	lognormal(13.88314, 3.01568)
33	lognormal(2.07474, 0.37751)	lognormal(11.92054, 4.77422)
34	lognormal(2.38499, 0.24319)	lognormal(15.91064, 2.22007)
35	lognormal(2.56981, 0.09029)	lognormal(16.76079, 1.66501)
36	lognormal(1.70539, 0.74131)	lognormal(7.52043, 3.34088)
37	lognormal(2.33673, 0.90547)	lognormal(4.82346, 1.50798)
38	lognormal(2.29548, 1.01242)	lognormal(5.09277, 1.65683)
39	lognormal(2.33981, 1.32594)	lognormal(5.17495, 1.70660)
40	lognormal(2.25844, 0.93296)	lognormal(5.29727, 1.56156)
41	lognormal(2.24153, 0.97575)	lognormal(5.25491, 1.62587)
42	lognormal(2.24827, 1.01593)	lognormal(7.02907, 1.56251)
43	lognormal(2.24594, 1.11014)	lognormal(6.08658, 1.61824)
44	lognormal(1.96943, 1.77477)	lognormal(5.97317, 1.97051)
45	lognormal(2.16283, 1.05126)	lognormal(6.21966, 1.67580)
46	lognormal(2.21456, 1.19721)	lognormal(6.70880, 1.65652)
47	lognormal(2.10237, 1.20643)	lognormal(5.34076, 2.68709)
48	lognormal(2.39150, 1.65886)	lognormal(5.25424, 2.70820)
49	lognormal(1.90872, 1.98661)	lognormal(6.84927, 1.64772)
50	lognormal(2.31834, 0.79197)	lognormal(7.19218, 1.51617)
51	lognormal(2.33459, 1.03045)	lognormal(7.12169, 1.62819)
52	lognormal(2.05765, 1.34475)	lognormal(7.08464, 1.61540)
53	lognormal(2.11053, 1.13388)	lognormal(7.52746, 1.59474)
54	lognormal(1.62264, 2.40794)	lognormal(7.51769, 1.58485)
55	lognormal(2.09650, 1.21748)	lognormal(7.61234, 1.61327)
56	lognormal(2.14648, 1.16351)	lognormal(7.51021, 1.64907)
57	lognormal(2.17169, 0.92771)	lognormal(8.42747, 1.50481)
58	lognormal(2.17015, 0.90624)	lognormal(6.90399, 2.49996)

Number	**Error Bursts**	**Errorfree Bursts**
59	lognormal(1.55489, 1.87134)	lognormal(7.87504, 1.67811)
60	lognormal(1.96984, 0.86645)	lognormal(8.32111, 1.54775)
61	lognormal(2.07766, 0.91500)	lognormal(7.83356, 1.68036)
62	lognormal(1.63617, 1.95679)	lognormal(8.59745, 1.95698)
63	lognormal(2.04116, 0.75146)	lognormal(9.23614, 1.45738)
64	lognormal(1.99575, 1.09730)	lognormal(8.87908, 1.53494)
65	lognormal(2.00362, 1.15384)	lognormal(8.28889, 1.90417)
66	lognormal(2.03277, 0.93320)	lognormal(9.40858, 1.46130)
67	lognormal(1.46344, 2.36705)	lognormal(8.84825, 1.65122)
68	lognormal(2.08532, 0.90236)	lognormal(9.30292, 1.82554)
69	lognormal(1.16293, 3.18293)	lognormal(9.93257, 1.46561)
70	lognormal(2.10719, 0.78150)	lognormal(8.92576, 1.70463)
71	lognormal(2.09577, 0.86877)	lognormal(8.82304, 1.71352)
72	lognormal(1.69237, 0.79751)	lognormal(6.80030, 4.95080)
73	lognormal(nan, nan)	lognormal(16.66501, 0)
74	lognormal(nan, nan)	lognormal(16.66501, 0)
75	lognormal(nan, nan)	lognormal(16.66501, 0)
76	lognormal(1.55362, 0.65028)	lognormal(12.17623, 2.70655)
77	lognormal(nan, nan)	lognormal(17.76362, 0)
78	lognormal(1.56846, 0.95029)	lognormal(10.98699, 4.26444)
79	lognormal(nan, nan)	lognormal(17.76362, 0)
80	binomial(22,1)	lognormal(16.78368, 0.57308)
81	lognormal(1.49649, 0.69173)	lognormal(10.75595, 4.39094)
82	lognormal(2.22965, 0.32433)	lognormal(14.37378, 2.11769)
83	lognormal(nan, nan)	lognormal(18.20545, 0)
84	binomial(16,0.96875)	lognormal(16.75521, 0.70326)
85	lognormal(nan, nan)	lognormal(18.20545, 0)
86	lognormal(2.27239, 0.33668)	lognormal(12.68497, 3.02489)
87	lognormal(2.43039, 0.20205)	lognormal(16.06763, 1.50295)
88	binomial(16,1)	lognormal(18.00997, 0.39095)
89	lognormal(2.49457, 0.17029)	lognormal(13.34871, 2.80074)

