UNIVERSITY OF SOUTHAMPTON
FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Cross-Layer Operation Aided Wireless Networks

by

Hong Chen

A thesis submitted in the partial fulfilment of the requirements for the award of Doctor of Philosophy at the University of Southampton

October 2010

SUPERVISOR:
Prof. Lajos Hanzo and Dr. Robert G. Maunder

University of Southampton
Southampton SO17 1BJ
United Kingdom

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

A thesis submitted in the partial fulfilment of the requirements for the award of Doctor of Philosophy

Cross-Layer Operation Aided Wireless Networks

by Hong Chen

In this thesis, we propose several cross-layer operation aided schemes conceived for wireless networks. Cross layer design may overcome the disadvantages of the network's layered architecture, where layering is most typically represented by the Transport Control Protocol (TCP) / Internet Protocol (IP) suite.

We invoke Fountain codes for protecting file transfer at the application layer, since they are suitable for erasure channels. They are also often referred to as rateless codes. When implementing Fountain code aided file transfer, the file will be firstly partitioned into a number of blocks, each of which contains K packets. Fountain codes randomly select several packets from a block and then combine them using exclusive-OR additions for generating an encoded packet. The encoding continues until all blocks are successfully received. Considering an 802.11 Wireless Local Area Network (WLAN) scenario, the packet size has to be appropriately chosen, since there exists a trade-off between the packet size and the transmission efficiency, which is defined as the number of primary information bits to the total number of all transmitted bits including headers, control packets and retransmitted replicas. In order to find the optimum packet size, the transmission efficiency is formulated as a function of the Packet Loss Ratio (PLR) at the application layer and of the total load imposed by a single packet. The PLR at the application layer is related both to the packet size, as well as to the 802.11 MAC retransmission mechanism and to the modulation scheme adopted by the physical layer. Apart from its source data, the total load imposed by an information packet also contains the control packets of the 802.11 Media Access Control (MAC) protocol such as the Request To Send (RTS) / Clear To Send (CTS) messages, the retransmitted replicas and the Acknowledgement (ACK) messages. According to these relations, the transmission efficiency may finally be expressed as a function of packet size. Based on the numerical analysis of this function, the optimum packet size may be determined. Our simulation results confirmed that indeed the highest transmission efficiency may be achieved, when using the optimum packet size.

Since turbo codes are capable of achieving near capacity performance, they may be successfully combined with Hybrid Automatic Repeat reQuest (HARQ) schemes. In this thesis, the classic Twin Component Turbo Codes (TCTCs) are extended to Multiple Component Turbo Codes (MCTCs). In order to apply classic two-dimensional Extrinsic Information Transfer (EXIT) charts for analyzing them, we divided an N-component MCTC into two logical parts. This partitioning was necessary, because otherwise an N-component scheme would require an N-dimensional EXIT chart. One of the parts is constituted by an individual Bahl, Cocke, Jelinek and Raviv (BCJR) decoder, while the other so-called composite decoder consists of the remaining $(N-1)$ components. The EXIT charts visualized the extrinsic information exchange between these two logical parts of MCTCs. Aided by this partitioning technique, we may find the so-called 'open tunnel SNR threshold' for MCTCs, which is defined as the minimum SNR for which the EXIT chart at the specific coding rate used has an open tunnel. It may be used as a metric to compare the achievable performance to the Discrete-input Continuous-output Memoryless Channel's (DCMC) capacity. Our simulation results showed that the achievable performance of MCTCs is closer to the DCMC capacity than that of non-systematic TCTCs, but a bit further than that of systematic TCTCs, if generator polynomials having an arbitrary memory length - and hence complexity - are considered. However, for the lowest-memory octally represented polynomial $(2,3)_o$, which implies having the lowest possible complexity, MCTCs outperform non-systematic and systematic TCTCs. Furthermore, MCTC aided HARQ schemes using the polynomial of $(2,3)_o$ exhibit significantly better PLRs and throughput performances than systematic as well as non-systematic TCTC aided HARQ schemes using the same polynomial. If systematic TCTC aided HARQ schemes relying on the polynomial of $(17,15)_o$ are used as benchmarkers, MCTC aided HARQ schemes may significantly reduce the complexity, without a substantial degradation of the PLR and throughput.

When combining turbo codes with HARQ, the associated complexity becomes a critical issue, since iterative decoding is immediately activated after each transmission. In order to reduce the associated complexity, an Early Stopping (ES) strategy was proposed in this thesis to substitute the fixed number of BCJR operations invoked for each iterative decoding. By observing the EXIT charts of turbo codes, we note that the extrinsic information increases along the decoding trajectory of an open or closed tunnel. The ES aided MCTC HARQ scheme curtails iterative decoding, when the Mutual Information (MI) increase becomes less than a given threshold. This threshold was determined by an off-line training in order to achieve a trade-off between the throughput and complexity. Our simulation results verified that the complexity of

MCTC aided HARQ schemes may be reduced by as much as 80%, compared to that of systematic TCTC aided HARQ schemes using a fixed number of 10 BCJR operations.

Moreover, the complexity of turbo coded HARQ schemes may be further reduced by our Look-Up Table (LUT) based Deferred Iteration (DI) method. The DI method delays the iterative decoding until the receiver estimates that it has received sufficient information for successful decoding, which may be represented by the emergence of an open tunnel in the EXIT chart corresponding to all received replicas. Therefore, the specific MI that a 'just' open tunnel appears when combining all previous $(i-1)$ MIs will be the threshold that has to be satisfied by the ith reception. More specifically, if the MI received during the ith reception is higher than this threshold, the EXIT tunnel is deemed to be open and hence the iterative decoding is triggered. Otherwise, iterative decoding will be disabled when the tunnel is deemed to be closed. This reduces the complexity. The LUT stores all possible MI thresholds for N-component MCTCs, which results in a large storage requirement, if N becomes high. Hence, an efficient LUT design was also proposed in this thesis. Our simulation results demonstrated the achievable complexity reduction may be as high as 50%, compared to the schemes operating without the DI method.

Acknowledgements

I would like to express my heartfelt thanks to my supervisor Professor Lajos Hanzo for his outstanding supervision and support throughout my research. His constant inspiration and unfailing encouragement have greatly benefited me. Especially, his diligence and endless energy deserve my sincere respect.

Moreover, I wish to thank my second supervisor Dr. Rob G. Maunder for his patient and professional guidance. I may not complete this thesis without those thorough discussions with him. I am very grateful for his careful explanation and unreserved teaching.

I also appreciate the help from the other staffs of the communication group, Professor Lie-Liang Yang, Professor Sheng Chen and Dr. Soon Xin Ng. Many thanks to all colleagues for their help, discussions and friendship throughout my PhD study. Special thanks to Mr. and Mrs. Robson and my lovely housemates who make my life in UK full of fun and joy. The financial support of the China-UK Scholarship Council, of the EPSRC, UK under the IU-ATC initiative and of the EU under the auspices of the Optimix project is gratefully acknowledged.

Finally, this thesis is dedicated to my family.

Contents

List of Publications

Journal papers

1. **H. Chen, R. G. Maunder and L. Hanzo**, "Low-Complexity Multiple-Component Turbo Decoding Aided Hybrid ARQ", *IEEE Transactions on Vehicular Technology*, vol. 60, pp. 1571-1577, May 2011.
2. **H. Chen, R. G. Maunder and L. Hanzo**, "Look-up Table Based Deferred-Iteration Aided Low-Complexity Turbo Hybrid ARQ", *IEEE Transactions on Vehicular Technology*, vol. 60, pp. 3045 - 3053 September 2011.
3. **H. Chen, R. G. Maunder and L. Hanzo**, "Low-Complexity Hybrid Automatic Repeat Request", *Communications Surveys & Tutorials*, (under review).

Conference papers

1. **H. Chen, R. G. Maunder and L. Hanzo**, "Fountain-Code Aided File Transfer in 802.11 WLANs", *Proceedings of IEEE Vehicular Technology Conference (VTC) Fall*, Anchorage, AK, September 2009.
2. **H. Chen, R. G. Maunder and L. Hanzo**, "Multi-level Turbo Decoding Assisted Soft Combining Aided Hybrid ARQ", *Proceedings of IEEE Vehicular Technology Conference (VTC) Spring*, Taipei, Taiwan, May 2010.
3. **H. Chen, R. G. Maunder and L. Hanzo**, "An EXIT-Chart Aided Design Procedure for Near-Capacity N-Component Parallel Concatenated Codes", *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Miami, USA, December 2010.
4. **H. Chen, R. G. Maunder and L. Hanzo**, "Deferred-Iteration Aided Low-Complexity Turbo Hybrid ARQ Relying on a Look-up Table", *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Houston, USA, December 2011.

List of Symbols

Information theory

$\mathbf{X}$	Random variable denoting source symbols.
x_k	Values for the random variable $\mathbf{X}$.
$P(x_k)$	Probability that the random variable $\mathbf{X}$ takes the value of x_k.
$H(\mathbf{X})$	Entropy associated with the random variable of $\mathbf{X}$.
$\mathbf{Y}$	Random variable denoting received symbols.
y_j	Values for the random variable $\mathbf{Y}$.
$I(\mathbf{X};\mathbf{Y})$	Average mutual information.
C	Capacity of a specific channel.

Turbo codes

$\mathbf{a}$	Uncoded information bit sequence.
$\mathbf{b}$	Encoded bit sequence.
s	A trellis state.
s^+	Next state after state s in a trellis.
$\bar{\mathbf{b}}$	Observed symbols corresponding to $\mathbf{b}$.
$\tilde{\mathbf{b}}$	Channel LLR sequence corresponding to $\mathbf{b}$.
$\tilde{\mathbf{a}}$	Information LLR sequence corresponding to $\mathbf{a}$.
$\tilde{\mathbf{a}}^a$	*a priori* LLR sequence.
$\tilde{\mathbf{a}}^e$	Extrinsic LLR sequence.
$\tilde{\mathbf{a}}^p$	*a posteriori* LLR sequence.
$\alpha(s_{k-1})$	Probability that the trellis stays in the state of s_{k-1} at the instant $(k-1)$, when the observed symbol is $\bar{\mathbf{b}}_{j<k}$ before the instant k.
$\beta(s_k)$	Probability that the trellis stays in the state of s_k at the instant (k), when the observed symbol is $\bar{\mathbf{b}}_{j>k}$ after the instant k.
$\gamma(s_{k-1}, s_k)$	Transition probability from the state s_{k-1} to the state s_k, given that the received symbol is $\bar{b}_k$ at the instant k.

$A(s_{k-1})$	Logarithmic form of $\alpha(s_{k-1})$.
$B(s_k)$	Logarithmic form of $\beta(s_k)$.
$\Gamma(s_{k-1}, s_k)$	Logarithmic form of $\gamma(s_{k-1}, s_k)$.
F_{bcjr}	Bahl, Cocke, Jelinek and Raviv (BCJR) function.
F_{exit}	Extrinsic Information Transfer (EXIT) function.

Fountain codes

$\mathbf{G}$	Generator matrix.
$\mathbf{P}$	Source packet vector.
$\mathbf{T}$	Encoded packet vector.
$\mathbf{G}^{-1}$	Inverse generator matrix.
K	Number of source packets.
ϵ	Extra number of packets beyond K required for successful decoding, assuming a perfect channel.
$\rho(d)$	Probability Density Function (PDF) of ideal soliton distribution.
$\mu(d)$	PDF of robust soliton distribution.
$\delta(x)$	PDF of truncated Poisson 1 distribution.

Fountain code aided file transfer

R	Transmission efficiency.
P_{lost}	Packet loss ratio at the application layer.
L_{data}	Packet length of the source data at the application layer.
D	Load imposed by a packet.
L	Packet length at the physical layer.
$P_p(L)$	Function of the packet loss ratio at the physical layer.
P_s	Symbol error ratio.
L_{rts}	Packet length of the Request To Send (RTS) message.
L_{cts}	Packet length of the Clear To Send (CTS) message.
L_h	Header length.
L_{ack}	Packet length of the Acknowledgement (ACK) message.
P_r	Failure probability for a single RTS/CTS exchange.
P_{f_i}	Probability of the ith transmission attempt.
R_1	Retry limit for RTS/CTS exchange.
R_2	Retry limit for data packets.
D_r	Average channel occupancy associated with each RTS/CTS exchange.
D_R	Total RTS/CTS related channel occupancy associated with each transmission.

D_d	Average channel occupancy imposed by each data packet's transmission.

Multiple-component turbo codes (MCTC)

R_c	Component code's rate.
R	Overall coding rate.
A	Area under the inner component decoder's EXIT curve.
C	Discrete-input Continuous-output Memoryless Channel's (DCMC) capacity.
$\mathbf{a}$	Uncoded source bit sequence.
$\mathbf{a}_i$	Interleaved uncoded source bit sequence.
$\mathbf{b}_i$	Encoded bit sequence generated by the *ith* component of a MCTC.
$\mathbf{b}$	Multiplexed encoded bit sequence.
$\tilde{\mathbf{b}}$	Channel LLR sequence corresponding to $\mathbf{b}$.
$\tilde{\mathbf{b}}_i$	Channel LLR sequence de-multiplexed from $\tilde{\mathbf{b}}$ for the *ith* BCJR decoder.
$\tilde{\mathbf{a}}_i^a$	*a priori* LLR sequence for the *ith* BCJR decoder.
$\tilde{\mathbf{a}}_i^e$	Extrinsic LLR sequence output from the *ith* BCJR decoder.
$\tilde{\mathbf{a}}_p$	*a posteriori* LLR sequence.
T_{ind}	EXIT function of the individual BCJR decoder.
T_{comp}	EXIT function of the $(N-1)$-component composite decoder.
T'_{ind}	Fitted EXIT function of the individual BCJR decoder.
T'_{comp}	Fitted EXIT function of the $(N-1)$-component composite decoder.
m_k	First-order polynomial coefficients of the spline-fitted EXIT functions.
n_k	Zero-order polynomial coefficients of the spline-fitted EXIT functions.
$\mathbf{n}_A$	Gaussian random variable for the virtual extrinsic Additive White Gaussian Noise (AWGN) channel.
σ^2	Variance of Gaussian random variable for the virtual extrinsic AWGN channel.
$J(\sigma)$	Accumulated mutual information of the virtual extrinsic AWGN channel.
I	Mutual information value.
$J^{-1}(I)$	Inverse function of $J(\sigma)$.

MCTC aided Hybrid Automatic Request reQuest (HARQ)

D	Cyclic Redundancy Check (CRC) bits.
Q	FEC codewords.
$\tilde{Q}$	Received noisy FEC codewords.
$\tilde{D}$	Received noisy CRC bits.

$\mathbf{u}$	Information bit sequence without CRC bits.
l	Length of the information bit sequence $\mathbf{u}$.
$\tilde{\mathbf{a}}^j$	jth repetition of the systematic LLR sequence.
$\tilde{\mathbf{b}}_1^j$	jth repetition of the channel LLR sequence for the first BCJR decoder.
$\tilde{\mathbf{b}}_2^j$	jth repetition of the channel LLR sequence for the second BCJR decoder.
m	Memory length of the generator polynomials of Recursive Convolutional Codes (RCC).
K	Number of BCJR operations.
$I_k(\tilde{\mathbf{a}}_i^e)$	Mutual information corresponding to the extrinsic LLRs $\tilde{\mathbf{a}}_i^e$ for the kth execution of the ith BCJR decoder.
T_{inc}	Dumping threshold of early stopping strategy.
T_{max}	Convergence threshold of early stopping strategy.
G	Granularity of the deferred iteration Look-Up Table (LUT).
T_i	ith sub-table of the LUT
$I_{th}(i)$	Mutual information threshold stored in the ith sub-table of the LUT.
$I(\tilde{\mathbf{b}}_i)$	Mutual information corresponding to the channel LLRs $\tilde{\mathbf{b}}_i$ for the ith BCJR decoder.
π	Integer vector for permutation.
m_{ik}	First-order polynomial coefficients of the spline fitted-EXIT function for the ith BCJR decoder.
n_{ik}	Zero-order polynomial coefficients of the spline fitted-EXIT functions for the ith BCJR decoder.
r	Retransmission index.
I_{diff}	Mutual information safety margin.

Chapter 1

Introduction

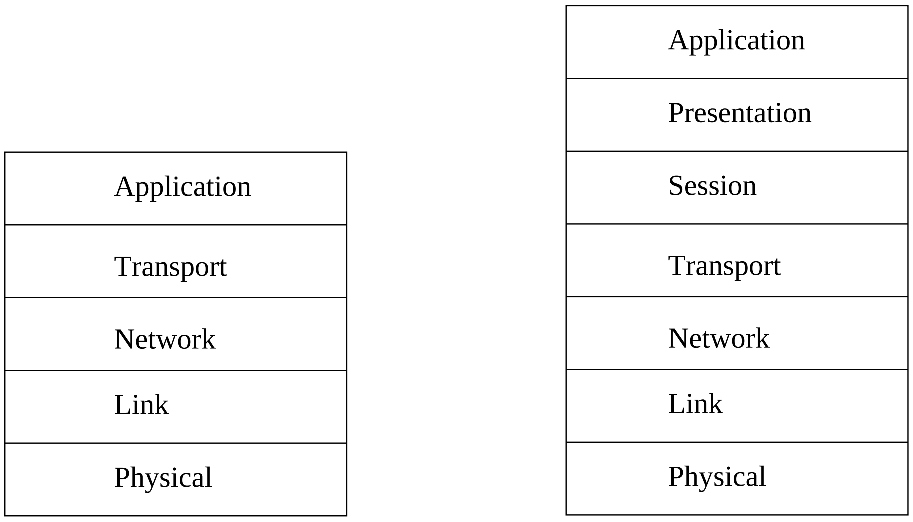
In this chapter, we briefly outline the history of computer networks and introduce their layered architecture. The advantages and disadvantages of traditional layered networks are highlighted, arguing that a cross-layer design may be adopted to overcome the shortcomings of the layered architecture. This thesis aims for improving the performance of wireless networks with the aid of innovative cross-layer designs. Therefore, the cross-layer design concepts are touched upon first. Following a brief state-of-the-art review, we present the organization of this thesis and list its novel contributions.

1.1 Layered Network Architecture

In the early 1960s, telephone networks based on the classic circuit switching technique were dominant, where a physical circuit connection was established between the transmitter and receiver during their communication. As a design alternative, the packet switching technique was conceived in the early 1960s by Leonard Kleinrock, who was then a graduate student at Massachusetts Institute of Technology (MIT). In contrast to circuit switching, packet switching will not establish a circuit connection before transmissions. Instead, in packet switching networks, the source node partitions messages into individual packets and attaches the destination address to each one. These packets are transmitted through the links and switches which are chosen by the network, to the destination. Each packet is independently buffered and forwarded at the switches, where other packets from different sources may have been already queued to wait for forwarding. Thus, the network resources are actually shared among all nodes. The first computer network based on the packet switching technique was planed by the Advanced Research Projects Agency (ARPA) in the United States in 1967. Hence, it was referred to as the ARPAnet, growing to approximately 15 nodes and running end-to-end protocols by 1972.

With the emergence of the first computer network, other proprietary networks were also developed, such as ALOHANet [1], Telenet [2] and IBM's Systems Network Architecture (SNA) [3] etc. The increasing demands of connecting these networks facilitated the development of network protocols, which define the format and order of messages exchanged between the communication nodes and the actions taken following the transmission or reception of them [3]. Back in the late 1970s, three protocols were conceived for the Internet: Transport Control Protocol (TCP) [4], User Datagram Protocol(UDP) [5] and the Internet Protocol (IP) [6]. From then on, the layered concepts were implanted into the protocols design in order to conveniently integrate different networks, which may have diverse applications for different terminals and physical media.

Most networking textbooks like [3, 7] introduce the networks' layered architecture at the beginning. Simply speaking, the network protocols are organized in layers, each of which implements its own functions by invoking the services provided by the layer directly below and then offering services to the next directly above. There are two main protocol stacks in the evolution of computer networks: the seven-layer Open Systems Interconnection (OSI) model, and the five-layer TCP/IP model, which are seen in Figure 1.1.



(a) 5-layer TCP/IP model (b) 7-layer OSI model

Figure 1.1: Two main protocol stacks in computer networks.

The TCP/IP model has become the Internet's protocol stack, which consists of five layers: the application, transport, network, link and physical layers. Here, the following bullets concisely explain each layer's functions from top to bottom.

- Application Layer: provides useful services to network users, who do not have to be concerned about the details of message flow through the network. The application layer contains a number of protocols, such as the Hypertext Transfer Protocol (HTTP) [8] supporting Web browsing, File Transfer Protocol (FTP) [9] supporting file down-loading and the Simple Mail Transfer Protocol (SMTP) [10] conceived for E-mail systems, etc.

- Transport Layer: provides end-to-end transmissions of messages passed to it from the application layer. The TCP and UDP protocols belong to this layer. The TCP protocol supports reliable connection-oriented transmissions, while the UDP provides a connectionless service without error control, flow control and congestion control. By contrast, these functions are part of the TCP protocol.
- Network Layer: responsible for addressing hosts and for forwarding packets from one host to the next. Here, the packets are formed at the transport layer and are passed down to the network layer. This layer includes the well-known IP protocol running on almost all Internet components for their addressing, as well as a lot of routing protocols, which determine the rules of forwarding packets.
- Link Layer: conveys packets along the point-to-point link between two adjacent hosts. The operation of link layer protocols crucially depends on the quality of the physical medium encountered. For example, Institute of Electrical and Electronics Engineers (IEEE) specifies the link layer protocol: the 802.13 for Media Access Control (MAC) in the Ethernet and the 802.11 MAC for Wireless Local Areas Networks (WLANs).
- Physical Layer: controls the actual transmissions over the physical medium, which is often termed as the PHY layer, including techniques such as channel coding, modulation, channel estimation, equalization, etc. Different propagation media may require specific PHY protocols. The above 802.13, 802.11 specifications also define the contents of the PHY layer.

As mentioned, layered network architecture makes it easy to integrate different types of networks. Based on the concept of Figure 1.1, separate layers can be independently designed by different developers, resulting in a high design efficiency. However, the layering philosophy faces some opposition from researchers, who pointed out two major problems. Firstly, some functions are duplicated at different layers. For example, both the TCP and the 802.11 MAC protocols provide error recovery mechanism. The other is that owing to the strict boundaries between two layers, one layer cannot use any of the information present at other layers. Some time-sensitive information may become obsolete, as it passes through all layers.

1.2 Cross-Layer Optimization

Interactive 3-Dimension (3D) games, voice conferencing and real-time videos, as well as a range of novel multimedia applications have become more and more attractive, as the Internet penetrated into the home. Many of the existing low data rate applications, such as web browsing, instant messaging and E-mail impose different Quality of Service (QoS) requirements on the lower layers. Compared to wired transmissions, wireless channels are more hostile. Reflection, refraction, shadowing and the Doppler phenomenon are familiar causes of wireless channel impairments. Intuitively, agile adaption to variable QoS requirements and time varying channel conditions may significantly improve the network's performance. However, the traditional layered network architecture was not designed to satisfy this demand. In recent years, the methodology of cross-layer design was proposed to resolve this problem.

The authors of [11] stated that conscious cross-layer design does not imply discarding the concept of layers, the layering based design still retains its benefits. Cross-layer design may be viewed as a novel network architecture design principle that exploits the interaction among different layers and supports optimization across layer boundaries. More particularly, Cross Layer Operation (CLO) allows the designer to expose the internal layer-specific protocol parameters and functions to other layers, which are hidden in the layered network architecture. Based on the access to other layers' parameters or functions, several separate layers may jointly adapt the transmission strategies to promptly respond to environmental dynamics and hence achieve the desirable performance.

Recently, numerous researchers have made substantial progress in the field of cross-layer design and a plethora of papers have been published on different angles of CLO [11–20]. The authors of [12, 13] proposed the theoretical frameworks of CLO by detailing the cross layer operation problem. The authors of [12] provided a theoretical cross layer framework for Orthogonal Frequency-Division Multiplexing (OFDM) assisted wireless networks, in order to optimize the subcarrier assignment and power allocation problem. The authors of [13] analyzed the disadvantages of a centralized approach and introduced a layered Dynamic Programming (DP) technique, which can retain the benefits of layering, while achieving a performance close to that of the centralized scheme. Furthermore, some authors [11, 14] advocated practical cross-layer approaches conceived for delay-constrained applications in order to improve the reconstructed video quality. Other researchers considered specific scenarios, such as sensor networks [15, 16], or specific PHY techniques, such as OFDM [12, 17]. Furthermore, CLO also leads to energy-efficient designs, since the power consumption is influenced by all layers ranging from silicon to applications [18]. Many authors focused their attention on the combination of the application layer and the lower layers such as the MAC and PHY. The authors of [15] proposed a joint routing and MAC protocol for reducing the signaling overhead, while the authors of [19, 20] suggested a cross layer design combining the TCP protocol and the routing protocol for the sake of achieving an increased throughput.

Motivated by these previous effects, this thesis will also adopt a cross-layer operation aided approach in wireless networks. We dedicate special attention to coding techniques applied in different layers to enhance the attainable transmission reliability, and hence to improve the system's performance. Furthermore, the CLO schemes designed in this thesis endeavor to strike a desirable tradeoff between conflicting performance metrics, such as throughput, Packet Loss Ratio (PLR) and complexity.

1.3 Objectives and Organization of the Thesis

In his breakthrough paper [21], Shannon defined the scope of information theory, which has facilitated the development of coding techniques. Error control in communication systems may rely on diverse codes. Fountain codes have been designed for erasure channels. They are usually employed to protect transmissions at the application layer, while turbo codes have been conceived for channel coding at the PHY layer. In this thesis, we investigate their employment in wireless networks from a CLO perspective. Specifically, the optimal choice of Fountain codes parameters is explored to assist the file transfer related operations at the application layer in IEEE 802.11 WLANs, while relying on the specific properties of the 802.11 MAC protocol and on the PHY layer's modulation scheme. Additionally, we investigate Hybrid Automatic Repeat reQuest (HARQ) schemes by combining them with turbo codes. HARQ schemes also rely on cross-layer design, since they combine the retransmission at the MAC layer and the channel coding at the PHY layer. This thesis will improve the cross-layer operation of HARQ schemes by using Multiple Components Turbo Codes (MCTCs) to exploit all the Incremental Redundancy (IR) transmitted from all retransmissions.

The thesis is organized as follows.

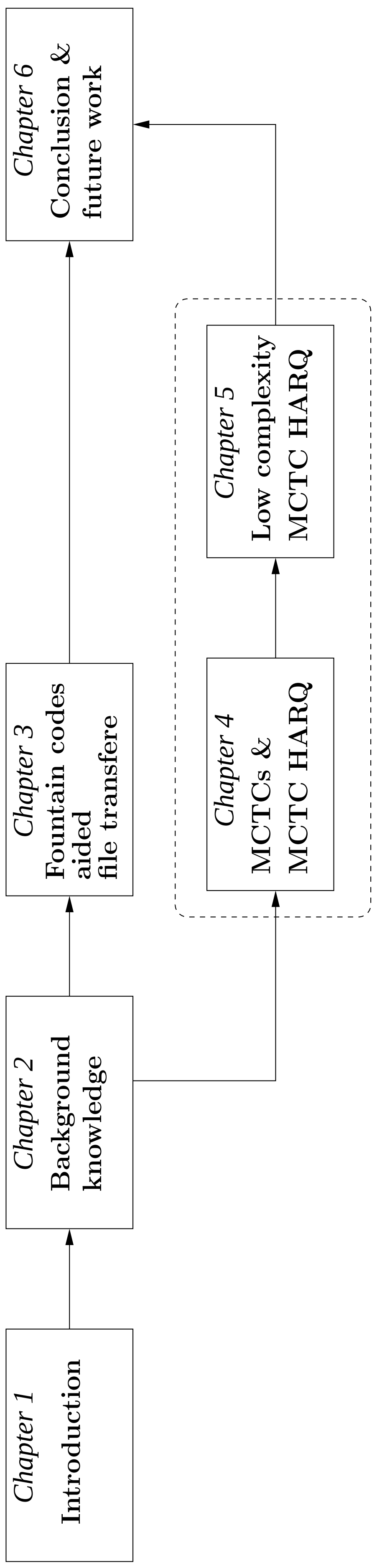


Figure 1.2: Thesis outline.

- Chapter 2 introduces the key techniques employed in this thesis. We commence with a discussion of Shannon's capacity theorem in Section 2.1. Then, the fundamentals of Fountain codes and turbo codes including their encoding and decoding processes are detailed in Sections 2.2 and 2.3. Additionally, the software tools of NS2 and IT++ are also described in Section 2.4.
- Chapter 3 exploits the cross-layer application of Fountain codes in the scenario of IEEE 802.11 WLANs. In general, Fountain codes operate on a number of packets of a file to be transmitted. When they are employed to protect file transfer at the application layer, the packet length predetermines the attainable transmission efficiency, since the packet headers will be appended to the packet by each of the lower layers. In order to find the optimal packet length, Section 3.1 firstly reviews the state-of-the-art in the applications of Fountain codes. Section 3.2 formulates the transmission efficiency based on the analysis of the retransmission mechanisms of the IEEE 802.11 MAC protocol and on the specific characteristics of the modulation scheme employed at the PHY layer. The transmission efficiency formula is derived as a function of the packet length. Our simulation results provided in Section 3.3 demonstrate that the maximum transmission efficiency is indeed achieved when using the optimal packet length. Section 3.4 concludes this chapter and motivates the employment of the channel coding in the following chapters.
- Chapter 4 designs as well as analyzes MCTCs and combines them with HARQ schemes. The area properties of Extrinsic Information Transfer (EXIT) charts reveal that traditional Twin-Component Turbo Codes (TCTCs) suffer from an inherent capacity loss, when the coding rate becomes less than $\frac{1}{2}$. Therefore, in Section 4.1, the system model of N-components MCTCs is designed and we extend classic two-dimensional EXIT charts to analyze them, even though N-component schemes in theory would require N-dimensional EXIT. Section 4.1.4 provides simulation results to demonstrate that MCTCs have a better performance than that of TCTCs. Therefore, they are combined with HARQ schemes in Section 4.2, where we detail the transmission procedure of MCTC aided HARQ schemes and the construction of the MCTC decoder following each transmission. Section 4.2 compares the PLR, throughput and complexity of MCTC HARQ to those of the conventional TCTC HARQ benchmarkers. Section 4.3 summarizes this chapter.
- Chapter 5 focuses on reducing the complexity of MCTC aided HARQ schemes. In this chapter, we propose the novel strategies of Early Stopping (ES) and Deferred Iteration (DI) to eliminate any unnecessary iterations. Section 5.1 describes the

ES strategy that deactivates the turbo decoding process, when the MI increase becomes less than an appropriately chosen threshold; while in Section 5.2 a novel DI strategy is proposed to defer the commencement of iterations, until an open EXIT tunnel emerges in the i-component MCTC decoder, which is constructed for all i received transmissions. Finally, Section 5.3 offers our conclusions for this chapter.

- Chapter 6 sums up the entire thesis chapter by chapter in Section 6.1 and suggests future research in Section 6.2 based on the results provided by this thesis.

1.4 Novel Contributions

The thesis is based on the publications of [22–27]. The novel contributions of this thesis are listed below:

- Novel coding techniques conceived for cross-layer operation are proposed. A high-throughput, yet low-complexity wireless system has been constructed using Fountain codes and turbo codes.
- Optimal packetization is proposed for Fountain codes in the scenario of IEEE 802.11 WLANs [22]. More explicitly, Fountain codes are invoked for protecting file transfer at the application layer. The packet length critically influences the transmission efficiency, which is defined as the ratio between the number of information bits over the total overhead. In the scenario of IEEE 802.11 WLANs, each packet may impose a certain traffic load including the headers appended by each layer, the retransmitted packet replicas and the control packets, such as Request To Send (RTS)/Clear To Send (CTS) and the ARQ packets specified in the 802.11 MAC protocol. Furthermore, retransmissions are sensitive with the packet length, since long packets may suffer from a high PLR, while short packets benefit from a low PLR. Therefore, we derive a formula for the transmission efficiency considering both the MAC layer's retransmissions and the PHY layer's modulation scheme. Based on this formula, the optimal packet length is calculated. Our simulation results demonstrate that the maximum transmission efficiency may indeed be achieved, when employing the optimal packet length derived.
- By exploiting the area properties of EXIT charts, we observed the inherent capacity loss in classic TCTCs and circumvented it by conceiving MCTCs [24]. Moreover, a novel method of partitioning N-component parallel turbo codes into two logical parts was proposed, which allowed us to use classic two-dimensional EXIT charts for MCTCs. Based on these extended EXIT charts, we may find the SNR thresholds required for different coding rates to create an open tunnel, which

allows us to accurately characterize the achievable performance of MCTCs. Furthermore, we demonstrate with the aid of BER curves that MCTCs outperform TCTCs at the same complexity.

- We develop MCTC aided HARQ schemes proposed based on the improved-performance MCTCs [23]. The transmission and reception of each IR packet is detailed and it is shown with the aide of our simulation results that MCTC aided HARQ schemes have a lower PLR and a higher throughput compared to those of the relevant benchmarkers, when the same complexity is considered.
- An ES strategy is proposed to reduce the complexity of turbo coded HARQ schemes [25]. If the MI increment rate becomes less than a pre-defined threshold, the turbo decoding iterations will be terminated. This threshold is obtained by an off-line training process, which adjusts the threshold values to minimize the complexity, on the premise that a small amount of throughput loss is allowed. Our simulation results demonstrate that the complexity may be significantly decreased, while the PLR and throughput is similar to those of non-ES aided turbo coded HARQ schemes.
- A DI strategy is employed to further reduce the complexity of turbo coded HARQ schemes [26, 27]. This DI strategy delays the turbo decoding following each IR transmission, until it predicts that sufficient information has been received for the iterative decoding to succeed with a high probability. This is realized by comparing the currently received MI to a pre-stored threshold MI, which indicates that a marginally open EXIT tunnel emerges, when combining the most recently received MI contribution with all previously received MIs. Our simulation results demonstrate that a considerable complexity reduction may be attained for different turbo coded HARQ schemes, since no iterations are executed at all, until the MI received becomes sufficiently high for successful decoding.
- We conceive a Look-Up Table (LUT) to store all possible threshold MIs for the DI strategy, which requires only a small amount of off-line training and storage [26, 27]. This is achieved by employing a novel semi-analytic design procedure, which avoids time-consuming Monte Carlo simulations. Furthermore, we exploit the gradually evolving nature of the EXIT functions as the channel conditions fluctuate for the sake of minimizing the complexity of the search required for determining the MI threshold, at which an open EXIT tunnel emerges. Finally, we propose a novel method for exploiting the potential redundancy in the LUT, in order to minimize its size.
- Additionally in the DI strategy, we conceive special measures to cater for short packets [26, 27], for which the Monte-Carlo-decoding trajectory might in fact

reach the point of perfect convergence at $(1, 1)$ in the EXIT chart, even when the EXIT tunnel becomes 'just' closed. By contrast, sometimes the $(1, 1)$ point is not reached by the trajectory, even though the EXIT tunnel is open [28]. This inaccuracy is a consequence of failing to generate independent LLRs due to the insufficient packet length.

Chapter 2

Introduction to Coding and EXIT Charts

Since this thesis relies on Cross Layer Design (CLD), we apply Fountain codes for file transfer at the application layer and turbo coded Hybrid Automatic Repeat reQuest (HARQ) schemes at the Media Access Control (MAC) and Physical (PHY) layers, respectively. Firstly, the characteristics of these two code families are described in this chapter. Furthermore, we also introduce two software development platforms, namely IT++ and NS2, since our simulations rely on them.

2.1 Shannon Capacity

Error correction codes constitute one of the most important techniques of approaching capacity, which is achieved by imposing redundancy on the original information bits. In 1948, Shannon's milestone paper [21] laid down the foundations of coding theory, which quantified the performance limits of all existing codes. Therefore, the basic concepts of the related information theory will be introduced first.

A simplified communication system is illustrated in Figure 2.1, where $\mathbf{a}$ is the source information bit sequence and $\mathbf{b}$ is the encoded bit sequence, while $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{b}}$ are their counterparts at the receiver. Additionally, Binary Phase-Shift Keying (BPSK) is used in this system.

The source information conveyed by the bit sequence $\mathbf{a}$ is determined by its entropy. Generally, the entropy is defined as the average self-information of the source [29] given by the following expression:

$$H(\mathbf{X}) = \sum_{k=1}^{K} P(x_k) \cdot \log \frac{1}{P(x_k)} \ , \tag{2.1}$$

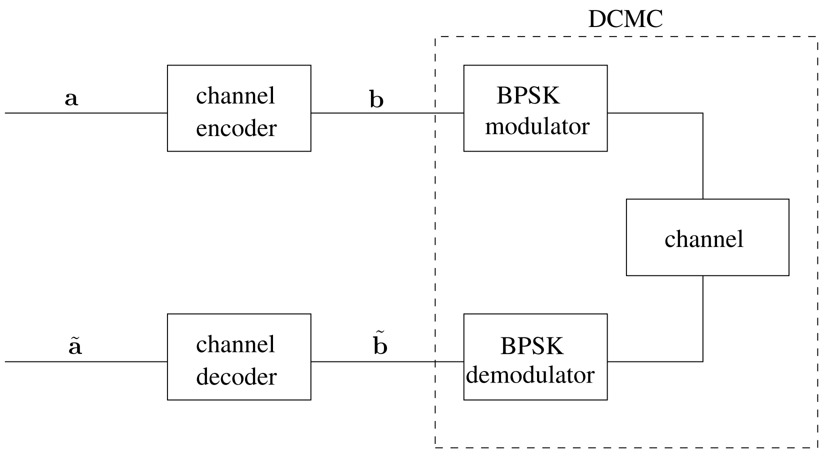


Figure 2.1: A simplified communication system.

where $\log\frac{1}{P(x_k)}$ represents the self information when the source symbol is $\mathbf{X} = x_k$, and its probability is $P(x_k)$. All source symbols belong to the sample space $\{x_1, x_2, \cdots, x_K\}$. In the binary communication system of Figure 2.1, the sample space of the bit sequence **a** only contains $K = 2$ symbols of $\{0, 1\}$, and their probabilities are $P(0) = P(1) = 0.5$. Hence, the source information of **a** becomes 1 according to Equation 2.1. Considering the output of the channel encoder **b**, the same formulas can be applied to compute its information, since the encoded bit sequence **b** has similar statistical properties to those of the input **a**.

Furthermore, Mutual Information (MI) is introduced to measure the information transferred by the channel. Assuming that the sample space of the received random symbols is a discrete set of $\{y_1, y_2, \cdots, y_J\}$, the MI between the transmitted symbol of $X = x_k$ and the received symbol of $Y = y_j$ can be expressed by the following equation:

$$I(x_k; y_j) = \log\frac{P(x_k|y_j)}{P(x_k)} \ , \tag{2.2}$$

where $P(x_k|y_j)$ is the probability that the transmitted source symbol is x_k, conditioned on receiving y_j. The conditional probability $P(x_k|y_j)$ is also referred to as the *a posteriori* probability, while $P(x_k)$ is termed as the *a priori* probability. When jointly considering all possible combinations of the transmitted and received symbols, the average MI may be expressed as:

$$I(\mathbf{X}; \mathbf{Y}) = \sum_{k=1}^{K}\sum_{j=1}^{J} P(x_k, y_j)\log\frac{P(x_k|y_j)}{P(x_k)} \ , \tag{2.3}$$

where both the joint probability $P(x_k, y_j)$ and the *a posteriori* probability $P(x_k|y_j)$ are dependent on the channel transition probability $P(y_j|x_k)$, according to Bayes' theorem.

The transition probability at the output of the channel is naturally different, for example for an Additive White Gaussian Noise (AWGN) channel or for a Rayleigh fading channel, which is given by the probability of the receiver receiving the symbol of y_j when the transmitter transmits the symbol of x_k over that channel. Hence, Equation 2.3 may be transformed to:

$$I(\mathbf{X};\mathbf{Y}) = \sum_{k=1}^{K}\sum_{j=1}^{J} P(x_k)P(y_j|x_k)\log\frac{P(y_j|x_k)}{P(y_j)} \ . \tag{2.4}$$

On the other hand, the average MI is also directly related to the difference between the source entropy before and after transmission, which is interpreted as the reduction of uncertainty concerning the source messages, when the receiver has received all the symbols. The source entropy inferred at the output of the receiver may be quantitated by the conditional entropy, expressed by the following equation:

$$H(\mathbf{X}|\mathbf{Y}) = -\sum_{k=1}^{K}\sum_{j=1}^{J} P(x_k, y_j)\cdot\log P(x_k|y_j) \ . \tag{2.5}$$

Then, the average MI gleaned may be formulated as:

$$I(\mathbf{X};\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}). \tag{2.6}$$

Equation 2.4 reveals that the information transferred by the channel is directly related to the channel's transition probability of $P(y_j|x_k)$ and to the statistical distribution of the channel input given by $P(x_k)$. Therefore, the capacity of a specific channel is defined as the maximum MI found for all possible channel input distributions, which is formulated as follows,

$$C = \max_{P(x_1),\cdots,P(x_K)} \sum_{k=1}^{K}\sum_{j=1}^{J} P(x_k)P(y_j|x_k)\log\frac{P(y_j|x_k)}{P(y_j)} \ . \tag{2.7}$$

To elaborate a little further, the capacity of a communication system, which relies on a specific modulation scheme is more considered as the upper bound of the information that the system can transmit. When considering the example shown in Figure 2.1, the Discrete input Continuous output Memoryless Channel (DCMC) consists of the communication channel and the BPSK modulation scheme, where the soft demodulator generates the continuous output values. Hence, the sum of each probability of y_j in

Equation 2.4 should be replaced by the integral quantifying the DCMC capacity as:

$$I(\mathbf{X};\mathbf{Y}) = \sum_{k=1}^{K} \int_{-\infty}^{\infty} P(x_k)P(y|x_k) \log \frac{P(y|x_k)}{P(y)} dy \ . \tag{2.8}$$

The authors of [30, Chapter 23.2] detailed the DCMC capacity calculation for both the AWGN and the Rayleigh fading channel. Here, we only present Figure 2.2 to quantify the DCMC capacities of both the AWGN and Rayleigh channels for BPSK modulation versus the Signal to Noise Ratio (SNR). Furthermore, Shannon's limit for both AWGN and Rayleigh channels is also displayed in Figure 2.2.

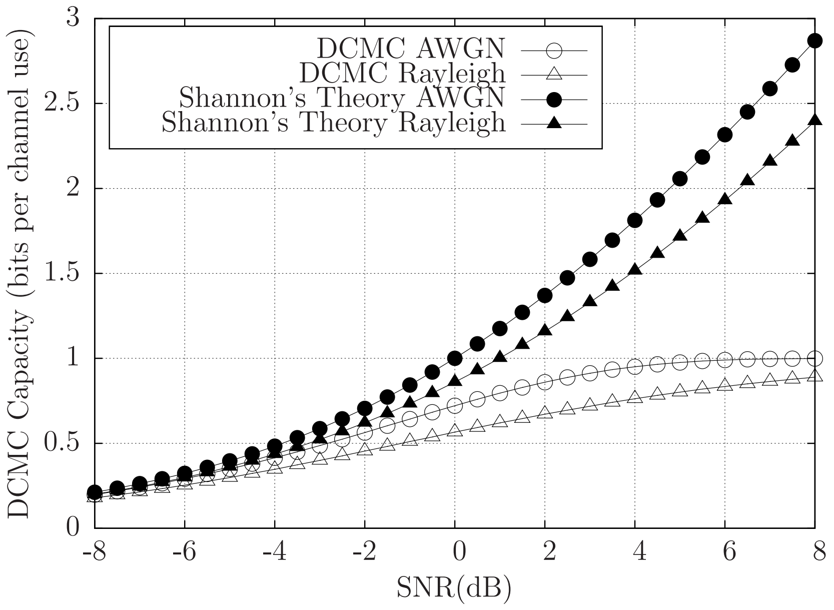


Figure 2.2: The Shannon's limit and the DCMC capacity when the BPSK modulation scheme is used [30].

Shannon's coding theorem has stated that the Bit Error Ratio (BER) can be kept arbitrarily low, as long as the information transmission rate is less than the capacity [31]. The transmission rate may be interpreted as the normalized throughput, which is defined as the ratio of the number of successfully delivered information bits over the total number of transmitted bits. Hence, Figure 2.2 shows the upper bound of the throughput, which researchers endeavor to achieve. Let us now briefly consider turbo codes and Fountain codes.

2.2 Turbo Codes

In 1993, Berrou, Glavieux and Thitimajshima proposed an iterative decoding for a pair of appropriately combined parallel concatenated Recursive Systematic Convolutional (RSC) codes. They termed these codes as turbo codes and characterized their near-capacity performance in [32]. The demodulator's soft-output is typically quantified in terms of the so-called Log Likelihood Ratio (LLR), which is the logarithmic ratio of the probability that the symbol is equal to '+1' over the probability that the symbol is equal to '-1', given that BPSK modulation is used. Therefore, the polarity of an LLR indicates the correspondingly decoded bit as '0' or '1', while its magnitude quantifies our confidence in taking that value. In turbo codes, two RSC decoders iteratively exchange their soft information. More explicitly, one of the RSC decoders will output its so-called *extrinsic* LLRs, which are input into the other RSC decoder as the *a priori* LLRs. The iterations continue, as long as each RSC decoder can provide an increasing *extrinsic* information, until a specific stopping criterion is satisfied, which may be a successful Cyclic Redundancy Check (CRC) or a pre-defined number of iterations.

A large amount of research has been carried out since turbo codes were invented [33–44]. These works considered their complexity, concatenation mode and interleaver design, etc. For example, the authors of [33] proposed a new Maximum *a posteriori* (MAP) algorithm for reducing the computational complexity of high-rate convolutional codes, which may become the component codes of high-rate turbo codes. The approach suggested by the authors of [34] showed a significant complexity reduction as a benefit of using a generalized stopping criterion for the iterative decoders. Besides the original parallel concatenation of RSC codes, [35, 36] as well as a large number of other papers studied the serial concatenation mode of turbo codes. The design of the interleaver between the components also catches researchers' eye [37], especially for short turbo codes [38], as the performance of turbo codes decreases when the interleaver length becomes shorter. Apart from improving the turbo codes themselves, the combination of the turbo concept with other schemes generated numerous new research topics, such as turbo detection [39,40], turbo equalization [41,42] and turbo coded modulation [43,44] etc. Furthermore, thousands of papers have been published about the applications of turbo codes in different scenarios. We may conclude that turbo codes have revolutionized the field of communication.

In this thesis, Multiple Components Turbo Codes (MCTCs) are studied and applied to Hybrid Automatic Repeat reQuest (HARQ) schemes. Hence, we first introduce classic Twin Components Turbo Codes (TCTCs) to augment the related concepts and principles in the following sections, noting that detailed tutorials can be found in [30].

2.2.1 Turbo Encoder

The component RSC encoders used in turbo codes may employ generator polynomials of arbitrary memory. However, the RSC encoders, which have feedforward and feedback polynomials of $(2, 3)_o$ expressed in an octal representation are preferable, since the shortest memory-1 length essentially has the lowest possible complexity. Furthermore, MCTCs using polynomials of $(2, 3)_o$ will be shown to have the best performance in

Chapter 4, compared to MCTCs using other polynomials and to TCTCs with $(2,3)_o$. Therefore, the structure of the RSC encoder having the generator polynomials of $(2,3)_o$ is illustrated in Figure 2.3.

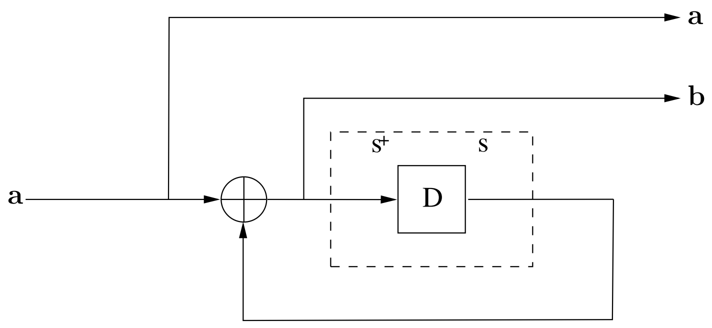


Figure 2.3: The RSC encoder having the generator polynomials of $(2,3)_o$.

In Figure 2.3, the input bits **a** are shifted into the RSC encoder in order to obtain the encoded bit sequence **b** and the systematic bits **a**. The shift register associated with a 1-bit memory has two states: 0 and 1. The states of the shift register vary between these two states. We let s denote the current state and s^+ denote the next state after one clock-cycle delay. The next state s^+ will depend on the current input bit a and the current state s. The following equations reveal the output expressions and the state transitions:

$$\begin{aligned} s^+ &= a \oplus s, \\ b &= s^+, \\ a &= a. \end{aligned} \tag{2.9}$$

Based on Equation 2.9, Table 2.1 further illustrates the relationship between the input plus the current state and the output plus the next state.

Table 2.1: The relationship between the state transition and the current input/output.

a	s	a	b	s^+
0	0	0	0	0
0	1	0	1	1
1	0	1	1	1
1	1	1	0	0

Table 2.1 may also be visualized by the state transition diagram of Figure 2.4. Each bit of the input sequence **a** in turn goes through the above state transition diagram, resulting in a trellis diagram. This trellis diagram aids us in understanding the encoding and decoding process. For instance, assuming that the input sequence is $\mathbf{a} = [1\ 1\ 0\ 0\ 1]$, the corresponding trellis diagram is shown in Figure 2.5, where the

bold lines illustrate the encoding paths. Following these paths, the encoded output bit sequence may be obtained as $\mathbf{b} = [1\ 0\ 0\ 0\ 1]$.

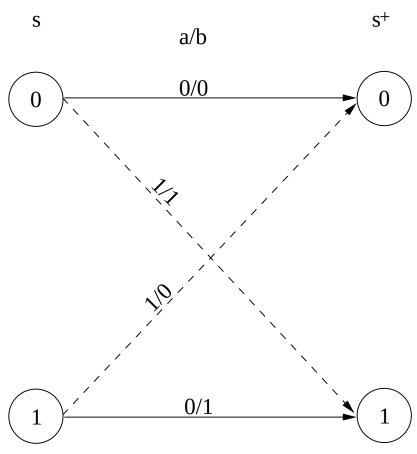


Figure 2.4: The state transition diagram of RSC having the polynomials of $(2, 3)_o$.

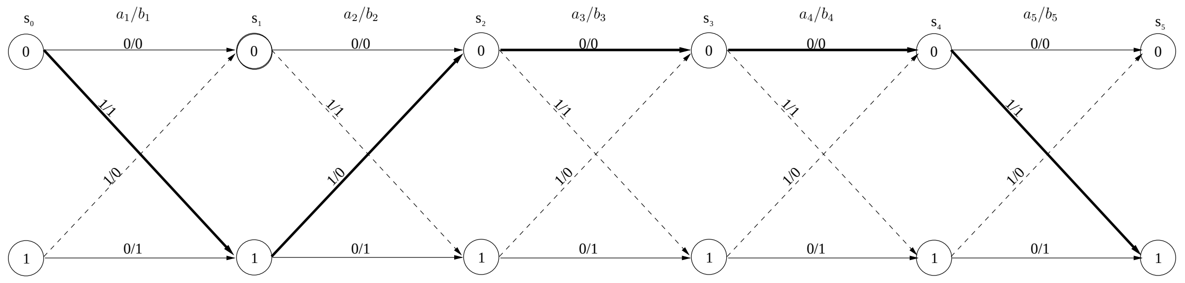


Figure 2.5: The trellis diagram for encoding the input sequence of $\mathbf{a} = [1\ 1\ 0\ 0\ 1]$.

Connecting two of the RSC encoders seen in Figure 2.3 in parallel, the resultant architecture of turbo codes is shown in Figure 2.6. The encoding process of turbo codes is as follows. The original bit sequence $\mathbf{a_1}$ and its interleaved replica $\mathbf{a_2}$ respectively are entered into two RSC encoders. The outputs of these two encoders are punctured and multiplexed to form the encoded bit sequence $\mathbf{b}$. The information bit sequence $\mathbf{a}$ is then combined with $\mathbf{b}$ to form the systematic turbo code's output sequence. These systematic turbo codes using two constituent RSC encoders automatically result in a coding rate of 1/3, when no puncturing is performed. If the coding rate has to be higher, e.g. 1/2, the systematic bits are generally retained and the parity bit sequence $\mathbf{b}$ may be generated by puncturing the odd bits from $\mathbf{b_1}$ and the even bits from $\mathbf{b_2}$. On the other hand, the coding rate may be further decreased below 1/3 by incorporating the repeated replicas of $\mathbf{a}, \mathbf{b_1}, \mathbf{b_2}$ in the output. However, we will demonstrate in Chapter 4 that our MCTCs perform best upon combining new parity bit sequences generated by multiple rate-1 encoders.

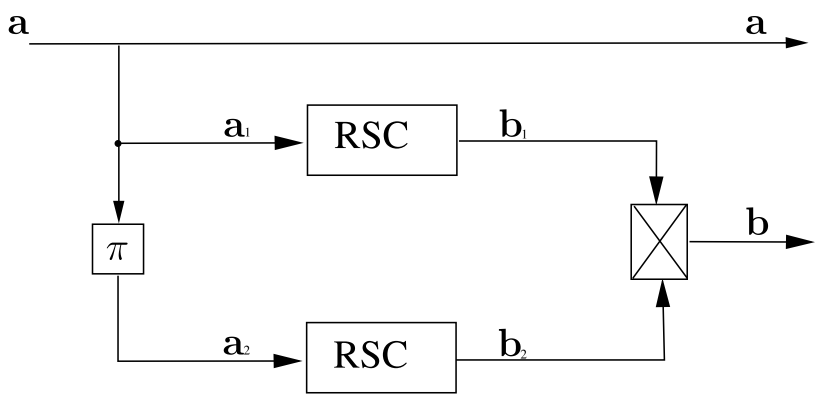


Figure 2.6: The structure of the turbo encoder using two RSC components.

2.2.2 Turbo Decoder

The performance of turbo codes benefits from iterative decoding operations exchanging extrinsic information between two components. The detailed structure of a systematic turbo decoder is shown in Figure 2.7, where the decoding operations are performed by two so-called Bahl, Cocke, Jelinek and Raviv (BCJR) decoders [45]. The information exchanged between these two BCJR decoders are all LLR values. Specifically, as observed in Figure 2.7, $\tilde{\mathbf{a}}$ represents the systematic LLRs corresponding to the received information bits; $\tilde{\mathbf{b}}_1$ and $\tilde{\mathbf{b}}_2$ are the parity LLRs generated by the de-puncturer from $\tilde{\mathbf{b}}$, which are then entered into two BCJR decoders; the extrinsic LLRs $\tilde{\mathbf{a}}_i^e$ output from one of the BCJR decoders is passed to the other one as the *a priori* LLRs, which are denoted by $\tilde{\mathbf{a}}_i^a$. Furthermore, the systematic LLRs $\tilde{\mathbf{a}}$ have to be incorporated into the *a priori* information. In Figure 2.7, $\tilde{\mathbf{a}}^p$ represents the *a posteriori* LLRs, which can be obtained by combining the extrinsic LLRs and the *a priori* LLRs generated from any of the BCJR decoders.

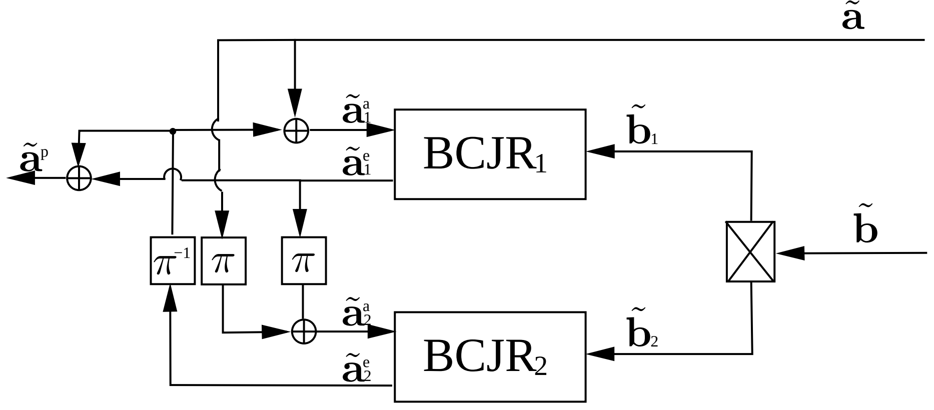


Figure 2.7: The structure of the turbo decoder.

2.2.2.1 The BCJR Algorithm

Bahl, Cocke, Jelinek and Raviv proposed a MAP algorithm for estimating the original information bits. Generally, the *a posteriori* probability of an information bit conditioned on the entire observed symbol sequence $\bar{\mathbf{b}}$ may be expressed as $P(a_k|\bar{\mathbf{b}})$, where $\bar{\mathbf{b}}$ is associated with but different from the parity LLR sequence $\tilde{\mathbf{b}}$. Since we assume that BPSK modulation is employed, the *a posteriori* LLR for that bit is then calculated as

$$\tilde{a}_k^p = L(a_k|\bar{\mathbf{b}}) = \ln\left(\frac{P(a_k=+1|\bar{\mathbf{b}})}{P(a_k=-1|\bar{\mathbf{b}})}\right) . \tag{2.10}$$

When k traverses through the whole sequence, the vector of *a posteriori* LLRs $\tilde{\mathbf{a}}^p$ can be obtained. In addition, a single parity LLR $\tilde{b}_k$ may be expressed by the channel transition probability as follows:

$$\tilde{b}_k = L(\bar{b}_k|b_k) = \ln\left(\frac{P(\bar{b}_k=+1|b_k)}{P(\bar{b}_k=-1|b_k)}\right) , \tag{2.11}$$

whose value may be provided by the soft demodulator's output related to the received symbol $\bar{b}_k$. Furthermore, a single *a priori* LLR $\tilde{a}_k^a$ may be expressed by the *a priori* probability as follows,

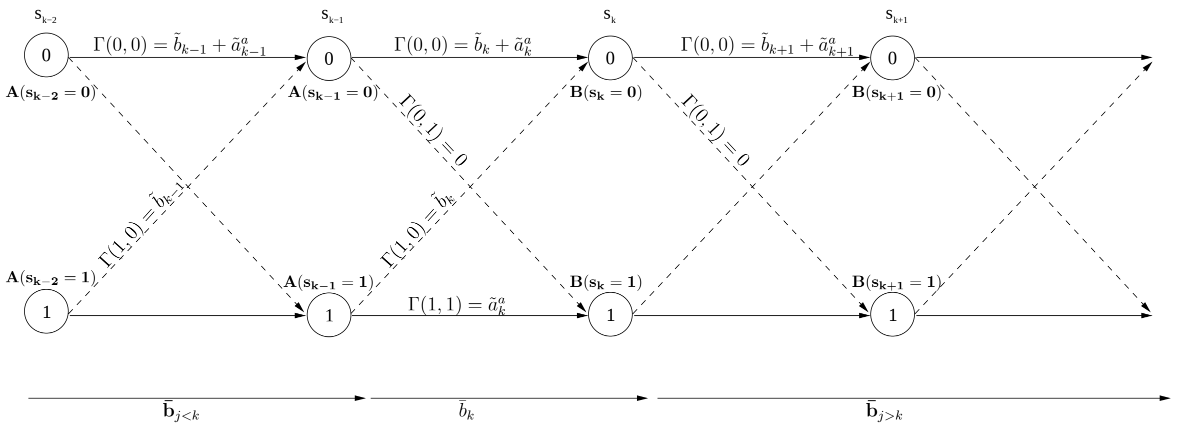
$$\tilde{a}_k^a = L(a_k) = \ln\left(\frac{P(a_k=+1)}{P(a_k=-1)}\right) . \tag{2.12}$$

According to [46], three intermediate variables α, β and γ may be introduced to evaluate Equation 2.10, leading to:

$$L(a_k|\bar{\mathbf{b}}) = \ln\left(\frac{\sum_{(s_{k-1}\Rightarrow s_k)_1}\alpha(s_{k-1})\cdot\gamma(s_{k-1},s_k)\cdot\beta(s_k)}{\sum_{(s_{k-1}\Rightarrow s_k)_0}\alpha(s_{k-1})\cdot\gamma(s_{k-1},s_k)\cdot\beta(s_k)}\right) . \tag{2.13}$$

In order to explain these three variables, in physically tangible terms, we resort to Figure 2.8 which illustrates a part of the trellis at the instants of $(k-1), k, (k+1)$, where the solid lines indicate the state transitions triggered by the input of '0' and the dashed lines indicate those triggered by the input of '1'. In Equation 2.13, $(s_{k-1}\Rightarrow s_k)_1$ indicates the set of transitions triggered by the input of 1, and similarly $(s_{k-1}\Rightarrow s_k)_0$ denotes the set of transitions triggered by the input of 0.

During the decoding process, the trellis may stay in the state of 0 or 1 with certain probabilities at the instant $(k-1)$, when the received symbol sequence is $\bar{\mathbf{b}}_{j<k}$ before the instant k. The variable $\alpha(s_{k-1}), s_{k-1}\in\{0,1\}$ is employed to denote this probability. By contrast, $\gamma(s_{k-1},s_k)$ represents the transition probability from the state s_{k-1} to the state s_k, given that the received symbol is $\bar{b}_k$ at the instant k. Finally, $\beta(s_k)$ denotes



s_{k-2}
s_{k-1}
s_k
s_{k+1}
$\Gamma(0,0)=\tilde{b}_{k-1}+\tilde{a}^a_{k-1}$
$\Gamma(0,0)=\tilde{b}_k+\tilde{a}^a_k$
$\Gamma(0,0)=\tilde{b}_{k+1}+\tilde{a}^a_{k+1}$
$\Gamma(0,1)=0$
$\Gamma(1,0)=\tilde{b}_{k-1}$
$\Gamma(1,0)=\tilde{b}_k$
$\Gamma(1,1)=\tilde{a}^a_k$
$\mathbf{A}(\mathbf{s_{k-2}}=0)$
$\mathbf{A}(\mathbf{s_{k-2}}=1)$
$\mathbf{A}(\mathbf{s_{k-1}}=0)$
$\mathbf{A}(\mathbf{s_{k-1}}=1)$
$\mathbf{B}(\mathbf{s_k}=0)$
$\mathbf{B}(\mathbf{s_k}=1)$
$\mathbf{B}(\mathbf{s_{k+1}}=0)$
$\mathbf{B}(\mathbf{s_{k+1}}=1)$
$\bar{\mathbf{b}}_{j<k}$
$\bar{b}_k$
$\bar{\mathbf{b}}_{j>k}$

Figure 2.8: The part of trellis transition diagram of RSC at the instants of $k-1, k, k+1$.

the probability that the trellis is in the state of s_k at the instant k, if the later received symbol sequence is $\bar{\mathbf{b}}_{j>k}$.

In this thesis, we leave out the detailed derivations which may be found in [46, 47]. The calculation of $\alpha(s_{k-1})$ and $\beta(s_k)$ is related to the function γ in a recursive manner. Their expressions are displayed as follows:

$$\alpha(s_{k-1}) = \sum_{all\ s_{k-2}} \alpha(s_{k-2}) \cdot \gamma(s_{k-2}, s_{k-1}),$$
$$\beta(s_k) = \sum_{all\ s_{k+1}} \beta(s_{k+1}) \cdot \gamma(s_k, s_{k+1}), \tag{2.14}$$

where '$all\ s_{k-2}$' in the first sub-equation denotes all possible states at the instant $(k-2)$ which may merge into the present state s_{k-1}, while '$all\ s_{k+1}$' in the second sub-equation means all possible states that the present state s_k may be traverse to. Then, the recursive calculation of $\alpha(s_{k-1})$ will proceed to the initial state of s_0, and the initial values are assumed to be $\alpha(s_0 = 0) = 1$ and $\alpha(s_0 = 1) = 0$. Similarly, the values of $\beta(s_k)$ are recursively calculated backwards to the last state s_n, where n is the length of the received sequence. The commonly used assumption is that $\beta(s_n) = 1$ for all final states, which is a reasonable assumption, as demonstrated in [47, Section 4.3.3.3].

