Block Error Correction Codes and Convolution Codes

Mei Yen, Cheong

***Abstract*—In wireless communications, error control is an important feature for compensating transmission impairments such as interference and multipath fading which cause high bit error rates in the received data. Forward Error Correction (FEC) is one of the data link layer protocols for error control. This paper gives an overeview to error control coding employed in FEC, particularly block codes and convolutional codes. Finally, some consideration of code selection will be discussed.**

I. INTRODUCTION

The trend towards portable personal computers or workstations are rising quickly in the recent years. WLAN networks are becoming increasingly popular. Users of the network are demanding for ever higher quality of service (QoS) and multi-variety services (not only data, but also packet voice, video, etc.). Unlike wireline systems, wireless and mobile environment is more prone to transmission impairments due to multipath propagation and movement of the mobile stations.

Particularly, in WLAN systems which employ omnidirectional antennae in order to obtain good coverage, the transmitted signal power is not concentrated to the intended user. On top of that, the transmitted signal is more scattered and multipath effect becomes more pronouns. Interference is also an issue in WLAN environment due to its unlicensed frequency spectrum.

With the aim of providing high QoS and reliable transmission in this hostile wireless environment with restricted received signal level, development of appropriate physical layer, data link layer and network layer protocols is important. In this paper, we are interested particularly in error control protocols in data link layer. There are largely two categories of error control mechanisms, namely FEC and Automatic Repeat Request (ARQ). The focus of the paper will be on error control coding employed in FEC.

Shannon's theory on channel coding states that, given a channel there exist error control codes such that information can be transmitted across the channel at rate less than the channel capacity with arbitrarily low error rate. Since the publication of this theory more than 40 years ago, control coding theorists have been working towards discovering these codes. Among the error control codes found since then are some block codes such as BCH and Reed-Solomon codes and convolutional codes which will be discussed in this paper.

The paper is organized as follows. In Section II the concept of FEC will be introduced. Section III discusses first block codes in general and then some particular cyclic codes, namely BCH codes and Reed-Solomon codes. Convolutional code is presented in section IV and finally section V discusses some considerations of code selection and some methods to enhanced error control schemes.

II. FORWARD ERROR CORRECTION

Forward error correction schemes add redundant bits to the original transmitted frame in order to obtain known structures or patterns in the final transmitted frame. Figure 1 depicts the FEC operation where the encoder maps the *k*-bit transmitted data into an *n*-bit codeword and decoding and error correction to obtain the original data.

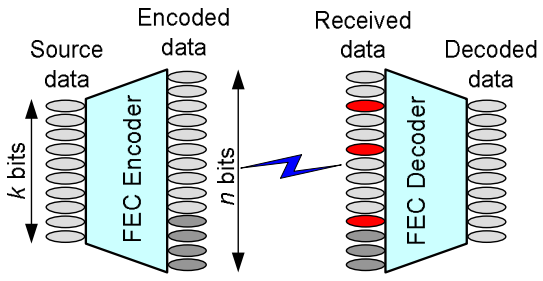

Figure 1: Operation of FEC based on block coding

With these known patterns, the decoder at the receiver will be able to detect and possibly correct the erroneous bits in the received frames. To be more precise, there are four possible outcomes at the decoder output:-

i. Received frame is identical to the transmitted frame. The FEC decoder produces the original transmitted data block as output.

ii. The received frame is different from the transmitted frame but has a certain error pattern know to the decoder. The FEC decoder will be able to detect and correct these erroneous bits and produces the original transmitted block as output.

iii. The received frame is different from the transmitted frame. The error pattern of the received frame can be detected but cannot be corrected by the FEC decoder. The FEC decoder reports an uncorrectable error.

iv. The received frame contains a typically rare pattern that the decoder cannot detect any errors. The FEC decoder produces a *k*-bit block that is different from the original transmitted block as output.

These scenarios are depicted in Figure 2.

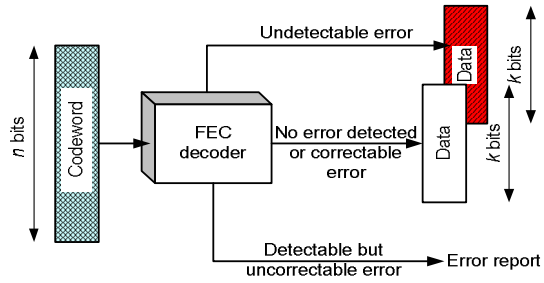

Figure 2: FEC decoding scenarios

III. BLOCK CODES

Block code schemes involves dividing the input data bit stream into block of *k*-bit streams and then mapping each *k*-bit block into *n*-bit block called codeword, where $n > k$ in the encoding process. $(n - k)$ check bits are added to each *k*-bit block. The ratio $(n - k)/k$ is called the redundancy of the code and the ratio k/n is called the code rate.