Table C.5: Distributions and Parameters for Semimarkovian Models for $k_0 = 8$

Number	**Error Bursts**	**Errorfree Bursts**
1	lognormal(2.66337, 0.21908)	lognormal(6.89569, 3.01291)
2	lognormal(2.66232, 0.09896)	lognormal(5.47747, 4.95722)
3	lognormal(2.63924, 0.52528)	lognormal(5.05573, 5.23313)
4	lognormal(2.71016, 0.40616)	lognormal(8.84652, 3.52136)
5	lognormal(2.88986, 0.08089)	lognormal(7.07978, 3.51767)
6	lognormal(2.59835, 0.28055)	lognormal(10.54121, 4.26783)
7	lognormal(2.56073, 0.38966)	lognormal(8.69204, 4.28323)
8	lognormal(2.69390, 0.14996)	lognormal(7.83160, 4.52848)
9	lognormal(2.83749, 0.09368)	lognormal(8.21247, 5.02040)
10	lognormal(2.13944, 0.76061)	lognormal(10.07223, 3.16760)
11	lognormal(2.99338, 0.15207)	lognormal(15.01573, 0.59859)

Number	Error Bursts	Errorfree Bursts
12	lognormal(2.74205, 0.13654)	lognormal(15.31231, 0.50795)
13	binomial(31,0.50919)	lognormal(6.47125, 6.53907)
14	lognormal(2.74528, 0.20199)	lognormal(9.60555, 4.50939)
15	lognormal(2.91692, 0.13450)	lognormal(9.58331, 3.65563)
16	lognormal(2.81566, 0.27337)	lognormal(10.40811, 4.06126)
17	binomial(16,0.975)	lognormal(16.15326, 1.90714)
18	lognormal(2.61509, 0.08467)	lognormal(10.76976, 4.74167)
19	lognormal(3.24703, 0.21739)	lognormal(15.49773, 1.67126)
20	lognormal(2.96428, 0.23640)	lognormal(12.83887, 2.66258)
21	lognormal(2.42650, 1.05256)	lognormal(7.12716, 7.22518)
22	lognormal(2.29756, 0.84101)	lognormal(11.21244, 5.27107)
23	binomial(21,0.72394)	lognormal(10.59636, 5.83433)
24	lognormal(2.65347, 0.49360)	lognormal(6.39711, 6.15066)
25	lognormal(2.57843, 0.51540)	lognormal(5.55127, 7.72618)
26	lognormal(2.42827, 0.92965)	lognormal(7.57845, 8.04117)
27	binomial(24,0.60335)	lognormal(10.16005, 5.22326)
28	binomial(28,0.55770)	lognormal(7.36724, 6.47542)
29	lognormal(2.62722, 1.04903)	lognormal(12.03270, 2.43189)
30	lognormal(2.61292, 0.62553)	lognormal(7.61202, 5.66010)
31	binomial(172,0.09176)	lognormal(7.75829, 5.94989)
32	binomial(20,0.74591)	lognormal(13.89827, 3.00562)
33	binomial(57,0.29043)	lognormal(12.39663, 4.45724)
34	lognormal(2.77724, 0.25931)	lognormal(16.25243, 1.99564)
35	lognormal(2.66817, 0.25436)	lognormal(16.90231, 1.57258)
36	lognormal(2.14026, 1.00776)	lognormal(7.93458, 3.06603)
37	lognormal(2.95140, 0.95816)	lognormal(5.48682, 1.07393)
38	lognormal(2.96999, 1.05854)	lognormal(5.83136, 1.17069)
39	lognormal(2.96945, 1.45836)	lognormal(5.95110, 1.19504)
40	lognormal(2.87486, 0.99997)	lognormal(5.96827, 1.11918)
41	lognormal(2.85743, 1.03698)	lognormal(5.92386, 1.18488)
42	lognormal(2.75395, 1.21595)	lognormal(7.67293, 1.13409)
43	lognormal(2.76762, 1.30940)	lognormal(6.75340, 1.17579)
