Quality of Service and the Architecture of Turbo Codes in Wireless Communication

Mohammad Ashraf Uddin Bhuiyan

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
December, 2005
Quality of Service and the Architecture of Turbo Codes in Wireless Communication

Submitted by
Mohammad Ashraf Uddin Bhuiyan
M. Sc. Engineering Student
Department of Computer Science and Engineering
Student ID: 100005021F

A thesis submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of Masters of Science in Engineering in Computer Science and Engineering

Supervised by
Dr. Md. Shamsul Alam
Professor, Department of CSE, BUET

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DHAKA, BANGLADESH
DECEMBER 2005
The thesis "Quality of Service and the Architecture of Turbo Codes in Wireless Communication", submitted by Mohammad Ashraf Uddin Bhuiyan, Roll No. 100005021F, Registration No. 92109, Session October 2000, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, is accepted as satisfactory for the partial fulfillment of the requirements for the degree of Master of Science in Engineering (Computer Science and Engineering) and approved as to its style and contents. Examination held on 26 December, 2005.

Board of Examiners

1. Dr. Md. Shamsul Alam 26/12/05
   Professor
   Department of CSE
   BUET, Dhaka – 1000

2. Dr. Muhammad Masroor Ali 26/12/05
   Professor & Head
   Department of CSE
   BUET, Dhaka – 1000

3. Dr. Md. Monirul Islam 26/12/05
   Associate Professor
   Department of CSE
   BUET, Dhaka – 1000

4. Dr. Md. Saidur Rahman 26/12/05
   Associate Professor
   Department of CSE
   BUET, Dhaka – 1000

5. Dr. Satya Prasad Mazumder 26/12/05
   Professor
   Department of EEE
   BUET, Dhaka – 1000.
Declaration

I, hereby, declare that the work presented in this thesis is the outcome of the investigation performed by me under the supervision of Dr. Md. Shamsul Alam, Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka. I also declare that no part of this thesis and thereof has been or is being submitted elsewhere for the award of any degree or diploma.

Ashraf Uddin
(Mohammad Ashraf Uddin Bhuiyan)
Acknowledgement

I wish to acknowledge my profound indebtedness to Dr. Md. Shamsul Alam, Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology. Deep knowledge and keen interest of Dr. Md. Shamsul Alam in the field of computer communications influenced me to carry out this project and thesis. His endless patience, scholarly guidance, continual encouragement and constant supervision have made it possible to complete this thesis.

I would like to thank the members of the examination committee, Professor Dr. Muhammad Masroor Ali, Head, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Professor Dr. Satya Prasad Mazumdar, Department of Electrical & Electronics Engineering, Dr. Md. Monirul Islam, Associate Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology and Dr. Md. Saidur Rahman, Associate Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, for their helpful suggestions and careful review of this dissertation. Lastly I would like to convey gratitude to all my course teachers whose teaching helped me a lot to start and complete this thesis work.
Abstract

Quality of Service (QoS) refers to the collective effect of service performance, which determines the degree of satisfaction of a user of the service. In wireless networks, the goal of QoS is to provide quality as Bit Error Rate (BER), delay, delay variation, and data rate. Among these, the major issue in wireless communications is the improvement of BER. Error correcting codes are the main techniques to improve BER in wireless communication. Among all error-correcting codes, Turbo Codes show the best performance in terms of BER as it can reach near the Shannon’s limit and require only 0.7 dB at BER of $10^{-5}$. So, we improve the architecture of Turbo codes and investigate its performance by changing the component of the architecture of the encoder and decoder.

We simulate the classical Turbo codes, UMTS, cdma2000 and CCSDS standards of Turbo codes and study their BER and Frame Error Rate (FER) performances for both the AWGN and Rayleigh channels. The thesis investigates the performance variation of Turbo codes for varying decoding iterations, coding rate and frame size of the codes. Distance spectrum, which is one of the performance measures of Turbo codes at low BER, are measured and compared for classical and third generation Turbo codes. The effect of frame size, code rate and number of decoding iterations on distance spectra of Turbo codes is shown. Then the influence of different interleavers on the performance of Turbo Codes is demonstrated. Among these interleavers, $S$-random interleaver shows the best performance. We suggest an improvement to the $S$-random interleaving algorithm. Simulation results to demonstrate the performance improvement due to modified structure of the $S$-random interleaver. Interleaver gains with improved $S$-random interleaver for different frame sizes, constraint lengths, code rates and number of decoding iterations are determined. Simulation time of Turbo Codes for different interleavers is determined. Then influence of encoder structure on the performance of Turbo codes' is investigated. It is shown that if the feedback polynomial of Turbo encoder is primitive polynomial, it performs better. Then we identify a Turbo encoder structure, which performs better than the UMTS encoder. Following these results, improved Turbo codes can be constructed with better BER performance.
# Contents

Declaration
Acknowledgement
Abstract
List of Figures
List of Tables

## 1. Introduction
1.1 Introduction
1.2 Literature Review
1.3 Problems of Existing Works
1.4 Scope of the Thesis

## 2. Quality of Service
2.1 History of Quality of Service
2.2 Fundamentals of QoS framework
2.2.1 Definitions
2.2.2 The General Model
2.2.3 The ITU/ITF Approach
2.2.4 The IETF Approach
2.2.5 QoS Parameters
2.2.6 Other Related Concept
2.2.7 Contract Between The Customer and The Service Provider
2.3 Customer Quality of Service requirement and measurement
2.3.1 Methodology to identify the customer's QoS requirements
2.3.2 Service Specific QoS Parameters
2.3.3 Measurements
2.3.3.1 Objective Measurement
2.3.3.2 Intrusive Measurement
2.3.3.3 Non-intrusive Measurement
2.3.3.4 Subjective Measurement
2.3.3.5 Who should perform the measurements

<table>
<thead>
<tr>
<th>Page #</th>
</tr>
</thead>
<tbody>
<tr>
<td>II</td>
</tr>
<tr>
<td>III</td>
</tr>
<tr>
<td>IV</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>XIII</td>
</tr>
<tr>
<td>Section</td>
</tr>
<tr>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>2.4 Service Concepts</td>
</tr>
<tr>
<td>2.4.1 Information transfer</td>
</tr>
<tr>
<td>2.4.2 Information Quality</td>
</tr>
<tr>
<td>2.4.3 Supported Bit Rates</td>
</tr>
<tr>
<td>2.5 Range of QoS requirement</td>
</tr>
<tr>
<td>2.6 QoS Based Traffic Classes</td>
</tr>
<tr>
<td>2.6.1 Conversational Class</td>
</tr>
<tr>
<td>2.6.2 Streaming Class</td>
</tr>
<tr>
<td>2.6.3 Interactive Class</td>
</tr>
<tr>
<td>2.6.4 Background Class</td>
</tr>
<tr>
<td>3. Error Control Codes</td>
</tr>
<tr>
<td>3.1 Introduction</td>
</tr>
<tr>
<td>3.2 Error Detection</td>
</tr>
<tr>
<td>3.2.1 Error Detection by Parity Check</td>
</tr>
<tr>
<td>3.2.2 Cyclic Redundancy Check</td>
</tr>
<tr>
<td>3.3 Error correction by retransmission</td>
</tr>
<tr>
<td>3.4 Forward Error Correction</td>
</tr>
<tr>
<td>3.4.1 Shannon and Information theory</td>
</tr>
<tr>
<td>3.4.2 Coding basics</td>
</tr>
<tr>
<td>3.4.2.1 Block Codes</td>
</tr>
<tr>
<td>3.4.2.2 Cyclic Codes</td>
</tr>
<tr>
<td>3.5 Convolutional Codes</td>
</tr>
<tr>
<td>3.5.1 Convolutional Code Representation</td>
</tr>
<tr>
<td>3.5.2 Recursive Systematic Convolutional Codes</td>
</tr>
<tr>
<td>3.5.3 Decoding Algorithm of Convolutional Codes</td>
</tr>
<tr>
<td>3.5.3.1 Hard-Decision Viterbi Algorithm</td>
</tr>
<tr>
<td>3.5.3.2 Soft-Decision Viterbi Algorithm</td>
</tr>
<tr>
<td>3.5.4 Performance Analysis of Convolutional Code</td>
</tr>
<tr>
<td>3.5.4.1 Transfer Function of Convolutional Code</td>
</tr>
<tr>
<td>3.5.4.2 Distance Properties</td>
</tr>
<tr>
<td>3.5.4.4 Degree of Quantization</td>
</tr>
</tbody>
</table>

VI
4. **Turbo Codes**
   4.1 Concatenated codes
   4.2 Turbo Code Encoder
      4.2.1 Recursive Systematic Convolutional (RSC) Encoder
      4.2.2 Trellis Termination
   4.3 Turbo Code Decoder
      4.3.1 Principle of the General Soft-Output Viterbi Decoder
      4.3.2 Reliability of the General SOVA Decoder
      4.3.3 Soft Channel Outputs
      4.3.4 SOVA Component Decoder for a Turbo Code
      4.3.5 SOVA Implementation

5. **Performance of Turbo Codes**
   5.1 Introduction
   5.2 Performance of Classical Turbo codes
      5.2.1 Decoding Delay
   5.3 UMTS Turbo codes
      5.3.1 Encoder
      5.3.2 Trellis termination technique
   5.4 Performance of UMTS Turbo codes
      5.4.1 Decoding delay of UMTS Turbo codes
   5.5 cdma2000 Turbo Codes
      5.5.1 Encoder
      5.5.2 Trellis Termination technique
   5.6 Performance of cdma2000 Turbo codes
      5.6.1 Decoding delay of cdma2000 Turbo codes

6. **Distance Spectrum of Turbo Codes**
   6.1 Introduction
   6.2 Performance Bound of Turbo codes
   6.3 Lowering the Error Floor of Turbo Codes
   6.4 Distance Spectrum Measurement of Turbo Codes
   6.5 Distance Spectrum of Turbo Codes of Different Standards
      6.5.1 ‘Best’ Turbo codes
6.5.2 UMTS Turbo Codes
6.5.3 cdma2000 Turbo Codes
6.5.4 CCSDS Turbo Code
6.6 Effect of Encoder structure on Distance Spectrum

7. Interleaver Design of Turbo Codes

7.1 Introduction
7.2 Interleaver Design
  7.2.1 Block Interleaver
  7.2.2 Random Interleaver
  7.2.3 Circular-Shifting Interleaver
  7.2.4 Pseudo-Random Interleaver
  7.2.5 Optimal (Near-Optimal) Interleaver
  7.2.6 UMTS TC Interleaver
    7.2.6.1 Bits input to a rectangular matrix
    7.2.6.2 Intra-row and inter-row operation
    7.2.6.3 Bits-output from matrix:
  7.2.7 cdma2000 TC Interleaver
7.3 Effect of Interleaver on Distance Spectrum
  7.3.1 Specific Patterns of Low Weight Interleaver Inputs
7.4 Effect of weight-2, 3, 4, 5 Input Pattern
7.5 Improvement of the s -random Interleaving Algorithm
7.6 Latency For Interleaver Design

8. Conclusion

8.1 Introduction
8.2 Contributions
8.3 Suggestions for Further Research

Appendix I
Appendix II
Appendix III
Appendix IV

References
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Basic Turbo Encoder</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Quality of Service process</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Quality of Service concepts</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>The general QoS model, and ITU/ETSI and IETF approach.</td>
<td>18</td>
</tr>
<tr>
<td>3.1</td>
<td>Error-recovered by detection and forward error correction Control</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Error-recovered by detection and retransmission</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Cyclic encoder for $g(D) = 1 + D + D^2$</td>
<td>40</td>
</tr>
<tr>
<td>3.4</td>
<td>Rate 1.2 convolutional encoder</td>
<td>42</td>
</tr>
<tr>
<td>3.5</td>
<td>State diagram for the convolutional code</td>
<td>43</td>
</tr>
<tr>
<td>3.6</td>
<td>Trellis diagram for convolutional code</td>
<td>43</td>
</tr>
<tr>
<td>3.7</td>
<td>A RSC Turbo encoder with generators $G(D) = [1, 15/13, 17/21]$</td>
<td>45</td>
</tr>
<tr>
<td>3.8</td>
<td>Trellis diagram for the RSC code</td>
<td>46</td>
</tr>
<tr>
<td>3.9</td>
<td>Hard- and Soft-decision decoding</td>
<td>47</td>
</tr>
<tr>
<td>3.10</td>
<td>Convolutional code system</td>
<td>48</td>
</tr>
<tr>
<td>3.11</td>
<td>The binary symmetric channel model, where $p$ is the crossover probability.</td>
<td>50</td>
</tr>
<tr>
<td>3.12</td>
<td>Modified state diagram</td>
<td>54</td>
</tr>
<tr>
<td>4.1</td>
<td>Serially Concatenated Code</td>
<td>57</td>
</tr>
<tr>
<td>4.2</td>
<td>Parallel concatenated code</td>
<td>57</td>
</tr>
<tr>
<td>4.3</td>
<td>Fundamental Turbo code encoder</td>
<td>58</td>
</tr>
<tr>
<td>4.4</td>
<td>The RSC encoder with $r = 1/2$ and $K = 3$</td>
<td>59</td>
</tr>
<tr>
<td>4.5</td>
<td>Trellis termination strategy for RSC encoder</td>
<td>60</td>
</tr>
<tr>
<td>4.6</td>
<td>A concatenated SOVA decoder</td>
<td>61</td>
</tr>
<tr>
<td>4.7</td>
<td>Example of survivor and competing paths for reliability estimation at time $t$.</td>
<td>62</td>
</tr>
<tr>
<td>4.8</td>
<td>SOVA component decoder</td>
<td>64</td>
</tr>
<tr>
<td>4.9</td>
<td>SOVA survivor and competing paths for reliability estimation.</td>
<td>67</td>
</tr>
<tr>
<td>4.10</td>
<td>SOVA iterative Turbo code decoder</td>
<td>69</td>
</tr>
<tr>
<td>5.1</td>
<td>Classical Turbo encoder</td>
<td>73</td>
</tr>
<tr>
<td>5.2</td>
<td>Classical Turbo RSC component Encoder</td>
<td>74</td>
</tr>
<tr>
<td>5.3</td>
<td>BER performance of TCs for AWGN and Rayleigh Channel</td>
<td>75</td>
</tr>
</tbody>
</table>
5.4 BER performance of TCs for different frame size
5.5 BER performance of TCs for different Interleaver
5.6 BER performance of the 4-state TCs for different iterations
5.7 BER performance of the code rates
5.8 BER performance of the TC for different constraint length
5.9 Frame Error Rate for different channel model
5.10 Frame Error Rate for different frame size
5.11 Frame Error Rate for number of iteration
5.12 Effect of decoding iterations on decoding delay.
5.13 UMTS encoder
5.14 BER performance of UMTS TCs for different channel
5.15 BER performance of UMTS TCs for different frame size
5.16 BER performance of UMTS TCs for different decoding iterations
5.17 Frame Error Rate for different channel model.
5.18 Frame Error Rate for different frame size
5.19 Frame Error Rate for different number of iteration
5.20 1/3 UMTS Turbo codes operating in AWGN channel
5.21 cdma2000 encoder
5.22 BER performance of cdma2000 TCs for two Channel
5.23 BER performance of cdma2000 TCs for different frame size
5.24 BER performance of cdma2000 TCs for different coding iteration
5.25 BER performance of cdma2000 TCs for different coding rate
5.26 1/3 cdma2000 Turbo codes operating in AWGN channel
6.1 Upper bound of the classical Turbo codes.
6.2 Free distance vs. code rate for different interleaver size.
7.1 Block interleaver.
7.2 A random (pseudo-random) interleaver with L=8.
7.3 A circular-shifting interleaver with L=8, a=3, and s=0.
7.4 A semirandom interleaver with L=16 and S=2.
7.5 Weight-3 sequence before and after interleaving
7.6 Weight-5 input sequence, before and after interleaving
7.7 BER performance of 4-state Turbo codes for different interleavers, Frame
7.8 BER performance of 4-state Turbo codes for different interleavers, Frame size: 1000, iteration: 10, Rate: 1/3
7.9 BER performance of 4-state Turbo codes for different interleavers, Frame size: 4000 bits, iteration: 10, Rate: 1/3
7.10 BER performance of 4-state Turbo codes for different interleavers, Frame size: 1000, iteration: 5, Rate: 1/3
7.11 BER performance of 8-state Turbo codes for different interleavers, Frame size: 1000, iteration: 10, Rate: 1/2
7.12 BER performance of 8-state Turbo codes for different interleavers, Frame size: 4000, iteration: 10, Rate: 1/3
7.13 FER performance of 4-state Turbo codes for different interleavers, Frame size: 1000, iteration: 10, Rate: 1/3
7.14 FER performance of 4-state Turbo codes for different interleavers, Frame size: 4000, iteration: 10, Rate: 1/3
7.15 Simulation time of Turbo code with Improved S-random Interleaver for different frame size
7.16 Simulation time of Turbo code with Improved S-random Interleaver for different iterations
List of Tables

2.1 Matrix to facilitate the capture of customer's QoS requirements 24
2.2 QoS requirements for real time and non-real time service 30
3.1 Conventional Bit Metric Values 50
5.1 Decoding delay of 1/3 classical Turbo codes operating in AWGN channel. 81
5.2 Latency of 1/3 UMTS Turbo codes operating in AWGN channel 87
5.3 Latency of 1/3 cdma2000 Turbo codes operating in AWGN channel 93
6.1 Distance Spectrum of 'best' Turbo code 101
6.2 Distance Spectrum of UMTS Turbo code 102
6.3 Distance Spectrum of cdma2000 Turbo code 102
6.4 Distance Spectrum of CCSDS Turbo code 103
6.5 Distance Spectrum of different encoders ( 2 Shift registers ) 105
6.6 Distance Spectrum of different encoders ( 3 Shift registers ) 105
6.7 Distance Spectrum of different encoders ( 4 Shift registers ) 106
7.1 Distance Spectrum of Different Interleavers 115
7.2 Encoder Output Using Impulse Responses of Low Weight Input 116
7.3 Relative Contribution of Low Weight Input on BER 118
7.4 Simulation time for different algorithm with 5 decoding iterations 128
7.5 Simulation time for different interleaving algorithm with frame size of 570 129
1.1 Introduction

Quality of Service (QoS) is the quality of a requested service as perceived by the customer. Network Performance of several network elements of the originating and terminating network contribute to the QoS. In order to offer the customer a certain QoS, the serving network need to take into account network performance components of their network, reflect the performance of the terminal and add sufficient margin for the terminating networks in case network performance requirements cannot be negotiated.

Main parameters of quality of services of a network are transfer delay, delay variation, bit error rate and data rate.

Transfer delay is the time between the requests to transfer the information at one access point to its delivery at the other access point. When delay is more or less constant, engineers of latency sensitive applications try to improve application performance in presence of it. But when delay varies, the problem complicates a lot because it is not as easy to take providences for several stages. The delay variation of the information has to be controlled to support real-time services.

The ratio between incorrect and total transferred information bits is the Bit Error Rate (BER). The data rate is the amount of data transferred between the two access points in a given period of time.

Depending upon the above-mentioned QoS parameters, a set of QoS classes can be defined. These can be referred to as traffic classes. There are four different QoS classes: Conversational class, Streaming class, Interactive class and Background class. The main distinguishing factors between these QoS classes are delay and bit error rate sensitivity. Conversational and Streaming classes are mainly intended to be used to carry real-time traffic flows. So they are the most delay sensitive applications.

Interactive class and Background class are mainly meant to be used by traditional Internet applications like WWW, Email, Telnet, FTP and News. Due to looser delay requirements,
compared to conversational and streaming classes, both provide better error rate by means of channel coding and retransmission. The main difference between Interactive and Background class is that Interactive class is mainly used by interactive applications, e.g. interactive Email or interactive Web browsing, while Background class is meant for background traffic, e.g. background download of Emails or background file downloading.

Among these qualities of service parameters, Bit Error Rate (BER) is one of the important issues in wireless communication, as the wireless channel is noisy and prone to error.

To improve the bit error rate performance that is to decrease BER, one of the proven methods is to use the channel coding. The history of channel coding starts after the publication of the paper, ‘The Mathematical Theory of Communication’ by Claude Shannon [1]. With this paper other two papers by Hamming and Golay [2, 3] in the fortieth decade of the last century also draw the attention of the mathematician and engineers to error correcting coding. In his paper, Shannon set forth the theoretical basis for coding, which has come to be known as information theory. By mathematically defining the entropy of an information source and the capacity of a communications channel, he showed that it was possible to achieve reliable communications over a noisy channel provided that the source’s entropy is lower than the channel’s capacity.

At the same time that Shannon was defining the theoretical limits of reliable communication, Hamming and Golay were busy developing the first practical error control schemes. Their work gave birth to a flourishing branch of applied mathematics known as coding theory.

Hamming presented a class of single error correcting codes in his paper [2]. But Hamming codes are not very efficient, as it requires 3 check bits for every 4 bits. Secondly, it only had the ability to correct a single error within the block. Marcel Golay, who generalized Hamming’s construction, addressed these problems [3]. His codes correct up to 3 errors. This type of code is known as a block code, and is referred to as a \((q, n, k, t)\) block code, where \(q\) is the number of bits in one symbols, \(n\) is the length of the code, \(k\) is the length of the information, \(t\) is the correctable symbols.

The next main class of linear Block codes to be discovered was the Reed-Muller (RM) codes. This type of codes allowed more flexibility in the size of the codeword and the number of correctable errors per codeword. Following the discovery of RM codes came
the discovery of cyclic codes, which have some additional property that any cyclic shift of codeword is also a codeword. The cyclic property adds a considerable amount of structure to the code, which can be exploited by reduced complexity encoders and more importantly reduced complexity decoders. Hocquenghem, Bose and Ray-Chaudhuri discovered an important subclass of the cyclic codes simultaneously and the code is known as BCH codes. BCH codes are for binary codes and it can be extended to non-binary case, which was developed by Reed and Solomon and named as RS codes.

Despite the success of Block codes, there are several fundamental drawbacks to their use. First, due to the frame-oriented nature of Block codes, the entire codeword must be received before decoding can be completed. This can introduce an intolerable latency into the system, particularly for large block lengths. A second drawback is that Block codes require precise frame synchronization. A third drawback is that algebraic-based decoders for Block codes usually work with hard-bit decisions, rather than with the unquantized, output of the channel is taken to be binary, while with soft-decision decoding the channel output is continuous-valued. The drawbacks of Block codes can be avoided by taking a different approach to coding, that of Convolutional coding which was first introduced in 1955 by Elias [4]. Rather than segmenting data into distinct blocks, Convolutional encoders add redundancy to a continuous stream of input data by using a linear shift register. Just as the data is continuously encoded, it can be continuously decoded with only nominal latency. Furthermore, the decoding algorithms can make full use of soft-decision information from the demodulator.

A key weakness of Convolutional codes is that they are very susceptible to burst errors. Using an interleaver, which scrambles the order of the code bits prior to transmission, can alleviate this weakness. A deinterleaver at the receiver places the received code bits back in the proper order after demodulation and prior to decoding. Scrambling the code bits' order at the transmitter and then reversing the process at the receiver can spread burst errors so that they appear independent to the decoder.

The first operational code was Reed Mular code used during 1969 Mariner mission to Mars. This code provided 3.2 dB coding gain and the code rate was very low as 0.1875. Although this was a modest coding gain, it reduced the overall system cost per dB as 1,000,000 USD. Convolutional code provides coding gain of 6.9 dB. Serial concatenated
code using Convolutional and Reed Solomon codes used in Galileo mission provided the coding gain of 9 dB with a very low coding rate (0.19).

The gap between practical coding systems and Shannon’s theoretical limit closed even further in June 1993 at the International Conference on Communication (ICC) in Geneva Switzerland. At this conference Berrou, Glavieux, and Thitimajshima coined the term “Turbo Codes” [5] to describe the new class of codes. A Turbo code is the parallel concatenation of two or more component codes as shown in Fig. 1.1. In its original form, the constituent codes were from a subclass of Convolutional codes known as Recursive Systematic Convolutional (RSC) codes. Turbo codes is also called Parallel Concatenation of Convolutional Codes (PCCC). Concatenation of Codes was first invented by Forney [6]. As shown in Fig. 1.1, the input data is interleaved before being fed into the lower encoder.

\[ \text{Data} \xrightarrow{} \text{RSC-1} \xrightarrow{} x \]

\[ \xrightarrow{} \text{Interleaver} \]

\[ \xrightarrow{} \text{RSC-2} \xrightarrow{} y_1 \]

\[ \xrightarrow{} y_2 \]

**Fig. 1.1: Basic Turbo Encoder.** \( x \) is the systematic output and \( y_1 \) and \( y_2 \) are the parity outputs.

The original Turbo code [5] used constraint length \( K = 5 \), RSC encoder and a 65,536 bit interleaver. The parity bits were punctured such that the overall code was a \( \{ n = 131,072, k = 65536 \} \) linear block code. Simulation results showed that a bit error rate of \( 10^{-5} \) could be attained at a \( E_b/N_0 \) ratio of just 0.7 decibels (dB) after 18 iterations of decoding. Thus Turbo code could come within a 0.7 dB of the Shannon limit. Shannon limit for the Additive White Gaussian Noise (AWGN) channel with rate \( \frac{1}{2} \) coding and binary input symbols is 0 dB for arbitrarily low bit error rate, which many author, assumed as \( 10^{-5} \).
After the introduction of Turbo codes researchers and engineers started to find the causes behind the extraordinary performance of Turbo codes and different aspects of its performance curve. The performance measure is mainly the BER vs. $E_b/N_0$ and FER vs. $E_b/N_0$ curves. It is also found that the architecture of Turbo codes has effect on its performance. If the different components of the architecture of Turbo codes are changed, its performance is also changed. These components are the size of interleaver, number of decoding iterations, decoding algorithm, number of shift registers i.e. constraint length and number of encoders. If the channel is modeled as AWGN, Turbo codes performs well than the Rayleigh channel. Rayleigh channel incorporate the multipath fading and the relative velocity of the sender and receiver. Type of interleaving algorithms also has the influence on the performance of Turbo codes. Another performance measure of Turbo codes is its distance spectrum. Free distance is the minimum Hamming distance of the output code of Turbo codes. It is found that at moderate to high $E_b/N_0$, Turbo code performance curve flattens i.e., its BER is not decreasing with the increase of $E_b/N_0$. Researchers find that the low free distance of the codes is responsible for it. Till now coding theorist is trying to improve the low free distance by taking different measures.

