

# LECTURE TWO: Channel Coding and Error Control:

## Forward Error Control; Error Detection Methods; Parity Checking; Linear Block Codes, Cyclic Redundancy Checking; Feedback Error Control.

### INTRODUCTION

The main aim of any communication schemes is to provide error-free data transmission. In a communication system, information can be transmitted by analog or digital signals. For analog means, the amplitude of the signal reflects the information of the source, whereas for digital case, the information will first be translated into a stream of '0' and '1'. Then two different signals will be used to represent '0' and '1' respectively. The main advantage of using digital signal communication is that errors introduced by noise during the transmission can be detected and possibly corrected. For communication using cables, the random motion of charges in conducting (e.g. resistors), known as thermal noise, is the major source of noise. For wireless communication channels, noise can be introduced in various ways. In the case of mobile phones, noise also includes the signals sent by other mobile phone users in the system. Figure 1 and Figure 2 show the flow of a simple digital communication system.

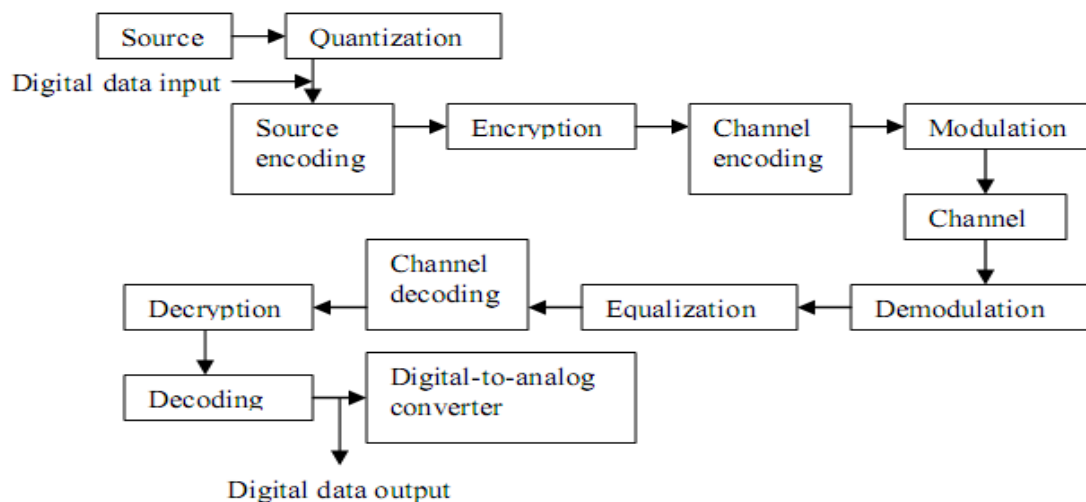


Fig 1: The flow diagram of a simple digital communication system.

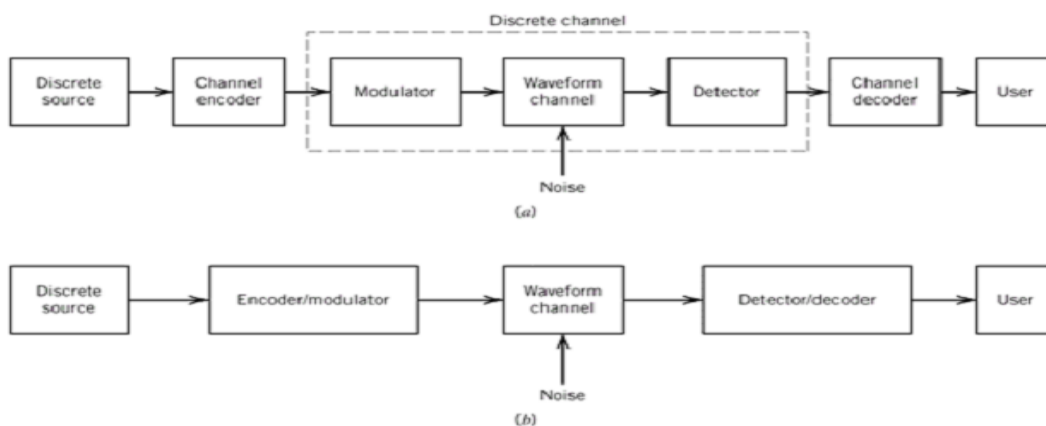
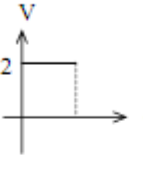
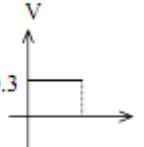
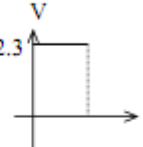
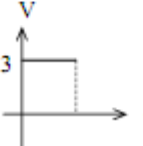
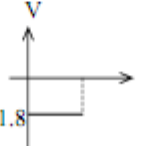
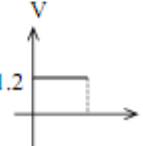


Fig 2: The flow diagram of a simple digital communication system showing noise addition.

Table 1 shows the difference between signal detection for analogue and digital systems. From the Table, it can be seen that error detection and correction as well as better accuracy of

transmitted information are possible for digital communication due to encoding and decoding.

Table 1: Difference between signal detection for analogue and digital systems.