44	lognormal(2.52235, 1.96721)	lognormal(6.68188, 1.50012)
45	lognormal(2.70617, 1.23070)	lognormal(6.88101, 1.23666)
46	lognormal(2.72186, 1.38259)	lognormal(7.35551, 1.22646)
47	lognormal(2.56839, 1.53046)	lognormal(5.99489, 2.25357)
48	lognormal(2.77191, 2.30660)	lognormal(6.10974, 2.11228)
49	lognormal(2.31785, 2.37301)	lognormal(7.51409, 1.20548)
50	lognormal(2.86457, 0.89043)	lognormal(7.81588, 1.10107)
51	lognormal(2.80645, 1.30924)	lognormal(7.78674, 1.18558)
52	lognormal(2.58389, 1.68951)	lognormal(7.83230, 1.11780)
53	lognormal(2.75376, 1.22222)	lognormal(8.25264, 1.11182)
54	lognormal(1.83411, 3.29120)	lognormal(8.23246, 1.10888)
55	lognormal(2.71865, 1.36033)	lognormal(8.35032, 1.12178)
56	lognormal(2.76328, 1.31398)	lognormal(8.25046, 1.15612)
57	lognormal(2.93953, 0.86093)	lognormal(9.19219, 0.99525)
58	lognormal(2.74713, 1.02684)	lognormal(7.55558, 2.06616)
59	lognormal(1.84285, 2.75407)	lognormal(8.60456, 1.19213)

Number	Error Bursts	Errorfree Bursts
60	lognormal(2.83095, 0.69828)	lognormal(9.07746, 1.04378)
61	lognormal(2.76587, 0.94970)	lognormal(8.54449, 1.20678)
62	lognormal(1.27231, 4.01332)	lognormal(9.28431, 1.49923)
63	lognormal(2.79303, 0.57580)	lognormal(9.87856, 1.02920)
64	lognormal(2.64756, 1.15065)	lognormal(9.56577, 1.07729)
65	lognormal(2.42328, 1.57367)	lognormal(8.92779, 1.47843)
66	lognormal(2.77180, 0.79954)	lognormal(10.07744, 1.01548)
67	lognormal(0.96055, 4.69812)	lognormal(9.53496, 1.19355)
68	lognormal(2.75515, 0.90420)	lognormal(9.97535, 1.37734)
69	lognormal(0.82192, 5.15708)	lognormal(10.62375, 1.00488)
70	lognormal(2.81135, 0.72431)	lognormal(9.59390, 1.25933)
71	lognormal(2.82714, 0.80687)	lognormal(9.52532, 1.24548)
72	lognormal(2.26699, 1.08863)	lognormal(7.35349, 4.58240)
73	lognormal(nan, nan)	lognormal(16.66501, 0)
74	lognormal(nan, nan)	lognormal(16.66501, 0)
75	lognormal(nan, nan)	lognormal(16.66501, 0)
76	lognormal(2.54167, 1.02149)	lognormal(13.01748, 2.16512)
77	lognormal(nan, nan)	lognormal(17.76362, 0)
78	lognormal(1.89802, 1.02878)	lognormal(11.23671, 4.09911)
79	lognormal(nan, nan)	lognormal(17.76362, 0)
80	binomial(22,1)	lognormal(16.78368, 0.57308)
81	lognormal(1.54831, 1.40698)	lognormal(11.00690, 4.22461)
82	lognormal(3.30654, 0.25489)	lognormal(15.21894, 1.57809)
83	lognormal(nan, nan)	lognormal(18.20545, 0)
84	binomial(16,0.96875)	lognormal(16.75521, 0.70326)
85	lognormal(nan, nan)	lognormal(18.20545, 0)
86	lognormal(3.29941, 0.22125)	lognormal(13.53873, 2.46405)
87	lognormal(3.00100, 0.21610)	lognormal(16.48727, 1.23903)
88	binomial(16,1)	lognormal(18.00997, 0.39095)
89	lognormal(2.67888, 0.07463)	lognormal(13.47214, 2.71927)