At this point, all expressions hinge on the calculation of the 'γ' values between any two states, which obeys the following expression [47, Section 4.3.3.4]:

$$\gamma(s_{k-1}, s_k) = P(\bar{b}_k|b_k) \cdot P(a_k), \tag{2.15}$$

where $P(\bar{b}_k|b_k)$ is the probability of the received symbol $\bar{b}_k$, conditioned on the transmitted bit b_k, and $P(a_k)$ is the *a priori* probability of a_k.

The *a posteriori* LLR in Equation 2.13 may be calculated with the aid of Equations

2.14 and 2.15. Although the resultant MAP algorithm is widely used, its computational complexity becomes significant due to its recursive calculation. In order to further reduce its complexity, researchers transformed the above multiplications to simpler additions in the logarithmic domain [47, Section 4.3.5]. The logarithmic forms of α, β and γ are correspondingly denoted by A, B and Γ, which may finally be represented by the soft LLRs of $\tilde{b}_k$ and $\tilde{a}_k^a$, according to Equations 2.11 and 2.12. This process is also referred to as the Log-BCJR algorithm, taking the soft LLR values as the input parameters.

As seen in Figure 2.7, the BCJR decoder accepts two inputs: the parity LLRs $\tilde{\mathbf{b}}$ as well as the *a priori* LLRs $\tilde{\mathbf{a}}^{\mathbf{a}}$, and then outputs the extrinsic LLRs $\tilde{\mathbf{a}}^{\mathbf{e}}$, which are calculated as the *a posteriori* LLRs minus the present *a priori* input $\tilde{\mathbf{a}}^{\mathbf{a}}$. Then, the BCJR decoder of Figure 2.7 may be interpreted as a function 'F_{bcjr}', which outputs the dependent variable $\tilde{\mathbf{a}^e}$ generated from the independent variables: $\tilde{\mathbf{a}}^{\mathbf{a}}$ and $\tilde{\mathbf{b}}$, where 'F_{bcjr}' may be expressed as:

$$\tilde{\mathbf{a}^e} = F_{bcjr}\left[\tilde{\mathbf{a}^a}, \tilde{\mathbf{b}}\right] \ . \tag{2.16}$$

As a result, the BCJR function of 'F_{bcjr}' represented by Equation 2.16 obeys the following steps:

- 1. *Calculate $\Gamma(s_{k-1}, s_k)$ for all possible trellis transitions emerging from state 's_{k-1}' and merging into state 's_k' of Figure 2.8.* Since 's_{k-1}' and 's_k' belong to the set of $\{0, 1\}$, all transitions between (s_{k-1}, s_k) have four possible combinations, as follows,
  - $\Gamma(0,0) = \tilde{b}_k + \tilde{a}_k^a$;
  - $\Gamma(0,1) = 0$;
  - $\Gamma(1,0) = \tilde{b}_k$;
  - $\Gamma(1,1) = \tilde{a}_k^a$;

  which are derived from Equation 2.15 by applying logarithmic-domain computations.
- 2. *Calculate $A(s_{k-1})$ defined in Figure 2.8 using forward recursion.* As mentioned above, $A(s_{k-1})$ represents the logarithmic form of $\alpha(s_{k-1})$ in Equation 2.14. Assuming that we have $s_{k-1} = 0$, the value of $A(s_{k-1} = 0)$ is related to all possible states at the previous instant of $(k-2)$, i.e. to $A(s_{k-2} = 0)$ and $A(s_{k-2} = 1)$, as seen in Figure 2.8. Furthermore, $A(s_{k-1} = 0)$ is also related to the corresponding trellis transitions from all of these s_{k-2} states to the state of $s_{k-1} = 0$, i.e. to $\Gamma(s_{k-2} = 0, s_{k-1} = 0)$ and $\Gamma(s_{k-2} = 1, s_{k-1} = 0)$. Explicitly, we have:
  - $A_1 = A(s_{k-2} = 0) + \Gamma(s_{k-2} = 0, s_{k-1} = 0)$

$= A(s_{k-2} = 0) + \tilde{b}_{k-1} + \tilde{a}^a_{k-1}$;

– $A_2 = A(s_{k-2} = 1) + \Gamma(s_{k-2} = 1, s_{k-1} = 0)$

$= A(s_{k-2} = 0) + \tilde{b}_{k-1}$;

– $A(s_{k-1} = 0) = Jac(A_1, A_2) = max(A_1, A_2) + \ln\left(1 + e^{-|A_1 - A_2|}\right)$;

Likewise, $A(s_{k-1} = 1)$ may be obtained in the same way.

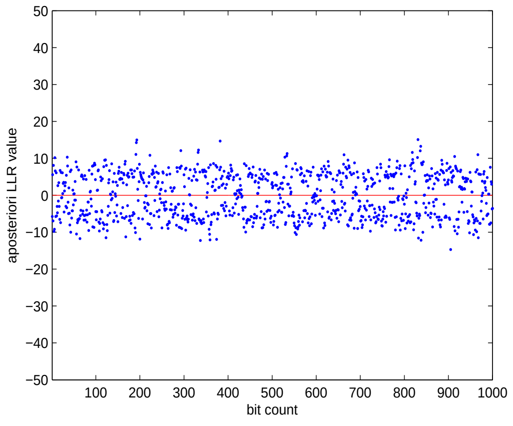
- 3. *Calculate $B(s_k)$ defined in Figure 2.8 using backward recursion.* This $B(s_k)$ represents the logarithmic version of $\beta(s_k)$ in Equation 2.14. Again, assuming that we have $s_k = 0$, $B(s_k = 0)$ is related to all possible states at the next instant of $(k+1)$, i.e. to $B(s_{k+1} = 0)$ and $B(s_{k+1} = 1)$, as seen in Figure 2.8. Furthermore, $B(s_k = 0)$ is also related to the corresponding trellis transitions emerging from the current state of $s_k = 0$ and leading to all of these s_{k+1} states, i.e. to $\Gamma(s_k = 0, s_{k+1} = 0)$ and $\Gamma(s_k = 0, s_{k+1} = 1)$. Hence, we have:
  - $B_1 = \Gamma(s_k = 0, s_{k+1} = 0) + B(s_{k+1} = 0)$
    $= \tilde{b}_{k+1} + \tilde{a}^a_{k+1} + B(s_{k+1} = 0)$;
  - $B_2 = \Gamma(s_k = 0, s_{k+1} = 1) + B(s_{k+1} = 1)$
    $= B(s_{k+1} = 1)$;
  - $B(s_k = 0) = Jac(B_1, B_2) = max(B_1, B_2) + \ln\left(1 + e^{-|B_1 - B_2|}\right)$;

  Likewise, $B(s_k = 1)$ may be obtained in the same way.
- 4. *The value of $\tilde{a}^p_k$ may be computed using the following steps:*
  - $\delta(s_{k-1}, s_k) = A(s_{k-1}) + \Gamma(s_{k-1}, s_k) + B(s_k)$;
  - $\delta_0 = Jac\left(\delta(s_{k-1}, s_k)_{all\ (s_{k-1} \Rightarrow s_k)_0}\right)$;
  - $\delta_1 = Jac\left(\delta(s_{k-1}, s_k)_{all\ (s_{k-1} \Rightarrow s_k)_1}\right)$;
  - $\tilde{a}^p_k = \delta_0 - \delta_1$

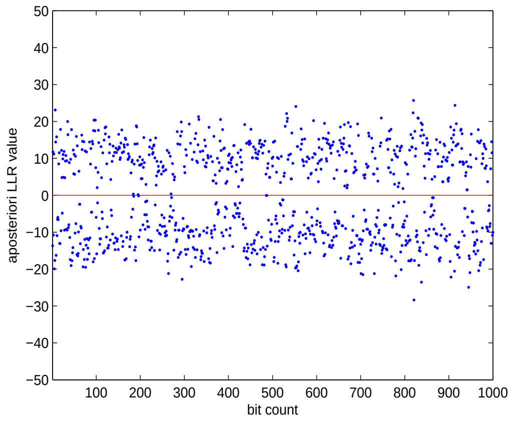
  where the Jac function operates similarly to those in Steps 3 and 4.
- 5. *Finally, the extrinsic LLR $\tilde{a}^e_k$ may be obtained by $\tilde{a}^e_k = \tilde{a}^p_k - \tilde{a}^a_k$.*

2.2.2.2 Iterative decoding

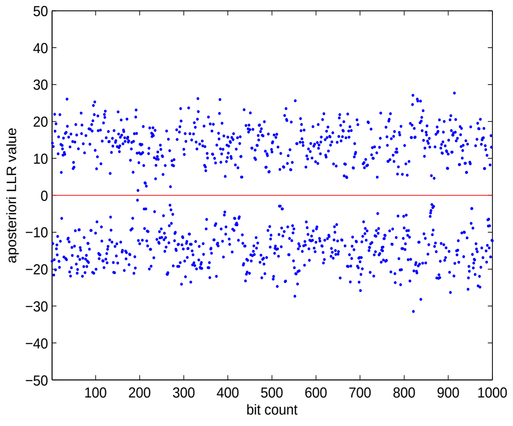
Turbo codes adopt iterative decoding for achieving near capacity performance [47]. During the iterative decoding process, two component decoders are activated in turn and exchange their extrinsic information. Firstly, BCJR1 decoder of Figure 2.7 processes the parity LLRs $\tilde{\mathbf{b}}_1$ as its intrinsic input and outputs the extrinsic LLRs $\tilde{\mathbf{a}}^e_1$. Initially, the systematic LLRs $\tilde{\mathbf{a}}$ act as the sole *a priori* information input of the BCJR1 decoder, since no extrinsic LLRs are available from the BCJR2 decoder. Then, the extrinsic LLRs of Figure 2.7 $\tilde{\mathbf{a}}^e_1$ output by the BCJR1 decoder are conveyed to the BCJR2 decoder, where they will be added to the systematic LLRs $\tilde{\mathbf{a}}$ to generate the *a priori* LLRs $\tilde{\mathbf{a}}^a_2$. The BCJR2 decoder employs the corresponding parity LLRs $\tilde{\mathbf{b}}_2$ and $\tilde{\mathbf{a}}^a_2$ to obtain an increased extrinsic $\tilde{\mathbf{a}}^e_2$, which are conversely fed back to the BCJR1 decoder of Figure 2.7. This iterative decoding process continues until a pre-defined



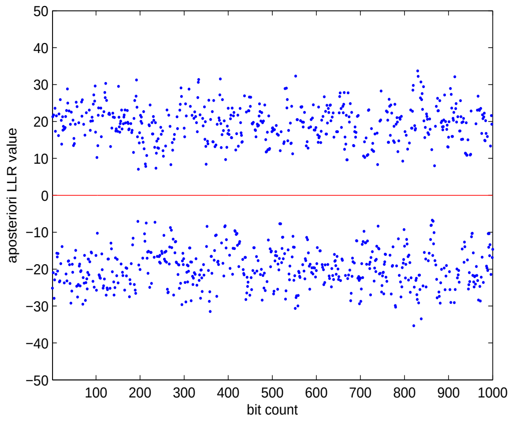
(a) 1st iteration



(b) 2nd iteration



(c) 3rd iteration



(d) 5th iteration

Figure 2.9: The evolution of the *a posteriori* LLRs of a BCJR decoder, when a 1000-bit packet is transmitted over an AWGN channel at the SNR of -2.0dB.

number of affordable BCJR operations is reached or some specific stopping strategy is satisfied.

Provided that a packet having a length of 1000 bits is transmitted at the SNR of -2.0dB over an AWGN channel using BPSK modulation, Figure 2.9 illustrates the evolution of the *a posteriori* LLR values of the BCJR1 decoder, where the *a posteriori* LLR values were recorded at the first, second, third and fifth iterations, respectively. As mentioned above, the magnitude of the LLR values indicate the associated decoding confidence. As seen in Figure 2.9, the *a posteriori* LLR values are increased with each iteration. This demonstrates that the number of bit errors may be significantly decreased with the aid of turbo decoding. Figure 2.10 shows the corresponding Bit Error Ratio (BER) curves, which were recorded after the first, third and fifth iterations, when a sufficiently high number of packets with a packet length of 1000 bits were transmitted.

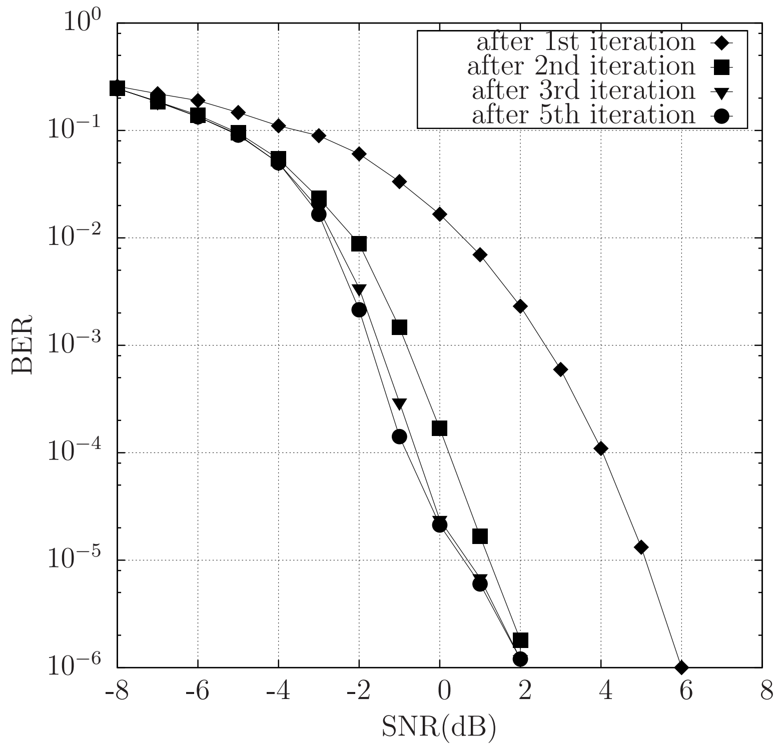
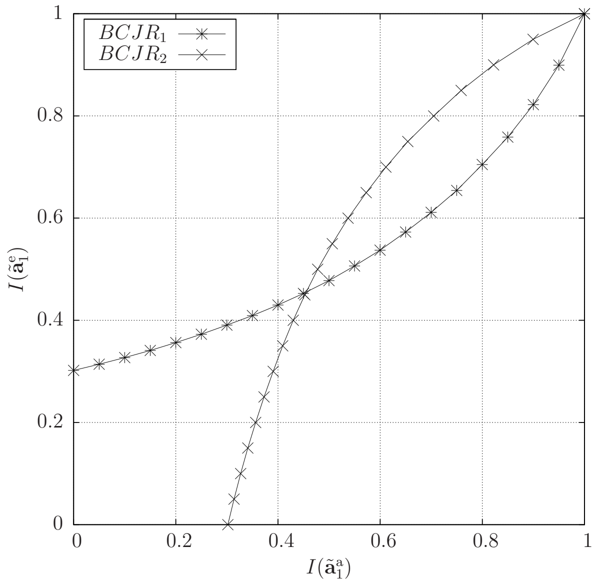


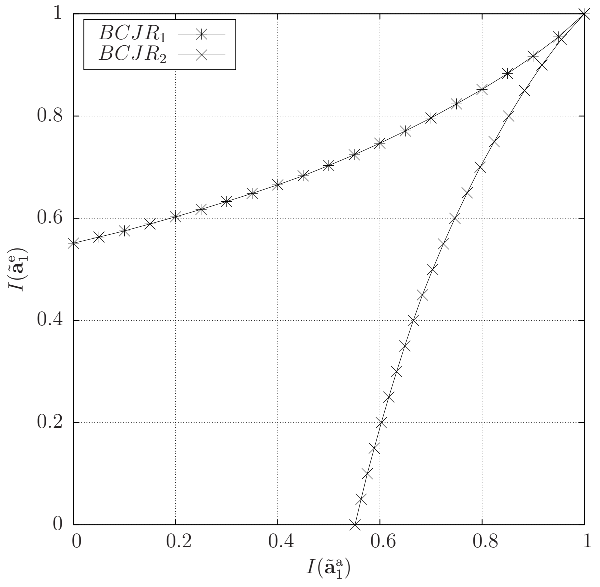
Figure 2.10: BER vs SNR when statistically relevant number of packets are transmitted over an AWGN channel, each having a 1000-bit length.

2.2.3 A Semi-Analytical Tool: EXIT charts

The performance of turbo codes has been classically evaluated in terms of their BER versus SNR characteristics. The BER curves of turbo codes may be divided into three regions: the high-BER region at low SNRs; the rapid BER reduction region at medium SNRs, which may also be referred to as the 'turbo-cliff' region; and the lowest-BER region at high SNRs, which is also referred to as the error floor region. These three regions correspond to different decoding convergence situations of turbo codes. Stephan ten Brink [48] introduced the concept of Extrinsic Information Transfer (EXIT) charts for analyzing the convergence behavior of iterative decoding. Since then, EXIT charts became widely used as an effective analytical tool for designing turbo codes and other iterative detection techniques. For example, based on EXIT charts, the authors of [49] proposed an optimization criterion for the design of general serially concatenated systems, while EXIT charts are also widely employed for aiding the design of turbo coded modulation [49–51]. However, conventional EXIT charts are sometimes inappropriate for specific designs. Therefore, the authors of [52] extended them to 3D forms to analyze periodically punctured turbo codes. In Chapter 4 of this thesis, we will transform N-dimensional EXIT charts to 2-dimensional ones, which then become more readily interpreted.



(a) -4.0dB



(b) -2.0dB

Figure 2.11: Two EXIT charts of turbo codes having two parallel concatenated BCJR decoders recorded at the SNRs of -4.0dB and -2.0dB.

A classic EXIT chart describes the extrinsic Mutual Information (MI)[1] exchange between two parallel concatenated BCJR decoders. The extrinsic MI provided by a BCJR decoder is denoted by $I(\tilde{\mathbf{a}}_i^e)$, which may be approximately calculated from the extrinsic LLRs $\tilde{\mathbf{a}}_i^e$ using the following equation introduced in [53]:

$$I(\tilde{\mathbf{x}}) \approx 1 - \frac{1}{L}\sum_{j=1}^{L} H_{\mathrm{b}}\left(\frac{e^{+|\tilde{x}_j|/2}}{e^{+|\tilde{x}_j|/2} + e^{-|\tilde{x}_j|/2}}\right) , \tag{2.17}$$

where H_{b} represents the binary entropy function, $\tilde{\mathbf{x}}$ is a general notation for a vector of LLRs, and L denotes the length of the vector $\tilde{\mathbf{x}}$. There are two curves in a classic EXIT chart, each reflecting a BCJR decoder's EXIT function F_{exit}, which may be expressed as:

$$I(\tilde{\mathbf{a}}_i^e) = F_{exit}\left(I(\tilde{\mathbf{a}}_i^a)\right) . \tag{2.18}$$

where the independent variable $I(\tilde{\mathbf{a}}_i^a)$ denotes the *a priori* MI input of the $BCJR_i$ decoder, which may also be calculated by replacing $\tilde{\mathbf{x}}$ with $\tilde{\mathbf{a}}_i^a$ in Equation 2.17. Since the output extrinsic MI $I(\tilde{\mathbf{a}}_1^e)$ of the $BCJR_1$ decoder will be passed to the $BCJR_2$ decoder as the *a priori* MI input during the iterative decoding process, classic EXIT charts use the horizontal axis for representing the independent variable of the $BCJR_1$ decoder's EXIT function, while the vertical axis represents that of the $BCJR_2$ decoder's. More explicitly, the independent variable of the $BCJR_1$ decoder's EXIT function - namely $I(\tilde{\mathbf{a}}_1^a)$ - is represented along the X-axis of EXIT charts, while the independent variable of the $BCJR_2$ decoder's - namely $I(\tilde{\mathbf{a}}_2^a)$ - is scaled along the Y-axis of EXIT charts. Hence, the Monte-Carlo simulation based decoding trajectory along the tunnel between two EXIT curves shows the MI exchange process and demonstrates whether the decoding trajectory reaches the point $(1, 1)$ of perfect convergence to an infinitesimally low BER, indicating whether potentially successful decoding can or cannot be achieved. Figure 2.11 exemplifies two EXIT charts recorded for turbo codes, whose structure was shown in Figure 2.7. The SNR was -4.0dB in Figure 2.11(a) and -2.0dB in Figure 2.11(b) for transmission over an AWGN channel, where we observe a closed tunnel and an open one, respectively.

In order to exploit the relationship between EXIT charts and the BER, we consider another example, namely that of non-systematic turbo codes having the octal generator polynomials of $(8, F)_o$ and transmit a sufficiently long packet over an uncorrelated

[1]When referring to MI, it is typically meant to be between a soft-bit value and its hard-decision-based binary representation, unless otherwise stated.

Rayleigh fading channel. The arrows in Figure 2.12 show the relationship between the BER and EXIT charts at the SNR of 0dB.

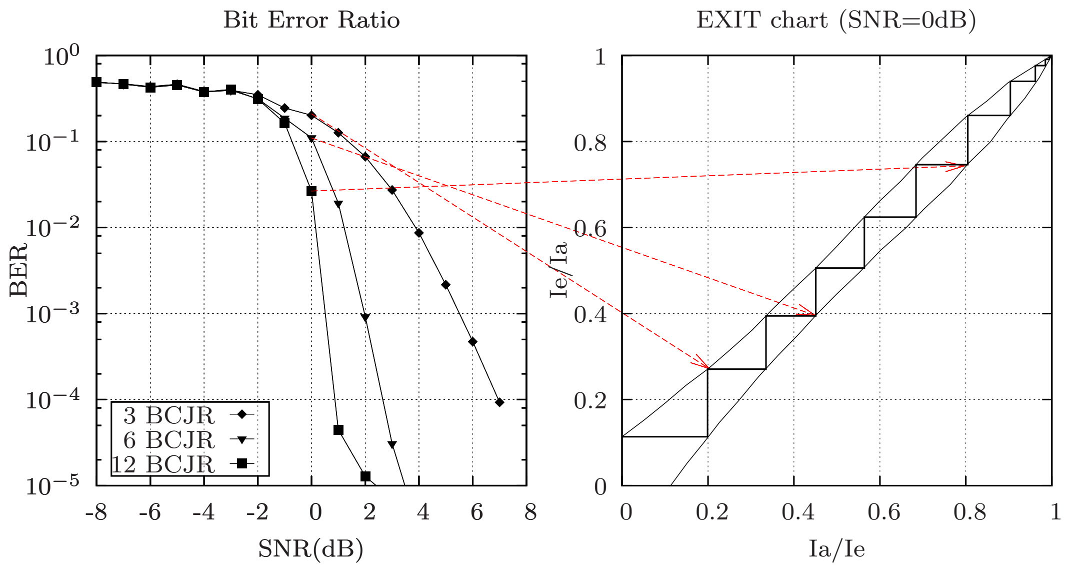


Figure 2.12: The relationship between EXIT charts and the BER.

2.3 Fountain Codes

The term of ‘coding rate’ in coding theory is defined as the ratio of the number of information bits K over the number of encoded bits N, namely as $R = K/N$. All traditional codes have had a coding rate limit until Fountain codes [54–56] were invented, where the terminology of ‘fountain’ is a metaphor of indicating that the encoder of this kind of codes is capable of potentially producing an endless supply of encoded bits, like a fountain spraying drops of water. This implies that the number of encoded bits N may theoretically reach infinity, which justifies the terminology of ‘rateless’ codes. Fountain codes have this remarkable rateless property.

Fountain codes were originally designed for erasure channels, where the packets are deemed to be either correctly received or lost. Generally, transmissions between two peers in the application layer can be treated as packets passing through a virtual erasure channel, since the protocols of lower layers guarantee that the received packets are received correctly and they discard the packets having errors. Therefore, Fountain codes have mainly been employed for protecting packet transmissions at the application layer [57–59]. The authors of [58] utilized Fountain codes in cognitive radio networks for protecting multimedia transmissions of the secondary user. Specifically, Fountain codes were employed to distribute the multimedia contents to different sub-channels which are temporarily accessed by the secondary user. However, standard Fountain codes cannot meet the Unequal Error Protection (UEP) requirements of video coding.

Hence, the authors of [59] introduced the class of so-called 'UEP expanding window Fountain codes' as a solution for real-time scalable video multicast.

Fountain codes have another beneficial property, namely that the receiver can pick any set out of a total of N transmitted encoded packets to decode the original K packets. When the number of packets picked is slightly higher than K, which is expressed as $K \cdot (1 + \epsilon)$ where $\epsilon > 0$ is a small real-value number, the original K packets may be recovered with a high probability. This property encourages the employment of Fountain codes in multicast scenarios, in order to avoid a potentially high number of retransmission requests from the different receivers. For example, the authors of [60] designed Fountain codes for multicast transmission. Furthermore, Fountain codes can also provide protection for broadcast systems, where no feedback channels exist. Hence, the authors of [61] also employed digital Fountain coding as Forward Error Correction (FEC) at the application layer for file transfer services over broadcast channels.

2.3.1 Basic Coding Process of Fountain Codes

Random linear Fountain codes, Luby transform codes and Raptor codes constitute popular types of Fountain codes, which can generate an arbitrary number of encoded packets. Before highlighting the basic encoding procedure of Fountain codes, some definitions are listed below.

- Source packet: Created from the appropriate partitioning of a source file or a stream, which consists of a number of bytes. During encoding, exclusive-OR additions are used for combining several randomly selected source packets.
- Source block: A set of K source packets. A source block is the basic encoding and decoding unit forwarded to the Fountain decoder for error correction.
- FEC payload identifier (ID): Generally, it contains the source block index and the Fountain encoded packet ID, which are combined and forwarded as a seed for the decoder to produce the same pseudo-random numbers as in the encoder, which define the specific bits and packets that are combined using the exclusive-OR function.
- Encoded packet: Each packet output from the encoder of Fountain codes, which is the exclusive-OR based sum of several randomly chosen source packets, will be extended by the FEC payload ID in order to form an encoded packet.

Fountain codes may be characterized by a factor graph having sparse edges, as seen in Figure 2.13, where circles denote the so-called variable nodes and squares denote check nodes. In other words, Fountain codes may be deemed to be sparse-graph codes, where the variable nodes represent the source packets and the check nodes represent the constraints. A notable characteristic of a sparse graph is that each constraint only involves a small number of variables in the graph ([62, Part VI]). Furthermore, sparsity implies having a low encoding and decoding complexity.

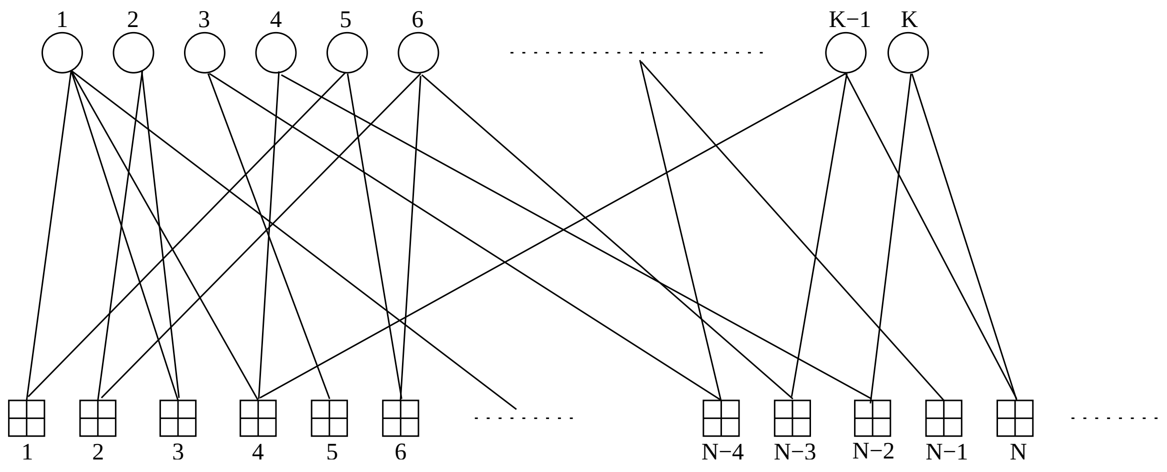


Figure 2.13: The sparse graph representing Fountain codes, where the hollow circles denote source packets. Each constraint represents the bitwise exclusive-OR sum of its several connected source packets.

Figure 2.14 illustrates the common coding process of Fountain codes. At the beginning, the source files or streams are partitioned into source blocks, which contain K source packets having the same packet lengths, as seen in Figure 2.14, where $P_1, P_2, ..., P_k$ are K constituent source packets in a source block. Then, the encoder continuously produces encoded packets. Each one of them is the sum of modulo-2 addition of several source packets, which are randomly chosen in a block. The positions of logical '1's in a specific column of $\mathbf{G}_n$ in Figure 2.14 indicate the corresponding source packets having the same positions in the source block, will be combined by the exclusive-OR function, in order to generate the nth encoded packets ($n \in [1, N]$). Moreover, the number of '1's, namely the number of source packets involved in the exclusive-OR operations for the sake of generating a single encoded packet is referred to as the 'degree' of that constraint. The constraint is denoted by a square in Figure 2.13. Furthermore, the number of lines connected to a specific square represents the 'degree' of that specific constraint. Hence, the average node degree determines the total number of operations at the encoder and decoder, i.e. the encoding and decoding complexity.

Mathematically, any encoded packet T_n may be expressed as:

$$T_n = \sum_{k=1}^{K} P_k G_{kn} \ . \tag{2.19}$$

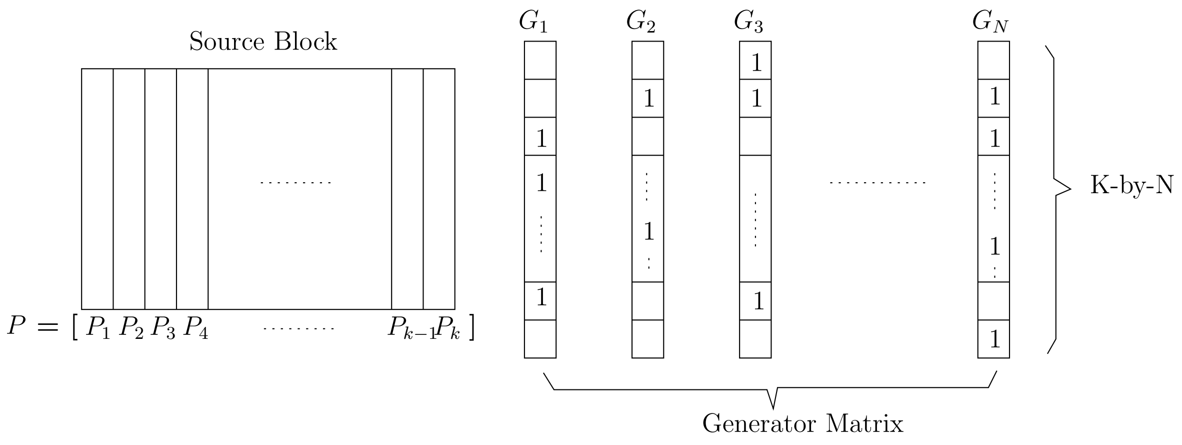


Figure 2.14: The encoding process of Fountain codes. In each column G_n, the number of logical '1's corresponds to the degree of resultant encoded packet, which is a random variable following a certain degree distribution. The positions of '1's is uniformly distributed and they identify the related source packets P_k, which will be added by a modulo-2 sum.

The following example demonstrates the action of Equation 2.19. Given $K = 8$ and $\mathbf{G}_1 = \{10110010\}$, we have:

$$\begin{aligned} T_1 =& \sum_{k=1}^{8} P_k G_{k1} \\ =& P_1 \cdot 1 \oplus P_2 \cdot 0 \oplus P_3 \cdot 1 \oplus P_4 \cdot 1 \oplus P_5 \cdot 0 \oplus P_6 \cdot 0 \oplus P_7 \cdot 1 \oplus P_8 \cdot 0 \\ =& P_1 \oplus P_3 \oplus P_4 \oplus P_7 \ . \end{aligned} \tag{2.20}$$

Theoretically, n may be increased to infinity, so that the transmitter acts like a fountain, 'spraying' the receiver with encoded packets. All instantiations of $\mathbf{G}_n$ may be interpreted as constructing a K-by-N-element matrix, which is referred to as the generator matrix $\mathbf{G}$, where K is the number of source packets and N is the number of transmitted encoded packets. Let $\mathbf{T}$ denote the vector including all encoded packets, which may be expressed as:

$$\mathbf{T} = \mathbf{P} \cdot \mathbf{G} \ , \tag{2.21}$$

where $\mathbf{P}$ is the vector of source packets. Hence, if we can find the inverse of the matrix $\mathbf{G}$, the vector of source packets may be determined as follows:

$$\mathbf{P} = \mathbf{T} \cdot \mathbf{G^{-1}} \ . \tag{2.22}$$

Finding the inverse generator matrix constitutes the first step of the decoding method that random linear Fountain codes adopt. By contrast, LT codes and Raptor codes exploit different techniques for reducing the complexity, as detailed in the following sections.

2.3.2 Random Linear Fountain Codes

The family of random linear Fountain codes [54] relies on the construction philosophy that the degree of each $\mathbf{G}_n$ obeys the uniform distribution over the interval $[1, K]$, and relies on the Gaussian elimination technique for finding the inverse generator matrix $\mathbf{G^{-1}}$.

Gaussian elimination [63] is a straightforward decoding algorithm employed for finding the inverse of the matrix $\mathbf{G}$ based on N transmitted packets. This algorithm firstly requires the generator matrix to be reproduced at the receiver by regenerating the same pseudo-random numbers as those used at the encoder. During this process, the FEC payload ID transmitted in the header of the received encoded packets is employed as a seed of the pseudo-random generator. More specifically, these random numbers indicate the number of '1's and their positions in $\mathbf{G}_i$ corresponding to successfully received encoded packets. Note that the reproduced generator matrix $\mathbf{G}$ represents only a part of the generator matrix at the encoder, since the corrupted encoded packets have been discarded. For example, assuming that we have $K = 4$ source packets and $N = 6$ transmitted packets, which are encoded based on the $\mathbf{G}_n$ sequence of Figure 2.15.

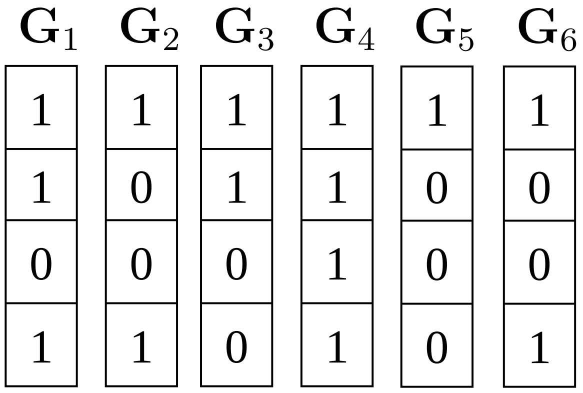


Figure 2.15: Exemplifying the generator matrix of a Fountain encoder.

More specifically, if the encoded packets corresponding to $\mathbf{G}_1$ and $\mathbf{G}_4$ are discarded due to bit errors, the reproduced generator matrix consists of $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_5$, $\mathbf{G}_6$. Since only full-rank matrices are invertible, the number of received encoded packets has to be at least K. However, the component $\mathbf{G}_i$ in this reproduced K-by-K-element matrix may be linearly dependent on the other component $\mathbf{G}_j (j \neq i)$, since both the number and the position of '1's are randomly determined at the encoder. In the above example,

$\mathbf{G}_2$ and $\mathbf{G}_6$ are identical and hence have a perfect correlation. This implies that the rank of the reproduced generator matrix is not full according to the Gaussian elimination algorithm. Therefore, the Fountain decoder requires another new encoded packet. For example, $\mathbf{G}_7$ will be generated for encoding the source packets in the example of Figure 2.15. If $\mathbf{G}_7$ happens to be correlated with some of the other components $\mathbf{G}_i (i < 7)$ again, the process of generating a new encoded packet will continue until the full-rank matrix may be reproduced at the receiver.

Therefore, even without considering any packet loss events during the transmission, the number of received packets may be higher than K in order to ensure there exists a full-rank K-by-K-element matrix. Let $(K + \epsilon)$ denote the number of received packets needed by this full-rank matrix, where ϵ indicates the excess number of packets beyond K. MacKay in [54] determined the upper bound of decoding failure probability, when ϵ excess packets have been received for any K, which is given by

$$\delta(\epsilon) \leq 2^{-\epsilon} \ , \tag{2.23}$$

where the function $\delta(\epsilon)$ explicitly quantifies the probability that the receiver cannot decode a specific source block after receiving ϵ excess encoded packets, which is equal to or less than $2^{-\epsilon}$. As observed in Figure 3 of [54], when the excess number of received packets is higher than 10, i.e. we have $\epsilon \geq 10$, the upper bound of failure probability becomes $2^{-10} \approx 0.001$. As the number of source packets K increases, the relationship of $K/(K + \epsilon) \approx 1$ indicates that Fountain codes are capable of approaching the Shannon capacity.

The encoding and decoding procedures of random linear Fountain codes impose a high complexity due to the associated Gaussian elimination and the uniform degree distribution. The complexity is analyzed quantitatively as follows,

- Encoding complexity: As mentioned in Section 2.3.1, the average degree of $\mathbf{G}$ determines the number of exclusive-OR additions during encoding, and hence determines the associated encoding complexity. In random linear codes, the average value of the uniform distribution over an interval of $[1, K]$ is $K/2$. Therefore, $O(K \cdot K/2)$ may be used to quantify the encoding complexity of random linear codes.
- Decoding complexity: According to Equation 2.22, the decoding of random linear codes includes two steps: finding the inverse of the generator matrix $\mathbf{G}$ by the Gaussian elimination and the matrix multiplication of $\mathbf{T} \cdot \mathbf{G}^{-1}$. The complexity of the classic Gaussian elimination used for inverting the matrix is $O(K^3)$ [64],

while $O(K \cdot K/2)$ characterizes the multiplication complexity of the K encoded packets involved, each of which has the average degree of $K/2$.

- Overhead: As argued above, random linear Fountain codes require an excess number of received packets than K for ensuring the reproduction of the full-rank generator matrix. Derived from Inequality 2.23, the estimated transmission overhead is $\approx K + \log_2(1/\delta)$, where δ is the decoding failure probability upon receiving $[K + \log_2(1/\delta)]$ encoded packets.

2.3.3 Luby Transform Codes

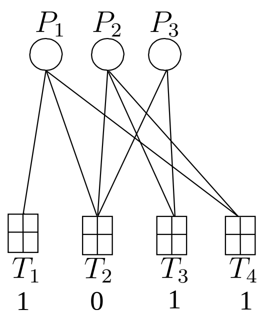
Inverting the matrix $\mathbf{G}$ in Equation 2.22 by Gaussian elimination does not result in an efficient decoding algorithm. Luby Transform (LT) codes attempt to reduce the encoding and decoding complexities and hence they employ a specifically designed degree distribution, as well as a message passing decoder [54], resulting in attractive low-complexity Fountain codes.

The encoding procedure of LT codes is similar to that of random linear Fountain codes. At the encoder, the degree of $\mathbf{G}_n$ is randomly generated from a specially designed distribution, which will be detailed in Section 2.3.3.2. However, the positions of '1's in $\mathbf{G}_n$ are uniformly distributed, like those in random linear Fountain codes. Then, according to Equation 2.19, the encoded packet $\mathbf{T}_n$ is also generated with the aid of $\mathbf{P} \cdot \mathbf{G}_n$. The transmitter continues to transmit the encoded packets, while the receiver collects the encoded packets and commences decoding immediately when an encoded packet having a degree of one is received. Again, the decoder invokes the message passing algorithm detailed in Section 2.3.3.1.

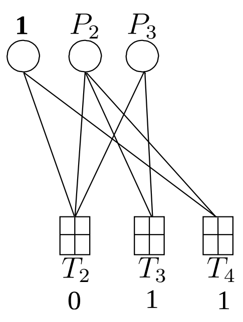
2.3.3.1 Message Passing Decoder

During the message passing based decoding, the key step is that of finding a degree-one encoded packet, which is dependent on a single source packet. Hence, this received packet indeed constitutes the source packet. Figure 2.16 [54] gives an example of the decoding procedure. For the sake of simplicity, we assume that the packet length is 1-bit.

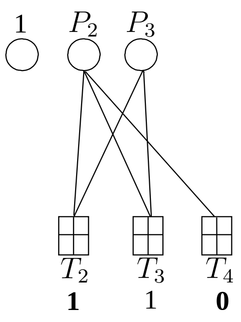
As seen in Figure 2.16(a), we have three source packets, namely P_1, P_2, P_3, and four constraints corresponding to four encoded packets, yielding $T_1 = 1, T_2 = 0, T_3 = 1, T_4 = 1$, which have a degree of $1, 3, 2, 2$ respectively. The encoded packet T_1 having a degree of one is only connected to the source packet P_1. After P_1 is decoded, all other encoded packets, which depend on P_1, are exclusive-OR added to P_1. In this example, the encoded packets T_2 and T_4 also depend on P_1. Therefore, the modulo-2 operations shown in Figure 2.16(b) are performed. Then, Figure 2.16(c) displays the isolated P_1 and the updated values of T_2 and T_4 after modulo-2 adding as well as removing the



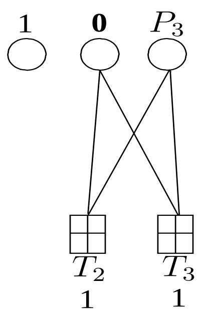
(a) T_1 is merely connected to P_1, hence P_1 is decoded as $P1 = T_1 = 1$.



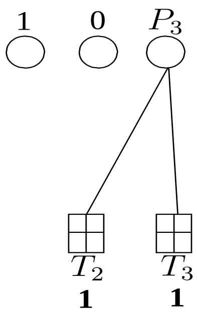
(b) The already known $P_1 = 1$ is also connected to T_2 and T_4. In this step, both T_2 and T_4 are exclusive-OR added by P_1.



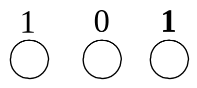
(c) $T_2 = T_2 + P_1 = 1$, $T_4 = T_4 + P_1 = 0$, then all the edges connected to P_1 are removed. As a result, T_4 becomes only connected to P_2, i.e. a degree-one node.



(d) After the step in sub-figure (c), P_2 can be decoded as $P_2 = T_4 = 0$, as repeating the step in sub-figure (a).



(e) The same step as in sub-figure (b): the decoded $P_2 = 0$ is also connected to T_2 and T_3, then T_2 and T_3 are, respectively, exclusive-OR added by P_2.



(f) Following the step in sub-figure (c): P_2 is removed from the graph, and P_3 is finally decoded as $P_3 = T_2 = 1$.

Figure 2.16: The explanation of LT decoding process for $K = 3$ source packets and $N = 4$ encoded packets, given that each packet has one bit length.

edges linked with P_1. It may be observed that a new degree-one constraint appears, namely that which is related to T_4. Hence, the following steps portrayed in Figures 2.16(d), 2.16(e), 2.16(f) gradually remove the inter dependencies of the packets, as in the previous steps, until the entire source block is recovered. However, in practical LT decoding, there is sometimes no new degree-one packet after decoding a certain source packet. In this case, the receiver has to pause the decoding process and wait for the transmitter to send extra encoded packets, until a degree-one packet arrives.

2.3.3.2 Degree Distribution

As introduced in Section 2.3.1, each constraint is associated with a certain degree, which is randomly generated from a specific degree distribution. Since the computational complexity of Fountain codes scales with the number of edges in the coding graph of Figure 2.13, an important objective of degree distribution design is to make the average degree as small as possible. As a result, the majority of packets has to have a low degree, so that the coding graph becomes sparser. Nonetheless, having a small fraction of high-degree constraints is also required for ensuring the protection of each source packet. This may assist us to avoid encountering decoding failures potentially caused by the total absence of some source packets amongst the encoded packets. Clearly, the degree-distribution design is heuristic and hence numerous designs may be conceived. Here we only consider the ideal soliton distribution [54] and the robust soliton distribution [54]. Their Probability Density Functions (PDFs) are highlighted below.

- Ideal soliton distribution [54]: It is expected to ensure that only a single degree-one encoded packet emerges at each decoding step. More explicitly, when the current degree-one encoded packet is processed, a new degree-one encoded packet is expected to emerge after all edges that were connected to the corresponding decoded source packet have been removed, as illustrated in Figure 2.16. If no degree-one packet appears at a specific decoding stage, the decoding process cannot continue. However, the ideal soliton distribution does not perform well in practice, because the random number generator used for determining the random degree of each of the packets is unable to consistently satisfy the above design constraints. Its PDF is formulated as [54]:

$$\begin{aligned} &\rho(1) = 1/K; \\ &\rho(d) = \frac{1}{d(d-1)} \quad for\ d = 2, 3, ..., K\ , \end{aligned} \tag{2.24}$$

  where $\rho(1)$ guarantees having only a single degree-one encoded packet at the initial decoding stage and $\rho(d)$ $(d \geq 2)$ satisfies the principle of having not necessarily just a single degree-one packet at each stage, but having a high proportion of low-degree (close to one) encoded packets. At the same time, we also need a certain small fraction of high-degree encoded packets for the sake of ensuring that none of the source packets remains unprotected. Furthermore, the average degree of this ideal solition distribution may be shown to be [55] $\sum_{d=1}^{K} [d \cdot \rho(d)] = \sum_{d=1}^{K} \frac{1}{d} \approx \ln K$. Naturally, it is possible to conceive other simple formulae for generating beneficial degree-distributions, which is the objective of the so-called robust soliton distribution introduced below. This distribution was plotted in Figure 2.17 for $K = 10,000$ source packets.

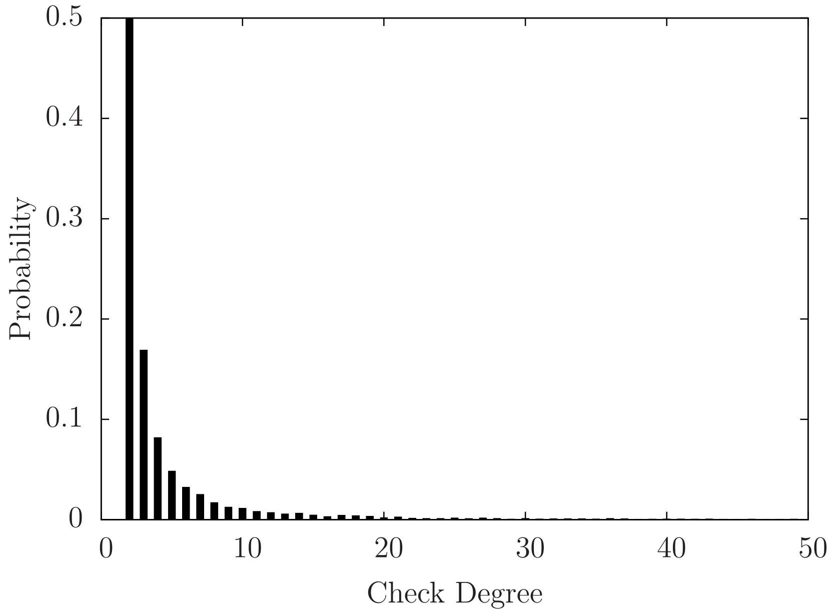


Figure 2.17: The ideal soliton distribution of $\rho(d)$ for $K = 10,000$.

- Robust soliton distribution [54]: With the objective of improving the above-mentioned soliton distribution, the authors of [54] introduced two new parameters, namely c and δ for the sake of adjusting the expected number of degree-one packets during the decoding process to be approximately:

$$S \equiv c \cdot \ln(K/\delta)\sqrt{K} \ , \tag{2.25}$$

rather than 1. The benefit of this is that it generates a higher fraction of degree-one encoded packets at any decoding stage and hence may reduce the probability of failure in comparison to the ideal soliton distribution, regardless of the pseudo-random behavior of the random number generator, because the modulo-2 connection of packets is controlled by a pseudo-random number generator, which is unable to maintain a fixed probability for finite-length sequences. The parameter c is a constant and δ denotes the decoding failure probability after receiving a certain number of encoded packets. The PDF of the robust soliton distribution, which is denoted by $\mu(d)$ may be expressed as follows [54]:

$$\mu(d) = \frac{\rho(d) + \tau(d)}{Z} \ , \tag{2.26}$$

where we have $Z = \sum_d \rho(d) + \tau(d)$ acting as the denominator for the normalization of $\mu(d)$. In order to adjust the expected number of degree-one packets to be S during the decoding process, the supplemental term $\tau(d)$ in Equation 2.26

may be chosen to have the following form [55]:

$$\tau(d) = \begin{cases} \frac{S}{K}\frac{1}{d} & \text{for } d = 1, 2, 3, ..., (K/S) - 1 \\ \frac{S}{K} & \text{for } d = (K/S) \\ 0 & \text{for } d > K/S \ , \end{cases}$$

although again, these heuristic formulae may be potentially replaced by others.

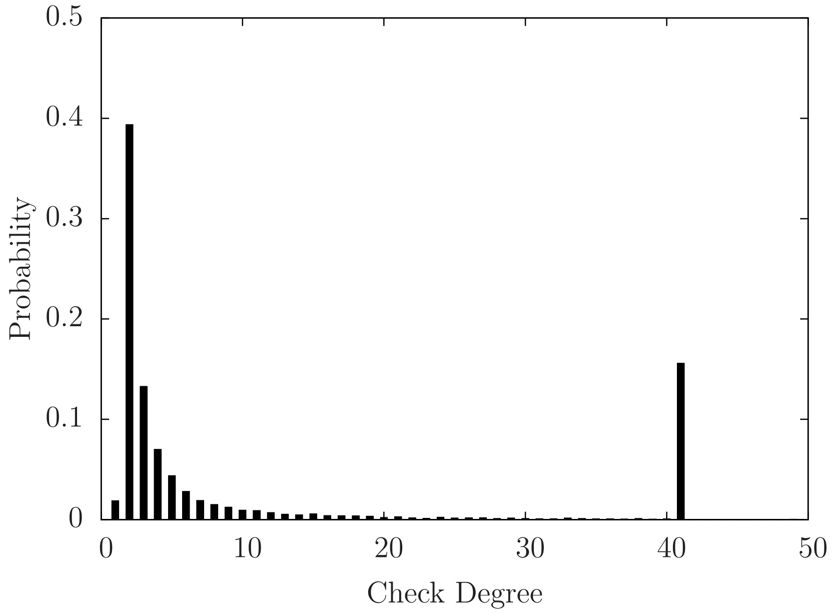


Figure 2.18: The robust soliton distribution of $\mu(d)$ for $K = 10,000$, $c = 0.2$ and $\delta = 0.05$, which gives $S \approx 244$, $K/S \approx 41$ and $Z \approx 1.31$.

Figure 2.18 illustrates the PDF of the robust soliton degree distribution for the specific parameters of $c = 0.2$ and $\delta = 0.05$, which exhibits a spike, when the degree is equal to $K/S = \frac{10,000}{244} \approx 41$. Based on this degree distribution, the average degree is on the order of $\ln(K/\delta)$ according to $\frac{\sum_{d=1}^{K} d \cdot [\rho(d)+\tau(d)]}{S}$ [55]. Hence, the encoding complexity must be at least on the order of $O\left(K \ln(K/\delta)\right)$, because at least K encoded packets are needed during LT encoding and each of them has an average degree of $\ln(K/\delta)$. Furthermore, the decoding complexity is also estimated to be on the order of $O\left[K \ln(K/\delta)\right]$, when the message passing decoder is employed, which involves the same number of exclusive-OR operations as the number of edges in the coding graph of Figure 2.13.

The packet overhead of LT codes may be adjusted by appropriately setting the parameters c and δ of Equation 2.25. Figure 7 of [54] has showed the actual number of received packets required in order to recover a file having $K = 10,000$ source packets for different combinations of c and δ. In general, this file may be recovered with a near-unity probability, when the packet overhead is between 5% to 10%, $\delta = 0.5$ and c varies from $0.01 - 0.1$.

2.3.4 Raptor codes

Raptor codes [56] combine the philosophy of LT codes with traditional channel codes, as illustrated in Figure 2.19. By doing so, Raptor codes may reduce the complexity further and achieve a linearly increasing encoding and decoding complexity, as a function of the number of packets involved. This makes them attractive for practical applications. For example, the authors of [65] employed Raptor codes for video packet loss recovery, while the authors of [66] combined Raptor codes with network coding conceived for Internet Protocol based Television (IPTV) services. In [67], a Raptor codes based scheme was proposed for solving the problem of distributed source coding. Furthermore, Raptor codes have been recommended in the Content Delivery Protocol (CDP) standard of the Digital Video Broadcasting (DVB)-H Annex C,E recommendation [68] as an FEC scheme for file transfer.

Figure 2.19 describes the encoding procedure of Raptor codes, where the original source packets are pre-encoded by high-rate traditional channel codes, such as systematic Low Density Parity Check (LDPC) codes, before entering the second-step of LT encoding. The high-rate pre-encoding produces M intermediate packets including K systematic packets and $(M - K)$ parity packets. These M intermediate packets are input into the second-step LT encoder, which then transmits a potentially limitless number of encoded packets to the receiver.

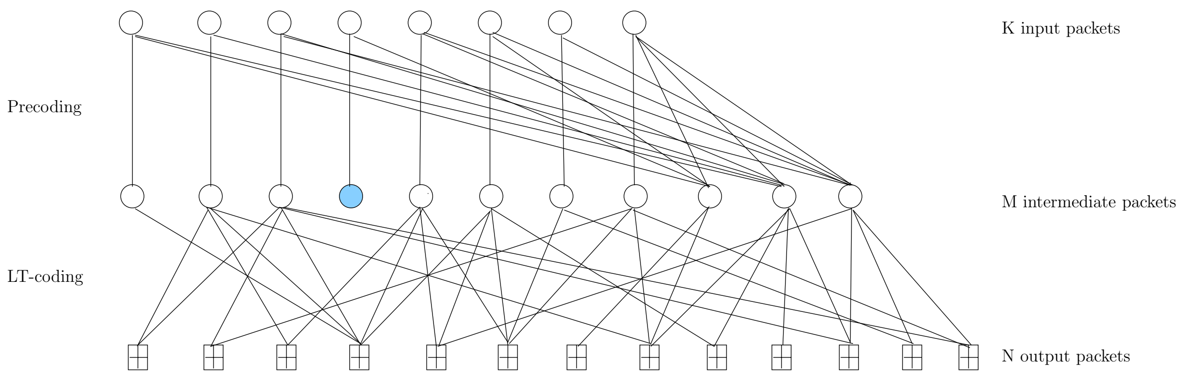


Figure 2.19: Raptor Codes: K input packets are pre-encoded to $M > K$ intermediate packets using traditional erasure codes; then the having a reduced average degree LT codes process the M intermediate packets to obtain N encoded packets, which will be transmitted to the receiver. The node denoted by the filled circle in the set of intermediate packets is not protected by the LT codes having a reduced average degree.

As introduced in Section 2.3.3, the degree distribution of LT codes has to have a number of high-degree packets to ensure that every source packet is represented with a high probability. However, with the aid of pre-encoding, the second-step LT codes in Raptor codes may adopt a degree distribution having a reduced average degree. For example, Shokrollahi introduced several degree distributions for various values of K in Table I of [56]. One of the frequently used distributions is the so-called Truncated Poisson 1 (TP1) distribution, which may be exemplified by the following polynomial:

$$\begin{aligned}\delta(x) = &\, 0.008 + 0.5x + 0.17x^2 + 0.073x^3 + 0.083x^4 + \\ &\, 0.056x^7 + 0.037x^8 + 0.056x^{18} + 0.025x^{64} + 0.003x^{65}\end{aligned} \tag{2.27}$$

and Figure 2.20 visualizes the TP1 distribution for $K = 10,000$ source packets. Based on this reduced-average degree distribution and on the appropriately selected pre-codes, Raptor codes may have an encoding and decoding complexity, which is linearly proportional to K.

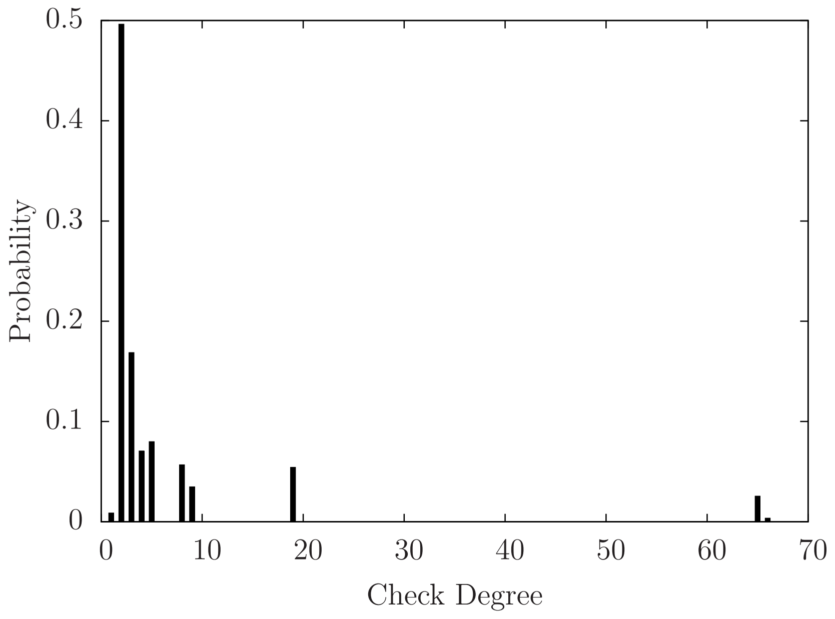


Figure 2.20: The TP1 distribution for $K = 10,000$ represented by Equation 2.27.

Again, the TP1 degree distribution may not represent every source packet in the output LT coded packet stream, as exemplified by the filled circle in Figure 2.19. However, these unrepresented packets may be treated as lost packets. The receiver also follows the message passing algorithm of LT codes in order to decode the M intermediate packets. Although the number of packets decoded during the first-step may be less than M, the ensuring decoding of the traditional codes, such as LDPC codes may still succeed in finally recovering all the original K source packets.

2.3.5 Practical Implementation Issues

A number of practical issues have to be taken into account, when implementing the encoder and decoder of Fountain codes.

In order to realize Fountain codes, we have to generate random packet degrees obeying a specific distribution, such as the LT codes' robust soliton distribution or TP 1 degree distribution. The so-called 'inverse transform sampling' method may be employed for generating random numbers obeying an arbitrary distribution, provided that the Cumulative Distribution Function (CDF) of that distribution is known. It has been demonstrated in [69, Chapter 2, Section 2] that the dependent variable of any distribution's CDF obeys the uniform distribution on $[0, 1]$. Therefore, the uniform random number generator and the inverse function of the CDF may be combined to produce samples from a specific degree distribution. More particularly, for example, the PDF of the robust soliton distribution was formulated in Equation 2.26, while its CDF can be derived by integrating the PDF, expressed in discrete terms as:

$$F(d) = \sum_{d_i \leq d} \mu(d_i) \ , \tag{2.28}$$

where the independent variable d is a discrete degree varying across the range of $[1, K]$. The dependent variable of $F(d)$ varies in the range of $[0, 1]$, obeying the uniform distribution. When the uniform random generator creates a random number θ between 0 and 1, the corresponding degree d may be determined by comparing each $F(d_j)$ for all valid $d_j \in [1, K]$ values to this random number θ, since the values of $F(d)$ monotonically increase. More explicitly, if θ satisfies $F(d_{j-1} \leq \theta < F(d_j)$, the determined degree will be $d = d_{j-1}$.

As mentioned before, the indices of source packets involved in the exclusive-OR additions are uniform-randomly chosen over $[1, K]$ in all Fountain codes, after the degree of a encoded packet has been determined. Namely, the number of 1s for any G_i of the generator matrix in Equation 2.21 was known, its random positions of '1's obeys the uniform distribution over $[1, K]$. During the generation of these uniform random positions, generating the same 'pseudo-random' number should be avoided. More explicitly, the result of exclusive-OR additions of the same source packets will be zero, if the degree of this G_i happens to be an even number. For example, given that the current degree is 2, if two identical random numbers e.g. two 1s are generated, indicating that the same source packet of P_1 will be XOR added twice, the encoded packet is then equal to $P_1 \oplus P_1 = \mathbf{0}$. This particular transmission is not beneficial, since it does not provide any useful information for the receiver at all. Although the probability of this particular case is low, it is still wasteful especially for a short block length K.

Despite the advantages of the message passing algorithm in the context of LT codes, the Gaussian elimination based decoder may still find practical applications.

For example, Annex E of the standard formulated in [68] recommends a decoding algorithm for Raptor Codes based on Gaussian elimination. This is likely to be, because practical applications show that the message passing decoder has a relatively long delay, since it might have to pause and wait for the arrival of a degree-one encoded packet. However, the inverse generator matrix based on Equation 2.21 where all G_i constituent components are gleaned from the received encoded packets may satisfy the full-rank condition at a faster pace than the decoding relying on the arrival of a degree-one encoded packet. On the other hand, when K has a moderate value, the complexity of Gaussian elimination may be acceptable for state-of-the-art Central Processing Units (CPU) used even in embedded devices.

If a Gaussian elimination based decoder is chosen, it is better to calculate the inverse generator matrix in a so-called streaming mode as explained below, since the receiver receives encoded packets one by one. When the receiver has received K LT-encoded packets, it becomes possible to obtain a full-rank generator matrix, and then the matrix inversion operation of Equation 2.22 may be triggered. Returning to the example of Figure 2.15, Figure 2.21 portrays the first step of the generator matrix inversion based on the Gaussian elimination, where $\mathbf{T}_i$ represents the received encoded packets and the reproduced matrix is also constituted by $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_5$, $\mathbf{G}_6$ of Figure 2.15.

$$[\mathbf{T}_2, \mathbf{T}_3, \mathbf{T}_5, \mathbf{T}_6] \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\Bigg\Downarrow \begin{array}{l} column1 \cdot (-1) + column2 \\ column1 \cdot (-1) + column3 \\ column1 \cdot (-1) + column4 \end{array}$$

$$[\mathbf{T}_2, \mathbf{T}_3 - \mathbf{T}_2, \mathbf{T}_5 - \mathbf{T}_2, \mathbf{T}_6 - \mathbf{T}_2] \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Figure 2.21: Exemplifying the Gaussian elimination algorithm.

As Figure 2.21 shows, Gaussian elimination based matrix inversion attempts to convert the generator matrix to an identity matrix by a number of multiplication and addition operations. Meanwhile, the corresponding operations are performed on the encoded packets for the sake of generating the decoded packets. If the inverse generator matrix is not found after this matrix inversion operation based on K received packets due to failing to satisfy the full-rank condition, the receiver pauses decoding and waits for the new encoded packets. At this point, the generator matrix has already been transformed to a triangular matrix having several right columns constituted by zeros, as seen in Figure 2.21. When the new encoded packets arrive, all zero lines in the right part of the transformed generator matrix are replaced by their corresponding constituent component G_i. In this example, the new $\mathbf{G}_7$ will replace the last right column of the generator matrix and the corresponding $\mathbf{T}_7$ will replace $\mathbf{T}_6 - \mathbf{T}_2$ in the encoded packet vector, as seen in Figure 2.21. Then, the decoder restarts the Gaussian elimination based matrix inversion operation for this partially inverted matrix, instead of constructing the generator matrix from the original components $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_5$ and $\mathbf{G}_7$. Since the corresponding calculations are applied to the received LT-encoded packets along with the generator matrix inversion operations, the decoded packets are obtained immediately after the matrix inversion was completed.

2.4 Development Platforms

2.4.1 Introduction

NS2 [70] was developed by the Defense Advanced Research Projects Agency (DARPA), which is an agency of the United States Department of Defense in 1995 for network-layer research. As a popular network simulator, it may provide wide support for almost all different types of networks, such as wired Local Area Networks (LANs), wireless LANs/Personal Area Networks (WPANs), satellites, etc. For example, it has application layer protocols, such as the File Transfer Protocol (FTP) and the Constants Bit Rate (CBR), transport layer's Transport Control Protocol (TCP) as well as User Datagram Protocol (UDP) and the 802.11 and 802.15 wireless MAC layer protocols.