Recently there are lots of researches on the interleaving algorithm to improve the performance of Turbo codes. There are Block, Random, $S$-random, Circular Shift, Semirandom and many more interleaving algorithms. In this thesis, improvement of $S$-random algorithm is suggested. Another recent research area of the coding theorists is to design a RSC encoder, which will improve the performance of Turbo codes. This thesis also sheds some light in this research area.

Turbo codes are now incorporated as the error-correcting standard for the third generation wireless communication. Universal Mobile Telecommunication Systems (UMTS) and cdma2000, combined projects of different European, Asian and American wireless communication standard bodies, recommends Turbo codes for their error correction coding standards. Deep space communication standard named as Consultative Committee for Space Data System (CCSDS) also recommends Turbo code for the deep space data acquisition. The architectures of these standards of Turbo codes are different. The study of Turbo codes of different standards and the variation of the performance for the
variation of different architectural component of Turbo codes is necessary. This thesis analyzed the performance of Turbo codes of different standards.

1.2 Literature Review

The concept of quality of service started with the evolution of electronic communication in the Seventieth decade of the last century.

The ICCITT Study group II produced important result on dependability planning of telecommunications networks, on field data collection, on evaluation of the performance of equipment, networks and services [7]. The main aspects of user oriented performance and its translation to network-oriented performance for a telecommunication network may be classified as Accessibility, Integrity and Retainability [8].

In [8], two frameworks are presented for achieving overall quality of service, which is shown in the Fig. 1.2.

![Fig. 1.2: Quality of Service process](image-url)
In [9], P. Richardson et. al. focused on developing a new framework for the resource management on the wireless link that transfers information from the backbone through the ISP facility to the homes. The proposed approach orders the transmission of packets based on session state, and the specified QoS support. Their algorithms took into account the characteristics of the wireless channel as well as the Quality of Service contracts for each traffic stream.

In [10], G. Legrand et. al. developed the algorithm which reserves resources in the cells where the mobile is likely to go. They named the new protocol as MIR (Mobile IP Reservation Protocol).

In order to provide QoS support, it is necessary to effectively control the total traffic that can flow into the network system. And the key to a successful admission control in multimedia networks is QoS routing [11].

In [12], an end-to-end negotiation protocol for negotiating and coordinating QoS on an end-to-end basis both at application and network layer is presented. The protocol enables the negotiation of system capabilities and allows service provider to effectively influence the negotiation process. In [13], Flexible Resource-Allocation strategy with differentiated priorities and quality of service with prioritized levels is presented. The main distinguishing feature of these strategies is their capacity to prioritize some service types over others. Different standard bodies [14-15] define and explain different aspects of QoS. In [16], the QoS of Bluetooth is improved by using Turbo codes as error control coding scheme instead of Automatic Repeat Request (ARQ). Reconfiguration of Turbo codes for differentiated QoS is discussed in [17]. In this paper the main QoS parameter, BER is improved with the application of Turbo codes in wireless communication. To improve BER performance, for last fifty years, researchers, engineers and coding theorist tried to invent such types error correcting coding techniques so that it can reach the Shannon Limit [18-20]. But the old Block codes could not reach the Shannon’s Limit. Convolutional codes have more improved BER characteristics [21, 22] than Block codes, though it also cannot reach the Shannon Limit. Turbo codes [5], invented in 1993, and analyzed in [23-25], able to reach within 0.7 dB of the Shannon Limit. The weight distribution of Turbo codes and the computation of free distance of Turbo codes is analyzed in [26]. The analysis and interpretation of Turbo codes in terms of distance spectrum can be found in [27, 28]. To compute the free distance of Turbo codes and other
serially concatenated codes, an efficient algorithm is developed in [29]. Improvements of this algorithm are shown in [30]. In this thesis we have implemented this algorithm to computer free distance of Turbo codes. The hardware aspect of Turbo codes decoder to ensure the high performance and high throughput is discussed in [31]. In this paper, to improve the Soft-Output Viterbi Algorithm (SOVA)-based Turbo decoder, a method is proposed which uses a mapping function to compute a target-scaling factor to normalize the extrinsic information output from Turbo decoder. In [32], a simple method is developed to increase the free distance of turbo codes. It is found in [33] that the performance of Turbo codes varies by puncturing pattern. With the puncturing technique, code rate is increased but some information about the original data is lost. So puncturing technique degrades the performance of Turbo codes. The code rate allocation for different application and other aspect of code rate is discussed in [34]. Frame error rate measurement of Turbo codes and its interpretation is discussed in [35]. Interleaver design has a great impact on Turbo codes’ performance. For Classical bursty channel, different interleavers were introduced and explained by J. L. Ramsey [36] and G. D. Forney [37]. Interleaver properties and their applications to the trellis complexity analysis of Turbo codes is presented in [38]. In this paper, properties of all types of interleaver are generally explained. In [39], interleaver and puncturing method are simultaneously implemented for Turbo codes. In [40], a code-matched interleaver is designed for Turbo codes. M. C. Valenti in his Ph. D. Dissertation discussed various aspect of Turbo codes’ application [41]. Soft output Viterbi Algorithm, which is one of the decoding algorithms of Turbo codes, is explained in [42]. Ten years after the invention, Turbo codes becomes the standard in different applications [43, 44]. Now, Universal Mobile Telecommunication Standards (UMTS) [45], a third generation mobile communication standard, suggests Turbo codes as one of the error correcting codes. Different aspects of this standard Turbo codes are discussed in [46]. cdma2000, which is another standard of third generation wireless communication, also proposed Turbo codes as their error control scheme. The implementation issues of cdma2000 Turbo codes are discussed in [48-49]. Deep space communication standard body, named as Consultative Committee for Space Data Systems (CCSDS), also recommends Turbo codes as their error correcting codes [50]. Different aspects of deep space communication standard Turbo codes are discussed in [51]. Besides these applications Turbo codes is intended to be used in Digital Video Broadcasting [52], Bluetooth [53], Personal Communication Service (PCS) [54]. As
Turbo codes can reach within 0.7 dB of Shannon Limit, questions about the ultimate error control codes are arisen in the communication communities. Some theorist argues that Turbo codes will be the ultimate error control codes. This subject is discussed in [55]. In this paper, other codes like Low Density Parity Check Codes (LDPC) and product codes, which have high performance, are discussed. These codes will be the competitors of Turbo codes in the near future.

1.3 Problems of Existing Works

The problems of existing works are summarized as follows:

i) Quality of service concepts, QoS architecture, QoS supporting methods and protocols in wireless communications are not comprehensively studied in the recent literatures. There are many literatures, which have addressed some issues and parameters of QoS. But these studies are not adequate as, besides the voice and text, the evolving wireless and cellular communications is now adopting video and other multimedia services frequently which needs differentiated QoS.

ii) Present works on Quality of Service focuses mainly on delay, delay variation and data rate. Different protocols and methods have been developed to support differential services for video, voice and data. However, for wireless communications BER and Frame Error Rate (FER) are the most important QoS parameters. But different aspects of BER and FER performance for specific application are not addressed thoroughly in the literature.

iii) The structure of Turbo codes and its performance is addressed in literatures. But the fact that different components of the Turbo codes architecture have the effect on the performance of Turbo code is not extensively studied. Specially the performance of Turbo codes for UMTS, cdma2000 and CCSDS standards, with the varying architectural components, are not studied and compared in the literature.

iv) Distance spectrum of Turbo codes is generally investigated in the literatures. Nevertheless, the method of the distance spectrum measurement is not extended to the third generation wireless standards and CCSDS standards Turbo codes. Effect of code rate and frame size on distance spectrum is not investigated.
v) There are a large number of interleaving algorithms. $S$-random is one of the best algorithms. But in the literature discussing $S$-random interleaving algorithm, only 2 and 4-weight input sequences to the Turbo code encoder are taken into account. But 3 and 5-weight input sequences which have the role to produce the low free distance are not considered.

vi) The encoder structure of Turbo codes and its effect on the BER performance and distance spectrum are not thoroughly investigated.

1.4 Scope of the Thesis

Studying the Quality of service aspects of wireless communication and its relation to the architecture of Turbo codes the thesis has the following scope:

i) A complete picture of Quality of Service concepts in wireless communication, QoS framework, QoS classes, QoS requirement are presented.

ii) One of the main parameters of QoS in wireless communication is identified as Bit Error Rate (BER). Mainly the encoder structure, decoder structure, interleaver size, interleaver type, code rate of Classical, UMTS, cdma2000 and CCSDS Turbo codes are varied to get the different BER and Frame Error Rate (FER) performance.

iii) Distance spectrum is another performance measure of Turbo codes. In this thesis, distance spectrum of Classical, UMTS, cdma2000, and CCSDS standard Turbo codes are measured and compared. Architecture of Turbo codes (mainly frame size and code rate) has the influence on its distance spectrum, which is evaluated in this thesis.

iv) Interleaving algorithms are extensively studied and the effect of interleaving algorithms on the distance spectrum of Turbo codes of the above mentioned standards are shown by simulation results.

v) One of the best interleaving algorithms is identified as $S$-random algorithm by measuring the distance spectrum. Then an improvement of the $S$-random interleaving algorithm is suggested. The improvement, in terms of interleaver gain, is shown in terms of the BER and FER curves by using simulation. The interleaver gain is shown for different frame sizes, decoding iterations, code
rates and number of shift registers. Decoding complexities is increased for the improvement of interleaving algorithm, which is measured in terms of simulation time. Simulation time varies with the variation of frame size and number of decoding iterations.

vi) The effect of encoder structure of Turbo codes on its performance is investigated for 2, 3 and 4 shift registers. Among different encoders, best encoder is identified. By simulation it is shown that this encoder performs better than the UMTS encoder.

The remaining chapters of this thesis are organized as follows:

- Chapter 2 provides the concepts of Quality of Service. New concepts of QoS for wireless network and its adopted multimedia services are explained.
- Chapter 3 presents the architectures of basic error correcting coding. Mainly some well performed Block codes and Convolutional codes are explained.
- Chapter 4 presents a detailed description of Turbo codes encoder and decoder structure and associated decoding algorithm. Mainly Soft Output Viterbi Algorithm (SOVA) is explained as decoding algorithm.
- Chapter 5 shows the performance curves of Classical, UMTS, cdma2000 and CCSDS Turbo codes. The effect of the architectural variation on the Turbo codes' performance is studied. These are shown and compared in performance curves.
- Chapter 6 presents the concepts of distance spectrum of Turbo codes. The distance spectrums of different standard Turbo codes are measured by using a recent algorithm. Then the performance of different standard Turbo codes are compared in terms of distance spectrum. The effect of code rate and frame size of Turbo codes on distance spectrum is shown. The effect of different encoder structure on distance spectrum is studied. Best encoder is identified.
- In Chapter 7, most useful interleaving algorithms and their effect on Turbo codes' performance are shown. The improvement of the $S$-random algorithm is suggested. Interleaver gain for the improved interleaver is shown for different code rate, number of decoding iterations, frame size and number of shift
registers is shown in the BER and FER curves of Turbo codes. Increase in the interleaver complexity is shown in terms of simulation time. Simulation time of Turbo codes with UMTS, cdma2000, Classical and improved $S$-random interleavers is shown for different frame size and number of decoding iterations.

- Finally, in chapter 8, contributions, limitations and future works of the research are presented.
Chapter 2
Quality of Service

2.1 History of Quality of Service concepts

Quality of Service is the collective effect of service performances, which determine the degree of satisfaction of a user of a service. It is characterized by the combined aspects of performance factors applicable to all services, such as service operability performance, service accessibility performance, service retention performance and service integrity performance.

The concept of quality of service started with the evolution of electronic communication in the Nineteen Seventieth decade. Before 1980, the study of quality of service, was carried by Consultative Committee for International Telegraph and Telephone (CCITT) through a number of Study Groups (e.g., II, IV, XVIII etc.), each devoted to focus on recommending performance levels for equipment, interfaces, grade of service, transmission quality etc. The first successful attempt to rationalize and integrate the approach to quality of service was created at 1977 by a proper Study Group called "Joint Group on Noise Availability". The work of this group has a great importance because, for the first time, an international committee for standardization produced a systematic taxonomy of the main concepts of quality of service in telecommunication, shown in Fig. 2.1. Architecture of concepts was created, capable of easily accepting and enrichment necessary to cope with different fields of application, and which considers the users as reference point.

In 1984 this joint group was dissolved and incorporated to the study group II of ICCITT. Study group II produced important result on dependability planning of telecommunications networks, on field data collection, on evaluation of the performance of equipment, networks and services [7].

It is foreseen that more users will require more explicit guarantees of QoS for provision of services. Thus, the control and verification of QoS is becoming increasingly important to both service providers and their users. Having a structured collection of
Quality of Service concepts (i.e. a QoS framework) enables Public Network Operators (PNO) / Service Providers (SP) to handle QoS more efficiently. In particular, a QoS framework includes QoS terminology and related aspects like characterization and management. Various international bodies have made efforts towards making QoS frameworks e.g.:

- ITU (International Telecommunication Union) where part of Rec. E.800, [14], is adopted by IEC (International Electro-technical Commission) as terminology standard IEC 191.

- ETSI (European Telecommunication Standards Institute) framework [56]. This framework is based on the work of the FITCE (Federation of Telecommunication Engineers of the European Community) Study Commission [57].

- The ISO (International Standards Organization)/OSI (Open System Interconnection) QoS framework [58].

- The Telecommunication Information Networking Architecture Consortium (TINA-C) QoS framework [59].

- EURESCOM (European Institute for Research and Strategic Studies in Telecommunications), which has addressed QoS issues in a series of projects e.g. [60].

Considering the challenge of assuring QoS, one of the main achievements would be to arrive at harmonized understanding of QoS among the myriad of actors involved in service usage/provision (e.g. users, service providers, content providers, regulatory
authority, etc.). In view of the ever growing number of services, actors, networks, technologies and their structuring and relationships, such an understanding should be as general applicable as possible. However, further details can be supplied for specific service implementations without contradicting the principles of the generic QoS framework. A generic structure of interconnection agreements between involved actors will be described as part of a harmonized understanding. This is an area of steadily increasing interest, facing questions like how to provide adequate end-to-end QoS when several providers are involved. In particular, this is an essential issue for the interface towards the end-user, related to other interfaces in the service provision chain.

2.2 Fundamentals of Quality of Service Framework

In general, a QoS framework defines QoS-related terminology and concepts helping us to handle QoS. The QoS framework brings a harmonized understanding of terms essential to managing QoS.

2.2.1 Definitions

The term service in the telecommunications context seems to be obvious. It pertains to the capability to exchange information through a telecommunications medium, provided to a customer by a service provider. Features and parameters of the service are well specified. A service defined by ITU as “a service provided by the service plane to an end user (e.g., a host [end system] or a network element) and which utilizes the transfer capabilities and associated control and management functions, for delivery of the user information specified by the service level agreements” [14]. ITU describes parameters, attributes and classes of IP-based services. In [15], service is defined as, a type of telecommunication service that provides the capability of transmission of signals between access points.

The term quality, defined in [15], as “the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs,” is less tangible. In fact, the meaning of this term is very broad. In telecommunications, the term quality is commonly used in assessing whether the service satisfies the user’s expectations. The evaluation, however, depends on various criteria related to the party rating the service. Customers assess it based on a personal impression and in comparison to their expectations, while an engineer expresses quality in terms of technical parameters. This discrepancy may
sometimes lead to misunderstandings. Hence, the term QoS is used in many meanings ranging from the user's perception of the service to a set of connection parameters necessary to achieve particular service quality.

The ITU and ETSI approaches to QoS-related terminology are almost the same. In fact, both organizations adopted the concepts of each other while developing the notion of QoS. They use the same basic definition of QoS expressed as "the collective effect of service performance, which determine the degree of satisfaction of a user of the service." It is characterized by the combined aspects of performance factors applicable to all services, such as:

- Service operability performance;
- Service accessibility performance;
- Service retention performance;
- Service integrity performance;

In the EQoS (EURESCOM QoS) framework, QoS is defined as "the degree of conformance of the service delivered to a user by a provider with an agreement between them". QoS is described through the selection of a set of QoS parameters, specification of QoS target values and the choice of QoS measurements and evaluation mechanisms. A QoS parameter is a variable that characterizes QoS.

### 2.2.2 The General Model

There are three notions of QoS defined — intrinsic, perceived, and assessed — that constitute the general model. Intrinsic QoS pertains to service features stemming from technical aspects. Thus, intrinsic quality is determined by a transport network design and provisioning of network access, terminations, and connections. The required quality is achieved, among other things, by an appropriate selection of transport protocols, the QoS assurance mechanisms, and related values of parameters. Intrinsic QoS is evaluated by the comparison of measured and expected performance characteristics. User perception of the service does not influence the intrinsic QoS rating. Perceived QoS reflects the customer's experience of using a particular service. It is influenced by the customer's expectations compared to observed service performance. In turn, personal expectations are usually affected by the customer's experience with a similar
telecommunications service and other customers' opinions. Thus, various customers may perceive the QoS with the same intrinsic features differently. It follows that just ensuring particular service (network) parameters may not be sufficient to satisfy customers who are not concerned with how a service is provided. The QoS offered by a provider must reflect the intrinsic QoS as well as some nontechnical parameters that are meaningful to the customer and relevant to a particular community's expectations. The assessed QoS starts to be seen when the customer decides whether to continue using the service or not. This decision depends on the perceived quality, service price, and responses of the provider to submitted complaints and problems. It follows that even a customer service representative's attitude to a client may be an important factor in rating the assessed QoS. Neither ITU nor ETSI nor IETF deal with the assessed QoS. The assurance of a satisfactory level of intrinsic, perceived, and assessed QoS may be considered separately. The first is the responsibility of a network provider and depends on network architecture, planning, and management. It is mainly a technical problem dealt with by engineers, designers, and operators. An appropriate use of the intrinsic QoS capabilities adjusted to a particular service offered, together with market analysis, is necessary to ensure a high level of perceived QoS. This is the duty of the service provider. Advertising and marketing efforts have an impact on perceived QoS as well. The assessed QoS mainly depends on the charging policy of a provider as well as reliable customer service representatives and technical support.

2.2.3 The ITU/ETSI Approach

The ITU and ETSI approaches to QoS-related terminology are almost the same. In fact, both organizations adopted the concepts of each other while developing the notion of QoS. QoS in the ITU/ETSI approach adheres mainly to perceived rather than intrinsic QoS. Besides, they introduce the notion of network performance (NP) to cover technical facets. They make a clear distinction between QoS, understood as something focused on user-perceivable effects, and NP, encompassing all network functions essential to provide a service. QoS parameters are user-oriented and do not directly translate into network parameters. On the other hand, the network performance parameters determine the quality observed by customers but are not necessarily meaningful to them. But there must exist a consistent mapping between the QoS and NP parameters. Network performance, as mentioned above, corresponds to intrinsic QoS. It is defined as "the ability of a network
or network portion to provide the functions related to communications between users.” NP is defined and measured in terms of parameters of particular network components involved in providing a service. These parameters are the key to network efficiency and effectiveness in providing a service. A high level of NP is achieved by appropriate system design, configuration, operation, and maintenance. Some network performance parameters are defined by ITU. To cover various points of view on QoS, ITU and ETSI distinguish four particular definitions shown in Fig. 2.2:

- QoS requirements of the customer
- QoS offered by the provider
- QoS achieved by the provider
- QoS perceived by the customer

![Diagram of QoS definitions](image)

**Fig. 2.2:** ITU/ETSI approach of QoS.

The requirements of the customer state their preferences for a particular service quality. They may be expressed in technical or nontechnical language understandable to both the customer and the service provider. The latter designs the service offered to the customer based on their requirements, even though the service provider may not always be in a position to meet the customer’s expectations. The QoS offered may be influenced by the considerations of a service provider’s strategy, benchmarking, service deployment cost, and other factors. It is expressed by values assigned to parameters understandable to the customer (e.g., “service availability in a year is 99.95 percent”). The QoS achieved is usually expressed by the same set of parameters. Comparison of the quality offered and achieved gives the service provider a preliminary rating of perceived service
performance. However, the most important feedback, from the service provider's perspective, is QoS perceived by the customer, who finally rates the service quality comparing the experienced quality to his/her requirements.

QoS and NP are interrelated. Ensuring high network performance is crucial to a successful service provision. Parameters of the QoS offered can be categorized as network and non-network related. The former, in turn, can be translated into NP parameters. Target values of these parameters are assigned. The achieved network performance is obtained based on a parameter measurement. It gives feedback to the network provider. The combination of the NP achieved and non-network-related QoS constitutes the QoS achieved.

2.2.4 The IETF Approach

IETF focuses on intrinsic QoS and does not deal with perceived QoS. It stems from the main objectives of IETF, concerned with the Internet architecture and its development, dependability, and effectiveness. QoS is understood by IETF as "A set of service requirements to be met by the network while transporting a flow". It is closely equivalent to the notion of NP defined by ITU/ETSI and is defined in terms of parameters. During the past few years, IETF has devoted a lot of attention to QoS assurance in IP networks. It developed various QoS mechanisms for the Internet. It proposed two significant network architectures: IntServ and DiffServ. It standardized the Resource Reservation Protocol (RSVP) signaling protocol, originally intended for IntServ model implementation and extended later for other purposes. It also developed the notion of IP-QoS architecture as a comprehensive approach to QoS, and proposed several solutions. IETF defines some architecture-independent QoS parameters as well as specific parameters of network components, such as traffic meters, packet markers, droppers, or schedulers, constituting particular network architecture. There is a close relationship between parameters of network components and the "quality" experienced by packets.

2.2.5 QoS Parameters

Intrinsic QoS in packet networks is expressed by at least the following set of parameters that are meaningful for most IP-based services:

- Bit rate of transferring user data available for the service or target throughput that may be achieved.
• Delay experienced by packets while passing through the network. It may be considered either in an end-to-end relation or with regard to a particular network element.

• Variations in the IP packet transfer delay. Again, it can be applied to an end-to-end relation or a single network element.

• Packet loss rate usually defined as the ratio of the number of undelivered packets to sent ones.

These parameters describe the treatment experienced by packets while passing through the network. They can be translated into particular parameters of the network architecture components used to ensure QoS. They are finally mapped into the configuration of network elements. They are also closely connected with protocols used in the network and equipment abilities. Additionally, intrinsic QoS may have the following attributes depending on the network architecture as well as the application demands:

• End-to-end (e.g., in the IntServ model) or limited to a particular domain or domains (e.g., in the DiffServ model)

• Applied to all traffic or just to a particular session or sessions

• Unidirectional or bidirectional

• Guaranteed or statistical

QoS is usually an end-to-end characteristic of communication between end hosts. It should be ensured along the whole path between peers, but the path may cross several autonomous systems belonging to various network providers. Then performance of all autonomous systems contributes to the final service quality. Parameters of perceived QoS are, largely, more difficult to define. They depend on the network architecture, technique, or mechanisms used to ensure service quality. They are usually expressed in different terms but should be always somehow translatable into specific network parameters regardless of the network architecture. An example of an extensive set of parameters of the perceived QoS is provided by ITU. Parameters are grouped into four subsets regarding the aspects of:

• Service support

• Service operability
Service survivability

Service security

Service support, in general, reflects the provider’s ability to provide a service and assist in its utilization. Parameters related to service operability determine the service ability to be operated by a user. Survivability related parameters determine the service ability to be obtained when requested by the user and continue to be provided without excessive impairment for a requested duration. Service security specifies the level of a service’s protection against unauthorized monitoring, fraudulent use, natural disaster, and other impairments.

2.2.6 Other Related Concepts

Class of Service: The Class of Service (CoS) is a broad term describing a set of characteristics available with a specific service. Both IETF and ITU-T define the CoS term. It is defined by IETF as "The definitions of the semantics and parameters of a specific type of QoS". Services belonging to the same class are described by the same set of parameters, which can have qualitative or quantitative values. Usually, the set of parameters within the class is defined without assignment of concrete values, but these values can be bounded. The idea of service classification is relatively mature. For example, the original IP was intended to provide a simple way of classifying packets, but this capability of IP is rarely used. Traffic in asynchronous transfer mode (ATM) networks is divided into classes as well. Currently, concrete service classes are defined within IP-QoS architectures proposed by IETF, such as IntServ and DiffServ. The following three classes are defined within IntServ architecture: guaranteed, controlled load, and best effort. Also, in DiffServ three classes were initially defined (Olympic, premium, and best effort), but this classification is currently of historical importance. ITU-T introduced IP transfer capability, which is related to CoS. The following three transfer capabilities are defined:

- Dedicated bandwidth (DBW) IP transfer capability
- Statistical bandwidth (SBW) IP transfer capability
- Best effort (BE) IP transfer capability
The service model, traffic descriptor, conformance definition, and any QoS commitments characterize each IP transfer capability. IP transfer capabilities strive for compatibility with CoS defined in IETF QoS architectures. For example, DBW is related to guaranteed service and end-to-end services based on the expedited forwarding per-hop behavior.

**Grade of Service:** The notion of Grade of Service (GoS) is sometimes used to categorize services with respect to high-level requirements. Survivability issues or probability of physical damage of a connection due to natural disasters (earthquakes, volcano eruptions, etc.) may be taken into account. For example, services may be differentiated with respect to provision of a protection path that may be physically disjoint with the working path or not. Another criterion may be the possibility of offering connections over the path crossing regions of low damage probability. The term GoS applies, for example, to leased line service or switched connections in optical networks.