Type	Signal transmitted	Channel	Signal received	Information Detected
<p>Analog: All amplitudes are possible. The value represents the information (e.g. loudness of the voice)</p>	<p>Information: 2V</p> 	<p>Noise: 0.3V (The mean of noise is 0V)</p> 	<p>2.3V</p> 	2.3V
<p>Digital: Only two amplitudes (<math>\pm 3V</math>) will be transmitted to represent '1' and '0' respectively</p>	<p>Information: '2' which has been encoded as '1' and will be using 3V pass through the channel.</p> 	<p>Noise: -1.8V (The mean of noise is 0V)</p> 	<p>1.2V</p> 	<p>Since 1.2V &gt; 0V, so it is still detected as 3V. That means the code '1' is transmitted and therefore the information is '2'.</p>

The fundamental resources at the disposal of a communications engineer are signal power, time and bandwidth. For a given communications environment, these three resources can be traded against each other. A general objective, however, is often to achieve maximum data transfer, in a minimum bandwidth *while maintaining an acceptable quality of transmission*. The quality of transmission, in the context of digital communications, is essentially concerned with the probability of bit error,  $P_e$ , at the receiver.

The Shannon-Hartley law shown in equation 1 for the capacity of a communications channel demonstrates two things.

- (i) Firstly it shows (quantitatively) how bandwidth (B) and signal power (S/N) may be traded in an ideal system,
- (ii) Secondly it gives a theoretical limit for the transmission rate of (reliable, i.e. error free) data (R) from a transmitter of given power, over a channel with a given bandwidth, operating in a given noise environment.

$$R_{\max} = B \log_2 \left( 1 + \frac{S}{N} \right) \text{ bit/s} \dots\dots\dots 1$$

In order to realize this theoretical limit, however, an appropriate coding scheme (which the Shannon-Hartley law assures us exists) must be found.

## Coding in Engineering

There are basically two types of coding used in Engineering: (i) Source Coding and Channel coding. Irrespective of the type of coding used, it is generally aimed at achieving the following:

- (i) To encrypt information for security purposes (Encryption)
- (ii) To reduce space for the data stream (Data Compression)
- (iii) To change the form of representation of the information so that it can be transmitted over a communication channel.
- (iv) To encode a signal so that any error that occurs during transmission can be detected and possibly corrected.

In practice, the objective of the design engineer is to realize the required data rate (often determined by the service being provided) within the bandwidth constraint of the available channel and the power constraint of the particular application. For a fixed  $S/N$ , the only practical option available for changing data quality from problematic to acceptable is to use *error-control coding*. Another practical motivation for the use of coding is to reduce the required  $S/N$  for a fixed bit error rate. This reduction in  $S/N$  may, in turn, be exploited to reduce the required transmitted power or reduce the hardware costs by requiring a smaller antenna size in the case of radio communications.

Moreover, the use of error-control coding adds *complexity* to the system, especially for the implementation of decoding operations in the receiver. Thus, the design trade-offs in the use of error-control coding to achieve acceptable error performance include considerations of bandwidth and system complexity.

Bit Error rates (BER) can be made smaller by the following methods:

1. By increasing transmitter power but this may not always be desirable, for example in man-portable systems where the required extra battery weight may be unacceptable.
2. Use of diversity which is effective against burst errors caused by signal fading. There are three main types of diversity: space diversity, frequency diversity, and time diversity. All these schemes incorporate redundancy in that data is, effectively, transmitted twice: i.e. via two paths, at two frequencies, or at two different times. In space diversity two or more antennas are used which are sited sufficiently far apart for fading at their outputs to be de-correlated. Frequency diversity employs two different frequencies to transmit the same information. (Frequency diversity can be in-band or out-band depending upon the frequency spacing between the carriers.) In time diversity systems, the same message is transmitted more than once at different times.
3. By introducing full duplex transmission, implying simultaneous 2-way transmission. Here when a transmitter sends information to a receiver, the information is 'echoed' back to the transmitter on a separate feedback channel. Information echoed back which contains errors can then be retransmitted. This technique requires twice the bandwidth of single direction (simplex) transmission, which may be unacceptable in terms of spectrum utilization.
4. A fourth method for coping with poor BER is automatic repeat request (ARQ). Here a simple error *detecting* code is used and, if an error is detected in a given data block, and then a request is sent via a feedback channel to retransmit that block. ARQ is very effective, for example in facsimile transmission. On long links with fast transmission rates, however, such as is typical in satellite communications, ARQ can be very difficult to implement.
5. The fifth technique for coping with high BER is to employ forward error correction coding (FECC). In common with three of the other four techniques FECC introduces redundancy, this time with data check bits interleaved with the information traffic

bits. It relies on the number of errors in a long block of data being close to the statistical average and, being a forward technique, requires no return channel. The widespread adoption of FECC was delayed, historically, because of its complexity and high cost of implementation relative to the other possible solutions. Complexity is now less of a problem following the proliferation of VLSI custom coder/decoder chips.