NS2 has provided four types of Ad-hoc routing protocols, namely the Dynamic Source Routing (DSR), the Destination-Sequenced Distance Vector (DSDV) routing, the Ad-hoc On-Demand Distance Vector (AODV) routing and the Temporally-Ordered Routing Algorithm (TORA) protocols. On the other hand, the Institute of Electrical and Electronics Engineers (IEEE) society has established three wireless network standards for the MAC layer, namely 802.11 for Wireless Local Area Networks (WLANs), 802.15 for Wireless Personal Area Networks (WPANs) and 802.16 for Wireless Metropolitan Area Networks (WMANs). The implementations of all these standards can be found in the NS2 platform.

Unfortunately, the NS2 developers have not invested much effort in the simulation of a realistic physical layer, which is a particular problem in cross-layer research. Hence, NS2 does not invoke any parity check codes and no packet are lost owing to bit errors. Moreover, the data structure of a packet has an non-contiguous memory in the NS2 platform, which is inappropriate for considering correlated fading channels, for example.

In order to investigate whether a particular physical layer communication technique is capable of improving the entire system, it is necessary to extend the NS2 simulation platform based on its current functions, by improving the packet structure, adding CRC in the MAC layer, adding diverse modulation and channel models, as well as channel codes and Multiple-Input Multiple-Output (MIMO) components, etc.

These PHY layer functions have to be incorporated, which are part of the IT++ library. IT++ is a C++ library of mathematical, signal processing and communication classes and functions [71]. We integrated IT++ with NS2 for carrying out our cross-layer-operation aide research.

2.4.2 Integrating IT++ with NS2

We firstly provide a brief introduction to the NS2 framework, processing flow, as well as data structures.

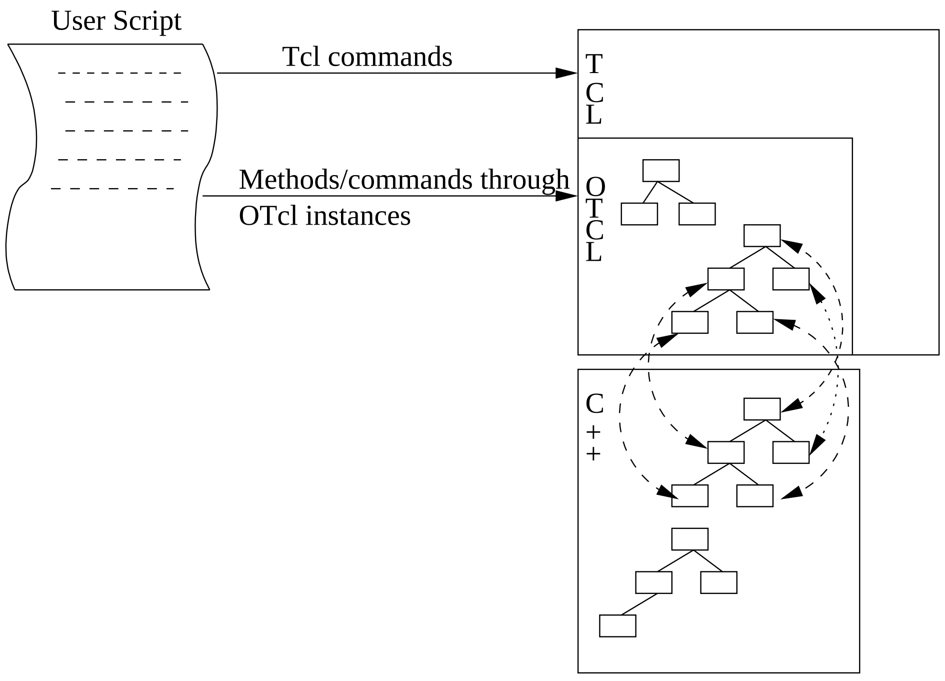


Figure 2.22: The relationship between TCL/OTCL with C++ classes.

NS2 is a network simulator platform based on mixed languages consisting of a script language - the so-called Tool Command Language (TCL) [72] and a compiling language of C++. Figure 2.22 portrays the framework of NS2, as well as reveals the relationship between the TCL and C++ programs. The script language of TCL used in the NS2 makes the construction of a network easier and faster, while the C++ code improves the speed of simulations. Developers may combine their own C++ classes with TCL commands and Object TCL Extensions (OTCL) classes, which are mapped to OTCL objects and hence are opened to the TCL layer. The combined broken lines with arrows in Figure 2.22 indicate this mapping between OTCL classes and

C++ classes. Additionally, an interpreter is responsible for executing the user's TCL programs, which reads the script lines one by one, then parses commands and executes them. More particularly, the execution of TCL scripts implies that the interpreter will allocate memory for the global variables of TCL. It will also load procedures, invoke the corresponding functions related to standard TCL commands, create OTcl instances, and so on.

An 802.11 WLAN model is basically composed of channels and mobile nodes. The structure of an 802.11 WLAN having two nodes in the NS2 platform is shown in Figure 2.23, where all elements in a node are derived from a basic class referred to as a '**Connector**', e.g. '**RouteAgent**', '**LL**', '**Queue**', '**MAC**' and '**NetIF**' in Figure 2.23. A specific element or the combination of several elements may implement the functions of a network layer and work in an event-driven mechanism. Additionally, NS2 offers an OTcl class referred to as '**Simulator**' as a running platform. The user's script as seen in Figure 2.22 may configure nodes, attach transport layer agents, build application sources and sinks, schedule time events, and finally start it with the aid of the '**Simulator**'. During program execution, the control will be handed to a C++ class referred to as '**Scheduler**' and the program enters into an event loop until the events in the event queue all took place or the user proactively exits. Therefore, the C++ classes in the NS2 may be divided into two types: one is related to the network components and the other is related to the event scheduler.

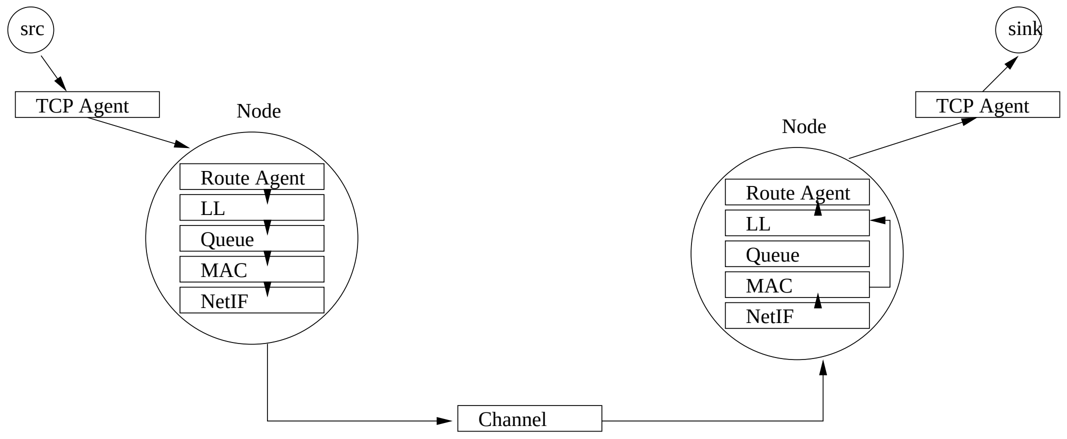
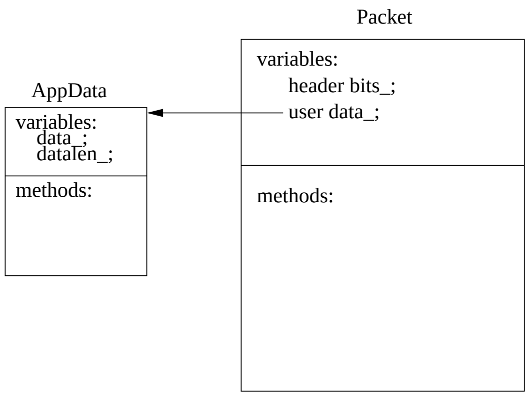


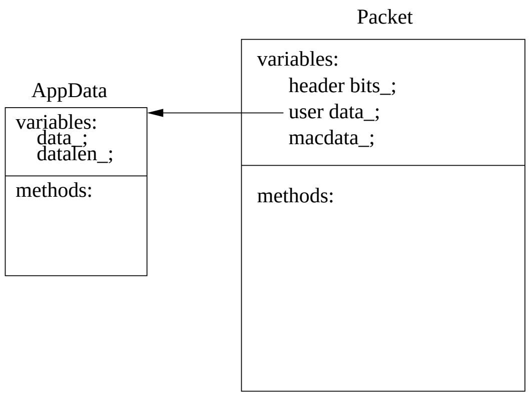
Figure 2.23: A simple NS2 wireless network structure.

The packets flow from the upper layers to the lower layers, passing through the channel in the physical layer and getting to the destination, where they are now fed from the lower layers to the upper layers. During this process, each **Connector**'s 'recv' method will be directly invoked for conveying packets between the network layers, or an event that will trigger a packet transfer is formed and submitted to the event queue, until the lower or upper elements handle it at the specified time. Here, the lower or upper elements depend on the direction of the data flow. In other words, the packets flowing among network components are conveyed under the control of the event scheduler of the NS2 platform.

As mentioned above, apart from the path loss, NS2 assumes perfect channels. For convenient access, the packet structure is initially designed as a C++ class having different variables for denoting the header bits of the various Open Systems Interconnection (OSI) layers and the user data, as seen in Figure 2.24(a). This design allows the data of a packet to be stored in separate memory locations. Furthermore, the header bits seen in Figure 2.24(a) represent the headers of network layers, but also contain some common information related to the NS2 simulator, which do not belong to a transmitted packet. Furthermore, if the programmer has not indicated in the TCL script the specific type of protocols that the current simulation requires, the header of all protocols that the NS2 supports will be embodied in the structure of a packet. This is inconsistent with a real packet's transmission. Moreover, the variable related to a specific packet's data is not a pure string, but an object of the class '**AppData**', hence it is quite a challenge to access its information for the sake of imposing the effects of noise or interference, as the packet is passing through the channel.



(a) Original packet data structure in NS2.



(b) New packet data structure added by a field with 'bvec' type.

Figure 2.24: Packet data structure

Nonetheless, it is possible to modify the C++ class '**Packet**' of Figure 2.24(a) in order to adapt to hostile channels. First of all, the common information related to the NS2 simulator records the specific size of a packet, since the size of a packet is dynamic, when it is being processed at any network layer. After this packet was passed down to the MAC layer, its size is finally determined before transmission. Therefore, we incorporate a new field referred to as 'macdata_' with the type of 'bvec' - a kind of IT++ data structure - into the class '**Packet**', as seen in Figure 2.24(b). Then, we generate the packet size of random data and store them in the above-mentioned field 'macdata_', before the packet is passed on to the channel. There is no need to copy the original headers and packet data into the new binary vector of 'macdata_', because lower layers do not care, what contents they process, their only concern is whether there are any errors. The 802.11 MAC protocol requires a CRC check for the reconstructed packet at the receiver. In order to invoke CRC checking, parity bits have to be calculated based on the random data stored in the field of 'macdata_', and then they are appended at the end of it at the transmitter. Again, the CRC classes provided by IT++ are used for carrying out this function.

Likewise, the C++ classes of modulation available in the IT++ library will be invoked by the NS2 PHY layer. Considering the modulation schemes specified in the 802.11 protocol, only BPSK and Quadrature Phase Shift Keying (QPSK) classes are imported into NS2. A new so-called '**Signal**' C++ class is built for the modulated symbols which is derived from the NS2 C++ class '**Packet**', so that the original packet information may be used, when the modulated symbols pass through the channel in the amalgamated NS2 IT++ environment. The last step is that of implementing a realistic noisy channel, which resorts to the NS2's class **Channel**. However, this class **Channel** simply checks, how many mobile nodes roam in the adequate-reception area of the source node and then broadcasts the packet to be transmitted to them. Since Rayleigh fading channels will be adopted in our scenarios, a new channel class is created by referring to the Rayleigh fading and Gaussian noise model of IT++. This new class retains the original channel functions of the NS2 channel class and it is exported to TCL, so that the programmer can invoke a Rayleigh channel in the scripts. Figure 2.25 portrays the integration of NS2 with IT++. Finally, IT++ library may be linked into the NS2 makefile.

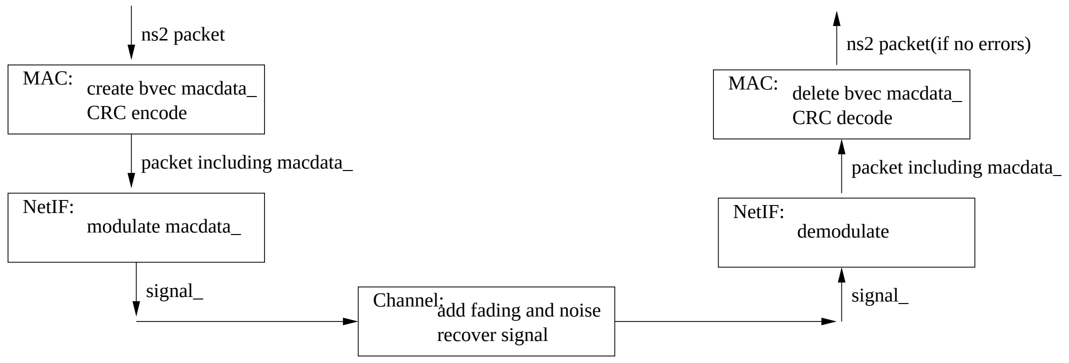

ns2 packet
MAC: create bvec macdata_ CRC encode
packet including macdata_
NetIF: modulate macdata_
signal_
Channel: add fading and noise recover signal
signal_
NetIF: demodulate
packet including macdata_
MAC: delete bvec macdata_ CRC decode
ns2 packet(if no errors)

Figure 2.25: Integration of ns2 and IT++.

2.5 Chapter Summary

This chapter commenced with the concept of Shannon's capacity, followed by a rudimentary introduction to turbo codes and Fountain codes, which constitute the foundation of the cross-layer schemes proposed in the later chapters. The key points of this background chapter are as follows:

- Shannon's capacity may be interpreted as the maximum MI that a specific channel can convey. There are three points that we should note concerning this concept. Firstly, the capacity is equivalent to the MI, which quantities the uncertainty reduction concerning the source's original information after receiving the transmitted symbols. Secondly, it represents the maximum MI for all possible input distributions. Finally, it is related to a specific channel, since different type of channels have their own capacities.
- Turbo codes are capable of achieving a near capacity performance by exchanging extrinsic information between two constituent components, which are generally convolutional codes concatenated in parallel. The extrinsic information increases along with number of iterations, until the turbo decoding process finally converges. Furthermore, this convergence property may be characterized with the aid of EXIT charts, which visualize the EXIT function of a BCJR decoder between the extrinsic information and the *a priori* information.
- Fountain codes are capable of providing an arbitrary high number of encoded packets, which were designed for transmissions over BECs. The encoder randomly chooses a number of packets from a block of K packets and generates their exclusive-OR connection. The determination of the number of that are combined by modulo-2 addition packets follows specific degree distributions. The robust soliton distribution is capable of maintaining low average degree, while representing all source packets at a high probability. The decoder may employ message passing algorithm for reducing the decoding complexity. This algorithm infers the original source packets from the degree-1 encoded packets directly. Eliminating the effects of this decoded packet from all others will reduce the degree of the other associated encoded packets. In order to continue decoding, a new degree-1 encoded packet is expected to emerge. Otherwise, the decoder fails or pauses, until it receives the next degree-1 encoded packet generated at the transmitter.
- NS2 and IT++ are two main platforms conceived for supporting the network simulations. NS2 concentrates on the upper layers beyond the PHY layer. By contrast, IT++ has a rich selection of classes and functions in the field of mathematics, signal processing and communication. The integration of NS2 and IT++ constitutes a potent amalgam.

Based on the material provided in this chapter, we will design several novel cross-layer strategies in the forthcoming chapters. More explicitly, we apply Fountain codes to aid file transfer in the scenario of IEEE 802.11 WLANs in Chapter 3 and utilize

MCTCs to construct low complexity turbo coded HARQ schemes, which will be detailed in Chapters 4 and 5.

Chapter 3

Fountain Code Aided File Transfer

In this chapter, we apply Fountain codes for protecting file transfer in 802.11 Wireless Local Area Networks (WLANs). Our objective is to optimize the parameters of Fountain codes, instead of the algorithm itself. We will demonstrate that the maximum transmission efficiency may be achieved based on the optimized parameters.

3.1 Benefits of Fountain Codes

In telecommunication networks reliable transmission constitutes the ultimate design objective. There are numerous solutions which are capable of enhancing the transmission reliability, but in this thesis we focus our attention on Forward Error Correction (FEC) and Automatic Repeat reQuest (ARQ).

The family of FEC codes is capable of recovering the information bits by incorporating carefully controlled redundancy based on different codes. The first FEC code was the single error correcting Hamming code [73], which was invented in 1950. Since then, more potent codes have been developed, such as Bose-Chaudhuri-Hocquenghem (BCH) block codes, Convolutional Codes (CC) and the aforementioned turbo codes as well as fountain codes of Chapter 2. Figure 1.1 of [47] outlines the brief history of FEC codes. According to Shannon's theory, any number of errors imposed by transmission over a specific channel may be corrected with the aid of parity bits. The ratio of the number of information bits to the total number of information and parity bits defines the coding rate. The capacity introduced in Section 2.1 determines the upper bound of the coding rate that any FEC code may be able to achieve at a certain Signal Noise Ratio (SNR). Since the transmission of these parity bits requires an increased bandwidth, the maximum coding rate that an FEC code can have, while still recovering the information bits becomes a useful criterion for quantifying the capability of FEC codes. Their decoding complexity is also a critical factor in the evaluation of FEC codes. Researchers have studied the associated tradeoff between these two aspects, when choosing a specific FEC code for a communication system.

On the other hand, ARQ techniques are also capable of improving the successful transmission probability by retransmitting the information packets. When the receiver detects that a packet has been correctly received, it will feed back an ACKnowledgement (ACK) message to the transmitter. Otherwise, it may send back a negative ACK or wait for the transmitter to time out. The transmitter will continue retransmitting the specific information packet, until it receives the ACK message or a maximum retry limit is reached. In the early ARQ designs, each retransmitted packet was independently processed. Therefore, a packet may become discarded, regardless of how many retransmissions are allowed, if all transmissions encounter hostile channel conditions. However, in more sophisticated schemes, all corrupted replicas may be combined with the aid of soft decisions to successfully decode the original information bits. This is philosophically similar to repetition codes [74]. In other words, this forms a naive example of the combination of FEC and ARQ, which is referred to as Hybrid ARQ (HARQ).

In Chapter 4, we will briefly summarize the family of HARQ types, which overcome the disadvantages of FEC and ARQ. More specifically, FEC may waste bandwidth by unnecessarily transmitting parity bits for protecting the transmissions over a channel having a good condition, while ARQ degrades the throughput drastically for transmissions over a hostile channel. Traditionally, FEC codes are employed in the PHY layer, while ARQs usually appear in the upper layers of networks. For example, both the Internet Transmission Control Protocol (TCP) and the IEEE 802.11 Media Access Contention (MAC) protocol have adopted ARQs for ensuring reliable transmissions.

In fact, FEC codes may be applied in any layer of networks. We refer to 'AL-FEC', when FEC is employed at the application layer. As multimedia applications spread to the Internet, FEC was invoked for avoiding packet loss events in video streaming. This motivated the employment of AL-FEC. Table 3.1 reviews the literature of AL-FEC, where we may find that the AL-FEC is mainly used for streaming or file download services in delay-tolerant broadcast or multicast networks. Furthermore, the AL-FEC based on Raptor codes has been adopted by the Digital Video Broadcasting (DVB) [68, 75] and by the 3rd Generation Partnership Project (3GPP) [76].

In recent years, the IEEE 802.11 WLAN protocol [85] has gained popularity for end-user connectivity. However, 802.11 data transmissions may suffer from high Packet Loss Ratio (PLRs) owing to the hostile characteristics of wireless channels, like multipath fading, shadowing and noise. Therefore, improving the PLR performance of

Table 3.1: Major contributions addressing AL-FEC in wireless networks.

Author(s)	Contribution
Wang *et al.* 1998 [77]	reviewed the error control strategies conceived for video communication and studied the transmission of MPEG-2 packets in a wireless ATM local area network, protected by Reed-Solomon (RS) codes.
Xu *et al.* 2003 [78]	investigated the performance of FEC and ARQ in the MAC and video application layers in the Point Coordination Function (PCF) mode of 802.11 WLANs, also based on RS codes.
3GPP 2005 [76]	Raptor codes were standardized in the 3GPP specification: Multimedia Broadcast Multicast Service (MBMS) for file download.
DVB 2006 [68]	Raptor codes were adopted for file delivery in the Content Delivery Protocol (CDP) of DVB standards.
DVB 2007 [75]	Raptor codes were adopted for protecting packet flows in the DVB system: Transport of MPEG-2 TS Based DVB Services over IP Based Networks.
Luby *et al.* 2007 [79]	investigated the AL-FEC of MBMS in an overall system context as a function of user mobility in a cellular network.
Degrande *et al.* [80] 2008	compared the family of application layer techniques conceived for mitigating packet loss in Internet Protocol based Television (IPTV) networks, which specifically included FEC based on RS codes and ARQ.
Gomez-Barquero [81] *et al.* 2009	extended the standardized AL-FEC using Raptor coding for streaming services in DVB-H. A significant robustness gain may be obtained at the expense of increased network latency.
Gomez-Barquero [82] *et al.* 2010	combined the raptor coding AL-FEC and multiprotocol encapsulation interburst FEC for the sake of increasing the transmission robustness in DVB-H systems.
Seferoglu *et al.* 2010 [83]	designed a dynamic AL-FEC for media flows, where the quality of real-time communications is degraded by TCP congestion.
Alexiou *et al.* 2011 [84]	studied the impact of AL-FEC in mobile multicast transmissions.

802.11 WLAN schemes constitutes a promising area of research. Traditional methods of protecting data transmissions include aforementioned ARQ [47], FEC [86] and hybrid FEC-ARQ schemes. The 802.11 MAC protocol has adopted the ACK and retransmission based approach for its reliable data exchange. Indeed, many recent studies [87, 88] have considered channel-quality-dependent adaptations of the physical (PHY) and MAC layers, in order to improve their reliability.

Nonetheless, it would be beneficial to improve the achievable end-to-end Quality of Service (QoS) without modifying the lower layers. For example, RS codes have found popularity as AL-FECs [89–91] owing to their maximum minimum free distances, which facilitate robust communications.

However, Fountain codes [54] have been shown to offer advantages over RS codes in terms of reduced complexity and improved coding efficiency [92, 93]. Again, RS codes have maximum-minimum free distances and their coding rate is K/N, where K is the number of non-binary information symbols, N is the number of encoded symbols and an (N, K) RS code is capable of recovering (i.e filling) up to $(N - K)$ erased symbols, provided that the index of the erased symbols is known. Similar statements may also be made in terms of filling entire erased packets. By contrast, as introduced in Section 2.3, Fountain codes are rateless erasure filling codes, which can send exactly the required number of encoded packets, which is needed to ensure that the original data file is recovered. For a block of K source packets, the Fountain decoder can recover the entire block with a high probability, when it receives at least $K(1+\epsilon)$ encoded packets, where ϵ is referred to as the transmission overhead. Furthermore, Fountain codes are more amenable to soft-decision decoding than RS codes. Luby Transform (LT) and Raptor codes, both of which constitute members of the Fountain codes family have also been shown to impose significantly lower complexities [54, 56]. Owing to these benefits, the related family of Raptor codes was standardized both for DVB [68, 75] and by the 3GPP [76] as an AL-FEC scheme in their IPTV solution.

Therefore, a robust performance may be expected when Fountain codes are applied for protecting file transfer in 802.11 WLANs. However, we commence by investigating the File Transfer Protocol (FTP) [9] over TCP, which entirely depends on the TCP's ARQ mechanism for maintaining the required end-to-end reliability. The TCP's ARQ mechanism will guarantee the correct reception of each packet in the transport layer, regardless of how long the transmission of this packet will be delayed. If the transmitter does not receive the ACK message in time, it will 'back off' by a certain period of time and then transmits the packet again. The backoff-time is doubled for every retransmission, until it reaches an upper bound, which is defined as 60 seconds in RFC2988 [94]. The retransmissions will continue, unless the upper application layer shuts down the connection owing to its unacceptable delay. FTP over TCP may work well in wired networks, since the PLR is typically low. By contrast, it may fail even at high SNRs due to a timeout in wireless networks. Table 3.2 characterizes the performance of the FTP over TCP protocol in a 2-node 802.11 WLAN for transmission over a non-dispersive uncorrelated Rayleigh fading channel. More explicitly, it shows whether the file may or may not be successfully received as a function of the SNR, when transmitting a $201,582$-Byte file within the timeout limit of $5,000$ seconds and using a 100-Byte packet size. The average delay in Table 3.2 indicates the average length of time during which the whole file may be received.

SNR(dB)	Successfully received?	Avg Delay(s)
30	Y	18.07
29	Y	23.33
28	Y	34.92
27	Y	90.16
≤ 26	N	$---$

Table 3.2: The status and average delay of FTP over TCP in a 2-node 802.11 WLAN over a non-dispersive uncorrelated Rayleigh fading channel

As seen in Table 3.2, the file transfer fails, when the SNRs become less than 26dB. For a file having a 201K-Byte size, where $5,000$s was deemed to be the highest tolerable waiting time for the 1Mbps data rate of 802.11b. Furthermore, the situation will get worse, when the packet size is increased. This implies that the performance becomes unacceptable, if the file transfer in wireless networks only depends on a single error control mechanism, i.e. ARQ in the TCP and MAC layers. Hence, we invoke AL-FEC techniques for providing extra protection.

The architecture of the 802.11 WLAN equipped with an AL-FEC scheme is illustrated in Figure 3.1. When implementing Fountain codes as AL-FEC schemes, a range of further issues should be considered. These issues are not about the encoding and decoding algorithms described in Chapter 2, they are more related to the procedures combined with FEC coding, as defined in [95] and [92]. More specifically, these include the determination of the packet length, block partitioning method, FEC payload ID format and so on. The appropriate selection of these parameters controls the receiver's delay, while maintaining a low PLR. Each of above-mentioned standards has also paid attention to these considerations. The standards [68,75,76] have defined the method of constructing source blocks, the source block index, etc. However, they were designed for broadcast networks, rather than for half-duplex 802.11 WLAN environments.

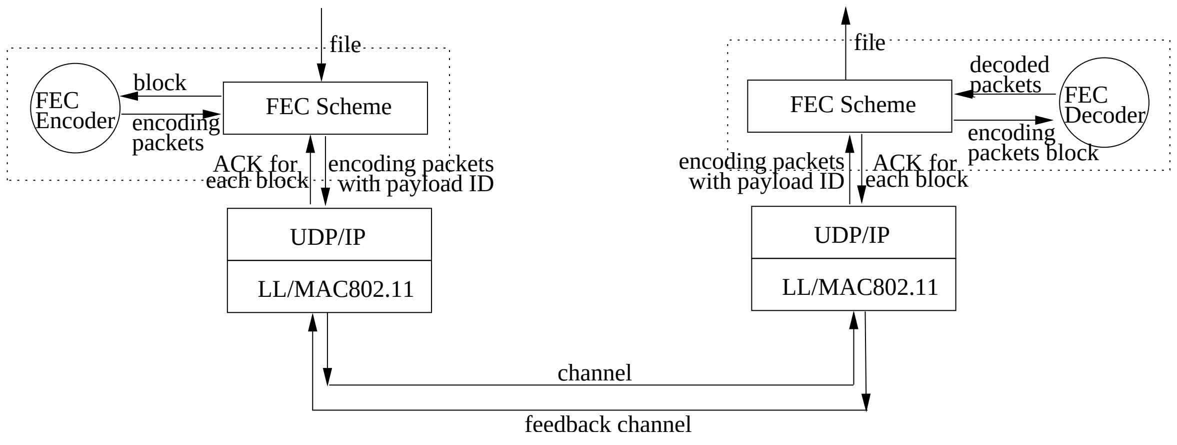


Figure 3.1: The architecture of the 802.11 WLAN equipped with an AL-FEC scheme.

In the following sections, an optimum scheme will be proposed for determining the packet length, the file partitioning method and feedback messages for Fountain codes, based on the MAC layer's retransmission mechanism and on the PHY layer's Symbol Error Ratio (SER). Section 3.2 investigates the maximum transmission efficiency of Fountain code aided file transfer. This is realized by transmitting FEC-encoded packets having an optimum packet size. More explicitly, Section 3.2.1 formulates the packet loss ratio at the application layer with respect to the packet size. Then, Section 3.2.2 derives the expression of the network load imposed by a single packet, parameterized by the packet size. Based on these two equations, Section 3.2.3 determines the optimum packet size. Simulation results are provided in Section 3.3 for characterizing the maximum transmission efficiency achieved. Finally, Section 3.4 concludes this chapter.

3.2 Fountain Coding Packet Size

In this section, we elaborate on the Fountain codes invoked in the application layer of 802.11 wireless LANs for file delivery and design an efficient cross-layer operation aided FEC scheme. As Figure 3.2 shows, after deciding the packet length L_{data} using the technique to be outlined later in this section, the FEC process begins by partitioning the L_{file}-byte source file into Z number of source blocks. Annex C in the Content Delivery Protocol description of the DVB standard [68] recommends a source-partitioning method, which uses the same number of source packets in all source blocks. We propose a number of justified modifications, since the maximum block size is based on the memory available in the decoder. The file is decomposed into $T = \left\lceil \frac{L_{file}}{L_{data}} \right\rceil$ source packets, which are then mapped to the Z number of source blocks as seen in Figure 3.2.

According to the algorithm introduced in Section 2.3.1, each of the Fountain-encoded packets seen in Figure 3.2 is generated by the modulo-2 addition of randomly chosen source packets of a specific source block. The number of source packets that

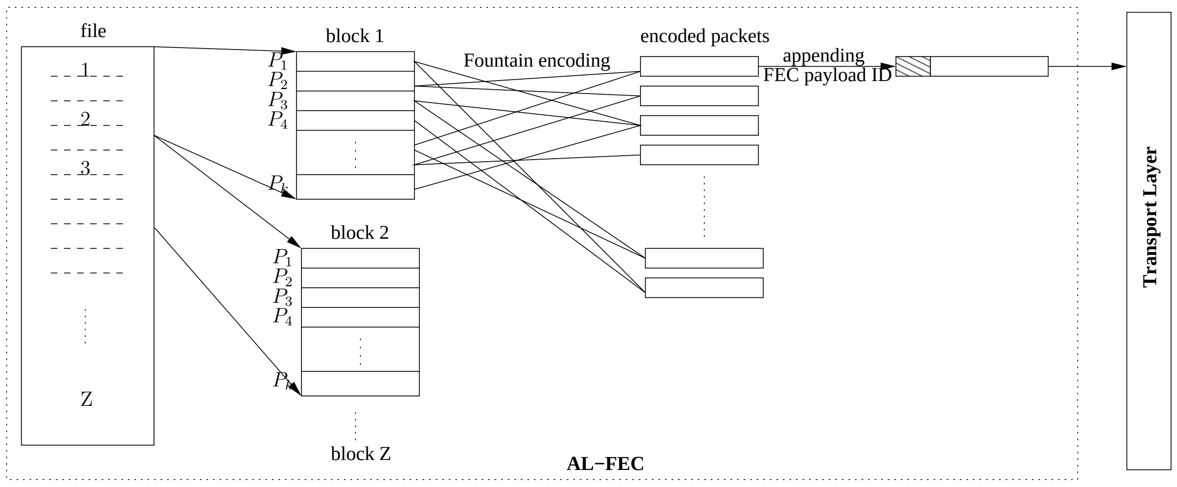


Figure 3.2: AL-FEC packetization using Fountain codes.

contribute to a specific Fountain-encoded packet is referred to as the 'degree' of the encoded packet, which is a random integer depending on the specific degree distribution employed [54], where the average degree of all encoded packets determines the complexity of the encoder and decoder. Finally, the Fountain-encoded packets of Figure 3.2 are given an FEC payload ID, which includes a Source Block Index (SBI) and the seed for the pseudo-random number generator, which will be used to generate the same random sequence in the decoder.

The transmitter continuously transmits the Fountain-encoded packets of a particular block, until an acknowledgement flag's reception indicates that the destination has successfully reconstructed the block. However, in poor channels, this feedback message may get lost. In order to circumvent this problem, the decoder may opt for transmitting the feedback flag continuously, as long as it continues to receive redundant Fountain-encoded packets having the same SBI as the reconstructed block. If the source transmitter consistently fails to receive the ACK message, after a timeout it will cease its transmission attempts, when the number of transmitted packets reaches the limit of $N_{max} = (1 + \alpha)N$, where $\alpha > 0$ is a parameter of the file transfer scheme and N is the number of encoded packets. This mechanism is capable of preventing the over loading of the network, when the PLR is excessive.

Our design objective is to maximize the average transmission efficiency required to successfully convey each block by carefully selecting the FEC parameters. Considering the PLR P_{lost} in the application layer, the transmission efficiency may be simply defined as

$$R = (1 - P_{lost}) \cdot \frac{L_{data}}{D} \ , \tag{3.1}$$

where D is the total load imposed by each packet on the network, which is expressed in bytes. Therefore $\frac{L_{data}}{D}$ may also be referred to as the normalized channel utilization ratio.

A Fountain-encoded packet will be extended by attaching the redundant header of the Fountain code, a User Datagram Protocol (UDP) header, an Internet Protocol (IP) header, a MAC and a Physical Layer Convergence Procedure (PLCP) header in sequence. Therefore, the size of the transmitted packet is the sum of L_{data} and the total size L_h of all the headers. Intuitively, long packets are more likely to be rejected by the 802.11 Cyclic Redundancy Check (CRC), since they comprise more bits which may become corrupted. In this event, the Fountain encoder will be required to send additional packets. On the other hand potentially, less packets are required for transmitting a given file, if large packets are employed, reducing the overhead associated with the various headers. Hence our forthcoming discussions will address striking an attractive tradeoff.

3.2.1 Packet Loss Ratio

Again, Fountain codes are intended for use in erasure channels, where packets are either correctly received or lost. These packet loss events may be imposed in 802.11 WLANs for example owing to routing queue overflow, node mobility, co-channel interference, fading etc. In this chapter, we only consider the effects of wireless propagation induced symbol errors, assuming that no other impairments inflict packet loss events, but its extension to other effects may be modeled without any significant difficulty.

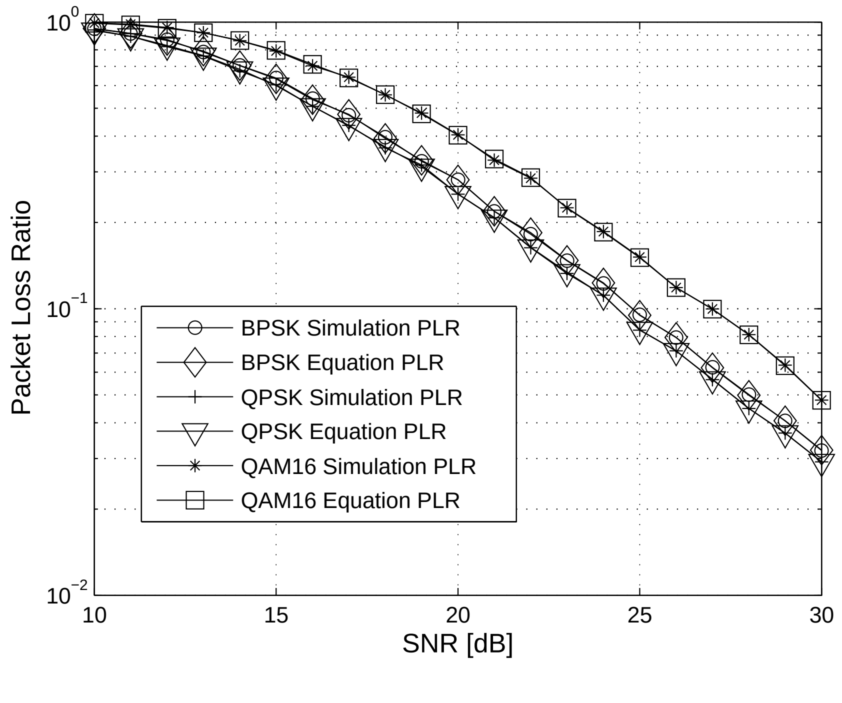


Figure 3.3: The PLR vesus SNR at the PHY layer using the parameters of Table 3.3.

When a Fountain-coded packet is transmitted over the channel, it will be modulated using an M-ary symbol alphabet, where we have $M = 2$ in Binary Phase-Shift Keying (BPSK), $M = 4$ in Quadrature Phase-Shift Keying (QPSK) and so on. The packet will be discarded, if residual symbol errors are detected by the CRC of 802.11. Assuming that all symbol errors are independent events, the SER can be used to derive the PLR associated with an uncorrelated Rayleigh fading channel, where the SER P_s of a certain modulation scheme is a decreasing function of the Signal Noise Ratio (SNR). Indeed the PLR $P_p(L)$ can be expressed as

$$P_p(L) \quad = \quad 1 - (1 - P_s)^{\frac{L \cdot 8}{log_2(M)}} \ , \tag{3.2}$$

where L is the length of the packet in bytes. In order to verify this relationship, we transmit a sufficiently high number of packets having the length of 16 Bytes over an uncorrelated Rayleigh fading channel for different modulation schemes, and record the PLR at the PHY layer. Figure 3.3 based on the parameters of Table 3.3 demonstrates that the simulation results closely match the theoretical PLR calculated by Equation 3.2, where the SER denoted by P_s was statistically calculated as the ratio of the number of successfully received symbols over the total number of transmitted symbols.

Table 3.3: The parameters used in the simulation for recording the PLR at the PHY layer.

Packet Length	16 Bytes
Modulation Scheme	BPSK, QPSK, QAM16
Channel Type	uncorrelated Rayleigh fading

However, deriving the PLR expression of the application layer is less straightforward, because the 802.11 MAC adopts Carrier Sense Multiple Access combined with Collision Avoidance (CSMA/CA) [86], which is further supported by ARQ for the sake of improved peer-to-peer reliability. In detail, the 802.11 MAC standard defines two mechanisms for accessing the medium. One of them is referred to as the Point Coordination Function (PCF), which is suitable for the centralized WLANs, where a bases station controls the communication among all nodes. The other is the so-called Distributed Coordination Function (DCF) [85], which is appropriate for Ad-hoc WLANs, where the nodes can communicate directly with each other. In this chapter, we only consider the basic DCF mechanism of the 802.11 MAC protocol.

DCF provides two techniques for transmitting packets [96, 97]. Firstly, the basic access method only includes data and ACK interaction. By contrast, the more sophisticated four-way handshaking procedures of 802.11 uses ‘Request To Send’ (RTS) and

'Clear To Send' (CTS) signals to test, whether the channel is sufficiently uninterfered before transmitting its data. In this section, we consider the more sophisticated four-way handshaking mode, but our analysis may be readily simplified for the case, where no RTS and CTS signals are employed.

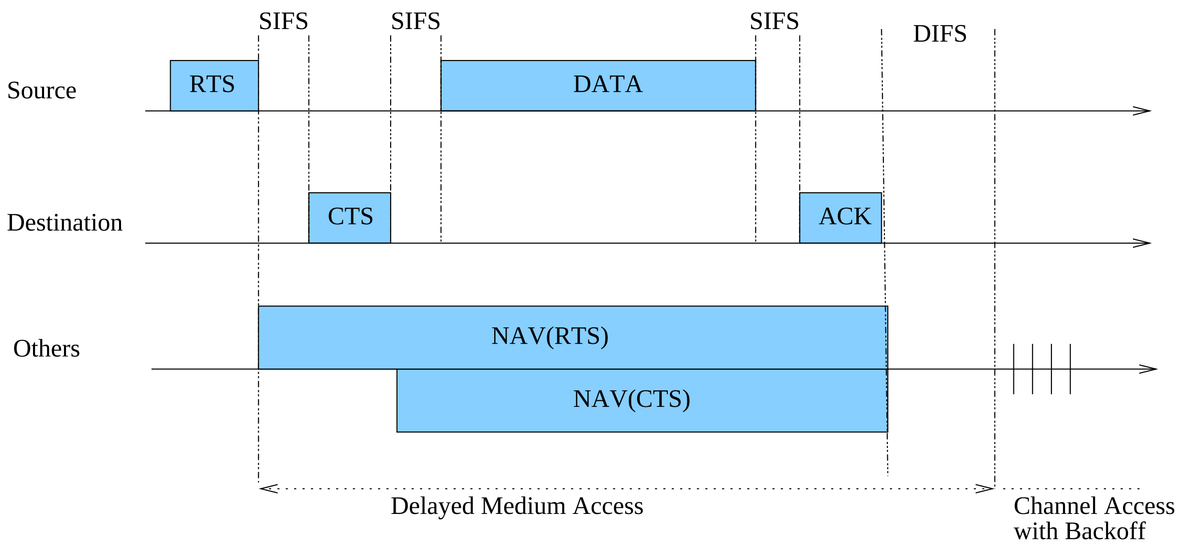


Figure 3.4: Four-way handshaking procedures of 802.11

In the four-way handshaking mechanism illustrated in Figure 3.4, when a node has a packet to transmit, it will wait for the medium to become idle. Next, it will send an RTS message to make a request and it will wait for the corresponding CTS confirmation from its intended destination. If no CTS packet is received after the window of Short Interframe Space (SIFS) owing to a packet collision or bit errors, more RTS attempts are made, until the so-called short retry limit R_1 is reached. A packet collision may occur if other nodes happen to send their RTS messages to request the medium simultaneously. Before the transmitter retransmits its RTS message, it will back off by a certain period of time, which is randomly selected in an interval between 1 and the size of the so-called contention window. The initial size of the contention window is defined by the different PHY specifications of the standard, for example it is 15 slot-durations for the Frequency Hopping (FH) PHY and 31 slot-times for the Direct Sequence (DS) PHY. As the retransmissions continue, the contention window size will be doubled every time, until a maximum value of 1023 slot-durations is reached.

After a successful RTS/CTS exchange, the source node transmits its data packet and waits for the corresponding ACK message. If no ACK is received after a window constituted by the sum of the data packet propagation delay plus SIFS seen in Figure 3.4, the data packet must be retransmitted, which requires a new RTS/CTS exchange. Likewise, the number of data packet retransmissions is constrained by the so-called long retry limit R_2. The packet will be discarded all together, if either the short or the long retry limits of R_1 and R_2 are reached. The flow-chart of the 802.11 retransmission mechanism is shown in Figure 3.5.

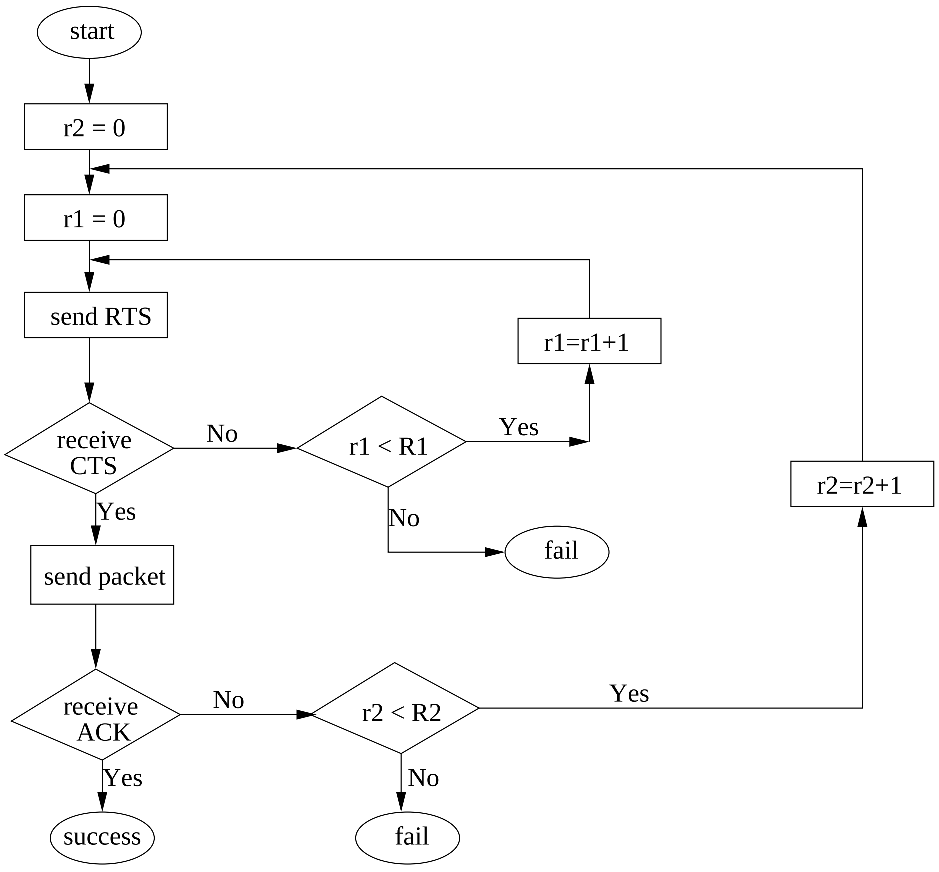


Figure 3.5: The flow chart of the 802.11 retransmission protocol.

During data transmission, no collisions will happen again, since the successful RTS/CTS exchange silences other nodes within the 'adequate-reception area' around the transmitter and receiver. More explicitly, the RTS and CTS packets carry a Network Allocation Vector (NAV), as seen in Figure 3.4, which contains the information specifying the period of time for which the channel will be occupied by the transmitter and receiver. Based on this NAV, the RTS/CTS pair may help us to avoid the problem of so-called 'hidden terminals', which is illustrated by Figure 3.6, where node 'C' and node 'A' are hidden from each other. Still considering Figure 3.6, when node 'B' is sending back a CTS message to node 'C', node 'A' also receives the CTS message and hence knows that the channel is busy for a period of time indicated by NAV. Hence node 'A' will not commence its transmission again.

Additionally, the DCF's four-way handshaking mechanism may improve the attainable system performance for transmission of long data packets in 802.11 WLANs, since the RTS and CTS packets are short and they may have a quick response for the sake of avoiding collisions. Otherwise, the collision would only be detected after a long

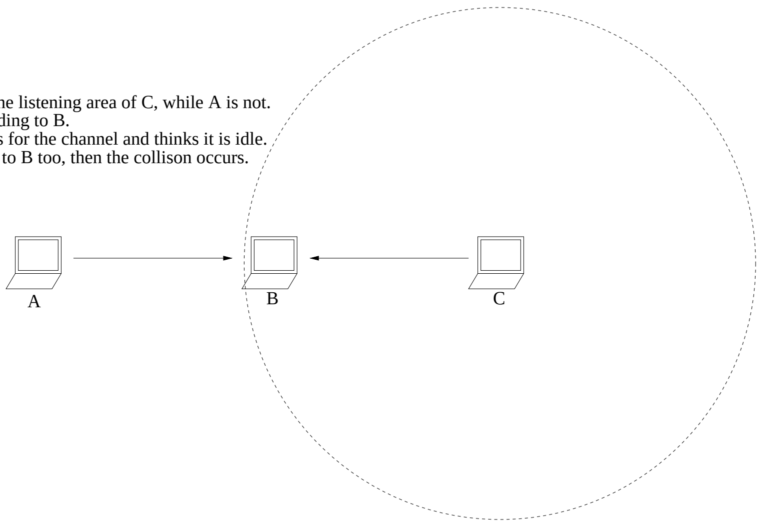

B is in the listening area of C, while A is not.
C is sending to B.
A listens for the channel and thinks it is idle.
A sends to B too, then the collison occurs.
A
B
C

Figure 3.6: The illustration of the hidden terminal problem.

duration imposed by long packets. This increases the bandwidth demand, when long data packets are transmitted.

Based on the above rudimentary introduction to DCF and on Equation 3.2, the probability of an RTS/CTS exchange failure may be expressed as:

$$P_r = [P_p(L_{rts} + L_{cts})]^{R_1} \ , \tag{3.3}$$

where L_{rts} and L_{cts} are the RTS and CTS packet length expressed in bytes, respectively. As mentioned above, R_1 represents the retry limit of the RTS/CTS exchange.

In the forthcoming analysis, we do not have to consider ACK failures, since they do not prevent the packet from being successfully received. The i*th* transmission[1] of the data packet will be required, when its first $(i-1)$ transmissions fail but the RTS/CTS exchanges in all of these $(i-1)$ transmissions were successful[2]. The probability of this event is given by

$$P_{f_i} = [(1 - P_r) \cdot P_p(L_{data} + L_h)]^i \ , \tag{3.4}$$

where $(L_{data} + L_h)$ is the aforementioned data packet size expressed in bytes. Therefore, the total PLR experienced at the application layer is constituted by the sum of the probabilities of RTS/CTS exchange failures in any round and the data packet failure

[1]This is actually a i*th* retransmission for the same packet in the 802.11 MAC layer and the value of it is confined by the retry limit R_2.

[2]If the RTS/CTS exchange fails in any round, the transmission of the packet is terminated immediately and the packet is then discarded. In this case, another transmission will not happen.

during the last retransmission, which can be expressed as

$$P_{lost} = \sum_{i=0}^{R2-1} P_{f_i} \cdot P_r + P_{f_{(R2)}} \ , \tag{3.5}$$

where $(P_{f_i} \cdot P_r)$ denotes the packet loss probability during the ith transmission attempt imposed by the RTS/CTS exchange failure. Note that the data failure during the ith transmission attempt will not cause a packet loss, if i does not reach the limit R_2. However, the data packet failure during the last transmission must be counted, hence $P_{f_{(R2)}}$ is embodied in Equation 3.5. Finally, for the fixed SER of P_s, and for the fixed values of L_{rts}, L_{cts}, L_h, R_1 and R_2, the packet loss probability P_{lost} is an increasing function of L_{data}.

3.2.2 Retransmission Cost

In an 802.11 WLAN, each packet may be retransmitted and acknowledged several times. Furthermore, each transmission may invoke several RTS/CTS exchanges. Therefore the successful delivery of each packet may impose significantly longer channel occupancy than their duration. The average channel occupancy associated with each RTS/CTS signal is given by

$$D_r = L_{rts} + [1 - P_p(L_{rts})] \cdot L_{cts} \ , \tag{3.6}$$

where $[1 - P_p(L_{rts})]$ quantifies the probability that the RTS packet is received by the destination, therefore triggering the transmission of a CTS signal. Hence, the total RTS/CTS related extra channel occupancy D_R associated with each transmission is given by

$$D_R = \sum_{i=0}^{R1} [P_p(L_{rts} + L_{cts})]^i \cdot D_r \ , \tag{3.7}$$

where $[P_p(L_{rts} + L_{cts})]^i$ indicates the probability that the ith transmission of the RTS/CTS signal will be required.

Whenever a data packet is successfully received, an ACK message will be generated. These ACK messages must be carefully considered, because they impose extra channel occupancy, hence extra interference and another retransmission related to the data packet will be performed, whenever the ACK is not successfully returned. The average channel occupancy imposed by each data packet's transmission is given by

$$\begin{aligned} D_d = & D_R + \\ & (1 - P_r) \cdot \{(L_{data} + L_h) + [1 - P_p(L_{data} + L_h)] \cdot L_{ack}\} \ , \end{aligned} \tag{3.8}$$

where L_{ack} denotes the number of bytes constituting the ACK message, $(L_{data} + L_h)$ is the data packet length in bytes, and $[1 - P_p(L_{data} + L_h)] \cdot L_{ack}$ represents the channel occupancy imposed by ACK based on the premise of a successful data packet reception.

Then, the probability that another retransmission will be attempted for this data packet is given by

$$P_{df_i} = [(1 - P_r) \cdot P_p((L_{data} + L_h) + L_{ack})]^i \ . \tag{3.9}$$

Finally, the total channel occupancy imposed by each data packet is composed of the sum of each retransmission's occupancy, which is given by

$$D = \sum_{i=0}^{R2} P_{df_i} \cdot D_d \ . \tag{3.10}$$

3.2.3 Maximum Transmission Efficiency

Theoretically, the average number of packets to be transmitted for the successful delivery of a single source packet, namely X may become high, if the PLR is high. However, in practice, only a limited latency can be tolerated and hence an upper limit $X_{max} = (1 + \alpha)$ is imposed on X, since each block has a timeout, as discussed before. Hence, the maximum tolerable PLR can be expressed as $P_{max} = \alpha/(1 + \alpha)$. Indeed, this confines the maximum data packet length $L_{data_{max}}$ according to Equation 3.5. On the other hand, the data packet length L_{data} must have a value of at least one byte, which hence determines the minimum value α_{min}.

Our objective is to select α and L_{data} in order to strike an attractive compromise, when transferring a file. Since L_{data} is required to be an integer between one and $L_{data_{max}}$ bytes, all legitimate values can be substituted into Equation 3.1 in order to determine the optimum packet size, which maximizes the transmission efficiency.

In the NS2 simulator, the total header size is 82 bytes for a data packet and the specific implementation of the 802.11 MAC employs a 44-byte RTS packet plus a 38-byte CTS and ACK packet. As a result, Figure 3.7 illustrates the optimal packet length versus SNR for these parameters and others parameterised by the retransmission limits and the value of α for QPSK modulation, when communicating over a non-dispersive

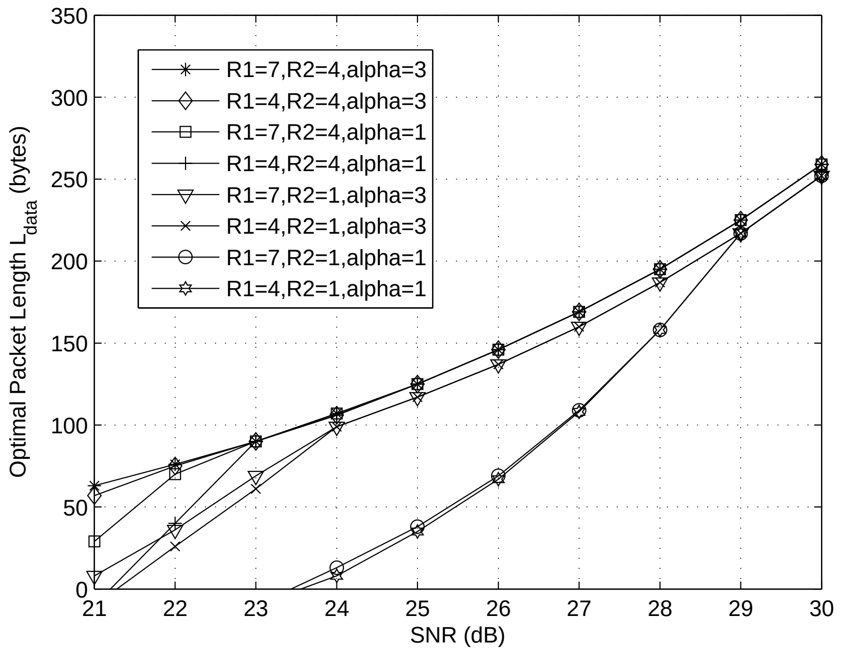


Figure 3.7: Optimal packet length in bytes versus the channel SNR for $L_h = 82$ bytes, $L_{rts} = 44$ bytes, $L_{cts} = 38$ bytes and $L_{ack} = 38$ bytes, when communicating over a non-dispersive uncorrelated Rayleigh fading channel.

uncorrelated Rayleigh channel. The SER P_s of QPSK was found for a Rayleigh channel in [98, Chapter 14.4.2] which is substituted into Equation 3.1, yielding Figure 3.7.

3.3 Simulation Results

In order to demonstrate the benefits of the Fountain-coded scheme, we simulated an 802.11 WLAN using the NS2 simulator for file transfer employing LT codes. Again, QPSK modulation was employed for transmission over an uncorrelated non-dispersive Rayleigh fading channel. The Destination Sequenced Distance Vector (DSDV) routing protocol was adopted. We considered the transmission of a medium sized file constituted by $201,582$ bytes and set the memory limit of the decoder to $102,400$ bytes. Each packet was forwarded at the application layer at 0.5s intervals. The value of α was set to $\alpha = 3$, since this value would tolerate a maximum PLR of 0.75. Finally, the retransmission parameters were set to the default values of the 802.11 MAC specification, namely to $R_1 = 7$, $R_2 = 4$. All above values of different network layers are summarized in Table 3.4.

Sufficiently long simulations were conducted in order to generate statistically relevant results. We chose four different packet lengths for each SNR, namely 16 bytes, the relevant optimal length minus 32 bytes, the optimal length and the optimal length plus 32 bytes for comparison of the attainable transmission efficiency. Firstly, the PLR

Table 3.4: Parameters used in the simulation.

Application Layer	
α	3
Transmission Interval	0.5s
Memory Limit	$102,400$bytes
File Size	$201,582$bytes
Transport Layer	UDP
Routing Protocol	DSDV
MAC	
R_1	7
R_2	4
PHY Modulation Scheme	QPSK
Channel	uncorrelated non-dispersive Rayleigh

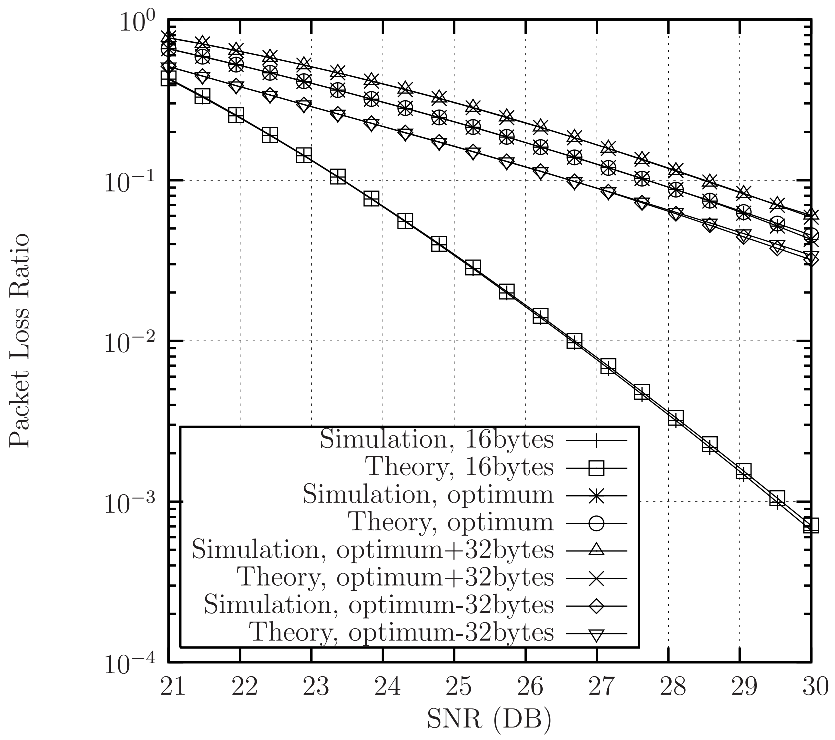


Figure 3.8: PLR versus SNR at the application layer. The theoretical results were evaluated from Equation 3.5.

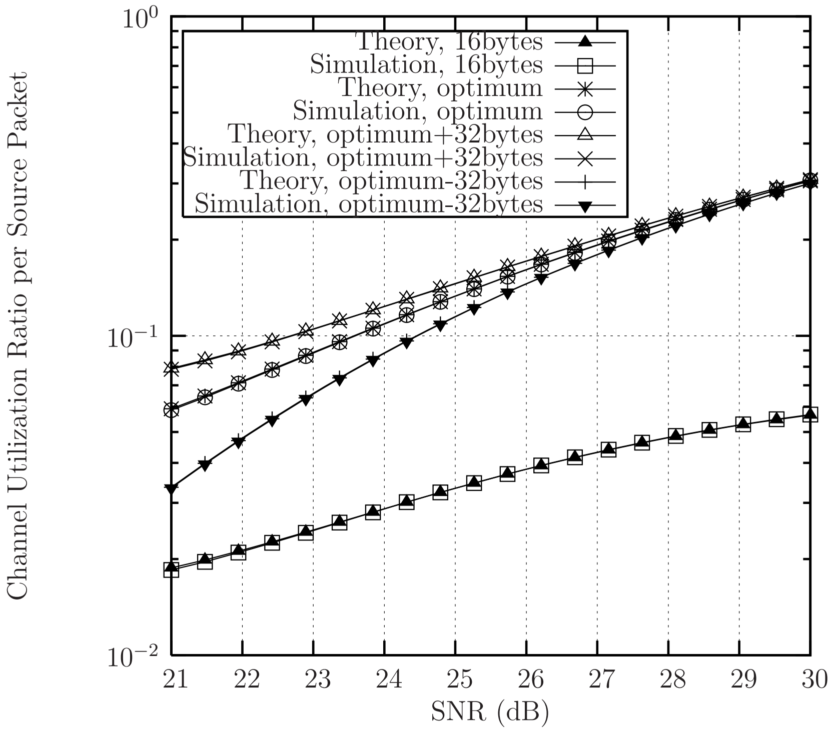


Figure 3.9: Channel utilization ratio per source packet versus SNR for transmission over an non-dispersive uncorrelated Rayleigh channel. The theoretical results were evaluated from Equation 3.10.

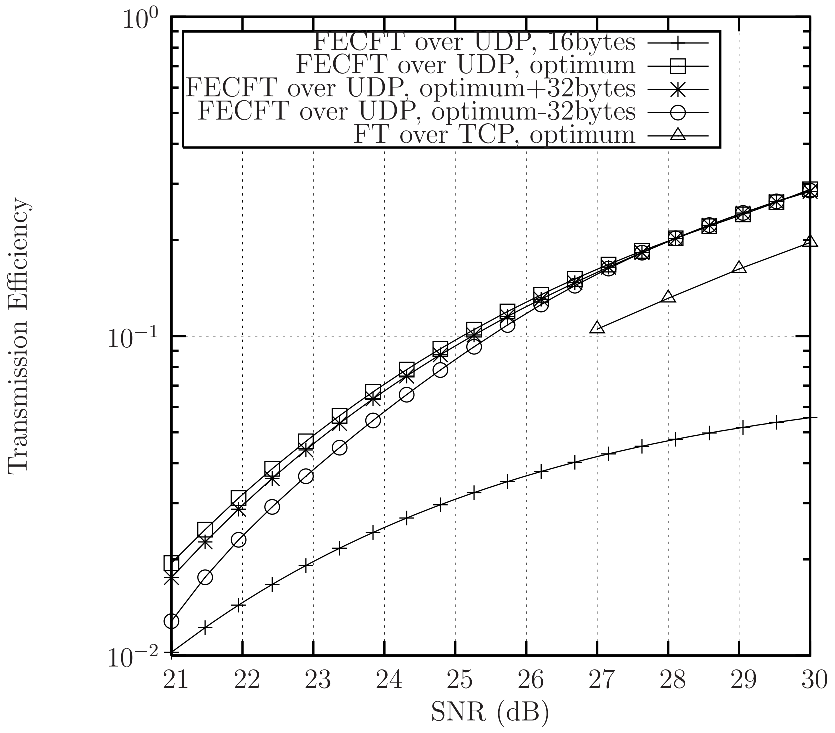


Figure 3.10: Transmission Efficiency

versus the SNR recorded at the application layer was illustrated in Figure 3.8, which verifies the accuracy of Equation 3.5 with the aid of Monte-Carlo simulations.

Similarly, Figure 3.9 shows the relationship between the theoretical and Monte-Carlo simulation results quantified in terms of the channel occupancy ratio, which verifies the accuracy of Equation 3.10.

