Message Passing Algorithm for Iterative Decoding of Channel Codes

Muhammad A. Imran and Mustafa K. Gurcan
Department of Electrical and Electronic Engineering,
Imperial College London, London SW7 2AZ

Abstract

Packet data broadcast systems require error free delivery of packet data without retransmission requests. Turbo codes can be used to achieve very low packet error rates. To achieve this low error rate with a very small gap from the system capacity, several design considerations need to be made. We need to use higher constraint-length codes (increasing complexity exponentially) with larger block sizes. Efficient interleavers need to be used and a larger number of iterations needs to be performed. To reduce the complexity and delay at the decoder, smaller constraint-length codes with smaller block sizes can be used with random interleavers and the number of iterations can be fixed. This, however, results in larger gaps from the capacity. This paper proposes a simple yet powerful two-tier channel coding scheme that has the potential to reduce the gap from capacity (from 5 dB to 4.1 dB, for target packet error $1 \times 10^{-4}$), while using smaller sized packets and smaller constraint length codes with random interleavers. The use of multiple packet detectors identified with the proposed scheme, reduces the packet error floor effect and even very low packet error rates like $1 \times 10^{-6}$ can be targeted maintaining a modest gap of 4 dB where other schemes exhibit significantly large gap values due to packet error rate floors. The improvement in the gap is nearly 2-3 dB in this region.

List of Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>variable index</td>
</tr>
<tr>
<td>$A$</td>
<td>maximum value of index</td>
</tr>
<tr>
<td>$\mathbf{a}$</td>
<td>column vector</td>
</tr>
<tr>
<td>$A$</td>
<td>matrix</td>
</tr>
<tr>
<td>$A(\cdot)$</td>
<td>$a$ posteriori log likelihood ratio for sample in argument.</td>
</tr>
<tr>
<td>$\lambda(\cdot)$</td>
<td>$a$ priori log likelihood ratio for sample in argument.</td>
</tr>
<tr>
<td>$a[i]$</td>
<td>sample at time index $i$</td>
</tr>
<tr>
<td>$\mathcal{A}$</td>
<td>set</td>
</tr>
<tr>
<td>$p_{pe}$</td>
<td>probability of packet error</td>
</tr>
</tbody>
</table>

I. INTRODUCTION

A packet broadcast system requiring very low error rates can achieve this on the downlink of the W-CDMA mobile network by using multiple receivers and cooperative broadcasting [1] to reduce the effect of interference. Gap – as defined by (1) – is an important measure to judge system performance for cooperative broadcasting systems. The systems operating with a smaller gap have higher sum capacity. As multiple receivers are required, the decoding complexity needs to be kept low in addition to the requirement of keeping the gap as small as feasible, when Turbo codes are used.

The complexity of decoding a Turbo code depends on the constraint length of the constituent encoders used and the number of iterations performed. The length of the block size and the design of the interleaver also
plays a role in achieving a good performance. For a low complexity code, a constraint length of 3 and a fixed number of decoder iterations (e.g. 6) can be used. We can also use smaller block sizes and random interleavers to improve the delay involved in decoding.

All these measures to keep decoding arrangements simple, results in an increase in gap values when very low frame error rate targets (such as a packet error rate of $1 \times 10^{-4}$) are considered. The low complexity Turbo coder depicted in Figure 2 with the corresponding packet error rate performance shown in Figure III-C (also reported in [2]) has a gap of 5 dB calculated using (1).

We present a coding scheme – referred to as Packet Parity Coding (PPC) scheme – and the corresponding iterative message passing decoding algorithm that can reduce the gap from 5 to 4.1 dB without increasing the constraint length and the block size. We also use the multiple packet detectors inherent in the code structure to improve the packet error rate approximately by seven orders of magnitude (from $10^{-3}$ to $10^{-10}$ at 0 dB) and to reduce the gap value to 3.9 dB for the target packet error rate of $1 \times 10^{-4}$.

We consider the system composed of three main blocks as the transmitter, the channel and the receiver as shown in Figure 1. The discrete information from the data source is in binary form. The data stream is segmented into blocks of equal size. These blocks of data are called the source packets. The proposed Packet Parity Coding (PPC) scheme encodes these packets to produce a larger number of encoded packets, having the same packet size. These are called the parity packets.

Each parity packet is appended with a Cyclic Redundancy Check (CRC) to detect the success of packet decoding at the receiver end. The resulting packets are encoded using Turbo code. This is a concatenated channel coding scheme with PPC acting as the inner code providing the encoded data to the outer code i.e. Turbo code. The encoded binary data, provided by the outer code, is converted into a binary antipodal stream of +1 and −1 for efficient transmission over the channel which adds the additive white Gaussian noise.

The receiver passes the noise corrupted data to the channel decoder block. The ‘outer code’ (Turbo code in
this case) is decoded first, feeding the decoded data to the decoder of ‘inner code’ (PPC code) to improve the decoding process.

The system performance can be characterised by performance measures, namely bit error rate (BER) and packet error rate (PER). For packet data transmission, PER is the more appropriate criterion for the transmission performance rather than the BER. The packet error of the Turbo code also depends on the constraint length of the constituent convolutional codes used and the number of message passing iterations between these constituent codes.

We assume a low complexity encoder, in octal form as shown in Figure 2, with constituent convolutional encoders of a constraint length of 3 and the generator matrix $[7, 5]$. We assume random interleaver separating two encoders. The packet error rate for rate half Turbo coding is determined using simulations. The proposed channel encoding and corresponding message passing decoding is performed using the same low complexity codes to determine the PER using an analysis developed in the paper.

