Performance of Cooperative Relay Protocols over
an Audio Channel

Examensarbete utfört i Kommunikationsystem
vid Tekniska högskolan i Linköping
av

Thomas Wärme

LiTH-ISY-EX-09/4267-SE
Linköping 2009
Performance of Cooperative Relay Protocols over an Audio Channel

Examensarbete utfört i Kommunikationssytem vid Tekniska högskolan i Linköping
av

Thomas Wärme

LiTH-ISY-EX--09/4267--SE

Handledare: Erik G. Larsson
Communication Systems, ISY, LiU

Examinator: Danyo Danev
Communication Systems, ISY, LiU

Linköping, 26 May, 2009
In wireless transmissions the communication is often degraded by random fades, noise and other performance reducing phenomena. One way of improving the stability and reducing the error rates is to use relaying techniques where several nodes cooperate in a transmission between two of them.

This thesis analyzes some of the available Decode-and-Forward relaying schemes for wireless transmission. The investigated schemes are conventional repetition coding, partial repetition coding and non-collaborative direct transmission. I have developed a three-node communication system using an audio channel to test the performance of repetition coding and direct transmission. This audio communication system can also be used to demonstrate some basic phenomena in wireless transmissions and how different scenarios change the performance of the communication. A theoretical performance analysis and computer simulations of the schemes performance over a Rayleigh fading channel are done as a basis for comparison. As a result we see that in the audio communication system repetition coding actually degrades the performance, compared to direct transmission, when using a relatively slow data rate in comparison to the speed of the fading in the audio channel.
Abstract

In wireless transmissions the communication is often degraded by random fades, noise and other performance reducing phenomena. One way of improving the stability and reducing the error rates is to use relaying techniques where several nodes cooperate in a transmission between two of them.

This thesis analyzes some of the available Decode-and-Forward relaying schemes for wireless transmission. The investigated schemes are conventional repetition coding, partial repetition coding and non-collaborative direct transmission. I have developed a three-node communication system using an audio channel to test the performance of repetition coding and direct transmission. This audio communication system can also be used to demonstrate some basic phenomena in wireless transmissions and how different scenarios change the performance of the communication. A theoretical performance analysis and computer simulations of the schemes performance over a Rayleigh fading channel are done as a basis for comparison. As a result we see that in the audio communication system repetition coding actually degrades the performance, compared to direct transmission, when using a relatively slow data rate in comparison to the speed of the fading in the audio channel.
Acknowledgments

I would like to thank my supervisor Erik G. Larsson, my examiner Danyo Danev, Mikael Olofsson and the others at Communication Systems, LiU, for all the help and input. I want to thank my opponent Fredrik Andersson and also my better half Joksie Biesemans for proofreading.
Contents

1 Introduction ................................................................. 1
   1.1 Background .......................................................... 1
   1.2 Purpose of the Project .............................................. 2
   1.3 Delimitations ......................................................... 2
   1.4 Thesis Layout ........................................................ 3

2 Relays in Wireless Networks ............................................. 5
   2.1 Potential Applications of Relaying ................................. 6
   2.2 Cooperative Diversity .............................................. 7
   2.3 Combination Methods ............................................... 7
   2.4 Performance Measures ............................................. 8

3 Transmission Schemes for DF Relaying ................................. 11
   3.1 Transmission Protocol ............................................. 11
   3.2 Channel Model ...................................................... 12
   3.3 Direct Transmission ............................................... 13
   3.4 Repetition Coding ............................................... 14
   3.5 Partial Repetition Coding ....................................... 15

4 Computer Simulations ................................................... 17
   4.1 How the Simulations were performed .............................. 18
   4.2 Simulation Results ................................................. 18

5 Audio Communication System ........................................... 21
   5.1 System Requirements .............................................. 21
   5.2 System Design ...................................................... 22
     5.2.1 The Data Acquisition Toolbox ................................. 22
     5.2.2 Coding and Modulation ....................................... 24
     5.2.3 Design Choices ............................................... 24
     5.2.4 Time Synchronization ......................................... 26
   5.3 The Audio Channel ................................................. 26
### Contents

5.4 Testbench Setup .................................. 28  
5.5 Test Results .................................... 29

6 Analysis and Conclusions 33  
6.1 Comparisons .................................... 33  
6.2 Analysis ......................................... 33  
6.3 Conclusions ..................................... 34  
6.4 Future Work ..................................... 35

Bibliography 37

A Matlab code - Source 39  
A.1 source.m .......................................... 39  
A.2 DLsource.m ....................................... 40  
A.3 RCsourse.m ...................................... 41  
A.4 PRsource.m ....................................... 42  
A.5 bfskmod.m ........................................ 44  
A.6 code-hamming.m .................................. 44

B Matlab Code - Relay 46  
B.1 relay.m ............................................. 46  
B.2 RCrelay.m ........................................ 47  
B.3 PRrelay.m ......................................... 49  
B.4 bfskdemod.m ...................................... 52  
B.5 decode-hamming.m ................................ 52  
B.6 errorcheck.m ..................................... 54

C Matlab Code - Destination 56  
C.1 destination.m ..................................... 56  
C.2 DLdestination.m .................................. 57  
C.3 RCdestination.m .................................. 59  
C.4 PRdestination.m .................................. 61  
C.5 bfsk-softdemod ................................... 63  
C.6 bfsk-harddemod ................................... 64  
C.7 combine.m ........................................ 64  
C.8 synchronize.m .................................... 65  
C.9 truncate.m ........................................ 66  
C.10 truncate-start.m ................................ 67  
C.11 errorcalc.m ...................................... 68
Chapter 1

Introduction

1.1 Background

In wireless transmissions a large problem is the unavoidable fades introduced in the channel due to multiple path propagation of the signals. Traditionally, relays have been used to extend the range of wireless communication systems [7] but as the research into cooperative transmission has gained a lot of interest in the last few years it has proved to be a very interesting concept to also improve the performance as well as to save energy. This interest is justified as cooperative transmission schemes in many cases can increase the performance of a communication system, by increasing the diversity and thereby counteracting fading, without notably increasing the complexity. The basic concept is to make use of additional nodes in the network when transmitting from a source to a destination. These additional nodes that help to improve the performance of transmission between other nodes are called relays and I will start by presenting an example of usage.

Example 1.1

In a cellular network, with several mobile nodes in one area, a specific node, $A$, wants to transmit a message, $msg$, to a nearby base station, $B$. Other mobile units that are idle at the moment, meaning that they have time and power to spare without affecting their own interests, might overhear that $A$ is trying to transmit $msg$ to $B$. They can then act as relays and assist in the communication by forwarding $msg$ to $B$, who then receives several copies of $msg$ coming from different nodes in the network and the risk of $msg$ not being delivered correctly to $B$ is very low. If the link between $A$ and $B$ is in a bad state the message might otherwise not be delivered at
This could be done in an ad-hoc fashion so that $A$ later on could act as a relay to help and improve the communication between other nodes.

The theoretical parts of this thesis are largely based on the work of Majid Nasiri Khormuji and Erik G. Larsson [1] [4] in which the performance of several Decode-and-Forward, $DF$, relaying schemes in a Rayleigh fading channel are presented.

### 1.2 Purpose of the Project

The starting point of this thesis was to investigate the performance of relaying schemes and incorporate them into an audio communication system for demonstration purposes. The schemes to analyze were non-collaborative *direct transmission* and the cooperative schemes *repetition coding* and *partial repetition coding*. The scheme *partial repetition coding* was recently proposed in [1] and the performance of all these schemes in a Rayleigh fading channel has also been derived and shown in [1] in closed form as analytical expressions along with simulation results to support them. I was to construct a testbed environment, with three computers as nodes, using an audio-channel to get test results about these schemes under the influence of controlled disturbances. The software that I develop to operate this audio communication system should also be able to demonstrate some basic phenomena in wireless transmissions and to show the impact that different scenarios have on the performance of the communication. The goal has been to get results about the performance of the schemes and about cooperative transmission in general compared to direct transmission when using an audio-channel.

### 1.3 Delimitations

There are two main categories of relaying protocols, Amplify-and-Forward, $AF$, and Decode-and-Forward, $DF$. This thesis is limited to only investigating $DF$ relaying protocols and only setups using one single relay. I have limited the study to only investigating the schemes *direct transmission*, *repetition coding* and *partial repetition coding*. I will not investigate the effects of different hardware setups because one setup is enough for the purpose of comparing the communication schemes.
1.4 Thesis Layout

Chapter 2 processes the fundamentals of relaying, ways of combining signals at the receiver and the performance measures used for comparing the schemes. Chapter 3 explains the workings of the schemes and presents analytical expressions of the performance of the relay schemes over a Rayleigh fading channel, based on previously available results. Analytical expressions for optimizing resource allocation are also presented. Chapter 4 shows the assumptions that are made and the results of the performed computer simulations.

Chapter 5 introduces the audio communication system that I have developed, how the system was constructed, the design choices made and a test environment for comparing the implemented schemes. Chapter 6 summarizes and analyzes the results.
Chapter 2

Relays in Wireless Networks

It lies in the nature of wireless transmission that information is broadcasted and that anyone with the right equipment can listen in. This fact can be used to lower the error rates between a transmitting source and the intended destination by letting one or several extra nodes, relays, assist in the communication.

The wireless channel is often haunted by random variations in the strength of the received signal. The phenomenon is referred to as fading and it occurs when transmission is maintained through a number of stochastic reflections [3]. This continuously changes the quality of the channel and the reliability of the communication. By letting a signal pass through extra nodes, that listen to the transmitted signal and forward it to its intended destination, the signal takes several routes from the original source to the final destination. The source and the assisting relays usually share access to the same channel through time division by using separate time slots. The separate links between each of the nodes are influenced by individual distortion, independent of each other, and the improvement we get is that if the direct link from the source to the destination is in a bad state, the signal may still be delivered correctly via another node. When a signal takes multiple paths from its source to its destination this often translates into diversity gains [1] which are discussed in section 2.2. The nodes that assist in the transmission are called relays. From here on I assume that we are working with one single relay and a basic layout of the three-node relay setup can be seen in Fig. 2.1. For short we can call the source, relay and destination $S$, $R$ and $D$. When one single relay is used, $D$ receives two copies of the messages transmitted by $S$. One version that comes directly from $S$ and one version that has passed through $R$. 
Relaying protocols can be divided into the two main categories Amplify-and-Forward protocols, \( AF \), and Decode-and-Forward protocols, \( DF \). In \( AF \), the relay simply amplifies the received signals and re-transmits them. In \( DF \), the relay only re-transmits a received message if it has been able to successfully decode it first. The relay then has the option to re-encode messages in a different channel code than the original messages were coded in and error correction can be performed at the relay as well. Although \( DF \) relaying can be very efficient in some cases it is very much relies on the capacity of the link between \( S \) and \( R \). The relay only forwards a message if it has been perfectly decoded and a poor link between \( S \) and \( R \) therefore leaves \( R \) unused at times and direct transmission, not using the relay at all, could be a better option [5].

