Turbo Coded Hybrid Type II ARQ System

Jing Xu

Supervisors: Sorour Falahati and Bartosz Mielczarek

Examiner: Professor Arne Svensson

CHALMERS

Department of Signals and Systems

April 2002

EX018/2002

Abstract

In this report, the throughput and the delay performance of the Rate Compatible Punctured Turbo (RCPT) codes on Rayleigh fading channels are evaluated through simulations and compared with the corresponding ones of the Rate Compatible Punctured Convolutional (RCPC) codes.

In RCPT codes, higher rate codes than the parent rate codes are obtained through puncturing and the effects of optimal puncturing and random puncturing on the performance are studied. The simulation results show the superior performance of the optimal puncturing as expected. Moreover, the simple repetition and the multiple component encoding are studied as two alternatives to obtain the lower rate codes than the parent code rate one. The results demonstrate negligible difference between the corresponding performances. Additionally, the performance of RCPT codes is compared with RCPC codes for different block lengths. The results show that at high SNRs, almost same performance is obtained by both RCPT and RCPC codes. However at low SNRs, RCPT codes provide superior performance compared to RCPC codes as the block size increases.

Acknowledgments

The purpose of this thesis report is to fulfill the requirements for the Master of Science degree in Electrical Engineering at Chalmers University of Technology. The project was carried out at the Communication Systems Group at the Department of Signals and Systems.

I would like to take this opportunity to thank all the staff at the Communication Systems Group. Specially I would like to give my most sincere gratitude to Professor Arne Svensson for providing me with such a great opportunity to work on this interesting topic. I also own the most to my two supervisors in the group: Sorour Falahati and Bartosz Mielczarek for their enthusiastic support, guidance and invaluable discussions.

I would like to acknowledge the grant I received from the STINT foundation which supported me during my studies at Chalmers.

Finally, I would like to thank all my friends and my family for help and endless encouragement during my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The modern communication systems play an important role in our life and the reliability of these systems is crucial. There has been a great interest in combining different codes and Hybrid type II ARQ scheme. Turbo codes differ from convolutional codes since the encoder contains interleaver and two or more, identical or different recursive systematic convolutional (RSC) encoders, and the decoder is a SISO (Soft-In Soft-Out) a-posteriori probability (APP) decoder with one or more iterations. This unique structure makes turbo codes more powerful than convolutional codes.

A lot of communication systems need to change code rate according to source and channel requirements. The Rate Compatible Punctured Turbo (RCPT) codes use only one encoder and one decoder instead of changing the basic system structure, and switching a set of encoders and decoders. The RCPT codes will become more powerful if used together with hybrid type II ARQ scheme. A typical model of a digital communication system employing generalized hybrid type II ARQ scheme based on RCPT codes will be introduced in the following section.

1.2 System Model

The block diagram of our system is shown in figure 1.1. A message block from data source is encoded with a Cyclic Redundancy Check (CRC) encoder first. The CRC encoded bits are fed into the RCPT encoder. Initially, the RCPT encoder selects the puncturing scheme according to the higher rate code of the family. The output from the RCPT encoder is sent into an interleaver. The interleaved signal is modulated and dumped into channel. In the receiver side, the received signal first passes a BPSK demodulator and a de-interleaver. The bits from the de-interleaver are combined with previously transmitted bits for the same message block, then transmitted into the RCPT decoder. The CRC decoder checks whether the output of the RCPT decoder is correct or not and sends corresponding ACK/NACK message to the transmitter. If the higher rate code is not sufficiently powerful to overcome channel errors, then the transmitter will send incremental redundancy bits which were deleted by the puncturing process. The above process is repeated, as needed, decreasing the code rate

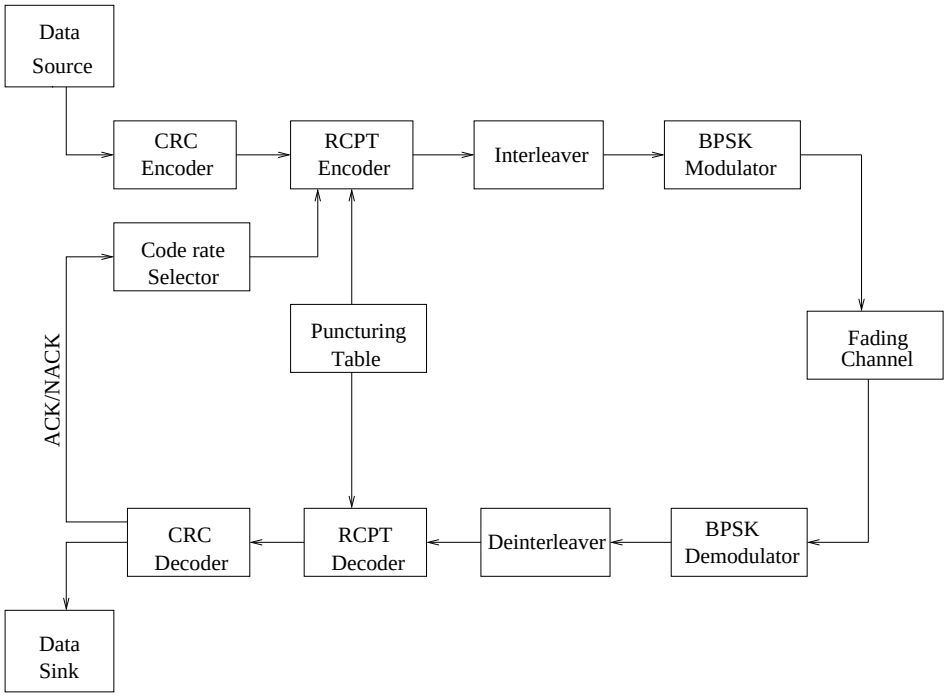


Figure 1.1: RCPT codes for type II hybrid ARQ system model

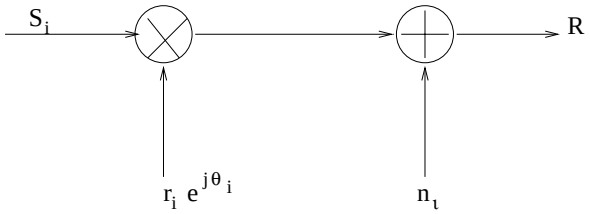


Figure 1.2: The Channel Model

down to that of the parent code. Should decoding fail even with the lowest rate $1/m$ parent code, either the whole process is started again, or codes such as simple repetition coding or multiple component encoding take over. The reason of using repetition coding or multiple component coding is to achieve the low bit error rate by using lower code rate than the parent rate.

In our investigation, we use the Rayleigh fading channel model. The figure 1.2 shows the channel model.

Assume that s_i is a sample of the transmitted signal to the channel. r_i and θ_i are the samples of the Rayleigh fading envelope and phase respectively. θ_i is assumed to be uniformly distributed in the interval $[0, 2\pi]$. r_i has Rayleigh distribution and its probability density distribution function (pdf) is

$$f_r(r) = \begin{cases} \frac{r}{\sigma^2} \exp(-\frac{r^2}{2\sigma^2}) & 0 \leq r \leq \infty \\ 0 & r < 0 \end{cases} \tag{1.1}$$

The mean value $\bar{r}$ of the Rayleigh distribution is given by

$$\bar{r} = E[r] = \int_0^\infty r f_r(r) dr = \sigma\sqrt{\frac{\pi}{2}} = 1.2533\sigma \tag{1.2}$$

The average power of the Rayleigh fading envelope $E[r^2] = 2\sigma^2$, and the variance of the Rayleigh distribution is given by

$$\sigma_r^2 = E[r^2] - (E[r])^2 = 0.4292\sigma^2 \tag{1.3}$$

which also represents the ac power in the signal envelope.

In our simulation, when the sample signal is sent to the channel, first it is multiplied by $r_i e^{j\theta_i}$, the Rayleigh fading envelope, then the noise n_i which is a sample of the complex Gaussian noise with zero mean and variance N_o is added. After the channel, the received sample signal r_i is deinterleaved, and fed into the Maximal Ratio Combiner which is assumed to have the perfect channel state information (CSI) available. Soft decoding is employed by the turbo decoder with the parent code rate $1/m$ (the number of the RSC encoders in turbo encoder is $m-1$).

1.3 Outline of the thesis

We start with an introduction to the turbo codes in chapter 2. In this chapter, we show the structure of turbo encoder and then examine in more details how turbo decoder works.

Chapter 3 explains three basic ARQ schems: Stop-and-Wait ARQ, Go-Back-N ARQ, Selective-Repeat ARQ. We also show how two FEC/ARQ hybrid schemes work.

In Chapter 4, we first show how Rate Compatible Punctured Turbo (RCPT) codes work and how they are combined with ARQ scheme. The simulation results are discussed subsequently. The simulation results are divided into three parts. In the first part, we compare the RCPT codes based ARQ system with corresponding Rate Compatible Punctured Convolutional (RCPC) codes. In the second part, we discuss the effects of different puncturing schemes in our system. In the last part, we show the simulation results of two low rate solutions, simple repetition and multiple component encoding for the RCPT-based hybrid type II ARQ schemes.

Finally, Chapter 5 gives a summary of this thesis work.

Chapter 2

Turbo Codes

2.1 Introduction

Turbo codes are very powerful channel codes which were developed by Berrou, Glavieux and Thitimajshima in 1993[1]. The general encoder uses at least two, identical or different, rate $1/k$ convolutional encoders in systematic feedback form in a parallel or serial concatenation configuration. The decoder consists of a number of component decoders corresponding to the convolutional encoders in turbo encoder. Each component decoder is a soft-output decoder, generating a-posteriori probabilities (APP). The component decoders operate independently, and feed their own soft-output information to the other component decoder.

In the following sections, we will review how turbo encoder and decoder work.

2.2 Turbo Encoder

Turbo codes are the parallel or serial concatenation of two or more error control codes. A basic turbo encoder is shown in figure 2.1.

In this figure, a data block u of N bits enters the turbo encoder. This N-bit data block is also fed into a recursive systematic convolutional (RSC) encoder and a parallel interleaver followed by another RSC encoder. The interleaver scrambles the systematic bit streams in a pseudo-random fashion. Each of the systematic convolutional encoders produces a parity sequence (p_1, p_2). The two parity bit streams are multiplexed with the systematic bit streams and punctured to generate one large block code.

2.2.1 Recursive Systematic Convolutional (RSC) Encoder

The most common convolutional encoder is the nonrecursive nonsystematic convolutional encoder such as the one shown in figure 2.2a.