Before we go further, some important terms shall be defined to ensure clarity in the discussion to follow.

i) Hamming distance $d(\mathbf{c}_1, \mathbf{c}_2)$ between two binary code words $\mathbf{c}_1$ and $\mathbf{c}_2$ is the number of bits the two code words disagree.

ii) For an error correcting code that consists of codewords $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_s\}$, the minimum distance is defined as

$$d_{\min} = \min_{i \neq j} \{d(\mathbf{c}_i, \mathbf{c}_j)\}.$$

A code is capable of correcting up to *t* bits of errors if the code satisfy $d_{\min} \geq 2t + 1$. Putting this in another way, the maximum number of guaranteed correctable errors per codeword is

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

where $\lfloor x \rfloor$ stands for the largest integer not exceeding *x*. For error detection, the code that satisfies $t = d_{\min} - 1$ can detect up to *t* bits of errors in a block.

In general, the encoding operation of linear block codes can be represented with multiplying the generator matrix with the data block to be transmitted as

$$\mathbf{c} = \mathbf{mG},$$

where

$$\mathbf{m} = \{m_0 m_1 ... m_k\} \text{ and}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}.$$

G is a matrix consisting of *k* rows and *n* column. Since an (n, k) linear block code has 2^k valid codewords, it is a subspace of dimension *k*. Consequently, the row of **G** must be linearly independent, i.e. they must be the basis for the (n, k) code. The basis vectors for the code are not unique, and hence G is not unique.

A linear block code is said to be a systematic code if the first *k* bits of the code words are the data bits and the following $n - k$ bits are the parity check bits. Any generator matrix G can be reduced by row operations to the systematic form as follow.

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & \cdots & 0 & p_{11} & \cdots & p_{1n-k} \\ 0 & 1 & \cdots & 0 & p_{21} & \cdots & p_{2n-k} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & p_{k1} & \cdots & p_{kn-k} \end{array}\right] = [\mathbf{I}_k \mid \mathbf{P}]$$

At the receiver, the decoding operation of a linear block code involves a parity check matrix **H** that is used to detect and possibly correct bit errors. Associate with any (n, k) linear code, there is a dual code of dimension $n - k$. The generator matrix for the dual code is the parity check matrix **H** for the

linear block code. The rows of **H** spans a subspace of dimension $n - k$, which is the null space of **G**. For systematic codes, **H** can be deduce from **G** as

$$\mathbf{H} = [-\mathbf{P}' | \mathbf{I}_{n-k}].$$

To prove the claim that the generator matrix for the dual code is the parity check matrix, we denote the received data block as

$$\begin{aligned}\mathbf{y} &= \mathbf{c} + \mathbf{e} \\ &= \mathbf{mG} + \mathbf{e},\end{aligned}$$

where **e** is the error vector.

Since **H** is the null space of **G**, the matrices are orthogonal and $\mathbf{GH}' = \mathbf{0}$. Then

$$\begin{aligned}\mathbf{yH}' &= (\mathbf{mG} + \mathbf{e})\mathbf{H}' \\ &= \mathbf{mGH}' + \mathbf{eH}' \\ &= \mathbf{eH}'.\end{aligned}$$

$\mathbf{S} = \mathbf{eH}'$ is known as the syndrome of the error pattern. Note that the syndrome is purely a function of the error pattern and not the transmitted code word. Hence **H** must be the dual space of **G**.

The parity check results in $\mathbf{S} = \mathbf{0}$ when no error detected, which is equivalent to $\mathbf{e} = \mathbf{0}$. When errors are detected, **S** is nonzero and indicates the positions of the error bits. Hence the error bits can be corrected. In the cases of the number of errors exceeding the error correcting capacity of the code but within the detection capability, the syndrome indicates uncorrectable errors. There exist some rare error patterns that the syndrome reports no error.

A. Cyclic codes

An (n,k) linear block code **C** is said to be a cyclic code if the cyclic shift of a codeword is another codeword, i.e. for $\mathbf{c} = (c_0, c_1, ..., c_{n-2}, c_{n-1}) \in \mathbf{C}$, the following is also a valid codeword $\mathbf{c}' = (c_{n-1}, c_0, c_1, ..., c_{n-2}) \in \mathbf{C}$.