### 2.2.7 Contract Between The Customer and The Service Provider

In compliance with the ITU definition, a service level agreement (SLA) is "a negotiated agreement between a customer and the service provider on levels of service characteristics and the associated set of metrics. The content of SLA varies depending on the service offering and includes the attributes required for the negotiated agreement" [14]. An SLA may be in form of a document containing names of the parties signing the contract. According to [14], it should be composed of service level objectives, service monitoring components, and financial compensation components. Service level objectives encompass QoS parameters or class of the service provided, service availability and reliability, authentication issues, the SLA expiry date, and so on. Service monitoring specifies the way of measuring service quality and other parameters used to assess whether the service complies with the SLA. It may also include an agreement on form and frequency of delivering the report on service usage. The financial component may include billing options, penalties for breaking the contract, and so forth. The IETF defines an SLA in a similar way as "a service contract between a customer and a service provider that specifies the forwarding service a customer should receive". An SLA should be expressed in a way intelligible to a customer. It encompasses basic features of the service and well-defined unambiguous criteria of assessing whether the service delivered is consistent with the contract. On the other hand, limits imposed on the
customer must be clear. An SLA must consist of responsibility rules for breaking the contract by the service provider as well as by the customer. It usually includes other parameters such as those defined by ITU. Regarding the IETF definition, SLA may also include traffic conditioning rules that, at least in part, constitute a TCA. Summarizing, SLA is a broad term encompassing technical features and parameters of the service as well as legal and charging aspects. The notion of service level specification (SLS) was introduced to separate a technical part of the contract from SLA. It is defined as "a set of parameters and their values which together define the service offered to a traffic" [14]. In other words, it specifies a set of values of network parameters related to a particular service. The IP transport services are technically described by SLSs. Work aimed at specifying a set of basic parameters that will compose the elementary contents of an SLS is in progress. A traffic conditioning agreement (TCA) is an agreement specifying packet classification rules and traffic profiles as a description of the temporal properties of a traffic stream, such as the rate and burst size. The customer is obliged to adjust the generated traffic streams to a contracted profile. In order to force a customer's traffic conformance to the profile particular metering, marking, discarding, and shaping rules are defined. The treatment of out-of-profile packets is also specified by a TCA. Moreover, according to the IETF definition, "TCA encompasses all of the traffic conditioning rules explicitly specified within a SLA along with all of the rules implicit from the relevant service requirements and/or from a DiffServ domain's service provisioning policy".

2.3 Customer Quality of Service requirement and measurement

2.3.1 Methodology to identify the customer's QoS requirements

Different users may be happy with different levels of QoS as well as level of performance. Users in this aspect might be end-users or operators or service providers as well. This is why defining the user requirements is crucial to ensure an optimal quality/cost ratio. The matrix which is a useful methodology to capture the users'/customers' quality requirements is shown in Table 2.1 below.
Table 2.1: Matrix to facilitate the capture of customer's QoS requirements

<table>
<thead>
<tr>
<th>Service Function</th>
<th>Alteration</th>
<th>Service Support</th>
<th>Repair</th>
<th>Cessation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management</td>
<td>Call</td>
<td>Technical</td>
<td>Connection</td>
<td></td>
</tr>
<tr>
<td>Quality</td>
<td>Management</td>
<td>Establishment</td>
<td>Information Transfer</td>
<td>Connection release</td>
</tr>
<tr>
<td>Billing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All the service functions are assessed based on the following criteria:

1. Speed
2. Accuracy
3. Availability
4. Reliability
5. Security
6. Simplicity
7. Flexibility

Questionnaires are most suited for public enquiries as well as telephone interviews. Face-to-face interviews are more appropriate for surveys in business areas or to get confirmation on some specific issues. Depending on which area the QoS in question is to be evaluated, appropriate samples have to be defined since different categories of users have often differing requirements even for the same service. While doing that, it should be kept in mind that users have not the same knowledge of the technology as the providers and therefore appropriate language should be used. This is still more difficult when the issue is about a new technology not implemented yet. In such case, analogy with existing services has to be found in order to refer to current usage. In any case, users are expecting their QoS requirements be seen in an end-to-end perspective not from a narrow technical viewpoint. Finally, it is important to remember that not every parameter will be relevant to every user.

2.3.2 Service Specific QoS Parameters

There are a number of criteria that can be used for most of the telecommunications services to assess the QoS. For the telecommunication service itself, these criteria are:

a) Failure in setting-up the communication;

b) Time to set up the communication;

c) Disconnection after the communication is set-up;
d) Defect in the communication with a different weight according to the type of use of the communication.

For failures and service breakout:
   a) Performances in back-up situation;
   b) Frequency of failures and service breakout, time to repair, cumulated time of failures (in particular total failure).

   These criteria have to be adapted according to the service provided, fixed, mobile communications or Internet access and email services either to evaluate more accurately the QoS of the service considered or to replace some inappropriate criteria, e.g. time to set up the communication when it is permanently set. It is also obvious that the same service may be used for different applications with different requirements. Therefore, in many cases specific criteria have to be defined in order to take into account any particular aspect of the application using the service considered. Other aspects of the provision of the service itself are more generic and should fit almost all the services.

For delivery:
   a) Time to deliver;
   b) Conformance to the delivery time;
   c) Conformance to the specification;
   d) Conformance to the specification of the documentation;

For the help desk:
   a) Response time;
   b) Relevance of the answer.

For the billing:
   a) Accuracy of the counting;
   b) Exactness of the accounting with respect of the tariff;
   c) Clarity.

For the supplier-customer interface:
   a) Reliability: The ability to provide what was promised, dependably and accurately;
   b) Assurance: The knowledge and courtesy of employees and their ability to convey trust and confidence;
d) Responsiveness: The willingness to help customers and provide prompt services.

2.3.3 Measurements

The assessment of the QoS is expected to be evaluated in checking criteria against reference values. As seen above these criteria are measured either objectively via technical means or subjectively (perceived QoS) via surveys amongst the users. Experts often agree that a mix of objective and subjective measurements is the best means to get the whole QoS picture. As stated above, both ends of the communication may influence the QoS and have to be taken into account for the measurements. In particular, the telecommunication network architecture is more and more often designed to include access networks and transport networks, the influence of which on QoS has to be taken into account in a QoS measurement policy.

2.3.3.1 Objective Measurements

Criteria like call set up time, call failures, interruptions can quite easily be measured via adequate probes in appropriate locations. Measurements can be made either on real traffic or on artificially generated traffic. This can be done either on public traffic or private networks. Again, since QoS may be different with respect to the location, the geography of the network should be taken into account for the measurements particularly if the choice is done not to monitor all the parts of the network. A compromise should be set between the wish to monitor everything all the time and the costs and the possible oversizing of the network to ensure the management traffic.

2.3.3.2 Intrusive Measurements

This type of measurements is performed on artificially generated traffic and can provide more information since the traffic can be tailored to check almost everything. The drawback of intrusive measurements is to add traffic to the actual one and therefore to lead to additional costs and some possible disturbance.

2.3.3.3 Non-intrusive measurements

This type of measurements is performed on real traffic conditions and therefore is expected to give a more realistic vision of the QoS but its drawback is that some deficiencies might be missed since not all the possibilities are checked.
2.3.3.4 Subjective measurements

Subjective measurements are the only means to assess the psychological aspects of the QoS, e.g. those aspects that cannot be measured easily by technical means or that may be missed due to a reduced number of measurement points. This is the case for instance for billing accuracy, quality of customer care or relevance of the answer of the helpdesk. Such measurements may be carried out once a year or if a complaint is raised.

2.3.3.5 Who should perform the measurements

There are various ways to perform the measurements. Big corporations may have their own organization to deal with this issue or, alternatively, the task may be given to a third party. Another possibility is to entrust the provider himself to supply also the QoS information. It is expected in such case that a mechanism be set to ensure the confidence in the information provided. Taking into account that the private users (general public) have requirements and resources different from Business users, it is expected that the measurements related to their requirements are performed by a third party, e.g. some kind of public authority, and the results made publicly available.

2.4 Service Concepts

Network Performance of several network elements of the originating and terminating network(s) contribute to the QoS as perceived by the customer including terminals and terminal attachments. In order to offer the customer a certain QoS the serving network need to take into account network performance components of their network, reflect the performance of the terminal and ad sufficient margin for the terminating networks in case network performance requirements cannot be negotiated. As far as the QoS to the subscriber is concerned network elements have to provide sufficient performance (reflecting possible performance constraints in terminating networks) so that the PLMN cannot be considered as a bottleneck.

2.4.1 Information transfer

Both connections oriented and connectionless services shall be supported.

Traffic type: It is required that the bearer service provides one of the following:

- Guaranteed/constant bit rate,
- Non-guaranteed/dynamically variable bit rate, and
- Real time dynamically variable bit rate with a minimum guaranteed bit rate,

Real time and non real time applications shall be supported.

- Real time video, audio and speech shall be supported. This implies the:
- Ability to provide a real time stream of guaranteed bit rate, end-to-end delay and delay variation.
- Ability to provide a real time conversational service of guaranteed bit rate, end-to-end delay and delay variation.
- Non real time interactive and file transfer service shall be supported. This implies the:
- Ability to support message transport with differentiation as regards QoS between different users.
- Multimedia applications shall be supported. This implies the ability to support several users flows to/from one user having different traffic types (e.g. real time, non real time)

Traffic characteristics: It shall be possible for an application to specify its traffic requirements to the network by requesting a bearer service with one of the following configurations

1) Point-to-Point
   - Uni-Directional
   - Bi-Directional

2) Uni-Directional Point-to-Multipoint
   - Multicast
   - Broadcast

   A multicast topology is one in which sink parties are specified before the connection is established, or by subsequent operations to add or remove parties from the connection. The source of the connection shall always be aware of all parties to which the connection travels.
A broadcast topology is one in which the sink parties are not always known to the source. The connection to individual sink parties is not under the control of the source, but is by request of each sink party.

In the case of a mobile termination with several active bearer services simultaneously, it shall be possible for each bearer service to have independent configurations and source/sink parties.

2.4.2 Information Quality

Information quality characterizes the bit integrity and delay requirements of the applications. Other parameters may be needed.

**Maximum transfer delay:** Transfer delay is the time between the requests to transfer the information at one access point to its delivery at the other access point. In clause 5.5 requirements on maximum transfer delay is defined.

**Delay variation:** The delay variation of the information-received information over the bearer has to be controlled to support real-time services. The possible values for delay variation are not a limited set, but a continuous range of values.

**Bit error ratio:** The ratio between incorrect and total transferred information bits. The possible values for Bit error ratio are not a limited set, but a continuous range of values.

**Data rate:** The data rate is the amount of data transferred between the two access points in a given period of time.

2.4.3 Supported Bit Rates

It shall be possible for one application to specify its traffic requirements to the network by requesting a bearer service with any of the specified traffic type, traffic characteristics, maximum transfer delay, delay variation, bit error ratios & data rates. It shall be possible for the network to satisfy these requirements without wasting resources on the radio and network interfaces due to granularity limitations in bit rates. It shall be possible for one mobile termination to have several active bearer services simultaneously, each of which could be connection oriented or connectionless. The only limiting factor for satisfying application requirements shall be the cumulative bit rate per mobile termination at a given instant (i.e. when summing the bit rates of one mobile termination’s simultaneous connection oriented and connectionless traffic, irrespective of
the traffic being real time or non real time) in each radio environment. The bit rates for satellite radio and rural outdoor radio environment are at least 144 kbps. For urban or suburban outdoor radio environment data rates are at least 384 kbps. For indoor or low range outdoors radio environment data rates are at least 2048 kbps.

2.5 Range of QoS Requirements

It is possible for one application to specify its QoS requirements to the network by requesting a bearer service with any of the specified traffic type, traffic characteristics, maximum transfer delay, delay variation, bit error ratios & data rates. Table 2.2 indicates the range of values that shall be supported. These requirements are valid for both connection and connectionless traffic. It shall be possible for the network to satisfy these requirements without wasting resources on the radio and network interfaces due to granularity limitations in QoS.

Table 2.2: QoS requirements for real time and non-real time service

<table>
<thead>
<tr>
<th>Operating environment</th>
<th>Real Time (Constant Delay)</th>
<th>Non Real Time (Variable Delay)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Satellite</strong></td>
<td>Max Transfer Delay &lt; 400 ms BER $10^{-3}$ - $10^{-7}$</td>
<td>Max Transfer Delay &lt; 1200 ms BER $10^{-5}$ to $10^{-8}$</td>
</tr>
<tr>
<td><strong>Rural outdoor</strong></td>
<td>Max Transfer Delay = 20 - 300 ms BER $10^{-3}$ - $10^{-7}$</td>
<td>Max Transfer Delay &gt;= 150 ms BER $10^{-5}$ to $10^{-3}$</td>
</tr>
<tr>
<td><strong>Urban/ Suburban outdoor</strong></td>
<td>Max Transfer Delay = 20 - 300 ms BER $10^{-3}$ - $10^{-7}$</td>
<td>Max Transfer Delay &gt;= 150 ms BER $10^{-5}$ to $10^{-8}$</td>
</tr>
<tr>
<td><strong>Indoor/ Low range outdoor</strong></td>
<td>Max Transfer Delay 20 - 300 ms BER $10^{-3}$ - $10^{-7}$</td>
<td>Max Transfer Delay &gt;= 150 ms BER $10^{-5}$ to $10^{-8}$</td>
</tr>
</tbody>
</table>

2.6 QoS Based Traffic Classes

When defining the UMTS QoS classes, also referred to as traffic classes, the restrictions and limitations of the air interface have to be taken into account. It is not reasonable to define complex mechanisms, as have been in fixed networks due to different error characteristics of the air interface. The QoS mechanisms provided in the cellular network have to be robust and capable of providing reasonable QoS resolution.
There are four different QoS classes. They are conversational class, streaming class, interactive class and background class.

2.6.1 Conversational Class

The most well known use of this scheme is telephony speech (e.g. GSM). But with Internet and multimedia a number of new applications will require this scheme, for example, voice over IP and video conferencing tools. Real time conversation is always performed between peers (or groups) of live (human) end-users. This is the only scheme where the required characteristics are strictly given by human perception.

Real time conversation scheme is characterized by that the transfer time shall be low because of the conversational nature of the scheme and at the same time that the time relation (variation) between information entities of the stream shall be preserved in the same way as for real time streams. The maximum transfer delay is given by the human perception of video and audio conversation. Therefore the limit for acceptable transfer delay is very strict, as failure to provide low enough transfer delay will result in unacceptable lack of quality. The transfer delay requirement is therefore both significantly lower and more stringent than the round trip delay of the interactive traffic case.

2.6.2 Streaming Class

When the user is looking at (listening to) real time video (audio) the scheme of real time streams applies. The real time data flow is always aiming at a live (human) destination. It is a one-way transport. This scheme is one of the newcomers in data communication, raising a number of new requirements in both telecommunication and data communication systems. It is characterized by that the time relations (variation) between information entities (i.e. samples, packets) within a flow shall be preserved, although it does not have any requirements on low transfer delay. The delay variation of the end-to-end flow shall be limited, to preserve the time relation (variation) between information entities of the stream. But as the stream normally is time aligned at the receiving end (in the user equipment), the highest acceptable delay variation over the transmission media is given by the capability of the time alignment function of the application. Acceptable delay variation is thus much greater than the delay variation given by the limits of human perception.
2.6.3 Interactive Class

When the end-user, that is either a machine or a human, is on line requesting data from remote equipment (e.g. a server), this scheme applies. Examples of human interaction with the remote equipment are: web browsing, data base retrieval, server access. Examples of machine interaction with remote equipment are: polling for measurement records and automatic data base enquiries. Interactive traffic is the other Classical data communication scheme that on an overall level is characterized by the request response pattern of the end-user. At the message destination there is an entity expecting the message (response) within a certain time. Round trip delay time is therefore one of the key attributes. Another characteristic is that the content of the packets shall be transparently transferred (with low bit error rate).

2.6.4 Background Class

When the end-user, that typically is a computer, sends and receives data-files in the background, this scheme applies. Examples are background delivery of E-mails, SMS, download of databases and reception of measurement records.

Background traffic is one of the Classical data communication schemes that on an overall level is characterized by that the destination is not expecting the data within a certain time. The scheme is thus more or less delivery time insensitive. Another characteristic is that the content of the packets shall be transparently transferred (with low bit error rate).
Chapter 3

Error Control codes

3.1 Introduction

Error control codes refers to the class of signal transformation designed to improve communications performance by enabling the transmitted signals to better withstand the effects of various channel impairment, such as noise, interference, and fading. The essence of this ability to correct errors is that we make use of redundancy in the information. The controlled redundancy is the central function of error-control techniques. With this controlled redundancy only a subset of all possible transmitted messages contains valid messages. The subset is then called a code, and the valid messages are called codewords or code vectors. A good code is one in which the codewords are so “separated” that the likelihood of errors corrupting one into another is kept small.

Error detection then is simplified to answering this question: Is the received message a codeword or not? If it is a codeword, one assumes that no errors have occurred. The probability of an undetected error getting through is then the probability of sufficient errors occurring to transform the real transmitted codeword into another, apparently correct but in reality a false one.

If an error is detected, it can be corrected in principle by one of two methods:

- The recipient rejects the received message as erroneous and requests the original transmitter for a repeat transmission as shown in Fig. 3.1. This recovery by retransmission is commonplace in communication systems where it is possible. However, if propagation delays, due to distance, are large, the technique may become so inefficient as to be useless.

- The recipient corrects the errors by finding the valid codeword “nearest” to the received message, on the assumption that the nearest is the most likely, because few corrupting errors are more likely than many! It is shown in Fig. 3.2. This procedure is often called Forward Error Correction (FEC) and is one of the principal topics of this thesis.
So error correction can be handled in two ways:

a. Error correction by retransmission (ARQ)

b. Forward Error Correction

![Fig. 3.1: Error-recovered by detection and forward error correction](image)

![Fig. 3.2: Error-recovered by detection and retransmission](image)

### 3.2 Error Detection

Although the goal of error checking is to correct errors, most of the time, we first need to detect errors. Error detection is simpler than error correction and is the first step in the error correction process.

Error detection uses the concept of redundancy, which means adding extra bits for detecting errors add the destination.

#### 3.2.1 Error Detection Parity check

The most common and least expansive mechanism for error detection is the parity check. Parity checking can be simple or two-dimensional.
Simple Parity Check: In parity check, a parity bit is added to every data unit so that the total number of 1s is even (or odd for odd-parity).

Suppose the sender wants to send the word ‘world’. In ASCII the five characters are coded as

\[
\begin{align*}
\text{w} & : 1110111 \\
\text{o} & : 1101111 \\
\text{r} & : 1110010 \\
\text{l} & : 1101100 \\
\text{d} & : 1100100
\end{align*}
\]

Each of the first four characters (w, o, r, l) has an even number of 1s, so the parity bit is a 0. The last character (d), however, has three 1s (an odd number), so the parity bit is a 1 to make the total number of 1s even. The following shows the actual bits sent (the parity bits are underlined).

\[
\begin{align*}
1110111 & \quad 1101111 & \quad 1110010 & \quad 1101100 & \quad 1100100 \\
\end{align*}
\]

Simple parity check can detect all single bit errors. It can detect burst errors only if the total number of errors in each data unit is odd.

Two-Dimensional parity check: In two-dimensional parity check, a block of bits is divided into rows and a redundant row of bits is added to the whole block.

Suppose the following block is sent

\[
\begin{align*}
10101001 & \quad 00111001 & \quad 11011101 & \quad 11100111 & \quad 10101010 \\
\end{align*}
\]

Let us assume that this block is hit by a burst noise of length 8, and some bits are corrupted.

\[
\begin{align*}
10100011 & \quad 10001001 & \quad 11011101 & \quad 11100111 & \quad 10101010 \\
\end{align*}
\]

When the receiver checks the parity bits, some of the bits do not follow the even parity rule and the whole block is discarded (the non matching bits are shown in bold).

\[
\begin{align*}
10100011 & \quad 10001001 & \quad 11011101 & \quad 11100111 & \quad 10101010 \\
\end{align*}
\]

Two dimensional parity checks increase the likelihood of detecting burst errors. A redundancy of $n$-bits can easily detect a burst error of $n$-bits. A burst error of more than $n$-bits is also detected by this method with a very high probability. There is, however, one pattern of errors that remains elusive. If two bits in one data unit are damaged and
two bits in exactly the same positions in another data unit are damaged, the checker will not detect an error. Consider, for example, two data units: 11110000 and 11000011. If the first and last bits in each of them are changed, making the data units read 01110001 and 01000010, the errors cannot be detected by this method.

3.2.2 Cyclic Redundancy Check

The most powerful, error-detecting codes are the cyclic redundancy check (CRC). Unlike the parity check, CRC is based on binary division. Given a \( k \)-bit block of bits, or massage, the transmitter generates an \((n-k)\)-bit sequence, known as a frame check sequence (FCS), such that the resulting frame, consisting of \( n \)-bits, is exactly divisible by some predetermined number. The receiver then divides the incoming frame by that number and, if there is no remainder, assumes there is no error. If there is remainder, there are some errors.

**Performance:** CRC is a very effective error detection method. If the divisor is chosen according to the rules, it has the following advantage

1. CRC can detect all burst errors that affect an odd number of bits.
2. CRC can detect all burst errors of length less than or equal to the degree of the polynomial.
3. CRC can detect, with a very high probability, burst error of length greater than the degree of the polynomial.

3.3 Error correction by retransmission

Error correction by retransmission is also called as automatic repeat request (ARQ). In error correction by retransmission, when an error is discovered, the receiver can have the sender retransmit the entire data unit.

When the error control consists of error detection only, the communication system generally needs to provide a means of alerting the transmitter that an error has been detected and that a retransmission is necessary.

There are three techniques for ARQ:

**Stop-and-wait ARQ:** It requires a half-duplex connection only, since the transmitter waits for an acknowledgment (ACK) of each transmission before it proceeds with the next transmission. If any transmission block is received in error, the receiver responds
with a negative acknowledgment (NAK), and the transmitter retransmits this block before transmitting the next in the sequence.

Continuous ARQ with pullback: Here a full duplex connection is necessary. Both the terminals are transmitting simultaneously. The transmitter is sending message data and the receiver is sending acknowledgment data. Let us assume that a NAK corresponding to corrupted message is received. In the ARQ procedure, the transmitter "pulls back" to the message in error and retransmits all message data, starting with the corrupted message.

Continuous ARQ with selective repeat: A full-duplex connection is needed. In the case, only corrupted message is repeated.

The major advantage of ARQ over forward error correction (FEC) is that error detection requires much simpler decoding equipment and much less redundancy. Also ARQ is adaptive in the sense that information is retransmitted only when errors occur. On the other hand, FEC may be desirable to place of, or in addition to, error detection, for any of the following reason:

1. A reverse channel is not available or the delay with ARQ would be excessive.
2. The retransmission strategy is not conveniently implemented.
3. The expected number of errors, without corrections, would require excessive retransmission.

For wireless application ARQ technique is inadequate for two reasons:

1. The bit error rate on a wireless link can be quite high, which would result in a large number of retransmissions.
2. In some cases especially satellite links, the propagation delay is very long compare to the transmission time of a single frame. The result is a very inefficient system.

The common approach to retransmission is to retransmit the frame in error plus all subsequent frames. With a long data link, an error is a single frame necessitates retransmitting many frames.

3.4 Forward Error Correction

In forward error correction (FEC), a receiver can use an error-correcting code, which automatically corrects certain errors. In theory, it is possible to correct error automatically. Error correcting-codes, however, are more sophisticated than error detection codes and require more redundant bits.
3.4.1 Shannon's Information theory

In 1948, Shannon showed that if \( f(t) \) is produced by a discrete source or a sampled analog source, and then it is possible to achieve reliable communications with finite bandwidth [1]. The key to achieving the goal of reliable communications with finite bandwidth is the proper use of coding. Furthermore, he made the distinction between source coding and channel coding and showed that the two operations could be separated with no loss in fidelity.

The symbols are passed through a source encoder, which removes redundancy from a set of \( I \) consecutive symbols and produces a message \( m \) composed of \( k \) \( q \)-ary symbols. The rate of the source encoder is \( 1/k \) while the rate of the channel encoder is \( k/n \). The overall source rate is \( R = 1/n \) bits per code symbol. The codeword \( x \) is transmitted over a channel with capacity \( C \) bits per channel use. The channel capacity is a quantity that depends on the particular channel under consideration. The received version of the codeword \( y \) is first passed through a channel decoder, then through a source decoder.

The channel coding theorem state that, it is possible to achieve reliable communication through proper coding if the source rate is less than the channel capacity, \( R < C \). The term "reliable communication" does not imply zero probability of error, but rather that the error probability can be made arbitrarily low. The channel coding proceeds by considering a collection of \( q^k \) codewords chosen at random from the set of \( q^n \) possible codewords. Then it is shown that as the length \( n \) of the codeword approaches infinity, an arbitrarily low probability of error can be maintained for any rate \( R \) less than the capacity \( C \). Thus, a random code can achieve channel capacity for sufficiently long block lengths. However, random codes of sufficient length are not practical to implement. Practical codes behave much differently than randomly chosen codes and are usually unable to achieve the channel capacity bound.

3.4.2 Coding basics

The channel coding theorem of Shannon showed that if long random codes are used, reliable communications can take place at the minimum required signal to noise ratio. However, truly random codes are not practical to implement. Codes must possess some structure in order to have computationally tractable encoding and decoding algorithms.
Cyclic Block codes, convolution codes, and Turbo codes all have an underlying structure, which makes them practically realizable.

3.4.2.1 Block codes

Let a block code $C$ be a list of $T$ codewords $x_i$, $i = 1, \ldots, T$, each $n$-tuple with entries from the Galois field with $q$ members, denoted $GF(q)$. For binary code, $q = 2$ and $T = 2^k$. The rate of such a code is $r = k/n$. The encoding process consists of breaking up the data into messages, and then performing a one-to-one mapping of messages $m$ to codewords $x$. In linear Block codes, which have an encoding process that can be described by the matrix multiplication

$$x = mG,$$

where $G$ is the $k \times n$ generator matrix and the matrix multiplication is over the field $GF(2)$. Linear codes possess the property that the sum of two codewords also a codeword, and thus all linear Block codes must contain the all-zeros codeword.