This common convolutional encoder can be made systematic without affecting its minimum distance properties by feeding back one of the outputs to the input. Such an encoder is called

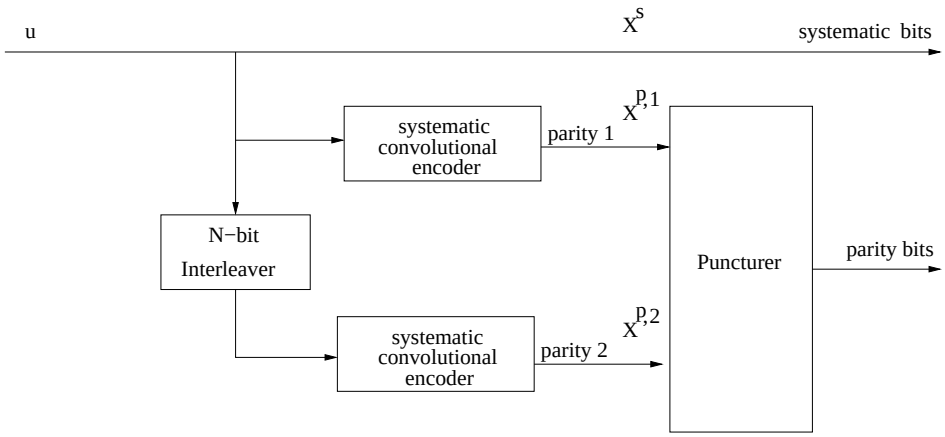
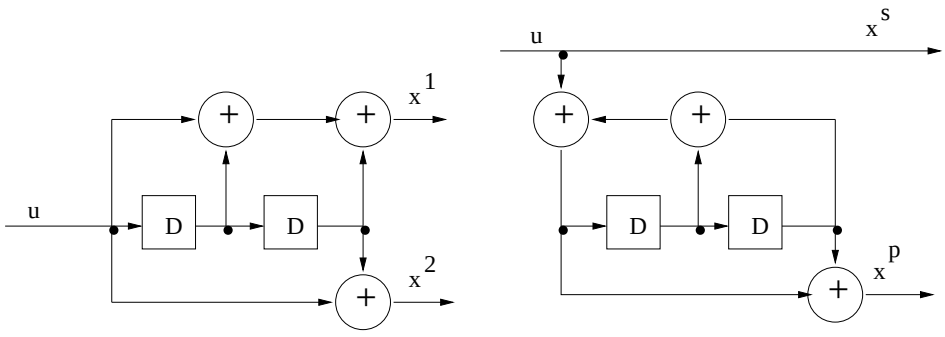


Figure 2.1: Turbo Encoder



a. Example convolutional encoder (K=3) b. Example Recursive Systematic Convolutional encoder (K=3)

Figure 2.2: Example Convolutional Encoder ($k = 3$)

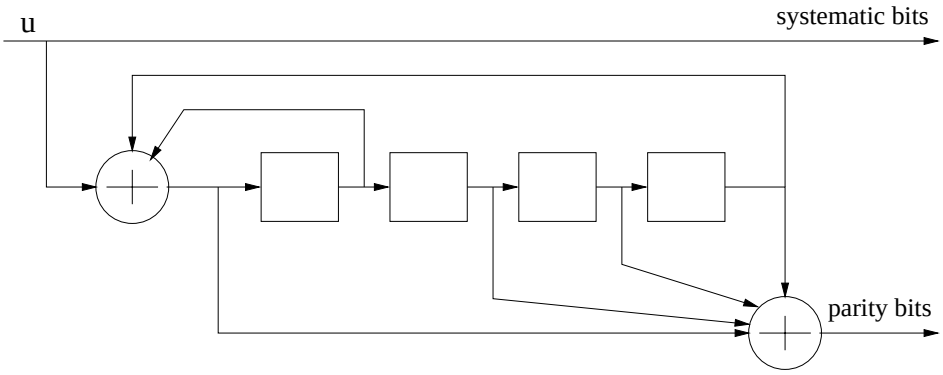


Figure 2.3: Recursive systematic convolutional encoder for code generators $(g_1, g_2) = (31, 27)$

a Recursive Systematic Convolutional (RSC) encoder. An example RSC encoder shown in figure 2.2b can be derived from the nonsystematic encoder of figure 2.2a.

The RSC encoder of figure 2.3 can be represented as $G = [1, g_2/g_1]$. Here, the first output represented by g_1 is fed back to the input. Hence, for this RSC encoder, g_1 and g_2 are referred as the feedback and the feed together forward polynomials respectively.

2.2.2 Interleaver

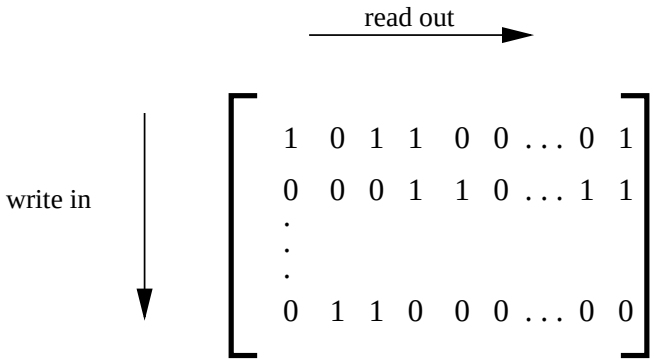
The interleaver is one of the important components of the turbo encoder. The interleaver improves the global Hamming distance of the turbo code by preventing short merges in both constituent trellis. These short merges are usually produced by short, low-weight input data streams and result in low output parity weight. The interleaver used in the turbo codes scrambles these bad input data streams in a pseudo-random fashion to increase the weights of the codes which are encoded by RSC encoder.

Because the interleaver affects the distance property of the code directly, the performance of the turbo codes will be affected by using different kinds of interleavers, such as block interleaver, pseudo random interleaver and so on.

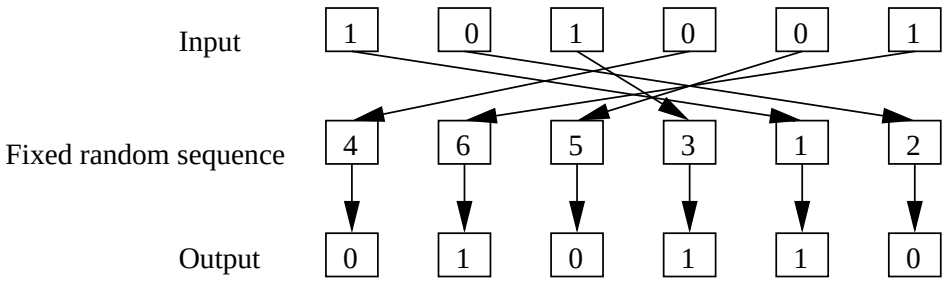
A block interleaver is the most commonly used interleaver in communication systems. Its basic idea is to write in column from top to bottom and left to right, and to read out in row from left to right and top to bottom. We can see it in figure 2.4(a). The pseudo random interleaver uses a fixed random sequence and maps the input streams according to this random sequence. This kind of interleaver is shown in figure 2.4(b).

2.2.3 Puncturer

Without puncturing any bits, the parent turbo code rate should be equal to 1/3 as the turbo encoder in figure 2.1. In order to get higher code rate, different puncturing matrices are needed. In chapter 4, puncturing matrices will be introduced in more detail.



(a) Block Interleaver



(b) Pseudo−Random Interleaver

Figure 2.4: Block and Random Interleaver

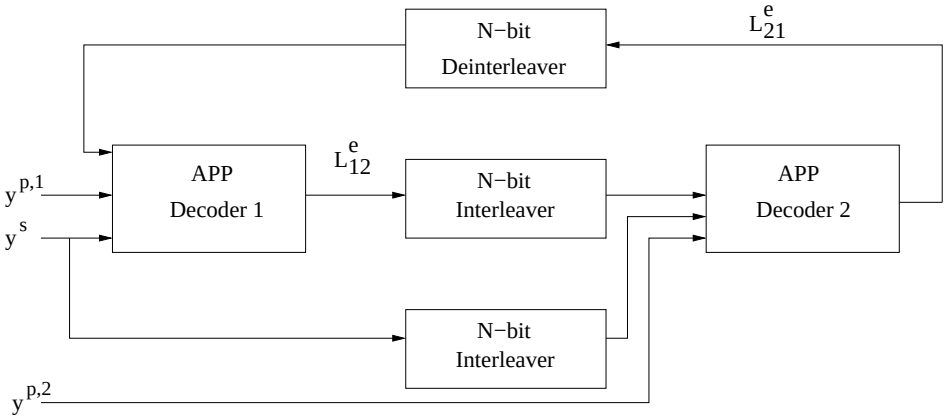


Figure 2.5: Block Diagram of Turbo Decoder

2.3 Turbo Decoder

In principle, a turbo code can be decoded using the maximum-likelihood (ML) principle. However, for codes with large data length N, the computational cost to compare 2^N different codewords is prohibitively high. Due to the presence of the interleaver, the Viterbi algorithm would be extensively complex even though there exists trellis for each RSC. Fortunately, Soft-In/Soft-Out decoding algorithm like BCJR[1] can be used to solve this problem. The idea is to let the component decoders exchange the reliability information for each bit. Although this solution is sub-optimal, the performance and the complexity of this decoding algorithm are acceptable.

2.3.1 Turbo Decoding Principle

The structure of the turbo decoder for two component decoders is shown in figure 2.5. The received noisy signal is split to create separate streams of data for systematic bits $\mathbf{y}^s$, the parity bits $\mathbf{y}^{p,1}$, created by the RSC 1 of the turbo encoder and the parity bits $\mathbf{y}^{p,2}$, created by the RSC 2 of that encoder. The systematic bits $\mathbf{y}^s$ and the parity bits $\mathbf{y}^{p,1}$ are fed into the first a-posteriori probability decoder (APP Decoder 1), creating extrinsic information for each bit. An interleaved version of extrinsic information from the APP decoder 1 with the parity bits $\mathbf{y}^{p,2}$ and an interleaved version of the received systematic bits $\mathbf{y}^s$ are fed into the APP decoder 2 creating another version of extrinsic information for each bit. The de-interleavered extrinsic information from APP decoder 2 will be fed into APP decoder 1 as a-priori information. The iterations will continue until the decoder converges to a stable solution.