Finally, in Figure 3.10, the acronym 'FEC-FT over UDP' represents the file transfer using LT codes with the aid of the UDP transport layer protocol. The curve corresponding to the optimum packet length with 'FEC-FT over UDP' is always at the top, confirming that the FEC scheme has the highest transmission efficiency. The simulation results recorded for file transfer using the Transmission Control Protocol (TCP) (i.e., 'FT over TCP') are also provided in Figure 3.10. The transmission efficiency of 'FT over TCP' is zero, when the SNR is below 27dB, since the file transfer cannot succeed at all within a timeout of 5000s, which is sufficiently long for 'FEC-FT over UDP' to successfully complete the file transfer.

3.4 Chapter Summary

In conclusion of this chapter, from the application layer's perspective, the retransmission and ACK mechanism of the 802.11 MAC is capable of reducing the PLR, but it imposes a substantial overhead. The appropriate choice of the packet length in the application layer indeed reduces the associated overhead. Fountain codes improve the successful file transfer probability even in hostile channel conditions. Compared to the file transfer regime operating without FEC over the classic TCP protocol, the proposed regime requires a lower threshold SNR for accomplishing a successful file transfer and hence enhances the transmission efficiency by about 50%.

Furthermore, the Fountain code aided AL-FEC scheme proposed in this chapter relies on cross-layer operation, namely across the application layer, the MAC layer and the PHY layer. More explicitly, the determination of the optimum packet length for the Fountain codes in the application layer depends on the MAC layer's retransmission properties and on the PHY layer's SER. This cross-layer operation-aided scheme has been verified to achieve the maximum transmission efficiency, when it is employed to aid file transfer in 802.11 WLANs.

3.4.1 Is the PHY-layer channel coding needed?

Even when Fountain codes are employed, Figure 3.8 still exhibits a high PLR at the application layer for the end-to-end transmission, since packets may get lost at any link during the transmission. The 802.11 retransmission mechanism is an efficient method conceived for ensuring reliable transmissions in peer-to-peer links. However again, Figure 3.8 indicates that the results are not satisfactory. This encouraged us to apply channel codes in the PHY layer to enhance the peer-to-peer transmission quality. Maximum-minimum-distance block codes - namely BCH codes were selected as an initial attempt to check as to whether and how channel coding may improve the attainable PLR performance.

When applying BCH codes in the 802.11 WLAN scenario considered, the PLR recorded at the application layer decreases significantly. Figure 3.11 shows the PLR performance, when employing the BCH$(32, 21, 5)$ code relying on low-complexity hard decisions in the PHY layer, where $d_{min} = 5$ implies that the code can correct $t = 2$ random errors in a 32-bit packet.

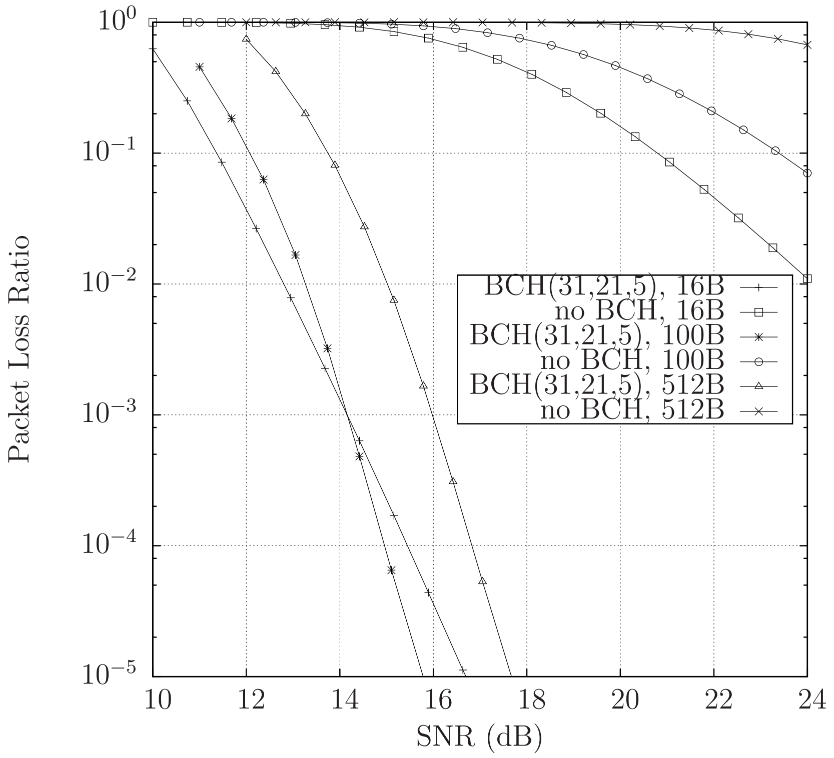


Figure 3.11: Packet loss ratio at the application layer over a non-dispersive uncorrelated Rayleigh fading channel using BCH(31,21,5) and BPSK modulation.

The results of Figure 3.11 confirm that channel codes indeed constitute beneficial methods of improving the achievable system performance. In Chapters 4 and 5 we will investigate, which particular channel codes constitute beneficial choices and how they may be efficiently integrated into our cross-layer design.

Chapter 4

Multiple Component Turbo Code Aided HARQ

As demonstrated at the end of Chapter 3, even simple channel codes have exhibited an attractive capability of improving the attainable Packet Loss Ratio (PLR) performance. When Forward Error Correction (FEC) codes are employed as PHYsical-layer FEC (PHY-FEC), the structure of Figure 3.1 will be extended to what is seen in Figure 4.1. In this chapter, we focus our attention on the PHY-FEC design and integrate it into the entire system in an efficient way.

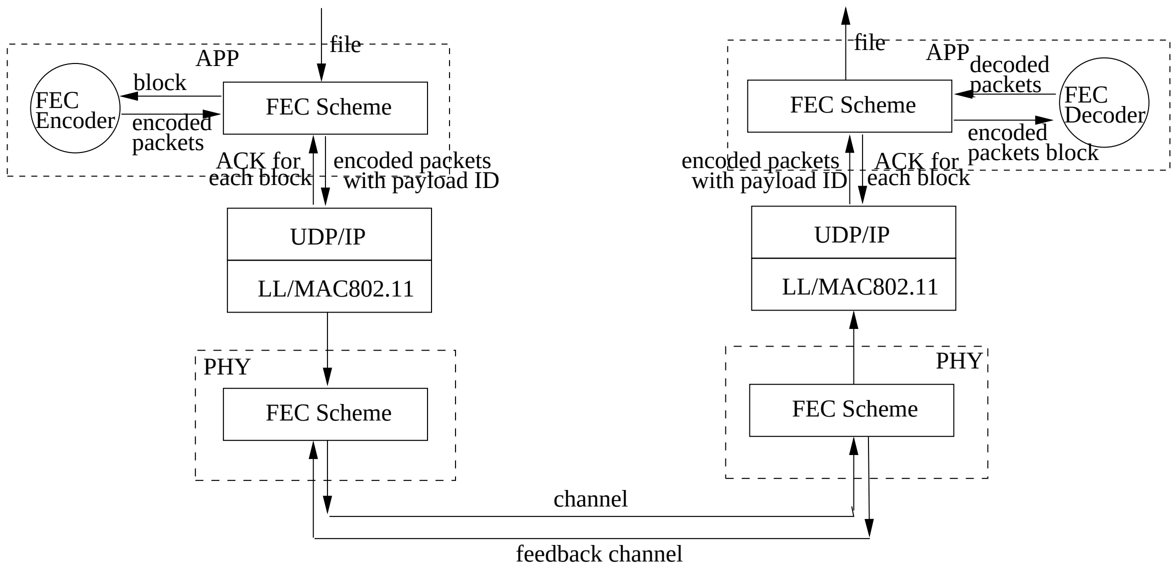


Figure 4.1: The system architecture of Figure 3.1 is extended with the aid of a PHY-FEC scheme.

More explicitly, near-capacity turbo codes are chosen for protecting the data transmission on each link. However, instead of classic Twin-Component Turbo Codes (TCTCs), Multiple Component Turbo Codes (MCTCs) will be recommended in Section 4.1 of this chapter, since the area properties of Extrinsic Information Transfer

(EXIT) charts detailed in [99] reveal that TCTCs suffer from an unavoidable capacity loss, while MCTCs may not. The motivation of MCTCs is detailed in Section 4.1.1. Then, in Section 4.1.2, we present the encoder and decoder structures of MCTCs, and extend EXIT charts to analyze N-component turbo codes in Section 4.1.3. Simulations are developed in Section 4.1.4 to compare the achievable Bit Error Ratio (BER) performance of MCTCs to those of systematic and non-systematic TCTCs. Based on this analysis, we further combine MCTCs, as well as systematic and non-systematic TCTCs with Hybrid Automate Request reQuest (HARQ) in Section 4.2 to evaluate their PLR and throughput in the light of their complexities. Finally, Section 4.3 concludes this chapter.

4.1 Multiple Component Turbo Codes

MCTCs are constituted by the parallel concatenation of more than two component codes. The concept of MCTCs was proposed by Divsalar as early as 1995 [100, 101], where the author introduced the encoder and decoder structures, analyzed the effects of different interleavers and characterized the achievable BER performance. Since then, researchers have proposed further beneficial designs [102, 103]. They mainly used the traditional metric of BER versus Signal Noise Ratio (SNR) and showed that MCTCs may achieve a better performance than conventional TCTCs. Hence, MCTCs have been widely employed in applications ranging from HARQ schemes [23] to cooperative relaying networks [104]. Table 4.1 highlights the literature, which reflects the development of MCTCs.

Our MCTCs have similar encoder and decoder structures to previous proposals, but exhibit the following novelty:

- Based on the area properties of EXIT charts, we will demonstrate that MCTCs are capable of overcoming the capacity loss that classic TCTCs exhibit.
- We employed low-order polynomials as spline functions to model our EXIT functions. This simplifies the process of drawing the MCTCs' N-dimensional EXIT charts, which illustrates the extrinsic information exchange between a composite decoder having $(N-1)$ components and an individual decoder.
- We introduce the concept of 'open tunnel SNR threshold', which directly quantifies the MCTCs' achievable performance in contrast to the Discrete-input Continuous-output Memoryless Channel's (DCMC) capacity.

4.1.1 Motivation

For avoiding obfuscating complications, Binary Phase-Shift Keying (BPSK) modulation is employed in the following investigation. According to the area properties of EXIT charts [113, Section VIII], for Berrou's classic $N = 2$-component turbo codes,

Table 4.1: Major contributions addressing MCTCs.

Author(s)	Contribution
Divsalar *et al.* 1995 [100] [101]	introduced multiple turbo codes and a suitable decoder structure, focusing on the design of constituent encoders and random interleavers between them.
Berrou *et al.* 1999 [105]	proposed multidimensional turbo codes based on circular Recursive Systematic Convolutional (RSC) codes, which have the same initial and final trellis states.
Huettinger *et al.* 2002 [106]	designed three-component turbo codes with the aid of extrinsic information transfer charts and information combining of component codes.
Berrou 2003 [107]	summarized the ten-year development of turbo codes and presented the structural diagram of multiple parallel concatenated circular RSC codes.
Zhao *et al.* 2003 [108]	investigated distributed turbo code aided diversity schemes conceived for relay channel, and mentioned their extension to multiple relay based arrangements using multiple turbo codes.
Gnaedig *et al.* 2005 [109]	designed multiple sliced turbo codes, where each information frame may be divided into several slices. The paper focused on the design of interleavers conceived for facilitating the parallel encoding or decoding of slices.
Brannstrom *et al.* 2005 [110]	analyzed the convergence behaviour of multiple turbo codes based on EXIT charts and proposed a method of finding an optimal activation order for multiple component codes.
Hausl *et al.* 2006 [111]	proposed to use three-component turbo codes in a relay network, where each component code is used at the mobile, base station and the relay respectively.
He *et al.* 2006 [112]	studied the problem of joint interleaver design in the context of multiple turbo codes and based on the properties of interleavers characterized the minimum distance of turbo codes.

the area A under the inner component decoder's EXIT curve is related to both the DCMC's capacity C and to the component code's rate R_c, which may be expressed by the equation of $A \leq C/R_c$. On the other hand, an open EXIT chart tunnel emerges, if the two EXIT curves do not intersect before reaching the point $(1, 1)$ of perfect convergence to a vanishingly low BER at a given SNR. More explicitly, this implies that the code is capable of achieving a low BER at this SNR, as long as the interleavers' length is sufficiently high. Hence, the effective number of bits transmitted per channel use at the corresponding SNR is approximately equal to the current overall coding rate R, which may be given by $R = R_c/N$. For classic TCTCs having $N = 2$ and associated with an area of $A \geq 0.5$ under the decoder's EXIT curve implies that the EXIT functions may be reshaped to create an open tunnel, for example by choosing an appropriate generator polynomial for the component codes. This may be illustrated by Figure 4.2, where we can observe that in the second reshaped EXIT chart, the open tunnel area between the two EXIT curves implies a rate loss compared to C.

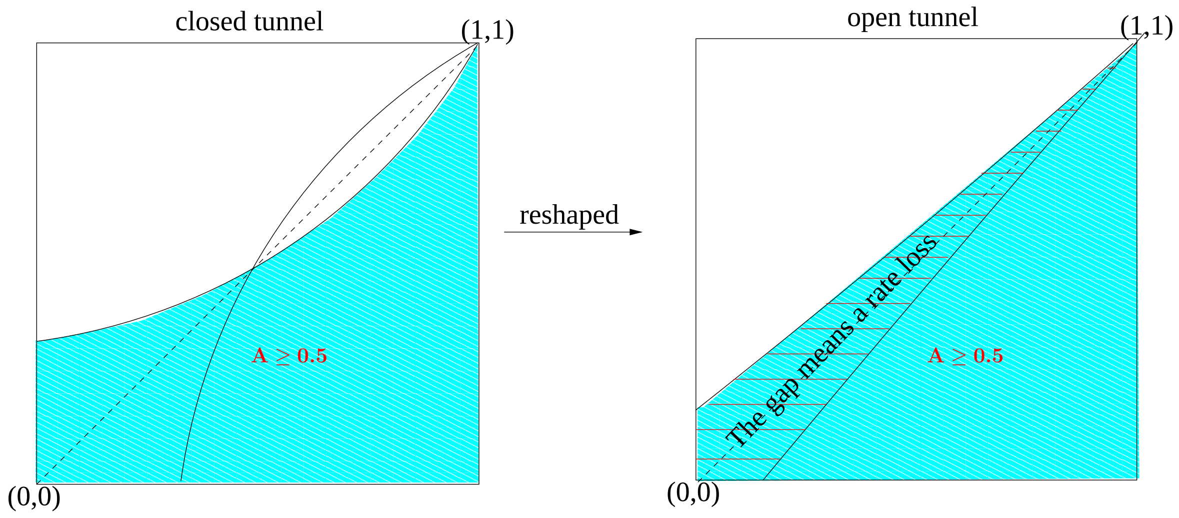


Figure 4.2: The original and reshaped EXIT charts of classic TCTCs having $N = 2$ and associated with an area of $A \geq 0.5$ under the decoder's EXIT curve.

Given $R_c = 1$, if the open EXIT-tunnel is eliminated by reshaping the EXIT curves, TCTCs may avoid any capacity loss, as Figure 4.3 shows. By contrast, for $R_c < 1$, we have $R_c \cdot A = C - L$, where L is a positive constant, representing a capacity loss. Then, we have: $A = \frac{C-L}{R_c}$. For the classic $N = 2$-component parallel concatenated turbo codes, the overall coding rate is $R = \frac{R_c}{2}$. Since $A \geq 0.5$ is required, we may deduce that $(C - L) \geq (0.5 \cdot R_c = R)$.

To summarize, the following two expressions hold in the case of BPSK modulation:

$$R \leq C \qquad (if\ R_c = 1)\ ; \tag{4.1}$$

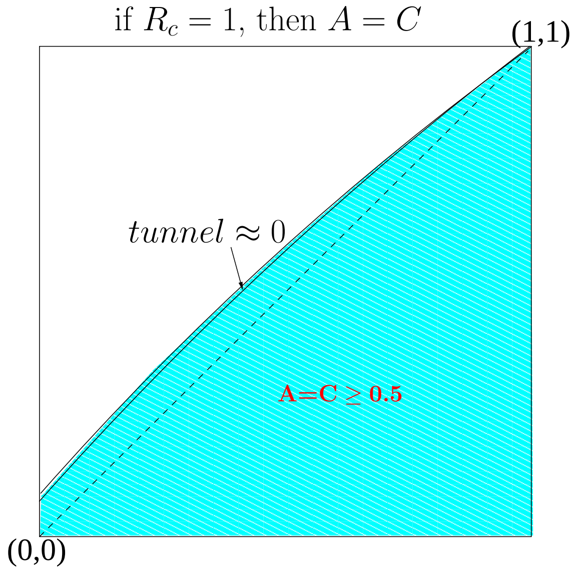


Figure 4.3: Avoiding a capacity loss if the open EXIT-tunnel of TCTCs is eliminated by reshaping the EXIT curves, given the component code's rate $R_c = 1$.

$$R \leq C - L \qquad (if\ R_c < 1)\ , \tag{4.2}$$

where the positive constant L depends on both the SNR and on the component codes' parameters, physically representing the capacity loss. Equation 4.2 demonstrates that a capacity loss is unavoidable for TCTCs, even if the equality is satisfied.

Since typically a strong code associated with a coding rate below 1/2 is required for transmission over hostile channels, classic TCTCs [47] may not constitute attractive choices due to the above-mentioned capacity loss. By contrast, MCTCs constituted by N Unity Rate Code (URC) [114] components are attractive turbo codes having a rate of $\frac{1}{N}$. Although the above-mentioned potential capacity loss may be overcome by MCTCs, it is necessary to confirm whether they are capable of approaching the DCMC capacity more closely than TCTCs at the same coding rate, which has not been resolved in the open literature, except that the existence of an SNR difference between the BER curves of the MCTCs found and the Shannon limit was mentioned by Huettinger and Huber [115].

Motivated by solving this open problem, in this section, we invoke 'the open tunnel SNR threshold' as our metric of estimating the asymptotic performance of MCTCs. EXIT charts [48] [99] play an important role in analyzing the convergence behaviour of classic TCTCs. We will demonstrate that the MCTC decoder may be viewed as being separated into two logical decoder components, each potentially hosting a group of component decoders, where the extrinsic information of all component decoders of a logical decoder will be combined to generate the overall extrinsic output of this logical decoder. This partitioning of the MCTC decoder into two logical parts is necessary, because otherwise we would need an N-dimensional EXIT chart for visualizing the extrinsic information exchange of the N component codes. As a benefit, the EXIT chart may be drawn based on the extrinsic information exchange between the two logical decoder parts and then the corresponding 'open tunnel SNR threshold' may be found for the MCTCs.

4.1.2 System Model

Our system model employs URCs as the component codes of the proposed MCTC. Figure 4.4 shows the encoder's structure, where the source information is $\mathbf{a}$ and its interleaved copies $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \cdots, \mathbf{a}_N\}$ are entered into the N encoders. The multiplexer seen at the output of the encoder in Figure 4.4 assembles the encoded bits $\mathbf{b}$, where the resultant bit stream has N times the original sequence length. Hence, the overall coding rate becomes $1/N$.

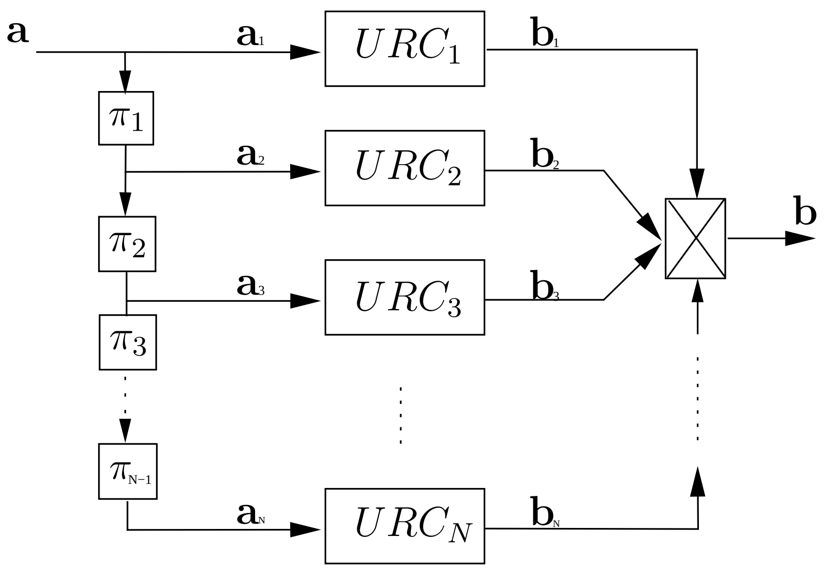


Figure 4.4: The encoder structure of a MCTC using N URC components.

Figure 4.5 shows the corresponding N Bahl, Cocke, Jelinek and Raviv (BCJR) decoders' structure, where each BCJR decoder has two inputs, namely the *a priori* LLRs $\tilde{\mathbf{a}}_i^a$ combined from all other decoders' extrinsic LLRs and the channel's output information $\tilde{\mathbf{b}}_i$. Then each BCJR decoder outputs its own extrinsic LLRs $\tilde{\mathbf{a}}_i^e$. The decoding process proceeds as follows. Each time a specific BCJR decoder is invoked. Although there have been numerous strategies recommended [116] to choose the decoders' activation order and to appropriately combine the other decoders' extrinsic LLRs, two further specific strategies will be introduced in Section 4.1.3.1 and 4.1.4. Decoding operations iterate by exchanging extrinsic information among all N BCJR decoders, until the point of convergence is reached or the affordable number of iterations was exhausted. Then, the recovered bits are decided upon, based on their *a posteriori* LLRs $\tilde{\mathbf{a}}_1^p$.

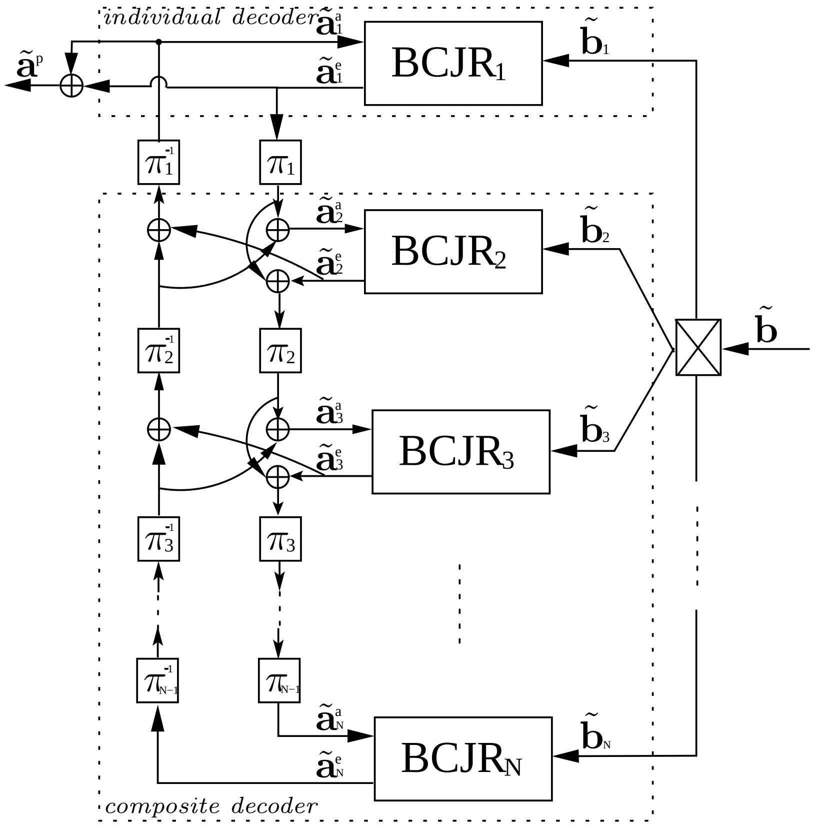


Figure 4.5: The decoder structure of a MCTC using N BCJR decoders. Its logical partitioning is showed in the dashed boxes. The corresponding encoder was portrayed in Figure 4.4.

4.1.3 EXIT Chart Analysis of Multiple-Component Turbo Codes

EXIT charts have become a proven technique of analyzing classic TCTCs [99]. In this section, we detail the above-mentioned logical MCTC partitioning method and demonstrate that EXIT chart analysis may also be beneficially applied to design MCTCs. The resultant EXIT charts - which we refer to as extended EXIT charts - directly reflect the convergence behaviour of the MCTCs.

4.1.3.1 Logical Partitioning of the MCTCs

For the sake of exploiting the advantages of EXIT charts, we partition the MCTC decoder of Figure 4.5 into two logical parts, each having a conventional EXIT curve in a conventional EXIT chart. Again, without this partitioning, we would be unable to use 2D EXIT charts to investigate the extrinsic information exchange of N components, because an N-dimensional EXIT chart would be required. The two logical parts are surrounded by the dashed rectangles in Figure 4.5. The MCTC decoder is then treated as the parallel concatenation of an individual URC-BCJR decoder and an $(N-1)$-component composite decoder. As in the EXIT chart analysis of a TCTC, the EXIT function of the individual decoder is expressed as $I(\tilde{\mathbf{a}}_1^e, \mathbf{a}) = \mathbf{T_{ind}}\left[I(\tilde{\mathbf{a}}_1^a, \mathbf{a})\right]$, interpreting the *a priori* Mutual Information (MI) $I(\tilde{\mathbf{a}}_1^a, \mathbf{a})$ as the abscissa and outputting the extrinsic MI $I(\tilde{\mathbf{a}}_1^e, \mathbf{a})$. By contrast, all the $(N-1)$ URC-BCJR decoders seen in the bottom part of Figure 4.5 are treated as a single amalgamated or composite decoder. The quantity $I(\tilde{\mathbf{a}}_1^e, \mathbf{a})$ output by the upper logical URC-BCJR decoder part of Figure 4.5 acts as the input *a priori* MI of the bottom part, while the combined extrinsic MI of the $(N-1)$-component amalgamated decoder seen in the bottom part of Figure 4.5 will be passed to the individual stand-alone decoder at the top of Figure 4.5 as the *a priori* MI $I(\tilde{\mathbf{a}}_1^a, \mathbf{a})$. The corresponding EXIT function therefore becomes $I(\tilde{\mathbf{a}}_1^a, \mathbf{a}) = \mathbf{T_{comp}}[I(\tilde{\mathbf{a}}_1^e, \mathbf{a})]$. Using this partitioning method, the beneficial attributes of EXIT charts may be essentially retained, despite analyzing N components in two dimensions instead of N dimensions. Hence, the area of the corresponding open EXIT tunnel can be used as per normal for characterizing the achievable performance of MCTC.

We emphasize here that the *a priori* LLRs are assumed to obey the Gaussian distribution, but their modeling accuracy can be improved using the experimentally evaluated LLR-histogram. The $(N-1)$ components of the composite decoder iteratively exchange their extrinsic information, until the MI improvements become marginal and hence their convergence is deemed to have been achieved. The resultant combined extrinsic MI is then fed to the upper individual stand-alone decoder as the current *a priori* MI.

4.1.3.2 Spline-fitting Based Drawing of EXIT Functions

A). Spline-fitting EXIT functions

As briefly alluded to above, instead of assuming that the LLRs are Gaussian distributed, often more accurate results may be generated by experimentally evaluating the LLR-histogram for drawing EXIT charts. However, this method has two potential disadvantages. It is sensitive to the interleaver's design, since it is specific for the particular interleaver used, hence potentially preventing the analysis of the asymptotically attainable performance. Moreover, the task becomes very time consuming, especially for N-component MCTCs, since the length of the interleavers should be as high as possible in order to reduce the correlation among the interleaved *a priori* and extrinsic LLRs. Hence, we propose a semi-analytical spline-fitting based method for mitigating the effects of both the interleaver's design and that of its length, which facilitates the creation of the extended EXIT charts of MCTCs. More specifically, each MCTC decoder's EXIT function is modeled using a set of low-degree polynomials generated by spline-fitting, which is expressed as:

$$I(\tilde{\mathbf{a}}_i^e, \mathbf{a}) = \mathbf{T}'_{\mathbf{ind}} \left[I(\tilde{\mathbf{a}}_i^a, \mathbf{a}) \right], \tag{4.3}$$

where $I(\tilde{\mathbf{a}}_i^a, \mathbf{a})$ is generated on the basis of combining all the other $(N-1)$ URC-BCJR decoders' extrinsic information, and we have $1 \leq i \leq N$ for N components. Note that if each component has the same polynomial and the same channel information, i.e. the same $I(\tilde{\mathbf{b}}_i, \mathbf{b}_i)$ for all i values, their EXIT functions are the same. Otherwise, each has its specific EXIT function. More explicitly, the fitted function $\mathbf{T}'_{\mathbf{ind}}$ mathematically consists of a set of spline-functions, which are formulated as follows,

$$\mathbf{T}'_{\mathbf{ind}} = \begin{cases} m_1 \cdot I(\tilde{\mathbf{a}}_i^{\mathrm{a}}, \mathbf{a}) + n_1 & (0.0 \leq I(\tilde{\mathbf{a}}_i^{\mathrm{a}}, \mathbf{a}) < 0.1) \\ m_2 \cdot I(\tilde{\mathbf{a}}_i^{\mathrm{a}}, \mathbf{a}) + n_2 & (0.1 \leq I(\tilde{\mathbf{a}}_i^{\mathrm{a}}, \mathbf{a}) < 0.2) \\ \qquad \vdots & \\ m_{10} \cdot I(\tilde{\mathbf{a}}_i^{\mathrm{a}}, \mathbf{a}) + n_{10} & (0.9 \leq I(\tilde{\mathbf{a}}_i^{\mathrm{a}}, \mathbf{a}) \leq 1.0), \end{cases} \tag{4.4}$$

where m_k, n_k $(k = 1, 2, \cdots, 10)$ can be obtained for each region, by linear fitting to the corresponding sections of the EXIT curve. Here, the EXIT curve is only generated once by the LLR-histogram experiment at a certain SNR.

Practically, the MI range $[0, 1]$ may be partitioned flexibly using a more accurate granularity, according to the specific shape of the EXIT curve, when relying on spline-fitting. For example, Figure 4.6 shows an LLR-histogram based experimental EXIT function and its spline-fitted version, which were recorded for the RSC generator polynomial of $(2, 3)_o$ and for the channel output mutual information of $I(\tilde{\mathbf{b}}_i, \mathbf{b}_i) = 0.4$. As observed in Figure 4.6, the specific abscissa-range $[0.9, 1]$ of the EXIT curve is quite steep. There may be a substantial difference between them, if the whole section is fitted using a 1st-order polynomial, which corresponds to a line. Therefore, our solution is to separate this range of $[0.9, 1]$ into 10 smaller sub-intervals. Furthermore, the granularity of 0.05 is adopted for the range of $[0, 0.9)$ for the sake of higher accuracy. Therefore, the curve represented by the spline-fitted EXIT function of Figure 4.6 nearly overlaps with that drawn based on the experimental LLR-histogram data. The parameterized values of m_k, n_k are listed in Table 4.2 for the spline-fitted EXIT function.

B). Information Combining

Our goal is then to find the appropriate expression for combining each of the $(N-1)$ URC-BCJR decoder's extrinsic information for the sake of generating $I(\tilde{\mathbf{a}}_i^a, \mathbf{a})$ for each individual decoder, one-by-one. As explained in [53], the extrinsic LLRs $\tilde{\mathbf{a}}_i^e$ of each URC-BCJR decoder of MCTCs may be modeled as the output of a virtual extrinsic

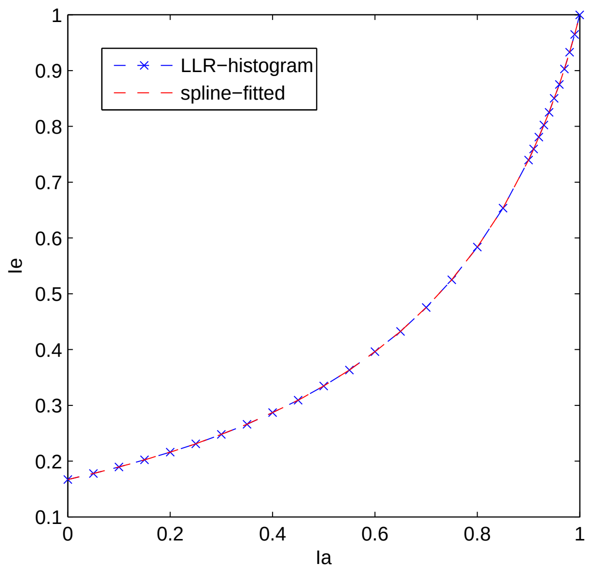


Figure 4.6: The LLR-histogram based EXIT curve and its spline-fitted approximation.

channel, as seen in Figure 4.7. Since the extrinsic LLRs will act as the *a priori* input entered into the other URC-BCJR decoders in the next stages, the virtual extrinsic channel is also referred to as the virtual *a priori* channel.

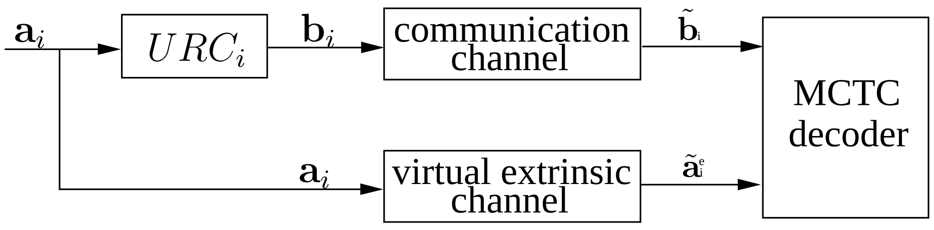


Figure 4.7: The illustration of the virtual extrinsic channel in turbo codes, which provides *a priori* information.

As we discussed before, the *a priori* LLRs of each URC-BCJR $\tilde{\mathbf{a}}_i^a$ may be obtained by adding the extrinsic LLRs $\tilde{\mathbf{a}}_j^e (j \neq i)$ from all other $(N-1)$ decoders for drawing EXIT charts based on the LLR-histogram experiments, since each LLR value has been generated following the LLR distribution for a specific extrinsic channel. However, the combination of the corresponding information $I(\tilde{\mathbf{a}}_j^e, \mathbf{a})$ becomes challenging. We use the terminology of '**combine**' to express the relationship between the *a priori* information $I(\tilde{\mathbf{a}}_i^a, \mathbf{a})$ and the $(N-1)$ extrinsic information components $I(\tilde{\mathbf{a}}_j^e, \mathbf{a})$, as follows:

$$I(\tilde{\mathbf{a}}_i^a, \mathbf{a}) = \textbf{combine}\left[I(\tilde{\mathbf{a}}_1^e, \mathbf{a}), I(\tilde{\mathbf{a}}_2^e, \mathbf{a}), \cdots, I(\tilde{\mathbf{a}}_N^e, \mathbf{a})|_{except} I(\tilde{\mathbf{a}}_i^e, \mathbf{a})\right], \tag{4.5}$$

where the concrete **combine** procedure depends on the type of the virtual extrinsic channel, which relies on the theory of information combining [117, 118].

Table 4.2: The coefficients of each polynomial of the example spline-fitted EXIT function.

MI region	m_k	n_k
[0, 0.05)	0.215337	0.167048
[0.05, 0.1)	0.232805	0.166174
[0.1, 0.15)	0.257639	0.163691
[0.15, 0.2)	0.275013	0.161085
[0.2, 0.25)	0.299558	0.156176
[0.25, 0.3)	0.339520	0.146185
[0.3, 0.35)	0.361140	0.139699
[0.35, 0.4)	0.417332	0.120032
[0.4, 0.45)	0.442507	0.109962
[0.45, 0.5)	0.511980	0.078699
[0.5, 0.55)	0.571029	0.049176
[0.55, 0.6)	0.657804	0.001448
[0.6, 0.65)	0.729220	-0.041401
[0.65, 0.7)	0.855534	-0.123505
[0.7, 0.75)	0.995660	-0.221593
[0.75, 0.8)	1.173320	-0.354838
[0.8, 0.85)	1.395716	-0.532755
[0.85, 0.9)	1.723711	-0.811551
[0.9, 0.91)	1.984597	-1.046348
[0.91, 0.92)	2.125600	-1.174661
[0.92, 0.93)	2.147085	-1.194427
[0.93, 0.94)	2.273273	-1.311782
[0.94, 0.95)	2.529253	-1.552403
[0.95, 0.96)	2.485549	-1.510884
[0.96, 0.97)	2.774523	-1.788299
[0.97, 0.98)	2.993200	-2.000416
[0.98, 0.99)	3.178800	-2.182304
[0.99, 1.0)	3.528953	-2.528955

A number of publications [53, 113, 119–121] modeled the virtual extrinsic channel as Binary Erasure Channels (BECs). In this case, Equation 4.5 is reformulated as:

$$I(\tilde{\mathbf{a}}_i^a, \mathbf{a}) = 1 - \prod_{j=1, j\neq i}^{N} \left[1 - I(\tilde{\mathbf{a}}_j^e, \mathbf{a})\right] \ . \tag{4.6}$$

However, in most cases the extrinsic channel is modeled as an Additive White Gaussian Noise (AWGN) channel. The authors of [48] provided two possible reasons for the validity of this assumption: a) the communication channel is a Gaussian channel model; b) sums many values involved in the extrinsic LLRs calculation, which leads to Gaussian-like distributions. Based on this assumed AWGN extrinsic (or *a priori*) channel, the *a priori* input may be expressed as follows,

$$\tilde{\mathbf{a}}^{\mathbf{a}} = \mu_A \cdot \mathbf{a} + \mathbf{n}_A \ , \tag{4.7}$$

where $\mathbf{a}$ denotes the source information bit sequence and $\tilde{\mathbf{a}}^{\mathbf{a}}$ indicates the *a priori* LLR sequence. Equation 4.7 may be interpreted by applying a Gaussian random variable $\mathbf{n}_A$ with a variance of σ^2 and a mean of zero to the information bits $\mathbf{a}$. Furthermore, we have $\mu_A = \frac{\sigma^2}{2}$ according to [48].

Furthermore, the probability density function (PDF) which reflects the transition probability of the extrinsic channel conditioned on the transmitted bits with realizations of -1 or $+1$ may be derived from Equation 4.7, yielding:

$$P(\tilde{a}|a = \pm 1) = \frac{e^{-((\tilde{a}-\frac{\sigma^2}{2}\cdot a)^2/2\sigma^2)}}{\sqrt{a\pi}\sigma} \ . \tag{4.8}$$

Mathematically, each $I(\tilde{\mathbf{a}}_i^a, \mathbf{a})$ may be calculated as the output information of an extrinsic AWGN channel, which has a closed form based on Equation 2.8 by replacing the channel transition probability with the above Equation 4.8. In this way, we may obtain a function for each $I(\tilde{\mathbf{a}}_i^a, \mathbf{a})$ with respect to the corresponding extrinsic channel's parameter σ, instead of each concrete LLR value of the sequence. It is referred to as the function $J(\sigma)$ and has the following form [122]:

$$J(\sigma) = 1 - \int_{-\infty}^{\infty} \frac{e^{-((\tilde{a}-\frac{\sigma^2}{2}\cdot a)^2/2\sigma^2)}}{\sqrt{a\pi}\sigma} \cdot \ln\left[1 + e^{-\tilde{a}}\right] d\tilde{a} \ . \tag{4.9}$$

The integral calculation of Equation 4.9 is challenging. In order to find a simpler solution, the fitting is employed to match the result of this integration. According to [122], the fitting may be divided into three sub-functions, corresponding to three consecutive intervals, as follows:

$$J(\sigma) = \begin{cases} -0.0421061\sigma^3 + 0.209252\sigma^2 - 0.00640081\sigma & (0 \le \sigma \le 1.6363) \\ 1 - e^{0.00181491\sigma^3 - 0.142675\sigma^2 - 0.0822054\sigma + 0.0549608} & (1.6363 < \sigma < 10) \\ 1 & (\sigma \ge 10) \ . \end{cases} \tag{4.10}$$

The inverse function of $\sigma = J^{-1}(I)$ may be expressed as [122]:

$$J^{-1}(I) \approx \begin{cases} 1.09542I^2 + 0.214217I + 2.33727\sqrt{I} & (0 \le I \le 0.3646) \\ -0.706692\ln\left[0.386013(1-I)\right] + 1.75017I & (0.3646 < I < 1) \ . \end{cases} \tag{4.11}$$

When combining the extrinsic information glean from $(N-1)$ URC-BCJR decoders, the sum of several independent Gaussian random variables still follows a Gaussian distribution with the variance of $\sum_{i=1}^{(N-1)} \sigma_i^2$. Furthermore, the above function $I = J(\sigma)$ of Equation 4.10 may be used for calculating the extrinsic information I according to the parameter σ of the virtual extrinsic channel, and its inverse function of $\sigma = J^{-1}(I)$ may be invoked for transforming the extrinsic information I into the corresponding σ. Therefore, the input *a priori* information $I(\tilde{\mathbf{a}}_i^a, \mathbf{a})$ entered into the i^{th} individual URC-BCJR decoder may be obtained using the following equation:

$$I(\tilde{\mathbf{a}}_i^{\mathrm{a}}, \mathbf{a}) = J\left(\sqrt{\sum_{j=1, j\neq i}^{N} \left[J^{-1}(I(\tilde{\mathbf{a}}_j^{\mathrm{e}}, \mathbf{a}))\right]^2}\right) . \tag{4.12}$$

Let us consider $i = 2$ as an example. The input *a priori* information $I(\tilde{\mathbf{a}}_2^a, \mathbf{a})$ of Figure 4.5 is given by $I(\tilde{\mathbf{a}}_2^a, \mathbf{a}) = J\left(\sqrt{\left[J^{-1}(I(\tilde{\mathbf{a}}_1^{\mathrm{e}}, \mathbf{a}))\right]^2 + \left[J^{-1}(I(\tilde{\mathbf{a}}_3^{\mathrm{e}}, \mathbf{a}))\right]^2 + \cdots + \left[J^{-1}(I(\tilde{\mathbf{a}}_N^{\mathrm{e}}, \mathbf{a}))\right]^2}\right)$.

C). Drawing EXIT Charts

As discussed in Section 4.1.3.1, the N-dimensional EXIT charts of MCTCs will be visualized by the extrinsic information exchange between an individual URC-BCJR decoder and a composite decoder having $(N-1)$ components. Using the iterative computation of the combined extrinsic information based on Equation 4.3 by taking into account the contributions of all the $(N-1)$ URC-BCJR decoders in the composite decoder of Figure 4.5, the composite EXIT function $I(\tilde{\mathbf{a}}_1^a, \mathbf{a}) = \mathbf{T}'_{\mathbf{comp}}\left[I(\tilde{\mathbf{a}}_1^e, \mathbf{a})\right]$ can be determined. Figure 4.8 shows the associated flow chart, where the notations indicating information are simplified as $I(\tilde{\mathbf{a}}_i^a)$ or $I(\tilde{\mathbf{a}}_i^e)$, and '**combine**' combines the corresponding information using Equation 4.12.

Figure 4.9 provides the resultant extended EXIT charts recorded for three different BPSK-modulated Rayleigh fading channel SNRs for MCTCs having rates of $R = \frac{1}{4}, \frac{1}{5}, \frac{1}{6}$ bits per channel use, when invoking component encoders employing only a single memory element. In order to view the increment of MI along with the number of components, the EXIT curves of MCTCs constructed from $N = 2$ components are shown for these three SNRs. Note that in the EXIT charts of Figure 4.9, the dashed lines show the EXIT curves that were generated using the above spline-fitted EXIT-function models. These can be seen to nearly overlap with the EXIT functions drawn in solid lines, which were obtained using the Gaussian LLR assumption and an interleaver length of $1,000,000$ bits. This demonstrates that our spline-fitting based EXIT chart modeling used for obtaining the composite EXIT function offers a desirable low-complexity alternative to the employment of the Gaussian LLR assumption.

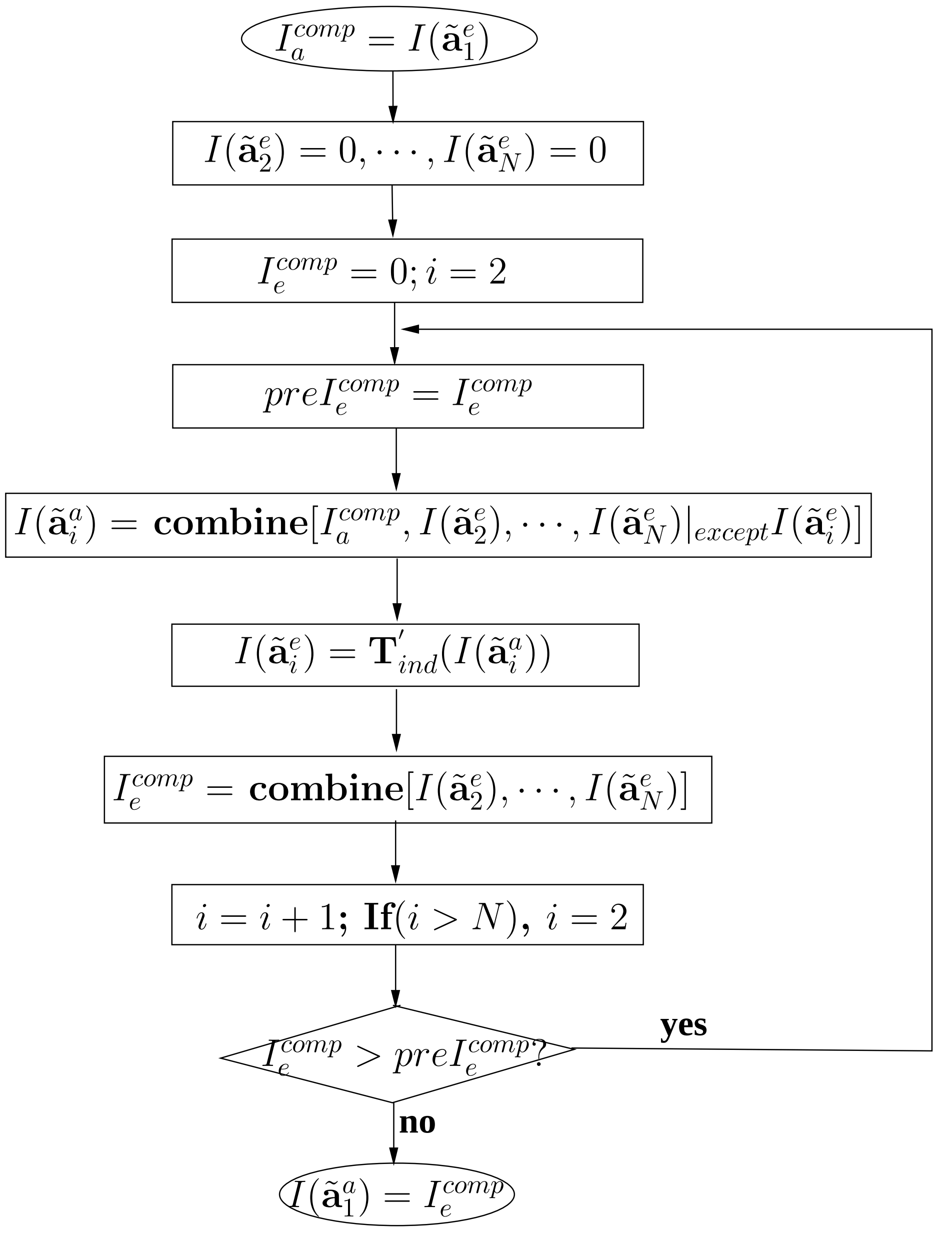

$I_a^{comp} = I(\tilde{\mathbf{a}}_1^e)$
$I(\tilde{\mathbf{a}}_2^e) = 0, \cdots, I(\tilde{\mathbf{a}}_N^e) = 0$
$I_e^{comp} = 0; i = 2$
$preI_e^{comp} = I_e^{comp}$
$I(\tilde{\mathbf{a}}_i^a) = \mathbf{combine}[I_a^{comp}, I(\tilde{\mathbf{a}}_2^e), \cdots, I(\tilde{\mathbf{a}}_N^e)|_{except} I(\tilde{\mathbf{a}}_i^e)]$
$I(\tilde{\mathbf{a}}_i^e) = \mathbf{T}'_{ind}(I(\tilde{\mathbf{a}}_i^a))$
$I_e^{comp} = \mathbf{combine}[I(\tilde{\mathbf{a}}_2^e), \cdots, I(\tilde{\mathbf{a}}_N^e)]$
$i = i + 1$; **If**$(i > N)$, $i = 2$
$I_e^{comp} > preI_e^{comp}$?
yes
no
$I(\tilde{\mathbf{a}}_1^a) = I_e^{comp}$

Figure 4.8: The flow chart of the composite EXIT function $\mathbf{T'_{comp}}$, which reflects the relationship between the *a priori* information and the extrinsic information for the composite decoder having $(N-1)$ URC-BCJR decoders, as seen in Figure 4.5.

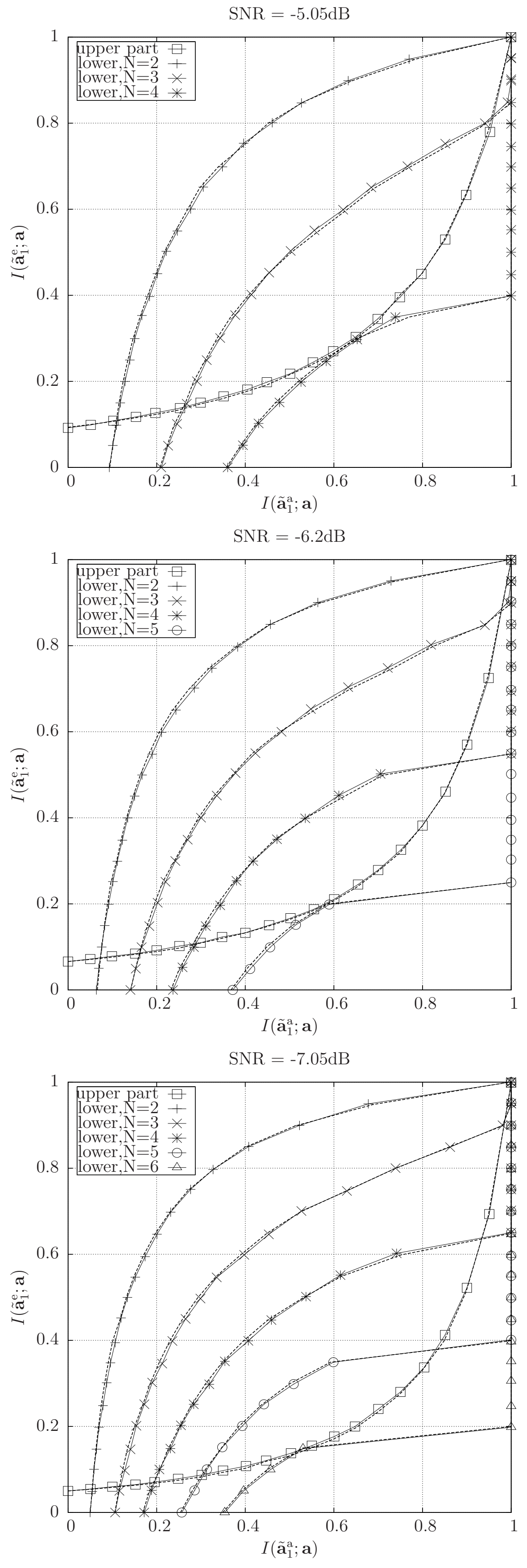


Figure 4.9: The extended EXIT charts for the MCTC structure of Figure 4.4 for $N = 2, 3, 4, 5, 6$. The URCs have a memory-one generator and feedback polynomial of $(2, 3)_{Octal}$. The communication channel is an uncorrelated Rayleigh channel. The continuous line shows the EXIT-functions based on the Gaussian assumption, while the dashed line is its spline-fitting based approximation.

4.1.3.3 The SNR Threshold of Creating an Open EXIT Chart Tunnel

As mentioned before, the creation of an open tunnel in an EXIT chart at a particular SNR is necessary for achieving iterative decoding convergence towards the maximum likelihood decoder's BER, provided that the interleaver is sufficiently long. We refer to the minimum SNR for which the EXIT chart has an open tunnel as 'the open tunnel SNR threshold'. Observe from Figures 4.9(a), (b) and (c) that 'the open tunnel SNR thresholds' of MCTCs using single-delay URC components and having throughputs of $R = \frac{1}{4}, \frac{1}{5}, \frac{1}{6}$ bits per channel use are -5.05dB, -6.2dB and -7.05dB, respectively.

This motivates the investigations documented in Figure 4.10, which provides a scatter plot of the throughput or rate R versus 'the open tunnel SNR' for MCTCs having a variety of generator and feedback polynomials. These polynomials were selected, because they were found to offer lower 'open tunnel SNR thresholds' than all other polynomials having the same length, at the specific throughputs considered. Figure 4.10 also plots the DCMC capacity of a BPSK-modulated Rayleigh fading channel as a function of its SNR [30]. It is the horizontal displacement of a particular symbol from this DCMC capacity plot that quantitatively characterizes the ability of the corresponding MCTC to approach the DCMC capacity. Note that upon using the spline-fitting based approximation of the EXIT function of the composite decoder, the resultant open-tunnel SNR threshold becomes the same as that obtained using the Gaussian LLR approximation method associated with an interleaver length of $1,000,000$ bits.

4.1.4 Simulation Results

In this section, we characterize the performance of the MCTCs designed and compare them to suitable systematic and non-systematic TCTC benchmarkers. The encoders of these benchmarkers employ two URC encoders for generating the encoded bit sequences $\mathbf{b_1}$ and $\mathbf{b_2}$, which are transmitted the required number of times in order to achieve the desired throughput or rate of R bits per channel use. For example, the output sequence is arranged to be $(\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_1})$ for non-systematic TCTCs and $(\mathbf{a}, \mathbf{b_1}, \mathbf{b_2})$ for systematic TCTCs, when a throughput of $R = \frac{1}{3}$ bits per channel use is desired. For $R = \frac{1}{4}$, it is $(\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_1}, \mathbf{b_2})$ for non-systematic TCTCs and $(\mathbf{a}, \mathbf{b_1}, \mathbf{b_2}, \mathbf{a})$ for systematic TCTCs. The corresponding arrangements conceived for lower R may be deduced by analogy. At the receiver, the LLRs corresponding to different replicas of the same encoded bit sequence are summed, before the iterative decoding process commences.

Figure 4.10 displays the 'open-tunnel SNR thresholds' of MCTCs, of non-systematic and of systematic TCTCs, when transmitting over BPSK-modulated Rayleigh fading channels. Here, only the lowest open-tunnel SNRs derived for TCTCs are recorded among all the polynomials having a memory length of $m \leq 3$. Upon observing the achievable performance of MCTCs and non-systematic TCTCs, we conclude that the

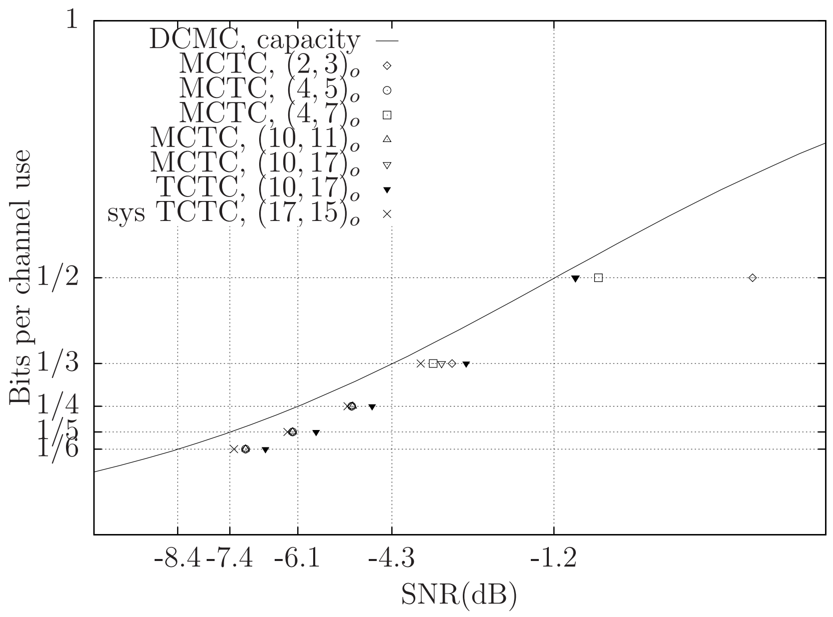
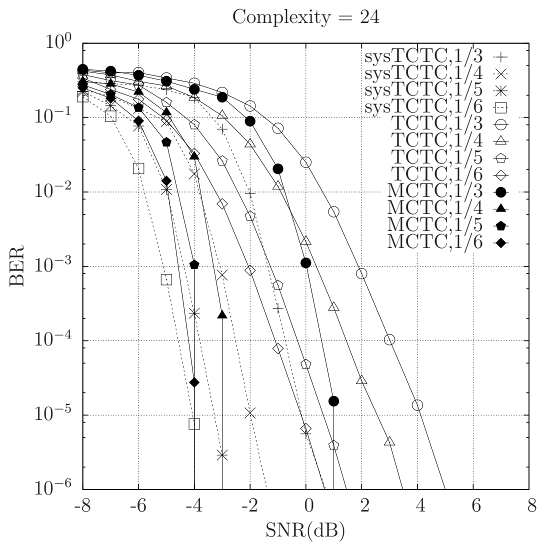
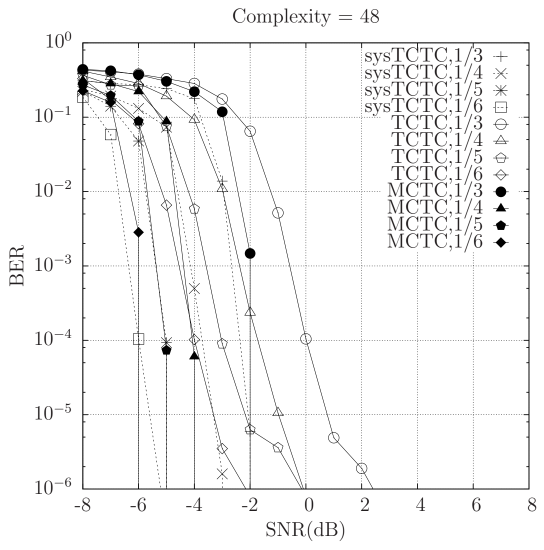


Figure 4.10: Throughput versus open-tunnel-SNR threshold for both MCTCs, as well as systematic and non-systematic TCTCs using various generator polynomials, when communicating over a BPSK-modulated Rayleigh fading channel.

SNRs at which MCTCs are capable of creating open EXIT tunnels are lower than those required for non-systematic TCTCs, when considering throughputs below $R = \frac{1}{2}$ bits per channel use. Furthermore, in order to create an open tunnel at these SNRs, the non-systematic TCTCs are required to use the polynomial of $(10, 17)_o$, which corresponds to $m = 3$ memory elements in the URC encoder and to $2^m = 8$ states in the BCJR decoder. By contrast, MCTCs create open tunnels at the lowest SNRs, when using the polynomial $(4, 7)_o$ at a throughput of $R = \frac{1}{3}$ and the polynomial $(2, 3)_o$ at throughputs below $R = \frac{1}{3}$. These polynomials are associated with lower BCJR decoding complexities, since they correspond to as few as $2^m = 4$ and 2 states, respectively. However, Figure 4.10 also shows that systematic TCTCs require the lowest SNRs, when the $m = 3$ polynomial of $(17, 15)_o$ is adopted. This indicates that the presence of systematic information is beneficial for TCTCs, if an appropriate polynomial can be found.

Although systematic TCTCs exhibit a better performance than MCTCs, they have to depend on the $m = 3$ polynomial $(17, 15)_o$, while MCTCs benefit from having lower complexities per BCJR algorithm activation than both systematic and non-systematic TCTCs. For this reason, Figure 4.11 compares the BERs that can be achieved both by MCTCs and by TCTCs, when fixed decoding complexities are used for recovering the information transmitted over a BPSK-modulated Rayleigh fading channel using an interleaver length of 2048 bits. Here, the complexity recorded in Figure 4.11 is defined as *complexity* $= 2^m \cdot K$, where m is the number of memory elements in each URC encoder and K is the total number of BCJR operations that are performed in





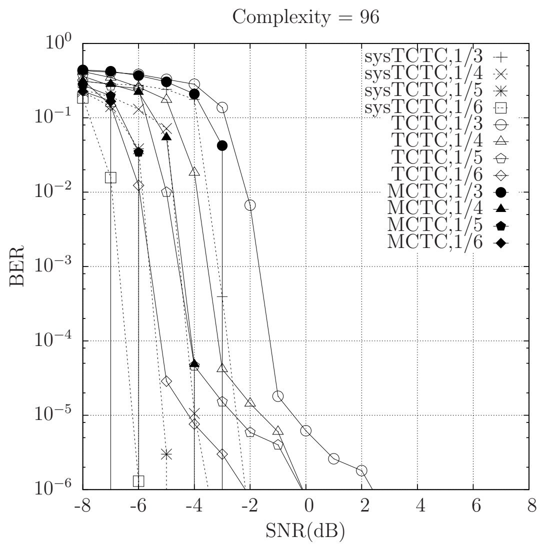


Figure 4.11: The BER vs. SNR for MCTCs and TCTCs having different complexities. All the non-systematic TCTCs having different rates R employ the polynomial of $(10, 17)_o$, and all the systematic TCTCs having different rates R employ the polynomial of $(17, 15)_o$, while the $R = \frac{1}{3}$ MCTC employs $(4, 7)_o$ and the MCTCs having $R < \frac{1}{3}$ employ $(2, 3)_o$.

the receiver, where a sequential decoder activation order is employed. As expected, Figure 4.11 shows that higher affordable complexities result in improved BER performances. The MCTCs have significantly steeper turbo cliffs and significantly lower error floors than the corresponding non-systematic TCTCs at all the complexities considered. As a result, at a complexity of 48, for example, the MCTCs offer 3.5dB to 4.5dB gain over the non-systematic TCTCs at a BER of 10^{-6}. By contrast, the MCTCs have slightly flatter turbo cliffs but perceivably lower error floors, compared to the systematic TCTCs having the same coding rates and complexities. For example, at a complexity of 96, the turbo cliff SNRs of the MCTCs and the systematic TCTCs are similar. Still considering the complexity of 96, the BER of MCTCs may become vanishingly low without exhibiting an error floor, whereas that of the systematic TCTCs is on the order of 10^{-7}.

4.1.5 Conclusions

The area properties of the EXIT charts reveal that classical TCTCs inherently suffer from a capacity loss, when the overall turbo coding rate is required to be less than $\frac{1}{2}$. This motivates the design of MCTCs, which overcome this impediment by increasing the number of concatenated component codes, rather than reducing the component code rate below unity. In this section, we provided a procedure for employing 2D EXIT charts for investigating N-component paralled concatenated codes, which would require N-dimensional EXIT charts. We demonstrated that 'the open-tunnel SNR thresholds' of MCTCs are closer to the DCMC capacity than those of non-systematic TCTCs, albeit they are slightly outperformed by their systematic TCTCs counterparts. Furthermore, we showed that MCTCs offer significantly steeper BER turbo cliffs and lower error floors than non-systematic TCTCs, while exhibiting slightly flatter turbo cliffs but lower error floors than systematic TCTCs, when considering the same coding rate and the same complexity.

In order to facilitate the efficient exploitation of the available bandwidth, the channel coding rate R of a communication system should vary as a function of the channel conditions. More explicitly, a near-unity R is adequate for good channel conditions, while a lower rate is required for hostile environments. HARQ schemes have the essential property of automatically adjusting the coding rate according to the prevalent channel conditions without any knowledge of the Channel State Information (CSI). Since MCTCs have shown an attractive performance in this section and may provide Incremental Redundancy (IR) for each transmission, we will combine the MCTCs with HARQ to realize a dynamic coding rate for our system in the following section and investigate the corresponding PLR, throughout and complexity.