Table C.6: Distributions and Parameters for Semimarkovian Models for $k_0 = 15$

Number	Error Bursts	Errorfree Bursts
1	lognormal(2.64914, 0.38205)	lognormal(6.93376, 2.98763)
2	lognormal(2.65396, 0.19522)	lognormal(5.49946, 4.94265)
3	lognormal(2.87359, 0.99927)	lognormal(5.40506, 5.00176)
4	lognormal(2.83339, 0.87367)	lognormal(9.09864, 3.35362)
5	lognormal(2.84989, 0.29730)	lognormal(7.12708, 3.48621)
6	lognormal(2.76286, 0.98922)	lognormal(10.91729, 4.01826)
7	lognormal(2.71199, 0.47007)	lognormal(8.81672, 4.20019)
8	lognormal(2.69237, 0.25615)	lognormal(7.86367, 4.50712)
9	lognormal(2.77825, 0.35733)	lognormal(8.25797, 4.99010)
10	lognormal(4.56901, 0.23085)	lognormal(12.00754, 1.90922)
11	lognormal(2.99338, 0.15207)	lognormal(15.01573, 0.59859)

Number	Error Bursts	Errorfree Bursts
12	lognormal(2.74205, 0.13654)	lognormal(15.31231, 0.50795)
13	binomial(34,0.46475)	lognormal(6.47189, 6.53864)
14	lognormal(2.73356, 0.24023)	lognormal(9.61083, 4.50587)
15	lognormal(2.94652, 0.13811)	lognormal(9.60596, 3.64054)
16	lognormal(2.82146, 0.27378)	lognormal(10.41288, 4.05808)
17	binomial(16,0.975)	lognormal(16.15326, 1.90714)
18	lognormal(2.62165, 0.10097)	lognormal(10.77937, 4.73528)
19	lognormal(3.24703, 0.21739)	lognormal(15.49773, 1.67126)
20	lognormal(2.96924, 0.29820)	lognormal(12.86559, 2.64486)
21	lognormal(4.94787, 0.18221)	lognormal(9.24320, 5.81505)
22	lognormal(5.09991, 0.08215)	lognormal(13.62988, 3.66789)
23	binomial(21,0.72394)	lognormal(10.59636, 5.83433)
24	lognormal(2.62551, 1.07895)	lognormal(6.58833, 6.02333)
25	lognormal(2.89834, 1.15905)	lognormal(6.02934, 7.40783)
26	lognormal(3.54070, 0.95542)	lognormal(8.56045, 7.38667)
27	binomial(24,0.60335)	lognormal(10.16005, 5.22326)
28	binomial(31,0.50394)	lognormal(7.36743, 6.47529)
29	lognormal(3.10019, 1.14340)	lognormal(12.47601, 2.13652)
30	lognormal(2.93135, 1.19548)	lognormal(8.08171, 5.34711)
31	binomial(221,0.07143)	lognormal(7.75836, 5.94984)
32	lognormal(2.66809, 0.10910)	lognormal(13.90589, 3.00056)
33	binomial(57,0.29043)	lognormal(12.39663, 4.45724)
34	lognormal(2.84503, 0.61919)	lognormal(16.40382, 1.89642)
35	lognormal(2.68282, 0.45783)	lognormal(16.97846, 1.52288)
36	lognormal(4.42532, 0.24461)	lognormal(9.63912, 1.93875)
37	lognormal(3.57350, 1.05502)	lognormal(6.05614, 0.70904)
38	lognormal(3.65908, 1.09416)	lognormal(6.46713, 0.75754)
39	lognormal(3.68363, 1.34481)	lognormal(6.58305, 0.78305)
40	lognormal(3.43614, 1.15532)	lognormal(6.49971, 0.77354)
41	lognormal(3.54728, 1.08717)	lognormal(6.54025, 0.78340)
42	lognormal(3.29541, 1.37510)	lognormal(8.21219, 0.77608)
43	lognormal(3.39453, 1.40227)	lognormal(7.34879, 0.78293)
44	lognormal(3.46886, 1.64119)	lognormal(7.41095, 1.01847)
45	lognormal(3.17724, 1.49191)	lognormal(7.38181, 0.90585)
46	lognormal(3.19299, 1.62570)	lognormal(7.87372, 0.88291)
47	lognormal(3.16184, 1.68669)	lognormal(6.56225, 1.88028)
48	lognormal(2.96198, 3.60630)	lognormal(7.01838, 1.51299)
49	lognormal(3.07988, 2.04665)	lognormal(8.06907, 0.83717)
50	lognormal(3.43479, 1.06621)	lognormal(8.38184, 0.72513)
51	lognormal(3.45591, 1.40464)	lognormal(8.42901, 0.75888)
52	lognormal(3.36315, 1.53036)	lognormal(8.46941, 0.69451)
53	lognormal(3.39605, 1.32438)	lognormal(8.87283, 0.69929)
54	lognormal(2.78959, 2.63920)	lognormal(8.81280, 0.72288)
55	lognormal(3.30587, 1.49725)	lognormal(8.93627, 0.73196)
56	lognormal(3.39924, 1.42454)	lognormal(8.87872, 0.73821)
57	lognormal(3.49349, 0.93749)	lognormal(9.70910, 0.65098)
58	lognormal(3.10619, 1.42653)	lognormal(8.00918, 1.76470)
59	lognormal(2.46151, 2.39870)	lognormal(8.94855, 0.96320)