2.3.2 APP Decoder

The two a-posteriori probability (APP) decoders are the main components of the turbo decoder. The object of APP decoder is to produce the log-likelihood value $L(\hat{u}_k|\mathbf{y})$ for each bit based on the received samples and the extrinsic information (a-priori information). The

log-likelihood value is defined as

$$L(\hat{u}_k|\mathbf{y}) = \log\left(\frac{Pr(u_k=+1|y)}{Pr(u_k=-1|y)}\right) \quad (2.1)$$

$$= \log\left(\frac{Pr(y|u_k=+1)}{Pr(y|u_k=-1)}\right) + \log\left(\frac{Pr(u_k=+1)}{Pr(u_k=-1)}\right) \quad (2.2)$$

The second item above representing a-priori information, can be denoted as

$$L^e(u_k) = \log\left(\frac{Pr(u_k=+1)}{Pr(u_k=-1)}\right) \quad (2.3)$$

In turbo decoder, such a-priori information is received from the other APP decoder and is used to estimate the priori probability of each bit by the following relation

$$Pr(u_k) = A_k \exp(u_k L^e(u_k)/2) \quad (2.4)$$

This relation will be used later for representing $L(\hat{u}_k|\mathbf{y})$.

If we take the code trellis into account, equation 2.1 can be written as

$$L(\hat{u}_k|\mathbf{y}) = \log\left(\frac{\sum_{s^+} Pr(s_{k-1}=s', s_k=s, y)/Pr(y)}{\sum_{s^-} Pr(s_{k-1}=s', s_k=s, y)/Pr(y)}\right) \quad (2.5)$$

where $s_k \in S$ is the state of encoder at time k, S^+ is the set of ordered pairs (s', s) corresponding to all state transitions $(s_{k-1}=s') \rightarrow (s_k=s)$ caused by input data $u_k=+1$. S^- is defined similarly for $u_k=-1$.

As proposed in BCJR algorithm, $Pr(s_{k-1}=s', s_k=s, \mathbf{y})$ can be factored as

$$Pr(s_{k-1}=s', s_k=s, \mathbf{y}) = Pr(s_{k-1}=s', \mathbf{y}_{j<k}) \cdot Pr(s_k=s, y_k|s_{k-1}=s') \cdot Pr(\mathbf{y}_{j>k}|s_k=s) \quad (2.6)$$

where $\mathbf{y}_{j<k} = (y_1, y_2, \cdots, y_{k-1})$ and $\mathbf{y}_{j>k} = (y_{k+1}, y_{k+2}, \cdots, y_N)$.

To simplify the notation, we denote

$$\alpha_k(s') = Pr(s_{k-1}=s', \mathbf{y}_{j<k}) \quad (2.7)$$

$$\gamma_k(s', s) = Pr(s_k=s, y_k|s_{k-1}=s') \quad (2.8)$$

$$\beta_k(s) = Pr(\mathbf{y}_{j>k}|s_k=s) \quad (2.9)$$

Now, we show how to calculate $\gamma_k(s', s)$, $\alpha_k(s')$ and $\beta_k(s)$.

Notice that $\gamma_k(s', s)$ can be expanded further as

$$\gamma_k(s', s) = Pr(s_k=s, y_k|s_{k-1}=s') \quad (2.10)$$

$$= Pr(y_k|s_k=s, s_{k-1}=s') \cdot Pr(s_k=s|s_{k-1}=s') \quad (2.11)$$

$$= Pr(y_k|u_k) \cdot Pr(u_k) \quad (2.12)$$

Previously, we have found $Pr(u_k)$ in terms of extrinsic information $L^e(u_k)$. This is where iterative decoding can be used. In the Rayleigh fading channel, the $Pr(y_k|u_k)$ can be written as

$$Pr(y_k|u_k) = B_k \exp\left(\frac{a_k^s y_k^s u_k + a_k^p y_k^p x_k^p}{\sigma^2}\right) \tag{2.13}$$

where B_k is a factor that will be cancelled out in the numerator and the denominator in the expression $L(\hat{u}_k|\mathbf{y})$, u_k and x_k^p are the systematic part and the parity part of k-th transmitted information bit x_k respectively. y_k^s and y_k^p are noisy versions of x_k^s and x_k^p. The σ^2 is the variance of the white Gaussian noise and a_k is the momentary fading amplitude. In the case of AWGN channel, a_k equals one.

Combining equations (2.12) and (2.13), we obtain

$$\gamma_k(s', s) \sim \exp(\frac{1}{2}(L^e(u_k) + L_c y_k^s)u_k) \cdot \gamma_k^e(s', s) \tag{2.14}$$

where $L_c = \frac{4E_c}{N_0} \cdot a_k^s$ and $\gamma_k^e(s', s) = \exp(\frac{1}{2} L_c y_k^p x_k^p a_k^p)$

Now $\alpha_k(s')$ and $\beta_k(s)$ can be calculated recursively in terms of $\gamma_k(s', s)$,

$$\alpha_k(s) = \sum_{s'} \gamma_k(s', s) \cdot \alpha_{k-1}(s') \tag{2.15}$$

$$\beta_{k-1}(s') = \sum_{s} \gamma_k(s', s) \cdot \beta_k(s) \tag{2.16}$$

with initial conditions for the terminated trellis

$$\alpha_0(0) = 1 \quad \text{and} \quad \alpha_0(s \neq 0) = 0 \tag{2.17}$$

$$\beta_N(0) = 1 \quad \text{and} \quad \beta_N(s \neq 0) = 0 \tag{2.18}$$

Finally, we have the expression for the log-likelihood ratio of each received bit

$$L(u_k|\mathbf{y}) = L_c \cdot y_k^s \cdot a_k + L^e(u_k) + \log(\frac{\sum_{S^+} \alpha_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \beta_k(s)}{\sum_{S^-} \alpha_{k-1}(s') \cdot \gamma_k^e(s', s) \cdot \beta_k(s)}) \tag{2.19}$$

The first term above is sometimes called channel value, the second term represents a-priori information about u_k provided by the other decoder and the third term represents extrinsic information that can be passed to a subsequent APP decoder.

When completing the number of iterations, the decoding decision can be made as follows:

$$\hat{u}_k = +1 \quad \text{if} \quad L(u_k|\mathbf{y}) > 0 \tag{2.20}$$

$$\hat{u}_k = -1 \quad \text{if} \quad L(u_k|\mathbf{y}) < 0 \tag{2.21}$$

Chapter 3

ARQ Schemes

Besides Forward Error Correction (FEC) codes, there exist other strategies for combating the transmission errors. One of them is called Automatic Repeat re-Quest (ARQ). Unlike FEC schemes, ARQ schemes take the advantage of two way communication links. Based on their complexities, ARQ schemes can be classified into two categories: the first one is simple ARQ scheme and the second one is ARQ-FEC hybrid scheme. Stop-and-Wait ARQ, Go-back-N ARQ and Selective-Repeat ARQ can be used with both simple ARQ and hybrid ARQ schemes. The ARQ-FEC hybrid scheme combines FEC strategy with basic ARQ schemes. There are a lot of variants here. We give a brief overview of hybrid type I and II ARQ schems in this chapter. In chapter 4, we will investigate a Turbo Codes combined with ARQ scheme in more detail.

3.1 Simple ARQ Schemes

There are three basic types of ARQ schemes: the stop-and-wait ARQ, the go-back-N ARQ, and the selective-repeat ARQ.

The stop-and-wait ARQ system is the simplest of these three schemes. The transmitter sends a codeword to the receiver and waits for an acknowledgment from the receiver. A positive acknowledgment (ACK) means the codeword is received successfully. A negative acknowledgment (NAK) means the errors are detected by the receiver. When the transmitter gets NAK, it will resend the previous codeword to the receiver. The transmitter will not send next message until it gets ACK signal back from receiver. We also illustrate the procedure in figure 3.1.

For the go-back-N ARQ system, the difference from the stop-and-wait ARQ system is that the codewords are transmitted continuously. We can see it in figure 3.2. When transmitter has finished sending one codeword, it starts sending the next codeword without waiting for an acknowledgment from receiver. After a round-trip delay, the transmitter will receive an ACK or NAK. If NAK is received, the transmitter will go back to the codeword that caused NAK and resend it and the N-1 succeeding codewords that were sent during the round-trip delay. This kind of ARQ scheme is more effective than the stop-and-wait ARQ scheme due to continuous transmission and retransmission. However, for large round-trip delay and high data

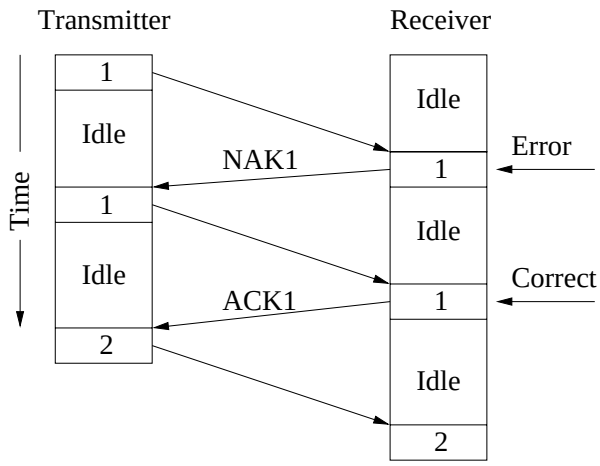


Figure 3.1: Stop and Wait ARQ Protocol

transmission rate, it becomes inefficient because of retransmitting many error-free codewords following a detected erroneous codeword.

Similarly to Go-back-N ARQ scheme, the Selective-Repeat ARQ scheme shown in figure 3.3 uses continuous transmission. In the Selective-Repeat ARQ scheme, the transmitter only resends those codewords which get NAK, without retransmitting other codewords. In order to get the codewords in correct order for the user, both the transmitter and receiver need a buffer to save the error-free codewords. If the buffer size is not sufficiently large, the buffer overflow will occur and data will be lost. The Selective-Repeat ARQ is the most efficient ARQ scheme among the three basic ARQ schemes in terms of throughput.

3.2 Hybrid ARQ Schemes

In pure FEC based systems, receiver tries its best to correct the errors that are detected in the received codeword and sends the decoded message to data sink regardless whether it is correct or not. In order to achieve low bit error rate, such a system needs to use powerful codes to correct a large collection of error patterns. Using powerful codes means usually more redundancy and more complex decoding. The simple ARQ systems also have a severe drawback: the throughput will fall rapidly when channel error rate increases [4]. In order to establish a reliable communication link with less redundancy in codes, less decoding complexity and higher throughput, we can combine ARQ with FEC together when the feedback channel is available. Such a combination is referred as hybrid ARQ scheme.

We will discuss two types of hybrid ARQ schemes here: type I hybrid ARQ scheme and type II hybrid ARQ scheme.