An important figure of merit for comparing codes is the minimum distance of the code, denoted $d_{\text{min}}$. The minimum distance is smallest Hamming distance between any two codewords, where the Hamming distance is the minimum Hamming weight of all codewords except the all-zero codewords, where the Hamming weight of a codeword is simply the number of non-zero elements in the codeword. A code with minimum distance $d_{\text{min}}$ is capable of correcting all codewords with $t$ or fewer errors, where

$$t = \left\lfloor \frac{d_{\text{min}} - 1}{2} \right\rfloor$$

3.4.2.2 Cyclic Codes

A code is cyclic if any cyclic shift of a codeword produces another valid codeword. That is, $C$ is cyclic if for every codeword $x = (x_0, x_1, \ldots, x_{n-1}) \in C$, there is also a codeword $x' = (x_{n-1}, \ldots, x_0, x_1) \in C$. The generator matrix of a cyclic code has the form

$$G = \begin{bmatrix}
g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\
0 & g_0 & g_1 & \cdots & g_{n-k} & \cdots & 0 \\
0 & 0 & g_0 & g_1 & \cdots & g_{n-k} & \cdots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & g_0 & g_1 & \cdots & g_{n-k} \\
g_0 & g_1 & \cdots & g_{n-k} & \cdots & \cdots & \cdots \\
\end{bmatrix}$$

39
A cyclic code can be compactly specified by a generator polynomial \( g(D) = g_0 + g_1 D + \cdots + g_n D^n \). Likewise the message can be expressed as message polynomial \( m(D) \) and the codeword can be expressed as a code polynomial \( x(D) \). The encoding operation then becomes the polynomial multiplication \( x(D) = m(D) g(D) \). Multiplication of polynomials is equivalent to convolving the polynomial coefficients, and the encoding operation can also be expressed as

\[
x_i = \sum_{j=0}^{n-k} m_{i-j} g_j
\]

Convolution of sequences of discrete-valued symbols can be implemented by a linear shift register network, and thus the encoder for cyclic codes is extremely simple. A sample cyclic encoder using a linear shift register is shown in the Fig. 3.3.

**Systematic Code:** A code is said to be systematic if the message \( m \) is contained within the codeword \( x \). The generator matrix of systematic codes can be partitioned as

\[
\tilde{G} = \begin{bmatrix} P & I_k \end{bmatrix},
\]

where \( I_k \) is the \( k \times k \) identity matrix and \( P \) is a \( (k \times n-k) \) matrix. The parity check matrix associated with a systematic code is

\[
\tilde{H} = \begin{bmatrix} I_{n-k} & P^T \end{bmatrix},
\]

where \( P^T \) is the transpose of \( P \).

Any code can be made systematic without affecting its minimum distance by performing row reduction on \( G \). Cyclic codes can be made systematic by calculating the remainder \( r(D) \) of the polynomial division \( D^{n-k} m(D) / g(D) \). Then encoding operation then becomes \( x(D) = D^{n-k} m(D) - r(D) \). The remainder of the polynomial division can be found using a shift register network with feedback.
3.5 Convolutional Codes

Convolutional Codes are similar to cyclic codes in the sense that the encoder can be implemented using a linear shift register network. However, Convolutional codes differ from cyclic codes in two fundamental ways. First, cyclic codes are Block codes and thus have a fixed block size, which implies a fixed size for the message and codeword. Convolutional codes remove the requirement of fixed block sizes by providing for the continuous encoding of input streams. Second, if a cyclic encoder were considered to be a linear system or "black box", it would be characterized as a single-input, single-output system. Convolutional encoders on the other hand can be thought of as multi-input, multi-output systems. The input message \( m \) is first split into \( k \) streams \( m^{(p)} = (m_{p}, m_{p+k}, m_{p+2k}, \ldots), p \in \{0, \ldots, k-1\} \). The \( k \) input streams are passed through a Convolutional encoder, which produces \( n \) output streams \( x^{(q)}, q \in \{0, \ldots, n-1\} \). The output streams are multiplexed to form the codeword \( x = (x_{0}^{(0)}, \ldots, x_{0}^{(n-1)}, x_{1}^{(0)}, \ldots, x_{1}^{(n-1)}, \ldots) \).

Like Block codes, the rate of Convolutional codes is \( r = k/n \). However, the values \( k \) and \( n \) now have a different meaning. \( k \) is the number of input streams, while \( n \) is the number of output streams. While cyclic codes could be completely specified by a single generator polynomial \( g(D) \), Convolutional codes must be specified by a set of \( n \times k \) generator polynomials \( g_{f}^{(p,q)}(D) \). There are \( k \) shift registers in the encoder, one for each input stream. The length of the shift register associated with input \( p \) is denoted \( M_{p} \), and the total memory in the encoder is

\[
M_{c} = \sum_{p=0}^{k-1} M_{p}
\]

The constraint length \( k_{c} \) of the encoder is the maximum number of bits in a single output stream that can be affected by any input bit, and is taken to be

\[
k_{c} = 1 + \max_{p} M_{p}.
\]

Encoding proceeds by first convolving the coefficients of each generator polynomial \( g_{f}^{(p,q)} \) with the corresponding input \( (m_{j}^{(p)}) \)

\[
v_{j}^{(pq)} = \sum_{j=0}^{M_{p}} m_{j}^{(p)} g_{f}^{(p,q)}
\]
diagram can unambiguously specify the operation of a Convolutional encoder. State diagrams consist of a set of nodes $S_j, j \in (0, \cdots 2^M - 1)$, corresponding to the possible internal states of the encoder. A sample state diagram for the encoder of Fig. 3.4 is shown in Fig. 3.5.

![State diagram for the Convolutional code](image)

**Fig. 3.5: State diagram for the Convolutional code**

The state diagram can be expanded into a trellis diagram, which explicitly shows the passage of time. The nodes in a trellis diagram correspond not only to a particular state, but also to a particular discrete time step. The nodes are labeled $S_{ji}$, where $j$ is the state and $i$ is the time index. Branches connect nodes at time $i$ with nodes at time $i+1$. Trellis diagram for the encoder of Fig. 3.4 is shown in Fig. 3.6.

![Trellis diagram for Convolutional code](image)

**Fig. 3.6: Trellis diagram for Convolutional code**
For Convolutional codes, every codeword is associated with a unique path through the trellis. The path begins at state $S_{0,0}$ and ends in state $S_{0,L}$. The smallest Hamming distance between two distinct paths through the trellis is called minimum free distance, and is denoted as $d_{\text{free}}$. In this example, $d_{\text{free}} = 5$. The minimum free distance for Convolutional codes is very similar to the minimum Hamming distance for Block codes. Unlike a block code, the length of a Convolutional codeword is not fixed. However, for many applications it is desirable to force the Convolutional codeword to be limited to a maximum length. This can be done by a process called trellis termination. In order for the codeword to terminate at state $S_{0,L}$, the last $M_c$ bits input to the shift register must be zeros. Thus, the trellis can be forced back to the all-zeros state by setting the last $M_c$ bits of the input message to zero. Trellis termination is common for frame oriented communication systems, such as the cellular standards IS-95, IS-54, IS-136 and GSM. Trellis termination actually turns a Convolutional code into a block code with rate $r = k/(L-M_c)/(nL)$. Note that the rate of a terminated Convolutional code is less than the rate $r = k/n$ of the encoder. The rate reduction is usually characterized by the fractional rate loss, defined by [62]

$$\xi = \frac{r-r'}{r}$$

$$= \left(\frac{n}{k}\right)\left(\frac{k}{n} - \frac{k(L-M_c)}{nL}\right)$$

$$= \frac{M_c}{L},$$

where $r$ is the rate of the Convolutional encoder and $r'$ is the effective rate of the terminated code. If $L$ gets large, the fractional rate loss becomes negligible.

The trellis representation is not limited only to Convolutional codes. In fact, a trellis can represent any linear block code. Block codes have trellises that are, in general time varying. However, some classes of Block codes, such as cyclic codes, have time-invariant trellises. The number of states for a block code trellis is at most $2^{n-k}$, which for many codes can be very large. Historically, trellises have not often been used to represent Block codes due to the irregular structure and large number of states. However, with the recent introduction of Turbo codes composed of simple block component codes, there is a renewed interest in representing Block codes by trellises.
3.5.2 Recursive Systematic Convolutional Codes

Just as cyclic codes, Convolutional codes is made systematic without reducing the minimum free distance. A rate 1/2 Convolutional code is made systematic by first calculating the remainder $r(D)$ of the polynomial division $m(D)/g^0(D)$. The parity output polynomial is then found by the polynomial multiplication $x^{(1)}(D) = r(D) \times g^1(D)$, and the systematic output is simply $x^{(0)}(D) = m(D)$. The remainder $r(D)$ can be found using a shift register network with feedback, in which case $g^0(D)$ is called the feedback polynomial and $g^1(D)$ is called the feedforward polynomial. Codes generated in this manner are called Recursive Systematic Convolutional (RSC) codes. RSC encoding proceeds by first computing the feedback variable

$$r_i = m_i + \sum_{j=1}^{M_r} r_{i-j}g^0_j$$

Parity output is

$$x_i^{(1)} = \sum_{j=1}^{M_r} r_{i-j}g^{(1)}_j$$

While conventional Convolutional encoders are finite impulse response filters, RSC encoders are infinite impulse response (IIR) filters. An example RSC is shown in Fig. 3.7.

![Fig. 3.7: A RSC Turbo encoder with generators $G(D) = [1, 15/13, 17/21]$](image)
Unless otherwise specified, the "Convolutional code" refers to conventional non-systematic Convolutional codes, while "RSC code" refers to recursive systematic Convolutional codes.

RSC encoders are finite state machine and can be represented by state and trellis diagrams. The trellis diagram for this code is shown in Fig. 3.8.

![Trellis diagram](image)

**Fig. 3.8:** Trellis diagram for the RSC code of Fig. 3.7

The state and trellis diagrams for the RSC code are almost identical to those for the conventional Convolutional code. With conventional Convolutional codes, the input bits labeling the two branches entering any node are the same. However, with RSC codes, the input bits labeling the two branches entering any node are complements of one another. Since the structure of the trellis and the output bits labeling the branches remain the same when the code is made systematic, the minimum free distance remains the same.

With conventional Convolutional codes, the trellis can be forced back to the all-zeros state by padding the message with $M_c$ zeros. Because of its infinite impulse response, the same cannot be said of RSC codes. In order to terminate the trellis of a TSC code, the message inputs $m_i$ must be such that both sides of equation (3.1) equal zero. Thus, the last $M_c$ bits of the input message must satisfy
3.5.3 Decoding Algorithm of Convolutional Codes

The first decoding algorithm of Convolutional Codes was the sequential decoder of Wozencraft and Reiffen in 1961 [64]. But it was not until the introduction of the Viterbi Algorithm in 1967 [65] that an optimal solution became practical. After the development of the Viterbi Algorithm, Convolutional coding began to see extensive application in communication systems. We will concentrate our analysis on the Viterbi Algorithm.

Decoding may be two types, hard-decision and soft-decision decoding. They refer to the type of quantization used on the received bits. Hard-decision decoding uses 1-bit quantization on the received channel values. Soft-decision decoding uses multi-bit quantization on the received channel values. For the ideal soft-decision decoding (infinite-bit quantization), the received channel values are directly used in the channel decoder. Fig. 3.9 shows hard and soft decision decoding.

Fig. 3.9: Hard- and Soft-decision decoding.
3.5.3.1 Hard-Decision Viterbi Algorithm

For a Convolutional code, the input sequence $x$ is “convoluted” to the encoded sequence $c$. Sequence $c$ is transmitted across a noisy channel and the received sequence $r$ is obtained. The Viterbi Algorithm computes a maximum likelihood (ML) estimate on the estimated code sequence $y$ from the received sequence $r$ such that it maximizes the probability $p(r|y)$ that sequence $r$ is received conditioned on the estimated code sequence $y$. Sequence $y$ must be one of the allowable code sequences and cannot be any arbitrary sequence. Figure 3.10 shows the described system structure.

![Diagram of Convolutional code system](image)

**Fig. 3.10:** Convolutional code system.

For a rate $r$ Convolutional code, the encoder inputs $k$ bits in parallel and outputs $n$ bits in parallel at each time step. The input sequence is denoted as

$$x = x_0(1), x_0(2), \ldots, x_0(k), x_1(1), \ldots x_L+m-1(k).$$

The coded sequence is denoted as

$$c = c_0(1), c_0(2), \ldots, c_0(n), c_1(1), \ldots c_L+m-1(n),$$

where $L$ denotes the length of input information sequence and $m$ denotes the maximum length of the shift registers. Additional $m$ zero bits are required at the tail of the information sequence to take the Convolutional encoder back to the all-zero state. It is required that the encoder start and end at the all-zero state. The subscript denotes the time index while the superscript denotes the bit within a particular input $k$-bit or output $n$-bit block. The received and estimated sequences $r$ and $y$ can be described similarly as
\[ r = r_0(1), r_0(2), \ldots, r_0(n), \eta(1), \ldots, \eta(n), \ldots, r_{L+m-1}(n) \]

\[ y = y_0(1), y_0(2), \ldots, y_0(n), y_1(1), \ldots, y_1(n), \ldots, y_{L+m-1}(n) \]

For ML decoding, the Viterbi Algorithm selects \( y \) to maximize \( p(r \mid y) \). The channel is assumed to be memoryless, and thus the noise process affecting a received bit is independent from the noise process affecting all of the other received bits. From probability theory, the probability of joint, independent events is equivalent to the product of the probabilities of the individual events. Thus,

\[
p(r \mid y) = \prod_{i=0}^{L+m-1} [p(r_i^{(1)} \mid y_i^{(1)}) p(r_i^{(2)} \mid y_i^{(2)}) \ldots p(r_i^{(n)} \mid y_i^{(n)})] \]

This equation is called the likelihood function of \( y \) given that \( r \) is received. The estimate that maximizes \( p(r \mid y) \) also maximizes \( \log(p(r \mid y)) \) because logarithms are monotonically increasing functions. Thus, a log likelihood function can be defined as

\[
\log p(r \mid y) = \sum_{i=0}^{L+m-1} \left( \sum_{j=1}^{n} \log p(r_i^{(j)} \mid y_i^{(j)}) \right)
\]

For an easier manipulation of the summations over the log function, a bit metric is defined. The bit metric is defined as

\[
M(r_i^{(j)} \mid y_i^{(j)}) = a[\log p(r_i^{(j)} \mid y_i^{(j)})] + b
\]

where \( a \) and \( b \) are chosen such that the bit metric is a small positive integer. The values of \( a \) and \( b \) are defined for binary symmetric channel (BSC) and hard-decision decoding. Figure 3.11 shows a BSC.
Fig. 3.11: The binary symmetric channel model, where $p$ is the crossover probability.

For BSC, $a$ and $b$ can be chosen in two distinct ways. For the conventional way, they can be chosen as

$$a = \frac{1}{\log p - \log(1 - p)}$$

And

$$b = -\log(1 - p)$$

The resulting bit metric is then

$$M(r^{(j)}_i | y^{(j)}_i) = \frac{1}{[\log p - \log(1 - p)]} \log p(r^{(j)}_i | y^{(j)}_i) - \log(1 - p))$$

From the BSC model, it is clear that $p(r^{(j)}_i | y^{(j)}_i)$ can only take on values $p$ and $1 - p$. Table 3.1 shows the resulting bit metric.

Table 3.1: Conventional Bit Metric Values

| $M(r^{(j)}_i | y^{(j)}_i)$ | Received Bit $r^{(j)}_i=0$ | Received Bit $r^{(j)}_i=1$ |
|---------------------------|-----------------------------|-----------------------------|
| Decoded Bit               |                             |                             |
| $y^{(j)}_i = 0$           | 0                           | 1                           |
| Decoded Bit               |                             |                             |
| $y^{(j)}_i = 1$           | 1                           | 0                           |
This bit metric shows the cost of receiving and decoding bits. For example, if the decoded bit $y^{(j)}_i = 0$ and the received bit $r^{(j)}_i = 0$, then the cost $M(r^{(j)}_i | y^{(j)}_i) = 0$. However, if the decoded bit $y^{(j)}_i = 0 = 0$ and the received bit $r^{(j)}_i = 1$, then the cost $M(r^{(j)}_i | y^{(j)}_i) = 1$. As it can be seen, this is related to the Hamming distance and is known as the Hamming distance metric. Thus, the Viterbi Algorithm chooses the code sequence $y$ through the trellis that has the smallest cost/Hamming distance relative to the received sequence $r$.

Furthermore, for an arbitrary channel (not necessarily BSC), the values of $a$ and $b$ are found on a trial and error basis to obtain an acceptable bit metric. From the bit metric, a path metric is defined. The path metric is defined as

$$ M(r | y) = \sum_{i=0}^{L-m} \left( \sum_{j=1}^{n} M(r^{(j)}_i | y^{(j)}_i) \right) $$

It indicates the total cost of estimating the received bit sequence $r$ with the decoded bit sequence $y$ in the trellis diagram. Furthermore, the $k$–th branch metric is defined as

$$ M(r_k | Y_k) = \sum_{j=1}^{n} M(r^{(j)}_i | y^{(j)}_i) $$

and the $k$–th partial path metric is defined as

$$ M^k(r | y) = \sum_{i=0}^{k} M(r_i | y_i) = \sum_{i=0}^{k} \left( \sum_{j=1}^{n} M(r^{(j)}_i | y^{(j)}_i) \right) $$

The branch metric indicates the cost of choosing a branch from the trellis diagram. The $k$–th partial path metric indicates the cost of choosing a partially decoded bit sequence $y$ up to time index $k$.

The Viterbi Algorithm utilizes the trellis diagram to compute the path metrics. Each state (node) in the trellis diagram is assigned a value, the partial path metric. The partial path metric is determined from state $s = 0$ at time $t = 0$ to a particular state $s = k$ at time $t \geq 0$. At each state, the “best” partial path metric is chosen. The “best” partial path metric is the larger metric. The selected metric represents the survivor path and the
remaining metrics represent the nonsurvivor paths. The survivor paths are stored while the nonsurvivor paths are discarded in the trellis diagram. The Viterbi Algorithm selects the single survivor path left at the end of the process as the ML path. Trace-back of the ML path on the trellis diagram would then provide the ML decoded sequence.

The Hard-Decision Viterbi Algorithm (HDVA) can be implemented as follows [42]:

\[ S_{k,t} \] is the state in the trellis diagram that corresponds to state \( S_k \) at time \( t \). Every state in the trellis is assigned a value denoted as \( V(S_{k,t}) \).

1. (a) Initialize time \( t = 0 \).
   (b) Initialize \( V(S_{0,0}) = 0 \) and all other \( V(S_{k,t}) = \infty \).

2. (a) Set \( t = t + 1 \).
   (b) Compute the partial path metrics for all paths going to state \( S_k \) at time \( t \).

First, find the \( i^{th} \) branch metric \( M(r_i | y_i) = \sum_{j=1}^{L} M(r_i^{(j)} | y_i^{(j)}) \). This is calculated from the Hamming distance \( \sum_{j=1}^{L} | r_i^{(j)} - y_i^{(j)} | \). Second, compute the \( i^{th} \) partial path metric \( M'(r_i | y_i) = \sum_{j=0}^{L} M(r_i | y_i) \). This is calculated from \( V(S_{k,t-1}) + M(r_i | y_i) \).

3. (a) Set \( V(S_{k,t}) \) to the "best" partial path metric going to state \( S_k \) at time \( t \).
   Conventionally, the "best" partial path metric is the partial path metric with the smallest value.
   (b) If there is a tie for the "best" partial path metric, then any one of the tied partial path metric may be chosen.

4. Store the "best" partial path metric and its associated survivor bit and state paths.

5. If \( t < L + m - 1 \), return to Step 2.

The result of the Viterbi Algorithm is a unique trellis path that corresponds to the ML codeword.

3.5.3.2 Soft-Decision Viterbi Algorithm

In soft-decision decoding, the receiver does not assign a zero or a one (single-bit quantization) to each received bit but uses multi-bit or infinite-bit quantized values [42].
ideally, the received sequence $r$ is infinite-bit quantized and is used directly in the soft-decision Viterbi decoder. The soft-decision Viterbi Algorithm is similar to its hard-decision algorithm except that squared Euclidean distance is used in the metric instead of Hamming distance. The details can be found in [42].

3.5.4 Performance Analysis of Convolutional Code

The performance of Convolutional codes can be quantified through analytical means or by computer simulation. The analytical approach is based on the transfer function of the Convolutional code, which is obtained from the state diagram. The process of obtaining the transfer function and other related performance measures are described below.

3.5.4.1 Transfer Function of Convolutional Code

The analysis of Convolutional codes is generally difficult to perform because traditional algebraic and combinatorial techniques cannot be applied. These heuristically constructed codes can be analyzed through their transfer functions. By utilizing the state diagram, the transfer function can be obtained. With the transfer function, code properties such as distance properties and the error rate performance can be easily calculated. To obtain the transfer function, the following rules are applied:

1. Break the all-zero (initial) state of the state diagram into a start state and an end state. This will be called the modified state diagram.
2. For every branch of the modified state diagram, assign the symbol $D$ with its exponent equal to the Hamming weight of the output bits.
3. For every branch of the modified state diagram, assign the symbol $J$.
4. Assign the symbol $N$ to the branch of the modified state diagram, if an input bit 1 causes the branch transition.

For the state diagram in Figure 3.4, the modified state diagram is shown in Fig. 3.12.
Fig. 3.12: The modified state diagram of Figure 3.4 where $S_a$ is the start state and $S_e$ is the end state [67].

Nodal equations are obtained for all the states except for the start state in Fig. 3.12. These results are

\[
\begin{align*}
S_b &= NJD^3 S_e + NJS_c \\
S_e &= JDS_b + JDS_d \\
S_d &= NJDS_b + NJDS_d \\
S_N &= JD^2 S_e 
\end{align*}
\]

The transfer function is defined to be

\[
T(D, N, J) = \frac{S_{end}(D, N, J)}{S_{start}(D, N, J)}
\]

From figure 3.12,

\[
T(D, N, J) = \frac{S_e}{S_a}
\]

By substituting and rearranging,

\[
T(D, N, J) = \frac{N J^3 D^5}{1 - (NJ + NJ^2)D}
\]

\[
= NJ^3 D^5 + (N^2 J^4 + N^2 J^5)D^6 + (N^3 J^5 + 2N^3 J^6 + N^3 J^7)D^7 + \ldots
\]

(Expanded polynomial form)
3.5.4.2 Distance Properties

The free distance between a pair of Convolutional codewords is the Hamming distance between the pair of codewords. The minimum free distance, $d_{free}$, is the minimum Hamming distance between all pairs of complete Convolutional codewords and is defined as

$$d_{free} = \min \{d(y_1, y_2) \mid y_1 \neq y_2\}$$

$$= \min \{w(y) \mid y \neq 0\}$$

where $d(\cdot, \cdot)$ is the Hamming distance between a pair of Convolutional codewords and $w(\cdot)$ is the Hamming distance between a Convolutional codeword and the all-zero codeword (the weight of the codeword). The minimum free distance corresponds to the ability of the Convolutional code to estimate the best decoded bit sequence. As $d_{free}$ increases, the performance of the Convolutional code also increases. This characteristic is similar to the minimum distance for Block codes. From the transfer function, the minimum free distance is identified as the lowest exponent of $D$. From the above transfer function for Figure 3.12, $d_{free} = 5$. Also, if $N$ and $J$ are set to 1, the coefficients of $D_j'$s represent the number of paths through the trellis with weight $D_j$. More information about the codeword is obtained from observing the exponents of $N$ and $J$. For a codeword, the exponent of $N$ indicates the number of 1s in the input sequence, and the exponent of $J$ indicates the length of the path that merges with the all-zero path for the first time [21].

3.5.4.3 Degree of Quantization

For soft-decision Viterbi decoding, the degree of the quantization on the received signal can affect the decoder performance. The performance of the Viterbi decoder improves with higher bit quantization. It has been found that an eight-level quantizer degrades the performance only slightly with respect to the infinite bit quantized case [21].
Chapter 4

Turbo Codes

4.1 Concatenated codes

A concatenated code is one that uses two levels of coding to achieve the desired error performance. Concatenated coding schemes were first proposed by Forney [6] as a method for achieving large coding gains by combining two or more relatively simple building block or component codes. The resulting codes had the error-correction capability of much longer codes, and they were endowed with a structure that permitted relatively easy to moderately complex decoding. A serial concatenation of codes is most often used for power-limited systems such as transmitters on deep-space probes. The most popular of these schemes consists of a Reed-Solomon outer code followed by a Convolutional inner code. The serial concatenated codes is shown in the Fig. 4.1. Operation of such systems with $E_b / N_0$ in the range 2.0 to 2.5 dB to achieve BER of $10^{-5}$ is feasible with practical hardware. The outer R-S code is formed from m-bit segments of the binary data stream. The performance of such a R-S code depends only on the number of symbol errors in the block. The code undisturbed by burst errors within an m-bit symbol. That is, for a given symbol error, the R-S code performance is the same whether the symbol error is due to one bit being in error or m-bits being in error. However, the concatenated system performance is severely degraded by correlated errors among successive symbols. Hence interleaving between codes at the symbol level needs to be provided. A Turbo code, which is the parallel concatenation of two or more codes [5], can be thought of as a refinement for the concatenated encoding structure plus an iterative algorithm for decoding the associated code sequence as shown in Fig. 4.2.

Turbo codes draws much attention to the application engineers because of the following reasons. Block and Convolutional codes are highly structured; they have encoders and decoders that can be implemented with reasonable complexity. However, the very same structure that facilitates practical implementation results in performance that is significantly inferior to the random coding bounds predicted by Shannon.
Turbo codes produces a nearly random codes. The total code rate for parallel concatenation is

\[ r_{\text{tot}} = \frac{k}{n_1 + n_2} \]
where $k$ is the information bit size, $n_1$ and $n_2$ are the number of outputs from the 1st and 2nd encoders.

For both serial and parallel concatenation schemes, an interleaver is often used between the encoders to improve burst error correction capacity and to increase the randomness of the code.

Turbo codes use the parallel-concatenated encoding scheme. However, the Turbo code decoder is based on the serial concatenated decoding scheme. The serial concatenated decoders are used because they perform better than the parallel concatenated decoding scheme due to the fact that the serial concatenation scheme has the ability to share information between the concatenated decoders whereas the decoders for the parallel concatenation scheme primarily decode independently.

While they still contain enough structure too admit practical encoding and decoding algorithms, Turbo codes possess some random-like properties. As a consequence, the performance of Turbo codes comes much closer to the Shannon bound than does the performance of more traditional block and Convolutional codes.