Number	Error Bursts	Errorfree Bursts
60	lognormal(3.18758, 0.97900)	lognormal(9.46457, 0.78600)
61	lognormal(3.21114, 1.22932)	lognormal(9.01995, 0.89036)
62	lognormal(2.10647, 3.37739)	lognormal(9.70511, 1.21890)
63	lognormal(3.08199, 0.98016)	lognormal(10.24014, 0.78827)
64	lognormal(2.99311, 1.45910)	lognormal(9.96454, 0.81162)
65	lognormal(2.65025, 2.20811)	lognormal(9.35988, 1.19068)
66	lognormal(3.16894, 1.05783)	lognormal(10.49170, 0.73943)
67	lognormal(1.80249, 4.24548)	lognormal(10.05763, 0.84533)
68	lognormal(3.19124, 1.17976)	lognormal(10.43945, 1.06806)
69	lognormal(1.21749, 5.26428)	lognormal(11.01239, 0.74585)
70	lognormal(3.23215, 1.08604)	lognormal(10.07269, 0.94032)
71	lognormal(3.34491, 1.04157)	lognormal(10.05017, 0.89580)
72	lognormal(4.38934, 0.34961)	lognormal(9.00802, 3.48119)
73	lognormal(nan, nan)	lognormal(16.66501, 0)
74	lognormal(nan, nan)	lognormal(16.66501, 0)
75	lognormal(nan, nan)	lognormal(16.66501, 0)
76	lognormal(4.54255, 0.23231)	lognormal(14.40750, 1.29608)
77	lognormal(nan, nan)	lognormal(17.76362, 0)
78	lognormal(4.61955, 0.22049)	lognormal(13.23420, 2.78774)
79	lognormal(nan, nan)	lognormal(17.76362, 0)
80	binomial(22,1)	lognormal(16.78368, 0.57308)
81	lognormal(4.94824, 0.13553)	lognormal(13.31993, 2.70517)
82	lognormal(3.30654, 0.25489)	lognormal(15.21894, 1.57809)
83	lognormal(nan, nan)	lognormal(18.20545, 0)
84	binomial(16,0.96875)	lognormal(16.75521, 0.70326)
85	lognormal(nan, nan)	lognormal(18.20545, 0)
86	lognormal(3.29941, 0.22125)	lognormal(13.53873, 2.46405)
87	lognormal(3.00100, 0.21610)	lognormal(16.48727, 1.23903)
88	binomial(16,1)	lognormal(18.00997, 0.39095)
89	lognormal(2.56862, 0.44112)	lognormal(13.49810, 2.70214)