3.2.1 Type I Hybrid ARQ Scheme

Type I hybrid ARQ scheme uses a linear code, which can detect and correct errors simultaneously. When an error in the received code word is detected, the receiver will first locate and

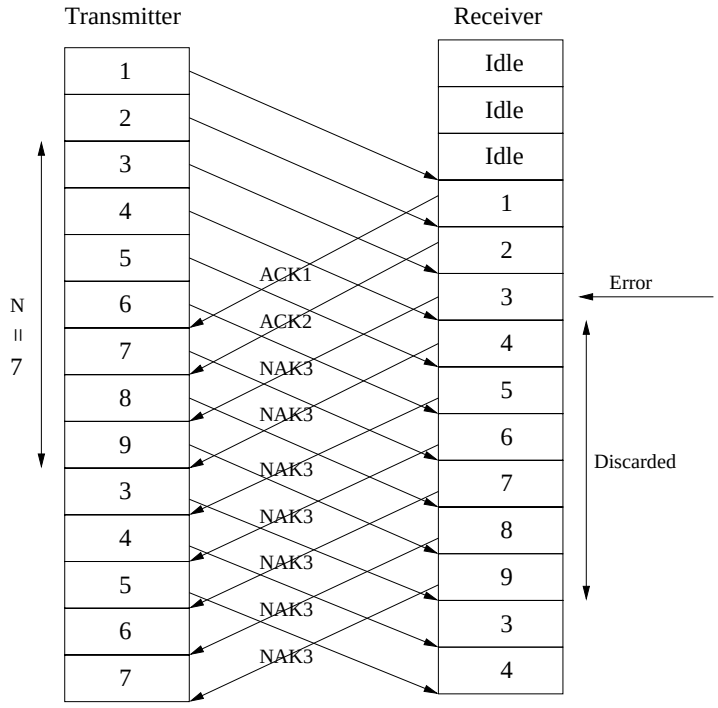


Figure 3.2: Go Back N ARQ Protocol

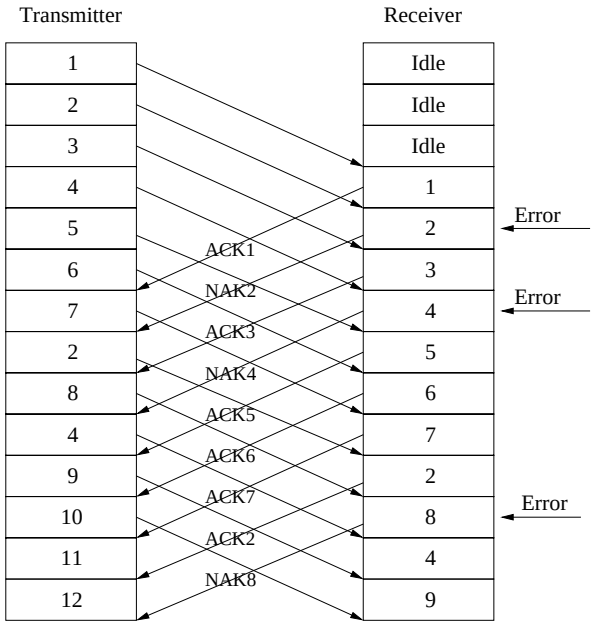


Figure 3.3: Selective Repeat ARQ Protocol

correct errors. If the number of errors is within the error-correcting capability of the linear code, they will be corrected and sent to the data sink. If the error pattern cannot be detected, the receiver will reject it and send NAK to the transmitter. If the retransmitted codeword is received, the receiver will repeat the steps as the above until the receiver receives and decodes the codeword successfully. The type I hybrid ARQ scheme is able to maintain significantly higher throughput over a wide range of channel error rates when the FEC code is capable of correcting most errors.

3.2.2 Type II Hybrid ARQ Scheme

The disadvantage of the type I hybrid ARQ is that the overhead due to the extra parity check bits for error correction must be included in each transmission or retransmission regardless of the channel error rate. In an extreme case when the channel is noise free, those parity check bits are wasted. The Hybrid type II ARQ scheme is designed to avoid such a situation. The idea is to make use of a set of FEC codes from high rates to lower rates. The lower rate codes are derived from higher rate codes.

In a case of the classical hybrid type II ARQ scheme, two linear codes are used: one is a higher rate (n, k) code C_0 which is designed for error detection only and the other is an invertible half rate $(2k, k)$ code C_1 which is designed for simultaneous error correction and detection. In the invertible codes, the information bits can be uniquely determined even if only the parity check digits of a code is known. When a transmission is initiated, the transmitter sends C_0. If the receiver finds no error after error detection, the k bits message is received successfully. Otherwise, when errors are detected in C_0, the transmitter sends the parity bits of code C_1. The parity bits will also be encoded by codes C_0. The receiver will then perform the error detection on these C_0 coded parity bits. If successful, these parity bits will be inverted to message bits, otherwise the k parity bits will be combined with k information bits in the previous transmission to form code C_1 for error correction. If error correction fails, the transmitter will start from sending code C_0 for message bits again.

The type II hybrid ARQ scheme is particularly attractive for high speed data communication systems where round-trip delay is large and error rate is non-stationary. One example is the satellite communication systems.

Chapter 4

Turbo Codes for Type II Hybrid ARQ Transmission

4.1 Rate Compatible Punctured Turbo (RCPT) Codes

A RCPT encoder consists of a common turbo encoder and a puncturing device as in figure 4.1a. The decoder for RCPT codes is the same as an ordinary turbo decoder. A block of message with block length N is fed into the turbo encoder with $m \cdot N$ bits output, where $m-1$ is the number of RSC encoder in turbo encoder. The $m \cdot N$ turbo coded bits will be fed into the puncturing device in order to get desired coded rates instead of $1/m$.

In figure 4.1a, the parent turbo code rate is $1/m$ and each RSC encoder has $1/2$ code rate. One example of such a RSC encoder is shown in figure 4.1b. Notice that the systematic bits of all RSC encoders are discarded. The resulting $m-1$ parity bit streams and one systematic bit stream are fed into each subblock by column. Each subblock has P rows and N/P columns. We call P the puncturing period in the sense that every P bits of systematic bit stream or parity bit streams will be punctured with the same pattern.

Similar to the Rate Compatible Punctured Convolutional (RCPC) codes, a set of code rates for the RCPT codes can be represented by:

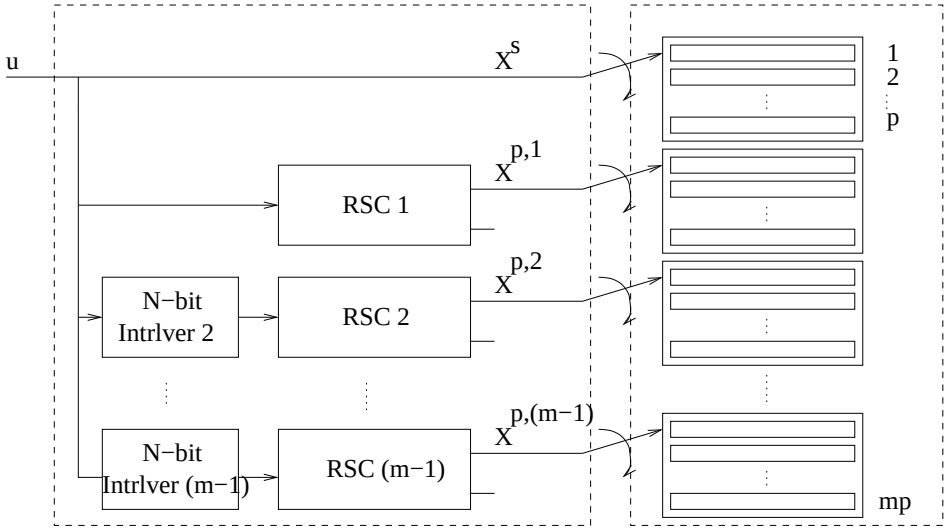
$$R_l = \frac{P}{P+l}, l = 0, 1, ..., (m-1)P \tag{4.1}$$

Notice that $\frac{1}{m} \leq R_l \leq 1$.

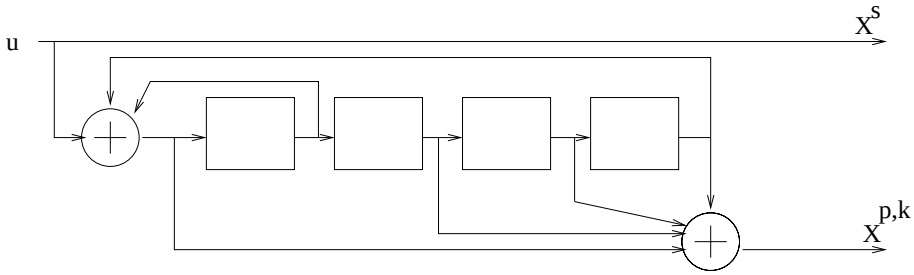
For every code rate R_l, we can use a binary matrix denoted as $a(l)$ of size $m \times P$ for representing the puncturing pattern. If $a_{ij}(l) = 0$, the output of i-th RSC parity bit (if $i > 0$) or systematic bit (if $i = 0$) at the position $k \cdot P + j$, for $k = \{0, 1, \cdots (N-1)/P\}$, will be punctured. Otherwise, it will be kept.

If $l = 0$, $R(l) = 1$ and there must exist P ones in the puncturing matrix $a(0)$. If we want to construct a family of RCPT codes with $m = 3$, $P = 8$ and two $1/2$ code rate RSC encoders, the RCPT code rate can be selected from this set

$$\left\{1, \frac{8}{9}, \frac{4}{5}, \frac{8}{11}, \frac{2}{3}, \frac{8}{13}, \frac{4}{7}, \frac{8}{15}, \frac{1}{2}, \frac{8}{17}, \frac{4}{9}, \frac{8}{19}, \frac{2}{5}, \frac{8}{21}, \frac{4}{11}, \frac{8}{23}, \frac{1}{3}\right\}$$



a. RCPT encoder



b. Recursive systematic convolutional encoder for code generators (g1,g2)=(31,27)

Figure 4.1: RCPT Encoder

As an example, one possible set of puncturing matrices is given as :

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdots \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

In the puncturing matrices above, the first row of each matrix is for systematic bit stream. The second and third rows are for the first and second encoders' parity bit streams.

As can be seen from the first puncturing matrix $a(0)$, all systematic bits are sent to the receiver but not the parity bits. In the second puncturing matrix $a(1)$, besides all systematic bits, one of every eight parity bits from the first RSC encoder is sent to receiver.

4.2 RCPT Codes in Hybrid Type II ARQ Schemes

In this communication system, we assume a selective-repeat ARQ strategy with infinite memory based on RCPT codes.