### 4.2 Turbo Code Encoder

The fundamental Turbo code encoder is built using two identical recursive systematic Convolutional (RSC) codes with parallel concatenation. An RSC encoder is typically $r = 1/2$ and is termed a component encoder. An interleaver separates the two component encoders. Only one of the systematic outputs from the two component encoders is used, because the systematic output from the other component encoder is just a permuted version of the chosen systematic output. Fig. 4.3 shows the fundamental Turbo code encoder.

![Fig. 4.3: Fundamental Turbo code encoder.](image)
Figure 4.3 shows \( r = 1/3 \) Turbo code encoder. The first RSC encoder outputs the systematic \( c_1 \) and recursive Convolutional \( c_2 \) sequences while the second RSC encoder discards its systematic sequence and only outputs the recursive Convolutional \( c_3 \) sequence.

4.2.1 Recursive Systematic Convolutional (RSC) Encoder

The recursive systematic Convolutional (RSC) encoder is obtained from the nonrecursive nonsystematic (conventional) Convolutional encoder by feeding back one of its encoded outputs to its input.

The RSC encoder of this conventional Convolutional encoder is represented as \( G = [1, g_1/g_2] \), where 1 denotes the systematic output, \( g_2 \) denotes the feedback polynomial, and \( g_1 \) is the feedforward polynomial. Figure 4.4 shows the resulting RSC encoder.

![Fig. 4.4: The RSC encoder with 1/2 Code rate](image)

It was suggested in [28], that good codes could be obtained by setting the feedback of the RSC encoder to a primitive polynomial, because the primitive polynomial generates maximum-length sequences, which adds randomness to the Turbo code.
4.2.2 Trellis Termination

For the conventional Convolutional encoder, the trellis is terminated by inserting \( m = K - 1 \) additional zero bits after the input sequence. These additional bits drive the conventional Convolutional encoder to the all-zero state (trellis termination). However, this strategy is not possible for the RSC encoder due to the feedback path. The additional termination bits for the RSC encoder depend on the state of the encoder and are very difficult to predict [41]. Furthermore, even if the termination bits for one of the component encoders are found, the other component encoder may not be driven to the all zero state with the same \( m \) termination bits due to the presence of the interleaver between the component encoders. Figure 4.5 shows a simple strategy.

![Fig. 4.5: Trellis termination strategy for RSC encoder.](image)

For encoding the input sequence, the switch is turned on to position \( A \) and for terminating the trellis, the switch is turned on to position \( B \).

4.3 Turbo Code Decoder

The Turbo code decoder is based on a modified Viterbi Algorithm that incorporates reliability values to improve decoding performance. First, the concept of reliability for Viterbi decoding is introduced. Then, the metric that will be used in the
modified Viterbi Algorithm for Turbo code decoding is described. Finally, the decoding algorithm and implementation structure for a Turbo code are presented.

### 4.3.1 Principle of the General Soft-Output Viterbi Decoder

The Viterbi Algorithm produces the ML output sequence for Convolutional codes. This algorithm provides optimal sequence estimation for one-stage Convolutional codes. For concatenated (multistage) Convolutional codes, there are two main drawbacks to conventional Viterbi decoders. First, the inner Viterbi decoder produces bursts of bit errors, which degrades the performance of the outer Viterbi decoders. Second, the inner Viterbi decoder produces hard decision outputs, which prohibits the outer Viterbi decoders from deriving the benefits of soft decisions. Both of these drawbacks can be reduced and the performance of the overall concatenated decoder can be significantly improved if the Viterbi decoders are able to produce reliability (soft-output) values. The reliability values are passed on to subsequent Viterbi decoders as a priori information to improve decoding performance. This modified Viterbi decoder is referred to as the soft-output Viterbi Algorithm (SOVA) decoder. Fig. 4.6 shows a concatenated SOVA decoder.

![Fig. 4.6: A concatenated SOVA decoder](image)

*Fig. 4.6: A concatenated SOVA decoder (y is the received channel values, u is the hard decision output values and L is the associated reliability values)*

### 4.3.2 Reliability of the General SOVA Decoder

The reliability of the SOVA decoder is calculated from the trellis diagram as shown in Figure 4.7.
In Fig. 4.7, a 4-state trellis diagram is shown. The solid line indicates the survivor path (assumed here to be part of the final ML path) and the dashed line indicates the competing (concurrent) path at time 1 for state 1. For the sake of brevity, survivor and competing paths for other nodes are not shown. The label $S_{1,t}$ represents state 1 and time $t$. Also, the labels $\{0,1\}$ shown on each path indicate the estimated binary decision for the paths. The survivor path for this node is assigned an accumulated metric $V_s(S_{1,t})$ and the competing path for this node is assigned an accumulated metric $V_c(S_{1,t})$. The fundamental information for assigning a reliability value $L(t)$ to node $S_{1,t}$'s survivor path is the absolute difference between the two accumulated metrics, $L(t) = |V_s(S_{1,t}) - V_c(S_{1,t})|$. The greater this difference, the more reliable is the survivor path. For this reliability calculation, it is assumed that the survivor-accumulated metric is always "better" than the competing accumulated metric. Furthermore, to reduce complexity, the reliability values only need to be calculated for the ML survivor path (assume it is known for now) and are unnecessary for the other survivor paths since they will be discarded later.
4.3.3 Soft Channel Outputs

From the system model the information bit $u$ is mapped to the encoded bits $x$. The encoded bits $x$ are transmitted over the channel and received as $y$. From this system model, the log-likelihood ratio of $x$ conditioned on $y$ is calculated as

$$L(x | y) = \ln \frac{P(x = +1 | y)}{P(x = -1 | y)}$$

By using Bayes' Theorem, this log-likelihood ratio is equivalent to

$$L(x | y) = \ln \left( \frac{p(y | x = +1)P(x = +1)}{p(y | x = -1)P(x = -1)} \right)$$

$$= \ln \frac{p(y | x = +1)}{p(y | x = -1)} + \ln \frac{P(x = +1)}{P(x = -1)}$$

The channel model is assumed to be flat fading with Gaussian noise. By using the Gaussian definition of pdf, we get

$$f(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-m)^2}{2\sigma^2}}$$

where $m$ is the mean and the $\sigma^2$ is the variance. It can be shown that

$$\ln \frac{p(y | x = +1)}{p(y | x = -1)} = \ln \frac{e_{E_b,2\sigma^2}}{e_{E_b,2\sigma^2}}$$

$$= \ln \frac{e_{E_b,2\sigma^2}}{e_{E_b,2\sigma^2}}$$

$$= 4 \frac{E_b}{N_0} a y,$$

where $\frac{E_b}{N_0}$ is the signal to noise ratio per bit (directly related to the noise variance) and $a$ is the fading amplitude. For nonfading Gaussian channel, $a = 1$.

The log-likelihood ratio of $x$ conditioned on $y$, $L(x | y)$, is equivalent to

$$L(x | y) = L_c y + L(x)$$

where $L_c$ is defined to be the channel reliability

$$L_c = 4 \frac{E_b}{N_0} a$$
Thus, $L(x | y)$ is just the weighted received value $L_{c} y$ summed with the log-likelihood value of $x$.

### 4.3.4 SOYA Component Decoder for a Turbo Code

The SOYA component decoder estimates the information sequence using one of the two encoded streams produced by the Turbo code encoder. Fig. 4.8 shows the inputs and outputs of the SOYA component decoder.

- **SOYA component decoder**:
  - **Inputs**: $L(u)$ and $L_{c} y$
  - **Outputs**: $u'$ and $L(u')$

Fig. 4.8: SOYA component decoder.

The SOYA component decoder processes the (log-likelihood ratio) inputs $L(u)$ and $L_{c} y$, where $L(u)$ is the a-priori sequence of the information sequence $u$ and $L_{c} y$ is the weighted received sequence. The sequence $y$ is received from the channel. However, the sequence $L(u)$ is produced and obtained from the preceding SOYA component decoder. If there is no preceding SOYA component decoder then there are no a-priori values. Thus, the $L(u)$ sequence is initialized to the all-zero sequence. A similar concept is also shown at the beginning of the chapter in Figure 5.1. The SOYA component decoder produces $u'$ and $L(u')$ as outputs where $u'$ is the estimated information sequence and $L(u')$ is the associated log-likelihood ratio ("soft" or L-value) sequence.

The SOYA component decoder operates similarly to the Viterbi decoder except the ML sequence is found by using a modified metric. This modified metric, which incorporates the a-priori value, is derived below.

The fundamental Viterbi Algorithm searches for the state sequence $S^{(m)}$ or the information sequence $u(m)$ that maximizes the a-posteriori probability $P(S^{(m)} | y)$. For binary ($k = 1$) trellises, $m$ can be either 1 or 2 to denote the survivor and the competing paths respectively. By using Bayes' theorem, the a-posteriori probability can be expressed as
\[
P(S^{(m)} | y) = P(y | S^{(m)}) \frac{P(S^{(m)})}{P(y)}
\]

Since the received sequence \( y \) is fixed for metric computation and does not depend on \( m \), it can be discarded. Thus, the maximization results to

\[
\max_m p(y | S^{(m)}) P(S^{(m)})
\]

The probability of a state sequence terminating at time \( t \) is \( P(S_t) \). This probability can be calculated as

\[
P(S_t) = P(S_{t-1}) P(S_t)
\]

\[
= P(S_{t-1}) P(u_t)
\]

Where \( P(S_t) \) and \( P(u_t) \) denote the probability of the state and the bit at time \( t \) respectively. The maximization can then be expanded to

\[
\max_m p(y | S^{(m)}) P(S^{(m)}) = \max_m \left\{ \prod_{i=0}^{t-1} p(y_i | S_i^{(m)}, S_{i-1}^{(m)}) P(S_i^{(m)}) \right\},
\]

where \((S_{i-1}^{(m)}, S_i^{(m)})\) denotes the state transition between time \( i-1 \) and time \( i \) and \( y_i \) denotes the associated received channel values for the state transition.

After substituting and rearranging,

\[
\max_m p(y | S^{(m)}) P(S^{(m)}) = \max_m \left\{ P(S_{t-1}^{(m)}) \prod_{i=0}^{t-1} p(y_i | S_i^{(m)}, S_{i-1}^{(m)}) P(u_i^{(m)}) P(y_i | S_i^{(m)}, S_{i-1}^{(m)}) \right\}
\]

Note that

\[
P(y_i | S_i^{(m)}, S_{i-1}^{(m)}) = \prod_{j=1}^{N} p(y_{i,j} | x_{i,j}^{(m)})
\]

Thus the maximization becomes

\[
= \max_m \left\{ P(S_{t-1}^{(m)}) \prod_{i=0}^{t-1} p(y_i | S_i^{(m)}, S_{i-1}^{(m)}) P(u_i^{(m)}) \prod_{j=1}^{N} p(y_{i,j} | x_{i,j}^{(m)}) \right\}
\]

This maximization is not changed if logarithm is applied to the whole expression, multiplied by 2, and added two constants that are independent of \( m \). This leads to

\[
\max_m \left\{ M_i^{(m)} \right\} = \max_m \left\{ M_i^{(m)} + [2 \ln P(u_i^{(m)})] - C_v \right\} + \sum_{j=1}^{N} \left[ 2 \ln P(y_{i,j} | x_{i,j}^{(m)}) - C_y \right],
\]
where

\[
\frac{M^{(m)}_{t-1}}{2} = \ln \left( \prod_{i=0}^{t-1} p(y_i | S^{(m)}_{t-1}, S^{(m)}_i) \right)
\]

And for convenience, the two constants are

\[C_u = \ln P(u_t = +1) + \ln P(u_t = -1)\]

\[C_y = \ln P(y_{t,j} | x_{t,j} = +1)) + \ln P(y_{t,j} | x_{t,j} = -1))\]

After substitution of these two constants, the SOVA metric is obtained as

\[M^{(m)}_t = M^{(m)}_{t-1} + \sum_{j=1}^N x_{t,j}^{(m)} \ln \frac{p(y_{t,j} | x_{t,j} = +1)}{p(y_{t,j} | x_{t,j} = -1)} + u_t^{(m)} \ln \frac{p(u_t = +1)}{p(u_t = -1)}\]

And is reduced to

\[M^{(m)}_t = M^{(m)}_{t-1} + \sum_{j=1}^N x_{t,j}^{(m)} L_{c,j} y_{t,j} + u_t^{(m)} L(u_t)\]

For systematic codes, this can be modified to become

\[M^{(m)}_t = M^{(m)}_{t-1} + \sum_{j=1}^N x_{t,j}^{(m)} L_{c,j} y_{t,j} + \sum_{j=1}^N x_{t,j}^{(m)} L_{c,i} y_{t,i} + u_t^{(m)} L(u_t)\]

As seen from the previous equations, the SOVA metric incorporates values from the past metric, the channel reliability, and the source reliability (a-priori value).

At time \(t\), the reliability value (magnitude of the log-likelihood ratio) assigned to a node in the trellis is determined from

\[\Delta_t^0 = \frac{1}{2} | M_t^{(1)} - M_t^{(2)} |,\]

where \(\Delta_t^{\text{MEM}}\) denotes the reliability value at memorization level MEM relative to time \(t\).

The proof that this term is reliability value is given in Appendix II. This notation is similar to the notation \(L(t-MEM)\) as used before and is shown in Fig. 4.9 for discussion.
The reliability values along the survivor path for a particular node at time $t$ are denoted as $\Delta_t^{MEM}$ where $MEM = 0 \ldots t$. For this node at time $t$, if the bit on the survivor path at $MEM = k$ were the same as the associated bit on the competing path, then there would be no bit error if the competing path were chosen. Thus, the reliability value at this bit position remains unchanged. However, if the bits differ on the survivor and competing path at $MEM = k$, then there is a bit error. The reliability value at this bit error position must then be updated using the same updating procedure as described at the beginning of the chapter. As shown in Fig. 4.9, reliability updates are required for $MEM = 2$ and $MEM = 4$.

The reliability updates are performed to improve the “soft” or $L$-values. It is shown in that the “soft” or $L$-value of a bit decision is

$$L(U_i^{MEM}) = U_i^{MEM} \sum_{k=0}^{M} \Delta_t^k$$

And can be approximated to become
\[ L(u_{t-MEM}) = u_{t-MEM} \min_{k=0..MEM} \{ \Delta^k \} \]

The soft output Viterbi Algorithm (along with its reliability updating procedure) can be implemented as follows:

1. (a) Initialize time \( t = 0 \).
   (b) Initialize \( M_0^{(m)} = 0 \) only for the zero state in the trellis diagram and all other states to \(-\infty\).

2. (a) Set time \( t = t + 1 \).
   (b) Compute the metric \( M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)} L_c y_t + \sum_{j=2}^{N} x_{t,j}^{(m)} I_c y_{t,j} + u_t^{(m)} L(u_t) \)
for each state in the trellis diagram where
   \( m \) denotes allowable binary trellis branch/transition to a state.
   \( M_t^{(m)} \) is the accumulated metric for time \( t \) on branch \( m \).
   \( u_t^{(m)} \) is the systematic bit (1st bit of \( N \) bits) for time \( t \) on branch \( m \).
   \( x_{t,j}^{(m)} \) is the \( j \)-th bit of \( N \) bits for time \( t \) on branch \( m \) \( (2 \leq j \leq N) \).
   \( y_{t,j}^{(m)} \) is the received value from the channel corresponding to \( x_{t,j}^{(m)} \).
   \( I_c = 4 \frac{E_b}{N_0} \) is the channel reliability value.
   \( L(u_t) \) is the a-priori reliability value for time \( t \). This value is from the preceding decoder. If there is no preceding decoder, then this value is set to zero.

3. Find \( \max_m M_t^{(m)} \) for each state. For simplicity, let \( M_t^{(1)} \) denote the survivor path metric and \( M_t^{(2)} \) denote the competing path metric.

4. Store \( M_t^{(1)} \) and its associated survivor bit and state paths.

5. Compute \( \Delta^0_t = \frac{1}{2} | M_t^{(1)} - M_t^{(2)} | \).

6. Compare the survivor and competing paths at each state for time \( t \) and store the \( MEM \) s where the estimated binary decisions of the two paths differ.

7. Update \( \Delta_t^{MEM} \approx \min_{k=0..MEM} \{ \Delta^k \} \) for all \( MEM \) s from smallest to largest \( MEM \).
8. Go back to Step (2) until the end of the received sequence.

9. Output the estimated bit sequence $u'$ and its associated "soft" or L-value sequence $L(u') = u' \cdot \Delta$, where ($\cdot$) operator defines element-by-element multiplication operation and $\Delta$ is the final updated reliability sequence. $L(u')$ is then processed (to be discussed later) and passed on as the a-priori sequence $L(u)$ for the succeeding decoder.

4.3.5 SOVA Implementation

The iterative Turbo code decoder is composed of two concatenated SOVA component decoders. Fig. 4.10 shows the Turbo code decoder structure.

The Turbo code decoder processes the received channel bits on a frame basis. As shown in Figure 4.10, the received channel bits are demultiplexed into the systematic stream $y_1$ and two parity check streams $y_2$ and $y_3$ from component encoders 1 and 2 respectively. These bits are weighted by the channel reliability value and loaded on to the
CS registers. The registers shown in the figure are used as buffers to store sequences until they are needed. The switches are placed in the open position to prevent the bits from the next frame from being processed until the present frame has been processed.

The SOVA component decoder produces the "soft" or L-value \( L(u'_i) \) for the estimated bit \( u'_i \). The "soft" or L-value \( L(u'_i) \) can be decomposed into three distinct terms as proved in Appendix III,

\[
L(u'_i) = L(u_i) + L_{c,Y_{r1}} + L_{e}(u'_i).
\]

Here,

- \( L(u_i) \) is the a-priori value and is produced by the preceding SOVA component decoder.
- \( L_{c,Y_{r1}} \) is the weighted received systematic channel value.
- \( L_{e}(u'_i) \) is the extrinsic value produced by the present SOVA component decoder.

The information that is passed between SOVA component decoders is the extrinsic value

\[
L_{e}(u'_i) = L(u'_i) - L(u_i) - L_{c,Y_{r1}}.
\]

The a-priori value \( L(u_i) \) is subtracted out from the "soft" or L-value \( L(u'_i) \) to prevent passing information back to the decoder from which it was produced. Also, the weighted received systematic channel value \( L_{c,Y_{r1}} \) is subtracted out to remove "common" information in the SOVA component decoders.

Fig. 4.10 shows that the Turbo codes decoder is a closed loop serial concatenation of SOVA component decoders. In this closed loop decoding scheme, each of the SOVA component decoders estimates the information sequence using a different weighted parity check stream. The Turbo code decoder further implements iterative decoding to provide more dependable reliability/a-priori estimations from the two different weighted parity check streams, hoping to achieve better decoding performance. The iterative decoding algorithm of Turbo code for the \( n \)-th iteration is as follows:

1. The SOVA1 decoder inputs sequences \( \frac{E_b}{N_0} Y_1 \) (systematic), \( \frac{E_b}{N_0} Y_2 \) (parity check), and \( L_{c2}(u') \) and outputs sequences \( L_4(u') \). For the first

70
iteration, sequence $L_{e2}(u') = 0$ because there is no initial a-priori value.
(no extrinsic values from SOVA2).

2. The extrinsic information from SOVA1 is obtained by

$$L_{e1}(u') = L_1(u') - L_{e2}(u') - L_{e1}y_1,$$
where $L_c = 4 \frac{E_b}{N_0}$.

3. The sequences $4 \frac{E_b}{N_0} y_1$ and $L_{e1}(u')$ are interleaved and denoted as

$$I \left\{ 4 \frac{E_b}{N_0} y_1 \right\} \text{ and } I \left\{ L_{e1}(u') \right\}.

4. The SOVA2 decoder inputs sequences $I \left\{ 4 \frac{E_b}{N_0} y_1 \right\}$ (systematic),

$$I \left\{ 4 \frac{E_b}{N_0} y_3 \right\} \text{ (parity check that was already interleaved by the Turbo code encoder), and } I \left\{ L_{e1}(u') \right\} \text{ (a-priori information)}$$
and outputs sequences $I \left\{ L_2(u') \right\}$ and $I \{u'\}$

5. The extrinsic information from SOVA2 is obtained by

$$I \left\{ L_{e2}(u') \right\} = I \left\{ L_2(u') \right\} - I \left\{ L_{e1}(u') \right\} - I \left\{ L_{e1}y_1 \right\}.$$

6. The sequence $I \left\{ L_{e2}(u') \right\}$ and $I \{u'\}$ are deinterleaved and denoted as $L_{e2}(u')$ and $u'$. $L_{e2}(u')$ is fed back to SOVA1 as a-priori information for the next iteration and $u'$ is the estimated bits output for the $n$-th iteration.
Chapter 5

Performance of Turbo Codes

5.1 Introduction

There are several factors that influence the performance of Turbo codes. The most influential factor is the block size of the code. As will be shown in this chapter, performance is improved as block size is increased. However, as block size is increased so does decoding delay, and thus a balance must be established between acceptable performance and tolerable latency. The interleaver design is also a factor, but only at high signal to noise ratios. At low SNR, Turbo codes perform well with almost any interleaver provided that the inputs at the two RSC encoders are sufficiently uncorrelated. At higher SNR, performance is dominated by the low weight codewords, which are significantly influenced by the interleaver design. As for most other codes, performance degrades as code rate increases. If puncturing is used to increase the code rate, puncturing pattern is also a performance factor and different puncturing matrices may need to be considered. The joint design of interleaver and puncturing matrices is perhaps the most important aspect of Turbo code design. Another important factor is the choice of decoding algorithm, which will be discussed in the next two chapters. Most decoding algorithms are iterative, and therefore the number of iterations has an impact on the performance. The choice of constituent RSC encoder influences the performance of Turbo codes. Turbo codes typically use simple constituent codes with constraint lengths of 3, 4 or 5.

Another issue that affects the performance of Turbo codes is trellis termination. While terminating the trellis for conventional Convolutional and RSC codes is straightforward, this is not the case for Turbo codes. Although it is possible to represent a Turbo code by a single "supertrellis", the high complexity generally precludes such a representation. Instead, Turbo codes are represented by two trellises, one for each component code, linked by and interleaver. Termination of the entire code's supertrellis can be achieved by terminating each of the component codes' trellises. Termination of either trellis is a straightforward application of the equation and requires a tail of $M_c$ bits. It follows that both trellises can be terminated by a tail of $2M_c$ bits. However, there are two
complicating issues: (1) the tail bits are not all located at the end of the message and (2) it is difficult to compute the values of the tail bits. While half of the tail bits are used to terminate the upper encoder's trellis and are located at the end of the message, the other half are used to terminate the lower encoder's trellis and because of interleaving are dispersed throughout the message. Although this first issue leads to an awkward implementation, it does not alone cause a significant problem. The second problem is a bit thornier and can be described as an instance of the "chicken and egg" problem. The problem is that due to the equation \( m_i = \sum_{j=1}^{M} r_{i-j} g_j^{(0)} \) the tail bits for either encoder are not known until that encoder has completed the encoding of its data. But the tail bits of one encoder become data at the other encoder, and therefore influence the value of the other encoder's tail bits. Thus, in order to compute the tail for the first encoder, the tail for the second encoder must first be known and vice versa. This problem makes it difficult to compute a tail that terminates both trellises.

There are several solutions to the trellis termination problem. One option is to terminate one of the trellises and leave the other one "open". This strategy has only a minimal impact on performance. Another solution to the trellis termination problem is to force both encoders back to the all-zeros state by using the encoder circuit.

The two switches are in the up position until the end of the data frame, at which time they are thrown to the down position. Each encoder then independently generates the tail required to terminate itself. The tail for the first encoder is included at the end of the systematic output, but the tail for the second encoder must be transmitted separately.

\[ \text{Input} \rightarrow \text{RSC 1} \rightarrow X_2 \]
\[ \text{Interleaver} \rightarrow \text{RSC 2} \rightarrow x_1 \rightarrow x_3 \]

**Fig. 5.1: Classical Turbo encoder**
While this approach does indeed terminate both encoders, it suffers from the fact that the set of bits at the inputs of the two encoders are not the same. This reduces the efficiency of the decoding algorithm, which is derived under the assumption that the two encoders receive the same set of input data, only in permuted order.

5.2 Performance of Classical Turbo codes

Turbo Code, as shown in Fig. 5.1, consists of two identical constituent encoders, denoted as RSC Encoder-1 and RSC Encoder-2. Each RSC encoder is shown in Fig. 5.2. The interleaver is the critical part of the TC. It permutes the order of the data bits in irregular manner. The input data stream and two outputs redundancy are then serialized into a single Turbo codeword. It is a rate 1/3 encoder. If puncturing is implemented to the outputs of the redundant bits the rate may be increased to 1/2. The interleaver is random interleaver which scramble the input data to the 2\textsuperscript{nd} encoder.

![Fig. 5.2: Classical Turbo RSC component Encoder](image)
Fig. 5.3: BER performance of Turbo Codes for AWGN and Rayleigh Channel (Frame size (FS)-1000, Rate-1/3)

Fig. 5.4: BER performance of Turbo Codes for different frame size (Rate-1/2, AWGN channel, iteration-5, Random inter.)
Fig. 5.5: BER performance of Turbo Codes for different interleavers (FS-400, Rate-1/2, AWGN channel, iteration-5)

Fig 5.6: BER performance of the 4-state Turbo codes for different iteration (Frame size- 400, Rate- 1/2, AWGN Channel.)
Fig. 5.7: BER performance of the 4-state Turbo Codes for different rate (FS-400, AWGN channel, iteration-5)

Fig. 5.8: BER performance of Turbo Codes for different constraint length (FS-400, AWGN channel, iteration-5)
In Fig. 5.3, Turbo Codes for two channels, i.e., AWGN and Rayleigh channel are modeled to estimate the performance of Turbo Codes. In AWGN channel model the performance of Turbo Codes is much better than Rayleigh Channel model. At $BER = 10^{-4}$, $E_b / N_0$ for AWGN channel is 2 dB and $E_b / N_0$ for Raleigh channel model is 2.3 dB. So the performance of Turbo Codes degrades by 0.3 dB at that BER in Raleigh channel model. In Fig. 5.4 performance of Turbo Codes for different frame size is demonstrated and it is shown that for larger block length BER performance is improved. This is inevitable as Claude Shannon proved that Shannon limit could be achieved by a completely random code that is a randomly chosen mapping set of codewords. And the code can be random if $k$ (frame size) and $n$ (codeword) tend to infinity. So if the frame size increased the randomness of the codeword is also increased and it leads to achieve the low BER at a fixed $E_b / N_0$. In Fig. 5.5, BER performance is shown for different interleaver design. Performance of Turbo Codes with Random interleaver and semirandom interleaver has no major difference. Random interleaver shows a little bit low BER than semirandom interleaver. But if we use the block interleaver the performance of Turbo Codes is degraded than the other two interleavers. As the block interleaver has a regular pattern, it cannot ensure the randomness of the codes. So the curve for Block interleaver is in upper right position than that for random and semirandom interleaver. In Fig. 5.6 the effect of decoding iteration is shown. One main potential of Turbo Codes is its iterative decoding. The Soft Input Soft Output (SISO) decoding incorporates the soft bit decision. To make full use of soft-decision decoding requires a component decoder that generates and makes use of it. It is likely that decoding errors will result by any decoder. So if one decoder makes some errors other decoder will correct it or at least it reduce the soft value associated with the errors. Thus by consecutive reapplication of soft decision, decoder may correct more of the errors and so on. So it is likely that if the number of iterations is increased, more errors are corrected and BER will be improved. In Fig. 5.7 the puncturing effect on the performance of Turbo Codes are shown. If there is no puncturing in the encoder then the code rate is 1/3. That is, for one information bit, 3 output bits are transmitted. But in the Classical Turbo Codes, there is an option that two bits will be transmitted for one information bits making the code rate 1/2. So it is likely that more errors can be corrected by decoding the 1/3 rate received bit than that of 1/2-rate bits, as some information about the original bits are lost.
The effect of constraint length on the performance is shown in Fig. 5.8. It is found that the BER performance is improved with the increase of constraint length.