4.2 MCTC Aided Hybrid ARQ

HARQ [123–125] plays an essential role in data communication systems, incorporating Forward Error Correction (FEC). When the system is equipped with a HARQ scheme, the system architecture of Figure 4.1 is extended to Figure 4.12, where the HARQ represented by the dotted rectangle amalgamates the MAC and PHY layers.

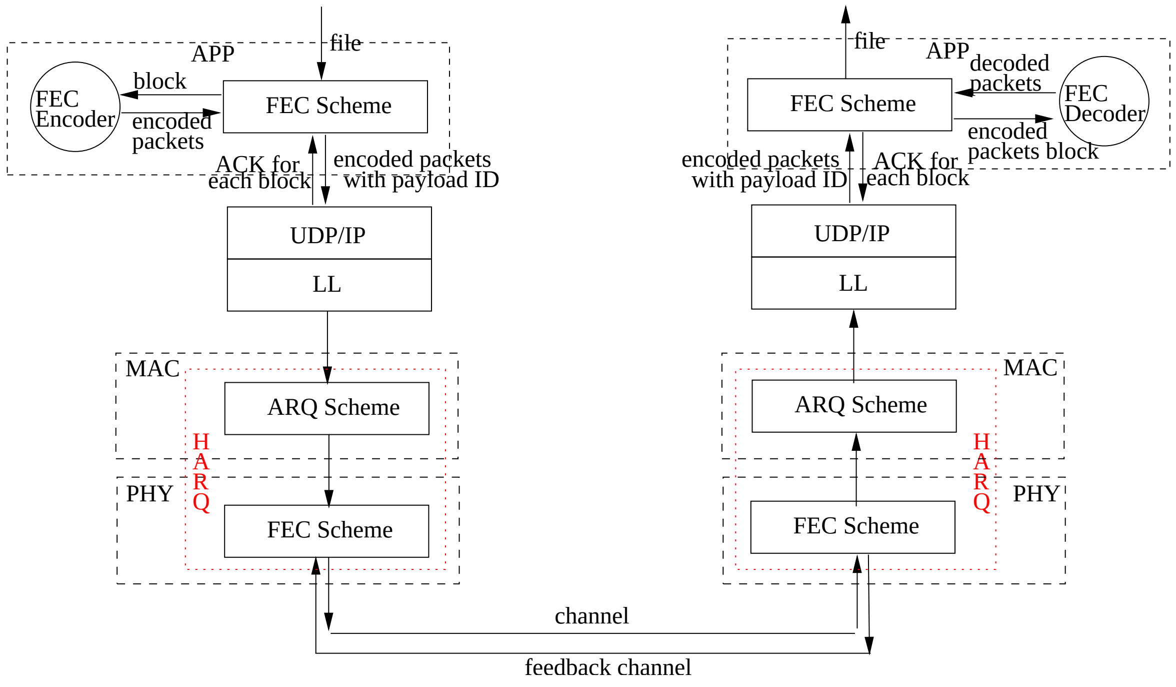


Figure 4.12: The architectural extension of Figure 4.1 by incorporating a HARQ scheme.

4.2.1 Introduction

In general, HARQ schemes combine ARQ with FEC techniques, as illustrated in Figure 4.13. It may be observed that each information packet will be accompanied by a number of check bits of an error detection code, for example a Cyclic Redundancy Check (CRC) scheme. These check bits are denoted by 'D' in Figure 4.13. Then, the resultant packet is passed to the FEC encoder. The encoded packet represented by 'Q' in Figure 4.13 may have different forms depending on the specific type of the HARQ scheme. For example, it may be the input packet accompanied by a number of error correction parity bits. In this case, the packet length is increased. Alternatively, it may only contain the parity bits having the same or even a lower length than the original packet.

At the receiver, the FEC decoder performs decoding based on the exact decoding algorithm and the error correction bits. Furthermore, different HARQ schemes have different combining strategies for this decoding operation. The decoding may only process on the currently received packet $\tilde{Q}$, as seen in Figure 4.13. However, combining the previous corrupted replicas for the sake of decoding them together becomes a better method. If errors are still detected in the estimated message, the current replica will be discarded, triggering a retransmission by sending a Negative ACKnowledgement (NACK) or by simply waiting for the transmitter to time out. This process will be repeated, until a retry limit is reached or the packet is correctly received.

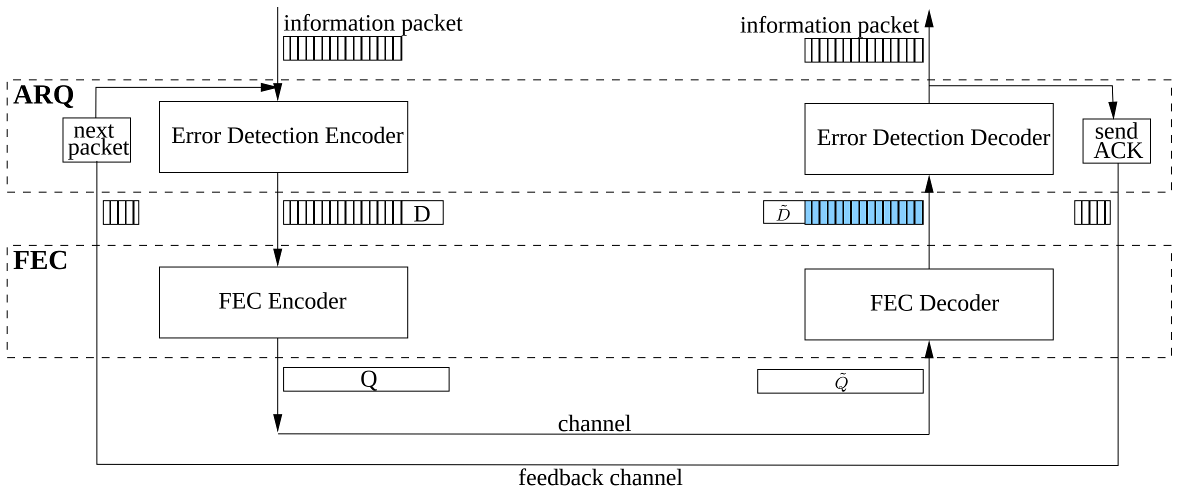


Figure 4.13: The general illustration of HARQ, where the Error Detection Encoder (EDE) and FEC encoder generate the codewords Q to be transmitted. The received codewords $\tilde{Q}$ are FEC decoded and in the presence of detection errors further, redundancy is requested from the transmitter.

Based on the successive development in FEC techniques, HARQ schemes have been classified into Type-I, Type-II and Type-III categories, which overcame the shortcomings of the previous versions. Furthermore, the achievable throughput also benefits from combining the retransmitted packets, which is evolved from using no combining at all in the early days, to combining parts of the retransmitted packets and finally to combining all received replicas.

4.2.2 Three Types of HARQ

As discussed in Section 3.1, HARQ schemes aggregate the advantages of ARQ and FEC arrangements for the sake of achieving a high throughput. However, for the earliest concept of Hybrid ARQ proposed and analyzed in 1970 [126], the authors of [127] claimed that separate ARQ or FEC may attain a better performance than HARQ. The straightforward combination of ARQ and FEC is referred to as Type-I HARQ. More explicitly, the FEC encoded packet 'Q' seen in Figure 4.13 is the original packet plus a number of parity bits. At the receiver, if errors cannot be corrected by the FEC decoder, the corrupted packet will be thrown away. Each FEC decoding is solely performed on a single encoded packet.

Referring to [123], Figure 4.14 shows the trends of throughput efficiency of a single ARQ, Type-I HARQ and Type-II HARQ. As seen in Figure 4.14, Type-I HARQ may improve the attainable throughput efficiency for moderate bit error ratios, compared to a single ARQ. This is because a coding gain results in a reduced number of retransmissions required for the successful delivery of a packet. However, if the channel conditions are sufficiently good, the reduced number of errors does not require an excessive FEC parity bits. In this situation, the resultant throughput efficiency is decreased by the excess parity bits. By contrast, if the channel conditions are hostile, the FEC code may not correct all errors for each transmission. Hence, the throughput tends to zero, similar to that of a single ARQ. Most of the early HARQ schemes designed in the 1970s and early 1980s [126, 128–131] belong to the family of Type-I HARQs.

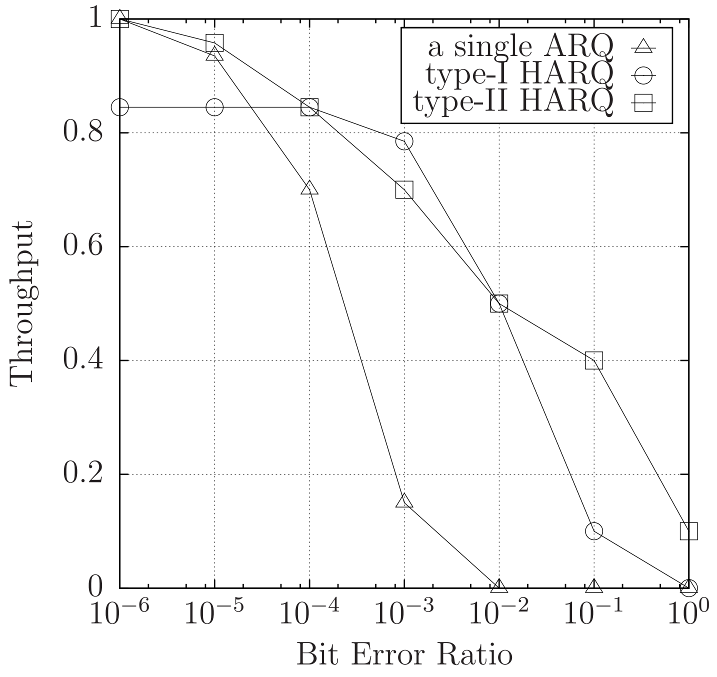


Figure 4.14: The throughput trends of a single ARQ, type-I HARQ and type-II HARQ.

Type-II HARQ was described by the authors of [123] in 1982. In order to eliminate the throughput degradation of Type-I HARQ for transmission over a benign channel, the first transmission of Type-II HARQ retains the same ARQ packet without adding any redundancy. More specifically, the first transmitted encoded packet ‘Q’ of Figure 4.13 is the original packet, i.e. without being FEC encoded. If the channel conditions are good, this uncoded packet may indeed be received correctly. Otherwise, the second transmission will be requested. Only parity bits are transmitted during the second transmission. Then at the receiver, FEC decoding is performed based on the combination of these new parity bits and the corrupted original packet of the prior transmission attempt. If the FEC decoder cannot correct all errors, the uncoded original packet will be transmitted in the third transmission attempt. At this time, it will be combined with the second transmitted parity bits for FEC decoding. If errors still exist, the same sequence of parity bits is transmitted again during the fourth transmission. This process is repeated until the packet is correctly received or the retry limit is reached.

Figure 4.14 illustrates that Type-II HARQ has the highest throughput for transmission over a benign channel among the single ARQ, Type-I and Type-II HARQ schemes. However, the throughput efficiency becomes lower than that of Type-I HARQ in the middle of the BER range, as seen in Figure 4.14. This is because it has to combine two transmissions for commencing FEC decoding, while Type-I HARQ carries parity bits by each transmission.

When using turbo codes [47], the concept of Type-III HARQ was created. Like in Type-II HARQ, Type-III HARQ uses different redundant information during each transmission attempt, but each of them is self-decodable. In more detail, the transmitted packet 'Q' of Figure 4.13 may contain redundant information, which is provided by the FEC encoder, as well as have the same length of original packet. This can enhance the probability of successful delivery for the transmissions of pure systematic bits in Type-II HARQ schemes. The authors of [132–135] characterized the attainable performance of Type-III HARQ schemes, which exhibited the best performance compared to the Type-I and Type-II HARQ schemes.

4.2.3 Combining Transmissions

During the evolution of HARQ schemes, minimizing the required number of retransmissions has received a significant research attention, because unnecessary retransmissions reduce the effective throughput. A typical approach has been that of combining the various corrupted retransmission components in order to provide a more reliable decision for the original bits. Two main combining strategies have been proposed, namely Chase combining [136] and the transmission of Incremental Redundancy (IR) [137]. Chase combining achieves diversity gain by beneficially combining the identical data replicas conveyed during the different retransmissions. By contrast, IR conveys different redundant information during each transmission attempt, which may be combined and reconstructed by a single FEC decoder at the receiver. More recently, the employment of incremental redundancy has found applications in cooperative networks [138], [139].

Soft decision aided Chase combining and incremental redundancy based techniques have been proposed for HARQ schemes that use iterative soft-decision-based FEC decoders [140], [141], like turbo codes. For example, the High Speed Downlink Packet Access (HSDPA) protocol [142] uses a punctured $R = 1/3$-rate turbo code as the basis of its HARQ scheme. Here, whenever a retransmission is received, the corresponding LLRs will be added to those that were recovered from previous transmissions or used to provide soft information for bits that were punctured during previous transmissions. Similarly, [124], [140] and [141] used the LLRs obtained from previous transmissions as *a priori* values during the decoding of the retransmissions. In a recent paper by Souza *et al.* [125] proposed a HARQ scheme that integrates Chase combining and incremental redundancy in a systematic TCTC. Here the LLRs obtained from each replica of a packet are added and iterative decoding is employed to recover the transmitted data.

4.2.4 Motivation

In contrast to the systematic TCTC of [125], MCTCs employ a parallel concatenation of more than two component codes, which are combined by iterative decoding at the receiver, as introduced in Section 4.1. Again, an N-component turbo code having an overall rate R can be interpreted as a parallel concatenation of N component codes, each having a coding rate of $R_c = RN$. Section 4.1 has compared the achievable BER performance of MCTCs, as well as of systematic and non-systematic TCTCs. Hence, we drew the conclusion that MCTCs outperform TCTCs at the same complexity, which motivates the design of our MCTC aided HARQ scheme. As in Souza's scheme [125], the overall coding rate R of our scheme decreases to 1/2, 1/3, 1/4 and so on with each subsequent retransmission. However, in our scheme the number of component codes N combined by the iterative decoder is accordingly increased to 2, 3, 4 and so on with each retransmission, therefore allowing us to maintain $RN = 1$ and hence avoiding the above-mentioned capacity loss in Section 4.1.

The rest of this section is organized as follows. The system model and the proposed HARQ scheme are presented in Section 4.2.5, where several different benchmarker HARQ schemes are also highlighted. Section 4.2.6 discusses our PLR, throughput and complexity results. Drawing on these results, Section 4.2.7 concludes the section.

4.2.5 System Model

In this section, we will describe four different turbo HARQ schemes. Before introducing the proposed MCTC aided HARQ scheme, we describe both the original and a puncturing-aided version of Souza's scheme[1] [125], both of which use a twin-component turbo code. We also describe a third benchmarker, which employs a three-component turbo code in order to allow the investigation of an intermediate design between Souza's scheme and our own.

Each of the four HARQ schemes employs a simple stop-and-wait ARQ protocol, which appends an $(k-l)$-bit CRC to the l-bit message $\mathbf{u} = [u_1, \cdots, u_l]$ in order to facilitate reliable error detection. The resultant k-bit packet $[u_1, \cdots, u_l, u_{l+1}, \cdots, u_k]$ is input to the corresponding FEC scheme. Each of the four HARQ schemes conveys the systematic bits $\mathbf{a} = [u_1, \cdots, u_l, u_{l+1}, \cdots, u_k]$ during the first transmission, in order to achieve the maximal throughput, when the channel is benign. In all cases, BPSK is used. If the hard-decision CRC detection fails at the receiver, it will send back a NACK or will simply wait for the transmitter's timeout to trigger a retransmission.

[1]This puncturing-aided scheme is introduced for the sake of direct comparability with the proposed design.

Table 4.3: Summary of the information transmitted by the four hybrid ARQ schemes.

Tx Number	Souza's scheme	Puncturing-aided Souza's scheme	Fixed 3-component scheme	Proposed scheme
1*st* Tx	$\mathbf{a}$	$\mathbf{a}$	$\mathbf{a}$	$\mathbf{a}$
2*nd* Tx	$\mathbf{b_1}$	$punc(\mathbf{b_1}, \mathbf{b_2})$	$punc(\mathbf{b_1}, \mathbf{b_2})$	$punc(\mathbf{b_1}, \mathbf{b_2})$
3*rd* Tx	$\mathbf{b_2}$	$\mathbf{b_1}$	$\mathbf{b_3}$	$\mathbf{b_3}$
4*th* Tx	$\mathbf{a}$	$\mathbf{b_2}$	$\mathbf{b_1}$	$\mathbf{b_4}$
5*th* Tx	$\mathbf{b_1}$	$\mathbf{b_1}$	$\mathbf{b_2}$	$\mathbf{b_5}$
6*th* Tx	$\mathbf{b_2}$	$\mathbf{b_2}$	$\mathbf{b_3}$	$\mathbf{b_6}$

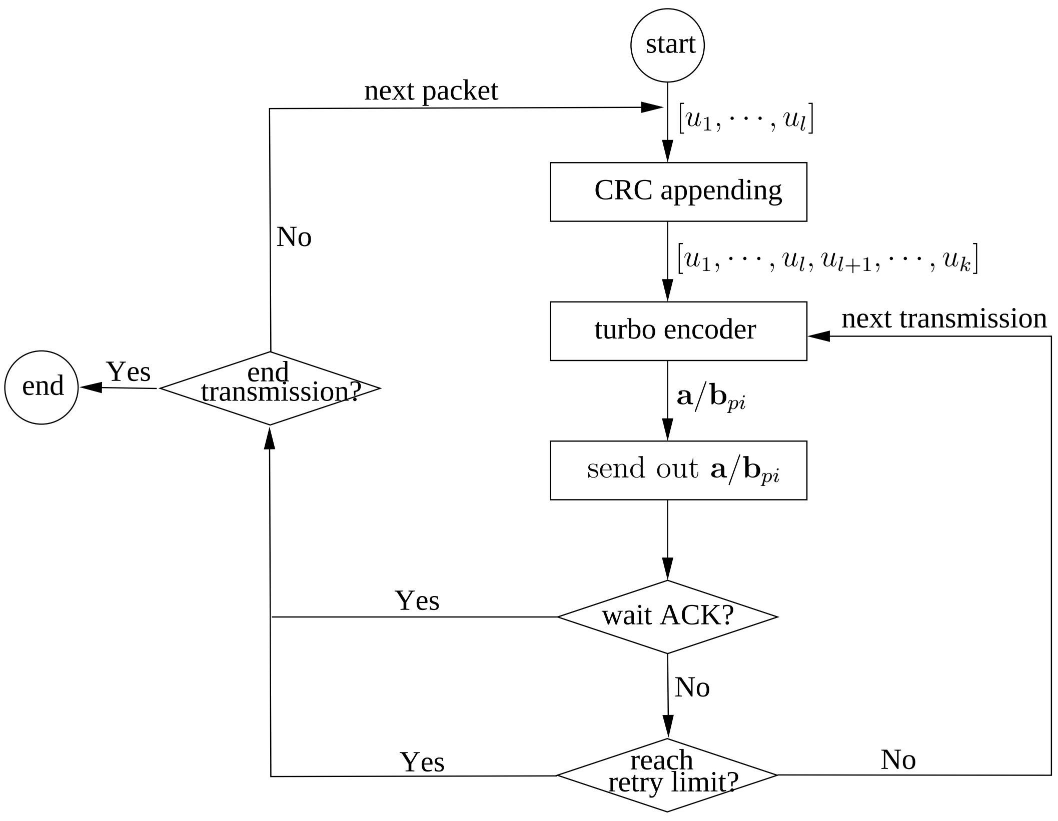


Figure 4.15: The turbo coded HARQ flow-chart used at the transmitter of the HARQ framework of Figure 4.13.

In this case, the FEC scheme generates the information to be retransmitted using a particular parallel concatenated URC [114], as it will be detailed in the context of Figures 4.17 and 4.19. The BCJR algorithm [45] is employed to facilitate iterative decoding. Retransmissions are continually generated, until the hard-decision based CRC suggests error-free detection or until a maximum of $M = 6$ transmissions have been sent, at which point, the packet will be discarded. Furthermore, Figures 4.15 and 4.16 illustrate the flow charts of these turbo coded HARQ schemes at the transmitter and receiver, respectively.

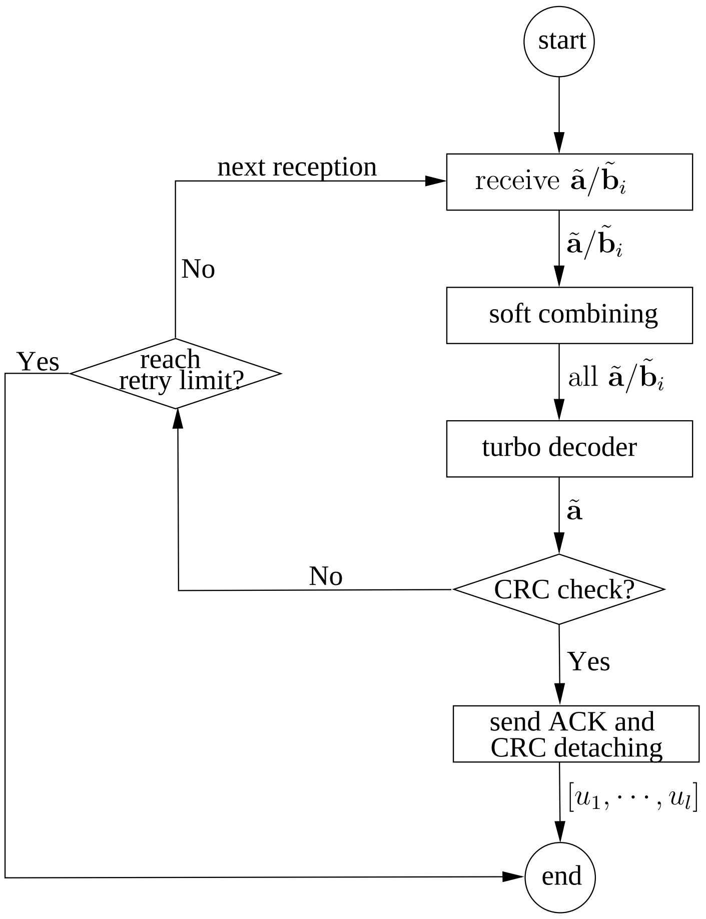

start
receive $\tilde{\mathbf{a}}/\tilde{\mathbf{b}}_i$
$\tilde{\mathbf{a}}/\tilde{\mathbf{b}}_i$
soft combining
all $\tilde{\mathbf{a}}/\tilde{\mathbf{b}}_i$
turbo decoder
$\tilde{\mathbf{a}}$
CRC check?
Yes
send ACK and CRC detaching
$[u_1, \cdots, u_l]$
end
No
reach retry limit?
No
next reception
Yes

Figure 4.16: The HARQ flow-chart employed at the receiver of the HARQ framework of Figure 4.13.

The main difference between the four HARQ schemes is in the choice of the particular parallel concatenation of the component URCs that they employ. Table 4.3 summarizes the information conveyed by each transmission in these four HARQ schemes, where $\mathbf{b_i}$ refers to the parity bits generated by the ith parallel concatenated URC encoder. Note that independent pseudo-random interleavers are employed to ensure that each parallel concatenated URC encoder considers a different ordering of the bit sequence $\mathbf{a}$.

4.2.5.1 Souza's Scheme

Figure 4.17 depicts the configuration of the parallel concatenated URC codes employed by Souza's scheme [125]. In simple terms, Souza's scheme transmits $\mathbf{a}, \mathbf{b_1}, \mathbf{b_2}, \mathbf{a}, \mathbf{b_1}$ and $\mathbf{b_2}$ during the $M = 6$ transmissions, as contrasted to the other schemes in Table 4.3. In Souza's original proposal, $(\mathbf{a}, \mathbf{b_1})$ are transmitted together during the first transmission. However, in order to maintain the same coding rate in the four HARQ schemes, we separate this first copy into two transmissions in order to attain the maximal throughput, when the channel SNR is sufficiently high.

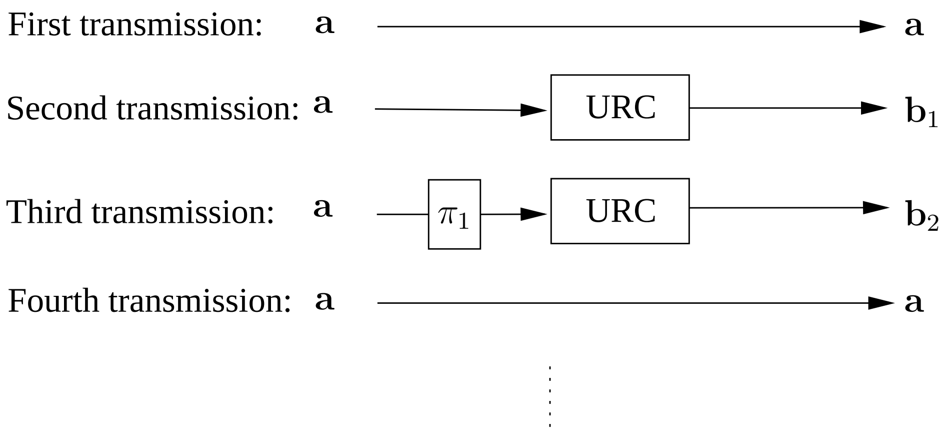


Figure 4.17: The configuration of the parallel concatenated URC codes in Souza's HARQ scheme, where the systematic bit sequence $\mathbf{a}$ and two parity bit sequences $\mathbf{b_1}$, $\mathbf{b_2}$ are transmitted in turn for each transmission.

At the receiver, we let $\tilde{\mathbf{a}}^{(j)}$ and $\tilde{\mathbf{b}}_i^{(j)}$ represent the LLRs of the received packets corresponding to the transmitted $\mathbf{a}$ and $\mathbf{b}_i$, where the superscript j denotes the jth repetition packet, for example, $\tilde{\mathbf{a}}^{(1)}$ and $\tilde{\mathbf{a}}^{(2)}$ indicate the systematic LLRs during the first and fourth transmission, respectively. Hard-decision based error detection is applied to the systematic bits recovered from the systematic LLRs $\tilde{\mathbf{a}}^{(1)}$ during the first transmission. For the parity LLRs $\tilde{\mathbf{b}}_1^{(1)}$ gleaned from the second transmission, only one BCJR operation is carried out, using the previous systematic LLRs $\tilde{\mathbf{a}}^{(1)}$ as the *a priori* input. Once the third transmission has been received, a twin-component turbo decoder is constructed and iterative decoding starts. From then on, the newly received repetitions of the packets owing to later retransmissions are added, as seen in Figure 4.18. More specifically, the fourth received $\tilde{\mathbf{a}}^{(2)}$ is added to $\tilde{\mathbf{a}}^{(1)}$, the fifth received $\tilde{\mathbf{b}}_1^{(2)}$ is added to $\tilde{\mathbf{b}}_1^{(1)}$ and so on. Once the turbo decoder has been constructed, up to five decoding iterations are performed following the reception of each packet, each comprising two BCJR URC decoding operations, as it will be detailed in the next section. By contrast, a reduced number of decoding iterations are performed, if the process converges or if a BCJR operation produces *a posteriori* LLRs that result in a legitimate codeword, hence satisfying the CRC, in which case the hard decision bits are output and an ACK flag is returned to the transmitter. The extrinsic information obtained following the reception of the previous retransmission is used as *a priori* information in order to initialize the iterative decoding process.

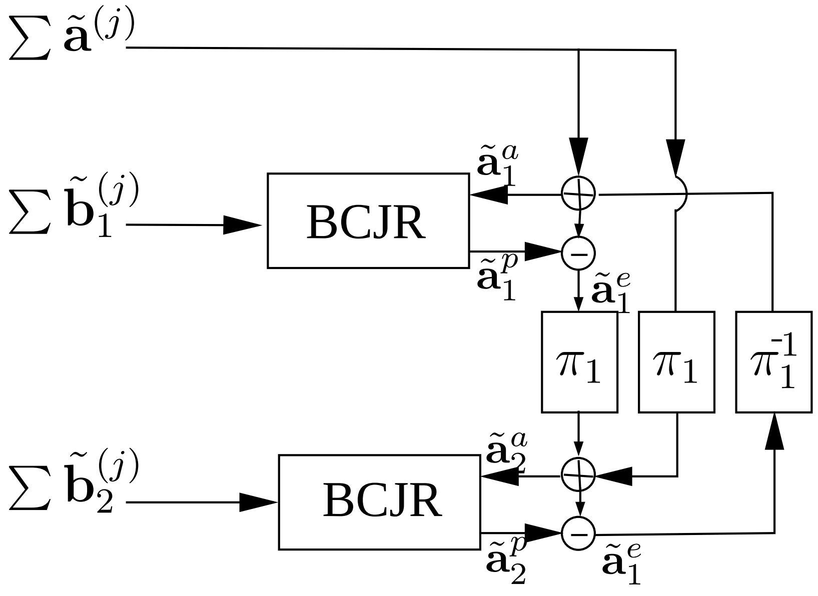


Figure 4.18: The decoder structure of Souza's scheme after six transmissions, corresponding to the encoder structure of Figure 4.17, where iterative decoding is always performed between two URC-BCJR decoders no matter how many replicas have been received. The LLRs of the repetition packet are combined for the decoding.

4.2.5.2 Our Proposed Scheme

Figure 4.19 illustrates the different transmissions generated by our proposed scheme. In contrast to Souza's original scheme, where the bits encoded by a rate-1/2 Recursive Systematic Convolutional (RSC) code are transmitted during the first transmission, our scheme initially transmits only the systematic bits $\mathbf{a}$ in order to achieve the maximal throughput, when the channel SNR is sufficiently high. The second transmission is generated by puncturing the encoded output bits of $N = 2$ URC encoders to generate exactly the same number of bits, as during the first transmission, as seen in Figure 4.19. This approach achieves a coding rate of $R = 1/2$ after two transmissions, maintaining $RN = 1$ and facilitating iterative decoding. Similarly, during the subsequent retransmissions, different interleavers and additional URCs are employed to incrementally generate further URC-encoded bits and to maintain $RN = 1$.

At the receiver, a multi-component turbo code is constructed. Following the second transmission characterized in Figure 4.19, an initial twin-component turbo code is formed and iterative decoding commences. Thereafter, a BCJR decoder is activated upon the reception of each transmitted packet, hence increasing the number of components in the turbo decoder. In our HARQ scheme, the *a priori* input provided for each newly activated BCJR decoder is generated as the sum of all the extrinsic information

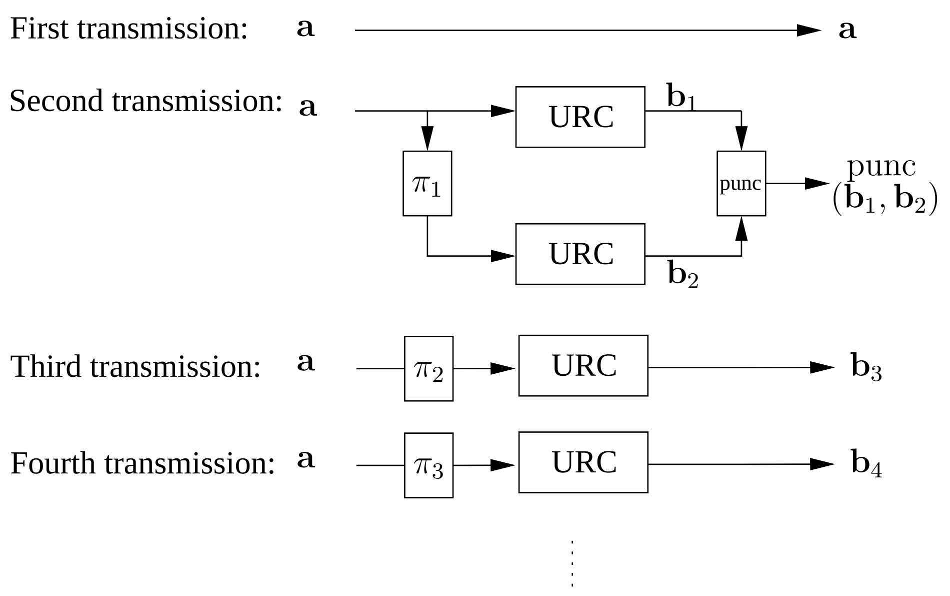


Figure 4.19: Encoder in different retransmissions in our MCTC aided HARQ, where the systematic bit sequence $\mathbf{a}$ is transmitted during the first transmission, the parity bits punctured from $(\mathbf{b}_1, \mathbf{b}_2)$ are transmitted during the second transmission, and for each of the following transmissions, there will be a new parity bit sequence transmitted.

contributions obtained from decoding the previous transmissions, as well as the interleaved systematic information obtained during the first transmission. In return, the extrinsic LLRs $\tilde{\mathbf{a}}^e$ of the new BCJR decoder are passed back to aid the other BCJR decoders. Figure 4.20 shows the decoder's structure. As in Souza's scheme seen in Figure 4.18, up to ten BCJR operations are performed following the reception of each transmission, ensuring that both schemes have the same complexity.

In detail, for the systematic bits received during the first transmission, only CRC decoding is carried out. If any bit errors have been detected, the receiver requests the second transmission. The second transmission contains the punctured encoded bits of the first two URC codes, hence depuncturing is employed to reinsert the punctured bits and therefore to provide the soft input for the two BCJR decoders respectively, while the systematic LLRs obtained during the first transmission are employed as the *a priori* input. Again, the depuncturing operation reinserts the punctured bits of $\tilde{\mathbf{b}}_1$ and $\tilde{\mathbf{b}}_2$, both of which initially have zero-valued LLRs. At this point, iterative decoding commences. If this iterative decoding process fails to obtain an *a posteriori* output that satisfies the CRC, the third transmission will be requested. In Figure 4.20, $\tilde{\mathbf{a}}_1^e$, $\tilde{\mathbf{a}}_2^e$ represent the extrinsic LLRs obtained by the first two BCJR decoders. These are added to the systematic LLRs received from the first transmission and employed as the *a priori* input for the third BCJR decoder. At this point, iterative decoding recommences, starting with the third BCJR decoder, whose extrinsic LLRs are represented by $\tilde{\mathbf{a}}_3^e$. This is added to the extrinsic LLRs obtained by the second BCJR decoder and the

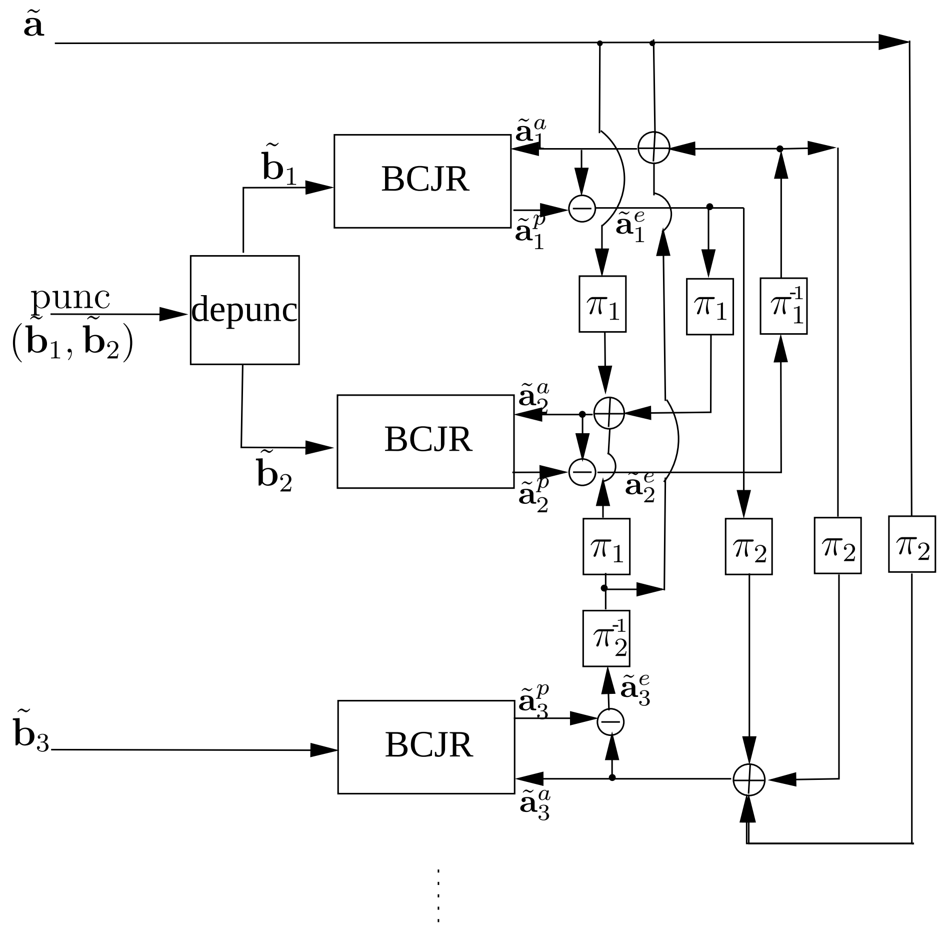


Figure 4.20: Decoder structure after six transmissions in our MCTC aided HARQ, corresponding to the encoder structure of Figure 4.19. The systematic LLR sequence **a** is input as the *a priori* information. A URC-BCJR decoder will be activated for each reception after the second one, and hence the turbo decoder contains N components after N transmissions. Iterations are performed among all component BCJR decoders. For every BCJR decoding, its *a priori* LLRs are generated by combining the extrinsic LLRs from all other decoders.

systematic LLRs in order to provide an *a priori* input for the first BCJR decoder. In this way, iterative decoding continues by exchanging extrinsic information among the three BCJR decoders. Likewise, if subsequent retransmissions are requested, further URC-encoded segments are appended and the LLR-addition operations will continue, until the packet becomes error-free or until the retransmission limit is reached.

4.2.5.3 The Puncturing-aided Souza Scheme and the Fixed Three-component Scheme

Since there are a number of differences between the operation of Souza's scheme and our own, we additionally consider two further benchmarkers, which represent the intermediate steps between the two schemes. In contrast to Souza's scheme, our approach employs puncturing. For this reason, our first additional benchmarker resembles Souza's scheme, but with the addition of puncturing. More specifically, this benchmarker adopts a twin-component turbo code and Chase combining of the packet replicas, as in Souza's scheme, but the second transmission uses the punctured URC-encoded bits of the two URC encoders. From the third transmission on, the parity bits $\mathbf{b_1}$, $\mathbf{b_2}$ are alternately transmitted, as shown in Table 4.3. Correspondingly, depuncturing is employed at the receiver similar to our proposed scheme.

We additionally consider a benchmarker that employs a three-component turbo code, in order to investigate the intermediate solution between a twin- and a multi-component turbo code. As shown in Table 4.3, this benchmarker consecutively transmits the parity bits $\mathbf{b_1}$, $\mathbf{b_2}$, $\mathbf{b_3}$. The schematic of the decoder designed for this benchmarker can be created by simply concatenating no more than three BCJR decoders in Figure 4.20 and including the sum of the LLRs extracted from the repeated packets.

Despite the above-mentioned differences, there are a number of similarities between the four HARQ schemes. Firstly, there is no need to restart the iterative decoding process for each new retransmission in any of the schemes. Instead, the process continues from the state reached during the iterative decoding process employed after the previous transmission was received. Secondly, all four schemes employ the same iterative decoding stopping criteria. Namely, as soon as any BCJR operation produces *a posteriori* information that satisfies the CRC, decoding may be concluded and the successful detection of the packet can be acknowledged. Furthermore, whenever the mutual information associated with the extrinsic LLRs fails to increase by more than 0.001 between two consecutive operations of the same BCJR decoder, the same action of curtailing further iterations is carried out. Finally, the maximum number of BCJR operations that is performed following the reception of each transmission is limited to ten, like in Souza's original scheme, regardless of the number of component decoders. Owing to this measure, all of our schemes are associated with the same computational complexity, allowing their fair and equitable comparison.

4.2.6 Simulation Results

In this section, we compare the link-level PLR, throughput and complexity characteristics of the four schemes introduced in the Section 4.2.5. Our simulations considered the transmission of a statistically relevant number of packets, each comprising 256 bytes, over an uncorrelated Rayleigh fading channel. This packet length is more appropriate in network applications than the eight-times shorter 256-*bit* packets considered by Souza [125], which are disproportionately small compared to the length of the headers that are appended by the network protocols. In Souza's scheme, the generator polynomial of the URC was assumed to be $(17, 15)_o$. However, the analysis of MCTCs in Section 4.1 shows that the lowest memory-1 generator polynomial $(2, 3)_o$ is capable of achieving the best performance for the lower coding rates. Thus, all four turbo HARQ schemes proposed in Section 4.2.5 will employ the generator polynomial $(2,3)_o$ in our simulations. In addition, we also recreated the Souza's scheme using his original polynomial of $(17,15)_o$. The common parameters used in all of these turbo HARQ schemes are listed in Table 4.4.

Table 4.4: The common parameters used in all turbo HARQ schemes.

Retry Limit	6
Packet Length	256 bytes
Modulation Scheme	BPSK
Channel Type	uncorrelated Rayleigh fading

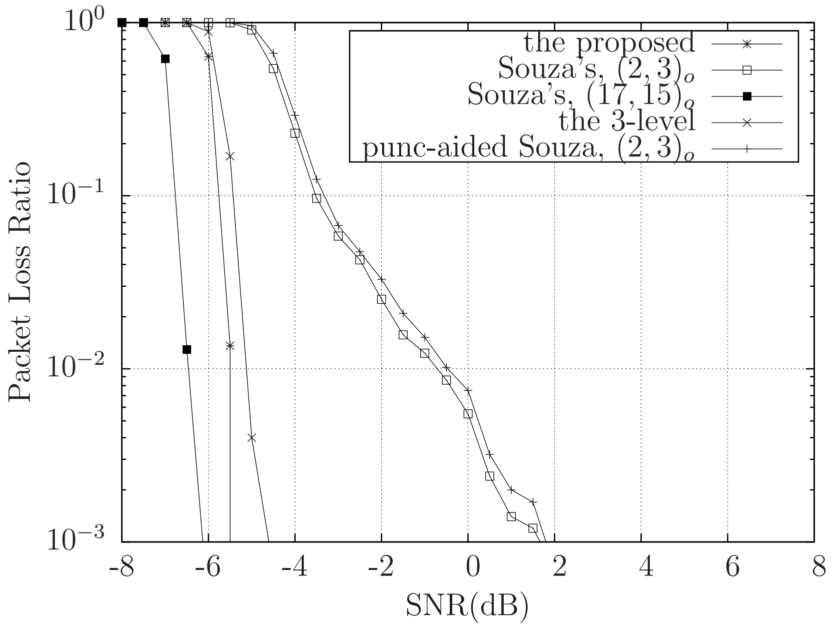


Figure 4.21: Link-level packet loss ratio versus SNR for transmission over uncorrelated Rayleigh channels when using the schematics of Figures 4.17, 4.18, 4.19, and 4.20 which were characterized in Table 4.3. Furthermore, the parameters of Table 4.4 were employed. The coding rate becomes 1/6 after six transmissions.

The link-level PLR versus SNR characteristics of the four schemes are shown in Figure 4.21. Here, a packet loss event occurs, whenever six transmissions are insufficient for the iterative decoding process to generate *a posteriori* information that satisfies the CRC. The results of Figure 4.21 show that for SNRs below −6.5dB, six transmissions are insufficient to allow error-free packet reconstruction in any of the four schemes, when the generator polynomial of $(2,3)_o$ is considered, while this SNR bound is decreased to −7.0dB for Souza's scheme relying on the polynomial $(17,15)_o$. However, when the channel SNR exceeds −6.5dB, the proposed scheme exhibits a significantly better performance than the three benchmarkers, despite having the lowest-memory polynomial of $(2,3)_o$, while the SNR corresponding to the same PLR is 1dB higher than that of Souza's original scheme using the polynomial of $(17,15)_o$. More specifically, observe in Figure 4.21 that Souza's scheme relying on the polynomial $(17,15)_o$ displays the steepest turbo cliff at the SNR of -6.5dB. Our scheme offers a less steep turbo cliff at the SNR of -5.5dB, both facilitating low PLRs in excess of these SNRs. By contrast, the PLRs of the twin-component turbo coded benchmarkers decrease more gradually, suffering from a near-unity PLR at the SNR of -6dB and a PLR of approximately 10^{-3} at the SNR of 2dB. As shown in Figure 4.21, the three-component turbo code benchmarker offers a PLR performance, which approaches that of our proposed scheme. This demonstrates that increasing the number of concatenated codes by even one may significantly enhance the attainable performance, under the premise that the polynomial $(2,3)_o$ is adopted. Figure 4.22 shows the four schemes' throughput

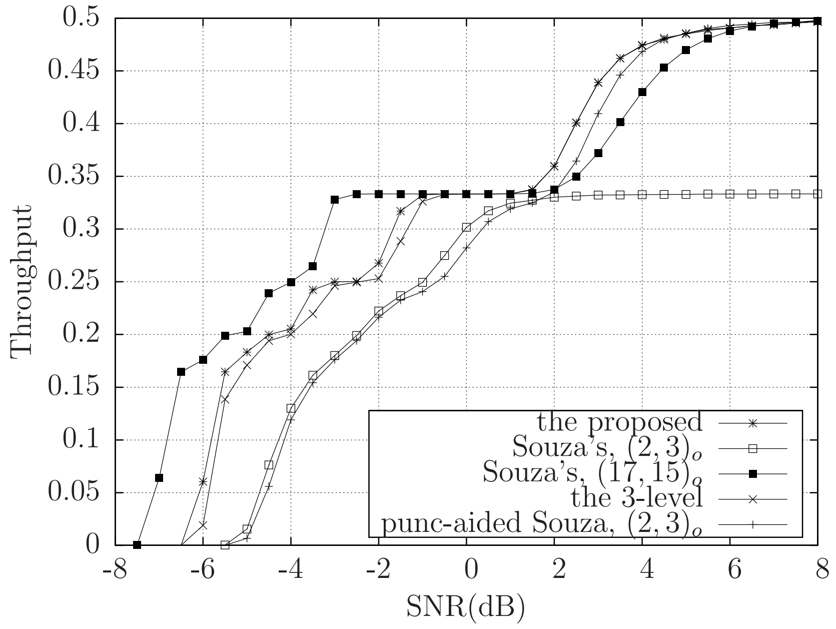


Figure 4.22: Throughput versus SNR for transmission over uncorrelated Rayleigh channels, when using the schematics of Figures 4.17, 4.18, 4.19, and 4.20 which were characterized in Table 4.3. Furthermore, the parameters of Table 4.4 were employed.

versus the SNR. Here, the normalized throughput is defined as the ratio of successfully recovered packets to the total number of transmitted packets. When the SNR is lower than -6dB, all schemes using the polynomial of $(2,3)_o$ have a zero throughput, indicating that no messages are successfully recovered and that the PLR is 1, as shown in Figure 4.21. However, the corresponding SNR degrades to -7dB for Souza's scheme in conjunction with the polynomial $(17,15)_o$. In the SNR range between -6dB and 0dB, our proposed scheme offers a 1.5dB to 2dB gain over the twin-component turbo coded benchmarkers in conjunction with the same polynomial of $(2,3)_o$. However, compared to that using the polynomial $(17,15)_o$, it has an approximately 1dB loss in the SNR range of -8dB to -2dB. There is however a significant throughput increase for SNRs between 1.5dB and 4dB for our scheme and for the three-component scheme, both of which offer a normalized throughput of about 0.5 in this region, which is significantly higher than the 0.33 throughput offered by Souza's scheme employing the polynomial $(2,3)_o$. A similar trend may be observed for Souza's puncturing-aided scheme in this region, which in fact requires a 0.5dB lower SNR than our scheme and than the three-component scheme. Furthermore, our proposed scheme also shows a 1dB gain over Souza's scheme having the polynomial $(17,15)_o$ in the SNR range of 2dB to 6dB. While we do not show simulation results for very high SNR values, all schemes are capable of approaching the normalized throughput of 1 when the channel is benign, because their initial transmissions are constituted by the original systematic information bits.

Finally, we consider how the iterative decoding complexity of the four schemes varies with the channel SNR. In Figure 4.23, this complexity is quantified in terms of the average number of BCJR operations performed during the reconstruction of each original message, which is denoted by K, while the polynomial's memory length by m. More explicitly, the complexity expression obeys the form of $K \cdot 2^m$. In general, the complexity of all the four schemes peaks in the SNR region that corresponds to the 'turbo cliff'. For SNRs below -8dB, the reduced complexity is explained by the rapid convergence of the iterative decoding process, when the amount of information received is low. Similarly, the low complexity at high SNRs is explained by the rapid acquisition of *a posteriori* information that satisfies the CRC. Observe that for SNRs in the range of $[-5\text{dB}, -2\text{dB}]$ and for SNRs in excess of 4dB, our proposed scheme offers the lowest complexity. In the other SNR regions, the proposed scheme does not have a significantly higher complexity than the benchmarkers having the same polynomial of $(2,3)_o$. When the SNR is below -3dB, our proposed scheme exhibits a significantly lower complexity than Souza's scheme having the polynomial of $(17,15)_o$. For this reason, the proposed scheme offers both PLR and throughput advantages without any significant complexity increase compared to the TCTC aided HARQ schemes using the least-complexity memory-1 polynomial. Moreover, it may significantly reduce the complexity at a minor degradation of the PLR and throughput compared to Souza's original HARQ scheme relying on the polynomial of $(17,15)_o$.

4.2.7 Conclusions

In this section, we have proposed a HARQ scheme based on multi-component turbo codes. Our design was motivated by the area properties of EXIT charts. More specifically, this approach avoids the capacity loss that is associated with using HARQ combined with twin-component, rather than multi-component turbo codes. Indeed,

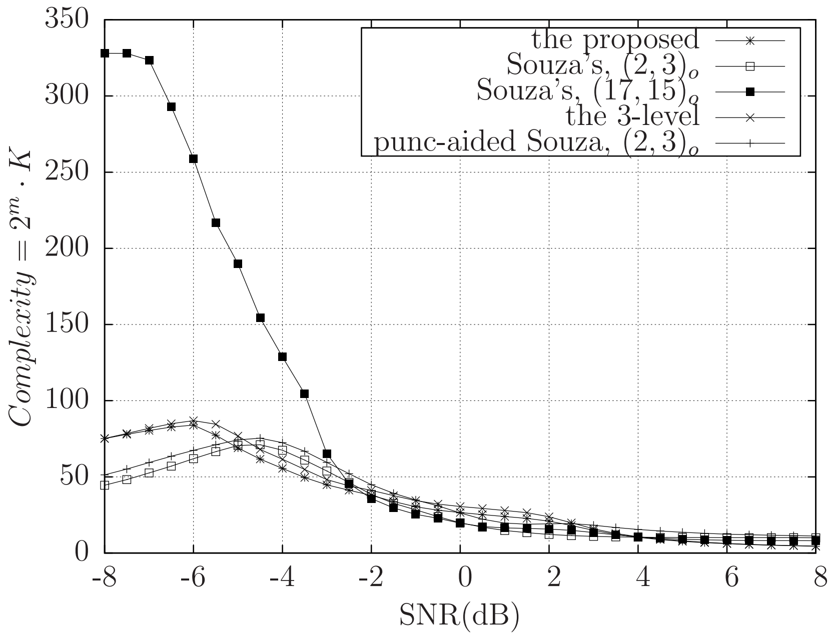


Figure 4.23: Complexity versus SNR for transmission over uncorrelated Rayleigh channels, when using the schematics of Figures 4.17, 4.18, 4.19, and 4.20 which were characterized in Table 4.3. Furthermore, the parameters of Table 4.4 were employed.

our simulation results have shown that the proposed approach outperforms Souza's scheme [125] in two aspects, when the polynomial of $(2, 3)_o$ is employed. This manifests itself in terms of its improved PLR and throughput, which is achieved without having an increased computational complexity. Furthermore, if the original generator polynomial $(17, 15)_o$ is used for Souza's scheme, the complexity may be significantly reduced by our scheme without a significant sacrifice in terms of PLR and throughput. However, our scheme requires the employment of several interleavers. In situations where this is unattractive, our results have demonstrated that a three-component turbo code offers a significant gain over the existing techniques, at the cost of requiring only a single additional interleaver.

4.3 Chapter Summary

In this chapter, we focused our attention on the channel coding and retransmission mechanisms of HARQ-aided communication systems. Since turbo codes have been shown to exhibit a near-capacity performance, they constitute the best choice for protecting the data in the PHY layer. Combining turbo codes with HARQ constitutes an attractive technique of integrating them into an intrinsically amalgamated system. The attainable system performance may be improved, if incremental redundancy can be provided by each HARQ transmission attempt, and MCTCs satisfy this requirement. Therefore, this chapter started out from the design and analysis of MCTCs. Section 4.1.2 outlined the MCTC encoder and decoder structures, while Section 4.1.3

partitioned N-component turbo codes into two logical parts, which makes it possible to visualize N-dimensional EXIT charts by recording the extrinsic information exchange between these two parts. Based on the extended EXIT chart concept of Section 4.1.3, open-tunnel-SNR thresholds may be obtained for characterizing the achievable performance of MCTCs, which is illustrated in Figure 4.10. Table 4.5 concludes the open-tunnel-SNR thresholds and the corresponding polynomials required for both MCTCs, as well as for systematic and non-systematic TCTCs.

Table 4.5: Open-tunnel-SNR thresholds of MCTCs, as well as systematic and non-systematic TCTCs, and their generator polynomials

	MCTCs	sys TCTCs	non-sys TCTCs
$R=\frac{1}{3}$	-3.51dB, $(4,7)_o$	-3.75dB, $(17,15)_o$	-2.88dB, $(10,17)_o$
$R=\frac{1}{4}$	-5.07dB, $(2,3)_o$	-5.15dB, $(17,15)_o$	-4.68dB, $(10,17)_o$
$R=\frac{1}{5}$	-6.2dB, $(2,3)_o$	-6.3dB, $(17,15)_o$	-5.75dB, $(10,17)_o$
$R=\frac{1}{6}$	-7.1dB, $(2,3)_o$	-7.32dB, $(17,15)_o$	-6.72dB, $(10,17)_o$

Additionally, our BER performance comparisons are displayed in Figure 4.11 of Section 4.1.4. Likewise, the turbo cliff SNRs and the approximate BER floors of both MCTCs, as well as systematic and non-systematic TCTCs are summarized in Table 4.6, when considering the maximum complexity of 96.

Table 4.6: Turbo cliff SNRs and BER-floor values for both MCTCs, as well as systematic and non-systematic TCTCs, at the decoder complexity of 96.

	MCTCs	sys TCTCs	non-sys TCTCs
$R=\frac{1}{3}$	-2.5dB, ≈ 0	-3.0dB, 10^{-7}	-2.0dB, 10^{-6}
$R=\frac{1}{4}$	-4.5dB, ≈ 0	-4.5dB, 10^{-7}	-3.5dB, 10^{-6}
$R=\frac{1}{5}$	-5.5dB, ≈ 0	-5.5dB, 10^{-7}	-4.5dB, 10^{-6}
$R=\frac{1}{6}$	-6.8dB, ≈ 0	-6.5dB, 10^{-7}	-5.5dB, 10^{-6}

Based on the above results, we combined MCTCs with HARQ in Section 4.2, where we detailed the procedures of the proposed MCTC HARQ scheme and of the TCTC HARQ benchmarkers. As a conclusion, Figure 4.24 shows the structure that a system having a MCTC aided HARQ scheme, which is a concretion of the general system illustration of Figure 4.12.

Furthermore, Figures 4.21, 4.22 and 4.23 of Section 4.2.6 have shown that the MCTC aided HARQ scheme significantly improves both the PLR and throughput performance of the TCTC aided HARQ scheme having the same polynomial of $(2,3)_o$, which is achieved without a significant increase of complexity. On the other hand, if the TCTC aided HARQ scheme employs the polynomial of $(17,15)_o$, the MCTC aided HARQ scheme may significantly reduce the complexity, without a substantial degradation of the PLR and throughput.

Although the systematic TCTCs relying on the polynomial of $(17, 15)_o$ exhibit the closest performance to the DCMC capacity, they depend on a memory-3 polynomial to achieve this. By contrast, MCTCs only rely on the lowest-complexity memory-1 polynomial of $(2, 3)_o$, which essentially implies having a lower complexity. Furthermore, the quasi-static Rayleigh channel model has been routinely used in publications, which is appropriate for the scenarios, when the packet length is far shorter than the coherence time of the channel, while the interval between two transmissions is far larger. The coding rate R of turbo HARQ schemes varies for transmissions over the quasi-static Rayleigh channel when aiming for successfully transmitting each packet. This suggests that it is necessary to investigate the attainable performance for transmissions over a quasi-static Rayleigh channel, while aiming for the lowest possible overall complexity.

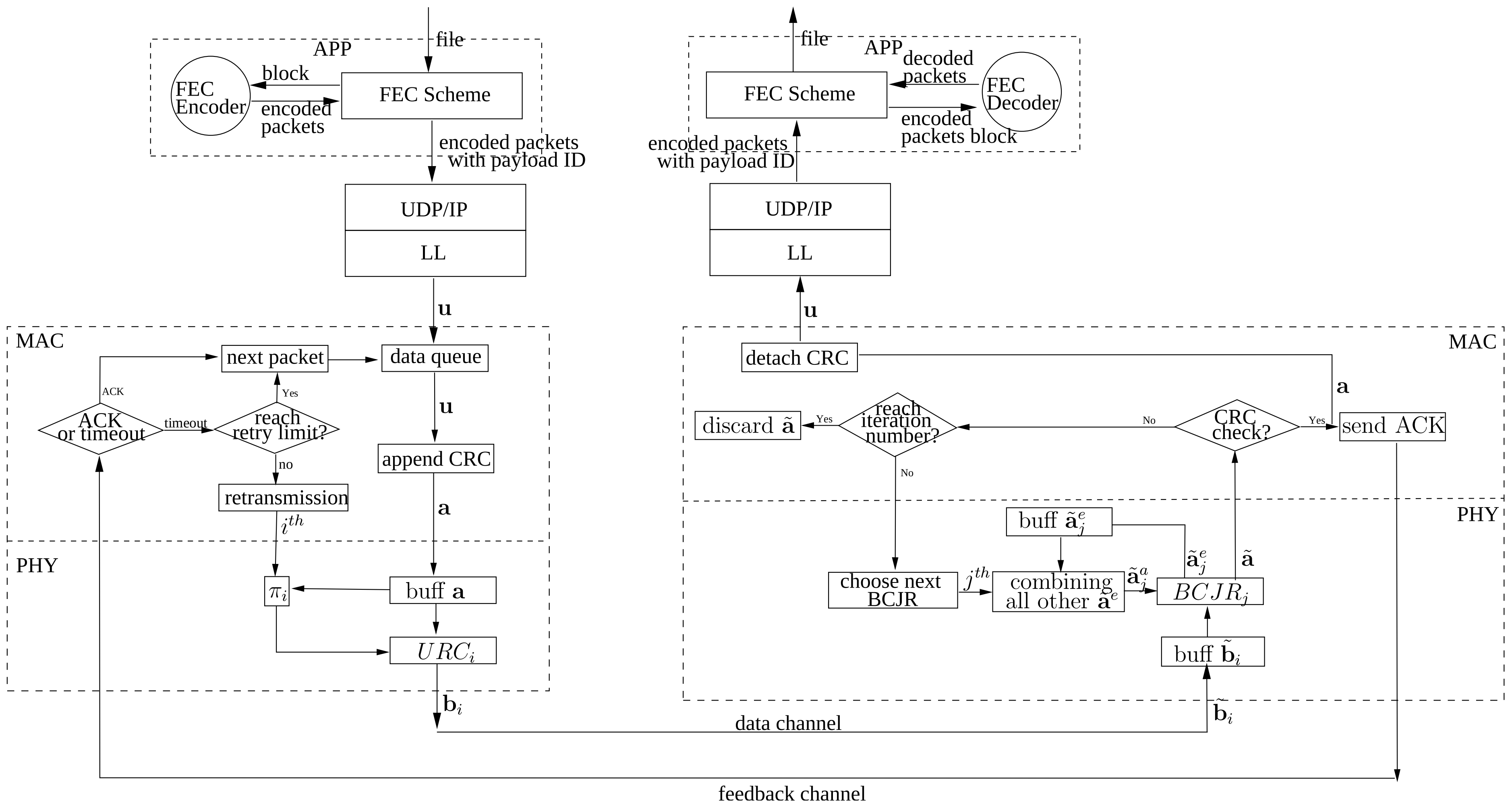


Figure 4.24: The concrete system structure with a MCTC aided HARQ scheme.

Chapter 5

Low-Complexity MCTC Aided Hybrid ARQ

In Chapter 4, Multiple Component Turbo Code (MCTC) aided Hybrid Automatic Repeat reQuest (HARQ) schemes have been demonstrated to achieve the desirable performance, when employing the low-complexity memory-1 generator polynomial of $(2,3)_o$. More explicitly, the employment of the lowest possible length polynomial implies having the lowest possible complexity of turbo decoding. As introduced in Section 4.2.6, the associated complexity is proportional to $K \cdot 2^m$, which is determined by the number of Bahl, Cocke, Jelinek and Raviv (BCJR) operations K, when the lowest-memory $m = 1$ polynomial is employed. At the time of writing, the most widely used stopping strategy for turbo decoding is that of fixing the number of BCJR iterations. In this chapter, we focus our attention on reducing the complexity of turbo HARQ schemes and propose an Early Stopping (ES) strategy in Section 5.1 and a Deferred Iteration (DI) method based on a Look-Up Table (LUT) in Section 5.2. Then, the system structure of Figure 4.24 is extended to Figure 5.1, which incorporates an MCTC aided HARQ scheme combined with the above-mentioned complexity reduction strategies, as introduced by the processing block printed in bold.

As seen in Figure 5.1, the file to be transmitted is firstly partitioned into several blocks at the application layer, where each block consists of a number of constant-length packets. These packets in one block are encoded by fountain codes, as described in Chapter 3. Each fountain-encoded packet passes through the User Datagram Protocol and Internet Protocol (UDP/IP) and Link layers down to the Medium Access Control (MAC) layer, where it will be buffered in a data queue. The MAC layer takes control of packet contentions as well as packet retransmissions. It fetches one packet each time from the data queue, and then appends Cyclic Redundancy Check (CRC) bits to it. The CRC encoded packet is conveyed to the Physical (PHY) layer for encoding by a rate-1 Recursive Convolutional Code (RSC). As soon as the resultant PHY-FEC encoded packet was sent out to the channel, the MAC layer triggers a timer and waits for an ACKnowledgement (ACK) message from the receiver. If the ACK message can be received in time, the transmitter sends out the next packet. Otherwise, the MAC layer retransmits the packet, when the retry limit is not reached. Each retransmitted packet will be differently interleaved before entering the RSC encoder.

At the receiver of Figure 5.1, the DI strategy is applied to determine whether the turbo decoding should be activated, following the reception of each incremental redundancy packet. The decoding commences when the received information is deemed to be sufficient by the DI strategy. For each BCJR operation, the sum of extrinsic LLRs gleaned from all other decoders will act as the *a priori* information. If the hard-decision output of the BCJR decoder satisfies the CRC, the check bits will be detached and then the uncoded packet will be transported up to the application layer, where the fountain decoder will collect them, until it can recover the original packets for a block. Meanwhile, the MAC layer sends an ACK message back to the transmitter. On the other hand, if the CRC fails, the ES strategy decides whether the receiver should curtail the decoding iterations. If not, the decoding process continues by choosing the next BCJR decoder. Otherwise, the receiver will wait for the next incremental redundancy transmission.

5.1 Early Stopping Assisted MCTC Aided Hybrid ARQ

HARQ has been shown to be an essential error control technique in communication networks [123, 124]. This combines ARQ and Forward Error Correction (FEC) for the sake of achieving a vanishingly low Bit Error Ratio (BER), which cannot be achieved by either method in isolation. A particular focus of recent research into HARQ schemes has been that of improving their throughput [124]. For this reason, combining the corrupted retransmitted replicas using turbo codes has received attention, as a benefit of their potential of approaching near-capacity operation. Souza's HARQ scheme [125] aided by systematic Twin-Component Turbo Codes (TCTC) and Chase combining, as well as our previously proposed MCTC HARQ scheme [23] have been shown to achieve high throughputs. The receivers of both schemes perform an iterative decoding process after each retransmitted packet is received. These iterative BCJR operations continue until the CRC is satisfied or a pre-defined number of BCJR iterations is reached, whereupon another retransmission is requested and iterative decoding recommences.