We choose the $m = 3$, $P = 8$ RCPT codes, and encoder A, B, and C provided in [5]

The RCPT codes based hybrid FEC/ARQ scheme works as follows:

1. Encode an information block of k bits based on (n, k) block error detection code (CRC), so we get a packet of n bits.
2. Input this packet n bits into the parent code rate 1/3 turbo encoder, then through the $m = 3$, $P = 8$ puncturing matrices.
3. Initialize $l = 0$.
4. Transmitter sends the encoded punctured packet corresponding to $a(l)$. This packet does not contain the bits that have been transmitted corresponding to $a(l-1)$ in the last transmission for the same information block.
5. If $l \neq 0$, the received packet needs to be combined with the previous received packet. At the same time, zeros need to be added into the punctured position of the received packet according to the puncturing matrix $a(l)$. The combined packets will be fed into turbo decoder.
6. The output of turbo decoder will be n bits. These n bits will be fed into CRC decoder. If no error is found, k bits information will be sent to data sink, ACK will be sent to the transmitter and the transmitter will proceed with step 1. Otherwise, when CRC decoder fails, NAK will be sent to the transmitter and the transmitter will start from step 4 with $l = l + 1$.

Table 4.1: Puncturing Table (In Octal) for Encoder A, RCPT Code: M = 3, Period = 8 and Block Length = 512

Puncturing Code Rate	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2
Optimal Puncturing Matric	376	377	377	377	377	377	377	377
	002	002	002	006	016	016	016	056
	002	002	006	006	006	016	036	036
Puncturing Code Rate	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
Optimal Puncturing Matric	377	377	377	377	377	377	377	377
	076	076	076	176	177	177	377	377
	036	037	137	137	137	177	177	377

Table 4.2: Puncturing Table (In Octal) for Encoder B ,RCPT Code: M = 3, Period = 8 and Block Length = 512

Puncturing Code Rate	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2
Optimal Puncturing Matric	376	377	377	377	377	377	377	377
	001	001	001	041	051	051	055	055
	001	001	041	041	041	045	045	245
Puncturing Code Rate	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
Optimal Puncturing Matric	377	377	377	377	377	377	377	377
	055	155	155	175	375	375	377	377
	265	265	275	275	275	375	375	377

4.3 Simulation Results

We examined three turbo codes by using constraint length $\nu = 5$ encoders proposed in [5]. Three optimal puncturing matrices are also suggested in [5] corresponding to the different encoders. They are shown in the following:

Encoder A: Rate 1/3; generators: $(1, 21/37)_{octal} + (21/37)_{octal}$.

Encoder B: Rate 1/3; generators: $(1, 23/35)_{octal} + (23/35)_{octal}$.

Encoder C: Rate 1/3; generators: $(1, 33/31)_{octal} + (33/31)_{octal}$.

We used simulation to estimate the throughput performance of the three different encoders using corresponding optimal puncturing matrices. The throughput is defined as

$$\eta = \frac{1}{T}(\frac{k}{n})$$

Table 4.3: Puncturing Table (In Octal) for Encoder C,RCPT Code: M = 3, Period = 8 and Block Length = 512

Puncturing Code Rate	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2
Optimal Puncturing Matric	376	377	377	377	377	377	377	377
	001	001	011	011	013	013	213	213
	001	001	001	011	011	013	013	113
Puncturing Code Rate	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
Optimal Puncturing Matric	377	377	377	377	377	377	377	377
	253	253	253	353	373	373	377	377
	113	133	173	173	173	177	177	377

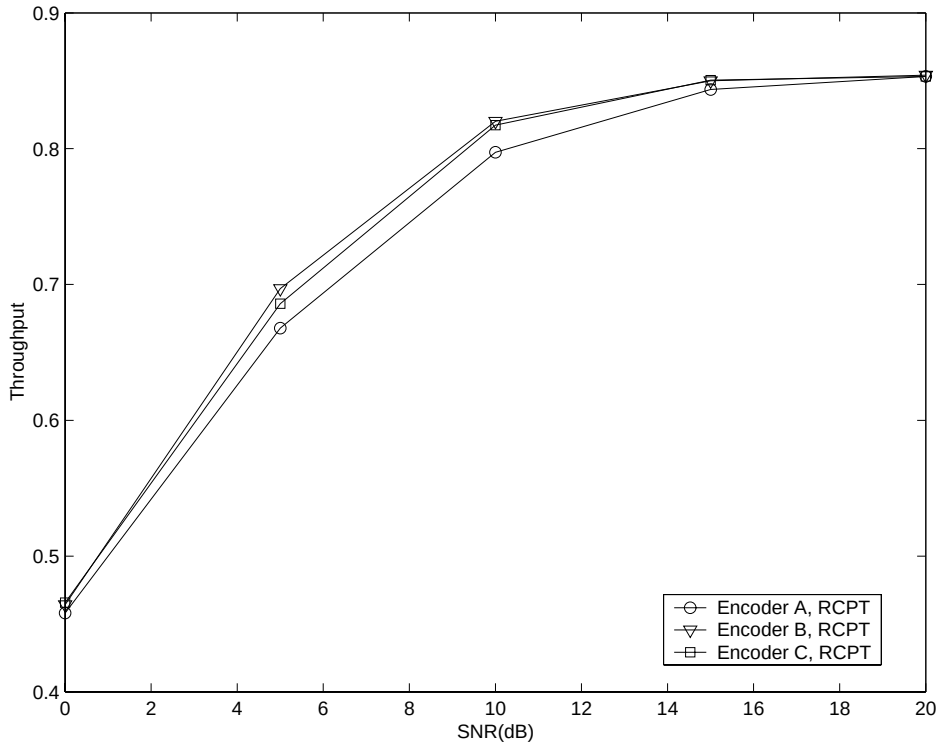


Figure 4.2: Simulated throughput for $N = 512$ and puncturing period $P = 8$. Codes A, B and C on the Rayleigh fading channel.

Where T is the average number of retransmissions (including the original transmission), k/n is the rate of the (n, k) code used in the system.

The figure 4.2 shows the result. As shown in the figure, systems using encoder B and C give almost identical throughput performance vs SNR. There is a little throughput loss when the encoder A based system is used.

4.3.1 Performance Comparison of Hybrid Type II ARQ Schemes based on RCPC and RCPT Codes

In this subsection, we will compare the performance of Rate Compatible Punctured Convolutional (RCPC) codes and RCPT codes based system. We are mainly concerned with factors such as the throughput and the number of retransmission.

For the RCPC codes, Hagenauer proposed using RCPC codes with Viterbi algorithm decoder for the successive parity transmission in [2]. The proposed ARQ/FEC scheme transmits additional code bits of a lower rate RCPC code until the code is powerful enough to decode the received message successfully. This scheme with the general Viterbi algorithm for decoding RCPC codes has been described in [3]. The scheme used in this project is similar to the scheme with RCPT codes in many aspects. After passing through a block error detection encoder, a transmitted packet is just $(k+n_p)$ bits, where k is the length of the information block, n_p is the number of added parity-check bits . For a parent code rate 1/3 tailbiting encoder, we obtain

Table 4.4: Puncturing Table (in Octal) for Convolutional Encoder,RCPC Code: M= 3, Period = 8 and Block Length =512

Puncturing Code Rate	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2
Puncturing Matrix	370	374	376	377	377	377	377	377
	0	0	0	0	0	0	0	0
	017	017	017	017	037	077	177	377
Puncturing Code Rate	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
Puncturing Matrix	377	377	377	377	377	377	377	377
	01	003	007	017	037	077	177	377
	377	377	377	377	377	377	377	377

Puncturing Code Rate	1/4	1/5	1/6
Puncturing Matrix	230	233	377
	105	165	377
	210	353	377

Table 4.5: Low rate puncturing matrices for RCPT code, M = 3, Period =8 and Block Length = 512

Puncturing Code Rate	1/4	1/5	1/6
Puncturing Matrix	377	000	000
	000	377	000
	000	000	377

a length of $3(k + n_p + n_t)$ codeword after the convolutional encoder, where n_t is the number of tail bits. The tail bits at the end of the encoded packet are used to force the encoder to go back to the initial state. This code word is first punctured by the highest rate puncturing matrix and then interleaved before it's delivered to the channel. After the received code word is deinterleaved, and decoded by the decoder using the general Viterbi algorithm. Then the parity checker computes the syndrome of the output from the general Viterbi decoder. If the syndrome is equal to zero, then the code word is accepted by the receiver and an ACK is sent to the transmitter. If the syndrome is not zero, a NAK is sent to the transmitter and then the additional code bits which were punctured initially according to the rate compatibility criterion are transmitted. If all the RCPC codes up to the parent rate 1/3 code are used and parity checking still fails, the whole procedure will repeat until decoded code passes parity checker.

For the ARQ/FEC scheme based on RCPC codes, the puncturing matrices used in this project are shown in Table 4.4. For the parent rate 1/3 convolutional encoder, we use encoder with generator polynomials $(25, 33, 37)_{octal}$ and constraint length $\nu = 5$ provided in [6, pp.222 Table 6.5.3]. The low rate puncturing matrices used here are from [7].

For the ARQ/FEC scheme based on RCPT codes, the puncturing matrices for Encoder C are shown in Table 4.3. Table 4.3 only gives the high rate from $\frac{8}{9}$ to $\frac{1}{3}$ puncturing matrices. The low rate puncturing matrices shown in the following table are obtained by simply repeating systematic bits or parity bits.

A packet is 508-bit long including 16 parity check bits. The throughput performance and retransmission comparison of our simulation results are given in Figure 4.3 and Figure 4.4, respectively. From the simulation we can see clearly that RCPT codes based system outperforms RCPC codes based system both in the throughput and the number of retransmission.

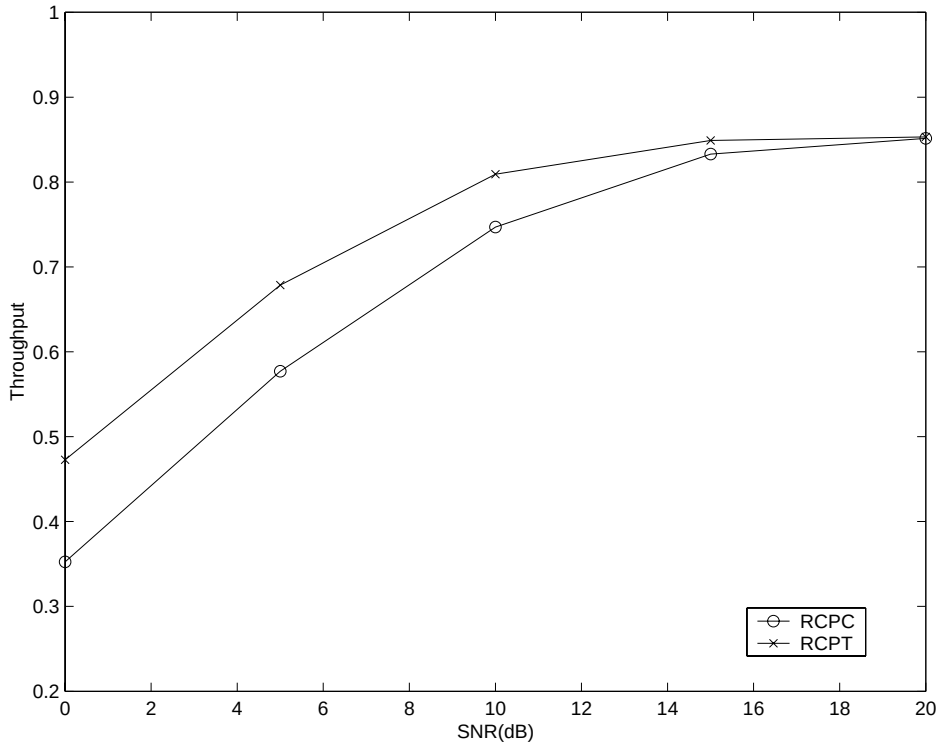


Figure 4.3: Simulation result for the throughput of RCPT and RCPC based systems, data packet 508 bits including 16 parity bits with normalized Doppler frequency 0.01. Constraint length $k = 5$.