We compare the different schemes using the gap measure presented by Cioffi in [3]. For code rate $r_c$ that achieves the desired error target at Signal to Noise Ratio (SNR) given as $\Gamma$, the gap $G$ from the system capacity is given by

$$G = \frac{\Gamma}{2^{2r_c}} - 1$$

Targeting a packet error rate of $1 \times 10^{-4}$ we find the gap from capacity using (1) for Turbo coding and the proposed message passing between three related packets. We show that the gap is reduced by around 1 dB. We also show that the packet error rate can be further reduced by approximately seven orders of magnitude by exploiting the structure of an outer code without increasing the gap. We introduce the PPC model in the next section.

The rest of the paper is organised as follows. We present the model for the proposed channel coding scheme and identify the need for the iterative decoding. The Tanner Graph is introduced as a tool to describe the structure and iterative decoding process. The iterative decoding process is then explained in detail and corresponding results are produced. A scheme to further reduce the packet error rate without increasing the gap is identified and the resulting packet error rate is derived. Results are presented and the conclusions are drawn. We start with the transmission system model in the next section.

II. TRANSMISSION SYSTEM MODEL

A. Packet Parity Coding (PPC)

A particular PPC encoding scheme encodes $V$ source packets into $W$ parity packets. Each parity packet is generated using $d$ different source packets from $V$. If $d$ is one, the parity packet is a copy of one of the source packets. If $d > 1$, the parity packet is a result of bit-wise XOR operation – addition over Galois Field of order 2, GF(2) – on $d$ source packets.

The source data in $V$ source packets, each of length $\bar{M}_p$, is organised in a source data matrix $\mathbf{V}_{\bar{M}_p \times V}$ with source packets as column vectors. The PPC code can be characterised using an encoding matrix $\mathbf{W}_{V \times W}$.

\[1\text{The product in the subscript of the matrix indicates the matrix size as rows} \times \text{columns.}\]
This matrix consists of \( W \) column vectors – represented as \( w_w \) – each of length \( V \). Entries in the matrix are from \( \{0,1\} \). The indices of ones in \( w_w \) indicate which columns of matrix \( V \) (source packets) should be used to generate the encoded packet indexed by \( w \). Hence the full matrix \( W \) completely defines the PPC encoding scheme. The encoded packets form the \( W \) columns of the matrix resulting from the matrix product \( [V \times W]_{M_p \times W} \). Note that the size of each encoded packet (the length of each column of the resulting matrix) is the same as the source packets.

The model in the above paragraph can be explained more easily using a simple PPC code example. Assume we label three source packets \( (V=3) \) as A, B and C and name our scheme as ABC coding scheme. Let each packet contain 250 binary bits \( (\bar{M}_p = 250) \). The source data matrix \( V \) would be a matrix of binary data with dimensions \( 250 \times 3 \). If we need to produce the parity packets\(^2 \) – A, B, C, A+B, A+C, B+C, A+B+C – the corresponding encoding matrix \( W \) is given by

\[
W = \begin{bmatrix}
1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 1
\end{bmatrix}.
\]  

(2)

The first three columns represent the parity packets generated by simply copying the source packets. The next three columns corresponds to packets formed by doing bitwise XOR on the ‘pairs’ of the source packets. For example the fourth column correspond to a packet formed by XOR of the first and second source packet. The

\(^2\)Symbol + denotes the bitwise modulo two addition or bitwise XOR operation.
last column corresponds to a parity packet formed by XOR of all three source packets.

We define a PPC coding scheme complete code if all possible unique parities are transmitted for the given number of source packets, \( V \). The maximum number of unique parities is given by \(^3\)

\[
W_{\text{max}} = \sum_{i=1}^{V} \binom{V}{i},
\]

as we add up all possible ways in which any collection of source packets can be mixed to form a unique parity packet. The ABC coding scheme introduced in this section is an example of a complete PPC code as \( V = 3 \) and \( W = W_{\text{max}} = 7 \). If \( W < W_{\text{max}} \), we term the coding scheme as punctured PPC code. Note that the effective code rate of the PPC code would be given by \( \frac{V}{W} \). For the given example of ABC code, it is \( \frac{3}{7} \).

We have \( W \) encoded packets, each with index \( w \), with the source data frame \(^4\) \( \hat{d}_w = \{d_w[0], \cdots, d_w[M_p - 1]\} \). The index \( i \) is used to represent the bit position, \( d_w[i] \). Note that each of these frames is \( w^{th} \) column of the matrix \( [V \times W]_{M_p \times W} \).

We encode each of these blocks using Turbo coding. We assume a Parallel Concatenated Convolutional (PCC) Turbo encoder [4], [5] that consists of two constituent Recursive Systematic Convolutional encoders linked with a random interleaver as shown in the block diagram in Figure 2. All constituent encoders are assumed to start at zero state and the first encoder is assumed to terminate at zero state.

Let \( \hat{x}_w[i] = \{x_w^0[i], x_w^1[i], x_w^2[i]\} \) denote the concatenation of systematic bit \( x_w^0[i] \) and output parity bits for encoder 1 and 2 respectively for each input bit \( d_w[i] \). To obtain a desired turbo code rate we puncture alternate parity bits from the encoded data \( \{\hat{x}_w[i]\}_{i=0}^{M_p - 1} \) and map the bits to the output frame of punctured data denoted by

\[
\hat{b}_w = \{b_w[0], \cdots, b_w[M_p - 1]\},
\]

where \( M_p \) is the size of the turbo encoded (and punctured) block. For \( M_p = 250 \) we obtain blocks of size \( M_p = 500 \) after rate half coding. Note that the last \( \nu \) bits of \( \hat{a}_w \) are the termination bits to terminate the first constituent decoder in the zero state, where \( \nu \) is the number of shift registers in the first constituent encoder [4].