Fig. 5.9: Frame Error Rate for different channel model and number of iteration, (Frame Size-400)

Fig. 5.10: Frame Error Rate for different frame size (AWGN Channel)
Frame error rate (FER) of Classical Turbo codes is shown in Fig. 5.9 to Fig. 5.11. Frame error rate is another performance criteria and is defined as the number of erroneous frames over total frames transmitted. It is shown that the frame error rate varies with the variation of architectural component of Turbo Codes. In Fig. 5.9, it is found that the frame error rate performance for AWGN channel is better than that for Rayleigh Channel. It is also shown that for specific channel, if the number of iterations is increased, the FER performance is improved. In Fig. 5.10, the effect of variation of frame size on the performance of Turbo Codes is shown. It is clearly found that if the frame size is increased, the required $E_b/N_0$ is small. In Fig. 5.11, the effect of number of iteration on the FER performance is shown. It is found that if the number of decoding iteration is increased, the required bit energy to noise spectral ratio is small.

5.2.1 Decoding delay

The received signal sequence is passed to a Turbo decoder, which produced an estimate of the message. One iteration of decoding time is equal to one transmit time of the frame.
Thus the total latency \( t_d \) is equal to the transmit time of the frame times the number of iterations.

\[
t_d = \frac{L}{R_b} \times Q \tag{5.1}
\]

Here,

\begin{align*}
L &= \text{Frame length} \\
R_b &= \text{Bit rate} \\
Q &= \text{Number of decoding iteration}
\end{align*}

Let us assume that data rate is 128 kilobits per second.

Table 5.1: Latency of 1/3 Classical Turbo codes operating in AWGN channel at \( E_b / N_0 = 1.5 \text{ dB} \), Frame Size of 400.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Latency (ms) @ 128kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.1</td>
</tr>
<tr>
<td>3</td>
<td>9.3</td>
</tr>
<tr>
<td>6</td>
<td>18.6</td>
</tr>
<tr>
<td>8</td>
<td>24.8</td>
</tr>
<tr>
<td>10</td>
<td>31</td>
</tr>
</tbody>
</table>

Table 5.1 shows different decoding latency for the performance shown in Fig. 5.6. Delay with the change of iterations is shown in Fig. 5.12. From this figure it is found that if we increase the decoding iteration decoding delay is also increased. Again with the increase of decoding iterations performance is also improved, which is shown in Fig. 5.6. For specific application trade off can be done between iterations and latency.
5.3 UMTS Turbo Codes

5.3.1 Encoder

As a forward error correction-coding scheme Third Generation Partnership Project (3GPP) uses two types of coding, Convolutional coding and Turbo Coding scheme. In the Convolutional coding technique constraint length is 9 and coding rates 1/3 and 1/2. Turbo Codes uses the Parallel Concatenated Convolutional Code (PCCC). Each constituent encoder is 8-state i.e., its constraint length is 3. The code generator $G(D)$ is equal to $[1, g_1(D)/g_2(D)]$. Where

$$g_1(D) = 1 + D^2 + D^3$$
$$g_2(D) = 1 + D + D^3$$

The initial value of the shift registers the constituent encoder shall be all zero when it starts to encode the input bit. The encoder is shown in the Fig. 5.13. The outputs of the encoder are $x_k$, $y_{1,k}$ and $y_{2,k}$. Here $x_k$ is systematic output, $y_{1,k}$ is parity output from the encoder 1 and $y_{2,k}$ is the parity output of the encoder 2. Before entering to the encoder 2, the input bits are scrambled by the internal interleaver. The systematic output bit $x_k'$ is used at the time of trellis termination.
5.3.2 Trellis termination technique

UMTS Turbo Codes differs from the Classical Turbo Codes in the trellis termination technique. The Classical Turbo Codes terminate one of its encoder's trellis keeping the other encoder unterminated. Whereas in UMTS Turbo Codes, the first three tail bits shall be used to terminate the first constituent encoder (switch of the upper encoder in upper position) while the 2nd encoder keeping disabled. The last three bits shall be used to terminate the second constituent encoder (switch of the upper encoder in upper position) while the first encoder is disabled.

5.4 Performance of UTMS Turbo Codes

In Fig. 5.14, UMTS Turbo Codes are simulated by modeling the channel as Additive White Gaussian Noise (AWGN) and Rayleigh. In AWGN channel model as only the additive and white noise are considered UMTS Turbo Codes are supposed to correct more errors than the Rayleigh channel, which considered the multipath fading and motion of the receiver. This figure shows that at a fixed BER, AWGN channel require a few dB less than the Rayleigh channel. In Fig. 5.15, performance of Turbo Codes for different frame
Fig. 5.14: BER performance of UMTS Turbo Codes for different channel (FS-800, rate-1/2)

Fig. 5.15: BER performance of UMTS Turbo Codes for different frame size (It-3, AWGN, rate-1/2)
size is shown. If the frame size is increased from 530 to 5114, at $BER = 10^{-3}$, required $E_b/N_0$ is reduced from 2.5 dB to 1.5 dB. So increase of intrleaver size increases the coding gain. In Fig. 5.16, the effect of number of decoding iterations on the performance of UMTS Turbo Codes are shown. It is likely that, by increasing the number of decoding iteration of Turbo Codes the performance is improved. At $BER = 10^{-3}$, if the number of iterations is increased from 6 to 10, the required bit energy to noise ratio is reduce from 2.2 dB to 1.7dB.

Fig. 5.17 shows that, FER performance for AWGN channel is improved than that for Rayleigh channel. If the frame size is increased FER is improved which can be seen in Fig. 5.18. With the increase of number of decoding iterations, FER performance is improved which is shown in Fig. 5.19.
Fig. 5.17: Frame Error Rate for different channel model.

Fig. 5.18: Frame Error Rate of UMTS Turbo code for different frame size
5.4.1 Decoding delay of UMTS Turbo codes

Frame and decoding delay is proportional to each other, which can be found in equation (5.1). Decoding delay of UTMS Turbo Codes for different frame size is shown in Table 5.2 and Fig. 5.20. Data rate is assumed as 128 kbps. Again if frame size is increased BER performance is improved, which is shown in Fig. 5.15. So there should be a trade off between decoding delay and BER performance.

Table 5.2: Latency of 1/3 UMTS Turbo codes operating in AWGN channel at

\[ \frac{E_b}{N_0} = 2 \text{ dB}, \text{ Decoding iterations} = 5. \]

<table>
<thead>
<tr>
<th>Frame size</th>
<th>Latency (ms) @ 128kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>530</td>
<td>20.7</td>
</tr>
<tr>
<td>1060</td>
<td>41.34</td>
</tr>
<tr>
<td>3460</td>
<td>134.94</td>
</tr>
<tr>
<td>5114</td>
<td>199.8</td>
</tr>
</tbody>
</table>
Fig. 5.20: 1/3 UMTS Turbo codes operating in AWGN channel at $E_b/N_0 = 2$ dB, Decoding iterations = 5.

5.5 cdma2000 Turbo Codes

As a forward error correcting coding technique, 3GPP2 recommends Convolutional Codes and Turbo Codes of variable rate. The Convolutional Code has the constraint length of 9 and the variable code rate is of 1/4, 1/3 and 1/2. The encoder structures can be found in [47] and analyzed in [48, 49]. Turbo Codes encodes the data, frame quality indicator and two reverse bits. At the end of the encoding, tail sequence are padded with the encoded bits.

5.5.1. cdma2000 Turbo Encoder

Turbo encoder is constructed to produce the output at a variable rate of 1/2, 1/3, 1/4 and 1/5. The constituent Convolutional code has the code generator of
Initially, the states of the constituent encoder registers are set to zero. Puncturing patterns on the encoder output can be found in [47] and they are applied to get variable code rate. The encoder structure is shown in the Fig. 5.21.

\[
G(D) = \begin{bmatrix}
g_1(D) & g_2(D) \\
g_0(D) & g_0(D)
\end{bmatrix},
\]

where \( g_1(D) = 1 + D + D^3 \), \( g_2(D) = 1 + D + D^2 + D^3 \), \( g_0(D) = 1 + D^2 + D^3 \).

Trellis termination technique has the effect on the performance of Turbo codes [19]. The encoder generates \( \frac{6}{\text{code rate}} \) tail output symbols after the encoding of the input data. The first \( \frac{3}{\text{code rate}} \) tail output symbols are generated by clocking Constituent

Fig. 5.21: cdma2000 encoder

5.5.2. Trellis Termination Technique

Trellis termination technique has the effect on the performance of Turbo codes [19].
Encoder 1 three times while the Constituent Encoder 2 is not clocked. The last $3/(\text{code rate})$ are generated by clocking Constituent Encoder 2 three times while the Constituent Encoder 1 is not clocked. Then the output symbols are punctured and repeated according to the pattern given in [47].

5.6 Performance of cdma2000 Turbo codes

In Fig. 5.22, cdma2000 Turbo Codes performance are shown for two-channel model, AWGN and Raleigh channel model. At each channel model performance for different iterations are shown. It is found that the code performs better in AWGN channel model than Rayleigh channel model. It is found that for 8 decoding iterations and BER of $10^{-3}$, required $E_b/N_0$ is 0.8 dB smaller for AWGN model than Rayleigh model. In Fig. 5.23, code performance for different frame sizes is shown and if we increase the frame size from 1530 to 12282, at BER $10^{-4}$, 0.5 dB can be reduced. In Fig. 5.24, code performance for different coding iteration is shown. It is found that if we increase the decoding iteration from 4 to 8, at BER $10^{-3}$, required $E_b/N_0$ can be reduced by 0.4 dB. In Fig. 5.25, variable code rate performance is shown. cdma2000 TC has the option to produce code rate of $1/2$, $1/3$, $1/4$ and $1/5$ by applying puncturing pattern. It is found from the figure that if we decrease code rate from $1/2$ to $1/5$, at BER $10^{-3}$, 0.9 dB improvements can be achieved. But with the decrease of code rate throughput is also increased. So there should be a trade-off between the $E_b/N_0$ and code rate. The code rate decision is also application dependent.
Fig. 5.22: BER performance of cdma2000 Turbo Codes for two Channel (FS-762, rate-1/2)

Fig. 5.23: BER performance of cdma2000 Turbo Codes for different frame size (it-5, rate-1/2, AWGN)
Fig. 5.24: BER performance of cdma2000 Turbo Codes for different coding iteration (FS-762, AWGN, rate-1/2)

Fig. 5.25: BER performance of cdma2000 Turbo Codes for different coding rate (FS-762, AWGN, it-5)
5.6.1 Decoding delay of cdma2000 Turbo codes

Delay for different frame size is shown in Table 5.3 and its alternate representation is shown in Fig. 5.26. Data rate is assumed as 128 kbps. From this figure and table it is found that the relation between delay and frame size is linear. With the increase of frame size BER performance of cdma2000 Turbo codes is improved (Fig. 5.23). So there should be a trade off between frame size and decoding delay.

Table 5.3: Latency of 1/3 cdma2000 Turbo codes operating in AWGN channel at $E_b/N_0 = 2dB$, Decoding iteration = 5.

<table>
<thead>
<tr>
<th>Frame size</th>
<th>Latency (ms) @ 128kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>378</td>
<td>14.7</td>
</tr>
<tr>
<td>762</td>
<td>29.7</td>
</tr>
<tr>
<td>6138</td>
<td>239</td>
</tr>
<tr>
<td>12282</td>
<td>478</td>
</tr>
</tbody>
</table>

Fig. 5.26: 1/3 cdma2000 Turbo codes operating in AWGN channel at $E_b/N_0 = 2dB$, Decoding iteration = 5.
Chapter 6

Distance Spectrum of Turbo Codes

6.1 Introduction

After the invention of Turbo Codes and near Shannon-capacity performance tremendous research efforts were taken to fully understand the new coding technique. In this thesis we addressed the performance of Turbo codes by examining the codes' distance spectrum. It is well known that error floor occurs at moderate signal-to-noise ratio in the Bit Error Rate (BER) vs. bit energy to noise spectral ratio ($E_b/N_0$). The cause of error floor is due to the relatively low free distance. Several techniques were proposed by researchers to lower the error floor. These techniques are assessed in this thesis. Again to determine the free distance several algorithms were developed by different researchers. We also assess these algorithms and using one algorithm we have evaluated the distance spectrum of Third Generation mobile standard Turbo codes specially UMTS (Universal Telecommunication System), cdma2000 and CCSDS (Consultative Committee for Space Data Systems) standards Turbo Codes.

This thesis will define and evaluate the upper bound to the average performance of the decoder for a Parallel Concatenated Convolutional codes (PCCC). The upper bound of Turbo codes can be expressed by the following equation [28].

$$P_b \leq \frac{2^{(v+1)N}}{d_{free}} \frac{N_{d} \bar{w}_{d}}{N} Q\left(\sqrt{d \frac{2RE_b}{N_0}}\right).$$

(6.1)

where $P_b$ is error probability, $v$ is the memory of Convolutional code, $d$ is the Hamming distance, $N$ is the interleaver size, $N_d$ is the multiplicity of weight-$d$ codeword, $\bar{w}_d$ is the average weight of the information sequences causing weight-$d$ codeword, $R$ is the code rate, $E_b$ is the signal energy and $N_0$ is the noise spectral ratio.

We can define the free distance term, which is the minimum Hamming distance between the codeword and all zero codeword. It is found that the free distance dominates on the BER performance of Turbo codes at medium to high SNR. It is found that at
medium to high SNR error floor occurs in the BER performance. The error floor is the flattening part of the performance curve, for moderate to high SNR. Many researchers prove that the error floor is mainly due to the free distance of Turbo codes. In this region, its free distance, $d_{\text{free}}$ and its multiplicities, dominates the performance of any binary code. As mentioned in [28], some concatenated codes with interleavers may have very low free distances, even when large interleaver lengths $N$ are used. This causes their BER curves to flatten according to the "error floor" imposed by $d_{\text{free}}$, after the "water fall" part of the curve. This behavior is not expected for applications requiring very low BER, e.g., between $10^{-6}$ and $10^{-10}$. So to assess the performance of Turbo codes, we have to measure the free distance of Turbo codes of a definite structure. We will explain the algorithm, developed by the R. Garello et. al. in [29] to find the distance spectrum of Turbo codes. The algorithm is improved by E. Rosnes et. al. in [30]. In this thesis this algorithm is implemented to measure the distance spectrum of third generation standard Turbo code, i.e., UMTS, cdma2000 and CCSDS Turbo codes. Then we show the dependence of distance spectrum on code rate, interleaver size and interleaver type.

6.2 Performance Bound of Turbo codes

The bit error rate (BER) performance of a Convolutional code with maximum-likelihood (ML) decoding on an additive white Gaussian noise (AWGN) channel can be upper-bounded using a union bound technique.

Input Redundancy Weight Enumerating Function (IRWEF) of any block code can be expressed as [22]

$$A^C(W, Z) = \sum_{w,j} A_{w,j} W^w Z^j$$

where $A_{w,j}$ denotes the number of codewords generated by an input information word of Hamming weight $w$ whose parity check bits have Hamming weight $j$, so that the overall Hamming weight is $w+j$. The bit error probability for maximum soft decoding of the code over a channel with additive white Gaussian noise in the form

$$P_b(c) \leq \left. \frac{w}{k} \frac{\partial A^C(W, Z)}{\partial W} \right|_{W = Z = e^{-RC\mathcal{E}_b/N_0}}$$

(6.2)
Using some approximations (6.2) can be rewritten as

$$P_b(e) \leq \sum_{w=1}^{k} w^w A^C_w (Z) W = Z = e^{R_C E_b / N_0}$$ \hspace{1cm} (6.3)

After some calculation we get

$$P_b \leq \frac{2^{(v + N)}}{N} \sum_{i=1}^{d} \frac{W_e}{N_0} Q \left( \frac{d E_b}{N_0} \right),$$ \hspace{1cm} (6.4)

where, \(w_i\) is information weight and \(d_i\) is total Hamming weight of the \(i^{th}\) codeword.

Let us define the average information weight per codeword as

$$\bar{w}_d = \frac{W_d}{N_d}$$

Here \(W_d\) is the total information weight of all codewords of weight \(d\) and \(N_d\) is the multiplicity of codewords of weight \(d\). So (6.4) becomes

$$P_b \leq \frac{2^{(v + N)}}{N} \sum_{d=d_{\text{free}}}^{N} \frac{N_d \bar{w}_d}{N_0} Q \left( \frac{d E_b}{N_0} \right)$$ \hspace{1cm} (6.5)

Equation (6.5) is the upper bound for the Convolutional code. The performance of a Turbo code with maximum-likelihood decoding can also be bounded using the union bound of (6.5). For moderate and high signal-to-noise ratios, it is well known that the free-distance term in the union bound on the bit error rate performance dominates the bound. Thus for a Turbo codes the asymptotic performance approaches

$$P_b = \frac{\bar{w}_{\text{free}} N_{\text{free}}}{N} Q \left( \frac{d_{\text{free}} E_b}{N_0} \right),$$ \hspace{1cm} (6.6)

where \(N_{\text{free}}\) is the multiplicity of free-distance codewords and \(\bar{w}_{\text{free}}\) is the average weight of the information sequences causing free-distance codewords. By using algorithm for finding the free distance and plugging the values in (6.6), the free distance asymptotes
graph can be generated. For the Turbo codes in [28], i.e., (37, 21, 65536) code was found to have $N_{\text{free}} = 3$ paths, $d_{\text{free}} = 6$. For this particular Turbo code, the free distance asymptote is given by

$$P_{\text{free}} = \frac{3 \times 2}{65536} Q\left(\frac{E_b}{N_0}\right)$$

(6.7)

By plotting $P_{\text{free}}$ vs. $\frac{E_b}{N_0}$ we get the asymptotic curve in Fig. 3.

For this code the effective multiplicity is

$$\frac{N_{\text{free}}}{N} = \frac{3}{65536}$$

The free-distance asymptotes and the simulation result are shown in Fig. 6.1. From this figure, it can be clearly seen that the simulation result does in fact approach the free-distance asymptote for moderate and high SNR. Since the slope of the asymptote is essentially determined by the free distance of the code, it can be concluded that the error-floor observed in the Turbo codes performance is due to the fact that they have a relatively small free distance and consequently a relatively flat free-distance asymptote.

6.3 Lowering the Error Floor of Turbo Codes

It is found that increasing the length of the interleaver while preserving the free distance and the multiplicity will lower the asymptote without changing its slope by reducing the effective multiplicity. In this case the performance curve of Turbo codes does not flatten out until higher SNR and lower BER are reached. If the size of the interleaver is fixed, then increasing free distance can modify the error floor distance of the code while preserving the multiplicity. This has the effect of changing the slope of the free-distance asymptote. That is, increasing the free distance increases the slope of the asymptote and decreasing the free distance, decreases the slope of the asymptote. It is also shown that for a fixed interleaver size, choosing the feedback polynomial to be a primitive polynomial result in an increased free distance and thus a steeper asymptote.

Another way to improve the error floor is that, we have to identify the information bit positions affected by low-distance error event, which are few in number due to the
sparseness of the spectrum. A modified encoder inserts dummy bits in these positions, resulting in a lower and steeper error floor, the bit-error-rate performance curve. For sufficiently large interleaver size, the only cost is a very slight reduction in the code rate.

6.4 Distance Spectrum Measurement of Turbo Codes

Different efforts were taken to measure the distance spectrum of Turbo codes. In this thesis we use the recent algorithm as described in [29] to evaluate the distance spectrum of different Turbo codes as standard of UMTS, cdma2000 and CCSDS. First we describe the algorithm of [29] in different notations.

Definition

A constrained set $F$ is defined as $\left\{ (p_i, u_{p_i}) : u_{p_i} \in \{0, 1\} \ \forall p_i \in F_p \right\}$

$F_p$ is defined as $F_p \subseteq \{0, 1, \ldots, N-1\}$
be the set of length-$N$ vectors defined as
\[ u = (u_0, \ldots, u_{N-1}) : u_j = u \text{ if } (j, u) \in F, u_j \in \{0, 1\} \]
if $j \notin F$.

length $l = l(F)$ be the number of constraints.

Turbo interleaver acting on $F$, we obtain a new constraint set $\pi F = \{(\pi(p_i), u_{p_i})\}$.

Here $\pi$ is the interleaving algorithm.

$\pi(p_i)$ is the scrambled version of bit in $p_i$ position.

$u_{p_i}$ is the original bit of $p_i$ position.

$C^{(F)}$ be the subset of the Turbo code and we get it by encoding the input vectors in $U^{(F)}$.

$w(F)$ be the minimum hamming weight of $C^{(F)}$.

Algorithm

Add an empty constraint set $F$ to a previously empty list $L$ of constraint sets.

(*) If $L$ is empty,

Terminate the process.

Otherwise,

Choose and take out a constraint set $F$ from $L$.

(**) If $w(F) \leq r$, then

If the length $l$ of $F$ is equal to $N$ then:

The single vector in $U^F$ produces a low-weight word.
Make a record of it.

Otherwise,

Construct two new constraint sets:
\[ F' = F \cup \{I, 0\} \], and
\[ F^* = F \cup \{I, 1\} \]

Add \( F' \) and \( F^* \) to \( L \).

Proceed from (*)

Finally the Turbo code free distance, code multiplicity and information multiplicity can be evaluated. In [30] some improvement of the stated algorithms are proposed. Here mainly two improvements were suggested. First is to reduce the total number of constraint sets \( (M_{\text{const}}) \) investigated. \( M_{\text{const}} \) depends on parameters of the problem and on the lower bound used in place of \( w(F) \). The minimum value of \( M_{\text{const}} \) can be obtained when the lower bound is made equal to \( w(F) \). Second is to reduce the average cost of processing each constraint set \( (V_{\text{cons}}) \), i.e., the complexity of carrying out the main loop of the algorithm (**).

6.5 Distance Spectrum of Turbo Codes of Different Standards

The preceding algorithm can be implemented to compute all the terms of distance spectrum together with their multiplicities. Applying the algorithm we compute the distance spectrum of UMTS, cdma2000 and CCSDS Turbo codes.

6.5.1 'Best' Turbo codes

'Best' Turbo code, described in [28] the rate-1/2 constituent Convolutional encoders were presented with optimized distance distribution which minimize the "average" bit error probability of a PCCC employing a uniform interleaver. The 16 state codes, the optimal constituent systematic encoder has the feedback polynomial of 
\[ g_2(D) = 1 + D + D^2 + D^3 + D^4 \]
and feedforward polynomial of \( g_1(D) = 1 + D^3 + D^4 \). Then for different size of interleaver we can measure the distance spectrum terms as follows:

\[ N = \text{Interleaver size}, \quad d_{\text{free}} = \text{average free distance over all the random interleaver of same size}, \quad (d_{\text{free}}, w_{\text{free}}^{(\text{best})}) \text{ is the maximum free distance and minimum multiplicity} \]
By using the algorithm we can measure the distance spectrum and it is reported in Table 6.1.

From Table 6.1 it is found that with the increase of interleaver size the free distance spectrum is increased, i.e., performance is improved.

<table>
<thead>
<tr>
<th>N</th>
<th>$d_{free}$</th>
<th>$d_{free}$</th>
<th>$w_{free}^{(best)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>8.0</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>8.9</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>8.93</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>9.26</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>9.33</td>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>9.4</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>9.55</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>9.7</td>
<td>13</td>
<td>29</td>
</tr>
</tbody>
</table>

### 6.5.2 UMTS Turbo Codes

UMTS is third generation partnership project (3GPP) standard. Its encoder has two-recursive systematic Convolutional encoder. Each Convolutional encoder is 8-state and 1/2 rate and the rate of Turbo codes is 1/3. A block interleaver with length $N$ is used. Its range is in between 41 to 5114. The algorithm of the interleaver is described in [45].

The distance spectrum of UMTS Turbo codes is reported in Table 6.2. Here $d_{free}$ is the free distance, $N_{free}$ is its multiplicities and $w_{free}$ is its information multiplicities.
Table 6.2: Distance Spectrum of UMTS Turbo code

<table>
<thead>
<tr>
<th>Distance spectrum</th>
<th>$d_{\text{free}}$</th>
<th>$N_{\text{free}}$</th>
<th>$w_{\text{free}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 40$</td>
<td>13</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>$N = 200$</td>
<td>20</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>$N = 320$</td>
<td>24</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

From Table 6.2, it is found that the free distance $d_{\text{free}}$ is increased with the increase of the interleaver size. At the same time the codeword multiplicities, $N_{\text{free}}$, and the information multiplicities, $w_{\text{free}}$, is decreased with the increase of the interleaver length $N$. So the performance of the Turbo codes improves with the increase of interleaver size, which was predicted.