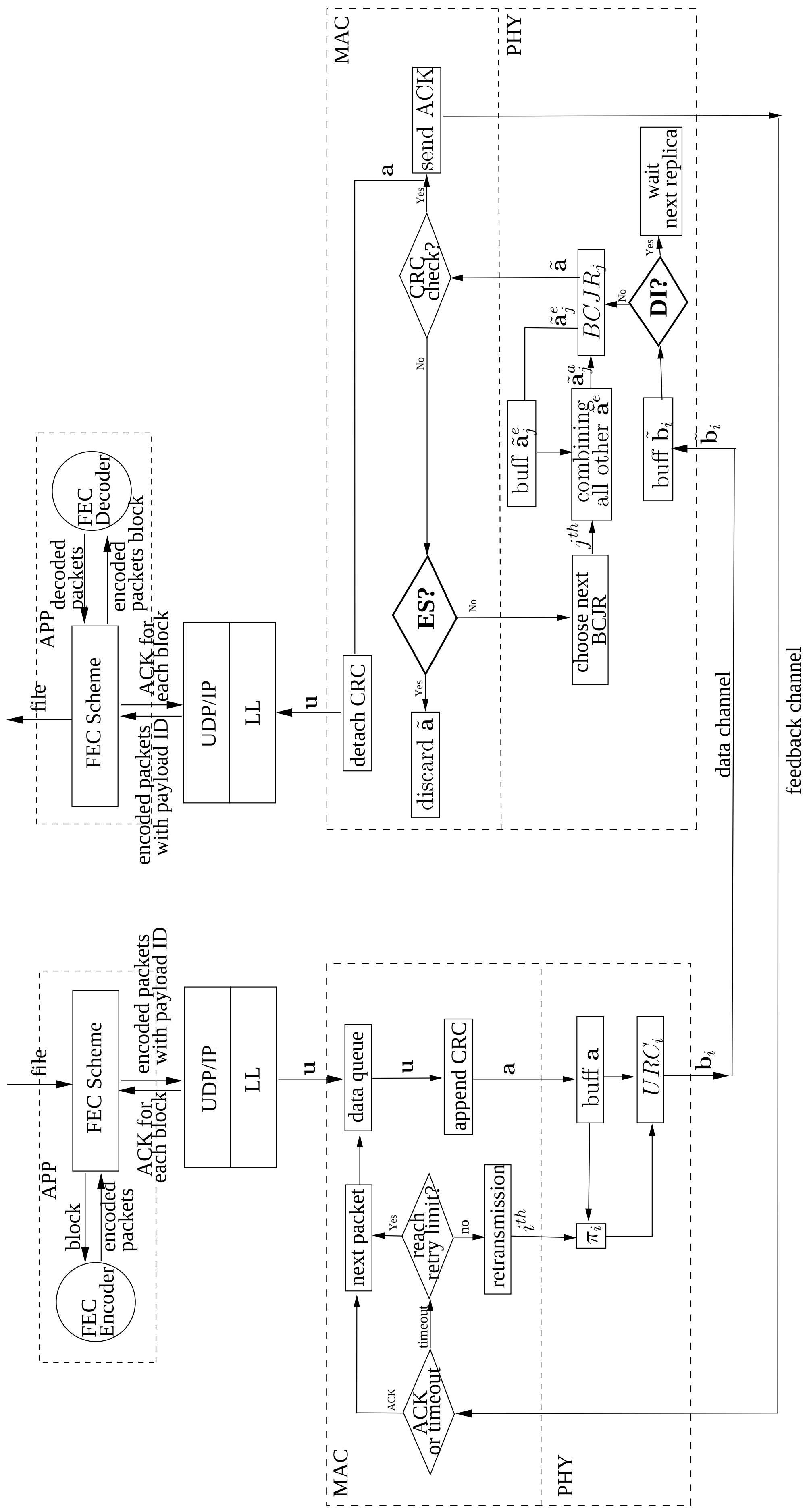


Figure 5.1: MCTC aided HARQ scheme relying on the ES and DI strategies, which was developed from the architecture of Figure 4.24.

However, the choice of the affordable number of BCJR iterations employed in these schemes significantly affects their performance. If the number is set too low, then unnecessary retransmissions may be requested and the throughput will suffer. If the number is set too high, then unnecessary BCJR operations will be performed and hence the complexity will be increased. In fact, we will demonstrate that different numbers of BCJR operations are appropriate at different stages of the HARQ receiver's operation. Since the appropriate number is difficult to predict in advance, an intelligent on-line ES approach is needed.

Many ES approaches have been proposed since turbo codes were invented. Most of these ES schemes [143–147] halt the iterative decoding process when the magnitude of Logarithmic Likelihood Ratios (LLRs) exceeds an appropriately chosen threshold. The earliest schemes quantify the cross entropy between the distributions of the *a posteriori* LLRs generated by the two BCJR decoders [143, 144]. The authors of [146] suggested estimating the mean of the LLRs' absolute values, while Li and Wu in [147] employed the cross correlation between the LLRs. In classic turbo codes operating without ARQ, these ES approaches determine the specific instant to curtail iterative decoding by considering the expected BER performance. However, in turbo HARQ schemes, the ES approach decides when to request a new incremental transmission, rather than increasing the number of BCJR operations for the current codeword, hence striking a tradeoff between the attainable throughput and the complexity imposed.

In this section, we propose a new ES approach that is specifically designed for turbo HARQ, based on the Mutual Information (MI) improvement after each BCJR operation. This is based on the observation that the MI is expected to gradually improve as iterative decoding proceeds, as quantified by the Extrinsic Information Transfer (EXIT) charts [48]. We refine this approach in order to strike an attractive tradeoff between the achievable throughput versus the complexity imposed for different packet lengths. We apply our ES approach to the above mentioned TCTC and MCTC aided HARQ schemes, demonstrating that the complexity is significantly decreased, while maintaining similar throughputs to those presented in [125] and [23].

The rest of this section is organized as follows. Section 5.1.1 details the schematic of our MCTC HARQ scheme and describes its encoding and decoding process. Section 5.1.2 describes and parameterizes our proposed ES approach. Then, the performance of the schemes operating both with and without our ES approach is characterized in Section 5.1.3.2. Section 5.1.6 concludes the section.

5.1.1 System Model

Again, the complexity of a tubo decoder may be quantified in terms of the total number of trellis states per bit [47], which is expressed as $Complexity = 2^m \cdot K$, where 'm'

is the number of memory elements employed in the convolutional encoders' generator polynomial, 2^m is the number of states in the corresponding trellis diagram and 'K' is the total number of BCJR operations performed during the iterative decoding process. Since the complexity exponentially increases with 'm', the generator polynomials of $(2, 3)_{Octal}$ having the lowest possible memory length of '$m = 1$' is desirable. However, for the systematic TCTC aided HARQ scheme, both the Packet Loss Ratio (PLR) and throughput performance degrade severely when adopting the polynomials of $(2, 3)_o$ [23], as we will demonstrate in Section 5.1.3.2. On the other hand, Section 4.1 and [24] have demonstrated that the polynomial pair of $(2, 3)_o$ is desirable for MCTC, since it is the one that achieves the lowest BER and facilitates the closest possible operation to the Discrete-input Continuous-output Memoryless Channel's (DCMC) capacity. Therefore, in this section, the MCTC is employed in the HARQ scheme in order to facilitate the employment of the low complexity generator polynomials $(2, 3)_o$.

It can be observed in Figure 5.2 for the medium to low Signal Noise Ratios (SNRs), for example the SNR range of $[-8dB, 8dB]$, that the PLRs are equal, when transmitting uncoded systematic bits and rate-1 encoded bits protected by our Unity Rate Code (URC) based on a RSC encoder over a quasi-static Rayleigh fading channel. Moreover, when considering the transmissions of rate-$\frac{1}{2}$ encoded bits from both the punctured systematic TCTCs and from the non-systematic TCTCs, they also have the same PLRs for this range of SNRs, as seen in Figure 5.3. The URC encoders combined with both the systematic and non-systematic TCTCs operating at these coding rates employ the octally represented generator polynomial of $(2, 3)_o$. Therefore, we may change the punctured systematic MCTC aided HARQ schemes of Figure 4.19 to non-systematic MCTC aided HARQ. This will avoid using the puncturing operations without affecting the PLR and throughput performance. Furthermore, the modified Type-III HARQ scheme may obviously improve the attainable throughput performance at high SNRs, albeit this has not been investigated in this thesis.

The transmitter schematic of MCTC aided HARQ is illustrated in Figure 5.4, where the input bits $\mathbf{a}$ are obtained by appending CRC to the information bits. Then, the differently interleaved copies $\mathbf{a}_i$ are forwarded to recursive URC encoders. Note that these interleaving operations π_1,π_1,...,π_{i-1} may be realized by the same interleaver design, without degrading the system performance. This approach has the advantage of reducing the implementational complexity. The encoded bits $\mathbf{b}_i$ are sequentially transmitted at regular intervals, until a positive ACK message is received.

The receiver schematic is displayed in Figure 5.5 and the flow chart of the decoding process is illustrated in Figure 5.6. As seen in Figure 5.6, the receiver performs a single BCJR operation following the reception of the first encoded bit sequence $\tilde{\mathbf{b}}_1$,

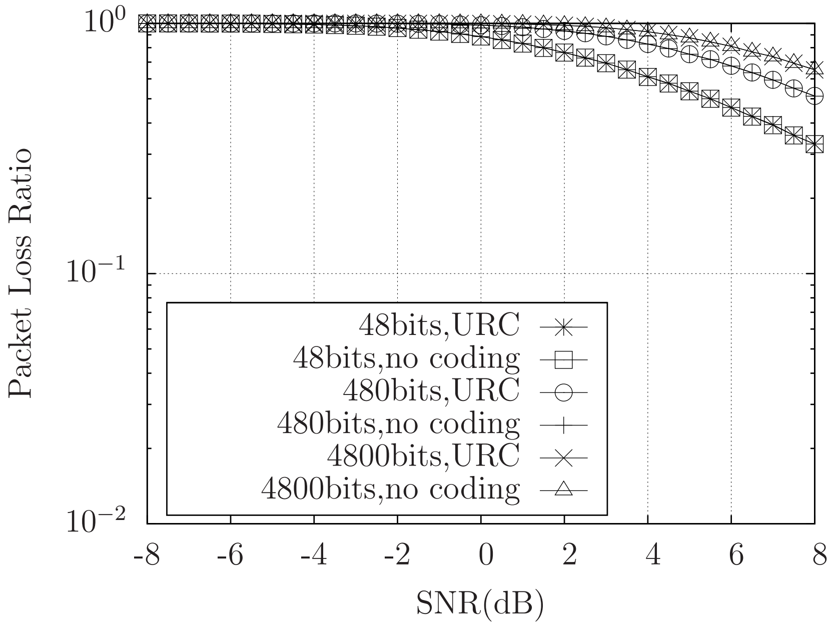


Figure 5.2: PLR versus SNR performance for the direct transmissions without coding, and for the transmissions of rate-1 URC encoded packets using the schematic of Figure 2.3 but without systematic bits over a quasi-static Rayleigh channel.

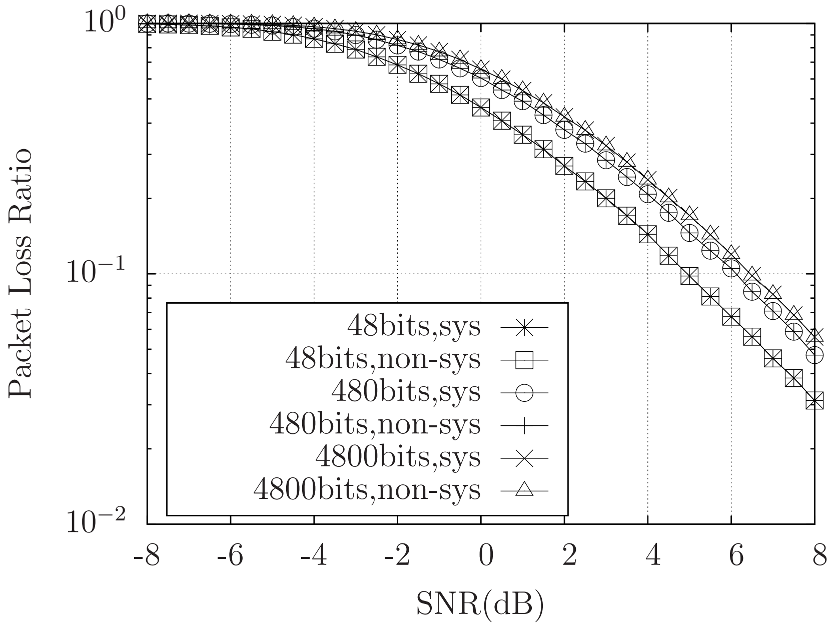


Figure 5.3: PLR versus SNR performance for the transmissions of rate-$\frac{1}{2}$ encoded packets over a quasi-static Rayleigh channel, respectively using the schematics of Figure 2.3 and Figure 4.4 with $N = 2$.

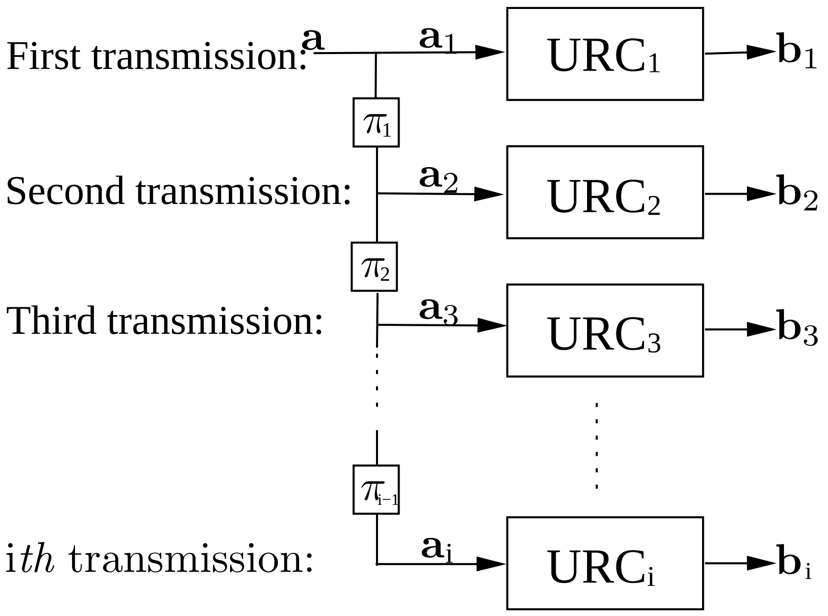


Figure 5.4: The MCTC encoder structure used in the HARQ regime of Figure 5.1.

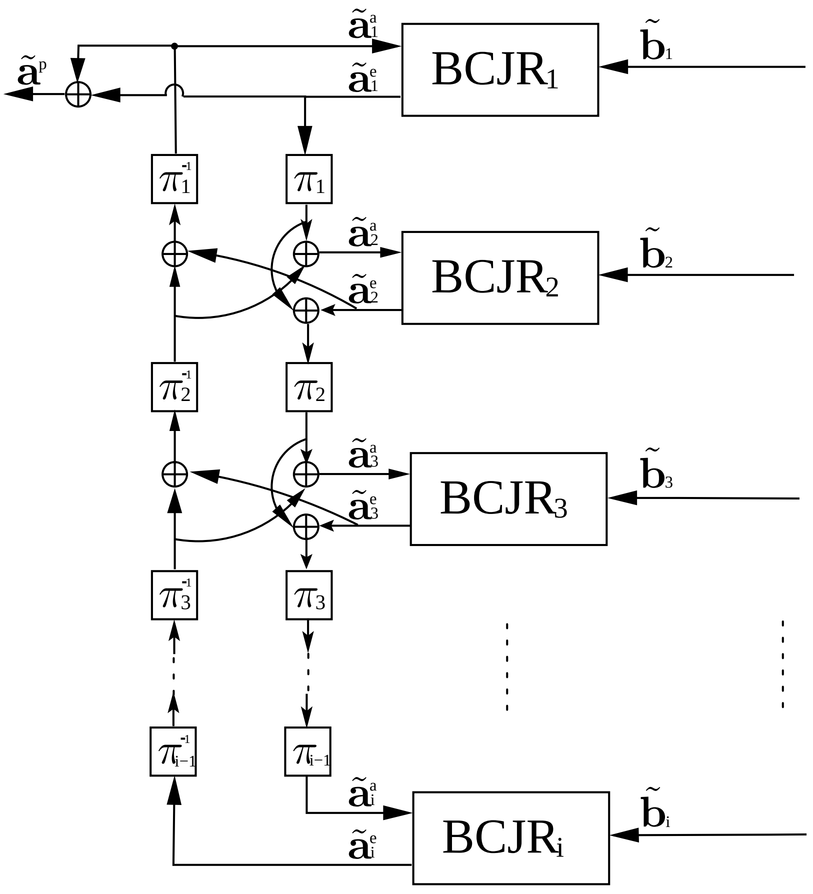


Figure 5.5: The MCTC decoder structure used in the HARQ regime of Figure 5.1 after i IR-transmissions.

when the *a priori* LLRs $\tilde{\mathbf{a}}_1^a$ of the BCJR$_1$ have zero values. A hard decision is carried out on the basis of the output *a posteriori* LLRs $\tilde{\mathbf{a}}^p$ in order to obtain the decoded bits. If the CRC fails for the decoded bits, the receiver will wait until the transmitter times out and hence the transmitter transmits the second encoded bit sequence $\tilde{\mathbf{b}}_2$. Turbo decoding commences after the reception of $\tilde{\mathbf{b}}_2$. Generally, whenever a set of encoded LLRs $\tilde{\mathbf{b}}_i$ $(i \geq 2)$ is received during the *ith* Incremental Redundancy (IR)-transmission, the corresponding decoder BCJR$_i$ is activated. It is immediately operated and becomes available for operation during the subsequent iterative decoding processes, yielding the i-component MCTC shown in Figure 5.5. Whenever a decoder BCJR$_i$ is operated, the *a priori* uncoded LLRs $\tilde{\mathbf{a}}_i^a$ are obtained by interleaving and summing the extrinsic uncoded LLRs provided by the most recent operation of all other activated decoders. The decoder BCJR$_i$ combines the *a priori* uncoded LLRs $\tilde{\mathbf{a}}_i^{\mathrm{a}}$ with the *a priori* encoded LLRs $\tilde{\mathbf{b}}_i$, in order to obtain the extrinsic uncoded LLRs $\tilde{\mathbf{a}}_i^e$. Following this, the *a posteriori* uncoded LLRs $\tilde{\mathbf{a}}^p$ are obtained by interleaving and then summing the extrinsic uncoded LLRs provided by the most recent operation of all activated decoders. A hard decision is made for each bit on the basis of the *a posteriori* uncoded LLRs $\tilde{\mathbf{a}}^p$ and then the CRC check is performed. If the CRC fails and our proposed ES approach (represented by the blocks in the dashed polygon of Figure 5.6 and detailed in Section 5.1.2) is not satisfied, the next BCJR decoder to be operated is selected as the one that was operated earliest, because it is the one that is most likely to provide the largest MI increment. As a result, a particular MI value can be reached using less BCJR operations, compared to other selection strategies. When the proposed ES approach is satisfied while the CRC still fails, a new IR-transmission is requested and the new round of turbo decoding is repeated until the corrected decoded bits are obtained, or the retry limit is reached.

5.1.2 Early Stopping Approach for the MCTC Aided HARQ Scheme

The complexity metric of Section 5.1.1 suggests that once the $(2,3)_o$ polynomials have been adopted, the total number of BCJR operations 'K' performed during turbo HARQ decoding is the key factor that determines the complexity. Our ES approach aims for an efficient yet low-complexity design[1], having a minimum number of parameters in order to maximize the generality and applicability of the algorithm. There are two elements to our ES approach, both of which employ MI thresholds. One MI threshold is proposed to dynamically adjust the number of BCJR operations that are performed following each IR-transmission. We refer to this threshold as the 'convergence threshold'. The

[1]A complex strategy may outperform this efficient design at the cost of a disproportionately higher complexity. For example, in Section 5.2 will determine whether to trigger the turbo decoding following each transmission, based on a prediction whether the MI of the current MCTC may get close to 1 according to a lookup table.

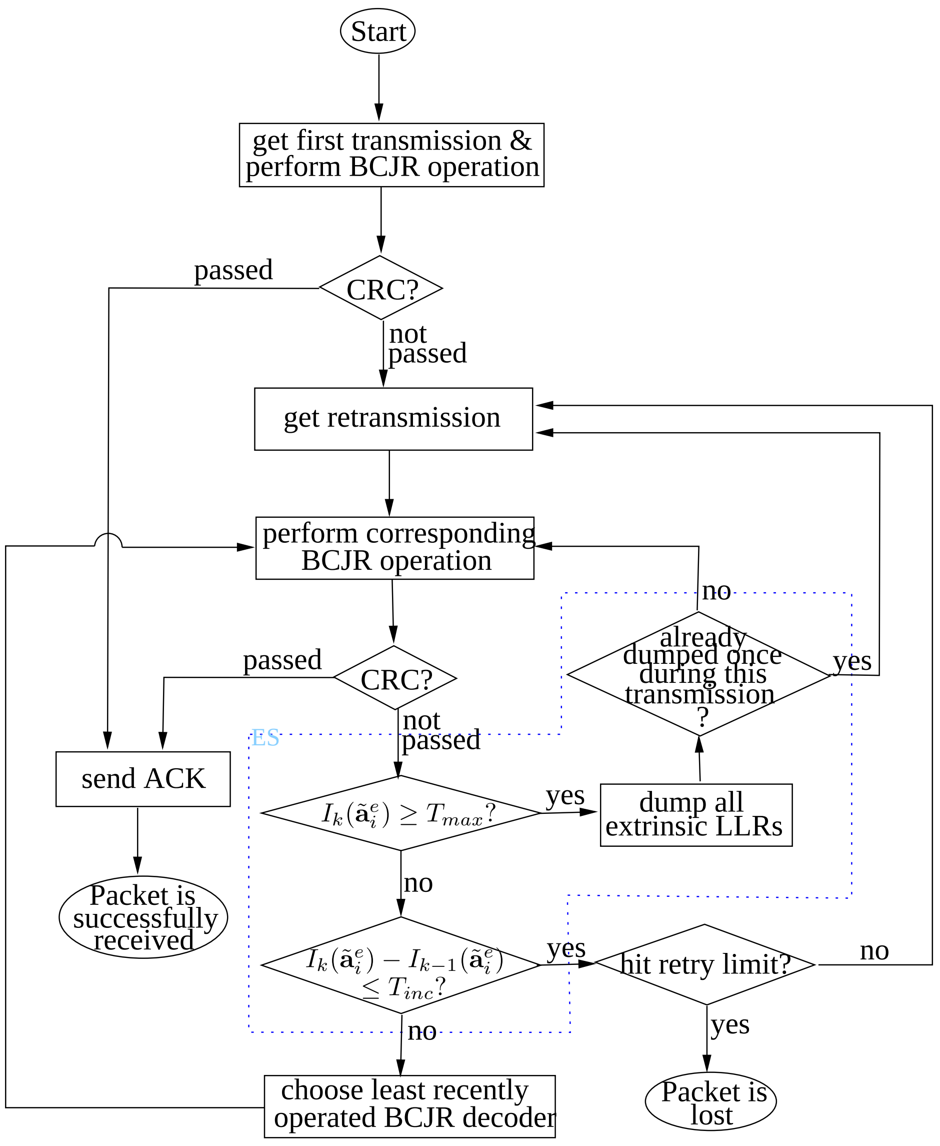

Start
get first transmission & perform BCJR operation
CRC?
passed
not passed
get retransmission
perform corresponding BCJR operation
CRC?
passed
not passed
ES
send ACK
Packet is successfully received
$I_k(\tilde{\mathbf{a}}_i^e) \geq T_{max}$?
yes
dump all extrinsic LLRs
already dumped once during this transmission ?
no
yes
no
$I_k(\tilde{\mathbf{a}}_i^e) - I_{k-1}(\tilde{\mathbf{a}}_i^e) \leq T_{inc}$?
yes
hit retry limit?
no
yes
no
choose least recently operated BCJR decoder
Packet is lost

Figure 5.6: The flow chart of the decoding process conceived for the MCTC aided HARQ regime of Figure 5.1.

second MI threshold is proposed to guarantee a zero PLR for the MCTC aided HARQ scheme relying on an infinite number of IR-transmissions. We refer to this threshold as the 'dumping threshold'. These two thresholds are discussed in the following sections.

5.1.3 Analysis of the Convergence and Dumping Thresholds

5.1.3.1 Convergence Threshold

As mentioned in Chapter 2, the MI of an LLR sequence provides a quantitative indication of our confidence in the corresponding hard decisions. Equation 2.17 maps the sequence of LLRs to a confidence metric confined to the range of $[0, 1]$, where 0 implies no confidence, 1 means absolute confidence. It is widely recognized that EXIT charts illustrate the MI transfer between the components of turbo codes. The bit-by-bit Monte-Carlo decoding trajectory steps within the EXIT chart tunnel may be treated as the MI increment after each iteration. The MI increment typically becomes smaller and smaller when the trajectory approaches the convergence point in the EXIT charts, where the turbo iterations should be stopped. Specifically, whenever a BCJR decoder is operated, the MI increment is expressed by $I_k(\tilde{\mathbf{a}}_i^e) - I_{k-1}(\tilde{\mathbf{a}}_i^e)$, where $I_k(\tilde{\mathbf{a}}_i^e)$ is the extrinsic MI obtained after the kth operation of BCJR$_i$ and $I_0(\tilde{\mathbf{a}}_i^e)$ is assumed to be 0. Therefore, when the MI increment falls below a certain threshold, this may be considered as the stopping criterion, which we refer to as the convergence threshold T_{inc}.

The decoding iterations continue, as long as the MI increment is higher than T_{inc}, while an IR-transmission is requested when the increment drops below T_{inc}, as shown in Figure 5.6. Different values of T_{inc} result in different tradeoffs between the attainable throughput and the complexity imposed. Since the MI value varies from 0 to 1.0, the MI increment must also be confined to this range. When we have $T_{inc} = 0$, the receiver will only request a new IR-transmission when it is sure that it is not possible to satisfy the CRC without doing so, because for example a high number of BCJR operations have been performed and hence convergence to the wrong legitimate codeword is achieved. As a result, this approach minimizes the number of IR-transmissions at the cost of imposing a potentially high complexity. When T_{inc} assumes the maximum value of 1.0, the receiver will request new IR-transmissions as often as it can, in order to minimize the number of BCJR operations that are performed before the CRC is satisfied. As a result, this approach minimizes the complexity at the cost of having a low throughput. In Section 5.1.3.2, we will observe the relationship between the attainable throughput and the complexity imposed by using different T_{inc} values for three typical packet lengths.

5.1.3.2 Dumping Threshold

Turbo aided HARQ schemes have an inherent problem, which prevents them from guaranteeing a vanishingly low PLR, even when an infinite number of IR-transmissions is permitted. Specifically, turbo decoders are capable of converging to a legitimate bit-sequence for an MI of 1, but they may still output the wrong bit sequence. This occurs when the received soft-sequences are ‘closer’ to the incorrect bit sequence than to the correct sequence. This effect accounts for the error floor in the BER performance of turbo decoders at high SNRs, where the EXIT chart tunnel is wide open. In the MCTC aided HARQ scheme, the decoding of the earlier few IR-transmissions may converge towards the wrong but legitimate bit sequence associated with a high confidence, like in a standard turbo code. In this case, the CRC fails and hence further IR-transmissions are requested. Unfortunately, the influence of these later IR-transmissions may become insufficiently decisive to guide the decoder away from the wrong bit sequence associated with MI $\approx$ 1 and towards the correct one, regardless of how many IR-transmissions are received. As mentioned in Section 5.1.1, when an IR-transmission is received, the decoder first decodes this most recent IR-transmission. In a conventional regime following simple logic, the previously received codewords would provide *a priori* information for this most recent IR-transmission. However, in this scenario the previously received codewords may have a high extrinsic MI as a result of a comparatively high number of iterations, which is the reason for the relatively low influence for the most recent IR-transmissions. In other words, this approach activates the BCJR decoders corresponding to the earlier IR-transmissions more frequently than the other decoders. This lends these earlier IR-transmissions a higher influence, allowing them to persistently sway the iterative decoding process towards the incorrect uncoded bit sequence associated with MI $\approx$ 1. Therefore, repeated IR-transmissions will be requested again and again. If the IR-transmission retry limit is high, both the throughput and the complexity suffer significantly. Furthermore, the CRC may never be satisfied.

A dumping threshold T_{max} is employed to circumvent this problem as detailed below. When the MI of the extrinsic LLRs $\tilde{\mathbf{a}}^e$ obtained after some BCJR operation become higher than T_{max}, while the CRC has not been satisfied, all extrinsic LLRs are reset to zero and an improved iterative decoding procedure is activated, since the decoder is now in possession of potentially numerous received replicas of the original information. More explicitly, until this event the early replicas may have activated a higher number of iterations and hence may have driven the MI to high values, which lent them an increased influence over the more recent replicas. Recall that IR-transmission was only requested, when the CRC of the earlier ones failed. At this stage, we hence reactivate iterative decoding by exchanging extrinsic information amongst all replicas, lending them an equal influence, as a benefit of the above-mentioned LLR dump operation. For most cases, this re-initialized iterative process may lead to a better chance of successful decoding. In the exceptionally rare case that the MI obtained following a BCJR operation again becomes larger than T_{max} without satisfying the CRC, the receiver dumps all extrinsic LLRs once more and request a new IR-transmission. This process is repeated until the packet is correctly decoded. Assuming a perfectly reliable CRC, this allows a zero PLR to be attained, provided that the IR-transmission retry limit is sufficiently high.

As with T_{inc}, there is a tradeoff associated with the specific setting of T_{max}. If T_{max} assumes a large value close to 1, the receiver may require more IR-transmissions than necessary to ensure that the CRC will be satisfied, unless the extrinsic LLRs are dumped and the decoder is re-initialized. As a result, both the throughput and complexity suffer compared to adopting a smaller value of T_{max}. On the other hand, if T_{max} is set too small, the receiver may mistakenly think that the CRC will not be satisfied by performing more BCJR operations. As a result, 'genuine' LLRs will be dumped and more IR-transmissions will be requested, degrading both the throughput and complexity. Thus, the most appropriate value of T_{max} solves the wrong decision problem as well as approaching the optimum throughput at a low complexity. In Section 5.1.3.2, we will determine the most desirable value of T_{max}.

5.1.4 Offline Simulation for Determining the Stopping Thresholds

In this section, we investigate the variation of the MCTC HARQ throughput and complexity with the value of stopping thresholds T_{inc} and T_{max}.

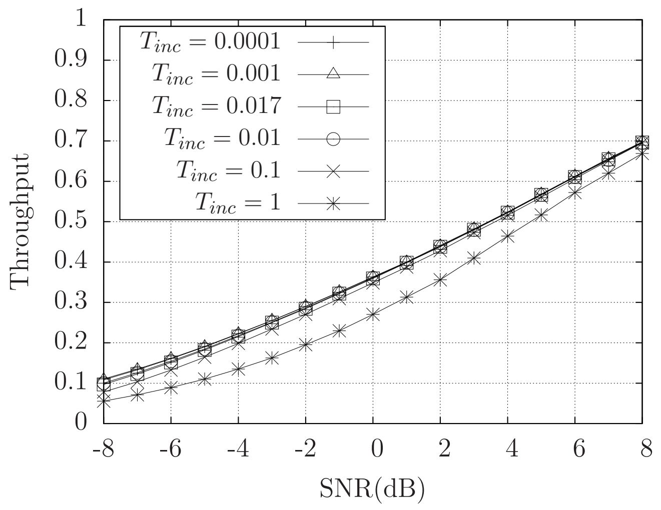
During our simulations, we determined the throughput and complexity that results for each combination of $T_{inc} \in \{1.0, 0.1, 0.01, 0.001, 0.0001\}$ and $T_{max} \in \{0.9, 0.99, 0.999\}$, at a range of channel SNRs and packet lengths. For the sake of observing the full influence of different stopping thresholds, an unlimited number of IR-transmissions was allowed in order to ensure that a PLR of zero was attained at all SNRs considered. Since the two stopping thresholds depend on the packet length, they were fitted to 100 bits, 1000 bits and 10000 bits in our simulations. In each simulation, a statistically significant number of packets was transmitted through a quasi-static Rayleigh fading channel using Binary Phase-Shift Keying (BPSK) modulation. The parameters of this offline training are summarized in Table 5.1.

Table 5.1: The parameters of investigating the variation of the MCTC HARQ throughput and complexity with the value of stopping thresholds T_{inc} and T_{max}.

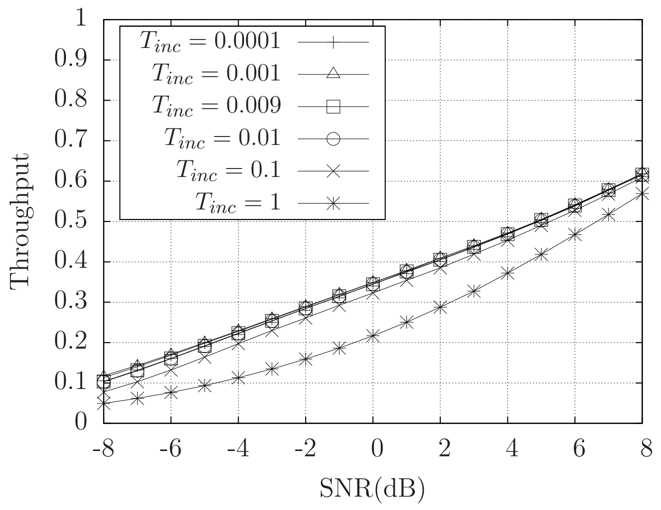
Retry Limit	infinite
Packet Length	100-bit, 1000-bit, 10000-bit
T_{inc}	1.0, 0.1, 0.01, 0.001, 0.0001
T_{max}	0.9, 0.99, 0.999
Modulation Scheme	BPSK
Channel Type	quasi-static Rayleigh fading

For the sake of conciseness, we detail only the process of selecting the preferred values of the convergence threshold T_{inc}, as provided in Table 5.2 for packets comprising 100, 1000 and 10000bits. The corresponding preferred values of the dumping threshold T_{max} of Table 5.2 were selected using an analogous process, which is not detailed in this section. In summary, when the dumping threshold T_{max} is set to 0.99 for the 100-bit packets, and to 0.999 for both the 1000-bit and 10000-bit packet lengths, we obtain the highest throughput and the lowest complexity for the scenarios considered.

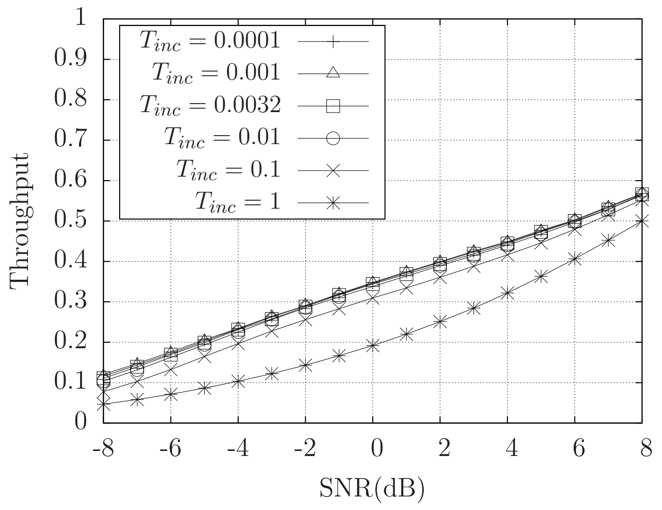
Figure 5.7 plots the throughput versus SNR for different convergence thresholds T_{inc} and packet lengths, when employing the corresponding dumping threshold values



(a) 100bits, $T_{max} = 0.99$

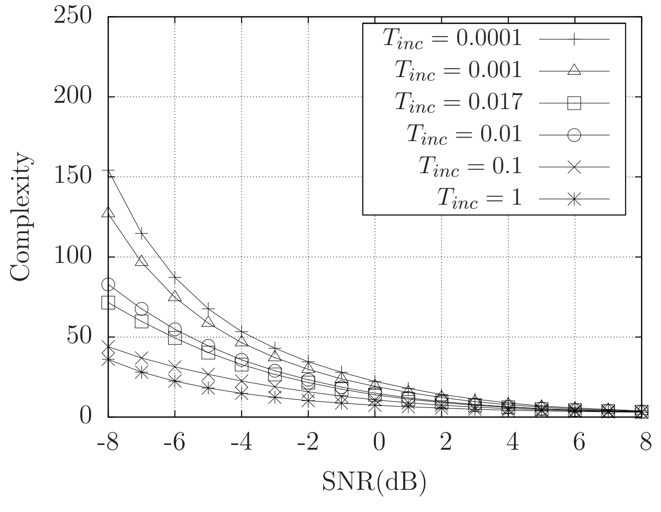


(b) 1000bits, $T_{max} = 0.999$

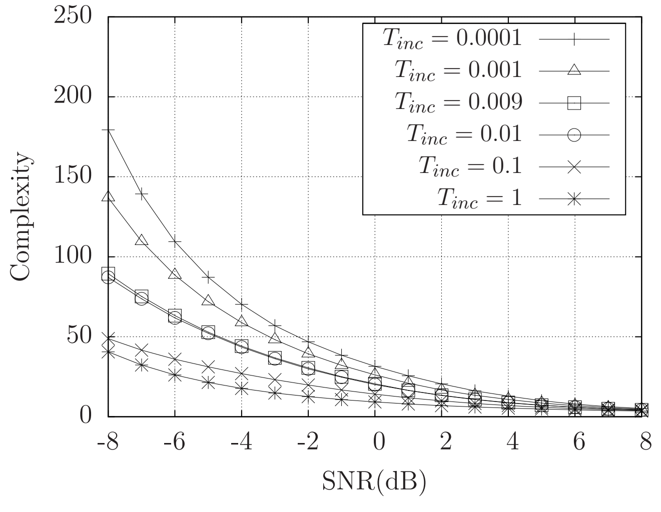


(c) 10000bits, $T_{max} = 0.999$

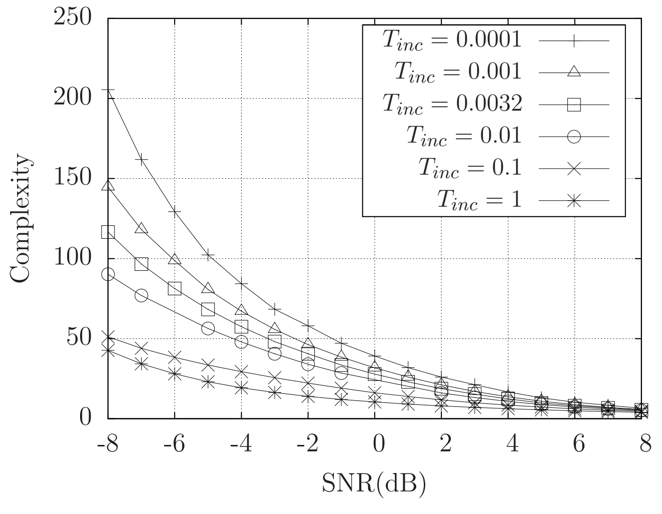
Figure 5.7: Throughput in conjunction with different T_{inc} values for three different packet lengths. Transmissions take place over a quasi-static Rayleigh channel with an infinite number of retransmissions for a given packet, until it is correctly received. The schematic of Figure 5.4, 5.5, and 5.6 as well as the parameters of Table 5.1 were used.



(a) 100bits, $T_{max} = 0.99$



(b) 1000bits, $T_{max} = 0.999$



(c) 10000bits, $T_{max} = 0.999$

Figure 5.8: Complexity in conjunction with different T_{inc} values for three different packet lengths. Transmissions take place over a quasi-static Rayleigh channel with an infinite number of retransmissions for a given packet, until it is correctly received. The schematic of 5.4, 5.5, and 5.6, as well as the parameters of Table 5.1 were used.

Table 5.2: The preferred thresholds for different packet lengths.

Packet Length	100 bits	1000 bits	10000 bits
T_{max}	0.99	0.999	0.999
T_{inc}	0.017	0.009	0.0032

T_{max} of Table 5.2. Here the throughput is defined as the ratio of the number of information bits delivered to the total number of transmitted bits. Figure 5.8 provides the corresponding complexity versus SNR curves, where the complexity metric was defined in Section 5.1.1.

Figures 5.7 and 5.8 show that as the convergence threshold T_{inc} was increased, both the throughput and the complexity was reduced for all the packet lengths considered, as predicted in Section 5.1.3. However, the throughput reduction was relatively modest, while the complexity was significantly reduced, as T_{inc} was increased from 0.0001 to 0.01. Since achieving a high throughput is the design target of HARQ schemes, we specify an average throughput reduction of 0.005 as the maximum loss that can be tolerated, when optimizing the corresponding T_{inc} values for all three packet lengths. We appropriately adjusted T_{inc} values in order to obey this throughput reduction limit. Quantitatively, we found that the preferred values were $T_{inc} = 0.017$ for the 100-bit, $T_{inc} = 0.009$ for the 1000-bit and $T_{inc} = 0.0032$ for the 10000-bit packet lengths, which yielded the lowest complexity. The curves of the throughput and complexity corresponding to these values were also included in Figures 5.7 and 5.8.

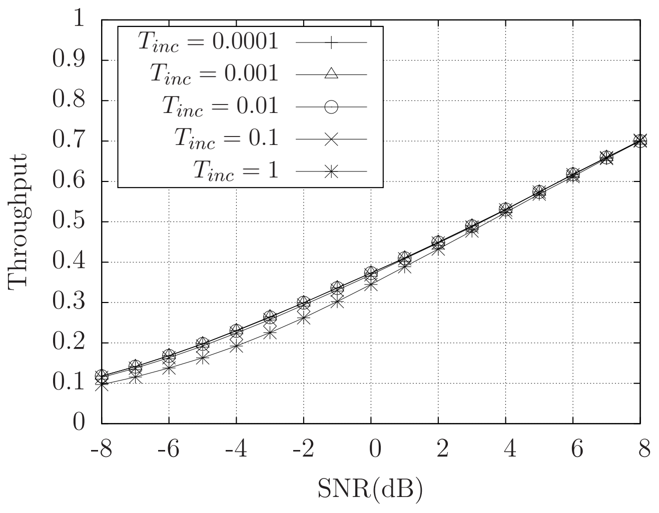
5.1.5 Performance Simulation

In order to demonstrate the advantages of our ES approach, the PLR, throughput and complexity metrics are compared for Souza's systematic TCTC aided HARQ scheme [125] and our MCTC aided HARQ scheme.

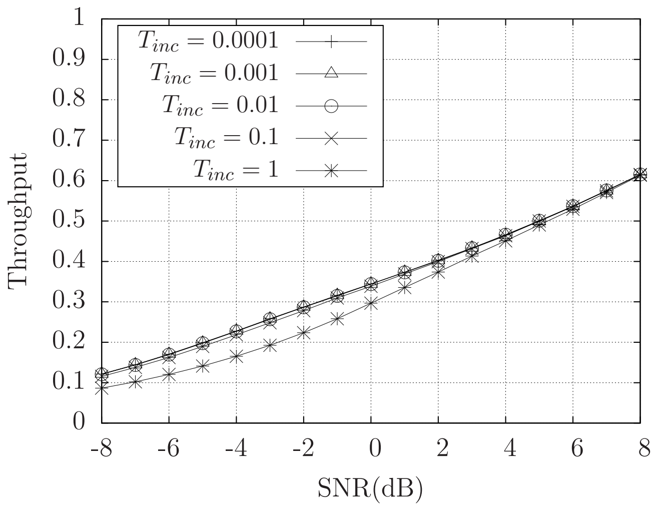
The systematic TCTC of Souza's scheme employs the polynomials of $(17, 15)_o$ and uses a transmission limit of $r = 6$. During the $r = 6$ transmissions, the transmitted sequences are equivalent to $\mathbf{a}$, $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{a}$, $\mathbf{b}_1$ and $\mathbf{b}_2$ from Figure 5.4, where $\mathbf{a}$ is the vector of systematic bits, while $\mathbf{b}_1$ and $\mathbf{b}_2$ are the encoded bits generated by a pair of URC encoders that are separated by an interleaver. This is in contrast to Souza's original scheme, in which $(\mathbf{a}, \mathbf{b}_1)$ are transmitted together during the first transmission, and the symbol energy E_s is doubled for the subsequent transmissions. However, we introduced this modification in order to maintain the same coding rate for the two HARQ schemes and to maintain the same symbol energy for each transmission. Furthermore, the receiver commences iterative decoding after the third transmission. The LLRs obtained from the incremental transmissions of two replicas of the same bit sequence are added together. For example, when the LLRs of the systematic bits $\mathbf{a}$ are soft-demodulated during the fourth transmission, they will be combined with the systematic LLRs obtained from the first transmission. Then, the twin-component turbo decoding recommences, preserving any extrinsic LLRs that were obtained following the third transmission. The same procedure continues for the fifth and sixth transmissions, if they are needed.

The packet length is set to a modest value of 1000 bits for both HARQ schemes. For the sake of fair comparison, the transmission limit is set to $r = 6$ for the MCTC aided HARQ scheme. Hence, some packet loss events are expected at low SNRs according to the flow chart of Figure 5.6. As suggested in [125], Souza's scheme performs a pre-defined number of 10 BCJR operations before stopping iterative decoding following each transmission. Correspondingly, our MCTC aided HARQ scheme uses the two stopping thresholds discussed in Section 5.1.2. Given that the permitted throughput reduction is 0.005, according to Table 5.2, the preferred dumping threshold T_{max} is 0.999 and the convergence threshold T_{inc} is 0.009 for the 1000-bit length packets. We also apply the proposed ES approach to Souza's HARQ scheme in order to evaluate our technique's generality. Likewise, the preferred convergence and dumping thresholds T_{inc} and T_{max} can be obtained using simulations similar to those described in Section 5.1.2. As a result, Figures 5.9 and 5.10 illustrate the throughput and complexity trends associated with different T_{inc} values from the set $\{1.0, 0.1, 0.01, 0.001, 0.0001\}$ for 100-bit, 1000-bit and 10000-bit interleaver lengths for Souza's systematic TCTC HARQ scheme, which employs the $(17, 15)_o$ octally represented generator polynomials. Here, we have not investigated the preferred T_{inc} values for the 100-bit and 10000-bit scenarios, but the preferred T_{inc} value for 1000-bit length was found to be $T_{inc} = 0.1$, when the average throughput loss is limited to 0.005. Furthermore, the resultant preferred T_{max} value for Souza's systematic TCTC HARQ scheme was found to be $T_{max} = 0.999$ for the 1000-bit sequence length, when the average throughput loss is limited to 0.005. Additionally, since Souza's original HARQ scheme adopts the memory length $m = 3$ generator polynomials of $(17, 15)_o$, we consider the effect of the code's memory length on the complexity by also employing the modified polynomials of $(2, 3)_o$ for Souza's scheme. For further exploring the ES advantages, our MCTC aided HARQ scheme was also configured to perform a pre-defined number of 10 BCJR operations following each transmission, in order to obtain an additional benchmarker that does not employ ES. All the transmissions are over a quasi-static Rayleigh fading channel using BPSK modulation. The simulation parameters are concluded in Table 5.3.

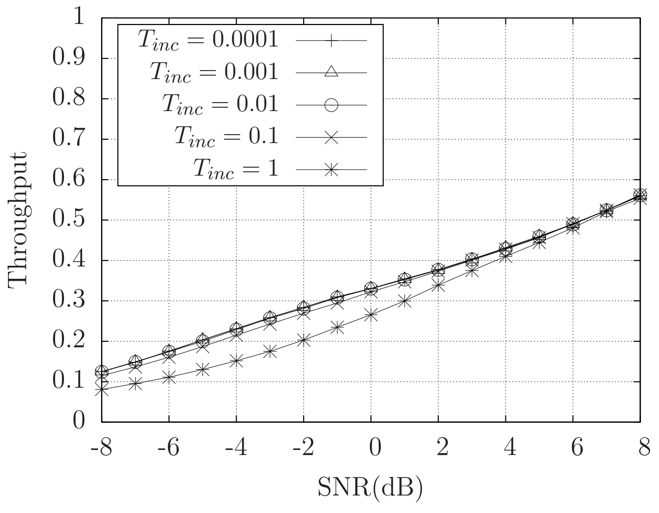
Figure 5.11 illustrates the attainable PLR versus SNR performance, while Figure 5.12 shows the throughput versus SNR, where the throughput has the same definition



(a) 100bits, $T_{max} = 0.99$

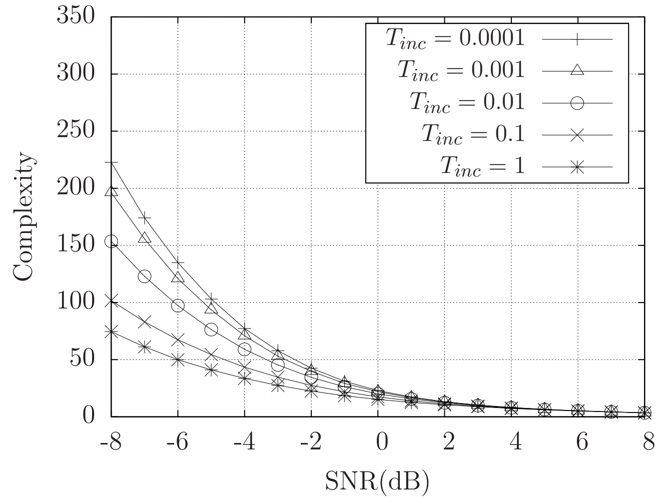


(b) 1000bits, $T_{max} = 0.999$

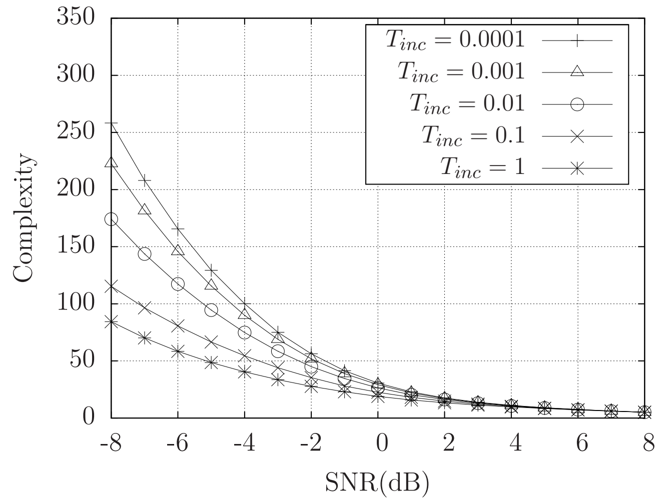


(c) 10000bits, $T_{max} = 0.999$

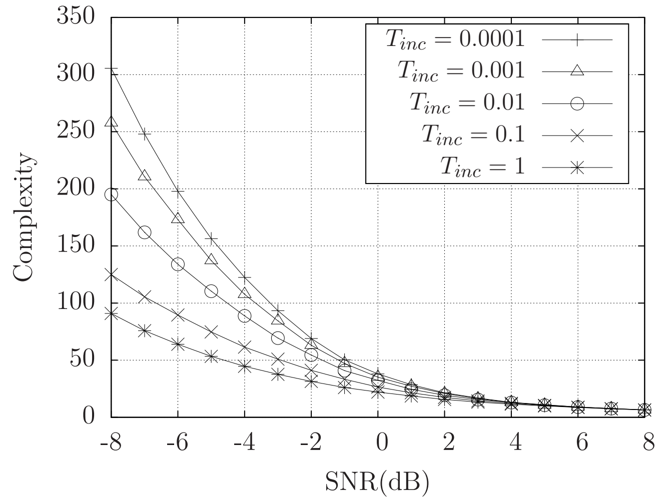
Figure 5.9: Throughput with different T_{inc} values for three different packet lengths for Souza's systematic TCTC HARQ scheme. Transmissions take place over a quasi-static Rayleigh channel with an infinite number of retransmissions for a given packet, until it is correctly received. The schematic of Figures 4.17 and 4.18, as well as the parameters of Table 5.1 were used.



(a) 100bits, $T_{max} = 0.99$



(b) 1000bits, $T_{max} = 0.999$



(c) 10000bits, $T_{max} = 0.999$

Figure 5.10: Complexity with different T_{inc} values for three different packet lengths for Souza's systematic TCTC HARQ scheme. Transmissions take place over a quasi-static Rayleigh channel with an infinite number of retransmissions for a given packet, until it is correctly received. The schematic of Figures 4.17 and 4.18, as well as the parameters of Table 5.3 were used.

Table 5.3: The simulation parameters for comparing the PLR, throughput and complexity performance for Souza's systematic TCTC aided HARQ scheme and our MCTC aided HARQ scheme.

Retry Limit	6
Packet Length	1000-bit
Stopping Strategy	ES (or) 10 BCJR operations
T_{inc}	0.009 for the MCTC HARQ scheme 0.1 for Souza's scheme
T_{max}	0.999 for both schemes
Modulation Scheme	BPSK
Channel Type	quasi-static Rayleigh fading

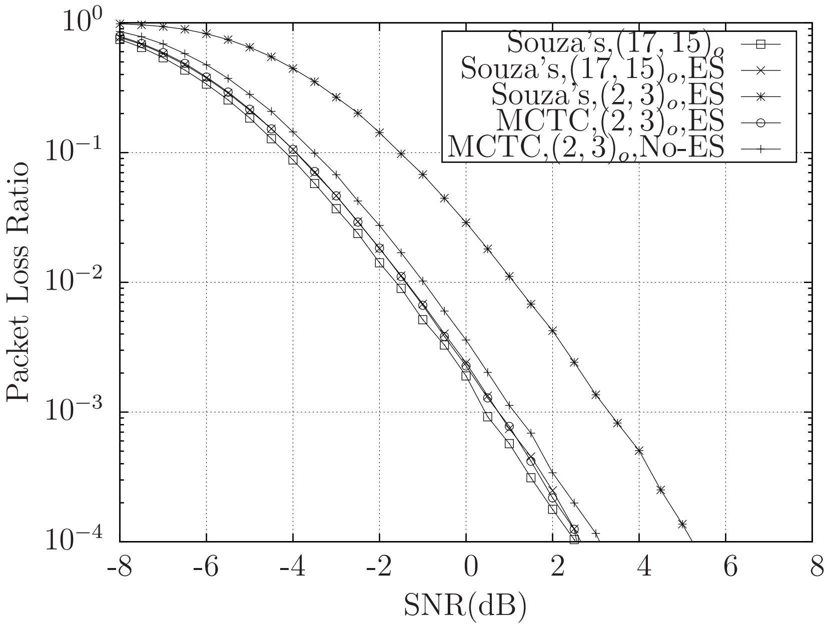


Figure 5.11: The packet loss ratio versus SNR performance for transmission over a quasi-static Rayleigh channel, with the retransmission limit set to 6 and the packet length set to 1000 bits. The schematic of 5.4, 5.5 and 5.6 for the MCTC HARQ scheme, the schematic of Figures 4.17 and 4.18 for Souza's scheme, as well as the parameters of Table 5.1 were used.

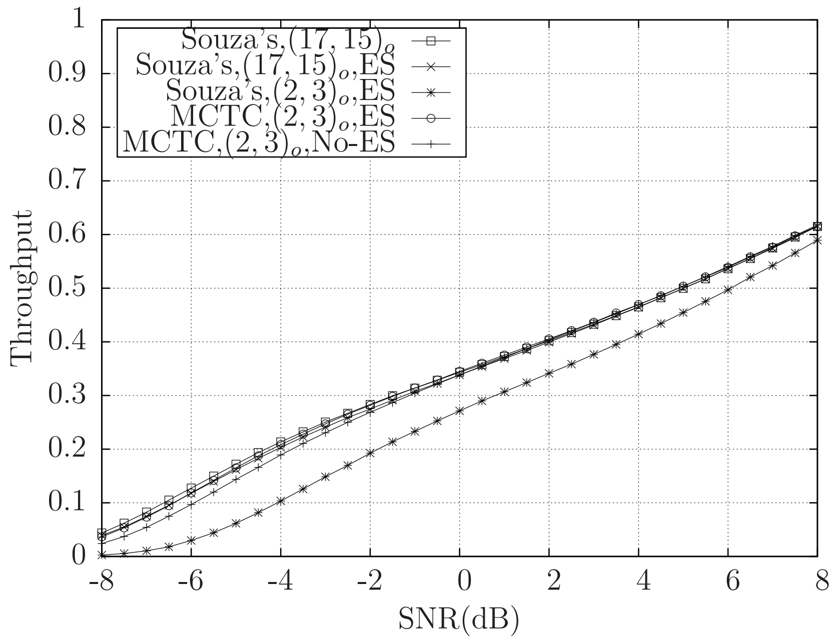


Figure 5.12: Throughput versus SNR performance for transmission over a quasi-static Rayleigh channel, with the retransmission limit set to 6 and the packet length set to 1000 bits. The schematic of 5.4, 5.5 and 5.6 for the MCTC HARQ scheme, the schematic of Figures 4.17 and 4.18 for Souza's scheme, as well as the parameters of Table 5.1 were used.

as in Figure 5.7. It can be observed from these two figures that our ES-based MCTC aided HARQ scheme using the $m = 1$ polynomials of $(2, 3)_o$ succeeds in maintaining a similar PLR and throughput performance as that offered by Souza's more complex scheme using the $m = 3$ polynomials of $(17, 15)_o$, regardless of whether the proposed ES approach is adopted or not. However, both the PLR and throughput are significantly decreased if the polynomials are changed to $(2, 3)_o$ for Souza's scheme, showing that this low-complexity polynomial pair is only appropriate for the proposed MCTC HARQ scheme.

Figure 5.13 shows the complexity benefits achieved by the proposed ES-based MCTC aided HARQ scheme, which has a significantly lower complexity than Souza's original systematic TCTC HARQ scheme relying on the $m = 3$ polynomials of $(17, 15)_o$. Specifically, for SNRs below -2dB, the complexity is reduced by 60% to 85%. At higher SNRs, the complexity is similar, because the CRC is typically satisfied for all schemes after a few BCJR operations. On the other hand, if Souza's systematic TCTC HARQ scheme using the $m = 3$ polynomials of $(17, 15)_o$ adopts our proposed ES approach, its complexity can be significantly reduced. However, its complexity still remains higher than that of our MCTC aided HARQ scheme, because it has to adopt those longer memory length polynomials. Although Souza's scheme using our proposed ES approach can achieve the lowest complexity for the $m = 1$ polynomials of $(2, 3)_o$, this is

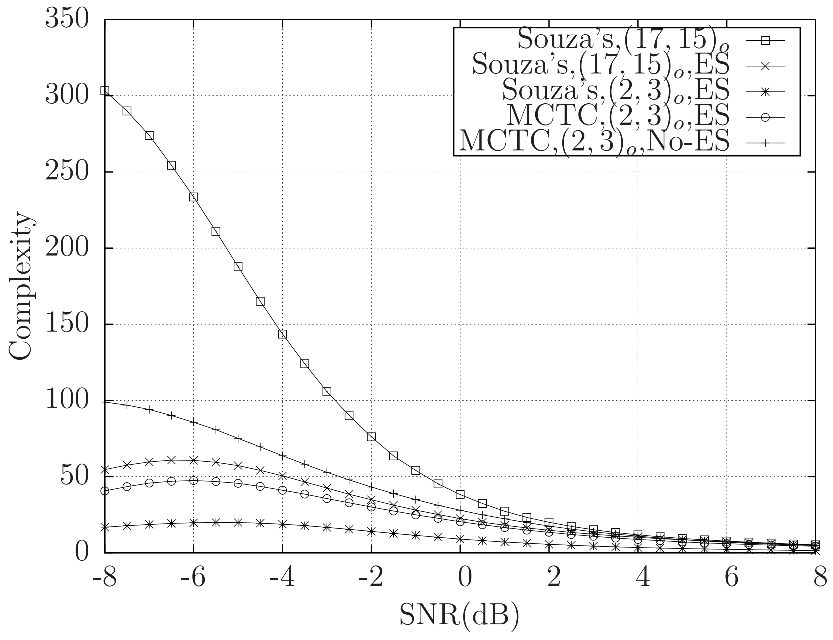


Figure 5.13: Complexity versus SNR performance for transmission over a quasi-static Rayleigh channel, with the retransmission limit equal to 6 and the packet length set to 1000 bits. The schematic of 5.4, 5.5 and 5.6 for the MCTC HARQ scheme, the schematic of Figures 4.17 and 4.18 for Souza's scheme, as well as the parameters of Table 5.1 were used.

achieved at the cost of sacrificing the PLR and throughput performance, as demonstrated in Figures 5.7 and 5.8.

The PLR, throughput and complexity curves were recorded for the MCTC HARQ scheme without ES in Figures 5.11, 5.12 and 5.13, which further demonstrate how critical the careful employment of ES is for a turbo HARQ scheme. The PLR and throughput performances recorded for the No-ES MCTC HARQ scheme exhibit an insignificant degradation. Furthermore, the complexity more than doubles compared to that of the ES aided MCTC HARQ scheme for low SNRs, namely below -2dB. This illustrates that a fixed number of 10 BCJR operations is insufficient for reaching the minimum PLR and maximum throughput that the MCTC HARQ scheme achieved for most situations, when the EXIT tunnel is open. By contrast, 10 BCJR operations appear to be excessive for most situations associated with closed EXIT tunnels, hence imposing an excessive complexity.

Finally, Table 5.4 compares the minimum number of variables that must be stored per bit of $\mathbf{a}$, for the MCTC HARQ scheme employing $r = 6$ BCJR decoders and for Souza's systematic TCTC HARQ scheme. Both schemes need to store one extrinsic LLR per BCJR decoder, namely 6 in the MCTC scheme and 2 in the TCTC scheme. Additionally, it is necessary to store one channel LLR per transmitted bit sequence, namely 6 in the MCTC scheme and 3 in the TCTC scheme. Furthermore, the authors of [148] demonstrated that at least 2^m alpha values per bit of $\mathbf{a}$ have to be stored between the forward and backward recursions of the log-BCJR algorithm, where alpha may be seen in Equation (5) of [45]. In summary, when assuming a retransmission limit of $r = 6$, the total memory requirements are 14 for the MCTC scheme and 13 for the TCTC schemes. Only one extra variable has to be required to be stored in memory per bit of $\mathbf{a}$ by the ES aided MCTC HARQ scheme, in order to achieve its remarkable performance improvement.

Table 5.4: Memory requirements comparison assuming that the retransmission limit is $r = 6$.

(per bit of $\mathbf{a}$)	MCTC HARQ	TCTC HARQ
extrinsic LLR	6	2
channel LLR	6	3
alpha value	2	8

5.1.6 Conclusions

Classic turbo HARQ schemes [125] [23] may have a considerable complexity, since a pre-defined fixed number of turbo decoding iterations are performed following each transmission, without considering the channel conditions. In this section, we have proposed an iterative decoding ES approach that maintains a high throughput and low PLR at a lower complexity. When comparing MCTC and systematic TCTC aided HARQ schemes, our simulation results of Figure 5.13 show that the proposed ES approach is capable of decreasing the complexity imposed by as much as 85%. In the next section, we will focus our attention on the prediction of the EXIT charts' required SNR for an open/closed tunnel under different channel conditions in order to delay the commencement of iterations and hence to further reduce the complexity imposed.