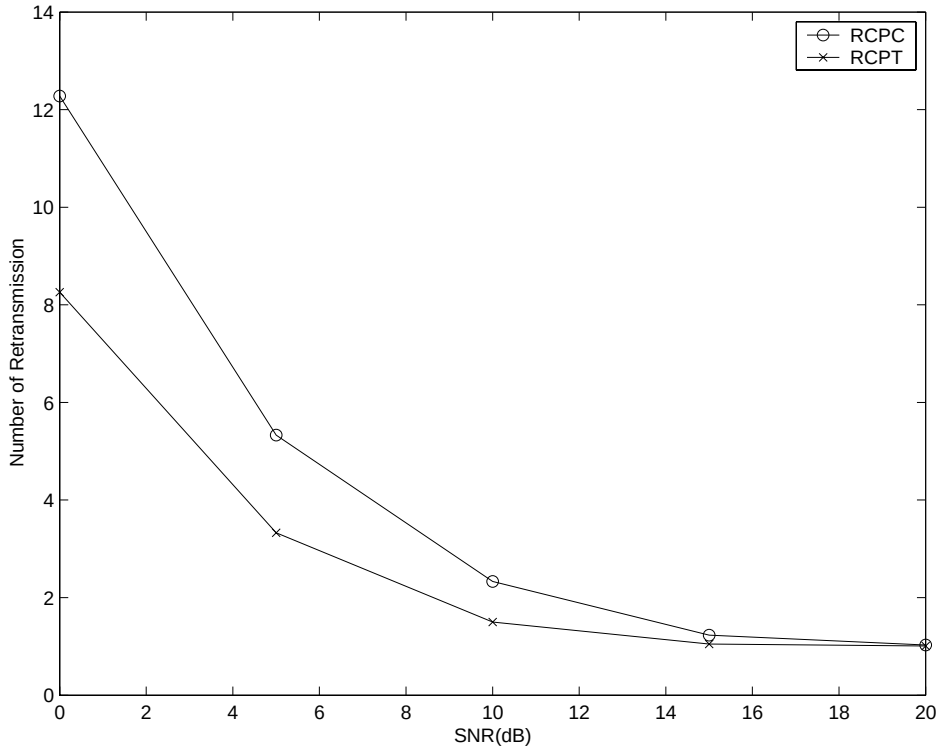


Figure 4.4: Simulation result for the number of retransmission of RCPC and RCPT based systems, data packet 508 bits including 16 parity check bits with normalized Doppler frequency 0.01, constraint length $k = 5$.

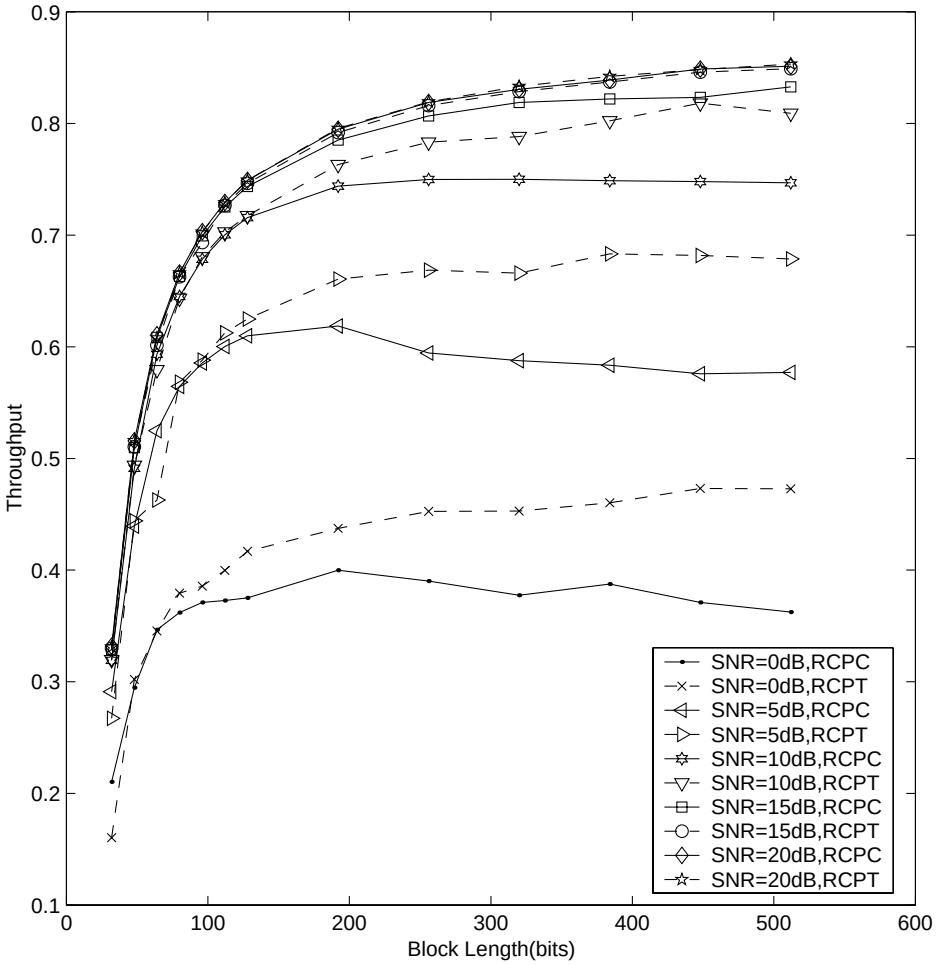


Figure 4.5: Simulation results of throughput for different lengths of blocks including 16 parity check bits with normalized Doppler frequency 0.01. Constraint length $k = 5$. Dashed line for RCPT codes, and solid line for RCPC codes.

We also compared the different data packet length with fixed 16 parity check bits for both systems. The simulated throughput performance and the average number of retransmission needed before the transmitter sending the ACK signal to the receiver are shown in Figure 4.5 and Figure 4.6, respectively. In these two figures, almost the same performance is obtained by both RCPC and RCPT codes at high SNRs. However, at low SNRs, RCPT codes provide superior performance compared to RCPC codes as the block size increases. But at a low SNR and with short block size, the RCPC codes are better than RCPT codes.

4.3.2 Effects of Random Puncture

One major factor that will affect the performance of RCPT based system is the puncturing scheme. We tried several different random puncturing matrices for our RCPT based system to compare with the system with optimal puncturing matrices presented by D.N. Rowitch[5]. In our simulation, we only tested eight random puncturing schemes to show the effects of random

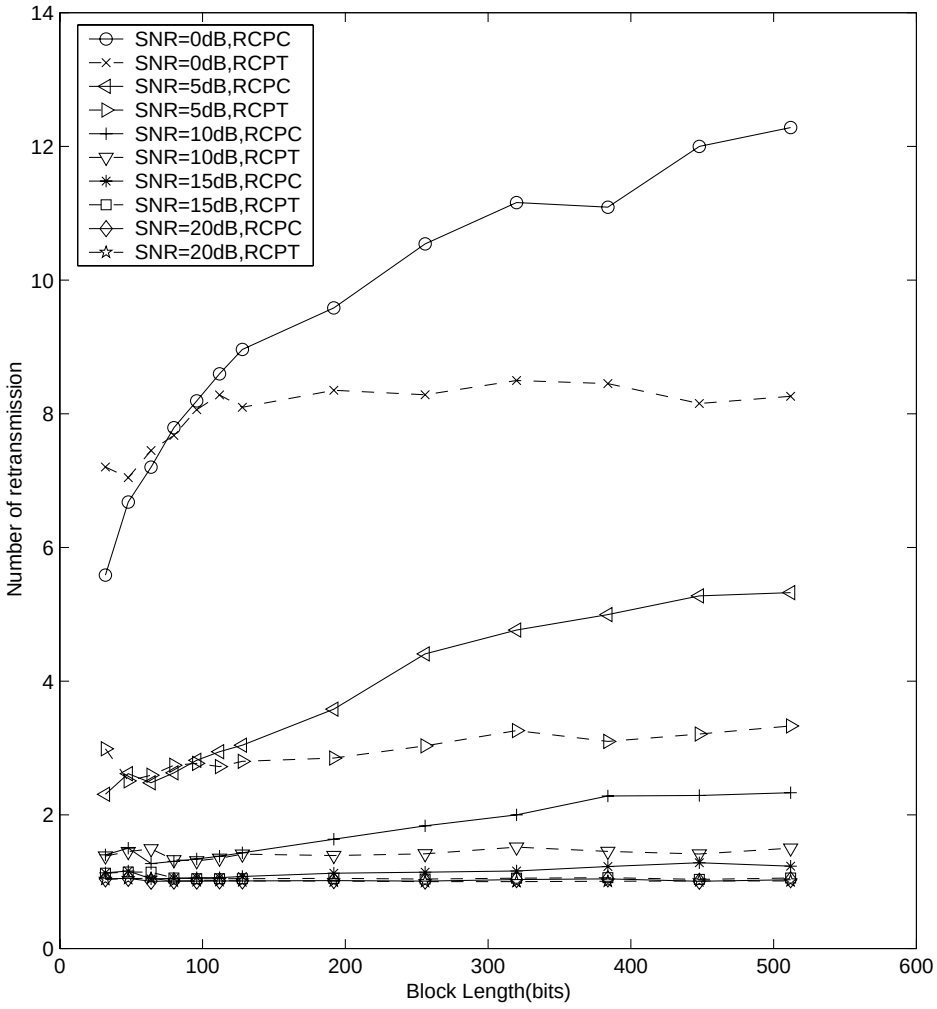


Figure 4.6: Simulation result of the average number retransmission for different lengths of blocks including 16 parity bits with normalized Doppler frequency 0.01. Constraint length $k = 5$. Dashed line for RCPT codes. Solid line for RCPC codes.