2.1 Potential Applications of Relaying

The concept of relaying can be incorporated into most wireless networks. Fixed relays could be implemented in cellular networks to assist mobile phones in their up and down links. This would bring the benefits of Multiple-Input Multiple-Output, \( MIMO \), resulting in diversity gains as a distributed \( MIMO \) system, without the practical constraint of having several antennas in the mobile unit. Conventional \( MIMO \) demands a lot of power and is therefore not well suited for mobile applications, where energy consumption is a big concern. Another advantage in relay-based systems, that make them suited for mobile applications, is their resistance to shadow fading, which occurs when large obstacles block the signal path. This is thanks to the fact that the placement of the relay is not restricted in space to being at the same place as the transmitter or receiver. When the trans-
mission path between two nodes is obstructed, the information could still flow through another path. Other potential applications could be wireless sensor networks and cognitive radio networks [1] [7].

2.2 Cooperative Diversity

A big advantage that comes with using cooperative relaying is the space diversity gain and this is the effect that helps to combat signal fading. In Fig. 2.2 we see a single-input-multiple-output SIMO-setup with two outputs and a multiple-input-single-output MISO-setup with two inputs. The SIMO setup is the way the source experiences the relay setup and the MISO setup is the way the destination experiences it. Each time slot the relay switches between being a receiver and a transmitter. This way we achieve space diversity in both transmission and reception which has come to be called cooperative diversity [8]. The advantage of the relay setup, Fig. 2.1, over conventional MISO/SIMO is that the relay is flexible in space and can move around. Also, different nodes in a network can take on the role of being a relay for any transmission which makes for a very flexible system. To summarize we can say that by enabling a cooperative relay to forward its received information, we exploit space diversity through cooperation among the distributed antennas belonging to the different nodes in the wireless network [6].

![Figure 2.2. SIMO and MISO setups as seen by the source and destination in the three node relay setup.](image)

2.3 Combination Methods

To make use of both messages that $D$ receives they have to be combined in some way when extracting the received data. There are different combination methods and they differ both in efficiency and complexity. The common basis of all combination algorithms is that they have to be able to sense the quality, the SNR, of the separate links $S - D$ and $R - D$ in order to make good decisions on what message or parts of messages that are most
likely not to have been corrupted during transmission. The most simple and straight-forward way is Selection Combining, SC, which only uses one of the received messages and ignores the other one. SC only chooses to use the message coming from the diversity branch that has the highest instantaneous SNR [2].

A superior method is Maximum Ratio Combining, MRC, which combines both received messages, each weighted with a gain factor proportional to the SNR of that link. To be optimal the gain factors should be chosen to maximize the SNR of the combined message [2]. If the relay uses a different channel code than the source the destination cannot simply merge the messages as they are but has to use code combining which is very computationally expensive. In the audio communication system that I have designed I always use the same channel code for the $S - D$ and $R - D$ links because then the combination process at the destination can be implemented with an MRC combiner.

2.4 Performance Measures

There are many ways of measuring the performance of a wireless link. One way is to calculate the bit-error-rate, BER, which denotes the number of bits of a received message that differ from the transmitted message divided by the total number of transmitted bits (i.e. the probability of a bit to be incorrectly received). I use the BER measurement when testing the audio communication system in chapter 5.

Another way of measuring performance is to look at the probability of the channel being in outage, $P_{out}$, which occurs during fades in the signal amplitude over the channel as seen in Fig. 2.3. For the computer simulations in chapter 4, the $P_{out}$ measurement is used to compare the different relaying schemes. Setting $\beta$ [bits per channel use] as the target spectral efficiency a link is said to be in outage when the instantaneously achievable spectral efficiency of that link is less than $\beta$. The outage probability shows how often the destination can successfully decode a packet transmitted with a fixed rate $\beta$ bits per channel uses [1].

The probability of outage is defined as

$$P_{out} = Pr\{\text{Outage}\} = Pr\{R < \beta\}$$

where $R$ is the achievable rate obtained by a given scheme [1].
Figure 2.3. Signal amplitude fluctuations at the receiver when transmitting over a Rayleigh Fading channel
Chapter 3

Transmission Schemes for DF Relaying

There are many transmission schemes for cooperative transmission described in the literature and I will analyze a few of them. A transmission scheme is a set of rules on how the communication is to be done. Rules on how to use and divide the available space, time and power between the nodes taking part in a transmission and also how channel codes should interact and other choices that form the basis for the transmission.

In this chapter the schemes direct transmission, repetition coding and partial repetition coding are explained. A Rayleigh fading channel model is presented and analytical expressions for the outage probabilities of the schemes for this model are presented on a closed form. Also expressions for optimizing the power and time allocation are presented. All of the analytical expressions of finite SNR performance and for optimization of resource allocation presented here are derived in [1] and I refer the reader who is interested in the calculations leading up to these expressions to this work. A plot of the $P_{out}$ for the relaying schemes based on the analytical expressions presented here can be seen in Fig. 4.1, as a comparison to the simulation results presented in Chapter 4, plotted on a logarithmic scale as functions of the SNR.

3.1 Transmission Protocol

There are two possible setups for the medium access, full-duplex and half-duplex. In a full-duplex setup the relay is able to transmit and receive simultaneously on the same frequency which in practice, if possible, is com-
plicated. I consider the simpler setup of a half-duplex relaying system in which the relay can not transmit and receive simultaneously, for both the computer simulations and the audio system tests. This is accomplished by using time division so that reception and transmission is done in non-overlapping time slots.

Assume, for all schemes, that the total energy per packet is

\[ E = PT = P_sT_s + P_rT_r \]

where \( P \) is the average transmit power and \( T \) is the total number of channel uses available per packet. We have a three-node channel where transmission takes place in two phases. In the first phase \( S \) transmits a packet using \( T_s \) channel uses and power \( P_s \) while both \( R \) and \( D \) listen. In the second phase, if the relay has successfully decoded the received packet, it re-encodes the packet and re-transmits it using \( T_r = T - T_s \) channel uses and power \( P_r \). If decoding at \( R \) fails it remains silent. The differences between the relaying schemes lie in the distribution of transmit power and channel uses between \( S \) and \( R \). By letting all available transmit power and channel uses be used by \( S \) any scheme reduces to direct transmission, leaving \( R \) unused.

### 3.2 Channel Model

Consider a time frame of length \( T \) which is divided into two parts, \( T_s = \delta T \) and \( T_r = (1 - \delta)T \). \( \delta \in [0.5, 1] \) is a coefficient that shows how much of the total available channel uses are used by the source. During the first time slot, \( T_s \), the source transmits a block of data \( s[n] \) while the relay and the destination listen to the transmitted signal. Then during the second time slot, \( T_r \), the relay transmits a processed form of the received data, \( s_r[n] \), after decoding and re-encoding it. In the \( n \)th time slot the received signals, \( y_{sr} \) and \( y_{sd} \), at the relay and at the destination can be described by the following channel model:

\[
y_{sr}[n] = h_{sr}s[n] + z_r[n] \tag{3.1}
\]
\[
y_{sd}[n] = h_{sd}s[n] + z_d[n] \tag{3.2}
\]

and in the \((n + 1)\)th time slot the relay transmits the signal \( s_r \) and the destination receives \( y_{rd} \) according to the model

\[
y_{rd}[n + 1] = h_{rd}s_r[n + 1] + z_d[n + 1], \tag{3.3}
\]

Here \( h_{sr}, h_{sd} \) and \( h_{rd} \) denote the channel gains of the links and they capture both the effects of path loss, shadowing and frequency non selective fading.
3.3 Direct Transmission

The added variables \(z_r[n], z_d[n]\) and \(z_d[n+1]\) denote the receiver noise and other forms of interference. They can be modeled as zero-mean, mutually independent Gaussian random variables with unit variances \(N_0 = 1\). The channel between the nodes are modeled as quasi-static Rayleigh fading, quasi-static meaning that the gain is constant during the transmission of one block. Let

\[
\alpha_{ij} = \frac{|h_{ij}|^2}{N_0}, \quad i \in \{s, r\}, j \in \{r, d\}
\]

where \(h_{ij}\) is the channel gain from node \(i\) to node \(j\). Then the received SNR for link \(i - j\) equals \(P_i\alpha_{ij}\) and it is exponentially distributed with mean \(P_i\gamma_{ij}\) where

\[
\gamma_{ij} = E|h_{ij}|^2
\]

3.3 Direct Transmission

In direct transmission, DT, the relay is not taking part in the communication, as can be seen in Fig. 3.1. Therefore all available power and time is used by the source and \(\delta = 1\) so that \(P_s = P, T_s = T, P_r = 0\) and \(T_r = 0\). Since DT only uses the direct link, \(S - D\), the scheme is in outage when that link is in outage and the probability of outage for DT over a Rayleigh fading channel is then given by:

\[
P_{out} = Pr\{\alpha_{sd} < \frac{2^3 - 1}{P}\} = 1 - \exp\left(\frac{1 - 2^3}{\gamma_{sd}P}\right) = \frac{2^3 - 1}{\gamma_{sd}P} + O\left(\frac{1}{P^2}\right) \quad (3.4)
\]

No diversity is achieved since there is only one transmitting and one receiving node.
3.4 Repetition Coding

In repetition coding, RC, all three nodes are taking part in the communication, as can be seen in Fig. 3.2. The total number of available channel uses are divided into two equal parts so that $\delta = 0.5 \Rightarrow T_s = T_r = \frac{T}{2}$. The transmit power however does not have to be equally distributed between $S$ and $R$ as long as it sums up to $P_s + P_r = 2P$ giving a total signal energy of $E = P_s T_s + P_r T_r = (P_s + P_r) \frac{T}{2} = PT$ which is the same as for DT. During $T_s$ the source transmits and the relay and destination listen. During $T_r$ the relay repeats the information, if has been successful in decoding it, and the destination listens. Since the channel uses are divided equally between the source and the relay it is possible for the relay to repeat all the data transmitted by the source. The destination can then use SC or MRC on whole messages to combine them. Only MRC is considered as the combination method here because it performs better than SC.

In the collaborative schemes, RC and PR, the messages travel two separate paths to the destination. For these schemes to be in outage the direct link, $S-D$, and at least one of the links that include the relay, $S-R$ or $R-D$, must be in outage at the same time. The probability of outage for RC in a Rayleigh fading channel is then given by:

$$P_{\text{out}} = (2^{2\beta} - 1)^2 \frac{1}{\gamma_{sd} P_s} \left[ \frac{1}{\gamma_{sr} P_s} + \frac{1}{2\gamma_{rd} P_r} \right] + O(\frac{1}{P^3}) \quad (3.5)$$

Repetition Coding provides a diversity of order two.