5.2 Look-up Table Based DI Aided Low Complexity Turbo HARQ

As introduced in Section 2.2, turbo codes [31, 32, 46] are characterized by an iterative exchange of increasingly reliable soft information between the constituent BCJR [45] decoders, which are concatenated in parallel and separated by an interleaver. Owing to their near-capacity performance, turbo codes can be successfully combined with HARQ schemes [124,125], in order to achieve a high throughput. In these turbo HARQ schemes, the transmitter continually transmits turbo-encoded IR to the receiver, where BCJR decoding operations may be performed iteratively following the reception of each transmission. Once the message is decoded successfully, the receiver returns an ACK to the transmitter, in order to cease its transmission of IR.

In general, there is a trade-off between the achievable throughput and the complexity imposed by turbo HARQ schemes. Naturally, the complexity is increased when more than necessary BCJR iterations are performed during the iterative decoding process following the reception of each IR transmission. For example, the early turbo HARQ schemes [23, 125] performed a sufficiently high number of BCJR decoder executions following each and every IR transmission in order to ensure that iterative decoding convergence had been achieved. In this way, they minimized the number of IR transmissions required, hence maximizing the throughput, at the cost of imposing an excessive complexity. On the other hand, the attainable throughput degrades, when sufficient IR contributions have been received for facilitating error-free decoding, but insufficient BCJR iterations have been performed. Our previous solution in Section 5.1 struck an attractive tradeoff by proposing an ES strategy for a turbo HARQ scheme. This ES strategy which has been published in [25] entirely curtails the iterative decoding process, when the rate of iterative MI improvement becomes lower than a pre-determined threshold. Since this approach eradicated unnecessary decoding iterations, the scheme of [25] exhibited a significantly lower complexity than those of [23, 125], without unduly compromising the attainable throughput.

Like the schemes of [23, 125], that of [25] activates iterative decoding following the reception of each transmission. However, a significant further decoding complexity reduction may be expected by complementing the ES strategy of [25] with the novel approach that we propose in this section, namely by Deferred Iterations (DI). This approach defers the commencement of iterative decoding, until sufficient IR contributions have been received to offer a sufficiently high likelihood of error-free decoding. This may be determined with the aid of the turbo decoder's EXIT chart [48] following the reception of each IR transmission. More specifically, for long packets, having an open EXIT tunnel implies that a sufficient amount of IR has been received to allow the iterative decoding trajectory to reach the $(1, 1)$ point of perfect convergence in the EXIT chart, where a vanishingly low BER is achieved. By contrast, having a closed tunnel implies that more IR is required before the decoding iterations should be commenced. The authors of [147] suggested to immediately cease the decoding iterations at SNRs below a certain threshold value given by that particular SNR, where decoding to convergence was 'just' possible for transmissions over an Additive White Gaussian Noise (AWGN) channel. Namely, the EXIT tunnel becomes marginally open at this threshold SNR, which is a unique SNR for transmissions over the AWGN channel. However, they did not consider the corresponding thresholds when transmitting over the quasi-static Rayleigh fading channels employed in [25], and did not exploit the supplemental information provided by IR transmissions in HARQ schemes. In this section, our approach determines the EXIT chart tunnel's state by using the MI of the IR received so far in order to consult a Look-Up Table (LUT).

The proposed DI technique is generically applicable, but in our design example we apply the above-mentioned LUT based DI strategy to turbo code aided HARQ schemes. Importantly, the proposed idea may also be extended to other diverse scenarios, which rely on iterative receivers. For instance in relay networks, a relay node may decide whether it should or should not continue the decoding and transmission of the source's message according to the LUT. Therefore, in Section 5.2.1.1 of this section, we first highlight the DI philosophy in the context of classic regular TCTCs, then extend it to MCTC aided HARQ schemes, which has been shown to have an attractive performance even when using the minimal possible constraint length of 2. Hence, Section 5.2.1 also introduces several methods of designing an efficient LUT for MCTC HARQ schemes. Then in Section 5.2.2, we apply the proposed LUT based DI strategy for further reducing the complexity of MCTC HARQ schemes. Section 5.2.3 compares the PLR, the throughput and the complexity of a suitably parameterized version of our scheme to those of appropriately chosen benchmarkers. Finally, Section 5.2.4 offers our conclusions for this section.

5.2.1 Look-up Table Design for MCTC HARQ Schemes

In this section, we firstly discuss the DI philosophy and demonstrate that the LUT size is potentially huge for MCTC HARQ schemes. In order to facilitate an efficient design, the following sub-sections aim for minimizing the LUT size and speed up the training. In Section 5.2.1.2, we conceive the proposed LUT structure specifically designed for the appropriately sorted MIs and invoke multiple-dimensional linear interpolation for generating its high granularity. Section 5.2.1.3 proposes a fast training procedure for generating the LUT. Section 5.2.1.4 analyzes the memory requirements of the resultant LUT designed for the MCTC HARQ scheme considered.

5.2.1.1 The Deferred Iteration Philosophy

Firstly, let us introduce our LUT concept for the rudimentary example of a classic TCTC using two parallel concatenated accumulators[2], while dispensing with HARQ assistance.

In a non-systematic TCTC encoder, one of the component encoders is employed to convert the information bit sequence $\mathbf{a}_1$ into the parity bit sequence $\mathbf{b}_1$, while the other is used for converting an interleaved version $\mathbf{a}_2$ of $\mathbf{a}_1$ into a second parity bit sequence $\mathbf{b}_2$. The transmitter then conveys these parity bit sequences to the receiver, where a soft demodulator [31] generates the corresponding LLR sequences $\tilde{\mathbf{b}}_1$ and $\tilde{\mathbf{b}}_2$. In this

[2]We define an accumulator to be a unity-rate recursive convolutional code having feedforward and feedback generator polynomials of $(2,3)_o$ respectively, which are expressed in an octal representation.

section, we incorporate diacritical tildes into our notation to denote the vectors of LLRs. These vectors of $\tilde{\mathbf{b}}_1$ and $\tilde{\mathbf{b}}_2$ are forwarded to two component decoders, which opt for iteratively exchanging the extrinsic LLR sequences $\tilde{\mathbf{a}}_1^{\mathrm{e}}$ and $\tilde{\mathbf{a}}_2^{\mathrm{e}}$. Following interleaving, these sequences become the *a priori* LLR sequences $\tilde{\mathbf{a}}_2^{\mathrm{a}}$ and $\tilde{\mathbf{a}}_1^{\mathrm{a}}$ [31]. This process may be characterized by a two-dimensional EXIT chart [48], comprising the EXIT functions $I(\tilde{\mathbf{a}}_1^{\mathrm{e}}) = f_1[I(\tilde{\mathbf{a}}_1^{\mathrm{a}}), I(\tilde{\mathbf{b}}_1)]$ and $I(\tilde{\mathbf{a}}_2^{\mathrm{e}}) = f_2[I(\tilde{\mathbf{a}}_2^{\mathrm{a}}), I(\tilde{\mathbf{b}}_2)]$. Here, the MI $I(\tilde{\mathbf{x}})$ quantifies the reliability of the information in the sequence of L LLRs $\tilde{\mathbf{x}} = \{\tilde{x}_j\}_{j=1}^{L}$ pertaining to the particular values of the L bits in the corresponding sequence $\mathbf{x} = \{x_j\}_{j=1}^{L}$, according to Equation 2.17.[3] In a quasi-static Rayleigh fading channel, consecutive IR transmissions may have MIs that are distributed over the entire legitimate MI range of $[0, 1]$. In general, the MI values may be quantized discretely with a granularity or step-size of $G = 0.01$ without a sacrifice in precision.

However, the receiver may refrain from activating the iterative decoding process, if it deems the MI of the LLR sequences to be insufficient for creating an open tunnel in the EXIT chart [31] [48], indicating that convergence towards the infinitesimally low BER is prevented. We assume that the receiver uses Equation 2.17 to determine the MI between the estimated hard-decision-based bits and the channel's output LLRs $\tilde{\mathbf{b}}_1$, which is for example $I(\tilde{\mathbf{b}}_1) = 0.15$. In the case where unity-rate accumulators are employed as component codes, this MI corresponds to the EXIT function $I(\tilde{\mathbf{a}}_1^{\mathrm{e}}) = f_1[I(\tilde{\mathbf{a}}_1^{\mathrm{a}}), 0.15]$ of Figure 5.14.

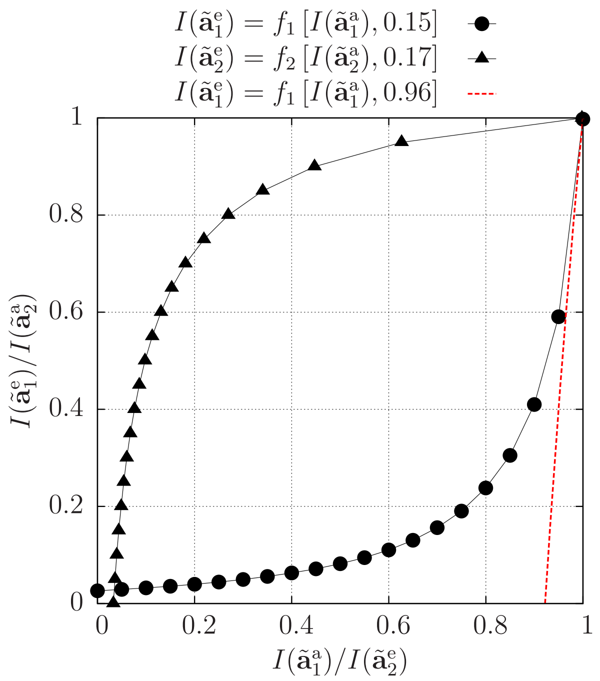


Figure 5.14: The EXIT chart of a TCTC using two parallel concatenated accumulators.

[3]For notational simplicity, we use $I(\tilde{\mathbf{x}})$ to replace $I(\tilde{\mathbf{x}}, \mathbf{x})$, which more explicitly denotes the MI between the LLRs $\tilde{\mathbf{x}}$ and the corresponding bit sequence $\mathbf{x}$.

This EXIT chart analysis reveals that if the channel's output LLRs $\tilde{\mathbf{b}}_2$ have an MI of at least $I(\tilde{\mathbf{b}}_2) = 0.96$, then an open EXIT chart tunnel will be created between $I(\tilde{\mathbf{a}}_1^{\mathrm{e}})$ and the EXIT function $I(\tilde{\mathbf{a}}_2^{\mathrm{e}}) = f_2[I(\tilde{\mathbf{a}}_2^{\mathrm{a}}), 0.96]$. This threshold MI value of $I(\tilde{\mathbf{b}}_2) = 0.96$ could be revealed to the receiver by rounding the value of $I(\tilde{\mathbf{b}}_1) = 0.15$ to two decimal places and using it to index an LUT comprising $(\frac{1}{G}+1) = 101$ entries, spanning the MI range of $I(\tilde{\mathbf{b}}_1) \in [0, 1]$. However, Equation 2.17 may actually suggest that the LLRs of $\tilde{\mathbf{b}}_2$ have an MI as low as $I(\tilde{\mathbf{b}}_2) = 0.17$, which is below the MI threshold of 0.96 and corresponds to the EXIT function of $I(\tilde{\mathbf{a}}_2^{\mathrm{e}}) = f_2[I(\tilde{\mathbf{a}}_2^{\mathrm{a}}), 0.17]$ in Figure 5.14. Using the LUT, the receiver becomes aware that the EXIT chart tunnel is closed and can therefore defer attempting the recovery of the information bit sequence $\mathbf{a}_1$, since its decoding attempt is very unlikely to be successful and yet, it would impose a certain complexity and power drain.

Let us now consider the application of our LUT technique to an MCTC HARQ scheme. A non-systematic MCTC encoder [24] comprises r number of component encoders, each of which $i \in \{1, 2, \ldots, r\}$ generates a parity bit sequence $\mathbf{b}_i$ by encoding a differently interleaved version $\mathbf{a}_i$ of the information bit sequence $\mathbf{a}_1$ (or use $\mathbf{a}_1$ itself for $i = 1$). In HARQ applications, the number of component encoders r is successively incremented and the corresponding parity bit sequences are successively transmitted until the number of transmissions r reaches a specified maximum IR limit R, or until the receiver acknowledges that the information bit sequence $\mathbf{a}_1$ has been successfully recovered. As each transmission is received at the receiver, the corresponding LLR sequence $\tilde{\mathbf{b}}_r$ is forwarded to a component decoder that is concatenated to those corresponding to the previous $(r-1)$ transmissions. At this point, the receiver may opt for iteratively exchanging the extrinsic LLR sequences $\tilde{\mathbf{a}}_i^{\mathrm{e}}$ among the r component decoders, each of which interleaves and sums the extrinsic LLRs provided by the other decoders, in order to generate the *a priori* LLRs $\tilde{\mathbf{a}}_i^{\mathrm{a}}$. This process may be characterized by an r-dimensional EXIT chart, comprising the r EXIT functions $I(\tilde{\mathbf{a}}_i^{\mathrm{e}}) = f_i[I(\tilde{\mathbf{a}}_i^{\mathrm{a}}), I(\tilde{\mathbf{b}}_i)]$. Alternatively, this process may be characterized by a two-dimensional EXIT chart, comprising the EXIT function $I(\tilde{\mathbf{a}}_r^{\mathrm{e}}) = f_r[I(\tilde{\mathbf{a}}_r^{\mathrm{a}}), I(\tilde{\mathbf{b}}_r)]$ and the composite EXIT function $I(\tilde{\mathbf{a}}_r^{\mathrm{a}}) = f_{\{1,2,\ldots,r-1\}}[I(\tilde{\mathbf{a}}_r^{\mathrm{e}}), I(\tilde{\mathbf{b}}_1), I(\tilde{\mathbf{b}}_2), \ldots, I(\tilde{\mathbf{b}}_{r-1})]$ [24]. This composite EXIT function characterizes the provision of the r^{th} decoder's extrinsic LLRs $\tilde{\mathbf{a}}_r^{\mathrm{e}}$ for the iterative operation of the other $(r-1)$ decoders until convergence is achieved, whereupon the sum of their extrinsic LLRs are provided for the r^{th} decoder for employment as $\tilde{\mathbf{a}}_r^{\mathrm{a}}$, namely as the *a priori* information.

Extending the above example to our MCTC HARQ design example, when the EXIT chart is deemed to be closed for the pair of received MIs contributions $I(\tilde{\mathbf{b}}_1) = 0.15$ and $I(\tilde{\mathbf{b}}_2) = 0.17$, the receiver would opt for deferring the start of the iterative decoding process by returning no acknowledgement to the transmitter and waiting for it to supply more IR. When unity-rate accumulators are employed as the component codes, the MIs $I(\tilde{\mathbf{b}}_1) = 0.15$ and $I(\tilde{\mathbf{b}}_2) = 0.17$ correspond to the composite EXIT function $I(\tilde{\mathbf{a}}_3^{\mathrm{a}}) = f_{\{1,2\}}[I(\tilde{\mathbf{a}}_3^{\mathrm{e}}), 0.15, 0.17]$ of Figure 5.15.

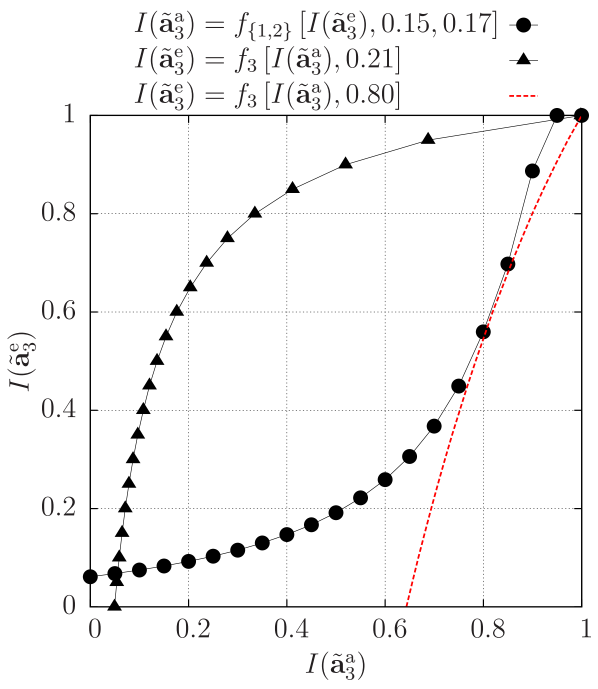


Figure 5.15: The EXIT chart of a 3-component MCTC.

This EXIT chart analysis reveals that if the channel's output LLRs $\tilde{\mathbf{b}}_3$ supplied by the $r = 3^{\mathrm{rd}}$ transmission have an MI of at least $I(\tilde{\mathbf{b}}_3) = 0.80$, then an open EXIT chart tunnel will be created with the EXIT function $I(\tilde{\mathbf{a}}_3^{\mathrm{e}}) = f_3[I(\tilde{\mathbf{a}}_3^{\mathrm{a}}), 0.80]$. In this case, there is a high probability that the information bit sequence $\mathbf{a}_1$ can indeed be successfully recovered by activating the iterative decoding process. If this is not the case, e.g. we have $I(\tilde{\mathbf{b}}_3) = 0.21$ as seen in Figure 5.15, then the start of the iterative decoding process can be further deferred, until an open EXIT chart tunnel is deemed to have been created. Here, the threshold value of $I(\tilde{\mathbf{b}}_3) = 0.80$ could be revealed to the receiver by rounding the values of $I(\tilde{\mathbf{b}}_1) = 0.15$ and $I(\tilde{\mathbf{b}}_2) = 0.17$ to two decimal places and using them as the address of an LUT comprising $(101)^2$ entries. This example demonstrates that a naive LUT implementation for an r-component MCTC decoder requires $(101)^{r-1}$ entries and that since r is successively incremented in MCTC HARQ schemes, a total of $\sum_{r=2}^{R}(101)^{r-1}$ entries is required, when a transmission limit of R is imposed. However, in Section 5.2.1.2 and 5.2.1.3, we will propose a sophisticated LUT design, which significantly reduces the number of LUT entries that must be trained and stored. In this way, a practical DI scheme is devised for MCTC HARQ schemes, facilitating a significantly reduced iterative decoding complexity.

5.2.1.2 Minimizing the Storage Requirements of the MCTC HARQ LUT

The LUT designed for MCTC HARQ schemes may be defined as a set of sub-tables T_i generated for describing the relationship between any particular set of $(i-1)$ MIs and the minimum supplemental MI $I_{th}(i)$ required for creating an open EXIT chart tunnel, according to:

$$I_{th}(i) = T_i \left[I(\tilde{\mathbf{b}}_1), I(\tilde{\mathbf{b}}_2), \cdots, I(\tilde{\mathbf{b}}_{i-1}) \right] \ , \tag{5.1}$$

where $2 \leq i \leq (R-1)$. Our DI strategy aided MCTC HARQ schemes does not require the R^{th} sub-table T_R, since the receiver should always exploit its final - namely the R^{th} opportunity of activating the turbo decoder after the reception of the R^{th} IR transmission. The justification of this action is that the correct reception of the information is more valuable than the price of the extra complexity required for performing this last-ditch decoding effort.

The LUT of MCTC HARQ only records the required MIs for a limited set of quantized and sorted $(i-1)$ MI values appearing in an ascending order, i.e. satisfying $I(\tilde{\mathbf{b}}_{\pi(1)}) \leq I(\tilde{\mathbf{b}}_{\pi(2)}) \leq ... \leq I(\tilde{\mathbf{b}}_{\pi(i-1)}) \leq I_{th}(i)$, where π contains the unique integers of $1, ..., (i-1)$ used for appropriately permuting the original IR transmission order. This method avoids storing a large amount of redundant entries, since for example $I_{th}(4) = T_4(0.21, 0.15, 0.17)$ is identical to $I_{th}(4) = T_4(0.15, 0.17, 0.21)$.

Figure 5.16 displays a subset of the LUTs recorded for BPSK transmission over a quasi-static Rayleigh fading channel, as considered in Section 5.2.1.3. As seen in Figure 5.16, the LUT is composed of several sub-tables, each of which corresponds to different IR indices i. Each sub-table T_i - except for the last one having the index $(R-1)$ - contains two columns. The first one stores the threshold MI $I_{th}(i)$, while the other provides an offset which can assist the indexing of the next sub-table T_{i+1}. More explicitly, the first element of each row in the sub-table T_i stores the $I_{th}(i)$ value that is valid for a particular set of MIs $\{I(\tilde{\mathbf{b}}_{\pi(1)}), I(\tilde{\mathbf{b}}_{\pi(2)}), ..., I(\tilde{\mathbf{b}}_{\pi(i-1)})\}$. The second element of each row stores a specific offset of the sub-table T_{i+1}, which indicates the starting index of the rows related to the sub-table T_i within T_{i+1}. Specifically, these rows are displayed in dashed boxes in Figure 5.16, all of which store the threshold MIs $I_{th}(i+1)$ corresponding to the sets of $\{I(\tilde{\mathbf{b}}_{\pi(1)}), I(\tilde{\mathbf{b}}_{\pi(2)}), ..., I(\tilde{\mathbf{b}}_{\pi(i-1)}), I(\tilde{\mathbf{b}}_{\pi(i)})\}$, where $I(\tilde{\mathbf{b}}_{\pi(i)})$ is varied but the previous $(i-1)$ IR MI values remain the same as in conjunction with the equivalent row in T_2.

Considering $I(\tilde{\mathbf{b}}_{\pi(1)}) = 0.15$ in Figure 5.16 as an example for providing further explanations, its corresponding threshold information is $I_{th}(2) = T_2(0.15) = 0.96$, implying that the minimum IR MI required for creating an open tunnel is 0.96, when

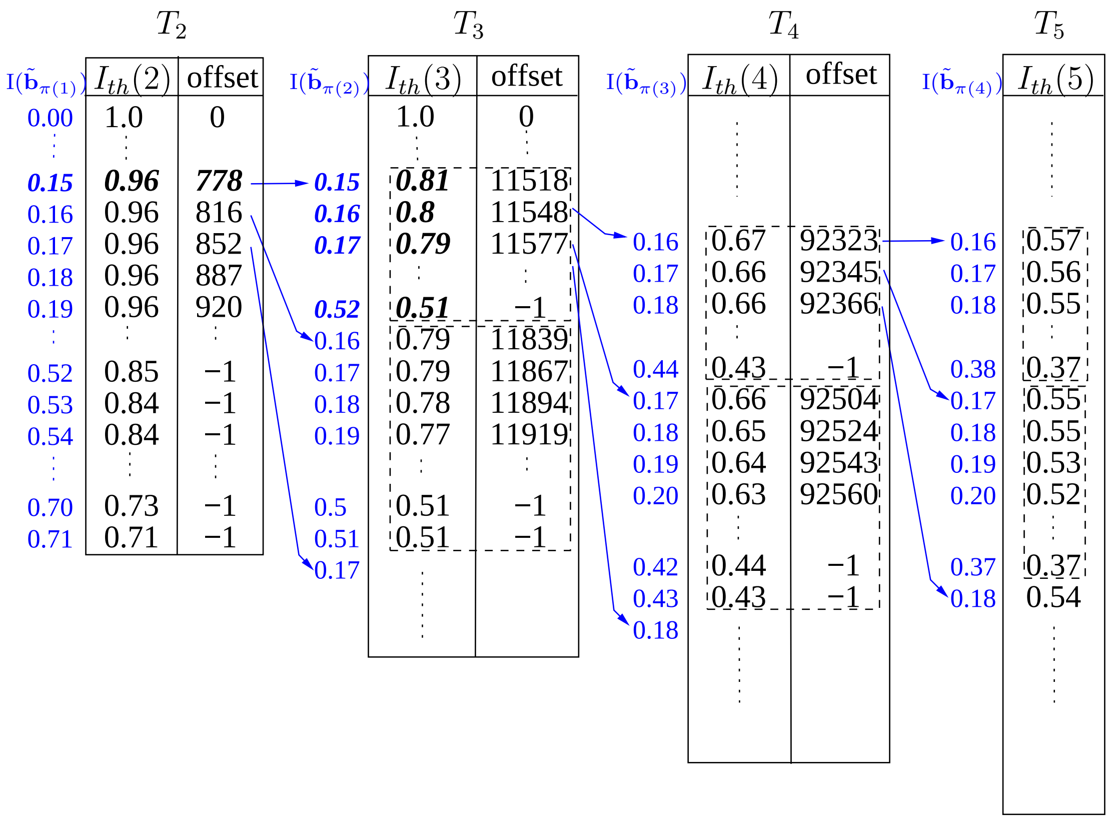


Figure 5.16: A subset of the MCTC HARQ LUT for $i = 2, 3, 4$ and 5, where the input MI step-size is $G = 0.01$.

the received MI of the first transmission is 0.15. This threshold is stored at the row index of $I(\tilde{\mathbf{b}}_{\pi(1)}) \cdot \frac{1}{G} = 15$ in sub-table T_2 having $G = 0.01$, as seen in Figure 5.16. Due to the sorting of $I(\tilde{\mathbf{b}}_{\pi(1)}) \leq I(\tilde{\mathbf{b}}_{\pi(2)}) \leq I_{th}(3)$, the initial value of $I(\tilde{\mathbf{b}}_{\pi(2)})$ is 0.15, thence we fix $I(\tilde{\mathbf{b}}_{\pi(1)}) = 0.15$ and increase $I(\tilde{\mathbf{b}}_{\pi(2)})$ from 0.15 by $G = 0.01$ each step (printed in bold fonts) to explore all possible $I_{th}(3)$ values corresponding to them. The operations continue until the $I_{th}(3)$ value required for perfect convergence becomes less than the current $I(\tilde{\mathbf{b}}_{\pi(2)})$ value, again, owing to the above-mentioned ordering. The resultant $I_{th}(3)$ values corresponding to $I(\tilde{\mathbf{b}}_{\pi(1)}) = 0.15$ and the incremental $I(\tilde{\mathbf{b}}_{\pi(2)})$ values are recorded in a block of continuous rows, as illustrated in the first dashed rectangle of sub-table T_3 in Figure 5.16. These 38 rows of sub-table T_3 have the offsets ranging from 778 to 815, which correspond to the 38 incremental $I(\tilde{\mathbf{b}}_{\pi(2)})$ values ranging from 0.15 to 0.52, as seen in the index area of sub-table T_3 in Figure 5.16. Here, the index of this block in sub-table T_3 starts at 778, since the rows 0 to 14 in sub-table T_2 have a total of 778 entries in sub-table T_3.

There are some special cases to be considered in Figure 5.16, for example, when the fixed $I(\tilde{\mathbf{b}}_{\pi(1)})$ has a larger value, such as $I(\tilde{\mathbf{b}}_{\pi(1)}) = 0.52$. Then, owing to the above-mentioned ordering, the incremental $I(\tilde{\mathbf{b}}_{\pi(2)})$ values start from 0.52. In this situation, the threshold IR MI $I_{th}(3) = 0.15$ corresponding to $\{0.52, 0.52\}$ becomes less than the current $I(\tilde{\mathbf{b}}_{\pi(2)}) = 0.52$. This suggests that no entries will be stored in sub-table T_3 for the fixed $I(\tilde{\mathbf{b}}_{\pi(1)})$ value of 0.52. We tag the corresponding offset as -1 for these cases.

Based on the above structure of the LUT, the search operation may be carried out at a low complexity using the offsets that are stored in the sub-tables. More specifically, the expression of 'offset+$[I(\tilde{\mathbf{b}}_{\pi(i)}) - I(\tilde{\mathbf{b}}_{\pi(i-1)})] \cdot 100$' may be applied recursively, where an initial 'offset' of 0 and $I(\tilde{\mathbf{b}}_0) = 0$ are assumed for $i = 1$, namely searching a certain $I_{th}(2)$ from the sub-table T_2. Once an offset of -1 has been found, this indicates that the EXIT tunnel is definitely open, since the received MIs are always sorted in an ascending order before the search is activated.

Although the LUT only records the sorted MIs, the size of the sub-table T_i increases approximately by an order of magnitude upon increasing i by one. However, increasing the step-size G has the potential of further decreasing the LUT size. In order to guarantee a high accuracy for the determination of the EXIT tunnel's open/closed state, the values of $I_{th}(i)$ in the LUT always maintain a step-size of 0.01, while $I(\tilde{\mathbf{b}}_1), I(\tilde{\mathbf{b}}_2), \cdots, I(\tilde{\mathbf{b}}_{i-1})$ tend to be increased in larger steps. As a result, the sub-table T_4 of Figure 5.16 changes to only store the threshold MIs for those larger stepped $I(\tilde{\mathbf{b}}_1), I(\tilde{\mathbf{b}}_2), \cdots, I(\tilde{\mathbf{b}}_{i-1})$. For example, for a step-size of $G = 0.1$, it stores the threshold MIs such as $0.80 = T_4(0.1, 0.1, 0.1)$, $0.73 = T_4(0.1, 0.1, 0.2)$, $0.65 = T_4(0.1, 0.2, 0.2)$ and so on. For a set of $(i-1)$ received MIs represented at a higher resolution, we follow the classic multi-dimensional linear interpolation technique for the sake of estimating the supplemental MI required for achieving perfect decoding convergence and hence for triggering iterative decoding. More specifically, the binary tree seen in Figure 5.17 was designed for mapping and interpolation, which exemplifies how to estimate the threshold MI $I_{th}(4)$ for the received MI values of $0.15, 0.17, 0.21$ in the preceding example.

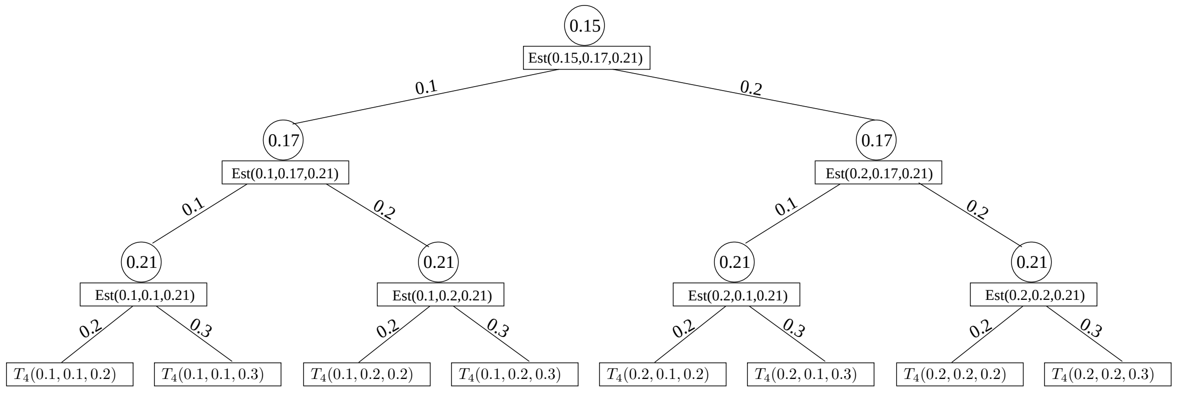


Figure 5.17: A binary tree exemplifying the mapping operations from top to bottom and the interpolation operations from bottom to top for multiple-dimensional linear interpolation, which is employed for enhancing the accuracy of the LUT originally generated for a larger step-size.

This binary tree exemplifies the mapping operations from top to bottom and the interpolation operations from bottom to top. During the first step of proceeding down the tree, the received MI values of $0.15, 0.17, 0.21$ are seen in the circles of the tree's layers, each layer corresponding to one received MI value, as seen in Figure 5.17. Then, these received MI values may be mapped downwards to the closest granularity value along the left branch, while upwards to the closest one along the right branch. For example, in the second layer of Figure 5.17, the received MI of 0.17 is mapped to the lower value of 0.1 and to the higher value of 0.2. These mapping operations continue, until reaching the leaf nodes of the tree. Then, the eight threshold MIs denoted by $T_4(0.1, 0.1, 0.2)$, $T_4(0.1, 0.1, 0.3)$ and so on as seen at the bottom of Figure 5.17 are checked out from the LUT according to the mapped values. More explicitly, these thresholds stored in the LUT corresponding to $(0.1, 0.1, 0.2)$ or $(0.1, 0.1, 0.3)$ may be found, using the method introduced earlier in this section. For example, the stored threshold MI for $(0.1, 0.1, 0.2)$ is 0.73, and for $(0.1, 0.1, 0.3)$ is 0.66. These checked out thresholds are utilized for calculating the target threshold MI during the interpolation operations from bottom to top, since each received MI is embraced by two adjacent mapped values. For example, as seen in Figure 5.17, the estimated threshold MI $Est(0.1, 0.1, 0.21)$ may be obtained by a linear interpolation between the checked out values of $0.73 = T_4(0.1, 0.1, 0.2)$ and $0.66 = T_4(0.1, 0.1, 0.3)$, which is 0.723. Proceeding further up the tree, $Est(0.1, 0.17, 0.21)$ may be interpolated between the estimated values of $Est(0.1, 0.1, 0.21)$ and $Est(0.1, 0.2, 0.21)$ in the third layer. When the interpolation operations are continued further up, all the way to the root node, the target threshold MI for $0.15, 0.17, 0.21$ may be finally calculated.

Based on the binary tree seen in Figure 5.17, the operations involved for multi-dimensional linear interpolation will require 2^{i-1} memory accesses to the LUT and $\sum_{k=0}^{i-2} 2^k$ linear interpolations for generating $I_{th}(i)$. Nonetheless, the complexity imposed is still far lower than the BCJR decoding complexity, when the value of i is moderate.

5.2.1.3 Minimizing the Training Complexity of the LUT

An offline training process was developed for generating the LUT of Figure 5.16 by finding the specific $I_{th}(i)$ values for all possible values of $I(\tilde{\mathbf{b}}_{\pi(1)}), I(\tilde{\mathbf{b}}_{\pi(2)}) \ldots I(\tilde{\mathbf{b}}_{\pi(i-1)})$. This process scans through the entire input MI range in steps of 0.01 - regardless of the specific value of G - in order to find the minimum MI for creating a marginally open EXIT tunnel, when it is combined with a certain set of MIs $I(\tilde{\mathbf{b}}_{\pi(1)}), I(\tilde{\mathbf{b}}_{\pi(2)}) \ldots I(\tilde{\mathbf{b}}_{\pi(i-1)})$. For the sake of efficient execution, we abandon the traditional LLR-histogram based experimental EXIT chart creation and instead we use a model based on the spline functions $I(\tilde{\mathbf{a}}_i^{\mathrm{e}}) = f_i'[I(\tilde{\mathbf{a}}_i^{\mathrm{a}}), I(\tilde{\mathbf{b}}_i)]$ by fitting the corresponding EXIT functions f_i, as introduced in Chapter 4. Each spline function f_i' consists of a group of linear polynomials having the following form

$$I(\tilde{\mathbf{a}}_i^{\mathrm{e}}) = \begin{cases} m_{i1} \cdot I(\tilde{\mathbf{a}}_i^{\mathrm{a}}) + n_{i1} & (0.0 \leq I(\tilde{\mathbf{a}}_i^{\mathrm{a}}) < 0.1) \\ m_{i2} \cdot I(\tilde{\mathbf{a}}_i^{\mathrm{a}}) + n_{i2} & (0.1 \leq I(\tilde{\mathbf{a}}_i^{\mathrm{a}}) < 0.2) \\ & \vdots \\ m_{i10} \cdot I(\tilde{\mathbf{a}}_i^{\mathrm{a}}) + n_{i10} & (0.9 \leq I(\tilde{\mathbf{a}}_i^{\mathrm{a}}) \leq 1.0) \ , \end{cases} \tag{5.2}$$

where $m_{i1} \cdots m_{i10}$ and $n_{i1} \cdots n_{i10}$ were determined in advance for any $I(\tilde{\mathbf{b}}_i) \in [0, G, 2G, \cdots, 1]$.

The training is accelerated by iteratively invoking r spline functions f_i' corresponding to the MCTC decoder's r constituent components, since each call of this function only involves simple calculations. More specifically, all $I(\tilde{\mathbf{a}}_i^{\mathrm{e}}), i = 1, 2, \cdots, r$ are firstly initialized to 0. Then, we progress from f_1' to f_r', each time with a corresponding $I(\tilde{\mathbf{a}}_i^{\mathrm{a}})$ input calculated according to the following equation:

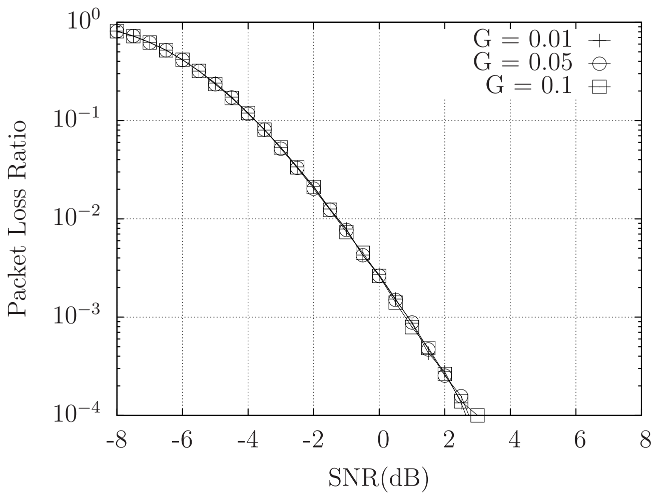
$$I(\tilde{\mathbf{a}}_i^{\mathrm{a}}) = J\left(\sqrt{\sum_{j=1, j\neq i}^{r} \left[J^{-1}(I(\tilde{\mathbf{a}}_j^{\mathrm{e}}))\right]^2}\right) \ , \tag{5.3}$$

where the function $J(\cdot)$ and $J^{-1}(\cdot)$ can be found in the Appendix of [122]. This procedure continues until the output extrinsic information $I(\tilde{\mathbf{a}}_i^{\mathrm{e}})$ approaches 1 or until the affordable number of iterations is exhausted.

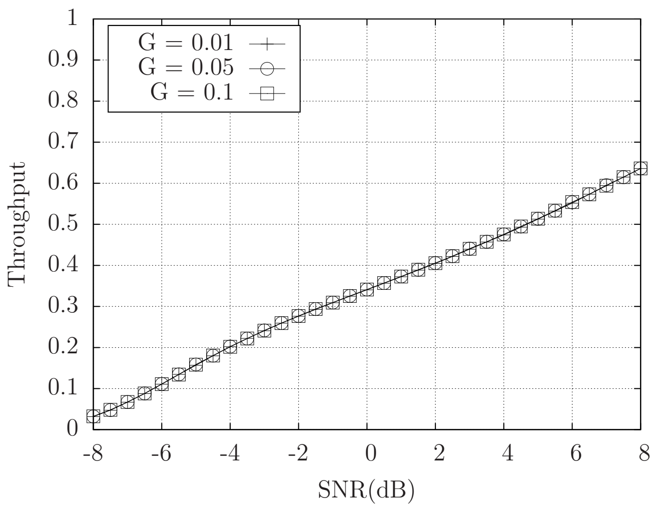
The training process can be accelerated by exploiting the monotonically decreasing nature of the function T_i, which ensures that each consecutive row in the LUT of Figure 5.16 requires a slightly lower MI value for achieving convergence than the previous one. For example, where we have $I(\tilde{\mathbf{b}}_{\pi(1)}), I(\tilde{\mathbf{b}}_{\pi(2)}), I(\tilde{\mathbf{b}}_{\pi(3)}) = \{0.15, 0.15, 0.17\}$, the additional MI required is $I_{th}(4) = 0.66$ in Figure 5.16, which is slightly lower than the previous value of $I_{th}(4) = 0.67$, corresponding to $I(\tilde{\mathbf{b}}_{\pi(1)}), I(\tilde{\mathbf{b}}_{\pi(2)}), I(\tilde{\mathbf{b}}_{\pi(3)}) = \{0.15, 0.15, 0.16\}$. This implies that the investigation of each new threshold test may commence from the value of the previous one, in decremental steps of 0.01. Statistically speaking, only two or three steps are required for determining the additional MI that yields an open EXIT tunnel.

5.2.1.4 Storage Requirements of the LUT

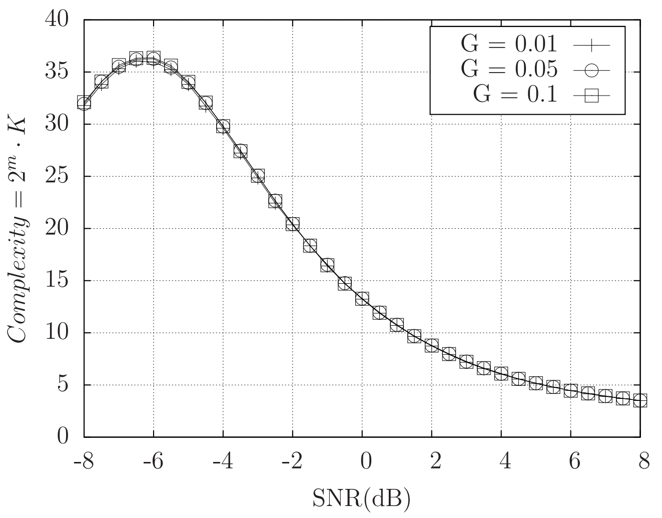
Based on the above LUT structure and training process, Table 5.5 shows the number of entries for T_2 to T_7 sub-tables of the LUT for different step-sizes, namely $G \in \{0.01, 0.05, 0.1\}$.



(a) PLR



(b) Throughput



(c) Complexity

Figure 5.18: PLR, throughput and complexity versus SNR for transmissions over a quasi-static Rayleigh channel, with the retransmission limit equal to 6, the packet length set to 480 bits and the LUTs of different step-sizes. The schematic of 5.4, 5.5 and 5.19, as well as the parameters of Table 5.6 were used.

Table 5.5: The number of entries for 6 sub-tables of the LUT for the step-sizes of 0.01, 0.05 and 0.1.

	T_2	T_3	T_4	T_5	T_6	T_7
0.01	71	1261	13337	95388	508767	2159981
0.05	16	80	264	602	1108	1752
0.1	9	32	70	110	137	164

Observe from Table 5.5 that the total number of entries is significantly reduced for the step-size of $G = 0.1$ in comparison to 0.01. Adopting the parameters of Section 5.2.3 again and transmitting a sufficiently high number of packets having a length of 480 bits over a quasi-static Rayleigh channel, our simulation results seen in Figure 5.18 demonstrated that the PLR, throughput and complexity curves of $G = 0.01, 0.05$ and 0.1 recorded for the LUT based DI aided MCTC HARQ are almost identical. Therefore, $G = 0.1$ is the best choice for the LUT, since it requires the minimum storage. When designing T_2, T_3, T_4 and T_5 with the step-size of 0.1 for our later simulations supporting $R = 6$ IR transmissions, only 212 MI thresholds have to be stored in these four sub-tables of the LUT. The complexity of the associated multiple-dimensional interpolation includes at most 16 memory accesses and 15 simple linear interpolations. Furthermore, if the LUT has to support more than $R = 6$ IR transmissions, e.g. $R = 8$, we may combine T_2, T_3 having $G = 0.01$ with T_4 to T_7 also having $G = 0.1$ in order to strike a trade-off between the memory requirements and the multi-dimensional interpolation cost.

Table 5.6: The simulation parameters for comparing the PLR, throughput and complexity performance with the DI strategy for MCTC aided HARQ schemes, when the LUTs are generated with different step-sizes.

Retry Limit	6
Packet Length	480-bit
Stopping Strategy	ES
T_{inc}	0.011
T_{max}	0.999
Deferred Iteration	with
LUT step-size	0.01, 0.05, 0.1
I_{diff}	0.01
Modulation Scheme	BPSK
Channel Type	quasi-static Rayleigh fading

5.2.2 The LUT Based DI Aided MCTC HARQ Scheme

Based on our efficient LUT, Figure 5.19 illustrates the flow-chart of the receiver when the DI strategy is applied to our MCTC HARQ design example. In detail, if the

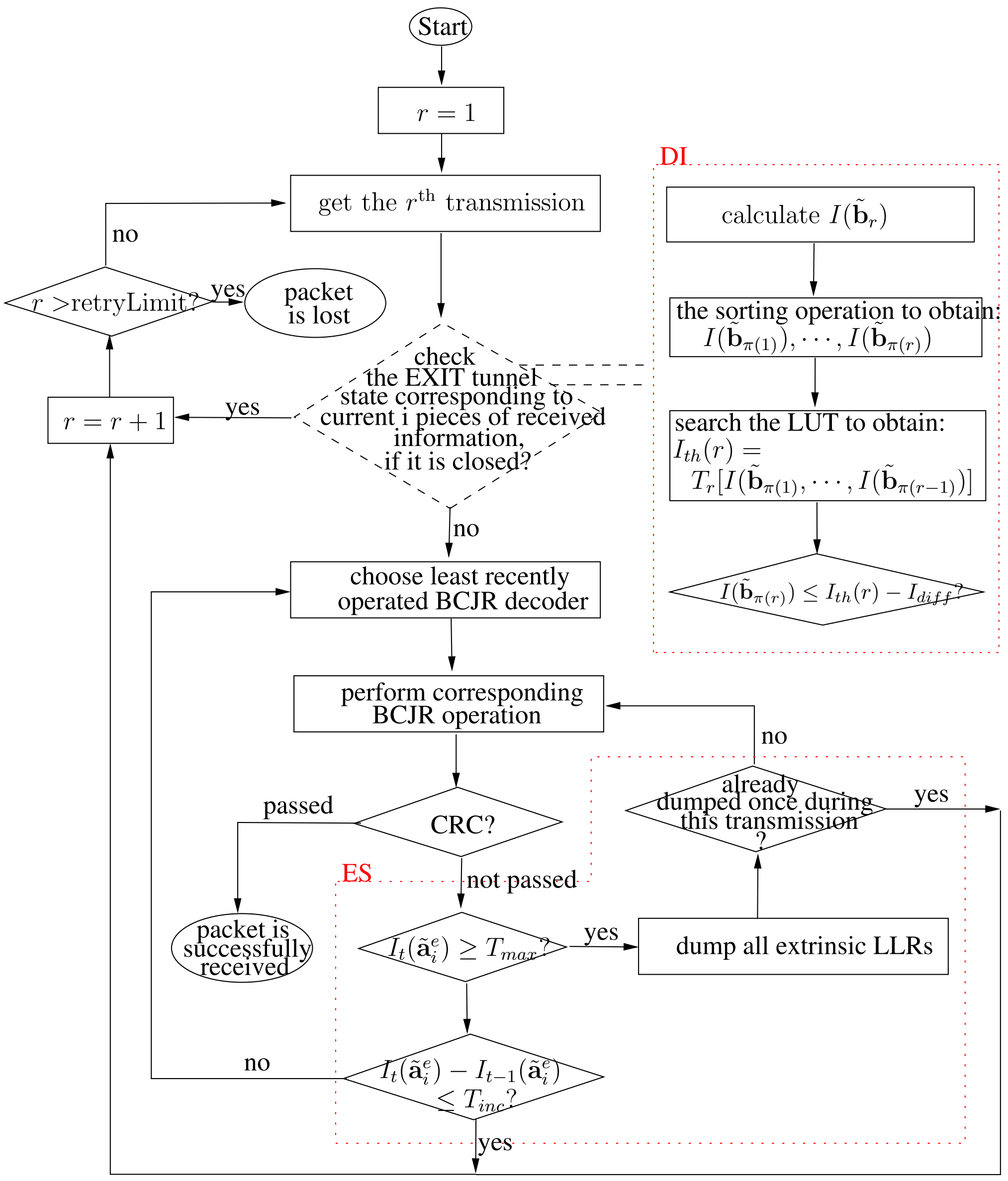

Start
$r = 1$
get the r^{th} transmission
no
r >retryLimit?
yes
packet is lost
check the EXIT tunnel state corresponding to current i pieces of received information, if it is closed?
yes
$r = r + 1$
no
DI
calculate $I(\tilde{\mathbf{b}}_r)$
the sorting operation to obtain: $I(\tilde{\mathbf{b}}_{\pi(1)}), \cdots, I(\tilde{\mathbf{b}}_{\pi(r)})$
search the LUT to obtain: $I_{th}(r) = T_r[I(\tilde{\mathbf{b}}_{\pi(1)}, \cdots, I(\tilde{\mathbf{b}}_{\pi(r-1)})]$
$I(\tilde{\mathbf{b}}_{\pi(r)}) \leq I_{th}(r) - I_{diff}$?
choose least recently operated BCJR decoder
perform corresponding BCJR operation
no
already dumped once during this transmission ?
yes
passed
CRC?
not passed
ES
packet is successfully received
$I_t(\tilde{\mathbf{a}}_i^e) \geq T_{max}$?
yes
dump all extrinsic LLRs
no
$I_t(\tilde{\mathbf{a}}_i^e) - I_{t-1}(\tilde{\mathbf{a}}_i^e) \leq T_{inc}$?
yes

Figure 5.19: The flow chart of the DI and ES aided MCTC HARQ scheme based on the look-up table and the regime of Figure 5.1.

EXIT tunnel is deemed to be closed and the retransmission retry limit has not been exhausted, the receiver waits for the next IR transmission. By contrast, if the tunnel is sufficiently close to becoming open, the r-component turbo decoder is activated and iterative decoding proceeds as in the original ES aided MCTC HARQ scheme[4], commencing from the box labeled as 'choose the least recently operated BCJR decoder' which is copied from the flow-chart of the decoding process seen in Figure 3 of [25]. Since no BCJR decoding has been performed at this point, the first BCJR decoder

[4]In our LUT based DI aided HARQ scheme, the convergence of the MCTC decoder is defined as reaching the state, when the MI increment of *every* component BCJR decoder becomes lower than a particular threshold. This is slightly different from the approach of [25], which declares that convergence was reached once *any one* component decoder satisfies this stopping condition. This modification continues the decoding process for longer and is justified, because the proposed scheme does not commence decoding until there is a good chance that it will become successful.

is activated, followed by gradually proceeding one-by-one to the last one. Therefore, 'choosing the least recently operated BCJR decoder' implies that r BCJR decoders will be iteratively activated commencing from the first to the last. The iterative decoding process continues, until the CRC or the ES criterion is satisfied. The ES strategy employed was detailed in [25], which implies that the iterative decoding stops if any of the BCJR decoder's MI increment becomes less than a pre-defined minimum threshold. In the DI aided MCTC HARQ, since iterative decoding is only activated after the EXIT tunnel is deemed to be open, the iterative decoding process of this MCTC decoder is typically capable of achieving error-free decoding of the current packet. In the rare cases when the EXIT tunnel is only marginally open, the MCTC decoder may converge to a legitimate but incorrect decoding decision, which is spotted by the CRC assumed to be perfectly reliable, hence triggering an IR transmission. The packet will be deemed lost or dropped, when the number of transmissions reaches a retry limit.

The DI process is detailed in the upper dashed rectangle of Figure 5.19. As seen in Figure 5.19, the MI $I(\tilde{\mathbf{b}}_r)$ is firstly calculated using Equation 2.17. Next, the MI-sorting operation is performed in order to obtain the sorted MIs $I(\tilde{\mathbf{b}}_{\pi(1)})$, $I(\tilde{\mathbf{b}}_{\pi(2)})$, $\cdots$, $I(\tilde{\mathbf{b}}_{\pi(r-1)})$, $I(\tilde{\mathbf{b}}_{\pi(r)})$, where the largest MI $I(\tilde{\mathbf{b}}_{\pi(r)})$ is used for estimating the EXIT tunnel's open/closed state. This is carried out by checking, whether it is above the threshold $I_{th}(r) = T_r[I(\tilde{\mathbf{b}}_{\pi(1)}), I(\tilde{\mathbf{b}}_{\pi(2)}), ..., I(\tilde{\mathbf{b}}_{\pi(r-1)})]$ from the LUT. When the packet length N is sufficiently high, a straightforward comparison between $I(\tilde{\mathbf{b}}_{\pi(r)})$ and $I_{th}(r)$ may be used for confidently estimating the EXIT tunnel's open/closed state. Specifically, if $I(\tilde{\mathbf{b}}_{\pi(r)}) \leq I_{th}(r)$, the EXIT tunnel is deemed to be closed, otherwise it is deemed to be open.

However, satisfying the condition of $I(\tilde{\mathbf{b}}_{\pi(r)}) \leq I_{th}(r)$ cannot always provide a sufficiently reliable judgement of whether the trajectory can or cannot navigate through the EXIT tunnel to the $(1, 1)$ point. This is, because for short packets the trajectory may sometimes navigate through the tunnel that is marginally closed and vice versa [28]. Since our primary objective is to approach the maximum possible throughput, rather than waiting for the EXIT tunnel to open, it is desirable to allow iterative decoding to commence, even if the tunnel is marginally closed, especially when the packet length is short. This is achieved by modifying the threshold test according to $I(\tilde{\mathbf{b}}_{\pi(r)}) \leq I_{th}(r) - I_{diff}$, where I_{diff} is chosen to be the appropriate MI 'safety margin' for the specific packet length N employed. More particularly, if I_{diff} is chosen to be too high, then iterative decoding might commence at too low MI values, when there is no chance for the trajectory to navigate through the tunnel, hence unnecessarily increasing the complexity. By contrast, if I_{diff} is chosen to be too low, then iterative decoding will be deferred, even when there is a chance for the trajectory to navigate through the 'just' closed tunnel. This may potentially reduce the throughput. For this reason, we conceived the simulations detailed in Section 5.2.3 to determine appropriate values for I_{diff} for a range of packet lengths.

5.2.3 Performance Results

In this section, we evaluate the PLR, throughput and complexity of our previously proposed MCTC HARQ scheme [25] relying on the proposed LUT based DI strategy. This was achieved by simulating the transmission of a statistically relevant number of packets over a BPSK-modulated quasi-static Rayleigh fading channel. We also apply the LUT based DI strategy to Souza's systematic TCTC HARQ [125] and to the LTE system's systematic TCTC HARQ [149]. These three HARQ schemes which rely on the ES strategy proposed in [25] are used as our benchmarkers.

Souza's systematic TCTC HARQ transmits the systematic bit sequences $\mathbf{a}$ and the two parity bit sequences $\mathbf{b}_1$ and $\mathbf{b}_2$. The receiver activates iterative decoding between two parallel concatenated BCJR decoders after the third IR transmission. From the fourth IR transmission onwards, the repeated frame replica's LLRs are added to those gleaned from the previous transmissions. In contrast to the MCTC HARQ scheme, which may employ a unity-rate accumulator for obtaining the desirable PLR and throughput performances, Souza's HARQ scheme relies on the URC codes using octally represented memory-3 generator polynomials of $(17, 15)_o$ for achieving similar results, as seen in [25].

The LTE HARQ scheme adopts URC codes having different memory-3 polynomials of $(15, 13)_o$. The LTE standard specifies a particular interleaver, and a so-called 'rate matching' operation for selecting specific transmitted bits rather than transmitting all bits [149]. More explicitly, the standard defines its own interleaver between two parallel concatenated turbo encoders/decoders for a range of specific packet lengths. Furthermore, the systematic bit sequence $\mathbf{a}$ and the two parity bit sequences $\mathbf{b}_1$, $\mathbf{b}_2$ are interleaved again, according to the standard's so-called sub-block interleavers. The interleaved systematic bits are entered into a circular buffer. The interleaved parity bits fill in the rear part of the circular buffer, where the odd positions are from $\mathbf{b}_1$ and the even positions are from $\mathbf{b}_2$. Next, N transmitted bits are continuously selected from a specific starting point of this circular buffer. This starting point advances along the circular buffer, based on a standard-specific equation, which is a function of the transmission frame index. As a result, turbo decoding can be activated right after the first frame's transmission, since it contains some of the systematic bits as well as some of the two parity bit sequences. The repeated LLRs are also Chase combined with the corresponding previously received replicas at the receiver.

For each of these HARQ schemes, the transmission retry limit was set to $R = 6$ in order to prevent any particular message packet from unduly reserving the network resources, when communicating in hostile environments requiring a high number of retransmissions. For the sake of fair comparison, we increase the maximum number of transmissions defined as $R = 4$ in the LTE HARQ scheme to $R = 6$, following the standardized rule of locating the starting point of the circular buffer for each IR transmission. Our results were collected by transmitting source message packets comprising 48, 480 and 4800 bits over quasi-static Rayleigh fading channels, since these packet lengths can be supported by the LTE HARQ scheme. Additionally, the appropriate I_{diff} values were selected for these packet lengths in order to limit the maximum normalized throughput loss imposed by the DI strategy to be as low as 0.003. Table 5.7 shows the preferred I_{diff} values for the three HARQ schemes considered. Observe from Table 5.7 that for Souza's HARQ scheme, the preferred I_{diff} values are

Table 5.7: The preferred I_{diff} values for 48, 480 and 4800 bits packet lengths, when the allowed maximum throughput loss is 0.003.

	48 bits	480 bits	4800 bits
MCTC HARQ	0.08	0.01	0.0
Souza's HARQ	0.0	0.0	0.0
LTE HARQ	0.11	0.0	0.0

all zeros, regardless of how short the packet length is. This is because the determination of the EXIT tunnel's open/closed state only starts after the third transmission, and because there is seldom a 'just' open or 'just' closed EXIT tunnel.

Furthermore, we summarize the parameters in Table 5.8, which are employed in the above simulations comparing the PLR, throughput and complexity performance with and without the DI strategy for MCTC aided HARQ, for Souza's systematic TCTC aided HARQ and for the LET HARQ schemes.

Figure 5.20 shows the complexity versus SNR performance for the three HARQ schemes both with and without the DI strategy. We employ the same complexity metric as in [25], which was formulated as $Complexity = 2^m \cdot K$, where m is the number of memory elements employed in the convolutional encoders' generator polynomials and K is the total number of BCJR decoder executions performed during iterative decoding. As shown in Figure 5.20, the 'MCTC,ES+DI' HARQ scheme offers complexity reductions of approximately 10%, 20% and 20% for the packet lengths of 48, 480 and 4800 bits respectively, when compared to the 'MCTC,ES' scheme. However, when the LUT based DI is applied to Souza's systematic TCTC HARQ, the complexity reductions become about 35%, 32% and 30% for the 48, 480 and 4800-bit packet lengths,

Table 5.8: The simulation parameters for comparing the PLR, throughput and complexity performance with and without the DI strategy for MCTC aided HARQ, Souza's systematic TCTC aided HARQ and LET HARQ schemes.

Retry Limit	6
Packet Length	48-bit, 480-bit, 4800-bit
Stopping Strategy	ES
T_{inc}	see Table 5.9
T_{max}	0.99 for 48-bit packet length 0.999 for 480-bit, 4800-bit
Deferred Iteration	with (or) without
LUT step-size	0.1
I_{diff}	see Table 5.7
Modulation Scheme	BPSK
Channel Type	quasi-static Rayleigh fading

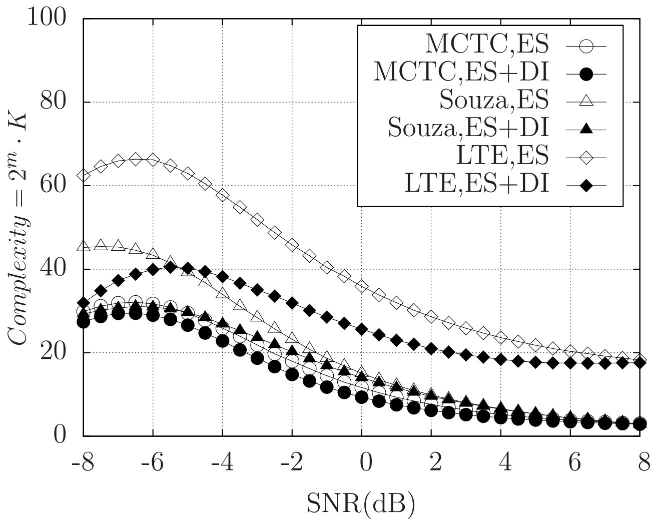
Table 5.9: The preferred T_{inc} values for 48, 480 and 4800 bits packet lengths which is found by the similar offline training in Section 5.1, when the ES strategy is equipped and the allowed throughput loss is less than 0.005.

	48 bits	480 bits	4800 bits
MCTC HARQ	0.018	0.011	0.005
Souza's HARQ	0.15	0.125	0.06
LTE HARQ	0.058	0.048	0.036

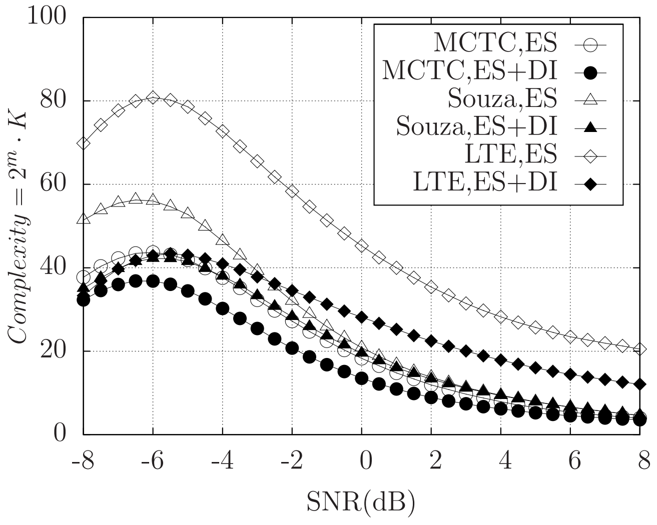
since Souza's scheme only relied on the ES strategy. Furthermore, the 'LTE,ES+DI' arrangement obtained the highest complexity reductions of up to 50% for all three packet lengths, since the LTE HARQ scheme activates the turbo decoding right away from the first transmission. The LUT-based DI aided MCTC HARQ scheme shows the lowest complexity among all HARQ schemes.

Let us now define the throughput as the ratio of the number of successfully delivered source message packets to the total number of transmitted packets. The left and right axes of Figure 5.21, respectively, illustrate the PLR and throughput performances, which are similar, regardless of which turbo HARQ scheme is used and whether the DI is employed, for all the three packet lengths considered. There is one exception, where the throughput of the LTE HARQ scheme becomes significantly lower than that of the other two HARQ schemes, namely at high SNRs[5]. This is because many packets may be successfully received after the first transmission attempt in the other two HARQ schemes, while in the LTE HARQ scheme, a minimum of two transmissions are needed

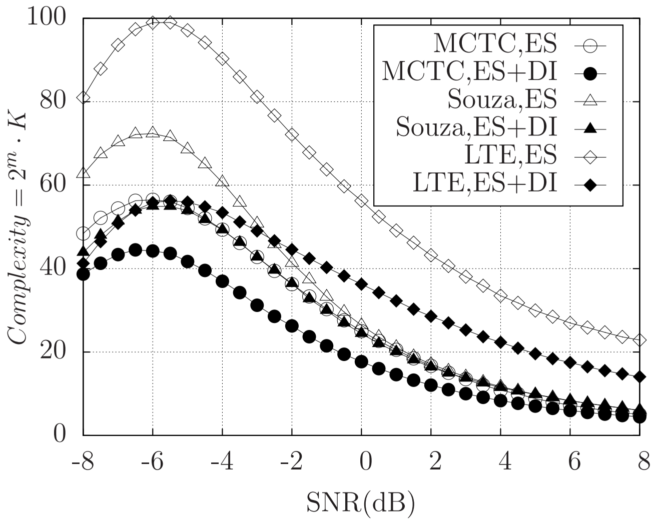
[5]In our simulations, the LTE HARQ has been implemented without the aid of other schemes specified in the LTE standard, for example the adaptive modulations.