The frames corresponding to all parity packets are concatenated to obtain a frame of length \( M = W \times M_p \) given by \( \hat{b}_k = \{b_1, \cdots, b_w, \cdots b_W\} \). The encoded data is transmitted over the channel and the corresponding received frames of estimates at the receiver are represented by \( \hat{z} = \{\hat{z}_1, \cdots, \hat{z}_w, \cdots \hat{z}_W\} \) where \( \hat{z}_w = \{z_w[0], \cdots, z_w[i], \cdots, z_w[M_p - 1]\} \). As each of these estimates is a noisy copy of the transmitted signal the received signal can be modelled as

\[
z_w[i] = ab_w[i] + \eta[i],
\]

where \( a \) is the received amplitude of the transmitted symbols and \( \eta[i] \sim \mathcal{N}(0, \nu^2) \) is a noise sample which is Gaussian distributed with mean 0 and variance \( \nu^2 \). We can calculate the Log Likelihood Ratio (LLR) of the

\(^3\)Notation \( \binom{n}{r} \) represents the ways in which \( r \) objects can be combined from \( n \) objects, \( \binom{n}{r} = \frac{n!}{r!(n-r)!} \).

\(^4\)Note the notation used for the data frames is the small case Franktur letters \( d, \hat{x} \) and \( \hat{b} \). The notation \( \{\cdot\} \) represent a concatenation of vectors.
bits of each parity packet \( w \) considering the Gaussian nature of the noise and the definition of LLR. This gives,

\[
\lambda_d(b_w[i]) = \frac{2a z_w[i]}{\nu^2}.
\]

These LLRs are used by the two (outer and inner) channel codes – the Turbo and the PPC code – to iteratively decode the information symbols by exchanging the LLR information. Before discussing the iterative decoding procedure we need to introduce the concept of \textit{a priori}, \textit{a posteriori} and \textit{extrinsic} LLR.

We use MAP algorithm and the structure of the constituent codes along with the relation between the packets connected to same check node (defined in section II-C) to determine the probability of either +1 or a −1 was transmitted, given the related frames of received samples. Let \( F \) denote a set of packets connected to same check node of the tanner graph introduced in section II-C. The concatenation of related frames of received samples, i.e. \( \{ \hat{z}_f | f \in F \} \) is denoted by \( \hat{z}_F \) and for a given parity packet \( w \), the corresponding detector and \( F \) can be identified from Table I. The \textit{a posteriori} LLR is calculated for each bit \( b_w[i] \) and is given as

\[
\Lambda(b_w[i]) \triangleq \log \frac{\Pr (b_w[i] = +1 | \hat{z}_F)}{\Pr (b_w[i] = -1 | \hat{z}_F)}.
\]

Using Baye’s rule, this can be rewritten as

\[
\Lambda(b_w[i]) \triangleq \log \frac{\Pr (\hat{z}_F | b_w[i] = +1)}{\Pr (\hat{z}_F | b_w[i] = -1)} + \log \frac{\Pr (b_w[i] = +1)}{\Pr (b_w[i] = -1)}.
\]

where the first term in (8) indicates the \textit{extrinsic} LLR provided by the decoding of the code as it purely depends on the received vector and does not get effected with the second term i.e. \textit{a priori} LLR provided by the external sources. The second term represents the \textit{a priori} LLR which is either zero (at the beginning of the iterative process) or obtained from the extrinsic information generated by the other related sources in some previous iteration. Note that the superscript \( p \) in the second term indicates the quantity obtained in the previous iteration of the iterative process. In the next section we discuss the motivation for performing the iterative decoding .

\section*{B. Need for Iterative Decoding}

To achieve the capacity promised by Shannon [6], the capacity approaching codes like Turbo (pioneered by Berrou et al. [7], [8]) and LDPC (first discovered by Gallager [9] and re-invented by MacKay [10], [11] and Luby [12]) widely use the iterative decoding schemes and provide low error rates due to their random nature [13]. In an iterative decoding scheme, the decoder decisions are iteratively improved by passing the information between different modules of the decoder.

The reliability of data transmitted in packets is sometimes increased by using coding schemes which operate on packets. One approach is an erasure resilient coding. Fountain coding schemes surveyed in [14] are one example. Luby has proposed efficient erasure correcting codes (LT codes) in [15] and various variants have emerged [16], [17], [18]. The proposed PPC coding scheme is similar to the LT coding scheme in the sense that we use parity packets formed by XOR combinations of source packets. They differ from LT codes in two aspects: The design is not for erasure correction and therefore does not involve random parity packet generation and in contrast to the so-called ‘rateless’ nature of LT codes they have a fixed predefined rate depending on the parity packets used.
Another approach to improve reliability of transmitted packet data is to transmit multiple instances of the coded data. This requires packet combining approaches at the receiver end with various variants like, diversity combining [19] and code combining [20], [21], [22], [23]. The proposed PPC scheme also uses the concept of packet combining while exchanging the information between related packets but it does not consider any retransmission request mechanism. Packet combining schemes have been investigated to work with iterative decoding methods as in the work by Stüber [24]. Some recent literature, by Jenkac et al. [25], [26], tackles the iterative decoding and packet combining approaches for erasure correction and proposes ‘Turbo Fountain’ and a corresponding iterative decoder. This decoder is similar in nature to the decoding scheme we use to pass messages between packets related to each other by the PPC encoding scheme.