## Source Coding

Suppose a word 'Zebra' is going to be sent out. Before this information can be transmitted to the channel, it is first translated into a stream of bits ('0' and '1'). The process is called source coding. There are many commonly used ways to translate that. For example, if ASCII code is used, each alphabet will be represented by 7-bit so called the code word. The alphabets 'Z', 'e', 'b', 'r', 'a', will be encoded as:

'1010101', '0110110', '0010110', '0010111', '0001110'

The **ASCII code** is an example of fixed-length code, because each of the code word is of the same length (7 bits). However, in the view of efficient communication, the occurrence of 'Z' is not as often as that of 'e' and 'a'. If there is a way of encoding information such that the alphabets with higher probability of occurrence are assigned with shorter code words, and longer for the other letters which seldom come out, then on the whole it may be able to conserve the number of bits to be sent to the channel while sending the same information. This is what the variable length code can do. Example of this type of codes is the **Huffman Codes**.

## Channel Coding

As already mentioned, error control coding is a method to detect and possibly correct errors by introducing redundancy to the stream of bits to be sent to the channel. The design goal of channel coding (also referred as Error Control Coding) is basically to increase the resistance of a digital communication system to a channel noise. Error control coding is used to detect and often correct symbols which are received in error. The two main methods of error control are: Automatic repeat request and Forward error control techniques.

Figure 3 shows different types of error control codes (or channel coding methods).

ARQ		FBC			
Stop & wait	Continuous ARQ (pipelining)		Block codes		Convolutional codes
	Go-back-N	Selective repeat	Others (non-linear)	Group (linear)	
			Others (non-cyclic)	Polynomially generated (cyclic)	
				Golay	BCH
				Reed-Solomon	Binary BCH
					Hamming ( $e=1$ ) $e>1$

Figure 3: Different types of error control codes (or channel coding methods).

### Automatic Repeat Request (ARQ).

In this method when a receiver circuit detects errors in a block of data, it request that the data is retransmitted.

In this section we introduce three common flow and error control mechanisms: Stop-and-Wait ARQ. Go –Back -N ARQ, and Selective-Repeat ARQ. Although these are sometimes referred as protocols, we prefer the term mechanisms.

## 1) STOP-AND-WAIT ARQ

It is the simplest flow and error control mechanism. It has the following features:

- The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame. Keeping a copy allows the sender to retransmit lost or damaged frames until they are received correctly.
- For identification purposes, both data frames and acknowledgment (ACK) frames are numbered alternate 0 and 1. A data (0) frame is acknowledged by an ACK 1 frame, indicating that the receiver has received data frame 0 and is now expecting data frame 1. This numbering allows for identification for data frames in case of duplicate transmission (important in the case of lost acknowledgment or delayed acknowledgment, as we will see shortly).
- A damaged or lost frame is treated in the same manner by the receiver. If the receiver detects an error in the received frame, it simply discards the frame and sends no acknowledgment. If the receiver receives a frame that is out of order (0 instead of 1 or 1 instead of 0) , it knows that a frame is lost. It discards the out-of-order received frame.
- The sender has a control variable, which we call S, that holds the number of the recently sent frame (0 or 1). The receiver has a control variable, which we call R

that holds the number of the next frame expected (0 or 1).

- The sender starts a timer when it sends a frame. If an acknowledgment is not received within an allotted time period, the sender assumes that the frame was lost or damaged and resends it.
- The receiver sends only positive acknowledgment for frames received safe and sound; it is silent about the frames damaged or lost. The acknowledgment number always defines the number of the next expected frame. If frame 0 is received ACK 1 is sent: if frame 1 is received, ACK 0 is sent.

In the transmission of a frame, we can have four situations: normal operation, the frame is lost, the acknowledgment is lost, or the acknowledgment is delayed.

**CASE 1: Normal Operation** In a normal transmission, the sender sends frame 0 and waits to receive ACK 1. When ACK 1 is received, it sends frame 1 and then waits to receive ACK 0, and so on. The ACK must be received before the timer set for each frame expires. Figure 4 shows successful frame transmissions.