For the conventional repetition coding scheme the only design choices are $P_s$ and $P_r$ and from Eq. 3.5 we see that, by dropping the $O(\frac{1}{P^3})$, a good choice of $P_s$ and $P_r$, that minimizes the outage probability, is obtained by
3.5 Partial Repetition Coding

In partial repetition coding, PR, the available channel uses do not have to be equally divided between $S$ and $R$. Since $R$ uses the same channel code as $S$ it can not repeat all the data it receives if $T_r < T_s$. With $T_s = \delta T$ and $T_r = (1 - \delta)T$ the relay only retransmits a fraction $\frac{1-\delta}{\delta}$ of the data it receives and discards the rest, where $0.5 < \delta < 1$. The transmit power is shared by $S$ and $R$ under the constraint that $\delta P_s + (1 - \delta)P_r = P$ so that the total energy used is $E = P_s T_s + P_r T_r = \delta TP_s + (1 - \delta)TP_r = PT$ which is the same as for the other schemes. So for the partial repetition scheme $D$ receives full packets from $S$ and parts of the packets from $R$ and message combining has to be done separately for the common parts of the packets and the parts only received from the source. MRC can be used for the common parts of the packets and the parts only received from the source needs no processing since it has nothing to be combined with. As for RC the scheme is in outage when the link $S-D$ and at least one of the links $S-R$ or $R-D$ are in outage. The probability of outage in a Rayleigh fading channel is then given by:

$$P_{out} = \frac{(1-2^{\beta_s})^2}{\gamma_{sr}\gamma_{rd}P_s^2} + \frac{1-2^{\beta_s} - 0.5(1-2^{\beta_s})^2 + 1-\delta}{\gamma_{rd}\gamma_{sr}P_r P_s} + O\left(\frac{1}{P^3}\right), \quad \delta \neq \frac{2}{3}$$

$$P_{out} = \frac{(1-2^{1.5\beta_s})^2}{\gamma_{sr}\gamma_{rd}P_s^2} + \frac{1-2^{1.5\beta_s} - 0.5(1-2^{1.5\beta_s})^2 + 1.5 \ln(2)2^{3\beta_s}}{\gamma_{rd}\gamma_{sr}P_r P_s} + O\left(\frac{1}{P^3}\right), \quad \delta = \frac{2}{3}$$

Figure 3.3. Network Schematic for Partial Repetition Coding.
where $\beta_s = \frac{\beta}{3}$ and $\beta_r = \frac{\beta}{1-\delta}$. Partial Repetition Coding also provides a diversity of order two. When optimizing the resource allocation for PR $\delta$ is also a variable and we get an optimization problem with three parameters. From Eq. 3.7 we see that a good choice of $P_s, P_r$ and $\delta$, that minimize the outage probability, is obtained by minimizing

$$J(P_s, P_r, \delta) = \frac{(1 - 2^{\beta_s})^2}{\gamma_{sr} P_s^2} + \frac{1 - 2^{\beta_s} - 0.5(1 - 2^{\beta_s})^2 + \frac{1-\delta}{2-3\delta}(2^{2\beta_s} - 2^{\beta_r})}{\gamma_{rd} P_r P_s}$$

under the limitations that $0.5 < \delta < 1$, $0 \leq P_s \leq \frac{P}{\gamma_s}$, $0 \leq P_r \leq \frac{P}{1-\delta}$ and $\delta P_s + (1-\delta)P_r = P$. 


Chapter 4

Computer Simulations

I have done a performance analysis of the relaying schemes by running Monte Carlo simulations in matlab. A Monte Carlo simulation, MC simulation, is used to evaluate expressions that include stochastic variables. Evaluating such an expression will give a different result every time because of its stochastic properties and an MC-simulation evaluates this expression several times to calculate a mean value. This estimated mean value is increasingly reliable as the number of evaluations increase. When simulating communication schemes it is the channel properties that vary in time and the strength of the received signal can be modeled with a certain distribution. In these simulations it is assumed that the channel is quasi-static Rayleigh fading and the signal strength at the receiver is then exponentially distributed. Assume a log-distance path loss model for the channel

\[ \gamma_{ij} = \frac{1}{d_{ij}^\alpha} \]

where \( \alpha \) is the path loss exponent and \( d_{ij} \) is the normalized distance from node \( i \) to node \( j \). The choice of path loss exponent, \( \alpha \), depends on the desired path loss model but is typically chosen as \( 2 < \alpha < 4 \). I choose \( \alpha = 4 \) which represents a dense urban area with some obstacles that degrade the communication [1]. We can see from Eq. 3.6 and Eq. 3.8 that when assuming the same channel distribution between all nodes the normalized distances between \( S \), \( R \) and \( D \) control the optimal choice of energy distribution since \( \gamma_{ij} \) is determined by \( d_{ij} \).

From section 3.2 we have that the received SNR for link \( i \to j \) is exponentially distributed with the mean \( P_i \gamma_{ij} = \frac{P_i}{d_{ij}^\alpha} \). For this simulation the normalized distances between the nodes \( S \), \( R \) and \( D \) are chosen as \( d_{sd} = d_{sr} = 1 \) and \( d_{rd} = 0.1 \) so that the relay is quite close to the destination. An optimized time distribution for the PR scheme is chosen as
\( \delta = 0.83 \) meaning that 83\% of the available channel uses are used by the source and only 17\% of each transmitted block is repeated by the relay. The choice of \( \delta = 0.83 \) for this setup was found through trial-and-error tests for varying values of \( \delta \). The average transmit power of \( S \) and \( R \) are for simplicity chosen as \( P_s = P_r \).

### 4.1 How the Simulations were performed

The target spectral efficiency is chosen to be \( \beta = 3 \) bits per channel uses for this simulation. I run MC-simulations until 200 outages has occurred per SNR point for SNR ranging from 0 to 40 dB. The simulations are based on the \( P_{\text{out}} \) according to equations in chapter 3 and those expressions were derived under the assumption of MRC being used as the combination method at \( D \).

### 4.2 Simulation Results

In Fig. 4.1 the results of the MC-simulation can be seen as the dashed curves, plotted as functions of the SNR. They are shown in the high-SNR regime because that is where the best results can be achieved when analyzing relaying protocols. That is because in the high-SNR regime the data rates are mainly limited by interference as opposed to in the low-SNR regime where the influence of channel noise prevails, and hence the gain from cooperation is reduced [5]. Also in Fig. 4.1, seen as solid lines, are the asymptotes of the theoretical curves for the schemes in the same setup, as a comparison. We can see that the cooperative schemes achieve full diversity, i.e., the outage probability decays proportional to \( 1/SNR^2 \) whereas it decays as \( 1/SNR \) for the DL scheme [8].
4.2 Simulation Results

Figure 4.1. Theoretical and MC-simulated outage probabilities for the schemes direct transmission, repetition coding and partial repetition coding. The solid lines are the asymptotes of the analytical expressions without the $O(1/P)$-term. The dashed lines are the simulation results. Plot made with $d_{sd} = d_{sr} = 1$, $d_{rd} = 0.1$, $\beta = 3$, $\alpha = 4$ and $\delta = 0.83$. 
Chapter 5

Audio Communication System

To test the performance of the transmission schemes over an audio channel I have developed programs to be used by three computers acting as source, relay and destination in a three-node audio communication setup. With these programs you can transmit data between computers using different transmission schemes and then measure the performance of the schemes by calculating the BER after transmissions. The system was developed in Matlab and has been embedded into a Matlab GUI for easy usability without knowledge about the system design details. All code is included in Appendix A-C. The system is meant to be used as an intuitive demonstration of the principles of cooperative transmission and to show some basic concepts and phenomena that are typical for wireless communication.

5.1 System Requirements

These programs work on any OS running Matlab version 7.0 or later. Matlab has to have the Data Acquisition Toolbox version 2.7 or later in order to control the computers sound card. The hardware needed is a microphone for the source computer, a pair of speakers for the destination computer and, for running the cooperative schemes, a relay computer with a microphone and a pair of speakers. The system has proven to work fine on computers with a 1.7 GHz processor and 512 MB RAM.
5.2 System Design

The programs, running on all three computers, control and monitor the whole transmission process. When starting a transmission the user chooses between different transmission schemes, what channel code to use and what data to transmit. Fig. 5.1 shows the GUI of the source program and the GUIs for the relay and destination programs look similar. After a transmission is finished the relay and destination programs will display the length of the received data, how many errors, if any, that occurred during transmission for the separate links \( S - R, S - D, R - D \) and the BER for the combined final estimate. When starting a transmission it is assumed that all three computers agree beforehand on what transmission scheme and channel code to use and the source and relay programs need to know what data is being transmitted to have something to compare the received data with in the error calculations.

![Matlab GUI for the source computer in the audio communication system.](image)

**Figure 5.1.** Matlab GUI for the source computer in the audio communication system.

5.2.1 The Data Acquisition Toolbox

The data acquisition toolbox, DAQ toolbox, in matlab helps to interact with hardware in the computer and is a very useful tool when working with input to and output from a computer. With the DAQ toolbox we can create analog input objects and analog output objects and configure them to be connected to inputs and outputs of a computers sound card. This
way signals can be output from matlab to the speakers and the input from a microphone can be recorded into matlab in real-time. A block schematic of a direct transmission between two computers can be seen in Fig. 5.2. The source program encodes a vector of binary data, that it has been told to transmit, according to the users choice of coding and then modulates the data into a signal that is stored in the stack of an analog output object. When this object is activated the contents of its stack flows to the output of the sound card, at a predetermined rate, and the speakers play the signal stored on the stack. The relay program has both an analog input object connected to the sound cards input and an analog output object connected to the sound cards output. It switches between the two and uses the input object to record a signal in the time slots where the source program is transmitting and then uses the output object to transmit in the following time slot. After recording a signal in the first time slot the relay program demodulates and decodes the signal to retrieve the original binary data of that block. If decoding is successful it re-encodes the data, modulates it and then puts it on the stack of the analog output object to be transmitted in the next time slot. The destination program only needs an analog input object to record the whole received transmission, coming from the sound card. After a transmission is completed the destination program has a recording that contains the signals transmitted by both the source and the relay. The original vector of binary data can then be retrieved by first combining the signals or by looking at them separately and then demodulate and decode them.

**Figure 5.2.** Block diagram of the system for direct transmission from the source to the destination.
5.2.2 Coding and Modulation

The modulation method used is non-coherent binary frequency shift keying, BFSK, using two distinct carrier frequencies to denote binary 1 and 0. The carrier frequencies used are $f_{c1} = 1100\, Hz$ and $f_{c2} = 1300\, Hz$, which are reasonable frequencies for audio transmission and lie in a frequency region where the speakers and microphones work well. With BFSK modulation each symbol represents a single bit of data and I use 200 samples of a cosine, of frequency $f_{c1}$ or $f_{c2}$, to represent each symbol in the modulated signal. Using a non-coherent modulation technique simplifies the receiver because knowledge about the phase of the signal is not required in the demodulation process. Due to shifting of the phase during transmission coherent detection can become troublesome. Demodulation is done using a simple fast fourier transform, FFT, which is an approximation of the discrete fourier transform, DFT. The FFT algorithm is used separately on parts of a received signal to extract the frequency information of the symbols in those parts. The demodulation process looks at those parts for frequency peaks at $f_{c1}$ and $f_{c2}$ to determine what data was received.