Table C.7: Distributions and Parameters for Semimarkovian Models for $k_0 = 50$

Number	Error Bursts	Errorfree Bursts
1	lognormal(2.58054, 0.95584)	lognormal(6.99517, 2.94718)
2	lognormal(2.52926, 0.79191)	lognormal(5.54198, 4.91487)
3	lognormal(2.98128, 1.30191)	lognormal(5.52303, 4.92452)
4	lognormal(2.87946, 0.87281)	lognormal(9.11534, 3.34252)
5	lognormal(2.82909, 0.44368)	lognormal(7.14259, 3.47596)
6	lognormal(2.79922, 0.98665)	lognormal(10.92921, 4.01036)
7	lognormal(2.68156, 0.69018)	lognormal(8.83970, 4.18490)
8	lognormal(2.62793, 0.47174)	lognormal(7.87499, 4.49959)
9	lognormal(2.74814, 0.49609)	lognormal(8.26886, 4.98285)
10	lognormal(4.74067, 0.14744)	lognormal(12.12517, 1.83402)
11	lognormal(2.99338, 0.15207)	lognormal(15.01573, 0.59859)

Number	Error Bursts	Errorfree Bursts
12	lognormal(2.74205, 0.13654)	lognormal(15.31231, 0.50795)
13	binomial(41,0.38582)	lognormal(6.47220, 6.53843)
14	lognormal(2.71207, 0.33934)	lognormal(9.61878, 4.50058)
15	lognormal(2.85792, 0.43171)	lognormal(9.62538, 3.62761)
16	lognormal(2.78990, 0.39200)	lognormal(10.42005, 4.05331)
17	binomial(16,0.975)	lognormal(16.15326, 1.90714)
18	lognormal(2.56144, 0.30314)	lognormal(10.78903, 4.72885)
19	lognormal(3.24703, 0.21739)	lognormal(15.49773, 1.67126)
20	lognormal(2.96924, 0.29820)	lognormal(12.86559, 2.64486)
21	lognormal(6.52270, 2.01069)	lognormal(12.39430, 3.71732)
22	lognormal(9.68707, 0)	lognormal(18.69928, 0.39704)
23	binomial(123,0.12482)	lognormal(10.59981, 5.83203)
24	lognormal(1.02009, 4.51496)	lognormal(6.62686, 5.99773)
25	lognormal(1.12938, 5.31730)	lognormal(6.17298, 7.31237)
26	lognormal(1.98817, 5.43561)	lognormal(9.04567, 7.06336)
27	binomial(570,0.02551)	lognormal(10.16087, 5.22272)
28	lognormal(2.71257, 0.08428)	lognormal(7.36874, 6.47442)
29	lognormal(1.15078, 5.93710)	lognormal(12.70535, 1.98373)
30	lognormal(0.87587, 6.02251)	lognormal(8.25181, 5.23385)
31	lognormal(2.72707, 0.06850)	lognormal(7.75886, 5.94951)
32	lognormal(2.66809, 0.10910)	lognormal(13.90589, 3.00056)
33	binomial(57,0.29043)	lognormal(12.39663, 4.45724)
34	lognormal(2.84503, 0.61919)	lognormal(16.40382, 1.89642)
35	lognormal(2.68282, 0.45783)	lognormal(16.97846, 1.52288)
36	lognormal(4.42532, 0.24461)	lognormal(9.63912, 1.93875)
37	lognormal(4.10143, 0.92452)	lognormal(6.36573, 0.51843)
38	lognormal(4.11319, 0.95535)	lognormal(6.75220, 0.57640)
39	lognormal(4.14892, 1.15148)	lognormal(6.87585, 0.59590)
40	lognormal(3.93552, 1.05396)	lognormal(6.80349, 0.58035)
41	lognormal(4.00705, 0.97042)	lognormal(6.82236, 0.60355)
42	lognormal(3.61782, 1.31779)	lognormal(8.41811, 0.63990)
43	lognormal(3.87532, 1.24803)	lognormal(7.64675, 0.58804)
44	lognormal(3.97428, 1.41267)	lognormal(7.72763, 0.81068)
45	lognormal(3.63865, 1.40922)	lognormal(7.66617, 0.71959)
46	lognormal(3.59226, 1.52731)	lognormal(8.12077, 0.71999)
47	lognormal(3.64658, 1.72461)	lognormal(6.92101, 1.64690)
48	lognormal(3.55536, 3.47507)	lognormal(7.54248, 1.16953)
49	lognormal(3.50555, 1.81669)	lognormal(8.29871, 0.68546)
50	lognormal(3.78453, 0.97882)	lognormal(8.59544, 0.58370)
51	lognormal(3.86217, 1.30273)	lognormal(8.70492, 0.57611)
52	lognormal(3.89625, 1.29874)	lognormal(8.79267, 0.48035)
53	lognormal(3.81759, 1.20310)	lognormal(9.14140, 0.52099)
54	lognormal(3.35386, 2.22893)	lognormal(9.08634, 0.54133)
55	lognormal(3.73721, 1.35462)	lognormal(9.20411, 0.55408)
56	lognormal(3.83489, 1.29858)	lognormal(9.16299, 0.54947)
57	lognormal(3.89074, 0.81231)	lognormal(9.94913, 0.49125)
58	lognormal(3.59820, 1.44233)	lognormal(8.34468, 1.54235)
59	lognormal(3.06617, 1.99616)	lognormal(9.19012, 0.80272)