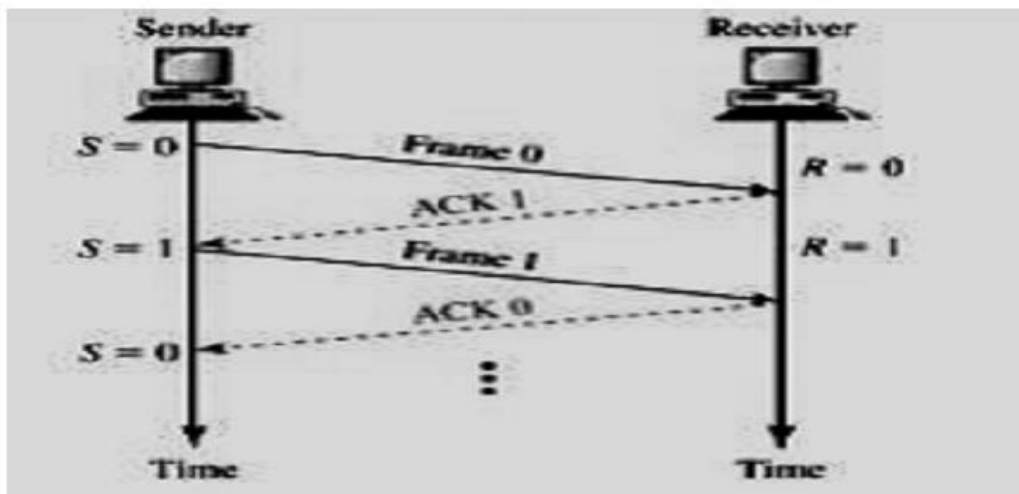


Fig 4

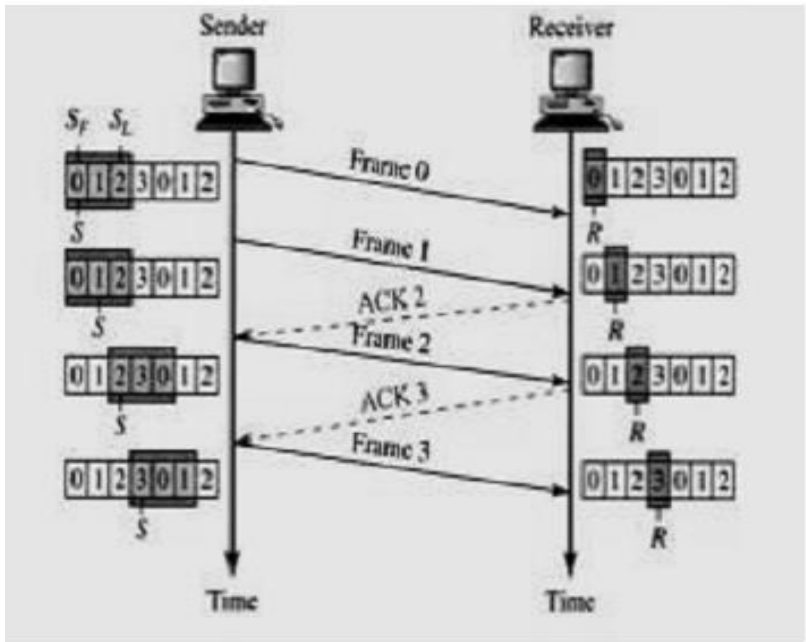
**CASE 2: Lost or Damaged Frame** A lost or damaged frame is handled in the same way by the receiver; when the receiver receives a damaged frame, it discards it, which essentially means the frame is lost. The receiver remains silent about a lost frame and keeps its value of  $R$ . For example, in Fig 5, the sender transmits frame 1, but it is lost. The receiver does nothing, retaining the value of  $R$  (1). After the timer at the sender site expires, another copy of frame 1 is sent.





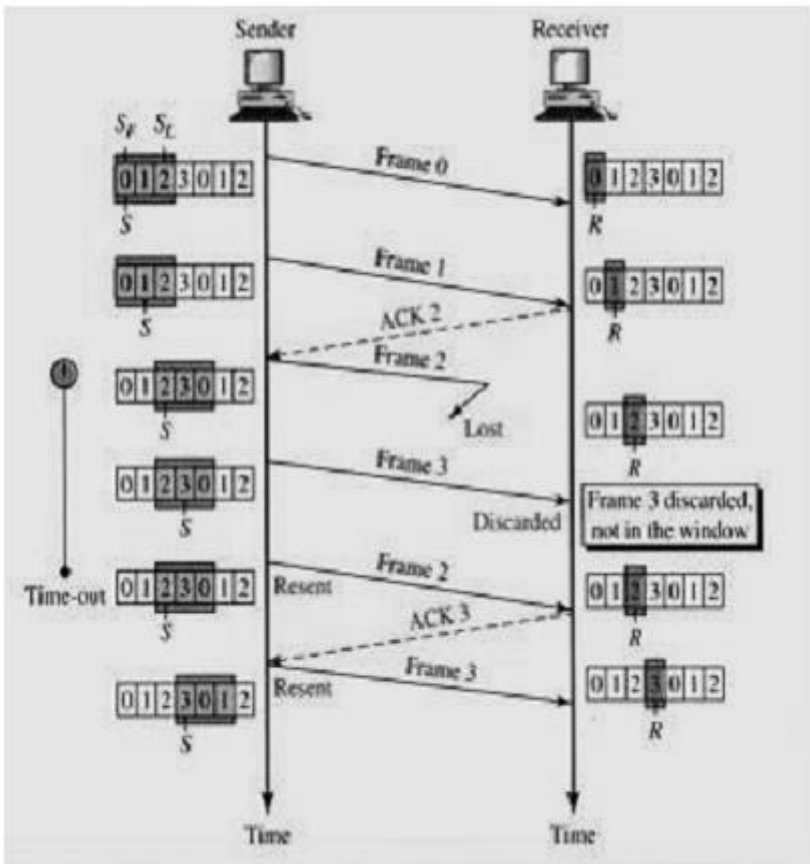






**Figure 8**

**CASE 2: Damaged or Lost Frame** Now let us see what happens if a frame is lost. Figure 9 shows that frame 2 is lost. Note that when the receiver receives frame 3 it is discarded because the receiver is expecting frame 2 not frame 3 (according to its window). After the timer for frame 2 expires at the sender site, the sender sends frames 2 and 3 (it goes back to 2).