The system is designed with the option of using an error correcting block code. Block codes split the data, that is to be transmitted, into blocks of $k$ symbols and add check symbols to each block to get corresponding codewords of length $n > k$. These check symbols can then be used in the decoding of the blocks at the receiver to correct errors that may have occurred during transmission [2]. I have chosen to use a hamming $(7, 4)$ block code, that is $k = 4$ and $n = 7$, because of its simple implementation into the system and because the blocks in each time slot consist of seven bits. When using this hamming code one error per four transmitted bits can be corrected at the cost of having to transmit $\frac{7}{4} = 1.75$ times as many symbols than when transmitting uncoded data.

5.2.3 Design Choices

For the cooperative schemes, when the relay is taking part in the communication, time is divided into time slots giving the source and the relay access to the channel every other time slot. The data that is to be transmitted is therefore divided into blocks, where one block is transmitted by the source in one time slot and the relay forwards that block in the following time slot and the process is repeated until all the data is transmitted. I have chosen to use a block length of seven symbols per block for my tests but this value can easily be varied. Transmission is done at the rate of 8000 samples per second and recording is done at the same sample frequency.
Using 200 samples of a cosine to represent each symbol results in a bit rate of 40 bits/s for DL transmission. For the cooperative schemes the bit rate is naturally cut in half because every block of data is transmitted twice, both by $S$ and $R$ in non-overlapping timeslots. But because of guard bands inserted between the transmission time slots of $S$ and $R$ for the cooperative schemes, see Fig. 5.3, the bit rate actually drops to about 13 bits/s. These guard bands are required to give the relay time to process its received signals before its time slot for re-transmission occurs. These guard bands also give $D$ a clear distinction between the transmissions from $S$ and $R$ and give room for good reception even if the synchronization is a bit off. When using hamming coding these bit rates drop to about 22.9 bits/s for DL transmission and 7.4 bits/s for the cooperative schemes.

<table>
<thead>
<tr>
<th>Coding</th>
<th>Scheme</th>
<th>Data rate [bits/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncoded</td>
<td>DL</td>
<td>40</td>
</tr>
<tr>
<td>Uncoded</td>
<td>Coop</td>
<td>13</td>
</tr>
<tr>
<td>Hamming</td>
<td>DL</td>
<td>22.9</td>
</tr>
<tr>
<td>Hamming</td>
<td>Coop</td>
<td>7.4</td>
</tr>
</tbody>
</table>

Table 5.1. Data rates for transmission in the audio communication system. Coop denotes both the RC and PR schemes.

At the destination MRC is used to combine the messages. First the received signal is demodulated into a matrix of soft demodulated symbols containing the frequency information of the signals from both $S$ and $R$ and then these values are combined with MRC before the hard-decision demodulation, that results in a final binary estimate of the received signal, is done. For MRC
to work well we have to have knowledge about the SNR of the separate links at all times in order to choose good gain factors and make use of the variation in link quality. This estimation process of the SNR of the separate links needs more work and does not work optimally.

5.2.4 Time Synchronization

It is very important for all the nodes to be synchronized in time so that demodulation can be done as reliably possible. I have chosen to synchronize the nodes with a pilot sequence that is transmitted before the start of any real data transmissions. In long transmissions the pilot sequence can be repeated during transmission to give the receiver an updated reference in time. The pilot sequence is a block of symbols that is known by all nodes and it is used to identify the start of a transmission and the same block is used as a trailer to determine the end of a transmission. The relay and the destination also use the pilot sequence to estimate the channels attenuation of signals to give a reference level for the demodulation of the actual data transmission.

Intersymbol interference is a phenomenon that occurs when the delay spread of a channel is larger than the symbol duration and each transmitted symbol will experience interference from neighbouring transmitted symbols [2]. In the audio communication system all symbols are affected by the previously transmitted symbols. At reception the symbols appear to have been spread in time during transmission and this forces the demodulator to sometimes make dubious decisions about which symbol was received. In Fig. 5.4 we can see this effect and to counter this issue I have set the system to separate all symbols, within each block, in time by placing guard bands between the symbols before transmission. I have chosen the guard bands to have a length of 40% of the symbol length which seems to protect against most of the intersymbol interference. The guard bands also give extra safety against the time synchronization being or becoming a little bit off.

5.3 The Audio Channel

In my setup of this transmission system the channel can be seen as a whole series of components stretching from when the data is being sent from matlab to the transmitting computers sound card to when the data is delivered to matlab from the receiving computers sound card. Along the way the signal is processed and affected by the computers sound-cards with built-in amplification, the speakers with their own characteristics affecting the
produced audio signal, the effects of the actual channel and the properties of the recording microphone. The signal recorded by the microphone is a distorted version of the signal transmitted from the speakers, affected by multiple path propagation, echoes, and random background noise as well as more controllable audio disturbances such as people talking in the background. The audio transmissions multiple path propagation yields an impulse response that is a sum of delayed attenuated impulses depending on the acoustic properties of the room that the transmission is being performed in. We can assume the channel to be linear within normal signal strengths and also to be time invariant, as long as the environment is not physically changing during transmission. A DC-component and other low frequency distortions appear in the received signals but do not degrade the performance since the communication is done in the $1100 \ldots 1300\, Hz$ region. The signals produced by the speakers have a time delay spread that is larger than the symbol duration because of the physical limitations of the speakers themselves. It is impossible for the speakers to go from playing to being completely quiet instantaneously. Combined with the effects of multiple path propagation we get a quite large time delay spread for the channel which can lead to intersymbol interference. Fig. 5.4 shows a received signal from when two consecutive symbols were transmitted over the audio channel in my test setup, with the transmission bands and guard bands marked. We can clearly see the channels delay spread and why the introduction of guard bands is necessary.
5.4 Testbench Setup

The basic setup of the system tests, see Fig. 5.5, were three computers running the developed programs in a systematic way to test the schemes for varying SNR. Each pair of transmitting speakers and receiving microphone were at a distance of one meter. I used different models of speakers for the source and relay computers which made for some volume adjustments to give them the same transmit power. This was done by measuring the recorded signal strength at the destination for both kinds of speakers and adjusting the volume on one of the pairs. The hardware used was normal low-priced desktop speakers and desktop microphones. The source and relay computers were two stationary AMD 1.8 Ghz computers with 1 GB RAM and the destination computer was a 1.7 Ghz, 512 MB RAM laptop. The stationary computers were running Matlab version 7.4.0.287 R2007a with data acquisition toolbox 2.10 and the laptop used Matlab version 7.1.0.246 R14 service pack 3 with data acquisition toolbox 2.7.

My first approach to estimate the performance of the system was to transmit a fairly large amount of data repeatedly while for every transmission lowering the transmit power at both the source and the relay to gradually raise the SNR. The destination program then calculated the BER for each step. A test like that incorporated errors introduced by faulty synchroniza-
tion into the results and when varying the transmit signal power over a large interval the receivers sometimes had problems adapting to the large differences in signal energy between the first and the last transmission. This proved to give very erratic results and another test method was needed. The next idea, that proved to give more stable results, was to first transmit a large chunk of data through the relay system and then save the whole recorded signal at the destination, which included the signals from both the source and the relay, and then varying the SNR of that received signal before demodulating and decoding. Truncating this received signal to make sure that the synchronization was satisfactory I could then save that synchronized recorded signal to run several tests on without worrying about synchronization errors. To vary the SNR I added white Gaussian noise to the signal varying the power of the noise to change the SNR step by step. This way of testing saves time as the actual audio transmission is not necessary for each SNR value and that made for much faster tests. In this test setup it is possible to vary the SNR separately for the $S - D$ and $R - D$ links, to test different scenarios, but the properties of the $S - R$ link can not be altered once the initial transmission is recorded at $D$. This method of testing was used to achieve the test results for the DL and RC schemes that are presented in Fig. 5.6 and Fig. 5.7. I used BER as the performance measure and for the RC scheme I calculated the BER for both the $S - D$ and $R - D$ links as well as for the MRC combined final estimate at $D$.

5.5 Test Results

In Fig. 5.6 we can see the curves of the BER for the DL scheme, the RC scheme and for the separate links $S - D$ and $R - D$ in the RC scheme for an uncoded transmission as a linear plot against the SNR. No scale is presented on the x-axis because of problems with estimating the SNR of the signals but the difference between the schemes is apparent. Fig. 5.7 shows BER curves for the DL and RC schemes both for uncoded and for hamming coded transmission with the y-axis in a logarithmic scale.
Figure 5.6. BER as a function of the SNR for uncoded transmission with Direct Link and Repetition Coding schemes and for the separate links, $S \rightarrow D$ and $R \rightarrow D$, in the RC scheme.
Figure 5.7. BER as a function of the SNR for uncoded and hamming coded transmission with Repetition Coding and Direct Link schemes, plotted in a logarithmic scale.
Chapter 6

Analysis and Conclusions

6.1 Comparisons

It is not really possible to compare the results from the MC-simulations in chapter 4 with the results from the audio system tests in chapter 5 because the assumptions and the test setups are very different. Comparisons can however be made between the schemes for each of the tests separately. The MC simulations show results that align very well with the asymptotes of the analytical expressions, see Fig. 4.1, and we see that the cooperative schemes achieve a diversity order of two which leads to steeper slopes in the logarithmic plots of the $P_{out}$. For high SNR we then see that $RC$ and $PR$ outperform $DL$ and we can also see that $PR$ provides a power gain of about 8dB over $RC$ for the assumed situation.

The results of the audio communication system tests show that $DL$ transmission, not using the relay at all, actually gives a better performance than $RC$ and in Fig. 5.7 we can see that my implementation of the $RC$ scheme gives a consistently higher BER than the $DL$ scheme. The hamming coded transmissions, as can be expected, give better performance results than the uncoded transmissions for both the $DL$ and $RC$ schemes and this improvement seems to grow as the SNR increases.

6.2 Analysis

To get the advantages of relaying it is very important for the destination to have good knowledge about the SNR of the separate links $S-D$ and $R-D$ in order for the combination process to work well. When fades occur in the different transmission paths, $S-D$ and $R-D$, they are unlikely to occur at the same time instant in the separate paths and $MRC$ combining can then
make use of the high-SNR parts of each signal and reduce the probability of errors in the final estimate of the transmitted data. As the conditions of the links change the MRC combiner has to be able to adapt and change the gain factors used when combining the signals from the source and the relay, otherwise the combiner will not know which link provides the most reliable signal and could, in a worst case scenario, cause an incorrect data estimate when a correctly received signal was available. Due to problems with estimating the SNR of the links, while running the system, the implementation of the MRC in the audio communication system is far from optimal and this could be one of the reasons that we see RC being outperformed by DL. I also believe that, although it very much exists in the audio channel, fading is not a big problem for the system. This could be explained by the use of guard bands between each transmitted symbol and having a slow data rate over the channel compared to the speed of the fading. This reduces some of the possible gain from using relaying techniques. If there is no big gain from using relaying in the audio communication system, an apparent risk is that a sub-optimally implemented relaying scheme introduces more errors than it counteracts and that the simpler non-cooperative DL scheme works better.