For a PPC decoding scheme the performance improvement is achieved using two techniques.

1) The parity packets are iteratively decoded to improve the error correction performance.

2) If some of the parity packets fail to decode correctly, incomplete combinations of successfully decoded packets are capable of regenerating the unknown source packets using simple XOR operations.

We start with the first improvement method. To explain the iterative decoding arrangement we introduce the Tanner Graphs in the next section.

C. Use of Tanner Graph

Tanner considered how a generalised Low Density Parity Check (LDPC) code can be effectively represented by a bi-partite graph now called a Tanner Graph [27] in which there are two types of nodes connected by edges. These two types are termed as variable nodes and check nodes.

The Tanner Graph corresponding to proposed ABC coding scheme is presented in Figure 3. Variable nodes are represented with a circle and check nodes are represented by a square with a plus sign inside it. It can be seen in the diagram that the variable nodes are the parity packets in our case. The parity packets formed by single source packet, \( d = 1 \), are drawn below the row of check nodes and the other variable nodes for which \( d > 1 \) are drawn above the row of check nodes.

Gallager introduced LDPC codes and provided a near optimal decoding algorithm in his work [9] now called the message passing algorithm (MPA). The algorithm iteratively estimates the variable nodes in a Tanner graph by exchanging messages at the check nodes. A group of packets is connected to same check node if they satisfy the following relationship. Each of the packets in the group can be obtained as a result of the XOR operation of the rest of the packets in the group of related packets.

Consider a check node connecting packets \( A, B \) and \( A + B \) drawn in Figure 4. Taking the XOR of any two of the connected packets yields the third, such that the information about any two can be used in the detection of the third packet. In terms of the Tanner graph in the figure, the information about any two neighbours \(^5\) can be used to provide the information for the leftover third neighbour of the check node.

The possible message flows are shown in three diagrams in Figure 4. The first flow originates from the definition of \( A + B \) as this packet is the result of XOR of information bits in \( A, B \). The other two message

\(^5\)The variable nodes connected to the same check node are termed as check-node neighbours.
flow diagrams result from the property of XOR operation as the result for XOR of $B$ and $A + B$ is the third related packet $A$ and similarly the XOR of $A$ and $A + B$ results in packet $B$.

The message flow diagrams indicate how the available information about some variable nodes can help in guessing the value of some other variable nodes. We use these multiple message flows to iteratively improve the performance of our decoding scheme. For example, information available for $A$ and $B$ can provide some information for $A + B$ and then $A + B$ can use this information with information about $A$ (or $B$) to provide information about $B$ (or $A$). This process can be iterated to improve the decoding performance.

In our case each of these check nodes correspond to a proposed ‘multiple packet detector’. These check nodes (or multiple packet detectors) are identified for full ABC code example and are tabulated in Table I in accordance with Figure 3. In the figure, two different types of check nodes can be identified. Some connect three and others connect four packets. The objective of the decoding process is to decode the variable nodes drawn below the row of check nodes. Any check node (e.g. decoder 7) that does not connect to these packets
Table I: Iterative Multiple Packet Detectors

<table>
<thead>
<tr>
<th>Number</th>
<th>Connected Packets</th>
<th>$\mathcal{F}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$A$ $B$ $A+B$</td>
<td>${1, 2, 4}$</td>
</tr>
<tr>
<td>2</td>
<td>$A$ $C$ $A+C$</td>
<td>${1, 3, 5}$</td>
</tr>
<tr>
<td>3</td>
<td>$B$ $C$ $B+C$</td>
<td>${2, 3, 6}$</td>
</tr>
<tr>
<td>4</td>
<td>$A$ $B+C$ $A+B+C$</td>
<td>${1, 6, 7}$</td>
</tr>
<tr>
<td>5</td>
<td>$B$ $A+C$ $A+B+C$</td>
<td>${2, 5, 7}$</td>
</tr>
<tr>
<td>6</td>
<td>$C$ $A+B$ $A+B+C$</td>
<td>${3, 4, 7}$</td>
</tr>
<tr>
<td>7</td>
<td>$A+B$ $A+C$ $B+C$</td>
<td>${4, 5, 6}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Connected Packets</th>
<th>$\mathcal{F}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>$B$ $C$ $A+B$ $A+C$</td>
<td>${2, 3, 4, 5}$</td>
</tr>
<tr>
<td>9</td>
<td>$A$ $C$ $A+B$ $B+C$</td>
<td>${1, 3, 4, 6}$</td>
</tr>
<tr>
<td>10</td>
<td>$A$ $B$ $A+C$ $B+C$</td>
<td>${1, 2, 5, 6}$</td>
</tr>
<tr>
<td>11</td>
<td>$A$ $B$ $A+C$ $B+C$</td>
<td>${1, 2, 3, 7}$</td>
</tr>
</tbody>
</table>

**III. ITERATIVE CHANNEL DECODING**

Figure 5 shows the setup of channel decoder. Based on the factor graph, the parity packets connected to the same check node are selected for iterative decoding and are taken into a multiple packet detector.

Block diagram of a three packet multiple packet detector is shown in Figure 6. Implementation of each multiple packet detector contains multiple Turbo decoders called the component decoders. Each component decoder (mainly a Turbo decoder) takes multiple inputs. One input is the LLR from the channel and the other inputs are the extrinsic LLR obtained in last iteration, $\lambda_p^e(x_0^f[i])$, from the other packets connected to the same check node. These are appropriately combined – calculating $\sum \lambda_p^e(x_0^f[i])$ – and used as the a priori LLR by the

---

**Figure 5: Block diagram of the receiver for one channel**

can help improve the decoding of connected parity packets but does not yield any useful source packet.