Cyclic codes are important practical error correcting codes due to their property and structure. One of the many reasons is that the general class of cyclic codes can be efficiently and easily implemented using linear feedback shift register based encoders and decoders.

The cyclical nature of these codes provides them with more structure that helps ease the encoding process. This structure allows us to associate a code with polynomials. For instance, we represent a four digit binary message, [1010] as $1 + x^2$. The encoding process involves a polynomial generator $g(x)$ and the data block to be transmitted.

For a given (*n*, *k*) cyclic code, the generator polynomial has certain properties related to *n* and *k*. The generator polynomial $g(x)$ must be a factors of $x^n + 1$ and of degree $n - k$. This procedure is best explained with an example.

Example 1: Encoding process

We consider a (7, 4) cyclic code for encoding a data block $m = [1010]$. To find the generator polynomial $g(x)$ for an $(n,k) = (7,4)$ code, we first factorize the polynomial $x^7 + 1$ and find the factor with degree $n - k = 3$.

$$g(x) = (x^3 + x^2 + 1)$$

To encode the data block $m = [1010]$, we multiply its corresponding polynomial $m(x) = x^2 + 1$ with $g(x)$.

$$\begin{aligned}c(x) &= m(x)g(x) \\ &= (x^3 + x)(x^3 + x^2 + 1) \\ &= x^6 + x^5 + x^4 + x.\end{aligned}$$

Thus the data block is encoded to the codeword $c = [1110010]$.

Another approach for encoding is without using polynomial function is to construct the generator matrix **G**. This approach allows us to encode systematic codes where the first *k* bits of the code are the transmitted data follow by *n-k* bits of parity check bits.

The generator matrix is an (*n* x *k*) matrix with its rows formed by the cyclic version of the generator polynomial.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

The systematic form can be obtained by performing row operation. We obtain

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

In the case of the example $c = \mathbf{mG}$, and the codeword obtained is $c = 1010011$, which the first 4 bits are the transmitted data **m**.

Example 2: Decoding process

Continuing from the systematic code, we assume that the code word is received with 1 error bit at position 4 i.e. $\mathbf{r} = 1011011$.

The parity check matrix **H** of the can be obtain from $\mathbf{H} = [-\mathbf{P}' | \mathbf{I}_{n-k}]$.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The syndrome is then S = *r*H´ which yields S = 011. By examining the **H** matrix, S is equal to column 4 which indicates a bit error in position 4 i.e. $e = 0001000$. We decode the data by adding the received sequence and the error pattern modulo 2 and drop the last 3 bits.

$$\hat{\mathbf{r}} = (\mathbf{r} + e) \bmod 2$$
$$= 1010011$$
$$\Rightarrow \mathbf{m} = 1010.$$

The H matrix carries some information of the code's error correction capability. The minimum distance of the code is equal to the minimum number of linearly combine column to produce all zeros. In our example, the code's minimum distance is 3 and therefore it's a single error correcting code but can detect 2 errors.

B. Bose-Chaudhuri-Hocqueghem (BCH) codes

BCH codes include a large class of cyclic codes. These codes employ binary as well as nonbinary alphabets. Reed-Solomon codes are a special class of nonbinary BCH codes which will be discussed later. Often expressed as an (n, k, t) BCH codes, the parameters are as follows.

$$n = 2^m - 1$$
$$n - k \le mt$$
$$d_{\min} = 2t + 1,$$

where *m* is any positive integer ≥ 3, i.e. $n \ge 7$.

These parameters provide larger selection of block lengths, code rates, and error correcting capabilities. Table 1 shows some examples of the parameters of binary BCH code. Observed that for a given pair of (*n*, *k*) for block length greater 7, there are some flexibility for the parameter *t*. System designers can adjust these parameters to suit the system requirements.

n	*k*	*t*	g(*x*) (octal form)
7	4	1	13
15	11	1	23
	7	2	721
	5	3	2467
63	57	1	103
	51	2	12471
	45	3	1701317
	39	4	166623567
	36	5	1033500423
127	120	1	211
	113	2	41567
	106	3	11554743
	99	4	3447023271
	92	5	624730022327
255	247	1	435
	239	2	267543
	199	7	7633031270420722341
	179	10	22624710717340332416300455

Table 1: Generator polynomials (in octal form) for BCH codes for various block lengths, data lengths and corresponding error correcting capabilities

For nonbinary BCH codes, *q*-ary alphabets are employed where *q* is any power of any given prime number. The block length of nonbinary BCH codes is given by $n = q^s - 1$. For *t*-error correcting, the number of check bit needed is $n - k \le 2st$.