6.3 Conclusions

I have constructed an audio communication system to be used by two or three computers and implemented relaying schemes into the system to test their performance over an audio channel. Monte Carlo simulations showed the gain that relaying schemes can achieve compared to direct link transmission under the right circumstances. We saw that different relaying schemes, that divide the available time and transmit power differently between the source and the relay, yield different performance and that partial repetition coding can give a large improvement over conventional repetition coding. However, there are very many parameters and design choices that all have an effect on the performance of communication. The distances between nodes, the data rate of transmission, block length, channel codes and other design choices all affect the performance and it is very hard to draw general conclusions about the schemes. In the audio communication system the RC scheme did not show the improvement over DL transmission that was expected. This, however, says little about the scheme itself and more about the situation with the audio channel, my implementation of the schemes into the audio communication system and how the tests were performed.
Audio channels are rarely or never used in real applications because of extensive interference, short transmission ranges and because it makes alot of noise. A reason for using an audio channel in this thesis was that it required no special equipment and to design a communication system working over the channel you encounter the same kind of design choices and have to take the same basic problems into consideration as for other types of wireless communication. My audio communication system can give an intuitive understanding of wireless communication and the advantages and drawbacks of different types of coding, modulation and the use of relays. It is well suited for further development and possible use in laborations for basic communication courses at the university to let students investigate and see how different setups and design choices change the performance of wireless communication.

6.4 Future Work

The programs for the source, relay and destination computers, that control the audio communication system, are developed in a block manner so that additional coding algorithms and communication schemes easily can be added to the system without altering the basic framework. Parts of the system, such as combination algorithms or synchronization techniques can also be further developed separately. Some examples of interesting future development that I have not had time to implement could be

- Improvement of the combination algorithm.
- Adding more coding algorithms and making the length of block codes scalable.
- Adding additional modulation techniques.
- Performing more tests with other setups and other parameter choices.
- Adding functionality for transmitting files from disk.
- Adding re-transmission functionality for incorrectly received blocks.
Bibliography


Appendix A

Matlab code - Source

A.1 source.m

function source(scheme , data , hamming, delta )
% syntax: source(scheme, time, hamming, delta )
% Called by sourceGUI.m to run the selected relaying-scheme with
% selected parameters.
% Written By: Thomas Wärme
clc;
% Parameters
F=8000; %Channel rate [ sampels/s ] 
Ns=200; %Sampels per symbol 
Nf=7; %Symbols per frame 
Ampl=2; %Amplification Factor
fc1=1300; %Carrier freq; binary 0
fc2=1100; %Carrier freq; binary 1
% Kill any open DAQ-objects
openDAQ=daqfind;
for i=1:length(openDAQ)
    stop(openDAQ(i));
    delete(openDAQ(i));
end
% Check that input data is binary
for i=1:length(data)
    if not(data(i)==0 | data(i)==1)
        fprintf( 'Error! Input data is not binary!\n' )
        return
    end
end
% Run the chosen scheme
if strcmp(scheme, 'DL')
    DLsource(data, hamming, fc1, fc2, F, Ns, Ampl);
end
if strcmp(scheme, 'RC')
    RCsource(data, hamming, fc1, fc2, F, Ns, Nf, Ampl);
end
if strcmp(scheme, 'PR')
    PRsource(data, hamming, fc1, fc2, F, Ns, Nf, Ampl, delta);
end

% Kill any open DAQ-objects
openDAQ=daqfind;
for i=1:length(openDAQ)
    stop(openDAQ(i));
    delete(openDAQ(i));
end

A.2 DLsource.m

function DLsource(data, hamming, fc1, fc2, F, Ns, Ampl)
% syntax: DLsource(data, hamming, fc1, fc2, F, Ns, Ampl)
% Transmits data over Audio-channel with BFSK-modulation using
% carrier frequencies fc1 and fc2, sample rate F, Ns samples/bit
% and amplification factor Ampl. Using hamming=1 gives hamming
% coded transmission, hamming=0 gives uncoded transmission.
% Input 'data' as a binary row vector.
% Written By: Thomas Wärme
fprintf('Direct Link Transmission
')
% Coding
if hamming==1
cdata=code_hamming(data);
else
cdata=data;
end
% Insert synchronization-sequences
pn=[1 1 1 0 0 1 1 1];
cdata=[pn, cdata, pn];
% Modulation BFSK
data_mod=bfskmod(cdata, fc1, fc2, Ns, Ampl);
% Transmission
ao=analogoutput('winsound');
addchannel(ao, [1 2]);
set(ao, 'SampleRate', F);
putdata(ao, [data_mod data_mod]);
function RCsource(data, hamming, fc1, fc2, F, Ns, Nf, Ampl)
% syntax: RCsource(data, hamming, fc1, fc2, F, Ns, Nf, Ampl)
% Transmits data over Audio-channel with BFSK-modulation and
% Repetition Coding Cooperative transmission scheme.
% Transmission uses carrier frequencies fc1 and fc2, sample rate F, Ns samples/bit, block size Nf and amplification
% factor Ampl. Using hamming=1 gives hamming coded
% transmission, hamming=0 gives uncoded transmission.
% Input 'data' as a binary row vector.
% Written By: Thomas Wärme

fprintf('Repetition Coding Transmission------------------\n')
if hamming==1
cdata=code_hamming(data);
else
    cdata=data;
end
% Insert synchronization-sequences-----------------------------------
pn=[1 1 0 0 1 1];
cdata=[pn, cdata, pn];
%Modulation BFSK-----------------------------------------------------
data_mod=bfskmod(cdata, fc1, fc2, Ns, Ampl);
%Block separation----------------------------------------------------
block_L=Nf*Ns;
space=zeros(block_L+Ns*8,1); %Incl two 4*Ns guard intervals
data_spaced=space;

fprintf('Number of bits to Transmit: ');
disp(length(data));
if hamming==1
    fprintf('Number of hamming coded bits to transmit: ');
disp(length(cdata)-2*length(pn));
end
fprintf('Estimated Time for Transmission [s]: ');
disp((get(ao, 'samplesavailable')-16*200)/F);
set(ao, 'TriggerType', 'Immediate'); %Trigger immediately
start(ao); %Activate hardware and DAQ-engine
wait(ao,120); %Wait for output to finish [max 2 min]
fprintf('Transmission Complete.\n');
end
while length(data_mod)>block_L
    data_spaced=[data_spaced; data_mod(1:block_L); space];
    data_mod=data_mod(block_L+1:end);
end

data_spaced=[data_spaced; data_mod; space];

% Printouts

if hamming==1
    fprintf('
Number of bits before coding: ')
disp(length(data))
    fprintf('
Number of hamming coded bits to transmit: ');
disp(length(cdata)-2*Nf)
else
    fprintf('
Number of uncoded bits to transmit: ')
disp(length(data))
end

fprintf('
Number of blocks to transmit: ');
disp(ceil((length(cdata)-2*Nf)/Nf))

% Transmission

ao = analogoutput('winsound');  %Create analog output object
addchannel(ao, [1 2]);  %Add two channels
set(ao, 'SampleRate', F);  %Output F sampels/s
set(ao, 'TriggerType', 'Immediate');
putdata(ao, [ data_spaced data_spaced ]);  %Enque modulated data

fprintf('
Estimated Time for Transmission [s]: ');
disp(((get(ao, 'samplesavailable')-6400)/F))

start(ao);  %Activates hardware and DAQ-engine.
wait(ao,120);  %Wait for output to finish or max of 2 minutes.

fprintf('
Transmission Completed.
');

A.4 PRsource.m

function PRsource(data, hamming, fc1, fc2, F, Ns, Nf, Ampl, delta)

% syntax: PRsource(data, hamming, fc1, fc2, F, Ns, Nf, Ampl, delta)
% Transmits data over Audio-channel with BFSK-modulation and
% Partial Repetition Coding Cooperative transmission scheme.
% Transmission uses carrier frequencies fc1 and fc2, sample rate F, Ns samples/bit, block size Nf and amplification factor
% Ampl. ‘delta’ varies the cooperation level. Using hamming=1 gives hamming coded transmission, hamming=0 gives uncoded
% transmission. Input 'data' as a binary row vector.
% Written By: Thomas Wärme
fprintf('Partial_Repetition_Coding_Transmission
')

if hamming==1
cdata=code_hamming(data);
else
cdata=data;
end

% Insert synchronization sequences
pn=[1 1 1 0 0 1 1];
cdata=[pn, cdata, pn];

% Modulation BFSK
data_mod=bfskmod(cdata, fc1, fc2, Ns, Ampl);

% Block separation
Frame_L=Nf*Ns;
space=zeros(Frame_L+Ns*8,1); % Incl two 4*Ns guard intervals
data_spaced=space;

while length(data_mod)>Frame_L
    data_spaced=[data_spaced; data_mod(1:Frame_L); space];
data_mod=data_mod(Frame_L+1:end);
end

data_spaced=[data_spaced; data_mod; space];

% Printouts
if hamming==1
    fprintf('Number of bits before coding : 
')
disp(length(data))
    fprintf('Number of hamming coded bits to transmit : 
')
disp(length(cdata)-2*Nf)
else
    fprintf('Number of uncoded bits to transmit : 
')
disp(length(data))
end

fprintf('Number of blocks to transmit : 
')
disp(ceil((length(cdata)-2*Nf)/Nf))

% Transmission
ao = analogoutput('winsound'); % Create analog output object
addchannel(ao, [1 2]); % Add two channels
set(ao, 'SampleRate', F); % Output F sampels/s
set(ao, 'TriggerType', 'Immediate');
putdata(ao, [data_spaced data_spaced data_mod]); % Enque modulated data
fprintf('Estimated Time for Transmission [ s ] : 
')
disp((get(ao, 'samplesavailable')-6400)/F)

%Activates hardware and DAQ-engine.
wait(ao,120); %Wait for output to finish or max of 2 minutes.
fprintf('Transmission Completed.\n');

function bfskdata = bfskmod(data, fc1, fc2, Ns, Ampl)
% syntax: bfskdata = bfskmod(data, fc1, fc2, Ns, Ampl)
% BFSK-modulation of a binary data vector 'data' with
% amplification factor Ampl and carrier frequencies
% fc1 and fc2. Returns a modulated signal with 200
% samples per bit including guard intervals between bits.
% Written By: Thomas Wärme 2009-02-12

t=linspace(0,1,0.6*Ns);

s0=Ampl*cos(2*pi*fc1*t); %Binary 0
s1=Ampl*cos(2*pi*fc2*t); %Binary 1

for i = 1:length(data)
    if data(i) == 0
        bfskdata=[bfskdata ; guard ; s0'; guard];
    end
    if data(i) == 1
        bfskdata=[bfskdata ; guard ; s1'; guard];
    end
end

function cdata=code_hamming(data)
% syntax: coded_data = hamming(data)
% Codes a single row binary vector 'data'
% with a (7,4) hamming-code.
% Written By: Thomas Wärme 2009-02-21

rowdata = []; 
for i = 1:length(data)/4
    rowdata(i,1:4)=data(1:4);
    data=data(5:end);
end
A.6 code-hamming.m

G = [[0, 1, 1, 0, 1, 1, 0; 1, 0, 1; 1, 1, 0; 1, 1, 1], eye(4)];
prod = Mod2MatMul(rowdata, G);

% Switch back to single row vector
[rows cols] = size(prod);
cdata = [];
for i = 1:rows
cdata = [cdata, prod(i, 1:7)];
end
end

function out = Mod2MatMul(matr1, matr2)
% Finds a modulo 2 based matrix product of binary matrices
[r1, c1] = size(matr1);
[r2, c2] = size(matr2);
if c1 ~= r2
out = zeros(r1, c2);
for i = 1:r1
    for j = 1:c2
        for k = 1:c1
            out(i, j) = xor(out(i, j), matr1(i, k) * matr2(k, j));
        end
    end
end
else
    Non_Matching_Matrices;
    out = zeros(r1, c2);
    for i = 1:r1
        for j = 1:c2
            for k = 1:c1
                out(i, j) = xor(out(i, j), matr1(i, k) * matr2(k, j));
            end
        end
    end
end
end
Appendix B

Matlab Code - Relay

The Relay Computer also uses bfskmod.m and code-hamming.m seen in Appendix A.5 and Appendix A.6.