**Figure 9**

### **CASE 3: Damaged or Lost Acknowledgment**

If an acknowledgment is damaged or lost, we can have two situations. If the next acknowledgment arrives before the expiration of any timer, there is no need for retransmission of frames because acknowledgments are cumulative in this protocol. ACK 4 means ACK 1 to ACK 4. So if ACK 1, ACK 2, and ACK 3 are lost. ACK 4 covers them. However, if the next ACK arrives after the time-out, the frame and all the frames after that are resent. Note that the receiver never resends an ACK. The figure and details are left out as an exercise.

### **3. SELECTIVE REPEAT ARQ**

Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend N frames when just one frame is damaged: only the damaged frame is resent. This mechanism is called Selective Repeat ARQ. It is more efficient for noisy links, but the processing at the receiver is more complex. Let us show the operation of the mechanism with an example of a lost frame, as shown in Figure 10. Frames 0 and 1 are accepted and received because they are in the range specified by the receiver window. When frame 3 is received, it is also accepted for the same reason. However, the receiver sends a NAK 2 to show that frame 2 has not been received. When the sender receives the NAK 2, it resends only frame 2, which is then accepted because it is in the range of the window.

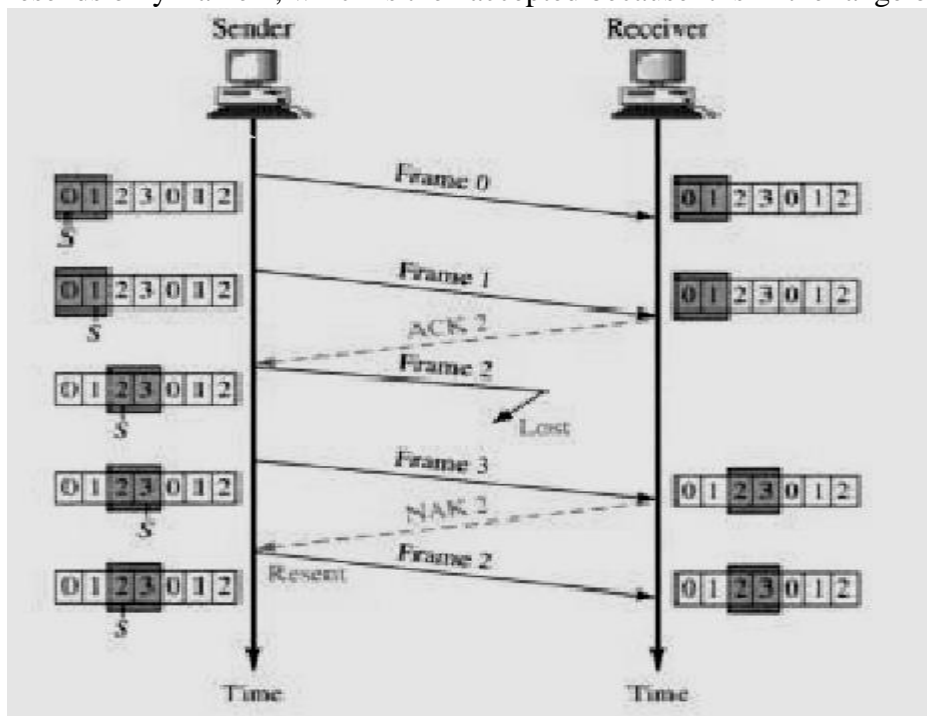


Figure 10

### **Forward Error Correction (FEC)**

In this method, the transmitted data is encoded so that the data can correct as well as detect errors caused by channel noise. The choice of ARQ or FEC depends on the particular application. ARQ is often used when there is a full duplex (2-way) channel because it is relatively inexpensive to implement. FEC is used when the channel is not full duplex or where ARQ is not desirable because real time is required.

When we talk about digital systems, be it a digital computer or a digital communication set-up, the issue of error detection and correction is of great practical significance. Errors creep into the bit stream owing to noise or other impairments during the course of its transmission from the transmitter to the receiver. Any such error, if not detected and subsequently corrected, can be disastrous, as digital systems are sensitive to errors and tend to malfunction if the bit error rate is more than a certain threshold level. Error detection and correction, involves the addition of extra bits, called check bits, to the information-carrying bit stream to give the resulting bit sequence a unique characteristic that helps in detection and localization of errors. These additional bits are also called redundant bits as they do not carry any information. While the addition of redundant bits helps in achieving the goal of making transmission of information from one place to another error free or reliable, it also makes it inefficient. The Channel Encoder will add bits to the message bits to be transmitted systematically. After passing through the channel, the Channel decoder will detect and correct the errors. A simple example is to send '000' ('111' correspondingly) instead of sending only one '0' ('1' correspondingly) to the channel. Due to noise in the channel, the received bits may become '001'. But since either '000' or '111' could have been sent. By majority logic decoding scheme, it will be decoded as '000' and therefore the message has been a '0'. In general the channel encoder will divide the input message bits into blocks of  $k$  messages bits and replaces each  $k$  message bits block with a  $n$ -bit code word by introducing  $(n-k)$  check bits to each message block. In this section, we will examine some common error detection and correction codes.