These nodes and the possible message flows form the main building block for the iterative decoding of the outer and inner channel codes to improve the decoding performance. We first discuss how this iterative decoding is performed and then show how to analyse the PER for multiple packet detectors.
component decoder to decode the bits. The \textit{a posteriori} information for the message bits obtained from second decoder – $\Lambda_{c,2}(x^o_0[i])$ – is converted to \textit{extrinsic} LLR, $\lambda_{c}(x^o_0[i])$, to be passed on to the other Turbo decoders for the packets connected to same check node, for processing in the next iterations.

In addition to this outer iteration loop, each turbo decoder contains two MAP decoders that can iteratively exchange the LLR information for a number of iterations improving the Turbo decoding performance before delivering the output and the extrinsic information for the outer iteration between component decoders in a multiple packet detector. The process which takes the \textit{extrinsic} LLRs from multiple packets connected to a check node and provide the \textit{a priori} LLR for one of the packets connected to the same check node is referred to as the LLR processing.

\textbf{A. LLR processing}

We will use a special algebra, as in [28], for the log likelihood ratio values $\lambda(x)$: We use the symbol $\boxplus$ as the notation for the addition defined by (all sub-scripts and superscripts are dropped as the definition is general)

$$\lambda(x_1) \boxplus \lambda(x_2) \equiv \lambda(x_1 + x_2) \quad \text{(9)}$$
where the + sign on right hand side represents the XOR operation or modulo 2 addition of the binary value of the symbols. The additional rules are

\[ \lambda(x) \oplus \infty \triangleq \pm \lambda(x), \quad \lambda(x) \oplus 0 \triangleq 0 \]  

(10)

The notation for algebra on more than two LLR values is extended and calculated as [28]

\[ \sum_{f=1}^{F} \lambda(x_f) \triangleq \lambda \left( \sum_{f=1}^{F} x_f \right) = -2 \arctanh \left( \prod_{f=1}^{F} \tanh(\lambda(x)) \right), \]  

(11)

where the summation sign represents the modulo two sum of \( F \) binary symbols and the summation sign with ‘a box and plus sign in it’ represents the algebraic combination of the corresponding LLRs to obtain the LLR of the XOR of all these symbols.

For our case, we use this notation to process the LLR of all attached packets to a check node, leaving the one for which LLR estimate is required. Given that a check node has a set of connections, \( \mathcal{F} \), the \textit{a priori} LLR to be passed out to connection \( w, w \in \mathcal{F} \) is given by the following expression. The notation \( \mathcal{F}^{(w)} \) represents the set \( \mathcal{F} \) excluding \( w \).

\[ \lambda_c(x_0^0) = \sum_{f \in \mathcal{F}^{(w)}} \lambda(x_f) \]  

(12)

B. Turbo Decoder Description

Given the \textit{a priori} LLR obtained from the channel for each message and parity bit, the Turbo decoder provides an estimate of the \textit{a posteriori} LLR for these bits. Consider the block diagram in Figure 7. The Turbo decoder comprises two constituent convolutional decoders, which are assumed to have perfect knowledge of the generator polynomials used at the encoder end. Each of these decoders uses the modified BCJR\(^6\) (Log-MAP) algorithm for decoding, as described by Ryan in [4] and Wang and Poor in [30].

We identify two types of Turbo code bits: the systematic or message bits represented with a 0 in the superscript of the term inside the brackets, following the LLR symbol, and the Turbo code parity bit represented with a 1 (or 2) in the superscript of the corresponding term.

There are two map decoders, labelled 1 and 2. Each decoder is termed as an auxiliary decoder with respect to the other. For every message bit, the two MAP decoders generate \textit{a posteriori} LLRs based on the code structure for the convolution code corresponding to each decoder and the input provided to them as the \textit{a priori} LLR. We represent the \textit{a posteriori} LLR by \( \Lambda_{c,1}(x_0^0[i]) \) and \( \Lambda_{c,2}(x_0^0[i]) \) and the input \textit{a priori} LLR by \( \lambda_{c,1}(x_0^0[i]) \) and \( \lambda_{c,2}(x_0^0[i]) \). We obtain \textit{a priori} LLR from three main sources: channel, auxiliary decoder, other Turbo decoders in the multi-packet detectors.

In the first iteration, the only source of \textit{a priori} LLR is from the channel and all other sources of \textit{a priori} LLR are initialised to zero. In the subsequent iterations, the extrinsic information from the auxiliary MAP decoder is combined with the LLRs coming from two other sources: LLRs from the other Turbo decoders operating on the packets connected to same check node and the LLRs from the channel. Hence the \textit{a priori} LLR for

\(^6\)Originally proposed by Bahl, Cocke, Jelinek and Raviv (BCJR) in [29].
The extrinsic LLR passed from MAP decoder 2 to 1 (similar for 1 to 2) is calculated as
\[ \lambda_{c,21}(x_w[i]) = \Lambda_{c,2}(x_w[i]) - \lambda_{c,2}(x_w[i]). \] (14)

The LLRs for the Turbo code parity bits are obtained directly from the signal detector, \( \lambda_{c,1}(x_w[i]) = \lambda_{d}(x_w[i]) \) and \( \lambda_{c,2}(x_w[i]) = \lambda_{d}(x_w[i]) \). If the corresponding Turbo code parity bit is not transmitted due to puncturing, the LLR is set to zero as no information is available for this bit from the channel.