B.1 relay.m

```matlab
function relayed=relay(scheme, comp_data, hamming, delta)
% syntax: relayed=relay(scheme, comp_data, hamming, delta)
% Written By: Thomas Wärme

clc;

% Parameters
F=8000; % Channel rate [samples/s]
Ns=200; % Samples per symbol
Nf=7; % Symbols per frame
Ampl=2; % Amplification Factor

fc1=1300; % Carrier freq 1= binary 0
fc2=1100; % Carrier freq 2= binary 1

if hamming==0
    time=10+round(length(comp_data)/11);
else
    time=10+round(length(comp_data)/11/4*7);
end

% Run selected scheme
if strcmp(scheme, 'RC')
    relayed=RCrelay(time, comp_data, hamming, fc1, fc2, F, Ns, Nf, Ampl);
end

if strcmp(scheme, 'PR')
```

46
B.2 RCrelay.m

```matlab
function relayed=RCrelay(time,comp_data,hamming,fc1,fc2,F,Ns,Nf,Ampl,delta);

end
end

% Stop and Delete any open DAQ-objects
openDAQ=daqfind;
for i=1:length(openDAQ)
    stop(openDAQ(i));
    delete(openDAQ(i));
end
end

B.2 RCrelay.m

function relayed=RCrelay(time,comp_data,hamming,fc1,fc2,F,Ns,Nf,Ampl)
% syntax: relayed=RCrelay(time,comp_data,hamming,fc1,fc2,F,Ns,Nf,Ampl)
% Records data from the Audio-channel during "time" [s] with BFSK-demodulation. Re-modulation of the signal and re-transmission.
% Written By: Thomas Wärne
clc;

%Parameters
G=800; %Guard Frame Length [samples]
T=Ns*Nf; %Transmit/Receive Frame Length [samples]
relayed =[]; %Initiate vector for relayed bits.
runs=1; %Start value for number of relay runs
relay_ok=1; %Initiate relay_ok as 1

%Initiate Input
ai = analoginput('winsound'); %Create analog input object
addchannel(ai, 1); %Add one channel
ai.SampleRate=F; %Sample rate [samples/s]
ai.SamplesPerTrigger=time*F; %Samples per trigger
ai.TriggerType='Immediate'; %Trigger immediately after start

%Initiate Output
ao = analogoutput('winsound'); %Create analog output object
addchannel(ao, [1 2]); %Add two channels
ao.SampleRate=F; %Output rate [samples/s]
ao.TriggerType='Immediate'; %Trigger immediately after start

%Detect Start sequence and Synchronize
start(ai);
fprintf('Waiting for signal ...
');
sensedata=getdata(ai, Ns);
chunk=(abs(fft(sensedata(1:Ns))));
```
signal_limit = \( \text{sum}(\text{abs}(\text{chunk}(21:40))) / 20; \)

% Detect pilot sequence
while chunk(50) < 5 * signal_limit
    sensedata=peekdata(ai, Ns);
    chunk=(\text{abs(fft(sensedata(1:Ns)))});
end

% Synchronize start
chunk=(\text{abs(fft(pre_data(1:200)))});
limit=5 * \text{sum(chunk(11:40)) / 30};
while chunk(50) < limit
    pre_data=pre_data(10:end);
    if length(pre_data) > 200
        chunk=(\text{abs(fft(pre_data(1:200)))});
    else
        return;
    end
end

overstep=length(pre_data) - 200 + 20;
sync_frame=getdata(ai, T + T + 2 * G - overstep);

% Start Relaying Loop
fprintf('Signal detected, starting relay!\n')
while runs < 100 & relay_ok == 1
    indata=getdata(ai, T);
    \[ blockskattn, demod_ok \] = bfskdemod(signal_limit, indata, Ns);
    if blockskattn == \[5 5 5 5 5 5 5\]
        relay_ok = 0;
    end
    if demod_ok == 1 && relay_ok == 1
        if hamming == 1
            decoded = decode_hamming(blockskattn);
            relayed = [relayed, decoded];
            reencoded = code_hamming(decoded);
            data_mod = bfskmod(reencoded, fc1, fc2, Ns, Ampl);
        else
            relayed = [relayed, blockskattn];
            data_mod = bfskmod(blockskattn, fc1, fc2, Ns, Ampl);
        end
        putdata(ao, [data_mod, data_mod]);
        start(ao)
    else
        relayed = [relayed, 5 * ones(1, 7)];
function relayed=PRrelay(time,comp_data,hamming,fc1,fc2,F,Ns,Nf,Ampl,delta)

% syntax: relayed=PRrelay(time,comp_data,hamming,fc1,fc2,F,Ns,Nf,Ampl,delta)
% Records data from the Audio-channel during "time" [s] with
% BFSK-demodulation. Re-transmission of the signal and
% Written By: Thomas Wärne

clc;

G=800; %Guard Frame Length [samples]
T=Ns*Nf; %Transmit/Recieve Frame Length [samples]
relayed=[]; %Initiate vector for relayed bits.
runs=1; %Start value for number of relay runs
relay_ok=1; %Initiate relaying active
dL=round((1-delta)*Nf*2); %Length of data to relay

%Initiate Input
ai = analoginput('winsound'); %Create analog input object
addchannel(ai, 1); %Add one channel
ai.SampleRate=F; %Sampelrate [samples/s]
ai.SamplesPerTrigger=time+F; %Sampels per trigger
ai.TriggerType='Immediate'; %Trigger immediately after start

%Initiate Output
ao = analogoutput('winsound'); %Create analog output object
addchannel(ao, [1 2]); %Add two channels
ao.SampleRate=F; %Output rate [sampels/s]
ao.TriggerType='Immediate'; %Trigger immediately after start

%Detect Start sequence and Synchronize
start(ai)
fprintf('Waiting for signal ...
')
sensedata=getdata(ai, Ns);
chunk=(abs(fft(sensedata(1:Ns))));
signal_limit=sum(abs(chunk(21:40)))/20;

% Detect pilot sequence
while chunk(50)<5*signal_limit
sensedata=peekdata(ai, Ns);
chunk=(abs(fft(sensedata(1:Ns))));
end
fprintf('Synchronization Sequence Identified
')
avail=ai.SamplesAvailable;
pre_data=peekdata(ai, avail);
flushdata(ai);
pre_data=pre_data(end-800:end);

% Synchronize start
chunk=(abs(fft(pre_data(1:200))));
limit=5*sum(chunk(11:40))/30;
while chunk(50)<limit
pre_data=pre_data(10:end);
if length(pre_data)>200
    chunk=(abs(fft(pre_data(1:200))));
else
    return;
end
overstep=length(pre_data)-200+20;
sync_frame=getdata(ai, T+T+2*G-overstep); %#ok<NASGU>

%Start Relaying Loop
fprintf('Signal detected, starting relay !
')
while runs<50 & relay_ok==1 %#ok<AND2>
%Demodulation

if blockskattn == [5 5 5 5 5 5 5] \#ok<BDSCA>
    relay_ok = 0;
end
if demod_ok == 1 && relay_ok == 1
    if hamming == 1
        decoded = decode_hamming(blockskattn);
        relayed = [relayed, decoded];
        data_mod = bfskmod(reencoded(1:dL), fc1, fc2, Ns, Ampl);
    else
        relayed = [relayed, blockskattn];
        data_mod = bfskmod(blockskattn(1:dL), fc1, fc2, Ns, Ampl);
        data_mod = [data_mod; zeros(Ns*(Nf-dL), 1)];
    end
    putdata(ao, [data_mod data_mod]);
    start(ao)
else
    relayed = [relayed, 5*ones(1,Nf)];
    break;
end

%Aftermath

fprintf('Relaying Finished.
')
fprintf('Cooperation Level [percentage]: ')
disp(round(200*(1-delta)))
if hamming == 1
    fprintf('Number of Blocks Received: ')
disp(length(relayed)/Nf/4*dL)
    fprintf('Number of Hamming Coded Bits Received: ')
disp(length(relayed)/4*dL)
    fprintf('Number of Hamming Coded Bits Relayed: ')
disp(length(relayed)/Nf/4*dL)
else
    fprintf('Number of Blocks Received: ')
disp(length(relayed)/Nf)
    fprintf('Number of Uncoded Bits Received: ')
disp(length(relayed)/Nf/4*dL)
end
disp(length(relayed))
fprintf(’Number of Uncoded Bits Relayed: ’)
disp(length(relayed)/Nf*dL)
end
fprintf(’−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
’)
% BER−calculation−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
errorcheck(relayed, comp_data);
end

B.4 bfskdemod.m

function [demodulated demod_ok] = bfskdemod(limit, indata, Ns)
% syntax: [demodulated demod_ok] = bfskdemod(limit, indata, Ns)
% BFSK−demodulation of recorded signal. Returns a estimated
% binary data vector ’demodulated’ and ’demod_ok’ which shows
% if all bits were decoded successfully or not.
% Written By: Thomas Wärme 2009−02−12

L=floor(length(indata)/(Ns)); %Number of symbols in input
demodulated=[]; %Initiate demodulated output vector
demod_ok=1;
for i = 1:L
    if length(indata((i*Ns−Ns)+1:end))>Ns
        symbol=indata((i*Ns−Ns)+1:i*Ns);
    else
        piece=indata(1+(i*Ns−Ns):end);
        symbol=[piece zeros(1, Ns−length(piece))];
    end
    S=abs(fft(symbol));
    if S(50)>S(16) && S(50)>5*limit
        demodulated(i)=1;
    end
    if S(16)>S(50) && S(16)>5*limit
        demodulated(i)=0;
    end
    if S(50)<5*limit && S(16)<5*limit
        demodulated(i)=5;
    end
    demod_ok=0;
end
end