(a) 48bits

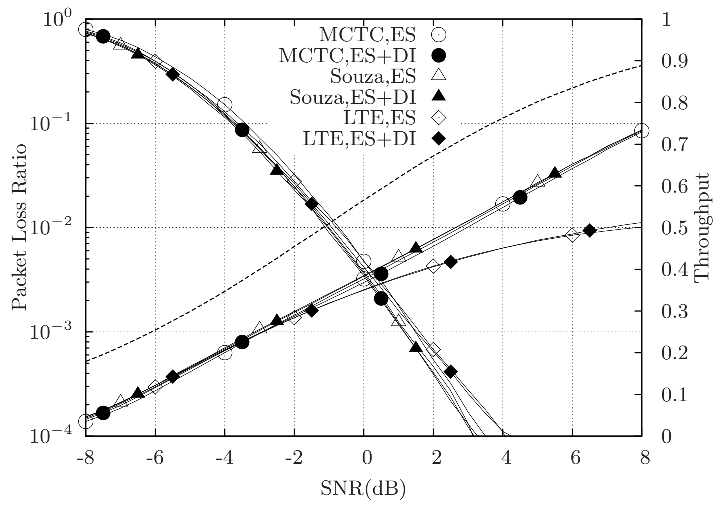


(b) 480bits

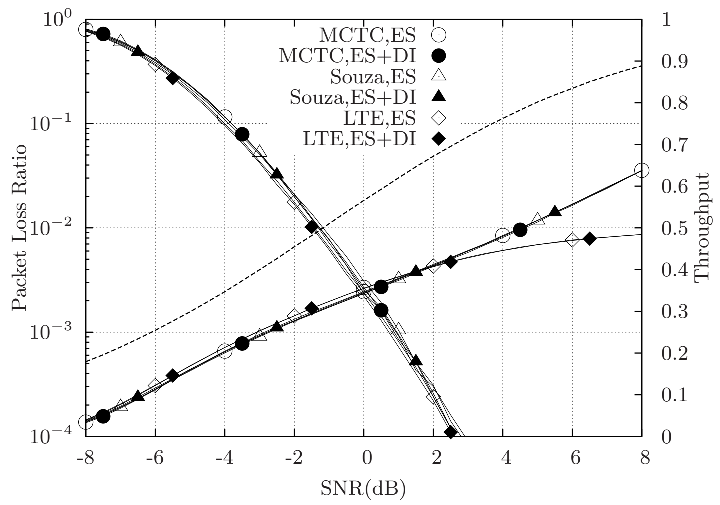


(c) 4800bits

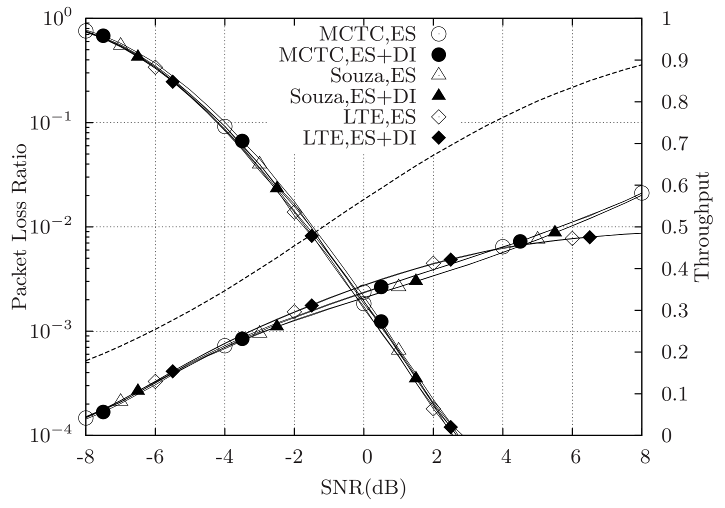
Figure 5.20: Complexity versus the quasi-static Rayleigh fading channel SNR for message packets of length a) 48 bits, b) 480 bits and c) 4800 bits. The schematic of 5.4, 5.5 and 5.19 for the MCTC HARQ scheme, the schematic of Figures 4.17 and 4.18 for Souza's scheme, as well as the parameters of Table 5.8 were used.



(a) 48bits



(b) 480bits



(c) 4800bits

Figure 5.21: PLR and throughput versus the quasi-static Rayleigh fading channel SNR for message packets of length a) 48 bits, b) 480 bits and c) 4800 bits. The dashed line represents the DCMC capacity. The schematic of 5.4, 5.5 and 5.19 for the MCTC HARQ scheme, the schematic of Figures 4.17 and 4.18 for Souza's scheme, as well as the parameters of Table 5.8 were used.

for recovering the source packet. Furthermore, the dashed curve seen in Figure 5.21 reveals the gap between the DCMC capacity and the throughput that these three HARQ schemes can achieve.

5.2.4 Conclusions

In this section, a generically applicable low-complexity DI aided turbo HARQ design was proposed and characterized. Complexity is a critical issue for any communication scheme employing turbo codes, especially for applications like HARQ, which may have to activate iterative decoding multiple times. As demonstrated in [25], the total complexity of the HARQ schemes dispensing with ES strategies may be particularly high. By contrast, the ES strategy aided turbo HARQ scheme of [25] was shown to significantly decrease this complexity. For the sake of decreasing the complexity further, this section proposed a more sophisticated DI strategy, which exploits the EXIT tunnel's open/closed state for determining, when iterative decoding should commence. At the cost of storing a modest LUT, the proposed scheme has been shown to decrease the complexity by 10% to 50% in the context of three recent turbo HARQ benchmarker schemes. This was achieved without imposing a significant degradation upon the throughput or PLR performance.

The DI strategy proposed in this section was conceived for transmission over quasi-static Rayleigh channels. However, it is also applicable in the situations where the MIs between two successive transmission are correlated. This may be readily illustrated. For example, the received MIs may be $\{0.15, 0.15, 0.16, 0.17\}$ during four consecutive transmissions, which have similar MIs for transmissions over a correlated Rayleigh fading channel. The threshold MI corresponding to the previous three MIs of $\{0.15, 0.15, 0.16\}$ is 0.68. Therefore, using the condition of $0.17 < 0.68$ suggests that we expect to have a closed EXIT tunnel for these four transmissions.

Although our LUT concept is specific for a particular code and channel, the LUT combined with MCTCs exhibited better performance than the LUT relying on TCTCs. Hence, the LUT combined with MCTCs may become the preferred design option for other scenarios, for example for relay networks. Furthermore, the LUT only has to be trained once for different scenarios, if the same code is used and similar channel conditions are encountered. Hence, when the LUT combined with MCTCs is applied in other scenarios, no additional training may be required. Therefore, the LUT based approach may be deemed general. Our future work will consider the application of the proposed DI strategy to other turbo coded schemes, including their distributed version used in relaying.

5.3 Chapter Summary

This chapter aimed for reducing the complexity of turbo HARQ schemes, since the complexity imposed may become significant due to the iterative decoding process following each reception. Observe from the expression of $Complexity = 2^m \cdot K$ that the memory length of the polynomial m has an exponential impact on the complexity. Therefore, the lowest-memory $m = 1$ polynomial $(2,3)_o$ becomes the best possible choice for low-complexity HARQ schemes. In Chapter 4, MCTC aided HARQ schemes have been demonstrated to attain the desirable performance based on this $(2,3)_o$ polynomial. Based on it, this chapter further considered the impact of the other factor, namely that of K in order to reduce the complexity of MCTC aided HARQ schemes.

Two strategies have been proposed in this chapter. The ES strategy introduced in Section 5.1 may curtail the iterative decoding process after each transmission attempt, depending on the MI increment provided. By contrast, the DI strategy proposed in Section 5.2 delays the commencement of iterative decoding according to the total received MI.

More specifically, in Section 5.1.1 we commenced by outlining the system model and the flow chart of our MCTC aided HARQ schemes in Figures 5.4, 5.5 and 5.6. Then, the above-mentioned ES strategy was detailed in Section 5.1.2, where two stopping thresholds were determined by an offline training process. Based on the preferred thresholds found by the offline training, the simulation results of Figure 5.13 have shown that the proposed ES strategy is capable of reducing the complexity by 85% compared to Souza's systematic TCTC aided HARQ scheme using a fixed number of 10 BCJR operations, which was achieved without degrading the PLR and throughput performance, as observed in Figures 5.11 and 5.12.

The DI philosophy was described in the context of the classic TCTCs and then it was extended to the family of the MCTC aided HARQ schemes in Section 5.2.1.1, which led to the creation of a LUT for storing the MI thresholds for all possible combinations of received MIs. Section 5.2.1.1 also demonstrated that the LUT size required for MCTC aided HARQ schemes may become excessive, if the LUT design is arbitrary. In order to minimize the associated storage requirement, we conceived a special LUT structure and employed multiple-dimensional linear interpolation in Section 5.2.1.2. Then, an efficient training process was proposed in Section 5.2.1.3 for generating the LUT of MCTC HARQ schemes. The resultant LUT size was summarized in Table 5.5 for both of these methods. We demonstrated that the LUT associated with the step size of $G = 0.1$ only has a few hundred rows for all T_2, T_3, $\cdots$, T_7 sub-tables. Based on the LUT generated, the DI aided MCTC HARQ strategy was elaborated on in Section 5.2.2. This section also proposed the appropriate DI strategy for short packet lengths by introducing a safety-margin for the MI thresholds stored in the LUT, which was referred to as I_{diff}. Finally, the simulation results of Figure 5.20 in Section 5.2.3 revealed that the DI strategy is capable of further reducing the complexity imposed by about 10% to 50%, yet maintaining a similar PLR and throughput performance, as seen in Figure 5.21.

Although the ES and DI strategies were invoked in the context of MCTC aided HARQ schemes in this chapter, both of them are general concepts. They may be extended into arbitrary scenarios employing turbo codes. For example, cooperative relay networks relying on distributed turbo codes may adopt the ES and DI strategies for improving their power efficiency.

Chapter 6

Conclusions and Future Work

In this chapter, we draw conclusions for each chapter and provide some ideas for our future work.

6.1 Conclusions

6.1.1 Chapter 1

Chapter 1 offered a brief introduction to the motivation and objectives of the thesis. During the evolution of networks, the seven-layer Open Systems Interconnection (OSI) architecture was formed. In this chapter, we introduced the basic functions of the five layers in the TCP/IP protocol suite, which is installed in almost all Internet terminals. The layered architecture facilitates the independent design of layers, but to a degree it suffers from the lack of flexible interaction between layers. However, cross-layer designs have been conceived by researchers to overcome the disadvantages of layering. In this chapter, the previous contributions on cross-layer optimization were reviewed and diverse cross-layer coding techniques conceived for wireless networks were highlighted. The chapter was concluded by the novel contributions of the thesis.

6.1.2 Chapter 2

In Chapter 2, we detailed the background used in this thesis. First of all, some basic concepts of information theory were highlighted in Section 2.1. In Section 2.2, we briefly reviewed the principles of turbo codes, including the encoder and decoder structures, the encoding process, the Log Likelihood Ratio (LLR) definition, the Bahl, Cocke, Jelinek and Raviv (BCJR) algorithm and iterative decoding, etc. This background was exploited in our later discourse on Multiple Component Turbo Codes (MCTCs) in Chapter 4, which will be combined with Hybrid Automatic Repeat reQuest (HARQ) in Chapters 4 and 5. Since Fountain codes were adopted for protecting file transfer at the application layer in Chapter 3, the encoding and decoding processes of three types of Fountain codes, namely random linear Fountain codes, Luby transform codes and Raptor codes were elaborated on in this chapter. Chapter 2 was concluded by commenting on the integration of two different simulation platforms: NS2 and IT++, both of which are employed in our simulations.

6.1.3 Chapter 3

Chapter 3 invoked Application Layer Forward Error Correction (AL-FEC), which employs Fountain codes to guarantee successful file transfer at the application layer in 802.11 Wireless Local Area Networks (WLANs). The 802.11 WLANs adopt the IEEE 802.11 MAC protocol to provide error control for link-layer transmissions, which relies on an Automatic Repeat reQuest (ARQ) mechanism. Furthermore, the 802.11 MAC protocol recommends a decentralized mode referred to as Distributed Coordination Function (DCF) to contend for access to the medium, which exchanges control packets between the transmitter and the receiver, such as Request To Send (RTS) and Clear To Send (CTS). Since Fountain codes operate on the basis of packets, which jointly constitute a file, the packet size has to be selected appropriately. A large packet may result in a high Packet Loss Ratio (PLR), while the short packet size implies having more packets, and hence results in an extra load imposed by headers and RTS/CTS control packets. Therefore, there is a trade-off between the packet size and the load. We formulate the PLR at the application layer considering the 802.11 MAC retransmission mechanism and the Quadrature Phase Shift Keying (QPSK) modulation scheme in the physical layer. Furthermore, we derive an analytical formula for characterizing the achievable transmission efficiency as a function of the packet size. Based on these expressions, the optimum packet size may be determined. Our simulation results verified the correctness of the theoretical PLR expressions and demonstrated that the optimum packet size indeed achieves the highest transmission efficiency.

6.1.4 Chapter 4

Chapter 4 theoretically analyzes MCTCs and combines them with HARQ. MCTCs concatenate more than two component codes in parallel and perform iterative decoding by exchanging extrinsic information between them. In Section 4.1, we proposed a novel method for partitioning N-component turbo codes into two logical parts. One of the parts is constituted by an individual BCJR decoder, while the other so-called composite decoder consists of the remaining $(N-1)$ components. Then we apply Extrinsic Information Transfer (EXIT) charts to visualize the extrinsic information exchange between these two parts. Based on these EXIT charts, it becomes possible to determine for MCTCs the 'open tunnel SNR threshold', which is defined as the minimum SNR for which the EXIT chart has an open tunnel. Furthermore, this metric characterizes the achievable performance of the codes considered. In order to find the 'open tunnel SNR threshold', the EXIT function of a BCJR decoder may be described by appropriately fitted spline functions. The *a priori* information input into one of the N BCJR decoders may be obtained by combining the extrinsic information gleaned from all other decoders. We also compared the achievable performances of MCTCs, non-systematic TCTCs and systematic TCTCs. The MCTCs outperformed non-systematic TCTCs but performed slightly worse than systematic TCTCs, when the employment of generator polynomials imposing a relatively higher complexity were affordable. By contrast, when considering the same complexity, MCTCs may achieve significantly lower Bit Error Ratios (BERs) than non-systematic TCTCs.

A desirable PLR and throughput performance is expected, when combining MCTCs with HARQ. In Section 4.2, four different turbo HARQ schemes were recommended and compared. One of them is the state-of-the-art systematic TCTC aided HARQ scheme proposed by Souza in [125], which suggests that the transmitter repeatedly transmits the systematic bit sequence $\mathbf{a}$ and the parity bit sequences $\mathbf{b_1}, \mathbf{b_2}$ from two Unity Rate Code (URC) encoders. The receiver combines the same replicas and performs iterative decoding between two BCJR decoders. By contrast, the transmitter used in our proposed MCTC aided HARQ scheme incrementally transmits its redundant sequences $\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}, \mathbf{b_4}, ...$ followed by the systematic bit sequence $\mathbf{a}$. The receiver combines them in order to construct an N-component turbo code and performs iterative decoding among these N components. The other two turbo HARQ schemes consider the intermediate solutions, for example, the puncturing-aided Souza scheme, which punctures $\mathbf{b_1}, \mathbf{b_2}$ during the second transmission, and the fixed 3-component turbo coded HARQ. Our simulation results showed that the proposed MCTC aided HARQ scheme achieves a higher throughput and a lower PLR than Souza's systematic TCTC aided HARQ scheme, when employing the generator polynomial $(2, 3)_o$. If Souza's scheme adopts the original polynomial $(17, 15)_o$ resulting in a higher complexity, it has a sightly better throughput and PLR performance, compared to that of the MCTC HARQ having the 1-memory polynomial of $(2, 3)_o$. However, this is obtained at the cost of significantly higher complexity than that of Souza's scheme using the polynomial $(17, 15)_o$.

6.1.5 Chapter 5

Chapter 5 improves the MCTC aided HARQ scheme of Chapter 4 by employing two novel methods for reducing the complexity imposed.

In Section 5.1, we proposed an Early Stopping (ES) aided decoding strategy to replace the original fixed number of BCJR operations during the iterative decoding following each transmission. The ES strategy estimates the Mutual Information (MI) increment between two BCJR operations performed by each individual decoder. When the MI increment of any of the BCJR decoders drops below a threshold, this implies that the iterative decoding has converged and using further iterations no longer enhances the attainable decoding performance. Therefore, the iterative decoding should be stopped if the MI increment becomes less than a pre-determined threshold. In order to find the appropriate threshold, a training process was designed to investigate the trends between the attainable throughput and the complexity imposed. The preferred threshold should not unduly degrade the throughput, while imposing a significantly reduced complexity. Our simulation results showed that the ES assisted MCTC aided HARQ scheme reduced the complexity by as much as 80%, compared to the MCTC aided HARQ scheme operating without the ES strategy and to Souza's original scheme, while maintaining a similar PLR and throughput performance.

Section 5.2.2 utilizes the EXIT tunnel's open/closed status to further decrease the complexity of the ES aided MCTC HARQ scheme. More explicitly, the new scheme delays the activation of the iterative decoding for the MCTC decoder until an open tunnel appears, which we refer to as the Deferred Iteration (DI) strategy. The EXIT tunnel's status may be determined by comparing the currently received Mutual Information (MI) to a pre-stored threshold, which is quantified by the MI that makes the EXIT tunnel to become 'marginally' open, when it is combined with all previously received MI contributions. In this section, a Look-up Table (LUT) was conceived for storing the threshold MIs for all possible combinations of received MIs in advance. The LUT was designed to minimize its storage requirement. Furthermore, a fast training algorithm was developed to generate the LUT, based on the analytical EXIT functions modeled by spline-fitting and using information combining techniques of Chapter 4. As a result, the LUT only contains a few hundred entries. When applying the DI strategy for MCTC aided HARQ schemes, the MI fluctuation associated with short packets has also been considered in our solution. Our simulation results demonstrated that the LUT based DI aided MCTC HARQ scheme may further reduce the complexity imposed by up to 50%, compared to that imposed by the ES strategy operating in isolation.

6.2 Future work

The results of the thesis may also be extended to relay-aided networks in the future.

6.2.1 Background

Relay-aided networks were proposed and analyzed in [150, 151]. They are capable of increasing the coverage area and throughput of networks. In a relay-aided network, when a source transmits data to a destination, the terminals roaming in the coverage area may assist the reliable information delivery between the source and destination.

Researchers have investigated cooperation aided relay networks based on distributed channel codes, for example in [104]. More explicitly, during the evolution of relay networks, three classic relay strategies have been developed: Amplify-and-Forward (AF) [152], Decode-and-Forward (DF) [153] and Compress-and-Forward (CF) [154].

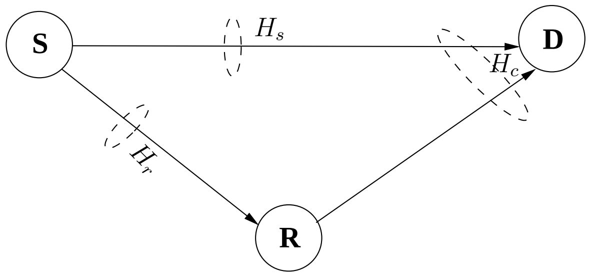


Figure 6.1: A single-hop, single-relay network.

The most common model is the single-hop single-relay network, which is illustrated in Figure 6.1. In this relay network, transmissions from the source to destination are divided into two phases. In phase one, the source broadcasts data and the relay listens. In phase two, the relay transmits the successfully received symbols to the destination. Assuming that the channel realizations are H_s, H_r and H_c respectively for the source to destination, the source to relay and the compound channel, as seen in Figure 6.1, the rate R of the source obeys the following inequalities [155, 156]:

$$R < fC(H_s, \gamma) + (1 - f)C(H_c, \gamma) \ , \tag{6.1}$$

and

$$R < fC(H_r, \gamma) \ , \tag{6.2}$$

simultaneously or

$$R < C(H_s, \gamma) \ , \tag{6.3}$$

where $f \in [0, 1]$ indicates that the relay listens during the entire period of the cooperative transmission and where γ is the SNR. A relay-aided system may also be viewed as a Multi-Input and Multi-Output (MIMO) system having distributed elements constituted by the individual elements of single-antenna-aided mobiles.

6.2.2 Motivation of Rateless Relay-Aided Networks

The outage probability constitutes an important metric conceived for characterizing relay networks, which is defined as the probability that the achievable rate R fails to satisfy the inequalities of 6.1, 6.2 and 6.3 [155, 156]. For DF relaying networks communicating over Rayleigh fading channels, the fixed rate R of conventional channel codes cannot ensure to have an outage probability of zero without having perfect instantaneous channel state information at the transmitter, since the channel conditions vary as a function of time. In order to attain the lowest possible outage probability, a low fixed-rate R may be adopted. This will however waste some of the bandwidth for transmissions under good channel conditions.

However, rateless codes such as the LT codes or Raptor codes discussed in Section 2.3 are capable of automatically adjusting the coding rate according to the time-varying channel quality. Furthermore, rateless codes have the beneficial property that successful decoding may be carried out based on any set of packets extracted from all encoded packets, regardless whether these encoded packets are from the source or the relay. Therefore, when rateless codes are employed to aid the operation of relaying networks, the transmitter does not have to know about the availability of relays, it does not even have to know the collaborative protocols that the relay adopted. The transmitter may stop transmitting encoded packets, when it received an ACKnowledgement (ACK) message from the relay or the destination. In recent years, these advantages have further fuelled the improvement of rateless relay-aided networks.

6.2.3 Rateless Relay-Aided Networks Review

A number of contributions have been published on rateless relay-aided networks [155–164]. They involve different research aspects of rateless relay-aided networks, including the design of distributed Luby Transform (LT) codes [159], the performance analysis of Fountain codes for transmissions over fading relay channels and so on. We summarized some of them in Table 6.1.

Apart from the simple three-node rateless relay network shown in Figure 6.1, the authors of [155–164] have considered more complex system models, such as multiple-relay aided networks, multi-hop networks and so on, which are summarized in Figure 6.2.

Based on their specific models, these contributions proposed the corresponding cooperative protocols using rateless codes. They are basically divided into the following categories:

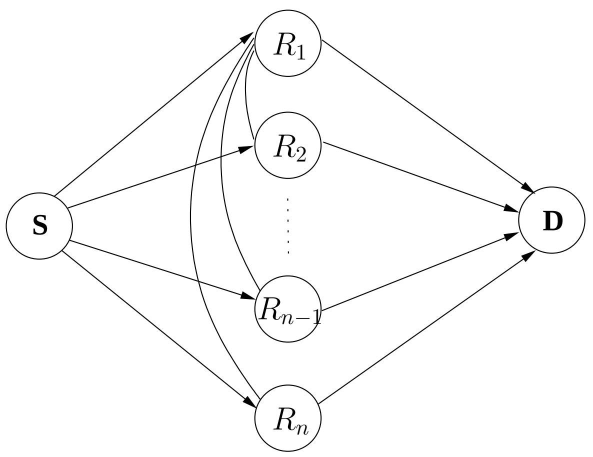
1). For the single-hop single-relay networks of Figure 6.1, the cooperative protocol contains two phases, namely the listening phase and the collaborative phase. In the listening phase, the source transmits Fountain encoded data to both the relay and to the destination. Both of them attempt decoding and if the relay succeeds first, it enters into the collaborative phase. In this second phase, different protocols activate different steps. For example, the source and the relay may cooperatively transmit the same encoded symbols using a space-time code [155, 156]. Alternatively, the source stops transmission, while the relay transmits the re-encoded data to the destination [160]. Another option is that the destination ignores the symbols received from the source during the first phase [160]. This is equivalent to the two-hop relay networks.

Table 6.1: Major contributions on rateless relay-aided networks.

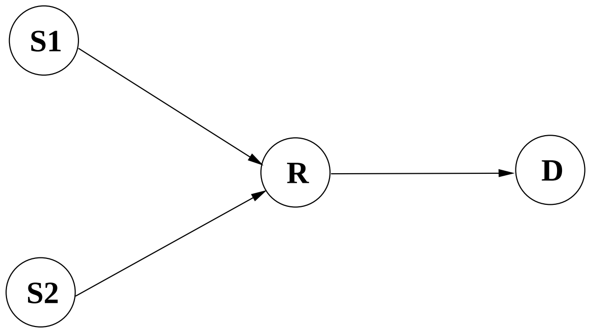
Author(s)	Contribution
Molisch *et al.* 2006 [157] Molisch *et al.* 2007 [158]	investigated cooperative protocols conceived for Fountain coded relay networks relying on N parallel relays, and optimized L when the relays start forwarding after L relay nodes have successfully decoded the source data.
Castura *et al.* 2007 [155, 156]	proposed a coding framework for rateless relay networks, and analyzed the achievable coding rate.
Puducheri *et al.* 2007 [159]	designed distributed LT (DLT) codes and detailed the decomposition of LT codes into DLT. The DLT can be used for multiple sources in a network. Simulation results showed substantial performance benefits.
Liu *et al.* 2009 [160]	conceived several cooperative protocols for fountain coded relay aided networks for transmissions over fading channels. The time division protocol was found to be superior to the space-time one.
Wang *et al.* 2010 [161]	proposed three rateless coded forwarding schemes for linear multi-hop networks and further analyzed their performance.
Mehta *et al.* 2011 [162]	analyzed the performance of rateless coded multiple-relay assisted networks, where the relays can buffer the successfully received packets. Simulation results demonstrated that buffering further improves the benefits of cooperation.
Ravanshid *et al.* 2011 [163]	investigated half-duplex dynamic DF relay networks based on rateless codes, where the duration of the listening phase was not fixed.
Uppal *et al.* 2011 [164]	proposed a cooperative protocol for rateless relay networks by combining DF as well as CF schemes, and derived their theoretical performance bounds.

2). For multiple-relay aided rateless coded networks such as the one shown in Figure 6.2-(a), the source broadcasts the Fountain encoded data to all relays. The source will stop its transmissions after L relays successfully decoded the data and sent back ACK messages. Then, these L relays commence their transmissions. They may transmit the same or different re-encoded data to the destination using for example direct-sequence spectrum spreading technology for the sake of avoiding interference [157, 158].

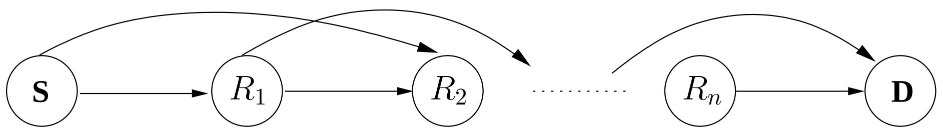
3). The authors of [157, 158] suggested an alternative protocol that a relay commences its transmission immediately, once it decodes the data successfully. Other relays still operating in the reception mode may receive the encoded data from both the source as well as from the relays operating in the transmission mode, which are distinguished by different spreading codes. The destination starts receiving, when the first relay successfully completed its reception.



(a) multiple relays



(b) multiple sources



(c) multiple hops

Figure 6.2: The schematic of rateless relay networks.

4). For multiple-source relay networks seen in Figure 6.2-(b), where the communication between the sources is impossible, the two sources encode their data using Fountain codes and then time-multiplex the resultant encoded data for transmission to the relay [159].

5). For the multi-hop relay networks portrayed in Figure 6.2-(c), three cooperative protocols were proposed in [161]: 'multi-hop forwarding with no spatial use', 'multi-hop forwarding with spatial use' and 'cooperative forwarding with spatial use'. In the multi-hop forwarding protocol relying on no spatial use, only one node transmits data at any time. The transmissions commence from the source to the closest relay, while all other nodes can overhear the encoded packets. Then, the forwarding continues from one relay to the next, until the destination decodes the data. In the context of the 'multi-hop forwarding with spatial use' protocol, the nodes K-hop away may transmit in a time division manner. Each node continuously transmits Fountain encoded data, until the next hop successfully decodes. Finally, in the 'cooperative forwarding with spatial use' protocol, the nodes K-hop away may also transmit in a time division structure, as well as the adjacent nodes in a K-node group may transmit simultaneously on different non-interference channels.

6.2.4 Future Research Topics on Rateless Relay Networks

As mentioned in Section 2.3, the complexity of Fountain codes is determined by the degree distribution. The design of the Fountain codes' degree distribution aims for generating less high-degree encoded packets and a high proportion of low-degree encoded packets. When the Fountain codes are invoked in relay networks, the degree distribution may be adapted according to the specific positions of the source and the relay, since the relay is usually located between the source and the destination, and hence the source-to-relay and the relay-to-destination channel may have a better quality than that of the direct link between the source and destination. Based on these different channel conditions, we may arrange the source and the relay to jointly decide upon the degree distribution used.

In rateless coded relay networks, the Fountain codes used at the source or the relay stop encoding and transmitting, when they receive an ACK feedback. When the Fountain codes are employed for transmission over a Binary Erasure Channel (BEC) such as the Internet, the decoding operation commences from the first received degree-one encoded packet based on the message passing algorithm of LT codes. By contrast, if the Fountain codes are used for transmission over an AWGN channel [165] or a fading channel [166], the soft iterative decoding commences after a fixed number of encoded symbols has been received. However, this is not a desirable method, since in the spirit of our DI-method of Chapter 5 fruitless decoding operations may be performed, if insufficient mutual information has been received. In our further work, we can exploit Equation 2.17 and the EXIT charts of Fountain codes to determine the amount of mutual information required for successful decoding and immediately send back an ACK message to cease transmissions.

6.2.5 Future Research Ideas

Increasingly, the related research ideas are concluded here:

- Study the delay of the ARQ schemes in more details, for example by generating the histogram of the number of IR-transmissions;
- Investigate and compare the benefits of puncturing in the context of the schemes conceived;
- Combine both the ES and DI ideas with relay-aided ARQs.

Glossary

3GPP	3rd Generation Partnership Project.
ACK	ACKnowledgement.
AF	Amplify-and-Forward.
AODV	Ad-hoc On-Demand Distance Vector.
ARPA	Advanced Research Projects Agency.
AWGN	Additive White Gaussian Noise.
BCH	Bose-Chaudhuri-Hocquenghem.
BCJR	Bahl, Cocke, Jelinek and Raviv.
BEC	Binary Erasure Channel.
BER	Bit Error Ratio.
BPSK	Binary Phase-Shift Keying.
CBR	Constants Bit Rate.
CC	Convolutional Codes.
CDF	Cumulative Distribution Function.
CDP	Content Delivery Protocol.
CF	Compress-and-Forward.
CLD	Cross Layer Design.
CLO	Cross Layer Optimization.
CPU	Central Processing Unit.
CRC	Cyclic Redundancy Check.
CSMA/CA	Carrier Sense Multiple Access combined with Collision Avoidance.
CTS	Clear To Send.
DARPA	Defense Advanced Research Projects Agency.

DCF	Distributed Coordination Function.
DCMC	Discrete input Continuous output Memoryless Channel.
DF	Decode-and-Forward.
DI	Deferred Iteration.
DS	Direct Sequence.
DSDV	Destination-Sequenced Distance Vector.
DSR	Dynamic Source Routing.
DVB	Digital Video Broadcasting.
ES	Early Stopping.
EXIT	Extrinsic Information Transfer.
FEC	Forward Error Correction.
FH	Frequency Hopping.
FTP	File Transfer Protocol.
HARQ	Hybrid Automatic Repeat reQuest.
HTTP	Hypertext Transfer Protocol.
IEEE	Institute of Electrical and Electronics Engineers.
IP	Internet Protocol.
IPTV	Internet Protocol based Television.
IR	Information Redundancy.
LDPC	Low Density Parity Check.
LLR	Log Likelihood Ratio.
LT	Luby Transform.
LUT	Look-Up Table.
MAC	Media Access Control.
MAP	Maximum *a posteriori*.
MBMS	Multimedia Broadcast Multicast Service.
MCTC	Multiple Component Turbo Code.
MI	Mutual Information.
MIMO	Multiple-Input Multiple-Output.
MIT	Massachusetts Institute of Technology.
NAV	Network Allocation Vector.

OFDM	Orthogonal Frequency-Division Multiplexing.
OSI	Open Systems Interconnection.
PCF	Point Coordination Function.
PDF	Probability Density Functions.
PLCP	Physical Layer Convergence Procedure.
PLR	Packet Loss Ratio.
QoS	Quality of Services.
QPSK	Quadrature Phase Shift Keying.
RS	Reed-Solomon.
RSC	Recursive Systematic Convolutional.
RTS	Request To Send.
SBI	Source Block Index.
SER	Symbol Error Ratio.
SIFS	Short Interframe Space.
SMTP	Simple Mail Transfer Protocol.
SNR	Signal to Noise Ratio.
TCL	Tool Command Language.
TCP	Transport Control Protocol.
TCTC	Twin-Component Turbo Code.
TORA	Temporally-Ordered Routing Algorithm.
TP1	Truncated Poisson 1.
UDP	User Datagram Protocol.
UEP	Unequal Error Protection.
URC	Unity Rate Code.
WLAN	Wireless Local Area Network.
WMAN	Wireless Metropolitan Area Networks.
WPAN	Wireless Personal Area Network.

Bibliography

[1] N. Abramson, "The ALOHA system: another alternative for computer communications," in *Proceedings of the fall joint computer conference (AFIPS 1970-Fall)*, pp. 281–285, November 1970.

[2] S. L. Mathison, "Telenet inaugurates service," *ACM SIGCOMM Computer Communication Review*, vol. 5, pp. 24–28, October 1975.

[3] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach (fifth edition)*. Boston, USA: Pearson Addition-Wesley, 2009.

[4] "RFC793: Transmission Control Protocol." Network Working Group, IETF, September, 1981. Downloadable at http://www.ietf.org/rfc/rfc793.txt.

[5] "RFC768: User Datagram Protocol." Network Working Group, IETF, August, 1980. Downloadable at http://www.ietf.org/rfc/rfc768.txt.

[6] "RFC791: Internet Protocol." Network Working Group, IETF, September, 1981. Downloadable at http://www.ietf.org/rfc/rfc791.txt.

[7] L. L. Peterson and B. S. Davie, *Computer networks: a system approach (fourth edition)*. San Francisco, USA: Morgan Kaufmann, 2007.

[8] "RFC2616: Hypertext Transfer Protocol – HTTP/1.1." Network Working Group, IETF, June, 1999. Downloadable at http://www.ietf.org/rfc/rfc2616.txt.

[9] "RFC959: file transfer protocol (FTP)." Network Working Group, IETF, October, 1985. Downloadable at http://www.ietf.org/rfc/rfc959.txt.

[10] "RFC5321: Simple Mail Transfer Protocol." Network Working Group, IETF, October, 2008. Downloadable at http://www.ietf.org/rfc/rfc5321.txt.

[11] S. Khan, Y. Peng, E. Steinbach, M. Sgroi, and W. Kellerer, "Application-driven cross-layer optimization for video streaming over wireless networks," *IEEE Communications Magazine*, vol. 44, pp. 122 – 130, January 2006.

[12] G. Song and Y. Li, "Cross-layer optimization for OFDM wireless networks-part I: theoretical framework," *IEEE Transactions on Wireless Communications*, vol. 4, pp. 614 – 624, March 2005.

[13] F. Fu and M. van der Schaar, “A new systematic framework for autonomous cross-layer optimization,” *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 1887 –1903, May 2009.

[14] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod, “Cross-layer design of ad hoc networks for real-time video streaming,” *IEEE Wireless Communications*, vol. 12, pp. 59 – 65, August 2005.

[15] P. Skraba, H. Aghajan, and A. Bahai, “Cross-layer optimization for high density sensor networks: distributed passive routing decisions,” *in Proceedings of Ad-Hoc Now' 04*, vol. 266–279, 2004.

[16] R. Madan, S. Cui, S. Lall, and A. Goldsmith, “Cross-layer design for lifetime maximization in interference-limited wireless sensor networks,” in *Proceedings of the INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1964 – 1975, March 2005.

[17] G. Song and Y. Li, “Cross-layer optimization for OFDM wireless networks-part II: algorithm development,” *IEEE Transactions on Wireless Communications*, vol. 4, pp. 625 – 634, March 2005.

[18] G. Miao, N. Himayat, Y. Li, and A. Swami, “Cross-layer optimization for energy-efficient wireless communications: a survey,” *Wireless Communications and Mobile Computing*, vol. 9, pp. 529–542, April 2009.

[19] S. H. Low and J. C. Doyle, “Cross-layer optimization in TCP/IP networks,” *IEEE/ACM Transactions on Networking*, vol. 13, pp. 582–595, June 2005.

[20] M. Catalan, A. Calveras, and S. Galvez, “Cross-layer optimization of reliable transmissions over IEEE 802.11 multi-hop networks,” in *Wireless Communication Systems, 2006. ISWCS '06. 3rd International Symposium on*, pp. 650 –654, September 2006.

[21] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, pp. 379–427, 1948.

[22] H. Chen, R. G. Maunder, and L. Hanzo, “Fountain-code aided file transfer in 802.11 WLANs,” in *Proceedings of the IEEE Vehicular Technology Conference (VTC 2009-Fall)*, September 2009.

[23] H. Chen, R. G. Maunder, and L. Hanzo, “Multi-level turbo decoding assisted soft combining aided hybrid ARQ,” in *Proceedings of the IEEE Vehicular Technology Conference (VTC 2010-Spring)*, May 2010.

[24] H. Chen, R. G. Maunder, and L. Hanzo, “An EXIT-chart aided design procedure for near-capacity N-component parallel concatenated codes,” in *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2010)*, December 2010.

[25] H. Chen, R. G. Maunder, and L. Hanzo, "Low-complexity multiple-component turbo-decoding-aided hybrid ARQ," *IEEE Transactions on Vehicular Technology*, vol. 60, pp. 1571 –1577, May 2011.

[26] H. Chen, R. G. Maunder, and L. Hanzo, "Deferred-iteration aided low-complexity turbo hybrid ARQ relying on a look-up table," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2011)*, December 2011.

[27] H. Chen, R. G. Maunder, and L. Hanzo, "Lookup-table-based deferred-iteration aided low-complexity turbo hybrid ARQ," *IEEE Transactions on Vehicular Technology*, vol. 60, pp. 3045 –3053, September 2011.

[28] J. W. Lee and R. E. Blahut, "Generalized EXIT chart and BER analysis of finite-length turbo codes," in *Proceedings of the IEEE Global Communications Conference, (GLOBECOM 2003)*, vol. 4, pp. 2067 – 2072, December 2003.

[29] R. G. Gallager, *Information theory and reliable communication.* John Wiley & Sons, 1968.

[30] L. Hanzo, S. X. Ng, T. Keller, and W. Webb, *Quadrature amplitude modulation.* John Wiley & Sons, 2004.

[31] L. Hanzo, R. G. Maunder, and L. L. Yang, *Near-capacity variable-length coding: regular and EXIT-chart-aided irregular designs.* John Wiley & Sons, 2011.

[32] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: turbo-codes. 1," *Proceedings of the IEEE International Conference on Communications (ICC 1993)*, vol. 2, pp. 1064–1072, May 1993.

[33] S. Riedel, "MAP decoding of convolutional codes using reciprocal dual codes," *IEEE Transactions on Information Theory*, vol. 44, pp. 1176–1187, May 1998.

[34] A. Taffin, "Generalised stopping criterion for iterative decoders," *IET Electronics Letters*, vol. 39, p. 993, June 2003.

[35] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Analysis, design, and iterative decoding of double serially concatenated codes with interleavers," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 231–244, February 1998.

[36] F. Daneshgaran, M. Laddomada, and M. Mondin, "Iterative joint channel decoding of correlated sources employing serially concatenated convolutional codes," *IEEE Transactions on Information Theory*, vol. 51, pp. 2721 – 2731, July 2005.

[37] J. Yuan, B. Vucetic, and W. Feng, "Combined turbo codes and interleaver design," *IEEE Transactions on Communications*, vol. 47, pp. 484–487, April 1999.

[38] M. Z. Wang, A. Sheikh, and F. Qi, "Interleaver design for short turbo codes," *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 1999)*, pp. 894–898, December 1999.

[39] J. Garcia-Frias and J. D. Villasenor, “Combined turbo detection and decoding for unknown ISI channels,” *IEEE Transactions on Communications*, vol. 51, pp. 79–85, January 2003.

[40] O. Alamri, J. Wang, S. X. Ng, L. L. Yang, and L. Hanzo, “Near-capacity three-stage turbo detection of irregular convolutional coded joint sphere-packing modulation and space-time coding,” *IEEE Transactions on Communications*, vol. 57, pp. 1486–1495, May 2009.

[41] C. Douillard, M. Jézéquel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, “Iterative correction of intersymbol interference: turbo-equalization,” *European Transactions on Telecommunications*, pp. 507–511, September 1995.

[42] M. Tuchler, R. Koetter, and A. C. Singer, “Turbo equalization: principles and new results,” *IEEE Transactions on Communications*, vol. 50, pp. 754–767, May 2002.

[43] S. Vishwanath and A. Goldsmith, “Adaptive turbo-coded modulation for flat-fading channels,” *IEEE Transactions on Communications*, vol. 51, pp. 964–972, June 2003.

[44] E. Rosnes and O. Ytrehus, “On the design of bit-interleaved turbo-coded modulation with low error floors,” *IEEE Transactions on Communications*, vol. 54, pp. 1563–1573, September 2006.

[45] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate (Corresp.),” *IEEE Transactions on Information Theory*, pp. 284–287, March 1974.

[46] L. Hanzo, J. P. Woodard, and P. Robertson, “Turbo Decoding and Detection for Wireless Applications,” *Proceedings of the IEEE*, vol. 95, June 2007.

[47] L. Hanzo, T. H. Liew, and B. L. Yeap, *Turbo coding, turbo equalisation and space-time coding for transmission over fading channels, second edition.* New York, USA: John Wiley & Sons, 2011.

[48] S. ten Brink, “Convergence behavior of iteratively decoded parallel concatenated codes,” *IEEE Transactions on Communications*, vol. 49, pp. 1727–1737, October 2001.

[49] M. Tuchler, “Design of serially concatenated systems depending on the block length,” *IEEE Transactions on Communications*, vol. 52, pp. 209–218, February 2004.

[50] H. Chen and A. Haimovich, “EXIT charts for turbo trellis-coded modulation,” *IEEE Communications Letters*, vol. 8, pp. 668–670, November 2004.

[51] R. Tee, R. G. Maunder, and L. Hanzo, “EXIT-chart aided near-capacity irregular bit-interleaved coded modulation design,” *IEEE Transactions on Wireless Communications*, vol. 8, pp. 32–37, January 2009.

[52] F. Babich, A. Crismani, and R. G. Maunder, “EXIT chart aided design of periodically punctured turbo codes,” *IET Electronics Letters*, vol. 46, p. 1001, July 2010.

[53] J. Hagenauer, "The EXIT chart - introduction to extrinsic information transfer in iterative processing," in *Proceeding of 12th European Signal Processing Conference (EUSIPCO)*, pp. 1541–1548, September 2004.

[54] D. J. C. MacKay, "Fountain codes," *IEE Proceedings-Communications*, vol. 152, pp. 1062 – 1068, December 2005.

[55] M. Luby, "LT codes," in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.*, pp. 271 – 280, 2002.

[56] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, pp. 2033–2051, May 2006.

[57] C. Berger, S. Zhou, Y. Wen, P. Willett, and K. Pattipati, "Optimizing joint erasure- and error-correction coding for wireless packet transmissions," *IEEE Transactions on Wireless Communications*, vol. 7, pp. 4586–4595, November 2008.

[58] H. Kushwaha, Y. Xing, R. Chandramouli, and H. Heffes, "Reliable multimedia transmission over cognitive radio networks using Fountain codes," *Proceedings of the IEEE*, vol. 96, pp. 155–165, January 2008.

[59] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. X. Xiong, "Scalable video multicast ssing expanding window Fountain codes," *IEEE Transactions on Multimedia*, vol. 11, pp. 1094–1104, October 2009.

[60] D. Sejdinovic, R. Piechocki, A. Doufexi, and M. Ismail, "Fountain code design for data multicast with side information," *IEEE Transactions on Wireless Communications*, vol. 8, pp. 5155–5165, October 2009.

[61] D. Gomez-Barquero, N. Cardona, A. Bria, and J. Zander, "Affordable mobile TV services in hybrid cellular and DVB-H systems," *IEEE Network*, vol. 21, pp. 34–40, March 2007.

[62] D. J. C. MacKay, *Information theory, inference, and learning algorithms*, vol. 22. Cambridge, UK: Cambridge University Press, June 2003.

[63] K. Atkinson, *An introduction to numerical analysis, second edition.* New York, USA: John Wiley & Sons, 1989.

[64] G. H. Golub and C. F. Van Loan, *Matrix computations, third edition.* Baltimore, Maryland, USA: John Hopkins, 1996.

[65] Q. Xu, V. Stankovic, and Z. X. Xiong, "Distributed joint source-channel coding of video using raptor codes," *IEEE Journal on Selected Areas in Communications*, vol. 25, pp. 851–861, May 2007.

[66] K. Noh, S. Yoon, and J. Heo, "Performance analysis of network coding with raptor codes for IPTV," *IEEE Transactions on Consumer Electronics*, vol. 55, pp. 83–87, February 2009.

[67] M. Fresia, L. Vandendorpe, and H. V. Poor, “Distributed source coding using Raptor codes for hidden Markov sources,” *IEEE Transactions on Signal Processing*, vol. 57, pp. 2868–2875, July 2009.

[68] ETSI, *IP datacast over DVB-H: content delivery protocols*, June 2006. TS 102472 V1.1.1.

[69] B. Schmeiser and L. Devroye, “Non-uniform random variate generation,” *Journal of the American Statistical Association*, vol. 83, p. 906, September 1988.

[70] “NS2 mannual.” Downloadable at http://nsnam.isi.edu/nsnam/index.php/Main_Page.

[71] “IT++ documentation.” Downloadable at http://itpp.sourceforge.net/current/.

[72] “TCL documentation.” Downloadable at http://www.tcl.tk/.

[73] R. Hamming, “Error dectecing and error correcting codes,” *Bell System Technical Journal*, vol. 29, pp. 147–160, April 1950.

[74] M. W. Williard, “Introduction to redundancy coding,” *IEEE Transactions on Vehicular Technology*, vol. 27, pp. 86–98, August 1978.

[75] ETSI, *Transport of MPEG 2 Transport Stream (TS) based DVB services over IP based networks*, October 2007. TS 102034 V1.3.1.

[76] “Third Generation Partnership (3GPP) Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs.” 3GPP TS 26.346 version 6.1.0 Release 6, June, 2005. Downloadable at http://www.3gpp.org/FTP/Specs/html-info/26346.htm.

[77] Y. Wang and Q. F. Zhu, “Error control and concealment for video communication: a review,” *Proceedings of the IEEE*, vol. 86, pp. 974–997, May 1998.

[78] X. F. Xu, M. van der Schaar, S. Krishnamachari, S. Choi, and Y. Wang, “Adaptive error control for fine-granular-scalability video coding over IEEE 802.11 wireless LANs,” in *Proceedings of the International Conference on the Multimedia and Expo, 2003.*, vol. 160, pp. I–669–672, November 2003.

[79] M. Luby, T. Gasiba, T. Stockhammer, and M. Watson, “Reliable multimedia download delivery in cellular broadcast networks,” *IEEE Transactions on Broadcasting*, vol. 53, pp. 235–246, March 2007.

[80] N. Degrande, K. Laevens, D. De Vleeschauwer, and R. Sharpe, “Increasing the user perceived quality for IPTV services,” *IEEE Communications Magazine*, vol. 46, pp. 94–100, February 2008.

[81] D. Gómez-Barquero, D. Gozálvez, and N. Cardona, “Application layer FEC for mobile TV delivery in IP datacast over DVB-H systems,” *IEEE Transactions on Broadcasting*, vol. 55, pp. 396–406, June 2009.

[82] D. Gómez-Barquero, P. Unger, T. Kürner, and N. Cardona, "Coverage Estimation for Multiburst FEC Mobile TV Services in DVB-H Systems," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 3491–3500, September 2010.

[83] H. Seferoglu, A. Markopoulou, U. C. Kozat, M. R. Civanlar, and J. Kempf, "Dynamic FEC algorithms for TFRC flows," *IEEE Transactions on Multimedia*, vol. 12, pp. 869–885, December 2010.

[84] A. Alexiou, C. Bouras, and A. Papazois, "A study of forward error correction for mobile multicast," *International Journal of Communication Systems*, vol. 24, pp. 607–627, May 2011.

[85] IEEE Computer Society, *Part 11: wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. New York, USA: IEEE Press, 2007.

[86] A. S. Tanenbaum, *Computer networks (fourth edition)*. New Jersey, USA: Prentice Hall PTR, 2002.

[87] D. Kliazovich and F. Granelli, "Dawl: a delayed-ack scheme for mac-level performance enhancement of wireless lans," *Mobile Networks and Applications*, pp. 607–615, October 2005.

[88] A. Basalamah and T. Sato, "Adaptive fec reliable multicast mac protocol for wlan," in *Proceedings of the IEEE Vehicular Technology Conference (VTC 2007-Fall)*, p. 244.

[89] A. Basalamah and T. Sato, "A comparison of packet-level and byte-level reliable fec multicast protocols for wlans," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2007)*, pp. 4702–4707, November 2007.

[90] E. Park, S. Han, H. Kim, K. Son, and J. Liu, "Efficient multicast video streaming for IPTV service over WLAN using CC-FEC," in *Proceedings of the International Conference on Internet Computing in Science and Engineering (ICICSE 2008)*, pp. 215 –222, January 2008.

[91] A. Moid and A. O. Fapojuwo, "Heuristics for jointly optimizing FEC and ARQ for video streaming over IEEE802.11 WLAN," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2008)*, pp. 2141 –2146, 31 2008-april 3 2008.

[92] M. Luby, T. Stockhammer, and M. Watson, "IPTV systems, standards and architectures: part II - application layer FEC In IPTV services," *IEEE Communications Magazine*, vol. 46, pp. 94 –101, May 2008.

[93] U. Demir and O. Aktas, "Raptor versus Reed Solomon forward error correction codes," in *Proceedings of the 2006 International Symposium on Computer Networks*, pp. 264 –269, July 2006.

[94] "RFC2988: computing TCP's retransmission timer." Network Working Group, IETF, November, 2000. Downloadable at http://www.ietf.org/rfc/rfc2988.txt.

[95] "RFC5052: Forward Error Correction (FEC) building block." Network Working Group, IETF, August, 2007. Downloadable at http://ietfreport.isoc.org/idref/rfc5052/index.html.

[96] G. Bianchi, "IEEE 802.11-saturation throughput analysis," *IEEE Communications Letters*, vol. 2, pp. 318–320, December 1998.

[97] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 535–547, March 2000.

[98] Proakis, J. G. , *Digital communication (fourth edition).* Bei Jing, CHINA: Publishing House of Electronics Industry, 2001.

[99] Nasruminallah and L. Hanzo, "EXIT-chart optimized short block codes for iterative joint source and channel decoding in H.264 video telephony," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 4306–4315, October 2009.

[100] D. Divsalar and F. Pollara, "Multiple turbo codes," in *Conference Record of the IEEE Military Communications Conference (MILCOM 1995)*, vol. 1, pp. 279–285, November 1995.

[101] D. Divsalar and F. Pollara, "Multiple turbo codes for deep-space communications," tech. rep., Jet Propulsion Lab., Pasadena, CA, 1995.

[102] E. Boutillon and D. Gnaedig, "Maximum spread of D dimensional multiple turbo codes," *IEEE Transactions on Communications*, vol. 53, pp. 1237–1242, August 2005.

[103] A. Huebner, K. S. Zigangirov, and D. J. Costello, "A new cycle-based joint permutor design for multiple turbo codes," *IEEE Transactions on Communications*, vol. 54, pp. 961–965, June 2006.

[104] S. X. Ng, Y. H. Li, and L. Hanzo, "Distributed turbo trellis coded modulation for cooperative communications," in *Proceedings of the IEEE International Conference on Communications (ICC 2009)*, pp. 1–5, June 2009.

[105] C. Berrou, C. Douillard, and M. Jézéquel, "Multiple parallel concatenation of circular recursive systematic convolutional (CRSC) codes," *Annals of Telecommunications*, vol. 54, no. 3, pp. 166–172, 1999.

[106] S. Huettinger and J. Huber, "Design of multiple-turbo-codes with transfer characteristics of component codes," in *Proceedings of the Conference on Information Sciences and Systems (CISS 2002)*, 2002.

[107] C. Berrou, "The ten-year-old turbo codes are entering into service," *IEEE Communications Magazine*, vol. 41, pp. 110–116, August 2003.

[108] B. Zhao and M. C. Valenti, "Distributed turbo coded diversity for relay channel," *IET Electronics letters*, vol. 39, pp. 3–4, May 2003.

[109] D. Gnaedig, E. Boutillon, and M. Jézéquel, “Design of three-dimensional multiple slice turbo codes,” *EURASIP Journal on Advances in Signal Processing*, vol. 2005, no. 6, pp. 808–819, 2005.

[110] F. Brannstrom, L. K. Rasmussen, and A. J. Grant, “Convergence analysis and optimal scheduling for multiple concatenated codes,” *IEEE Transactions on Information Theory*, vol. 51, pp. 3354–3364, September 2005.

[111] C. Hausl and J. Hagenauer, “Iterative network and channel decoding for the two-way relay channel,” *Proceedings of the IEEE International Conference on Communications (ICC 2006)*, pp. 1568–1573, June 2006.

[112] C. He, M. Lentmaier, D. J. Costello, and K. S. Zigangirov, “Joint permutor analysis and design for multiple turbo codes,” *IEEE Transactions on Information Theory*, vol. 52, pp. 4068–4083, September 2006.

[113] A. Ashikhmin, G. Kramer, and S. ten Brink, “Extrinsic information transfer functions: model and erasure channel properties,” *IEEE Transactions on Information Theory*, vol. 50, pp. 2657–2673, November 2004.

[114] D. Divsalar, S. Dolinar, and F. Pollara, “Serial concatenated trellis coded modulation with rate-1 inner code,” in *Proceeding of the IEEE Global Telecommunications Conference (GLOBECOM 2000)*, vol. 2, pp. 777–782, November 2000.

[115] S. Huettinger and J. Huber, “Analysis and design of power-efficient coding schemes with parallel concatenated convolutional codes,” *IEEE Transactions on Communications*, vol. 54, pp. 1251–1258, July 2006.

[116] L. Hanzo, O. Alamri, M. El-Hajjar, and N. Wu, *Near-capacity multi-functional MIMO systems.* John Wiley & Sons, 2009.

[117] J. Huber, T. Hehn, I. Land, and P. A. Hoeher, “Mutual information profile of a BISMC with applications,” *Preceedings of the Conference on Information Science and Systems (CISS 2005)*, pp. 95–102, March 2005.

[118] I. Land and J. Huber, *Information combining.* Now Publishers Inc., 2006.

[119] N. Dutsch, “Code optimization for lossless turbo source coding,” in *Proceedings of IST mobile & wireless communications summit*, (Dresden), 2005.

[120] I. Land, L. Rasmussen, and A. Grant, “Shaping EXIT functions of turbo codes,” *2008 Australian Communications Theory Workshop*, pp. 119–124, January 2008.

[121] A. Graell i Amat and E. Rosnes, “Good concatenated code ensembles for the binary erasure channel,” *IEEE Journal on Selected Areas in Communications*, vol. 27, pp. 928–943, August 2009.

[122] S. ten Brink, G. Kramer, and A. Ashikhmin, “Design of low-density parity-check codes for modulation and detection,” *IEEE Transactions on Communications*, vol. 52, pp. 670 – 678, April 2004.

[123] S. Lin and P. Yu, “A hybrid ARQ scheme with parity retransmission for error control of satellite channels,” *IEEE Transactions on Communications*, vol. 30, pp. 1701–1719, July 1982.

[124] K. R. Narayanan and G. L. Stuber, “A novel ARQ technique using the turbo coding principle,” *IEEE Communications Letters*, vol. 1, pp. 49–51, March 1997.

[125] R. D. Souza, M. E. Pellenz, and T. Rodrigues, “Hybrid ARQ scheme based on recursive convolutional codes and turbo decoding,” *IEEE Transactions on Communications*, vol. 57, pp. 315–318, February 2009.

[126] E. Y. Rocher and R. L. Pickholtz, “An analysis of the effectiveness of hybrid transmission schemes,” *IBM Journal of Research and Development*, vol. 14, pp. 426 –433, July 1970.

[127] H. O. Burton and D. D. Sullivan, “Errors and error control,” *Proceedings of the IEEE*, vol. 60, pp. 1293–1301, November 1972.

[128] A. Sastry, “Performance of hybrid error control schemes of satellite channels,” *IEEE Transactions on Communications*, vol. 23, pp. 689–694, July 1975.

[129] A. Sastry and L. Kanal, “Hybrid error control using retransmission and generalized burst-trapping codes,” *IEEE Transactions on Communications*, vol. 24, pp. 385–393, April 1976.

[130] C. Fujiwara, M. Kasahara, K. Yamashita, and T. Namekawa, “Evaluations of error control techniques in both independent-error and dependent-error channels,” *IEEE Transactions on Communications*, vol. 26, pp. 785 – 794, June 1978.

[131] A. Drukarev and J. D. J. Costello, “A comparison of block and convolutional codes in ARQ error control schemes,” *IEEE Transactions on Communications*, vol. 30, pp. 2449–2455, November 1982.

[132] S. Kallel, “Complementary punctured convolutional (CPC) codes and their applications,” *IEEE Transactions on Communications*, vol. 43, pp. 2005–2009, June 1995.

[133] Q. C. Chen and P. Z. Fan, “On the performance of type-III hybrid ARQ with RCPC codes,” in *Proceedings of 14th IEEE on Personal, Indoor and Mobile Radio Communications (PIMRC 2003)*, vol. 2, pp. 1297–1301, September 2003.

[134] Y. F. Wang, L. Zhang, and D. C. Yang, “Performance analysis of type III HARQ with turbo codes,” in *Proceedings of the IEEE Vehicular Technology Conference (VTC 2003-Spring)*, vol. 4, pp. 2740–2744, IEEE, April 2003.

[135] C. X. Wang, H. G. Ryu, H. H. Chen, and Y. He, “Hybrid ARQ with rate adaptation in multiband OFDM UWB systems,” *Proceedings of the IEEE International Conference on Communications (ICC 2009)*, pp. 1–5, June 2009.

[136] D. Chase, “A combined coding and modulation approach for communication over dispersive channels,” *IEEE Transactions on Communications*, vol. 21, pp. 159–174, March 1973.

[137] D. Mandelbaum, “An adaptive-feedback coding scheme using incremental redundancy (corresp.),” *IEEE Transactions on Information Theory*, vol. 20, pp. 388–389, May 1974.

[138] J. N. Laneman, D. N. C. Tse, and G. W. Wornell, “Cooperative diversity in wireless networks: Efficient protocols and outage behavior,” *IEEE Transactions on Information Theory*, vol. 50, pp. 3062–3080, Dec. 2004.

[139] R. H. Liu, P. Spasojevic, and E. Soljanin, “Incremental redundancy cooperative coding for wireless networks: Cooperative diversity, coding, and transmission energy gains,” *IEEE Transactions on Information Theory*, vol. 54, pp. 1207–1224, March 2008.

[140] E. Uhlemann, T. M. Aulin, L. K. Rasmussen, and P. A. Wiberg, “Packet combining and doping in concatenated hybrid ARQ schemes using iterative decoding,” in *Proceedings of the IEEE Wireless Communications and Networking (WCNC 2003)*, vol. 2, pp. 849–854, March 2003.

[141] I. D. Holland, H. J. Zepernick, and M. Caldera, “Soft combining for hybrid ARQ,” *IET Electronics Letters*, vol. 41, pp. 1230–1231, October 2005.

[142] 3GPP, “High speed downlink packet access: Physical layer aspects,” tech. rep., TR 25.858 V5.0.0, March 2002.

[143] M. Moher, “Decoding via cross-entropy minimization,” in *Technical Program Conference Record of the IEEE Global Telecommunications Conference, including a Communications Theory Mini-Conference. (GLOBECOM 1993)*, vol. 2, pp. 809–813, November 1993.

[144] J. Hagenauer, E. Offer, and L. Papke, “Iterative decoding of binary block and convolutional codes,” *IEEE Transactions on Information Theory*, vol. 42, pp. 429 –445, March 1996.

[145] R. Y. Shao, S. Lin, and M. P. C. Fossorier, “Two simple stopping criteria for turbo decoding,” *IEEE Transactions on Communications*, vol. 47, pp. 1117 –1120, August 1999.

[146] F. Q. Zhai and I. J. Fair, “Techniques for early stopping and error detection in turbo decoding,” *IEEE Transactions on Communications*, vol. 51, pp. 1617 – 1623, October 2003.

[147] F. M. Li and A. Y. Wu, “On the new stopping criteria of iterative turbo decoding by using decoding threshold,” *IEEE Transactions on Signal Processing*, vol. 55, pp. 5506 –5516, November 2007.

[148] L. Li, R. G. Maunder, B. Al-Hashimi, and L. Hanzo, “A low-complexity energy-efficient turbo decoder architecture (submitted),” *IEEE Transactions on Circuits and Systems II*, December 2010.

[149] “3rd Generation Partnership Project; Technical specification group radio access network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 10).” 3GPP TS 36.212, December, 2010. Downloadable at http://www.3gpp.org/FTP/Specs/html-info/26346.htm.

[150] A. Sendonaris, E. Erkip, and B. Aazhang, “User cooperation diversity. Part I. System description,” *IEEE Transactions on Communications*, vol. 51, pp. 1927–1938, November 2003.

[151] A. Sendonaris, E. Erkip, and B. Aazhang, “User cooperation diversity. Part II. Implementation aspects and performance analysis,” *IEEE Transactions on Communications*, vol. 51, pp. 1939–1948, November 2003.

[152] S. Yang and J. C. Belfiore, “Towards the optimal amplify-and-forward cooperative diversity scheme,” *IEEE Transactions on Information Theory*, vol. 53, pp. 3114 –3126, September 2007.

[153] K. R. Kumar and G. Caire, “Coding and decoding for the dynamic decode and forward relay protocol,” *IEEE Transactions on Information Theory*, vol. 55, pp. 3186–3205, July 2009.

[154] S. Simoens, O. Muoz-Medina, J. Vidal, and A. Del Coso, “Compress-and-Forward cooperative MIMO relaying with full channel state information,” *IEEE Transactions on Signal Processing*, vol. 58, pp. 781 –791, February 2010.

[155] J. Castura and Y. Y. Mao, “Rateless coding and relay networks,” *IEEE Signal Processing Magazine*, vol. 24, pp. 27–35, September 2007.

[156] J. Castura and Y. Y. Mao, “Rateless coding for wireless relay channels,” *IEEE Transactions on Wireless Communications*, vol. 6, pp. 1638–1642, May 2007.

[157] A. F. Molisch, N. B. Mehta, J. S. Yedidia, and J. Y. Zhang, “WLC41-6: cooperative relay networks using Fountain codes,” in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 2006)*, pp. 1 –6, November 2006.

[158] A. F. Molisch, N. B. Mehta, J. S. Yedidia, and J. Zhang, “Performance of Fountain codes in collaborative relay networks,” *IEEE Transactions on Wireless Communications*, vol. 6, pp. 4108–4119, November 2007.

[159] S. Puducheri, J. Kliewer, and T. E. Fuja, “The design and performance of distributed LT codes,” *IEEE Transactions on Information Theory*, vol. 53, pp. 3740–3754, October 2007.

[160] X. Liu and T. J. Lim, “Fountain codes over fading relay channels,” *IEEE Transactions on Wireless Communications*, vol. 8, pp. 3278–3287, June 2009.

[161] X. J. Wang, W. Chen, and Z. G. Cao, "Rateless coded chain cooperation in linear multi-hop wireless networks," in *Proceedings of the IEEE International Conference on Communications (ICC 2010)*, pp. 1–5, IEEE, May 2010.

[162] N. B. Mehta, V. Sharma, and G. Bansal, "Performance Analysis of a Cooperative System with Rateless Codes and Buffered Relays," *IEEE transactions on wireless communications*, vol. 10, pp. 1069–1081, April 2011.

[163] A. Ravanshid, L. Lampe, and J. B. Huber, "Dynamic decode-and-forward relaying using raptor codes," *IEEE Transactions on Wireless Communications*, vol. 10, pp. 1569–1581, May 2011.

[164] M. Uppal, G. S. Yue, X. D. Wang, and Z. X. Xiong, "A rateless coded protocol for half-duplex wireless relay channels," *IEEE Transactions on Signal Processing*, vol. 59, pp. 209–222, January 2011.

[165] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in *Proceedings of the International Symposium on Information Theory (ISIT 2004)*, p. 37, June 2004.

[166] J. Castura and Y. Mao, "Rateless coding over fading channels," *IEEE Communications Letters*, vol. 10, pp. 46–48, January 2006.

Author Index

Index