6.5.3 cdma2000 Turbo Codes

cdma2000 is third generation partnership project-2 (3GPP2) standard. Its encoder has two recursive systematic Convolutional encoders. Each Convolutional encoder is 8-state encoder. The rate of Turbo codes is 1/2, 1/3, 1/4, 1/5.

Table 6.3: Distance Spectrum of cdma2000 Turbo code

<table>
<thead>
<tr>
<th>cdma2000 Turbo codes</th>
<th>$d_{\text{free}}$</th>
<th>$N_{\text{free}}$</th>
<th>$w_{\text{free}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate = $\frac{1}{2}$</td>
<td>10</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>rate = $\frac{1}{3}$</td>
<td>21</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>rate = $\frac{1}{4}$</td>
<td>30</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

The puncturing pattern and the interleaving technique can be found in [47]. The distance spectrum is reported in Table 6.3. This table demonstrates that if the code rate is decreased the free distance, $d_{\text{free}}$, is increased.
6.5.4 CCSDS Turbo Code

In CCSDS (Consultative Committee for Space Data Systems) standards, the old channel coding standard has been updated to include Turbo codes [50]. Two equal binary systematic recursive Convolutional encoders with rate of 1/4 constitute the encoder. The block interleaver length is 1784, 3568, 7163 or 8920. The algorithm of the interleaver is described in [50]. The code rate is 1/2, 1/3, 1/4 and 1/6. By applying the algorithm to measure the distance spectrum we get the spectrum, which is reported, in Table 6.4 and Fig. 6.2.

**Table 6.4: Distance Spectrum of CCSDS Turbo code**

<table>
<thead>
<tr>
<th>Interleaver Size</th>
<th>N = 1784</th>
<th>N = 3568</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate</td>
<td>1/2</td>
<td>1/3</td>
</tr>
<tr>
<td>(d_{\text{free}})</td>
<td>17</td>
<td>32</td>
</tr>
<tr>
<td>(N_{\text{free}})</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>(w_{\text{free}})</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

From these table and figure, it is found that for a fixed interleaver size, if the code rate is decreased the free distance is increased, that is the performance is improved. It is also shown in the figure that, if the interleaver size is increased, for same code rate, the free distance is also increased.
6.6 Effect of Encoder structure on Distance Spectrum

Turbo codes uses two or more recursive systematic Convolutional codes. It is well known to the researchers that the structure of each recursive systematic Convolutional codes must have the influence on Turbo codes. In this thesis we have investigated the influence of the encoder of Turbo codes on its performance. In these investigation shift registers of 2, 3 and 4 are considered. Using the UMTS interleaver and 2 constituent RSC codes we have evaluated the distance spectrum and reported the results from Table 6.5 to Table 6.7.
Table 6.5 Distance Spectrum of different encoders (2 Shift registers)

<table>
<thead>
<tr>
<th>Polynomial</th>
<th>(d_{\text{free}})</th>
<th>(N_{\text{free}})</th>
<th>(w_{\text{free}})</th>
</tr>
</thead>
</table>
| \**Feedback** = \(1 + D + D^2\)  
Feedforward = \(1 + D^2\) | 16  | 1  | 2  |
| \**Feedback** = \(1 + D^2\)  
Feedforward = \(1 + D + D^2\) | 9  | 1  | 2  |

Table 6.6: Distance Spectrum of different encoders (3 Shift registers)

<table>
<thead>
<tr>
<th>Polynomial</th>
<th>(d_{\text{free}})</th>
<th>(N_{\text{free}})</th>
<th>(w_{\text{free}})</th>
</tr>
</thead>
</table>
| \**Feedback** = \(1 + D^2 + D^3\)  
Feedforward = \(1 + D + D^2 + D^3\) | 25  | 1  | 3  |
| \**Feedback** = \(1 + D + D^3\)  
Feedforward = \(1 + D^2 + D^3\) | 23  | 1  | 3  |
| \**Feedback** = \(1 + D + D^3\)  
Feedforward = \(1 + D^2 + D^3\) | 10  | 1  | 2  |
Table 6.7 Distance Spectrum of different encoders (4 Shift registers)

<table>
<thead>
<tr>
<th>Polynomial</th>
<th>Distance Spectrum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$d_{\text{free}}$</td>
</tr>
<tr>
<td>Feedback = $1 + D + D^4$</td>
<td>30</td>
</tr>
<tr>
<td>Feedforward = $1 + D + D^2 + D^3 + D^4$</td>
<td></td>
</tr>
<tr>
<td>Feedback = $1 + D^3 + D^4$</td>
<td>28</td>
</tr>
<tr>
<td>Feedforward = $1 + D + D^2 + D^3 + D^4$</td>
<td></td>
</tr>
<tr>
<td>Feedback = $1 + D + D^2 + D^4$</td>
<td>23</td>
</tr>
<tr>
<td>Feedforward = $1 + D + D^2 + D^3 + D^4$</td>
<td></td>
</tr>
<tr>
<td>Feedback = $1 + D^2 + D^4$</td>
<td>17</td>
</tr>
<tr>
<td>Feedforward = $1 + D + D^2 + D^3 + D^4$</td>
<td></td>
</tr>
<tr>
<td>Feedback = $1 + D^2 + D^3 + D^4$</td>
<td>24</td>
</tr>
<tr>
<td>Feedforward = $1 + D + D^2 + D^3 + D^4$</td>
<td></td>
</tr>
<tr>
<td>Feedback = $1 + D^4$</td>
<td>12</td>
</tr>
<tr>
<td>Feedforward = $1 + D + D^2 + D^3 + D^4$</td>
<td></td>
</tr>
<tr>
<td>Feedback = $1 + D + D^3 + D^4$</td>
<td>24</td>
</tr>
<tr>
<td>Feedforward = $1 + D + D^2 + D^3 + D^4$</td>
<td></td>
</tr>
</tbody>
</table>
From the Tables 6.5 to 6.7 it is found that if the feedback polynomials are primitive polynomials (**) then the distance spectrum is improved. But if they are not primitive the distance spectrum is not much improved. This result complies with the theoretical result predicted by [24]. We also found that, an encoder structure shows better performance than the UMTS encoder. The encoder structure that shows the best performance is as below:

Feedback polynomial = $1 + D + D^4$

Feedforward polynomial = $1 + D + D^2 + D^3 + D^4$
Chapter 7

Interleaver Design of Turbo Codes

7.1 Introduction

Interleaving is a process of rearranging the ordering of a symbol sequence. It has been widely used with error correction coding for channels, which have bursty error characteristics. However, in Turbo codes, interleaver is used to permute the input bits such that the constituent encoders are operating on the same set of input bits, but in different order. One of the basic roles of interleaver is to increase the free distance of the output codeword. Other role of it is to scramble the information sequence to the second constituent encoders and decorrelate the inputs to the two decoders [38-40].

A number of interleavers have been used with Turbo codes. A general periodic interleaver, which is called Convolutional interleaver, was introduced in [36]. In Convolutional interleaver data is multiplexed into and out of a fixed number of shift registers. A particular class of Convolutional interleaver is block interleaver. In a Block interleaver, input data are written along the rows of the matrix and then read out along the columns [36]. The recent applications of block interleaver are in Universal Mobile Telecommunication Systems (UMTS) and cdma2000 standard Turbo codes [45, 47]. Pseudo-random interleaver is a variation of the block interleaver in which the data is written sequentially and read out in a pseudorandom order. The $S$-random interleaver is an improved version of the pseudo-random interleaver. It can spread low weight input patterns to generate higher weight codewords; hence can achieve better performance compared to pseudorandom interleavers. Random interleaver randomly scrambles the input data. A class of algebraic interleavers that permute a sequence of bits with nearly the same statistical distribution as a randomly chosen interleaver is presented in [37].

An interleaver can be designed to break low weight input patterns to generate high weight parity-check sequences. In [39], an interleaving algorithm is developed so that at least one decoder output has a relatively high weight whenever the other encoder produces low weight output. But it is only suitable for short interleaver sizes since its complexity increases rapidly as the interleaver size grows.
There are two major criteria in the design of an interleaver: 1) the distance spectrum properties (weight distribution) of the code, and 2) the correlation between the soft output of each decoder corresponding to its parity bits and the information input data sequence. Criterion 2 is sometimes referred to as the iterative decoding suitability criterion. This is a measure of the effectiveness of the iterative decoding algorithm and the fact that if these two data sequences are less correlated, and then the performance of the iterative decoding algorithm improves.

7.2 Interleaver Design

7.2.1 Block Interleaver

The block interleaver is the most commonly used interleaver in communication systems. It writes in column wise from top to bottom and left to right and reads out row wise from left to right and top to bottom. Fig. 7.1 shows a block interleaver.

![Block Interleaver Diagram]

Fig. 7.1: Block interleaver [67]

From Fig. 7.1, the interleaver writes in row wise and reads out column wise.

7.2.2 Random (Pseudo-Random) Interleaver

The random interleaver uses a fixed random permutation and maps the input sequence according to the permutation order. The length of the input sequence is assumed to be \( L \). Fig. 7.2 shows a random interleaver with \( L = 8 \).
Fig. 7.2: A random (pseudo-random) interleaver with $L = 8$.

From Figure 7.2, the interleaver writes in $[0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0 \ 1]$ and reads out $[0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1]$.

7.2.3 Circular-Shifting Interleaver

The permutation $p$ of the circular-shifting interleaver is defined by

$$p(i) = (ai + s) \mod L$$

satisfying $a < L$, $a$ is relatively prime to $L$, and $s < L$ where $i$ is the index, $a$ is the step size, and $s$ is the offset. Figure 7.3 shows a circular-shifting interleaver with $L = 8$, $a = 3$, and $s = 0$. 
From Fig. 7.3, it can be seen that the adjacent bit separation is either 3 or 5. This type of interleaver has been shown to do a very good job of permuting weight-2 input sequences with low codeword weights into weight-2 input sequences with high codeword weights. However, because of the regularity (3 or 5 adjacent bit separation for Fig. 7.3 inherent in this type of interleaver, it may be difficult to permute higher weight input sequences with low codeword weights into other input sequences with high codeword weights.

### 7.2.4 Semi random Interleaver

The semirandom interleaver is a compromise between a random interleaver and a "designed" interleaver such as the block and circular-shifting interleavers. The permutation algorithm for the semirandom interleaver is described below.

**Step 1.** Select a random index $i \in [0, L-1]$

**Step 2.** Select a positive integer $S < \sqrt{L}$

**Step 3.** Compare $i$ to previous $S$ integers. For each of the $S$ integers, compare $i$ to see if it lies within $\pm S$. If $i$ does lie within the range, then go back to Step 1.

Otherwise, keep $i$.

**Step 4.** Go back to Step 1 until all $L$ positions have been filled.
The semirandom interleaver tries to introduce some randomness to overcome the permutation regularity; however, the algorithm does not guarantee to finish successfully.

7.2.5 Optimal (Near-Optimal) Interleaver

The optimal interleaver can be described as the interleaver that produces the fewest output-coded sequences with low weights. This interleaver design is both tedious and exhaustive. The following algorithm describes the interleaver design concept.

1. Generate a random interleaver.
2. Generate all possible input information sequences.
3. For all possible input information sequences, encode each of the input information sequences and determine the resulting codeword weight. This gives the weight distribution of the code.
4. Determine the minimum codeword weight and the number of codewords with that weight.

This algorithm is repeated for a "reasonable number of times" and keeps the interleaver that produces the largest minimum codeword weight with the lowest number of codewords of that weight.

7.2.6 UMTS TC Interleaver

Turbo Codes interleave scramble the input data by following three steps as bits-input to a rectangular matrix with padding, intra-row and inter-row permutations of the rectangular matrix with pruning. The bit input \((K)\) to the TC internal interleaver is recommended to be between 40 and 5114.

7.2.6.1 Bits input to a rectangular matrix:

1. Number of rows \((R)\) of the rectangular matrix:

\[
R = \begin{cases} 
5, & \text{if} \ (40 \leq K \leq 5114) \\
10, & \text{if} \ (160 \leq K \leq 200) \text{ or } (481 \leq K \leq 530) \\
20, & \text{if} \ (K = \text{other value})
\end{cases}
\]

2. Determine the of prime number and the number of columns \((C)\) of the rectangular matrix:
if \((481 \leq K \leq 530)\)
\[ p = 53 \text{ and } C = p \]
else
\[ K \leq R \times (p+1) \]
\[
C = \begin{cases} 
  p - 1, & \text{if } K \leq R \times (p-1) \\
  p, & \text{if } R \times (p-1) < K \leq R \times p \\
  p+1, & \text{if } R \times p < K 
\end{cases}
\]
\]
\]
end if

Now input bits are written into the \(R \times C\) rectangular matrix row by row. If \(R \times C > K\), dummy bits are padded.

7.2.6.2 Intra-row and inter-row operation

1. Select a primitive root \(v\) from table of [45]

2. Construct base sequence \(s(j) = (v \times s(j-1)) \mod p, j = 1, 2, 3, \cdots (p-2)\)

3. Construct the sequence \(\langle q_i \rangle_{i \in \{0, 1, 2, \cdots R-1\}}\), given that \(q_0 = 1\), \(q_i\) is be a least integer such that \(\gcd(q_0, p-1) = 1, q_i > 6, q_i > q_{i-1}\)

4. \(r_{T(i)} = q_i\), where the \(T(i)\) is the inter-row permutation pattern defined in a table of.

5. Algorithm of intra-row operation is as follows:
   if \((C = p)\) then
   \[
   U_t(j) = s((j \times r_t) \mod (p-1)), j=0, 1, \cdots(p-2), \text{ and } U_t(p-1) = 0\text{ where } U_t \text{ is the original bit position of } j\text{-th permuted bit of } i\text{-th row}.
   \]
   end if
   if \((C = p+1)\)
\[ U_i(j) = s((j \times r_j) \mod (p - 1)), \quad j=0, 1, ..., (p-2), \quad \text{and} \quad U_i(p-1) = 0, \quad \text{and} \quad U_i(p) = p \]

if \((K = R \times C)\) then

\[ \text{Exchange} \ U_{(R-1)}(p) \ \text{with} \ U_{(R-1)}(0) \]

end if

end if

if \((C = p-1)\) then

\[ U_i(j) = s((j \times r_j) \mod (p - 1)) - 1, \quad j=0, 1, ..., (p-2), \]

end if

6. Perform the inter-row permutation for the rectangular matrix based on the pattern

\[ \{T(i)\}_{i \in \{0,1,2,...,R-1\}} \]

7.2.6.3 Bits-output from matrix

The output of the TC internal interleaver is the bit sequence read out column-by-column and pruning the dummy bits.

7.2.7 cdma2000 TC Interleaver

The approach of the interleaver are equivalent to write the inputs bits sequentially into an array at a sequence of addresses, and then the entire sequences read out from a sequence of addresses [38]-[39]. cdma2000 interleaving algorithm is as follows:

1. Turbo interleaver parameter, \(n\) is determined, where \(K \leq 2^{n+5}\), \(K\) is the input bits to the Turbo encoder.

2. \((n+5)\)-bit counter is initialized to 0.

3. Extract the \(n\) MSBs from the counter and add one to form a new value. Discard all except \(n\) LSBs.

4. \(n\) bit output is found in a look-up table found in [47] using the five LSBs of the counter.
5. Multiply the values in steps 3 and 4. Discard all except the $n$ LSBs.

6. Five LSBs of the counter are reversed.

7. A tentative output address that has its MSBs equal to the value obtained in step 6 and its LSBs equal to the value obtained in step 5.

8. Accept the tentative output address as an output address if it is less than $K$, otherwise discard it.

9. Increment the counter and repeat steps 3 through 8 until all $K$ interleaver output addresses are obtained.

### 7.3 Effect of Interleaver on Distance Spectrum

For Turbo codes, the asymptotic performance approaches [28]

$$P_d(E_b/N_0) = \frac{\hat{w}_\text{free} N_{\text{free}}}{N} Q\left(\sqrt{\frac{2RE_b}{N_0}}\right)$$

(7.1)

In equation (7.1), $N_{\text{free}}$ is the multiplicity of free-distance codewords and $\hat{w}_\text{free}$ is the average weight of the information sequences causing free-distance codewords. By using algorithm for finding the free distance and plugging the values in (7.1), the free distance asymptotes graph can be generated.

**TABLE 7.1 Distance Spectrums of Different Interleavers**

<table>
<thead>
<tr>
<th>Inter. Type</th>
<th>Random</th>
<th>S-random</th>
<th>UMTS</th>
<th>cdma2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frame Size</td>
<td>378</td>
<td>762</td>
<td>378</td>
<td>762</td>
</tr>
<tr>
<td>$d_{\text{free}}$</td>
<td>10</td>
<td>11</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td>$N_{\text{free}}$</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>$\hat{w}_{\text{free}}$</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>
The distance spectrum of Classical Turbo codes for different interleaver and interleaver size are shown and compared in Table 7.1. Table 7.1 demonstrates that the free distance \( d_{\text{free}} \) of S-random interleaver is the largest. Between the two 3rd generation wireless standard interleavers, cdma2000 has improved distance spectrum.

### 7.3.1 Specific Patterns of Low Weight Interleaver Inputs

#### TABLE 7.2 Encoder Output Using Impulse Responses of Low Weight Input

<table>
<thead>
<tr>
<th>Input weight</th>
<th>Input pattern</th>
<th>Encoder output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight-2</td>
<td>100100000000..</td>
<td>111011011011...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11101101...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11100000000...</td>
</tr>
<tr>
<td>Weight-3</td>
<td>1000100010000..</td>
<td>111011011010...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11101101...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111000111000...</td>
</tr>
<tr>
<td>Weight-4</td>
<td>1001010010000..</td>
<td>111011011011...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11101101...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1110100111000...</td>
</tr>
<tr>
<td>Weight-5</td>
<td>11100010001000..</td>
<td>111011011010...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11101011011...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1110101101...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1110100111000...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101000111000...</td>
</tr>
</tbody>
</table>
It is well known that information weight of 2, 3, 4 and 5 of specific pattern produce the low weight codeword. Most of the literatures took into account the weight-2 and 4 information sequences. In this thesis, we consider the weight-3, 5 as well. From the impulse responses of weight-2, 3, 4, 5 information sequences, it is found that for the specific pattern of 1's in input sequence, the codeword becomes all zero pattern, i.e., self terminated path. These paths are called error path. We report it in Table 7.2.

Its relative contribution to the total BER can be represented as

\[ \tilde{P}_d = \frac{P_d(E_b/N_0)}{\sum_d P_d(E_b/N_0)} \]  

(7.2)

The relative contribution of the spectral line in a \((E_b/N_0)\) range, \([x, y]\) can be obtained from the following equation

\[ C_{xy} = \frac{Y}{X} \frac{P_d(E_b/N_0) \Delta(E_b/N_0)}{P_d(E_b/N_0) \Delta(E_b/N_0)} \]  

(7.3)

\(C_{xy}\) can be further normalized as

\[ \tilde{C}_{xy} = \frac{C_{xy}}{\sum_d C_{xy}} \]  

(7.4)

Using (7.4), we report the relative contribution for specific input weight and its corresponding codeword weight in Table 7.3. From this table it is evident that, low weight codeword is mainly due to the weight-2, 3, 4, 5, 6, and 8 information sequence. Weight-3, 5 have less effect on the BER performance of Turbo codes than the 2, 4, 6, and 8. So most researchers overlook weight-3 and 5 information sequences expecting that the
information sequence will be broken after the interleaving. However, this thesis addresses and analyzes these information sequences explicitly.

7.4 Effect of weight-2, 3, 4, 5 Input Pattern

Let us denote by $w$ the input weight, $w(z_1)$ the parity-check weight of the 1st encoder and $w(z_2)$ the parity-check weight of the 2nd encoder. The overall codeword weight is given by

$$d = w + w(z_2) + w(z_2)$$

(7.5)

Investigating the pattern shown in Table 7.2, a weight-2 input sequence that generates a finite weight codeword can be represented by a polynomial

<table>
<thead>
<tr>
<th>Input weight</th>
<th>Codeword weight</th>
<th>Relative contribution $\left( \bar{c}^w \right)$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>5.357e+01</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>1.725e+01</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>1.364+01</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>4.945e+00</td>
</tr>
</tbody>
</table>
In (7.6), \( m = 1, 2, 3 \ldots \), \( \delta \) is the minimum distance between two "1"s in the weight-2 input pattern that generate the finite weight codeword and \( \varphi_1 \) is the time delay.

The input to the 2nd encoder can be represented as

\[
Q_2(D) = (1 + D^{\delta m_2})D^{\varphi_2}
\]  

(7.7)

From (7.7), the overall codeword weight can be calculated as

\[
d = 6 + (m_1 + m_2 \times \gamma_{\min} - 2),
\]

where \( \gamma_{\min} \) is the minimum weight of the parity-check sequence generated by a weight-2 input pattern. If the maximum weight, that is to be eliminated, is \( d_{\max} \), then the condition that must be satisfied by the interleaver output is

\[
m_1 + m_2 \leq (d_{\max}^2 - 6)/(\gamma_{\min} - 2).
\]

If an interleaver mapping function \( \pi(\cdot) \) meets the condition 

\[
|j_1 - j_2| \mod \delta = 0 \quad \text{and} \quad |\pi(j_1) - \pi(j_2)| \mod \delta = 0,
\]

it will map the input sequence to another weight-2 input sequences that will generate a finite weight parity-check sequence. From Table II, it is found that, the weight-4 input sequence that will produce all zero sequences, is the combination of two weight-2 input sequence. The interleaver maps the input sequence to other weight-4 sequence. After interleaving, total number of patterns of two weight-2s is \( 4 \choose 2 = 6 \). Therefore, if these six patterns follow the condition (8), the mapped sequence will also produce the low weight codeword. In [40], only the four conditions are considered instead of six, without showing any reason. In this thesis, all six combinations are considered. For weight-3 sequence, (7.6) and (7.7) becomes

\[
Q_1(D) = (1 + D^{\delta m_3} + D^{\delta m_2})D^{\varphi_1}
\]  

(7.8)

\[
Q_2(D) = (1 + D^{\delta m_3} + D^{\delta m_2})D^{\varphi_2}
\]  

(7.9)
Here $\lambda$, $\theta$ is the minimum distance of two '1's among the three '1's. $n_1$, $n_2$, $n_3$ and $n_4 = 1, 2, 3, \ldots$. Before the interleaver mapping if the condition $|j_1 - j_2| \mod \lambda = 0$ and $|j_2 - j_3| \mod \theta = 0$ holds, the sequence produces low weight codeword. After the interleaving mapping, if the condition $|\pi(j_1) - \pi(j_2)| \mod \lambda = 0$ and $|\pi(j_2) - \pi(j_3)| \mod \theta = 0$ holds, the sequence also produces the low weight codeword from the 2nd decoder. This is shown in Fig. 7.5. Overall weight of the generated codeword of weight-3 input can be calculated as

$$d = n_1(y_{\min} - 2) + n_2(y_{\min} - 2) + n_3(y_{\min} - 2) + n_4(y_{\min} - 2) + 7,$$

where $n_1$, $n_2$, $n_3$ and $n_4$ has the value of 1, 2, 3, \ldots.

Let $d_{\text{max}}^\lambda$ is the maximum codeword weight for weight-3 input, that to be eliminated, then the condition that must be satisfied by the interleaver output integer is

$$n_1 + n_2 + n_3 + n_4 \leq \frac{(d_{\text{max}}^\lambda - 7)(y_{\min} - 2)}{(y_{\min} - 2)} \quad (7.10)$$

Weight-5 input sequence is the combination of weight-2 and weight-3 sequence. So if the conditions to produce low weight codeword for weight-2 and -3 hold in a weight-5 sequence, then the weight-5 sequence will also produce the low weight codeword. It is shown in Fig. 7.6.

### 7.5. Improvement of the $S$-random Interleaving Algorithm

To eliminate the weight-2, 3, 4, 5 information sequence, following constraints ($\Theta$) should be used with the interleaving algorithm

1. **Weight-2:** $|\pi(j_1) - \pi(j_2)| \mod \delta \neq 0$ if $|j_1 - j_2| \mod \delta = 0$ and $m_1 + m_2 \leq \frac{(d_{\text{max}}^\lambda - 6)(y_{\min} - 2)}{(y_{\min} - 2)}$

2. **Weight-3:** $|\pi(j_1) - \pi(j_2)| \mod \lambda \neq 0$ and $|\pi(j_2) - \pi(j_3)| \mod \theta \neq 0$ if $|j_1 - j_2| \mod \pi = 0$ and $|j_2 - j_3| \mod \theta = 0$ and $n_1 + n_2 + n_3 + n_4 \leq \frac{(d_{\text{max}}^\lambda - 7)(y_{\min} - 2)}{(y_{\min} - 2)}$
Fig. 7.5: Weight-3 sequence before and after interleaving

3) Weight-4: $|\pi(j_1) - \pi(j_3)| \mod \lambda \neq 0$ and

$|f_2 - f_3| \mod \theta = 0$ and $|\pi(j_2) - \pi(j_4)| \mod \lambda \neq 0$

if $|j_1 - j_2| \mod \delta = 0$,

Fig. 7.6: Weight-5 input sequence, before and after interleaving
$n_1 + n_2 + n_3 + n_4 \leq (d_{\text{max}}^4 - 12)/(y_{\text{min}} - 2)$; or $|\pi(j_1) - \pi(j_4)| \mod \lambda \neq 0$ and $|\pi(j_2) - \pi(j_3)| \mod \lambda \neq 0$ if $|j_1 - j_2| \mod d = 0$ and $|j_2 - j_3| \mod d = 0$ and $n_1 + n_2 + n_3 + n_4 \leq (d_{\text{max}}^4 - 12)/(y_{\text{min}} - 2)$; or $|\pi(j_1) - \pi(j_2)| \mod \lambda \neq 0$ and $|\pi(j_3) - \pi(j_4)| \mod \lambda \neq 0$ if $|j_1 - j_2| \mod d = 0$ and $|j_2 - j_3| \mod d = 0$ and $n_1 + n_2 + n_3 + n_4 \leq (d_{\text{max}}^4 - 12)/(y_{\text{min}} - 2)$.