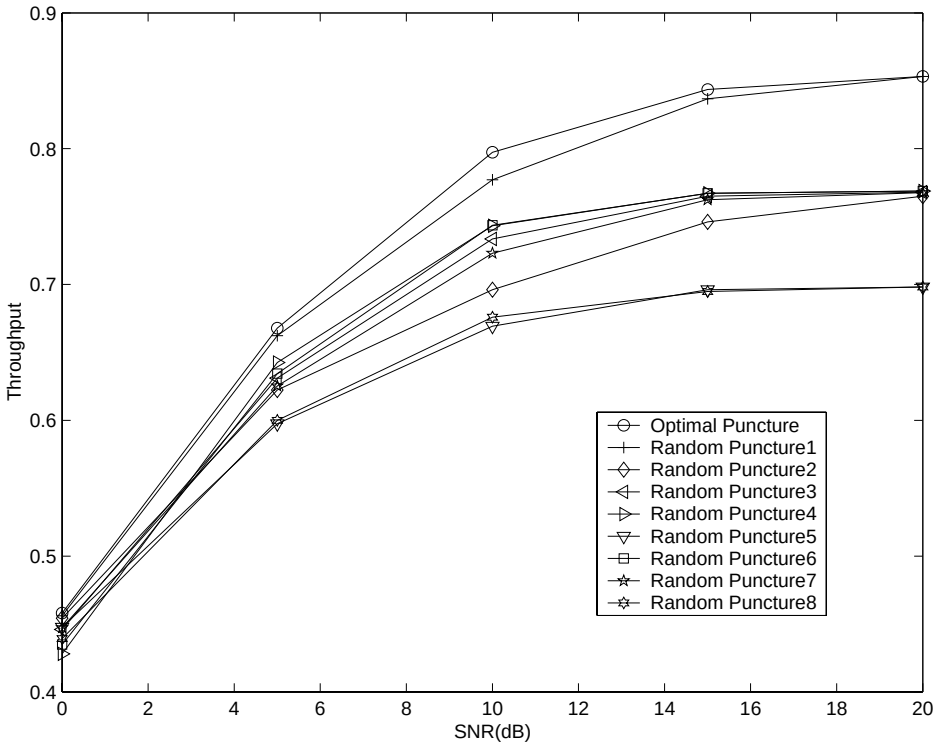


Figure 4.7: Simulation results of different random puncturing matrices and optimal one for 512 bits data block and normalized Doppler frequency 0.01 for encoder A

puncturing, although it cannot be used to prove anything. We also fix the puncturing period to 8 , and block length to $N = 512$ for three RCPT encoders A, B and C based system.

For Encoder A, B and C, their random puncturing matrices from 1 to 10 are the same. They are shown in the Table 4.6.

In the simulation, we estimate the throughput performance of system with these eight random puncturing matrices and we also compared the throughput performance of the system with the optimal puncturing scheme. The comparison are shown in figure 4.7 , 4.8 and 4.9. The systems using RCPT encoder B and C with optimal puncturing scheme show obvious thoughput gain over those with randomly chosen puncturing schemes. The performance gain is more obvious when channel SNR is high comparing to low SNR channel. This confirms the reliability of puncturing proposed by Rowitch.

In these three figures, we can see the throughput of the optimal puncturing schemes for encoder A, B and C are all better than the random puncturing schemes. A very interesting thing that can be found in these figures is that the scheme of the random puncturing 1 matrix show the better throughput than the others, just a little low than the optimal ones. By comparing the differences among these puncturing matrices, the puncturing matrices which give the higher throughput performance, puncture more nonsystematic bits of a turbo code and remain more systematic bits. This means that the systematic bits are more valuable to the decoder.

Table 4.6: Random Puncturing Table (in Octal) for Encoder A, B and C,RCPT Code: $M = 3$, Period = 8 and Block Length = 512

Puncturing Code Rate	8/9	4/5	8/11	2/3	8/13	4/7	8/15	1/2
Random Puncturing matrix1	035	035	035	075	077	177	177	177
	360	362	362	362	362	362	362	362
	200	200	210	210	210	210	212	312
Random Puncturing matrix 2	300	304	304	304	304	304	306	316
	072	072	172	176	176	177	177	177
	206	206	206	206	246	246	246	246
Random Puncturing matrix 3	114	114	114	114	114	114	314	314
	023	033	037	077	277	277	277	277
	203	203	203	203	203	207	207	307
Random Puncturing matrix 4	103	103	103	103	103	103	103	103
	170	171	175	177	177	177	377	377
	300	300	300	300	340	341	341	343
Random Puncturing matrix 5	202	202	202	202	202	206	216	236
	246	246	246	246	346	346	346	346
	103	113	117	157	157	157	157	157
Random Puncturing matrix 6	130	130	130	132	132	132	132	132
	203	203	203	203	203	303	323	323
	304	314	354	354	356	356	356	357
Random Puncturing matrix7	021	061	161	161	161	161	161	171
	034	034	034	234	334	334	374	374
	065	065	065	065	065	075	075	075
Random Puncturing matrix 8	100	110	110	112	112	312	332	332
	125	125	325	325	325	325	325	335
	341	341	341	341	343	343	343	343
Puncturing Code Rate	8/17	4/9	8/19	2/5	8/21	4/11	8/23	1/3
Random Puncturing matrix 1	377	377	377	377	377	377	377	377
	362	362	362	362	366	376	376	377
	312	332	333	337	337	337	377	377
Random Puncturing matrix 2	316	316	316	336	337	337	337	377
	177	377	377	377	377	377	377	377
	256	256	276	276	276	277	377	377
Random Puncturing matrix 3	314	334	334	374	374	376	377	377
	277	277	277	277	377	377	377	377
	347	347	367	367	367	367	367	377
Random Puncturing matrix 4	103	103	303	343	363	367	377	377
	377	377	377	377	377	377	377	377
	353	373	373	373	373	373	373	377
Random Puncturing matrix 5	236	236	236	336	337	377	377	377
	346	356	376	376	376	376	377	377
	177	177	177	177	177	177	177	377
Random Puncturing matrix 6	132	132	332	332	333	333	373	377
	323	333	333	373	373	377	377	377
	377	377	377	377	377	377	377	377
Random Puncturing matrix 7	371	371	375	373	375	375	375	377
	374	374	374	374	376	376	377	377
	075	077	077	277	277	377	377	377
Random Puncturing matrix 8	333	333	333	333	337	337	377	377
	335	335	337	337	337	377	377	377
	343	363	363	373	373	373	373	377

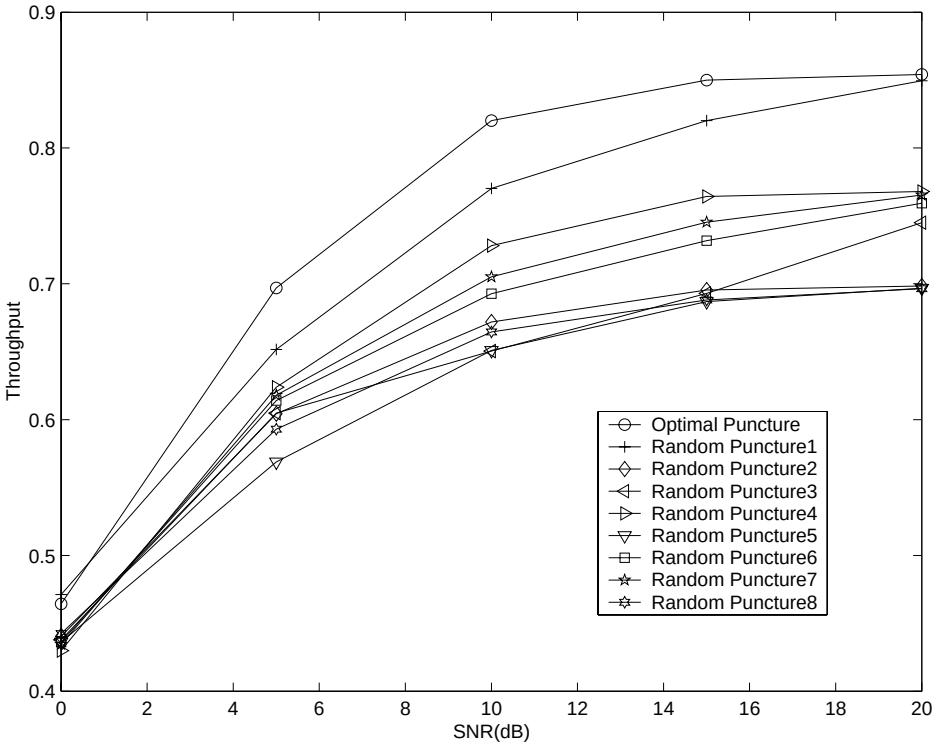


Figure 4.8: Simulation results of different random puncturing matrices and optimal one for 512 bits data block and normalized Doppler frequency 0.01 for encoder B

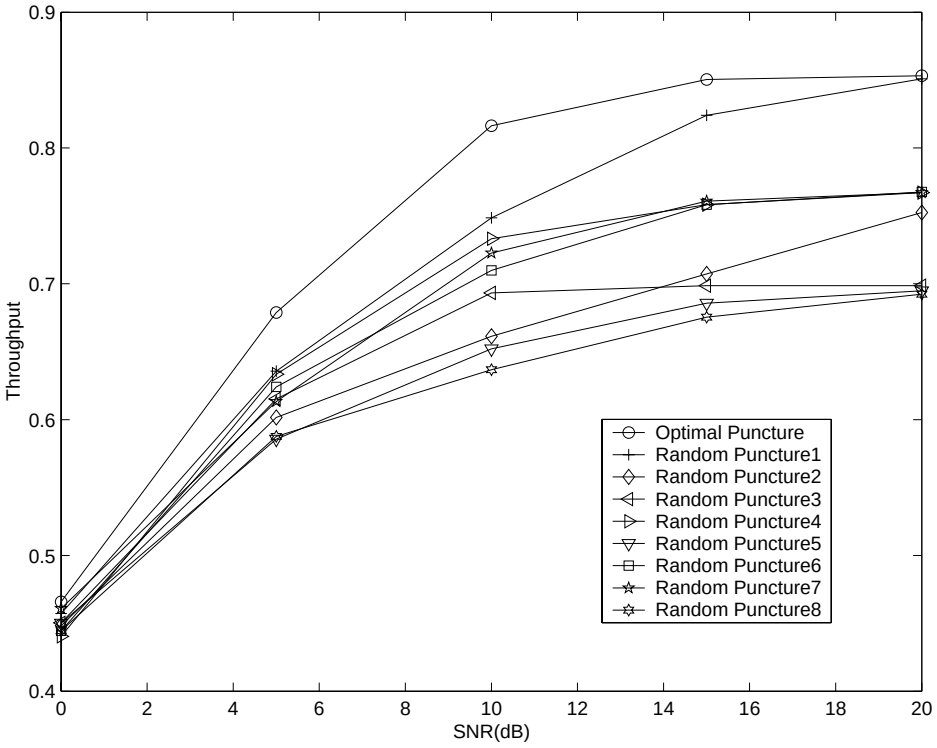


Figure 4.9: Simulation results of different random puncturing matrices and optimal one for 512 bits data block and normalized Doppler frequency 0.01 for encoder C