After fixed number of iterations, the a posteriori LLR for the message bits generated by the second decoder, \( \Lambda_{c,2}(x_w[i]) \), is set as the a posteriori LLR for the Turbo decoder, \( \Lambda_c(x_w[i]) = \Lambda_{c,2}(x_w[i]) \). The LLR provided by this decoder to the other Turbo decoders as extrinsic LLR is calculated as
\[ \lambda_c(x_w[i]) = \Lambda_{c,2}(x_w[i]) - \left( \lambda_{d}(x_w[i]) + \sum_{f \in P(w)} \lambda_{p}(x_f[i]) \right), \] (15)
and can be used to generate a priori information for the other Turbo decoders for the packets connected to the same check node.

After a fixed number of these message passing iterations, the a posteriori LLR \( \Lambda_c(x_w[i]) \) for each message bit can be used for hard decisions followed by PPC (hard decision) decoding.
C. Results for Iterative Channel Decoding

The performance of the iterative channel decoding on a single three packet check node is determined using a simulation experiment. For turbo encoding we use two constituent Recursive Systematic Convolutional (RSC) encoders separated by a random interleaver (Fig. 2). Both encoders use the generator polynomial \([7, 5]_{8}\) (in octal form) with a constraint length of 3. We use puncturing to achieve a rate \(\frac{1}{2}\) Turbo code. The information block size is taken to be 750 bits corresponding to three PPC source packets \(A, B\) and \(C\) in the ABC code, each of the size 250 bits, i.e. \(\tilde{M}_p = 250\) bits and \(M_p = 500\). For a fair comparison, when Turbo coding is used without PPC, we use the message block size of \(\bar{M}_p = 750\) bits and \(M_p = 1500\) bits for all three source packets encoded as a single packet (Turbo performance is better with larger block sizes). For different SNR values at the input of the channel decoder, the PER performance of a three neighbour check node (a multi packet decoder connecting three packets, like decoder 1) is determined.

The results are shown in Figure 8. For a target packet error rate of \(1 \times 10^{-4}\), the target SNR is approximately 1.8 dB for rate \(\frac{1}{5}\) three packet check node (\(A\) and \(B\) with third redundant packet \(A + B\) and each encoded with rate \(\frac{1}{2}\) code, rate is \(\frac{2}{5} \times \frac{1}{2} = \frac{1}{4}\)). This corresponds to a gap of 4.1 dB. The corresponding gap for the Turbo code (rate \(\frac{1}{2}\), target SNR 5 dB) is 5dB. We can achieve a reduction of around 1 dB in this case. We next show that using the multiple packet detectors, and performing the so-called hard decoding on the structure of the ABC codes, we can bring down the packet error rate without increasing the gap.
IV. HARD DECODING OF PPC

The outer channel code (Turbo code in our case) attempts to correctly estimate the data bits in each packet, possibly using the proposed iterative message passing. For example, we can use iterative message passing on detectors 1 to 7, identified in Figure 3, to improve the success rate for each packet. A packet is declared successfully detected if all the source data bits in the packet are estimated correctly. This is verified using CRC at the receiver end. This packet detection process has a probability of failure given by $p_{pe}$ and termed as 'raw packet error rate' in further discussions.

We can use the multiple packet detectors (various check nodes in Tanner graph) to find unrecovered packets from the successful packets using simple XOR operations. This is termed as hard decoding of the PPC code. We analyse the PER (an upper bound) for such hard decoding of PPC encoded packets and show that this reduces the packet error rate by approximately seven orders of magnitude without reducing the gap further.

Using the probability of detecting a packet incorrectly (with or without a message passing algorithm) we aim to find the probability of failing to decode a source packet (out of a total of $V$ source packets). The error rate for the source packets, called PER, is the average number of failed source packets out of a total of $V$ source packets transmitted. This is given by

$$PER = \sum_{l=0}^{V-1} Pr(l) \frac{V - l}{V}.$$  \hspace{1cm} (16)

where $l$ is the number of source packets which the inner channel code detects successfully and $Pr(l)$ represents the probability of this successful detection.

As the number $l$ depends on the number of successfully detected parity packets – represented by $z$ – we can express $Pr(l)$ as

$$Pr(l) = \sum_{z=0}^{W} Pr(l | z) Pr(z)$$ \hspace{1cm} (17)

If the probability of failure of decoding a parity packet is represented by $p_{pe}$, the probability of successfully decoding $z$ parity packets out of a total of $W$ parity packets is given by

$$Pr(z) = \binom{W}{z} p_{pe}^{W-z} (1 - p_{pe})^z$$ \hspace{1cm} (18)

In the following we would show how to determine the PER using (16)-(18). For any PPC scheme the key to evaluate the PER is to find the conditional probability $Pr(l | z)$. For ABC code the number of successfully decoded source packets, $l$, varies between 0 and $V = 3$. This depends on the number of parity packets, $z$, successfully decoded by the outer code (Turbo code) out of a total $W = 7$ parity packets. The values of $Pr(l | z)$ for all possible values of $l$ and $z$ are tabulated in Table II for ABC code.

We need to define the term degree and the neighbour of a parity packet for explaining the conditional probability $Pr(l | z)$ calculations.

**Definition 1:** A PPC parity packet indexed by $w$, has a degree $d_w$ which is the count of ones in the corresponding vector $w_w$.

$$d_w = \sum w_w$$ \hspace{1cm} (19)

where $1$ represents a vector of all ones and is the same length as $w_w$, i.e. $V$. 