### **PARITY CODE**

A parity bit is an extra bit added to a string of data bits in order to detect any error that might have crept into it while it was being stored or processed and moved from one place to another in a digital system. We have an even parity, where the added bit is such that the total number of "1"s in the data bit string becomes even, and an odd parity, where the added bit makes the total number of "1"s in the data bit string odd. This added bit could be a '0' or a '1'. As an example, if we have to add an even parity bit to 01000001 (the eight-bit ASCII code for 'A'), it will be a '0' and the number will become 001000001. If we have to add an odd parity bit to the same number, it will be a '1' and the number will become 101000001. The odd parity bit is a complement of the even parity bit. The most common convention is to use even parity, that is, the total number of 1s in the bit stream, including the parity bit, is even.

The parity check can be made at different points to look for any possible single-bit error, as it would disturb the parity. This simple parity code suffers from two limitations. Firstly, it cannot detect the error if the number of bits having undergone a change is even. Although the number of bits in error being equal to or greater than 4 is a very rare occurrence, the addition of a single parity cannot be used to detect two-bit errors, which is a distinct possibility in data storage media such as magnetic tapes. Secondly, the single-bit parity code cannot be used to localize or identify the error bit even if one bit is in error. There are several codes that provide self-single-bit error detection and correction mechanisms.

### **REPETITION CODE**

The repetition code makes use of repetitive transmission of each data bit in the bit stream. In the case of threefold repetition, '1' and '0' would be transmitted as '111' and '000' respectively. If, in the received data bit stream, bits are examined in groups of three bits, the occurrence of an error can be detected. In the case of single-bit errors, '1' would be received as 011 or 101 or 110 instead of 111, and a '0' would be received as 100 or 010 or 001 instead of 000. In both cases, the code becomes self-correcting if the bit in the majority is taken as the correct bit. There are various forms in which the data are sent using the repetition code. Usually, the data bit stream is broken into blocks of bits, and then each block of data is sent

some predetermined number of times. For example, if we want to send eight-bit data given by 11011001, it may be broken into two blocks of four bits each. In the case of threefold repetition, the transmitted data bit stream would be 110111011101100110011001. However, such a repetition code where the bit or block of bits is repeated 3 times is not capable of correcting two-bit errors, although it can detect the occurrence of error. For this, we have to increase the number of times each bit in the bit stream needs to be repeated. For example, by repeating each data bit 5 times, we can detect and correct all two-bit errors. The repetition code is highly inefficient and the information throughput drops rapidly as we increase the number of times each data bit needs to be repeated to build error detection and correction capability.

## Cyclic Redundancy Code

Cyclic redundancy check (CRC) codes provide a reasonably high level of protection at low redundancy level. The cycle code for a given data word is generated as follows. The data word is first appended by a number of 0s equal to the number of check bits to be added. This new data bit sequence is then divided by a special binary word whose length equals  $n+1$ ,  $n$  being the number of check bits to be added. The remainder obtained as a result of modulo-2 division is then added to the dividend bit sequence to get the cyclic code. The code word so generated is completely divisible by the divisor used in the generation of the code. Thus, when the received code word is again divided by the same divisor, an error-free reception should lead to an all '0' remainder. A nonzero remainder is indicative of the presence of errors.

The probability of error detection depends upon the number of check bits,  $n$ , used to construct the cyclic code. It is 100 % for single-bit and two-bit errors. It is also 100 % when an odd number of bits are in error and the error bursts have a length less than  $n+1$ . The probability of detection reduces to  $1 - (1/2)^{n-1}$  for an error burst length equal to  $n+1$ , and to  $1 - (1/2)^n$  for an error burst length greater than  $n+1$ .

## Cross Word Error Correction

- This is an extension of the use of parity bits to enable error recovery.
- Assume that data is sent in 7 bit words and a single parity bit is appended (Shown as  $R_x$  in the table below). This parity bit may be either even or odd.
- After 7 data words have been sent, another 8 bit check word is appended. Bit 1 of this word is a parity bit for bit 1 in all 7 data words. Bit 2 is a parity bit for bit 2 in all of the 7 data words etc. etc. These are shown as  $C_x$

bits in the table below.

- If bit 3 in word 4 is errored in transmission it will show up as two parity bit errors, i.e., parity bit  $R4$  and  $C3$ . This allows the errored bit to be identified and the error to be corrected.
- The problem with this correction method is in the low transmission efficiency. For example, the above arrangement sends  $7 * 7(49)$  bits of data but  $8 * 8(64)$  bits are required for error correction - the efficiency is  $49/64 = 77\%$ .

Word 1	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	R1
Word 2	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	R2
Word 3	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	R3
Word 4	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	R4
Word 5	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	R5
Word 6	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	R6
Word 7	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	R7
Check Word	C1	C2	C3	C4	C5	C6	C7	C8

## Hamming Code

We have seen, in the case of the error detection and correction codes described above, how an increase in the number of redundant bits added to message bits can enhance the capability of the code to detect and correct errors. If we have a sufficient number of redundant bits, and if these bits can be arranged such that different error bits produce different error results, then it should be possible not only to detect the error bit but also to identify its location. In fact, the addition of redundant bits alters the 'distance' code parameter, which has come to be known as the Hamming distance. The Hamming distance is nothing but the number of bit disagreements between two code words. For example, the addition of single-bit parity results in a code with a Hamming distance of at least 2. The smallest Hamming distance in the case of a threefold repetition code would be 3. Hamming noticed that an increase in distance enhanced the code's ability to detect and correct errors. Hamming's code was therefore an attempt at increasing the Hamming distance and at the same time having as high an information throughput rate as possible.