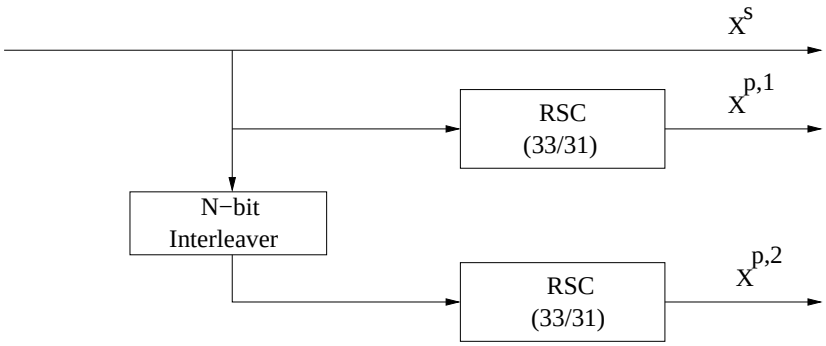


Figure 4.10: Simple Repetition (parent code rate = 1/3)

4.3.3 Low Rate Scheme and Performance

For the high rate schemes, we use different puncturing matrices. In order to get the low rate, there exist generally three possible solutions: using repetition coding , using multiple component turbo encoders and using multidimensional turbo codes. In our project, we investigated the first two solutions.

Given a parent rate 1/3 turbo encoder, the low code rate can only be chosen from $\{\frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}\}$ (if we use simple repetition) which is shown in figure 4.10. We encode the input bit block starting with the parent code rate 1/3. The systematic bit stream x^s, two parity-bit streams $x^{p,1}$ and $x^{p,2}$ are sent without puncturing. In the receiver side, the output of the RCPT decoder will also be fed into the CRC decoder. If CRC decoder fails, it will send NAK to the transmitter. The transmitter will then resend systematic bit stream x^s. This time the code rate is 1/4. After decoding, followed by combining of the received code word and previous received code words, the CRC decoder checks the output of the RCPT decoder again. If it fails again, the transmitter resends parity-bit stream $x^{p,1}$ again. Decoding procedure is the same as before. If it still fails, the transmitter will resend another parity-bit stream $x^{p,2}$. In the case of CRC decoder still failing, the transmitter will start from resending systematic bit stream x^s again.

In the multiple component turbo codes based scheme shown in figure 4.11, each RSC encoder will output two parity-bit streams, one for input message bit stream and the other for interleaved input bits. As an example, the three generator polynomials used in our simulation are $(1, 23/33, 23/25, 23/37)_{octal}$. By appropriate puncturing, the final code rate can range from 1/3 to 1/6 $(1/3, 1/4, 1/5, 1/6)$, given a parent rate 1/7. When starting to transmit a message, it is encoded and punctured to code rate 1/3 which means the systematic-bit stream x^s, two parity-bit streams $x^{p,11}$ and $x^{p,21}$ corresponding to generators $(23/33)_{octal}$ are sent. In the receiver side, if the CRC decoder reports failure, a NAK is sent to the transmitter and then the additional parity-bit stream $x^{p,13}$ corresponding to generators $(23/37)_{octal}$ are transmitted. If the CRC decoder still reports failure, an additional parity-bit stream $x^{p,23}$ which is also corresponding to generator $(23/37)_{octal}$ will be sent. If the received streams still cannot be decoded successfully at code rate 1/5, another parity-bit stream $x^{p,12}$ corresponding to generator $(23/25)_{octal}$ is sent. The transmitter will restart from rate 1/3 if all possible rates are exhausted.

We give the simulated throughput performance of two low rate solutions in figure 4.12. Due

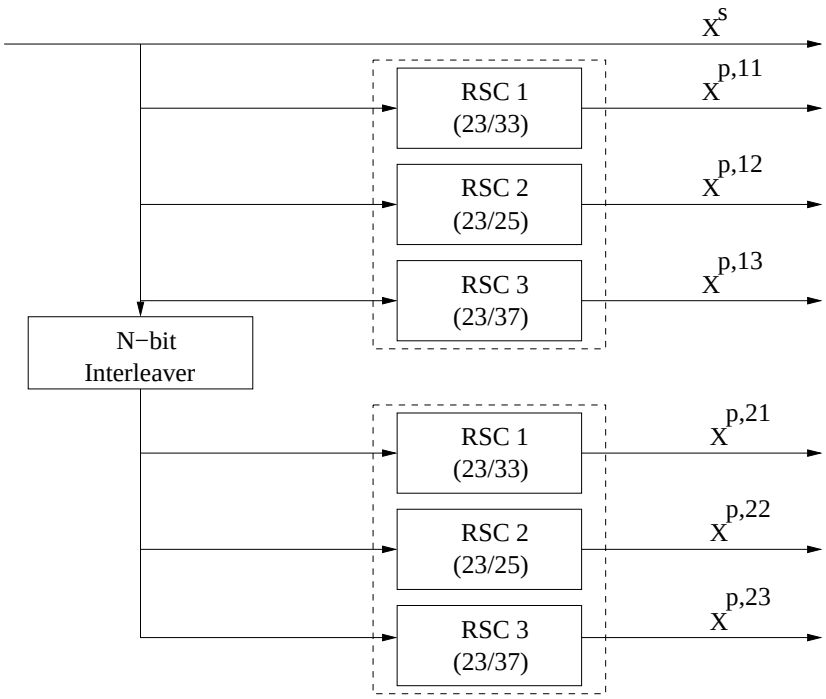


Figure 4.11: Multiple Component Encoding (parent code rate = 1/7)

to the power of turbo code both scheme are saturated when channel SNR is larger than 0 dB, which means that turbo code with rate 1/3 are powerful enough in those case without the need for retransmission. In the low SNR case (SNR $<$ 0dB), two schemes gives almost identical throughput performance. However, the complexity of multiple component turbo codes based low rate solution is higher than repetition code based solution.

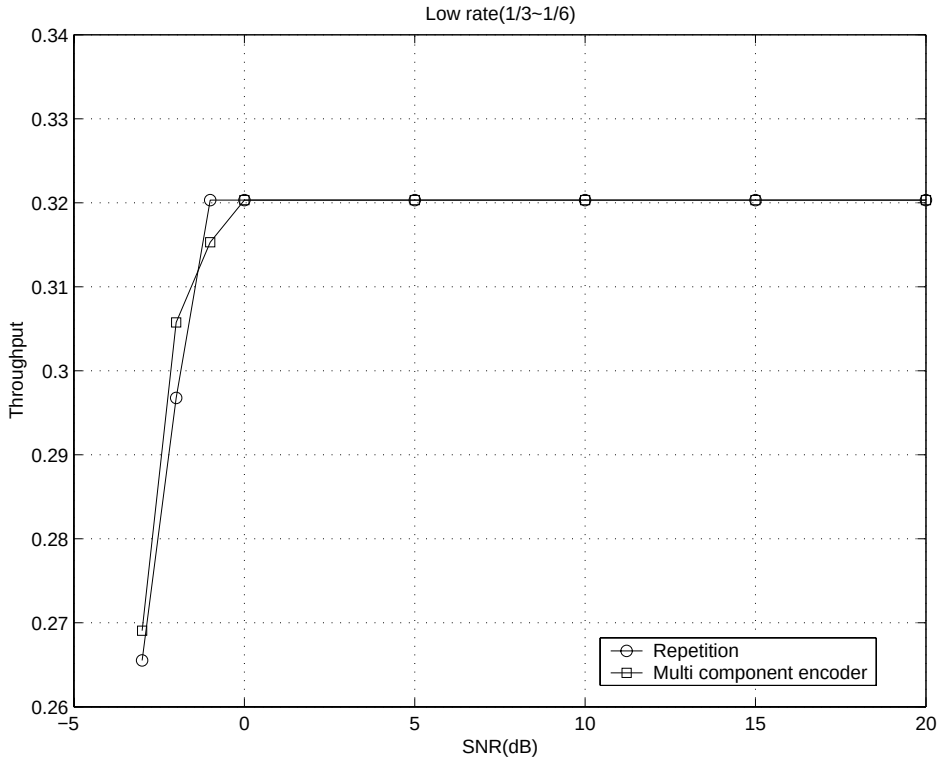


Figure 4.12: The simulation results for block length $N = 512$ including 16 parity check bits and normalized Doppler frequency 0.01.

Chapter 5

Summary

In this report, we investigated the turbo code and its application in hybrid type II ARQ system. The RCPT codes are combined with hybrid type II ARQ strategy in our system.

The throughput performance and the number of retransmission for the RCPT-based and the RCPC-based hybrid type II ARQ system are compared in our simulation. The simulation results show that the RCPT-based system performs better than the RCPC-based system in general. However, RCPT-based system is more complex than RCPC-based system due to the structure of the turbo codes.

The effects of different puncturing schemes to our system are also investigated. We confirmed that the optimal puncturing matrices proposed in [5] give the best performance.

Two low-rate solutions for our system are also investigated. One is the repetition coding based and the other is based on multi-component turbo codes. The simulation results show almost same throughput performance at the wide range of signal to noise ratio.

Bibliography

[1] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error correcting coding and decoding: Turbo codes. In *Proc. 1993 Int Conf. Comm*, pages pp 1064–1070, 1993.

[2] J. Hagenauer. Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their applications. *IEEE Trans. on Communications*, 36(4):389–400, April 1988.

[3] S. Kallel and D. Haccoun. Generalized type II hybrid ARQ scheme using punctured convolutional coding. *IEEE Trans. on Communications*, 38(11):1938–1946, 1990.

[4] S. Lin and D. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, 1983.

[5] Douglas N. Rowitch and Laurence B. Milstein. On the performance of hybrid FEC/ARQ systems using Rate Compatible Punctured turbo (RCPT) codes. *IEEE Trans. on Communications*, 48(6):948–959, 2000.

[6] Richard B. Wells. *Applied Coding and Information Theory for Engineers*. Prentice Hall, 1999.

[7] L. Zihuai. Rate Compatible Convolutional (RCC) codes and their application to hybrid ARQ/FEC transmission. Master's thesis, Chalmers University of Technology, School of Electrical and Computer Engineering, 1998.