C. Reed-Solomon (RS) codes

Reed-Solomon codes are a special subclass of nonbinary BCH codes with parameter *s*=1. Consequently, the parameters for an (*n*, *k*, *t*) RS-codes are as follows.

$$n = q - 1$$
$$n - k \le 2t$$
$$d_{\min} = 2t + 1.$$

RS codes are known to have optimal distance properties. For fixed number of check bits, RS codes provide higher error correcting capability compared to other block codes. A property related to the extra redundancy provided by nonbinary alphabets, 2 information symbols can be added to the length *n* RS code (resulting in length *n*+2) without reducing the minimum distance of the code. These codes are known as Extended-RS codes.

RS codes are effective in correcting burst errors and excellent for application with large set of input symbols. One famous practical application of RS codes is the Compact Disc (CD) error control system.

IV. CONVOLUTIONAL CODES

Convolutional encoders are encoders with memory. The outputs of the encoder not only depend on the current input bits but also certain amount of the previous bits. An (*n*, *k*, *m*) convolutional code encodes *k* input bits into *n* output bits. *m* is the memory of the decoder. The encoding involves the *k* current input bits and *m* previous input bits.

Unlike block codes, convolutional codes do not divide the bit stream into blocks. However, associated with the memory length of the encoder is the constraint length $K = m+1$ of the encoder. This parameter represents the *k*-tuple stages in the encoding shift registers. In practice, *n* and *k* are small and *K* is varied to control the redundancy of the code.

A. *Convolutional Encoding*

Figure 3 shows a ½ rate (2, 1, 2) convolutional encoder. The two outputs are serialized using a multiplexer, alternating y_1 and y_2 to the output line. The two code generators, upper and lower branch, can be represented with polynomial respectively as follow.

$$G_1(D) = 1 + D + D^2$$
$$G_2(D) = 1 + D^2 .$$

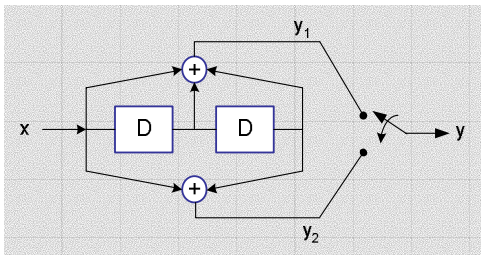

Figure 3:
½ rate (2, 1, 2) convolutional encoder

Example 3: Convolutional encoding

Consider the ½-rate encoder in Figure 3. Assume the initial content of the registers is zero, we wish to encode the bit sequence 101. In order to shift the whole sequence through the registers, 2 zeros are appended to the end of the sequence in order to flush the registers.

The code generator functions take the corresponding values in the shift register and input at each stage. Table 2 shows the encoding process that produces the output bits in column 3.

Input	Register contents	y1 y2
1	00	11
0	10	10
1	01	00
0	10	10
0	01	11

Table 2
Convolutional encoder:
Input, registers content and outputs

B. *Trellis diagram*

Trellis diagram is one of the many ways to represent a covolutional encoder. Figure 4 shows the trellis diagram of the encoder in Figure 3.

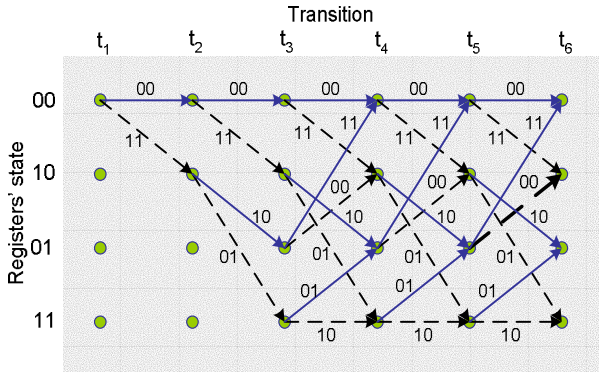

Figure 4
Trellis diagram

The trellis diagram represents all possible transition from the states in one time instant to the next. In figure 4, the solid blue arrows represent transitions caused by an input bit 0 while the dotted black arrows indicate transition caused by input bit 1. The binary digits on each arrow represent the encoder output of that transition known as branch code.

The information on the trellis diagram is useful especially in Viterbi decoding algorithm. Convolutional decoding consists of hard-decision decoding and soft-decision decoding. An example of hard decision decoding which eliminate unlikely paths in the trellis based on the Hamming distance between the branch code and the received sequence.