Number	Error Bursts	Errorfree Bursts
60	lognormal(3.62753, 0.92946)	lognormal(9.72380, 0.61355)
61	lognormal(3.72560, 1.15942)	lognormal(9.34645, 0.67340)
62	lognormal(2.75628, 2.90953)	lognormal(9.97541, 1.03897)
63	lognormal(3.40846, 1.06346)	lognormal(10.44755, 0.65014)
64	lognormal(3.42938, 1.34238)	lognormal(10.19728, 0.65668)
65	lognormal(3.09797, 2.20969)	lognormal(9.64683, 0.99977)
66	lognormal(3.47491, 1.04866)	lognormal(10.67353, 0.61830)
67	lognormal(2.50847, 3.64772)	lognormal(10.34068, 0.65686)
68	lognormal(3.60862, 1.09596)	lognormal(10.68030, 0.90762)
69	lognormal(1.69348, 4.84903)	lognormal(11.18649, 0.62984)
70	lognormal(3.65702, 1.07482)	lognormal(10.34252, 0.76064)
71	lognormal(3.80717, 0.97558)	lognormal(10.34184, 0.70158)
72	lognormal(4.61738, 0.24360)	lognormal(9.17014, 3.37340)
73	lognormal(nan, nan)	lognormal(16.66501, 0)
74	lognormal(nan, nan)	lognormal(16.66501, 0)
75	lognormal(nan, nan)	lognormal(16.66501, 0)
76	lognormal(4.54255, 0.23231)	lognormal(14.40750, 1.29608)
77	lognormal(nan, nan)	lognormal(17.76362, 0)
78	lognormal(4.82325, 0.10693)	lognormal(13.36876, 2.70056)
79	lognormal(nan, nan)	lognormal(17.76362, 0)
80	binomial(22,1)	lognormal(16.78368, 0.57308)
81	lognormal(5.03799, 0.08931)	lognormal(13.38873, 2.66061)
82	lognormal(3.30654, 0.25489)	lognormal(15.21894, 1.57809)
83	lognormal(nan, nan)	lognormal(18.20545, 0)
84	binomial(16,0.96875)	lognormal(16.75521, 0.70326)
85	lognormal(nan, nan)	lognormal(18.20545, 0)
86	lognormal(3.29941, 0.22125)	lognormal(13.53873, 2.46405)
87	lognormal(3.00100, 0.21610)	lognormal(16.48727, 1.23903)
88	binomial(16,1)	lognormal(18.00997, 0.39095)
89	lognormal(2.56862, 0.44112)	lognormal(13.49810, 2.70214)

Table C.8: Distributions and Parameters for Semimarkovian Models for $k_0 = 100$