The neighbours of an encoded packet are the source packets used to generate the encoded packet.

As an example, A and B are the two neighbours of packet A + B. We note that degree 1 packets have a single neighbour.

We can now explain how we obtain the values in the table. As a general formulation the terms \( \binom{Q_1}{q_1} \binom{Q_2}{q_2} \binom{Q_3}{q_3} \) indicate the possible ways to have \( q_1 \) degree 1 packets out of a total of \( Q_1 \), \( q_2 \) degree 2 packets out of a total of \( Q_2 \) and \( q_3 \) degree 3 packets out of \( Q_3 \), where \( z = q_1 + q_2 + q_3 \). The case where \( z = 0 \) (corresponding to the first row of the table) is trivial as no source packet can be decoded when no parity packet is available.

When \( z = 1 \) (second row), the available parity packet can only recover a source packet \( l = 1 \) if the parity packet is of degree 1. In total, there are \( \binom{7}{1} = 7 \) different ways of choosing any packet from a set of 7 parity packets. As there are only 3 packets of degree 1, therefore there are \( \binom{3}{1} \binom{4}{0} = 3 \) different ways of selecting this single parity packet from degree one packets leading to recovery of a source packet \( l = 1 \). This explains the conditional probability in column 2. Similarly, there are \( \binom{4}{1} \binom{3}{0} = 4 \) ways in which the single parity packet can be selected from the packets with a degree greater than one leading to \( l = 0 \), explaining the entry in column 1. As \( l \leq z \) (source packets detected cannot be greater than the parity packets available) all other entries in this row are zero.

\[
\begin{array}{|c|c|c|c|c|}
\hline
z & \Pr(l = 0 | z) & \Pr(l = 1 | z) & \Pr(l = 2 | z) & \Pr(l = 3 | z) \\
\hline
0 & 1 & 0 & 0 & 0 \\
1 & \binom{3}{3} \binom{4}{0} / \binom{7}{1} = 4/7 & \binom{3}{3} \binom{4}{0} / \binom{7}{1} = 3/7 & 0 & 0 \\
2 & \binom{3}{3} \binom{4}{2} \binom{1}{0} / \binom{7}{2} = 3/21 & \binom{3}{3} \binom{4}{2} \binom{1}{0} / \binom{7}{2} + \binom{2}{2} \binom{3}{1} \binom{1}{0} / \binom{7}{2} = 9/21 & \binom{3}{3} \binom{4}{2} \binom{1}{0} / \binom{7}{2} + \binom{2}{2} \binom{3}{1} \binom{1}{0} / \binom{7}{2} = 9/21 & 0 \\
3 & \binom{3}{3} \binom{4}{3} \binom{0}{0} / \binom{7}{3} = 3/35 & \binom{3}{3} \binom{4}{3} \binom{0}{0} / \binom{7}{3} = 3/35 & \binom{3}{3} \binom{4}{3} \binom{0}{0} / \binom{7}{3} = 3/35 & 28/35 \\
z > 3 & 0 & 0 & 0 & \binom{3}{3} / \binom{7}{3} = 1 \\
\hline
\end{array}
\]
Considering $z = 2$ (third row), no source packet is detectable ($l = 0$) if both available packets have a degree 2. This has $\binom{3}{0} \binom{3}{1} \binom{1}{0} = 3$ different ways from total $\binom{3}{2} = 21$ different ways of selecting any two parity packets giving the entry in column 1 of the row corresponding to $z=2$.

However for $z = 2$, one source packet can be detected in three cases. In the first case, a pair of degree one and degree two parity packet is available where they do not have a common neighbour. Note that if they do have a common neighbour then another source packets can be recovered from such a pair. In the second case a source packet is recovered when a pair of a degree one and a degree three parity packet is available. In the third case the pairing of any degree two with the single degree three packet can also provide a source packet.

To find the probability of the first case in the last paragraph, we identify the probability of the event that a given pair of parity packets having a degree $x$ and $y$ have $c$ packets as common neighbours.

Consider Figure 9. For any given degree $x$ packet, we can select the other member of the pair, a degree $y$ packet from a set of $\binom{y}{2}$ possible degree $y$ packets. To ensure that the second event is independent of the first, $x \neq y$. The given degree $x$ packet has $x$ source packets marked as its neighbours. For the chosen packet to have $c$ common neighbours with the given packet, at least $c$ neighbours should be selected from the $x$ marked source packets giving $\binom{x}{c}$. The rest of the neighbours $y - c$ can be chosen from the remaining unmarked $V - x$ source. This can obviously be done in $\binom{x}{c} \binom{V-x}{y-c}$ different ways. Hence the probability is given by

$$
Pr(c|x, y) = \frac{\binom{x}{c} \binom{V-x}{y-c}}{\binom{V}{y}}
$$

(20)

where $c \leq \min(x, y)$, $x \neq y$ and $x, y \leq V$.

We use this general formulation in our specific case where we are interested in finding the probability of the event that a degree one packet has zero neighbours common with a degree 2 packet for ABC coding where $V = 3$. We obtain $Pr(c = 0|x = 1, y = 2) = 1/3$, explaining the factor $1/3$ in the third column of the case when $z = 2$ and a pair of degree one and two packets is available.

When $z = 2$, two source symbols can be detected (column 4), if both are degree 1 packets or the pair
of degree one and degree two packets has a neighbour in common. The second event has the probability
\[ \Pr(c = 1|x = 1, y = 2) = \frac{2}{3}, \]
calculated using (20).