4) Weight-5: $|\pi(j_k) - \pi(j_l)| \mod d = 0$, $|\pi(j_p) - \pi(j_q)| \mod \lambda = 0$, $|\pi(j_r) - \pi(j_s)| \mod \lambda = 0$ if $|j_k - j_l| \mod d = 0$, $|j_p - j_q| \mod \lambda = 0$, $|j_r - j_s| \mod \lambda = 0$, $m_k + m_l \leq (d_{\text{max}}^2 - 6)/(y_{\text{min}} - 2)$ and $n_p + n_q + n_r \leq (d_{\text{max}}^2 - 7)/(y_{\text{min}} - 2)$, where $k \neq l \neq p \neq q \neq r$ and they can take any value from \{1, 2, 3, 4, 5\}.

In $S$-random algorithm, each randomly selected integer is compared to the previously selected integers. If the absolute value of the difference between the current selected integer and any of the $S$ previously selected integers is smaller than $S$, the current selected integer is rejected. An interleaver with $S$ random constraint can either break a short weight-3 input pattern with lengths up to $(S+1)$. However, all weight-3, 5 inputs cannot be broken by $S$-random interleaver. So to eliminate these low weight inputs, following improvements of the $S$-random interleaver are suggested:

1) Each output integer of $S$-random interleaver should satisfy the constraint set ($\Theta$). If it does not satisfy it is rejected.

2) If no interleaver output satisfy ($\Theta$) after all iterations, $S$ value is reduced by 1. Then 1) repeats.

The same constraint sets can be used with the interleaving algorithm of UMTS and cdma2000 standards and their performance can be improved.
Fig. 7.7: BER performance of 4-state Turbo codes for different interleavers, 1/3 rate, 
\( G = (1, (1 + D^2)/(1 + D + D^2)) \) Frame size: 1000, iteration: 10) (0.15 dB gain for BER at 10^{-6})

Fig. 7.8: BER performance of 4-state Turbo codes for different interleavers, 
1/3 rate, \( G = (1, (1 + D^2)/(1 + D + D^2)) \) Frame size: 4000 bits, iteration: 10) 
(0.2 dB gain for BER of 10^{-6})
Interleaver gain for the improved S-random interleaver for 4 state Turbo codes is shown in Fig. 7.7. In this figure it is found that 0.15 dB gain is achieved with respect to S-random interleaver by using this improved S-random interleaver. It is well known that frame size has the effect on the interleaver gain. In our simulation if we increase the frame size, interleaver gain is increased to 0.2 dB, which is shown in Fig. 7.8. This proves that, with the increase of the frame size, interleaver gain is also increased.

To show the effect of decoding iteration on interleaver gain, we decrease the number of decoding iterations from 10 to 5. The frame size, encoder structure, code rate are kept same as Fig. 7.7. The simulation result is shown in Fig. 7.9. From this figure it is found that the interleaver gain is unchanged with the change of decoding iteration. So there is no effect of decoding iteration on the interleaver gain.
Fig. 7.10: BER performance of 4-state Turbo codes for different interleavers, ½ rate, 
$G = (1, (1+D^2)/(1+D+D^2))$ Frame size: 1000, iteration: 10) (0.15 dB gain for BER of $10^{-6}$)

Fig. 7.11: BER performance of 8-state Turbo codes for different interleavers, 
1/3 rate, $(G = (1, (1+D+D^2+D^3)/(1+D+D^3))$ Frame size: 1000, iteration: 10) 
(0.15 dB gain for BER of $10^{-7}$)
Fig. 7.12: BER performance of 8-state Turbo codes for different interleavers, 1/3 rate. 
\[ G = \frac{(1 + D + D^2 + D^3)(1 + D^2 + D^3)}{(1 + D + D^2 + D^3)} \] Frame size: 4000, iteration: 10 (0.2 dB gain for BER of 10^-7)

To show the influence of code rate on the interleaving gain we simulate the Turbo codes with \( \frac{1}{2} \) code rate keeping all the other parameters of Turbo encoder and decoder same as Fig. 7.7. Simulation result is shown in Fig. 7.10. The interleaver gain is not changed and is equal to 0.15 dB. So it can be said that interleaver is not dependent on code rate.

Now the effect of increasing encoder state on interleaver gain is shown in Fig. 7.8 and Fig. 7.12. The number of encoder state has no effect on the interleaving gain. For 8-state Turbo codes with frame size of 1000 has the interleaver gain of 0.15 dB and Turbo codes with frame size of 4000 has the interleaver gain of 0.2 dB. So if the interleaver size is increased interleaver gain is also increased.
Fig. 7.13: FER performance of 4-state Turbo codes for different interleavers, 1/3 rate, 
\[ G = \frac{1}{(1, (1 + D^2) / (1 + D + D^2))} \] 
Frame size: 1000, iteration: 10, (0.15 dB gain for FER of 10^{-3})

Other performance parameter of Turbo codes is frame error rate (FER). The pattern of performance curve of FER is similar to that of bit error rate. We have simulated Turbo codes to investigate the effect of frame size on the interleaver gain in frame error rate performance and have shown in Fig. 7.13 and Fig. 7.14. It is found that if we increase the frame size from 1000 to 4000, we get the interleaver gain increased from 0.15 dB to 0.2 dB at bit error rate of 10^{-6}.  

127
Fig. 7.14: FER performance of 4-state Turbo codes, $G = (1, (1 + D^2)/(1 + D + D^2))$, 1/3 rate, Frame size: 4000, iteration: 10, (0.2 dB gain for FER of $10^{-3}$)

From the above results it can be conclude that interleaver design with the consideration of weight 2, 3, 4 and 5 inputs that produce low weight codewords improve the Turbo code performance at low bit error rate. It is found that for the frame size of 1000, 0.15 dB interleaver gain can be achieved by improved $S$-random interleaver with respect to $S$-random interleaver at BER of $10^{-6}$ for 4-state and 8-state Turbo codes. If the frame size is increased to 4000, both the 4-state and 8-state Classical Turbo codes show the interleaver gain of 0.2 dB with respect to $S$-random interleaver at BER of $10^{-6}$. Then the decoding iterations, code rate and number of encoder state are changed to investigate their effect on interleaver gain. But we don't get any influence of these parameters on the interleaver gain. Interleaver gain of improved $S$-random interleaver is also shown in the FER performance. It is found that for the frame size of 1000 the gain is 0.15 dB and for the frame size of 4000 the gain is 0.2 dB at the FER of $10^{-3}$. 

128
7.6. Latency For Interleaver Design

Interleaving algorithm has the effect on decoding latency. In this section we estimate the latency caused by different interleavers. Mainly we measure the simulation time taken by Turbo encoder and decoder with different interleavers. We use the Pentium IV machine with 2.8 GHz and 32-bit processor and the bus speed is 800 MHz for the simulation. With this processor, 256 MB RAM with bus speed 400 MHz is used. We choose the Turbo encoder as

\[ G = (1, (1 + D + D^2)/(1 + D^2)) \]

The channel is modeled as AWGN and code rate is 1/2. Programming language for the simulator is chosen as MATLAB. The simulation time is averaged for 1 frame of data. The results are shown in Table 7.4 for varying frame size and in Table 7.5 for varying number of iterations.

Table 7.4: Simulation time for different algorithm with 5 decoding iterations

<table>
<thead>
<tr>
<th>Frame size</th>
<th>Simulation time (Sec.) for different interleavers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Random Interleaver</td>
</tr>
<tr>
<td>378</td>
<td>1.091691</td>
</tr>
<tr>
<td>570</td>
<td>1.735547</td>
</tr>
<tr>
<td>762</td>
<td>2.175688</td>
</tr>
<tr>
<td>1146</td>
<td>4.098681</td>
</tr>
</tbody>
</table>

Table 7.4 shows that required simulation time is lowest for Random Interleaver. Simulation time for UTMS Interleaver is greater than that for cdma2000. Simulation time for S-random Interleaver is greater than that for cdma2000 and less than UMTS Interleaver. Simulate time for Turbo code with improved S-random Interleaver is slightly greater than Turbo code with S-random Interleaver as the former is more complicated.
Fig. 7.15: Simulation time of Turbo code with Improved $S$-random Interleaver for different frame size

Table 7.5: Simulation time for different interleaving algorithm with frame size of 570

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Random Interleaver</th>
<th>cdma2000 Interleaver</th>
<th>UMTS Interleaver</th>
<th>$S$-random Interleaver</th>
<th>Improved $S$-random Interleaver</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.497</td>
<td>0.622</td>
<td>0.7094</td>
<td>0.652142</td>
<td>0.656892</td>
</tr>
<tr>
<td>3</td>
<td>1.180894</td>
<td>1.19525</td>
<td>1.302697</td>
<td>1.235148</td>
<td>1.2632581</td>
</tr>
<tr>
<td>5</td>
<td>1.735547</td>
<td>1.777523</td>
<td>1.897555</td>
<td>1.812457</td>
<td>1.845684</td>
</tr>
<tr>
<td>7</td>
<td>2.309967</td>
<td>2.391633</td>
<td>2.487387</td>
<td>2.425641</td>
<td>2.456231</td>
</tr>
<tr>
<td>9</td>
<td>2.926505</td>
<td>2.973957</td>
<td>3.05506</td>
<td>2.975623</td>
<td>2.965421</td>
</tr>
</tbody>
</table>
than the later. From Table 7.4, it is found that for frame size of 1530 improved $S$-random Interleaver takes simulation time of 56.9 ms more than that of $S$-random Interleaver for the increased complexity. It is also found that simulation time is increased with the increase of frame size in almost linear fashion. This is shown in Fig. 7.15.

From Table 7.5 it is found that required simulation time is lowest for Turbo code with random interleaver. Simulation time for with Improved $S$-random Interleaver is greater than that for with $S$-random Interleaver. Simulation time is increased with the increase of number of decoding iteration. The increase of simulation time with the increase of number of decoding iteration is linear and it is also shown in Fig. 7.16
Chapter 8

Conclusion

8.1 Introduction

Quality of Service of wireless communication is one the important research area to the communication engineers and scientists. ITU-T, ETS, UMTS, cdma2000, all standard bodies for wireless communication recently trying to develop the concept of quality of service and also define QoS parameters. One of the important QoS parameters in wireless communication recognized by all standard bodies and researchers is BER. To improve BER error control mechanism must be used. For wireless and mobile communication, which is developing with real time application like multimedia, video streaming, audio data, BER improvement is one of the important demand. This thesis investigates the error control codes to improve the BER in wireless communication.

If the wireless channel is modeled as AWGN and BPSK modulation is used without error correcting codes 9.6 dB is required to achieve BER of $10^{-5}$. Block, Cyclic and Convolutional codes were invented but these codes could not reach the Shannon's limit. Only Turbo codes can reach the Shannon limit and provided the coding gain of 9.2 dB with coding rate of 0.5.

As Turbo code shows the extraordinary performance, coding communities are trying to reveal the causes behind its performance. Now it is found that the encoder structure, constraint length of the encoder, size of the interleaver, interleaving algorithm, type of decoding algorithm, number of decoding iteration have the effect on the BER and FER performance of Turbo codes. The other performance measure of Turbo codes is its distance spectrum. The "error floor" that occurs at moderate signal-to-noise ratios is shown to be a consequence of the relatively low free distance of the code. It is also shown that increasing the size of the interleaver without changing the free distance of the code can lower the "error floor". Alternatively, designing the interleaver may increase the free distance of the code. So interleaver design is other important aspect of Turbo codes. Encoder structure has the influence on BER, FER and distance spectrum of Turbo codes.
For its extraordinary performance now Turbo used in third generation wireless communication standard as UMTS and cdma2000 standard error correcting codes, CCSDS, PCS etc. Turbo principle can also be used in joint source-channel decoding, joint channel estimation and decoding, and multi-user communications.

8.2 Contributions

Effects of frame size, number of decoding iterations, code rate, decoding algorithm, channel models in the BER and FER performance of classical, UMTS, cdma2000 Turbo codes are shown.

Distance spectrum of Classical, UMTS, cdma2000 and CCSDS Turbo codes for varying frame size and code rate are measured. Effects of different frame size and coder rate on distance spectrum are shown.

An improvement in the $S$-random interleaving algorithm has been suggested. Improvement is shown in terms of interleaver gain in BER and FER curve for varying code rate, number of decoding iteration, constraint length and frame size. Simulation time of encoding and decoding of Turbo Codes with different interleaving algorithms are shown. Increased decoding complexities due to the improvement of $S$-random interleaving algorithm is shown in terms of increased simulation time.

Influence of encoder structure of Turbo codes on its performance is extensively studied. An encoder structure is identified which shows better performance than UMTS standard Turbo encoder.
8.3 Suggestions for Further Research

In this thesis mainly BER performance as quality of service parameter is emphasized. We discussed about different aspects of Turbo codes mainly to improve the BER performance. But other quality of service parameter like delay, delay variation and throughput are not emphasized. So there is a scope to investigate on delay and throughput related to Turbo codes.

Newer interleaving algorithm can be developed so that the free distance of Turbo code is increased. Algebraic interleaver and Bolt interleaver [61] may be used with Turbo codes. In UMTS interleaver and cdma2000 interleaver can be redesigned so that low weight input pattern may be broken by using dummy bits.

If a feedback path is available between sender and receiver, then the ARQ can be incorporated with the Turbo Coded system. In the original ARQ system, the corrupted packet is retransmitted if the receiver detects undetectable errors. To reduce the delay for transmission only the parity bits of the Turbo Codes can be retransmitted.

Turbo code's application can be extended to all the wireless communication and storage application. Turbo code can be used with personal communication service (PCS), joint channel estimation and decoding, and multi-user detection [62], CD recording and OFDM [63].

Number of encoders and number of interleavers can be increased to improve the performance. But with the increase of encoders code rate is decreased. So puncturing may be applied to increase the code rate. By thorough investigation best configuration can be identified.

It is well known that serial concatenation [64] performs better for high $E_b / N_0$ and parallel concatenation [5] performs better for low $E_b / N_0$. So if both types of concatenations are combined, best Turbo codes can be invented.

We have studied the performance for AWGN and Rayleigh channel model. So it should be find out how Turbo codes behaves with Rician channel [65] model.
Appendix I

Log-likelihood Algebra

The log-likelihood algebra used for SOVA decoding of Turbo codes is based on a binary random variable $u$ in $GF(2)$ with elements $\{+1,-1\}$, where $+1$ is the logic 0 element ("null" element) and $-1$ is the logic 1 element under modulo 2 addition. Table I shows the outcome of adding two binary random variables under these governing factors.

**Table I** Outcome of Adding Two Binary Random Variables $u_1$ and $u_2$

<table>
<thead>
<tr>
<th>$u_1$</th>
<th>$u_2$</th>
<th>$u_2 = +1$</th>
<th>$u_2 = -1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1 = +1$</td>
<td>$+1$</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>$u_1 = -1$</td>
<td>-1</td>
<td>$+1$</td>
<td></td>
</tr>
</tbody>
</table>

The log-likelihood ratio $L(u)$ for a binary random variable $u$ is defined to be

$$L(u) = \ln \frac{P(u = +1)}{P(u = -1)}$$

$L(u)$ is often denoted as the "soft" value or $L$-value of the binary random variable $u$. The sign of $L(u)$ is the hard decision of $u$ and the magnitude of $L(u)$ is the reliability of this decision. Table II shows the characteristics of the log-likelihood ratio $L(u)$.

Clearly from Table II, as $L(u)$ increase toward $+\infty$, the probability of $u = +1$ also increases. Furthermore, as $L(u)$ decreases toward $-\infty$, the probability of $u = -1$ increases.

As it can be seen, $L(u)$ provides a form of reliability for $u$.

The probability of the random variable $u$ may be conditioned on another random variable $z$. This forms the conditioned log-likelihood ratio $L(u \mid z)$ and is defined to be

$$L(u \mid z) = \ln \frac{P(u = +1 \mid z)}{P(u = -1 \mid z)}$$
Table II Characteristics of the Log-likelihood Ratio $L(u)$

<table>
<thead>
<tr>
<th>$P(u = +1)$</th>
<th>$P(u = -1)$</th>
<th>$L(u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>+∞</td>
</tr>
<tr>
<td>0.9999</td>
<td>0.0001</td>
<td>9.2102</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1</td>
<td>2.1972</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
<td>0.4055</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td>-0.4055</td>
</tr>
<tr>
<td>0.1</td>
<td>0.9</td>
<td>-2.1972</td>
</tr>
<tr>
<td>0.0001</td>
<td>0.9999</td>
<td>-9.2102</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-∞</td>
</tr>
</tbody>
</table>

$L(u | z) = \ln \frac{P(u = +1 | z)}{P(u = -1 | z)}$

The probability of the sum of two binary random variables, say $P(u_1 \oplus u_2 = +1)$, is found from

$P(u_1 \oplus u_2 = +1) = P(u_1 = +1)P(u_2 = +1) + P(u_1 = -1)P(u_2 = -1)$

With the following relation

$P(u = -1) = 1 - P(u = +1)$

The probability $P(u_1 \oplus u_2 = +1)$ becomes

$P(u_1 \oplus u_2 = +1) = P(u_1 = +1)P(u_2 = +1) + (1 - P(u_1 = +1))(1 - P(u_2 = -1))$

Using the following relation shown in

$P(u = +1) = \frac{e^{L(u)}}{1 + e^{L(u)}}$

It can be shown that

$P(u_1 \oplus u_2 = +1) = \frac{1 + e^{L(u_1)}e^{L(u_2)}}{(1 + e^{L(u_1)})(1 + e^{L(u_2)})}$

The probability $P(u_1 \oplus u_2 = -1)$ can then be calculated as

$P(u_1 \oplus u_2 = -1) = (1 - P(u_1 \oplus u_2 = +1))$
From the definition of log-likelihood ratio (5.1), it follows directly that

\[ L(u_1 \oplus u_2) = \ln \frac{P(u_1 \oplus u_2 = +1)}{P(u_1 \oplus u_2 = -1)} \]

Using (5.7) and (5.9), \( L(u_1 \oplus u_2) \) is found to be

\[ L(u_1 \oplus u_2) = \ln \frac{1 + e^{L(u_1)} e^{L(u_2)}}{e^{L(u_1)} + e^{L(u_2)}} \]

This result is approximated in as

\[ L(u_1 \oplus u_2) \approx \text{sign}(L(u_1)) \text{sign}(L(u_2)) \min(|L(u_1)|, |L(u_2)|) \]

The addition of two "soft" or \( L \)-values is denoted by \([+]\) and is defined as

\[ L(u_1)[+]L(u_2) = L(u_1 \oplus u_2) \]

With the following three properties

\[ L(u)[+]\infty = L(u) \]
\[ L(u)[+](\infty) = -L(u) \]
\[ L(u)[+]0 = 0 \]

By induction, it can be shown that

\[ \sum_{j=1}^{I} L(u_j) = L \left( \sum_{j=1}^{I} u_j \right) \]

\[ = \ln \frac{P \left( \sum_{j=1}^{I} u_j = +1 \right)}{P \left( \sum_{j=1}^{I} u_j = -1 \right)} \]

\[ = \ln \frac{\prod_{j=1}^{I} (e^{L(u_j)} + 1) + \prod_{j=1}^{I} (e^{L(u_j)} - 1)}{\prod_{j=1}^{I} (e^{L(u_j)} + 1) - \prod_{j=1}^{I} (e^{L(u_j)} - 1)} \]

By using the relation
The induction can be simplified to

\[ \sum_{j=1}^{J} L(u_j) = \ln \frac{1 + \prod_{j=1}^{J} \tanh(\frac{L(u_j)}{2})}{1 - \prod_{j=1}^{J} \tanh(\frac{L(u_j)}{2})} \]

\[ = 2 \tanh^{-1} \left( \prod_{j=1}^{J} \tanh(\frac{L(u_j)}{2}) \right) \]

This value is very tedious to complete. Thus, it can be approximated as before to

\[ \sum_{j=1}^{J} L(u_j) = L \left( \sum_{j=1}^{J} u_j \right) \]

\[ = \left( \prod_{j=1}^{J} \text{sign}(L(u_j)) \right) \min_{j=1,\ldots,J} \{ |L(u_j)| \} \]

So the reliability of the sum of “soft” or L-values is mainly determined by the smallest “soft” or L-value of the terms.
Appendix II

Reliability of Path Decision

The probability of path \( m \) at time \( t \) and the SOVA metric are stated in to be related as

\[
P(path(m)) = P(S_t^{(m)})
\]

At time \( t \), let us suppose that the survivor metric of a node is denoted as \( M_t^{(1)} \) and the competing metric is denoted as \( M_t^{(2)} \). Thus, the probability of selecting the correct survivor path is

\[
P(correct) = \frac{P(path(1))}{P(path(1)) + P(path(2))}
\]

\[
= \frac{\frac{M_t^{(1)}}{e^2}}{\frac{M_t^{(1)}}{e^2} + \frac{M_t^{(2)}}{e^2}}
\]

\[
= \frac{e^\Delta_t}{1 + e^\Delta_t}
\]

The reliability of this path decision is calculated as

\[
\log \frac{P(correct)}{1 - P(correct)} = \log \frac{e^{\Delta_t^0}}{1 + e^{\Delta_t^0}}
\]

\[
= \Delta_t^0
\]
Appendix III

Extrinsic Value Calculation

The path metric is defined as

$$M_k(S^{(i)}) = M_{k-1}(S^{(i)}) + 1/2 I_c(u_k)u_k^{(i)} + 1/2 I_c y_{k,1}u_k^{(i)} + 1/2 \sum_{v=2}^{n} I_c y_{k,v}x_{k,v}^{(i)}$$

And reliability value is defined as

$$\Delta_k^l = M_{k+l}(S^{(i)}) - M_{k+l}(S^{(j)})$$

Decoder's Soft value can be approximated as

$$L(\bar{u}_k) \approx \bar{u}_k \sum_{l=0}^{S} \Theta_{k} \approx \min_{l=0,\ldots,\delta} \Delta_k^l$$

For the first two equations it is found that

$$\Delta_k^l = (M_{j<k}^{(1)} - M_{j<k}^{(2)}) + (M_{k<j<k+1}^{(1)} - M_{k<j<k+1}^{(2)}) + 1/2 \sum_{v=2}^{n} I_c y_{k,v}(x_{k,v}^l - x_{k,v}^o) + 1/2 I_c y_{k,1}(\bar{u}_k - (-\bar{u}_k)) + 1/2 L(\bar{u}_k)(\bar{u}_k - (-\bar{u}_k))$$

After some derivation following is found:
Multiplying $\Delta_k^i$ by $\tilde{u}_k$ we get the soft value as follows

\[
L_{\text{sova}} = (M_{j<k}^{(1)} - M_{j<k}^{(2)}) + (M_{k<j<k+1}^{(1)} - M_{k<j<k+1}^{(2)})
\]
\[+ \frac{1}{2} \sum_{v=2}^{n} L_c y_{k,v} (x_{k,v}^1 - x_{k,v}^2)
\]
\[+ L_c y_{k,1} \tilde{u}_k
\]
\[+ L(\tilde{u}_k)
\]

It can be rewrite as:

\[
L_{\text{sova}} = L_e(\tilde{u}_k) + L_c y_{k,1} + L(\tilde{u}_k)
\]

Where $L_e(\tilde{u}_k)$ is

\[
( M_{j<k}^{(1)} - M_{j<k}^{(2)}) + (M_{k<j<k+1}^{(1)} - M_{k<j<k+1}^{(2)})
\]
\[+ \frac{1}{2} \sum_{v=2}^{n} L_c y_{k,v} (x_{k,v}^1 - x_{k,v}^2)
\]

The extrinsic information, which should be passed from one decoder to other decoder, is as the above expression.

We can get it by subtracting $L(\tilde{u}_k)$ and $L_c y_{k,1}$ from $L_{\text{sova}}$. $L_{\text{sova}}$ is found by the calculation in the trellis. $L_e(\tilde{u}_k)$ is used for the next SOVA decoder.
Appendix IV

List of Symbols

\( k \)  
Length of information bits

\( n \)  
Length of codeword

\( C \)  
Channel Capacity

\( r \)  
Code rate

\( g(D) \)  
Generator Polynomial

\( E_b / N_0 \)  
Bit energy per noise power spectral ratio

\( d_{\text{min}} \)  
Minimum Hamming Distance

\( P_{\text{free}} \)  
Free distance asymptotes

\( d_{\text{free}} \)  
Free distance of codewords

\( N_{\text{free}} \)  
Multiplicity of free-distance codewords

\( w_{\text{free}} \)  
Total weight of the information sequences causing free-distance codewords

\( \bar{w}_{\text{free}} \)  
Average weight of the information sequences causing free-distance codewords

\( \pi(j) \)  
Interleaved position of \( j \)

\( \Theta \)  
Constraint set for \( S \)-random

Interleaver
### List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>FER</td>
<td>Frame Error Rate</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RSC</td>
<td>Recursive Systematic Convolutional</td>
</tr>
<tr>
<td>PCCC</td>
<td>Parallel Concatenation of Convolutional Codes</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to noise ratio</td>
</tr>
<tr>
<td>SCCC</td>
<td>Serial Concatenation of Convolutional Codes</td>
</tr>
<tr>
<td>AWGN</td>
<td>Additive White Gaussian Noise</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunication Systems</td>
</tr>
<tr>
<td>CCSDS</td>
<td>Consultative Committee for Space Data System</td>
</tr>
<tr>
<td>ARQ</td>
<td>Automatic Repeat Request</td>
</tr>
<tr>
<td>SOVA</td>
<td>Soft-Output Viterbi Algorithm</td>
</tr>
<tr>
<td>DVB</td>
<td>Digital Video Broadcasting</td>
</tr>
<tr>
<td>PCS</td>
<td>Personal Communication Services</td>
</tr>
<tr>
<td>CoS</td>
<td>Class of Service</td>
</tr>
<tr>
<td>GoS</td>
<td>Grade of Service</td>
</tr>
<tr>
<td>IRWEF</td>
<td>Input Redundancy Weight Enumerating Function</td>
</tr>
</tbody>
</table>
References


144


[57] FITCE Study Commission: The study of Network defines an optimal degree of use of information exchange Performance considering Customer Framework.


[59] TINA-C: Quality of Service framework, doc. no TR_MRK.001_1.0_94.

[60] EURESCOM P603: Quality of Service: Measurement method selection, http://www.eurescom.de/Public/Projects/p600series/P603/P603.HTM