B.5 decode-hamming.m

function decoded = decode_hamming(indata)
B.5 decode-hamming.m

2  % syntax: decoded_data=decode_hamming(indata)
3  % Decodes single row vector 'data' coded with hamming (7,4)-code
4  % Written By: Thomas Wärme 2009–02–21
5  
6  %Switch into matrix of row length 7.
7  received_row =[];
8  for i=1:length(indata)/7
9      received_row(i,1:7)=indata(1:7);
10     indata=indata(8:end);
11  end
12  
13  H=[eye(3),transpose([[0,1,1;1,0,1;1,1,0;1,1,1]])];
14  %Calculate the Syndromes
15  Syndromes=Mod2MatMul(received_row,H');
16  %Error Correction
17  decoded7bit=xor(errorpattern(Syndromes),received_row);
18  decoded4bit=decoded7bit(:,[4,5,6,7]); %Remove the parity bits
19  
20  %Switch back to single row vector
21  [rows cols]=size(decoded4bit);  %#ok<NASGU>
22  decoded =[];
23  for i=1:rows
24      decoded=[decoded, decoded4bit(i,1:4)];
25  end
26  end
27  
28  function out=Mod2MatMul(matr1, matr2)
29  % syntax: product=Mod2MatMul(matr1,matr2)
30  % Finds a modulo 2 based matrix product of binary matrices
31  [r1,c1]=size(matr1);  
32  [r2,c2]=size(matr2);  
33  if c1==r2
34    'Non_Matching\,Matrices';
35  else
36    out=zeros(r1,c2);
37    for i=1:r1
38      for j=1:c2
39        for k=1:c1
40          out(i,j)=xor(out(i,j),matr1(i,k)*matr2(k,j));
41        end
42      end
43    end
44  end
45  
46  function out=errorpattern(syndrome)  
47  % function to get the error pattern given the syndrome
% Works only for parity check matrix
H=[eye(3),transpose([0,1,1;1,0,1;1,1,0;1,1,1])];

r=size(syndrome,1);
out=zeros(r,7);

for i=1:r
    if sum(xor(syndrome(i,:),[0,0,0]))==0
        out(i,:)=[0,0,0,0,0,0,0];
    end
    if sum(xor(syndrome(i,:),[0,0,1]))==0
        out(i,:)=[0,0,1,0,0,0,0];
    end
    if sum(xor(syndrome(i,:),[0,1,0]))==0
        out(i,:)=[0,1,0,0,0,0,0];
    end
    if sum(xor(syndrome(i,:),[0,1,1]))==0
        out(i,:)=[0,0,0,1,0,0,0];
    end
    if sum(xor(syndrome(i,:),[1,0,0]))==0
        out(i,:)=[1,0,0,0,0,0,0];
    end
    if sum(xor(syndrome(i,:),[1,0,1]))==0
        out(i,:)=[0,0,0,0,1,0,0];
    end
    if sum(xor(syndrome(i,:),[1,1,0]))==0
        out(i,:)=[0,0,0,0,0,1,0];
    end
    if sum(xor(syndrome(i,:),[1,1,1]))==0
        out(i,:)=[0,0,0,0,0,0,1];
    end
end

B.6 errorcheck.m

function errorcheck(relayed, comp_data)
% syntax: errorcheck (relayed, comp_data)
% Performs error checks with 'comp_data' as the reference of translanted data and displays the results.
% Written By: Thomas Wärme 2009–03–05

L_relayed=length(relayed);
L_comp=length(comp_data);

if L_relayed==L_comp
    if relayed==comp_data;
        biterrors=0;
        fprintf('BER_for,S>R,Signal: ')
        disp(biterrors)
        fprintf('Relaying,Succesful!
');
    end
end
else
    biterrors = 0;
    for i = 1:L_relayed
        if not(relayed(i) == comp_data(i))
            biterrors = biterrors + 1;
        end
    end
    fprintf('BER for S→R Signal: ')
    disp(biterrors/L_relayed)
    fprintf('Relaying Failure!
');
end
else
    fprintf('Relaying Failure!
');
    fprintf('Recieved data is of wrong length.
');
end
end
Appendix C

Matlab Code - Destination

The Destination Computer also uses \texttt{bfskdemod.m} and \texttt{decode-hamming.m} seen in Appendix B.4 and Appendix B.5.

C.1 destination.m

```matlab
function received=destination (scheme, comp_data, hamming, delta)

% syntax: received=destination (scheme, comp_data, hamming, delta)
% Written By: Thomas Wärme

clc;

% Parameters
F=8000; % Channel rate [samples/s]
Ns=200; % Sample per symbol
Nf=7; % Symbols per frame
fc1=1300; % Carrier freq 1= binary 0
fc2=1100; % Carrier freq 2= binary 1

if hamming==0
    time=5+round(length(comp_data)/11);
else
    time=5+round(length(comp_data)/11/4*7);
end

% Kill open DAQ-objects
openDAQ=daqfind;
for i=1:length(openDAQ)
    stop(openDAQ(i));
    delete(openDAQ(i));
end

if strcmp(scheme, 'DL')
    time=2+length(comp_data)/8;

56
```
recieved=DLdestination(time,comp_data,hamming,fc1,fc2,F,Ns,Nf);

if strcmp(scheme,'RC')
    recieved=RCdestination(time,comp_data,hamming,fc1,fc2,F,Ns,Nf);
end

if strcmp(scheme,'PR')
    recieved=PRdestination(time,comp_data,hamming,...
        fc1,fc2,F,Ns,Nf,delta);
end

---

function decoded=DLdestination(time,comp_data,hamming,fc1,fc2,F,
    Ns,Nf)

% syntax: decoded=DLdestination(time,comp_data,hamming,fc1,fc2,F,
% Ns,Nf)
% Written By: Thomas Wärme 2009−02−12

% Input
ai=analoginput('winsound'); %Create analog input object
addchannel(ai,1); %Add one channel
ai.SampleRate = F; %Sample rate [sampels/s]
ai.SamplesPerTrigger = F*time; %Sampels per trigger
ai.TriggerType = 'Immediate'; %Trigger immediately at start
fprintf('Recording from channel ...
');
start(ai)
[rec_signal,t] = getdata(ai);
fprintf('Post-processing ...
');

% Truncate recording
trunc_data=truncate(rec_signal);

% Check sync-sequences
[synced_avg_f]=synchronize(trunc_data,8);
if length(synced)<400
    decoded=[];
    return;
end

% Demodulation
sig_est=bfskdemod(avg_f,synced);

% Truncate tail
tail_ok=0;
while tail_ok==0
    if sig_est(end-7:end)==[1 1 1 0 1 1 1]
        tail_ok=1;
    end
    sig_est=sig_est(1:end-1);
end
sig_est=sig_est(1:end-7);

% Decoding
if hamming==1
    decoded=decode_hamming(sig_est);
    fprintf('Number of received hamming-coded bits: 
');
    disp(length(sig_est));
    fprintf('Number of bits after decoding: 
');
    disp(length(decoded));
else
    decoded=sig_est;
    fprintf('Number of received uncoded bits: 
');
    disp(length(decoded));
end

% Aftermath
BER_DL=errorcalc(decoded, comp_data);
fprintf('BER for Direct Link Transmission: 
');
disp(BER_DL)
if BER_DL==0
    fprintf('Transmission Successful! 
');
else
    fprintf('Transmission Failed! 
');
end
figure(2)
subplot(3,1,1)
stem(decoded)
title('Recieved_data')
subplot(3,1,2)
stem(comp_data)
title('Original Transmission')
if length(comp_data)==length(decoded)
    subplot(3,1,3)
    stem(comp_data-decoded, 'r')
title('Errors (if any)')
end
function combined_est=RCdestination(time ,comp_data,hamming, fc1 ,
fc2 ,F,Ns,Nf)
% Written By: Thomas Wärme 2009–02–12

ai = analoginput('winsound'); %Create analog input object
addchannel(ai, 1); %Add one channel
ai .SampleRate = F; %Sampels/s
ai .SamplesPerTrigger = F*time; %Sampels per trigger
ai .TriggerType = 'Immediate'; %Trigger immediately after start

fprintf('Recording from Channel ...
')
start (ai)
[ rec_signal , t]=getdata(ai);
fprintf('Post–processing ..
')

% Truncate recording
trunc_data=truncate(rec_signal);
if length(trunc_data)<1000
combined_est=[];
return;
end

% Channel estimation and remove pilot
pilot_chunk=abs(fft(trunc_data(1:Ns)));
limit=sum(pilot_chunk(21:40))/20;
data=trunc_data(4440:end);
Source_soft=[];
Relay_soft=[];

% Soft Decision Demodulation
while length(data)>6000
Source_block=data(1:Nf*Ns);
softblock=bfsk_softdemod(limit, Source_block, Nf);
Source_soft=[Source_soft, softblock];
data=data(1801:end);
[Relay_block trunc_ok]=truncate_start(limit, data(1:2600));
if trunc_ok==1
    Relay_block=Relay_block(1:Nf*Ns);
    softblock=bfsk_softdemod(limit, Relay_block, Nf);
    Relay_soft=[Relay_soft, softblock];
else
    Relay_soft=[Relay_soft, zeros(Nf,2)];
end
data=data(2601:end);
60 Matlab Code - Destination

% Combining
combined_soft=combine(Source_soft, Relay_soft, 1);

% Hard Decision Demodulation
combined_hard=bfsk_harddemod(combined_soft);
relay_hard=bfsk_harddemod(Relay_soft);
source_hard=bfsk_harddemod(Source_soft);

% Decoding
if hamming==1
    source_est=decode_hamming(source_hard);
    relay_est=decode_hamming(relay_hard);
    combined_est=decode_hamming(combined_hard);
    fprintf('Number of recieved blocks: 
    disp(length(combined_hard)/7);
    fprintf('Number of recieved, hamming-coded, bits: 
    disp(length(combined_hard));
    fprintf('Number of bits after decoding: 
    disp(length(combined_est));
else
    combined_est=combined_hard;
    relay_est=relay_hard;
    source_est=source_hard;
    fprintf('Number of recieved blocks: 
    disp(length(combined_est)/7);
    fprintf('Number of recieved, uncoded, bits: 
    disp(length(combined_est));
end

% Aftermath
BER_source=errorcalc(source_est, comp_data);
BER_relay=errorcalc(relay_est, comp_data);
BER_combined=errorcalc(combined_est, comp_data);
fprintf('BER for S->D Signal: 
disp(BER_source)
fprintf('BER for S->R->D Signal: 
disp(BER_relay)
fprintf('BER for Combined Signal: 
disp(BER_combined)
if BER_combined==0
    fprintf('Transmission Successful! 
else
fprintf('Transmission Failed!\n')
end

figure(2)
subplot(3,1,1)
stem(combined_est)
title('Recieved Data at Destination ')
subplot(3,1,2)
stem(comp_data)
title('Transmitted Data from Source ')
if length(comp_data)==length(combined_est)
    subplot(3,1,3)
    stem(comp_data-combined_est, 'r ')
title('Errors ( if any) ')
end

c.4 PRdestination.m

function combined_est=PRdestination(time,comp_data,hamming,fc1 , fc2 ,F,Ns,Nf,delta)
% Written By: Thomas Wärme 2009−02−12

dL=round((1−delta)*Nf*2); %Length of relayed data

ai = analoginput('winsound'); %Create analog input object
addchannel(ai, 1); %Add one channel
ai. SampleRate = F; %Sampels/s
ai. SamplesPerTrigger = F*time; %Sampels per trigger
ai. TriggerType = 'Immediate '; %Trigger immed. after start

fprintf('Recording from Channel ...
')
start(ai)
[rec_signal, t] = getdata(ai);
fprintf('Post-processing ...
')