As discussed before, when \( z = 2 \), probability of detecting more than two source packets \( (l > 2) \) is zero
explaining the remaining columns of this row.

When \( z = 3 \), no source packet is available only in the case when all three packets are the degree 2 packets.
Only 1 source packet is available if we have one packet of each degree 1,2 and 3 plus the degree 1 and degree 2
packets have zero common neighbours (hence \( c = 0 \) leading to factor \( \Pr(c = 0|x = 1, y = 2) = \frac{1}{3} \) calculated
from (20)).

Two source packets are available, \( l = 2 \), if two degree 1 packets and one degree 2 packets are available and
the degree 2 packet has both degree one packets as its neighbour. Note that we intend to find the probability
of having a specific mix of three parity packets. This is different from the case of finding the probability of
common neighbours in two parity packets as was done in (20). To find this new probability we proceed as follows.

If we represent the total number of degree 1 packets in our coding scheme by \( Q_1 \), the probability of selecting
a pair from \( Q_1 \), such that both are the neighbours of a given degree 2 packet, is given by
\[
\frac{\binom{2}{1} \left( Q_1 - \binom{1}{0} \right)}{\binom{2}{2}} = \frac{\binom{2}{2} \cdot 1}{\binom{3}{2}} = \frac{1}{3}
\] (21)
The probability of selecting both degree one packets from these specific ones and none from the rest forms
the numerator. The denominator represents the possibilities of selecting any two degree 1 packets from all \( Q_1 \)
possible candidates. For all other cases where \( z > 3 \), we can find all the source packets from the available
encoded packets. This can be verified by checking all possible combinations of different degree packets. We
should specially note the case when no degree 1 packet is available and all 4 ‘higher degree’ parity packets (3
degree 2 and 1 degree 1 parity packets) are available. Each degree 2 parity packet can decode a source packet
when used with the degree three packet. As an example, \( A + B \) can be used with \( A + B + C \) to generate source
packet \( C \). Hence although no source packet is directly available but all can be extracted using XOR operations
on the parity packets.

We can now evaluate the PER performance of the ABC coding scheme given the packet error performance
of the outer code, \( p_{pe} \). Finding \( \Pr(l|z) \) from Table II and \( \Pr(z) \) from (18) and substituting these in equation
(17) we get \( \Pr(l) \). Substituting this in (16) we find the average number of packets in error, i.e. PER. In the
next section we present the PER results for ABC codes.

A. Results for Using Multiple Packet Detectors

We run a simulation experiment. PPC encoded packets (using a full ABC coding scheme) are transmitted
over a channel. For a varying raw packet error rate (error rate before PPC decoding) on the channel we evaluate
the packet error rate after PPC decoding. The results are shown in Figure 10 that shows the packet error rate
for a given number of parity packets available.

Substituting (17) in (16) and rearranging the summation order we obtain an alternate equation for finding
Figure 10: Performance of full ABC coding scheme

Figure 11: Required SNR to meet the packet error rate target for different coding schemes
\[ \text{PER} = \frac{V - l}{V} \sum_{l=0}^{V-1} \sum_{z=0}^{W} \Pr(l|z) \Pr(z) \frac{V - l}{V} = \sum_{z=0}^{W} \sum_{l=0}^{V-1} \frac{V - l}{V} \Pr(l|z) \Pr(z) = \sum_{z=0}^{W} \text{PER}(z) \] (22)

The solid curves in the top plot of Figure 10 correspond to \( \text{PER}(z) \) for \( z = 0, 1, 2, \cdots, 7 \). The lowest visible curve corresponds to \( \text{PER}(z = 0) \) and the top visible curve corresponds to \( \text{PER}(z = 3) \). All other curves where \( z > 3 \) result in zero error rate which is not visible on the logarithmic scale.

Adding all these curves according to (22) yields the total \( \text{PER} \). These curves are separately plotted to emphasise that the dominant case is when \( z = 3 \) or “four out of seven packets are lost”. This event has a very low probability according to the binomial equation 18, explaining the low \( \text{PER} \) of the full ABC codes.

In Figure IV we show the \( \text{PER} \) corresponding to this scheme. The packet error rate of \( 1 \times 10^{-4} \) can be achieved at SNR of \(-0.7 \) dB using an overall code rate of \( \frac{3}{14} \). The calculated gap is 3.9 dB using (1) with \( \Gamma = 10^{-0.7/10}, r_c = 3/14 \). The gap is approximately the same as that achieved with the three packet check node scheme, however, it should be noted that the \( \text{PER} \) has been brought down to \( 10^{-10} \) from \( 10^{-3} \) at the SNR of 0 dB. This scheme results in minimising the packet error rate floors. Hence, smaller packet error targets can be achieved without increasing the gap value significantly as with other coding schemes which exhibit the error floors.

V. CONCLUSION

A two layered coding scheme is proposed that has groups of packets related to each other, i.e. connected to a check node. A message passing algorithm can be used to increase the decoding performance. Experimental results show that the iterative decoding provides a smaller gap (4.11 dB as compared to 5 dB for a low complexity Turbo code). The used example of the ABC code also demonstrate that after running the message passing algorithm, source packets that fail to decode correctly can still be recovered by exploiting the structure of multiple detectors or check nodes by simple XOR operations. Without reducing the gap further, we reduce the packet error rate by some orders of magnitude (\( 10^{-3} \) to \( 10^{-10} \) at 0 dB). This significantly reduces the flooring effect of packet error rate performance curves.

REFERENCES