The algorithm for writing the generalized Hamming code is as follows:



1. The generalized form of code is  $P_1P_2D_1P_3D_2D_3D_4P_4D_5D_6D_7D_8D_9D_{10}D_{11}P_5 \dots$ , where  $P$  and  $D$  respectively represent parity and data bits.
2. We can see from the generalized form of the code that all bit positions that are powers of 2 (positions 1, 2, 4, 8, 16, ...) are used as parity bits.
3. All other bit positions (positions 3, 5, 6, 7, 9, 10, 11, ...) are used to encode data.
4. Each parity bit is allotted a group of bits from the data bits in the code word, and the value of the parity bit (0 or 1) is used to give it certain parity.
5. Groups are formed by first checking  $N - 1$  bits and then alternately skipping and checking  $N$  bits following the parity bit. Here,  $N$  is the position of the parity bit; 1 for  $P_1$ , 2 for  $P_2$ , 4 for  $P_3$ , 8 for  $P_4$  and so on. For example, for the generalized form of code given above, various groups of bits formed with different parity bits would be  $P_1D_1D_2D_4D_5 \dots$ ,  $P_2D_1D_3D_4D_6D_7 \dots$ ,  $P_3D_2D_3D_4D_8D_9 \dots$ ,  $P_4D_5D_6D_7D_8D_9D_{10}D_{11} \dots$  and so on. To illustrate the formation of groups further, let us examine the group corresponding to parity bit  $P_3$ . Now, the position of  $P_3$  is at number 4. In order to form the group, we check the first three bits ( $N - 1 = 3$ ) and then follow it up by alternately skipping and checking four bits ( $N = 4$ ).

The Hamming code is capable of correcting single-bit errors on messages of any length. Although the Hamming code can detect two-bit errors, it cannot give the error locations. The number of parity bits required to be transmitted along with the message, however, depends upon the message length, as shown above. The number of parity bits  $n$  required to encode  $m$  message bits is the smallest integer that satisfies the condition  $(2^n - n) > m$ .

**Table 2.9** Generation of Hamming code.

	$P_1$	$P_2$	$D_1$	$P_3$	$D_2$	$D_3$	$D_4$
Data bits (without parity)			0		1	1	0
Data bits with parity bit $P_1$	1		0		1		0
Data bits with parity bit $P_2$		1	0			1	0
Data bits with parity bit $P_3$				0	1	1	0
Data bits with parity	1	1	0	0	1	1	0

The most commonly used Hamming code is the one that has a code word length of seven bits with four message bits and three parity bits. It is also referred to as the Hamming (7, 4) code. The code word sequence for this code is written as  $P_1P_2D_1P_3D_2D_3D_4$ , with  $P_1$ ,  $P_2$  and  $P_3$  being the parity bits and  $D_1$ ,  $D_2$ ,  $D_3$  and  $D_4$  being the data bits. We will illustrate step by step the process of writing the Hamming code for a certain group of message bits and then the process of detection and identification of error bits with the help of an example. We will write the Hamming code for the four-bit message 0110 representing numeral '6'. The process of writing the code is illustrated in Table 2.9, with even parity.

Thus, the Hamming code for 0110 is 1100110. Let us assume that the data bit  $D_1$  gets corrupted in the transmission channel. The received code in that case is 1110110. In order to detect the error, the parity is checked for the three parity relations mentioned above. During the parity check operation at the receiving end, three additional bits  $X$ ,  $Y$  and  $Z$  are generated by checking the parity status of  $P_1D_1D_2D_4$ ,  $P_2D_1D_3D_4$  and  $P_3D_2D_3D_4$  respectively. These bits are a '0' if the parity status is okay, and a '1' if it is disturbed. In that case,  $ZYX$  gives the position of the bit that needs correction. The process can be best explained with the help of an example.

Examination of the first parity relation gives  $X = 1$  as the even parity is disturbed. The second parity relation yields  $Y = 1$  as the even parity is disturbed here too. Examination of the third relation gives  $Z = 0$  as the even parity is maintained. Thus, the bit that is in error is positioned at 011 which is the binary equivalent of '3'. This implies that the third bit from the MSB needs to be corrected. After correcting the third bit, the received message becomes 1100110 which is the correct code.



### Example 2.6

By writing the parity code (even) and threefold repetition code for all possible four-bit straight binary numbers, prove that the Hamming distance in the two cases is at least 2 in the case of the parity code and 3 in the case of the repetition code.

#### Solution

The generation of codes is shown in Table 2.10. An examination of the parity code numbers reveals that the number of bit disagreements between any pair of code words is not less than 2. It is either 2 or 4. It is 4, for example, between 00000 and 10111, 00000 and 11011, 00000 and 11101, 00000 and 11110 and 00000 and 01111. In the case of the threefold repetition code, it is either 3, 6, 9 or 12 and therefore not less than 3 under any circumstances.

### Example 2.7

It is required to transmit letter 'A' expressed in the seven-bit ASCII code with the help of the Hamming (11, 7) code. Given that the seven-bit ASCII notation for 'A' is 1000001 and that the data word gets

Table 2.10 Example 2.6.