% Truncate recording
trunc_data=truncate(rec_signal);

% Channel estimation and remove pilot
pilot_chunk=abs(fft(trunc_data(1:Ns)));
limit=sum(pilot_chunk(21:40))/20;
data=trunc_data(4400:end);
Source_soft=[];
Relay_soft=[];

% Soft Decision Demodulation
while length(data)>6000
Source_block=data(1:Nf*Ns);
softblock=bfsk_softdemod(limit, Source_block, Nf);
Source_soft=[Source_soft, softblock];
data=data(1600:end);

[Relay_block trunc_ok]=truncate_start(limit, data(1:2800));
if trunc_ok==1
    Relay_block=Relay_block(1:Nf*Ns);
    softblock=bfsk_softdemod(limit, Relay_block, Nf);
    softblock(dL+1:end,:)=0;
    Relay_soft=[Relay_soft, softblock];
else
    Relay_soft=[Relay_soft, zeros(Nf,2)];
end
end
data=data(2801:end);

% Combining
combined_soft=combine(Source_soft, Relay_soft, 1);

% Hard Decision Demodulation
combined_hard=bfsk_harddemod(combined_soft);
relay_hard=bfsk_harddemod(Relay_soft);
source_hard=bfsk_harddemod(Source_soft);

% Decoding
fprintf('------------------------
')
fprintf('Cooperation Level [percentage]: ')
disp(round(200*(1-delta)));
if hamming==1
    combined_est=decode_hamming(combined_hard);
    relay_est=decode_hamming(relay_hard);
    source_est=decode_hamming(source_hard);
    fprintf('Number of Recieved Hamming Coded Bits: ');
disp(length(combined_hard));
    fprintf('Number of Bits after Decoding: ');
disp(length(combined_est));
else
    combined_est=combined_hard;
    relay_est=relay_hard;
    source_est=source_hard;
    fprintf('Number of Recieved Uncoded Bits: ');
disp(length(combined_est));
    fprintf('Number of Recieved Bits from Source: ');
disp(length(source_est));
    fprintf('Number of Recieved Bits from Relay: ');
disp(length(relay_est));
end
fprintf('------------------------
')
% Aftermath

BER_source=errorcalc(source_est, comp_data);
fprintf('BER for S-->D Signal: ')
disp(BER_source)

if length(relay_est)>0
    comp_data_PR=[];
    comp_data_P=comp_data;
    while length(comp_data_P)>6
        comp_data_PR=[comp_data_PR, comp_data_P(1:dL)];
        comp_data_P=comp_data_P(8:end);
    end
    BER_relay=errorcalc(relay_est, comp_data_PR);

    if hamming==0
        fprintf('BER for S-->R-->D Signal: ')
disp(BER_relay)
    end
end

BER_combined=errorcalc(combined_est, comp_data);
fprintf('BER for Combined Signal: ')
disp(BER_combined)

if BER_combined==0
    fprintf('Transmission Successful!
')
else
    fprintf('Transmission Failed!
')
end

figure(2)
subplot(3,1,1)
stem(combined_est)
title('Combined Signal')
subplot(3,1,2)
stem(comp_data)
title('Original Signal')
subplot(3,1,3)
if length(comp_data)==length(combined_est)
    stem(comp_data(combined_est), 'r')
title('Errors (if any) ')
end

C.5 bfsk-softdemod

function soft_demodulated=bfsk_softdemod(signalmedel, indata, Nf)
% syntax: soft_demodulated=bfsk_softdemod(signalmedel, indata, Nf)
% BFSK-demodulation of continuous signal that has been sampled
% and saved. Returns a matrix of frequency values for the two
% carrier frequencies after soft-decision demodulation.
% Indata is the received sampled signal.
% Written by Thomas Wärme.

Ns=200; %symbol length [samples]
soft_demodulated=zeros(Nf,2); %soft demodulated data to output

for i = 1:Nf
    if length(indata ((i * Ns - Ns)+1:end))>Ns
        symbol=indata ((i * Ns - Ns)+1:i * Ns);
    else
        snutt=indata(1+(i * Ns - Ns):end);
        symbol=[snutt zeros(1, Ns-length(snutt))];
    end
    S=abs(fft(symbol));
    soft_demodulated(i,1)=S(16);
    soft_demodulated(i,2)=S(50);
end

function hard_data = bfsk_harddemod(soft_data)
% syntax: hard_data = bfsk_harddemod(soft_data)
% Returns a binary vector after hard-decision demodulation.
% Written by Thomas Wärme.

S=size(soft_data);
rows=S(1); %number of rows in soft_data
cols=S(2); %number of cols in soft_data
hard_data=[];

for i =1:2:cols
    for j=1:rows
        if soft_data(j,i)>soft_data(j,i+1)
            hard_data=[hard_data, 0];
        end
        if soft_data(j,i)<soft_data(j,i+1)
            hard_data=[hard_data, 1];
        end
    end
end

C.6 bfsk-harddemod

C.7 combine.m
function combined_est=combine(Source_soft, Relay_soft, method)
% syntax: combined_est=combine(source_soft, Relay_soft, method)
% Combines two matrices of soft symbols and outputs a matrix of
% the same size as the largest input matrix. If size of inputs
% differ the smallest matrix will get added zeros to make them
% equal length.
% Written by Thomas Wärme.
Relay_L=size(Relay_soft);
Source_L=size(Source_soft);

% Gain factors for the MRC combining. [To do: Base on rel. SNR!]
k1=1;
k2=2^k1;

% Selection Combining
if method==0
% Choose the channel with the highest SNR! [Not working]
combined_est=Source_soft;
return;
end

% Maximum Ratio Combining
if method==1
if Relay_L(2)>Source_L(2)
Source_soft=[Source_soft, zeros(Source_L(1), Relay_L(2)-Source_L(2))];
end
if Source_L(2)>Relay_L(2)
Relay_soft=[Relay_soft, zeros(Relay_L(1), Source_L(2)-Relay_L(2))];
end
combined_est=(k1*Source_soft+k2*Relay_soft)/2;
end
end

C.8 synchronize.m

function [out recpower]=synchronize(in, sync_length)
% syntax [output avg_pow]=synchronize(trunc_input, sync_length)
% Checks if the sync-blocks are correctly received and then
% removes them. sync_length should be given as the length of
% the sync-block [bits]. Based on sync-blocks calculates the
% average received signal power.
% Written By: Thomas Wärme
Ns=200; %# samples per bit.

fprintf('START Sync-sequence:\n');

pn_start=bfskdemod(0, in(1:sync_length*Ns));
part_power=0;

%Calculate average received power
for i=1:sync_length
    pn_bit=abs(fft(in((Ns*i-Ns)+1:i*Ns)));
    part_power=part_power+sum(pn_bit(1:100));
end
recpower=part_power/(100*sync_length);

%Remove start sync-block from the received signal
out=in(sync_length*Ns+1:end);

if pn_start==[1 1 1 0 0 1 1 1]
    fprintf('Synchronization_OK.\n')
else
    fprintf('Error in the synchronization_sequences !\n')
    out=[];
end

function truncated_signal = truncate(indata)
% syntax: truncated_signal=truncate(signal)
% Truncate signal in the start and end.
% Sync-sequence before and after transmission.
% Written By: Thomas Wärme 2009-02-12

% Truncate start
chunk=(abs(fft(indata(1:200))));
while chunk(50)<10*sum(chunk(21:40))/20 %Look for sync
    indata=indata(50:end);
    if length(indata)>200
        chunk=(abs(fft(indata(1:200))));
    else
        fprintf('No_synchronization_sequence_detected.\n');
        fprintf('No_data_received\n')
        truncated_signal=[];
        return
    end
end
chunk_old=chunk;
indata=indata(4:end);
chunk=abs(fft(indata(1:200)));
while sum(chunk_old(48:52))<sum(chunk(48:52)) %Small steps
    chunk_old=chunk;
    indata=indata(4:end);
    chunk=abs(fft(indata(1:200)));
function [truncated trunc_ok] = truncate_start(limit, indata)
% syntax: [trunc_sig trnc_ok] = truncate_start(limit, signal)
% Truncate signal in the start and end.
% Synkroniseringssekvens före sändning och efter sändning
% Written By: Thomas Wärme 2009-02-12

trunc_ok=1;

if length(indata)<201
    trunc_ok=0;
    truncated = [0];
    return;
end

% Truncate start
chunk=abs(fft(indata(1:200)));
while chunk(50)<10*limit && chunk(16)<10*limit
    indata=indata(20:end);
    if length(indata)>1600
        chunk=abs(fft(indata(1:200)));
    else
        fprintf('The_signal_receved_makes_no_sense
')
        fprintf('No_data_recieved
')
        truncated_signal=[];
        return
    end
    chunk_old=chunk;
    indata=indata(1:end-4);
    chunk=abs(fft(indata(end-199:end)));
while sum(chunk_old(48:52))<sum(chunk(48:52)) % Small steps
    chunk_old=chunk;
    indata=indata(1:end-4);
    chunk=abs(fft(indata(end-199:end)));
end
truncated_signal=indata;
end

C.10  truncate-start.m
trunc_ok = 0;
truncated = [0];
return;
end

end

% Move small steps until perfect match
chunk_old = chunk;
indata = indata (4:end);
chunk = abs (fft (indata (1:200)));
while sum (chunk_old (48:52)) < sum (chunk (48:52))
chunk_old = chunk;
indata = indata (4:end);
chunk = abs (fft (indata (1:200)));
end

% truncated = indata;
truncated = indata (100:end);
end

c.11 errorcalc.m

function ber = errorcalc (decod_skattn, comp_data)
% syntax: ber = errorcalc (signalskattn, comp_data)
% Written By: Thomas Wärme
Ls = length (decod_skattn);
Lj = length (comp_data);
biterrors = 0;
if Ls == Lj
if decod_skattn == comp_data;
else
for i = 1:Lj
if not (decod_skattn (i) == comp_data (i))
biterrors = biterrors + 1;
end
end
end
if Ls < Lj
decod_skattn = [decod_skattn, 5 * ones (1, Lj - Ls)];
for i = 1:Lj
if not (decod_skattn (i) == comp_data (i))
biterrors = biterrors + 1;
end
end
if $L_s > L_j$
    decod_skattn = decod_skattn(1:L_j);
    biterrors = $L_s - L_j$;
    for $i = 1:L_j$
        if not (decod_skattn(i) == comp_data(i))
            biterrors = biterrors + 1;
    end
    end
ber = biterrors / L_j;
end
Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera åktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/

© Thomas Wärme