C. Convolutional decoding

Convolutional decoding is based on the Maximum Likelihood (ML) concept. Given a received sequence **Z**, the decoder chooses a sequence from all possible transmitted sequence that maximize the following likelihood function.

$$P\left(Z \mid X^{(m')}\right) = \max_{\text{all } X^{(m)}} \left\{P\left(Z \mid X^{(m)}\right)\right\}.$$

If the input sequences are equally likely, the ML decoder is an optimal decoder as it minimizes the error probability.

Viterbi decoding algorithm is one of the classical convolutional decoders. It performs ML decoding but reduces the computational complexity by exploiting the structure of the code trellis. The algorithm calculates a measure of similarity (eg. Hamming distance) between the received signal and all the trellis branch codes that is entering each state at any given instant t_i. The trellis path that is less unlikely or less similar to the received signal will be eliminated at each transition instant. By early elimination of unlikely paths, the decoding complexity is reduced.

Example 4: Convolutional decoding with Viterbi algorithm

In this example, we consider the output sequence from the encoder in Figure 3 that encode input data m = 11011 to output sequence y = 11 01 01 00 01. Suppose that the received sequence Z = 11 01 01 10 01 is the corrupted version of y.

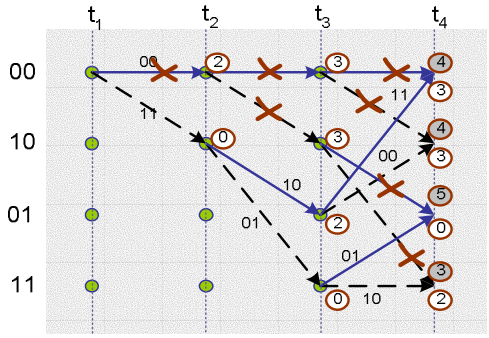

Figure 5
Viterbi decoding (4 transition instances)

The path metric is calculated based on the Hamming distance of the branch code and the received sequence. The numbers circled next to each node in Figure 5 indicate the path metrics.

As can be seen from Figure 5, at transition instant t_4, which is the constraint length of the code, there are two trellis paths entering each node (state). The paths that result in higher path metrics are eliminated.

Figure 6 shows the following transition into the 6-th transition instance with all unlikely paths eliminated. Observe that a single path has not been obtained in this example. Normally, the trellis path ends in state 00 corresponding to the flushed registers. However, in our example, by choosing the path that carries the lowest path metric i.e. 1, we can read the decoded data from the path (solid arrow = 0; dotted arrow =1).

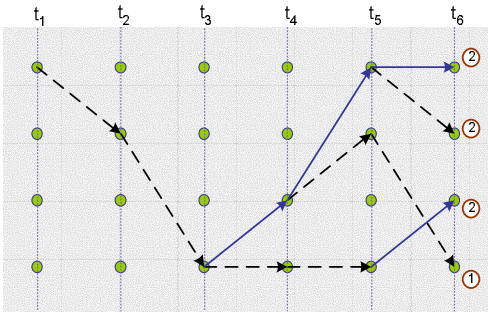

Figure 6
Viterbi decoding
(6 transition instances with eliminated paths)

The decoded data is 11011 which tally with the input data m in spite of the error in Z.

V. CODE SELECTION CONSIDERATIONS

When selecting a coding scheme for an application, often tradeoffs among performance, complexity and amount of redundancy can not be avoided. The selection criteria depend on the application's emphasis on coding gain, throughput and amount of processing delay that it can tolerate.

Many different methods involving the combination of different coding schemes and/or other error control schemes have been proposed in literatures. These methods overcome the shortcomings of individual coding schemes.

Examples are concatenated coding, interleaving and code puncturing which reduce the code length but maintain the coding gain or code performance. Hybrid ARQ combines error correcting coding and ARQ to improve throughput. For varying channel condition, long codes tend to introduce too much redundancy when the channel condition is good. Adaptive coding schemes are proposed to increase code efficiency in varying channel condition.

REFERENCES

[1] Bernard Sklar, "Digital Communications, Fundamentals and Application," Int'l ed., Prentice-Hall, 1998.

[2] J. G. Proakis, "Digital Communications," 4th ed., McGraw-Hill, 2001.

[3] William Stallings, "Wireless Communications and Networks," Upper Saddle River, New Jersey, Prentice-Hall, 2002.

[4] H. Liu et. al., "Error Control Schemes for Networks: An Overview," Baltzer Science Publishers BV, 1997.

[5] Pjilip K. McKinley, "A Study of Adaptive Forward Error Correction for Wireless Collaborative Computing," IEEE Trans. on Parallel & Distributed Systems, vol. 13 no. 9, Sept. 2002.