Binary number	Parity code	Three-time repetition Code	Binary number	Parity code	Three-time repetition code
0000	00000	000000000000	1000	11000	100010001000
0001	10001	000100010001	1001	01001	100110011001
0010	10010	001000100010	1010	01010	101010101010
0011	00011	001100110011	1011	11011	101110111011
0100	10100	010001000100	1100	01100	110011001100
0101	00101	010101010101	1101	11101	110111011101
0110	00110	011001100110	1110	11110	111011101110
0111	10111	011101110111	1111	01111	111111111111

corrupted to 1010001 in the transmission channel, show how the Hamming code can be used to identify the error. Use even parity.

#### Solution

- The generalized form of the Hamming code in this case is  $P_1P_2D_1P_3D_2D_3D_4P_4D_5D_6D_7 = P_1P_21P_3000P_4001$ .
- The four groups of bits using different parity bits are  $P_1D_1D_2D_4D_5D_7$ ,  $P_2D_1D_3D_4D_6D_7$ ,  $P_3D_2D_3D_4$  and  $P_4D_5D_6D_7$ .
- This gives  $P_1 = 0$ ,  $P_2 = 0$ ,  $P_3 = 0$  and  $P_4 = 1$ .
- Therefore, the transmitted Hamming code for 'A' is 00100001001.
- The received Hamming code is 00100101001.
- Checking the parity for the  $P_1$  group gives '0' as it passes the test.
- Checking the parity for the  $P_2$  group gives '1' as it fails the test.
- Checking the parity for the  $P_3$  group gives '1' as it fails the test.
- Checking the parity for the  $P_4$  group gives '0' as it passes the test.
- The bits resulting from the parity check, written in reverse order, constitute 0110, which is the binary equivalent of '6'. This shows that the bit in error is the sixth from the MSB.
- Therefore, the corrected Hamming code is 00100001001, which is the same as the transmitted code.
- The received data word is 1000001.

## Block Codes

Denoted by  $(n, k)$  a block code is a collection of code words each with length  $n$ ,  $k$  information bits and  $r = n - k$  check bits. It is linear if it is closed under addition mod 2. A **Generator Matrix**  $G$  (of order  $k \times n$ ) is used to generate the code.

$$G = [ I_k \quad P ]_{k \times n} \dots\dots\dots$$

where  $I_k$  is the  $k \times k$  identity matrix and  $P$  is a  $k \times (n - k)$  matrix selected to give desirable properties to the code produced.

For example, denote  $D$  to be the message,  $G$  to be the generator matrix,  $C$  to be code word. For

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

We get the collection of code words in the Table below

Table: Collection of Code words

Messages (D)	Code words (C)
0 0 0	0 0 0 0 0 0
0 0 1	0 0 1 1 1 0
0 1 0	0 1 0 1 0 1
0 1 1	0 1 1 0 1 1
1 0 0	1 0 0 0 1 1
1 0 1	1 0 1 1 0 1
1 1 0	1 1 0 1 1 0
1 1 1	1 1 1 0 0 0

In particular when  $D = [0 \ 1 \ 1]$ ,

$$C = DG = [0 \ 1 \ 1] \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} = [0 \ 1 \ 1 \ 0 \ 1 \ 1].$$

Now define the **Parity Check Matrix** to be

$$H = [ P^T \quad I_{n-k} ]_{(n-k) \times n} \dots\dots\dots$$

As long as the code word  $C$  is generated by  $G$ , the product  $CH^T = 0$ .

Denote the **received code word** after passing through the channel be  $R$ . It is made up of the original code word  $C$  and error bits  $E$  from the channel. Further define the **Error Syndrome** to be

$$S = RH^T \quad \text{Then,}$$

$$R = C + E \\ S = RH^T = (C + E)H^T = CH^T + EH^T = EH^T$$

If  $S = 0$ ,  $R \equiv C$  and  $D$  is the first  $k$  bits of  $R$ . If  $S \neq 0$  and  $S$  is the  $j$ th row of  $H^T$ , then it implies an error occurs in the  $j$ th position of  $R$ .

To illustrate, suppose after passing through the channel, the received code word is  $R = [0 \ 1 \ 1 \ 1 \ 1 \ 0]$ .

Thus  $RH^T = [0 \ 1 \ 1 \ 1 \ 1 \ 0]H^T = [1 \ 0 \ 1]$  which is the second row of  $H^T$ . Thus it implies there exists an error in the second bit of the received code word. So we can correct it and detect that  $[0 \ 0 \ 1 \ 1 \ 1 \ 0]$  should have been sent.

### 3. Applications for error control codes:

- 1) Compact disc players provide a growing application area for FECC.
- 2) In CD applications the powerful Reed-Solomon code is used since it works at a symbol level, rather than at a bit level, and is very effective against burst errors.
- 3) The Reed-Solomon code is also used in computers for data storage and retrieval.
- 4) Digital audio and video systems are also areas in which FEC is applied.
- 5) Error control coding, generally, is applied widely in control and communications systems for aerospace applications, in mobile (GSM).
- 6) Cellular telephony and for enhancing security in banking and barcode readers.