BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 3

DCA2201

COMPUTER NETWORKING

Unit 3

Data Link Layer – Framing and Error Detection

Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	Introduction	-	-	3
	1.1 Objectives	-	-	
2	Framing	1, 2, 3, 4, 5, 6	1	4 - 9
	2.1 Byte-Oriented Protocols (BISYNC, PPP, DDCMP)	-	-	
	2.2 Bit-Oriented Protocols (HDLC)	-	-	
	2.3 Clock-based Framing (SONET)	-	-	
3	Error Detection and Correction	7, 8, 9, 10, 11	2	10 - 16
	3.1 Error Correcting Codes	-	-	
	3.2 Error Detecting Codes	-	-	
4	Summary	-	-	17
5	Terminal Questions	-	-	17
6	Answers	-	-	18 -19

1. INTRODUCTION

In the previous unit, we discussed the basics of data link layer and local area networks and link virtualization in datalink layer. This unit introduces the concept of framing and error detection in Data link layer. Data link layer encapsulates each network layer packet within a link layer frame before transmission over the link. Each frame consists of a data field and a number of header fields. These data frames are not error free. Different types of errors can occur during transmission over a network. So various error control measures are required to ensure the reliability of the data sent.

We will start this unit with different types of framing protocols. In the next session, we will discuss different types of error correction and error detection methods.

1.1 Objectives

After studying this unit, you should be able to:

- *Describe framing*
- *Explain framing protocols*
- *Describe different types of error correcting methods*
- *Explain different types of error detecting methods*

2. FRAMING

Framing is the process of breaking down a stream of bits into smaller portions called frames. In framing, each frame can be separately addressed. The frame contains a header, payload (data) and a trailer. The header contains addressing information, trailer includes error handling information like check sum. Link layer either delivers entire frame payload or none of it.

Recognizing the frame limit, that is, finding out the bits constitute a frame, is the issue which needs to be addressed in framing.

There are several ways to address this framing problem. In this section, we will discuss different protocols such as Byte-oriented Protocols, Bit-oriented Protocols, and clock-based framing which are used for framing data.

2.1 Byte-Oriented Protocols (BISYNC, PPP, DDCMP)

This is one of the oldest approaches in framing, which considers each frame as a collection of bytes, rather than a collection of bits. Byte-oriented approach is represented by older protocols such as the Binary Synchronous Communication (BISYNC) protocol developed by IBM in the late 1960s, and the Digital Data Communication Message Protocol (DDCMP) used in Digital Equipment Corporation's DECNET. The more recent and widely used Point-to-Point Protocol (PPP) provides another example of this approach.

Following figure 3.1 shows the BISYNC protocol's frame format. Each packet contains a sequence of labelled fields. Above each field, there is a number indicating the length of that field in bits.

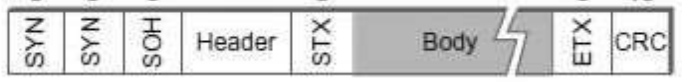

Figure 3.1: BISYNC Frame Format

BISYNC uses special characters known as *sentinel characters* to indicate where frames start and end. The beginning of a frame is denoted by sending a special SYN (synchronization) character. The data portion of the frame is then contained between two more special characters: STX (start of text) and ETX (end of text). The SOH (start of header) field serves much the same purpose as the STX field. The problem with the sentinel approach is that the ETX character might appear in the data portion of the frame. BISYNC overcomes this problem by “escaping” the ETX character by preceding it with a DLE (data-link-escape) character whenever it appears in the body of a frame; the DLE character is also escaped (by preceding it with an extra DLE) in the frame body. This approach is often called *character stuffing* because extra characters are inserted in the data portion of the frame. CRC field is used to detect transmission errors.

The Point-to-Point Protocol (PPP), frame format is shown in figure 3.2. The special start-of-text character, denoted as flag field, is 01111110. Address and control field include default values. The protocol field is used for demultiplexing. Frame payload size is 1500 bytes by default, but it can be negotiated. Checksum field is either 2 (default) or 4 bytes long.

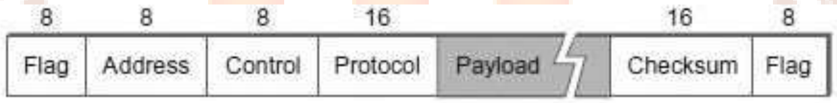

Figure 3.2: Point-to-Point Protocol (PPP) frame format

DDCMP uses a byte-counting approach in which the number of bytes contained in a frame can be included as a field in the frame header. Following figure 3.3 shows a DDCMP frame format.

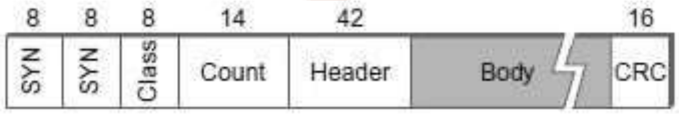

Figure 3.3: DDCMP frame format

In this frame, the COUNT field specifies how many bytes are contained in the frame’s body. If the count field is corrupted by a transmission error, then the end of the frame would not be correctly detected. This will create a framing error, because there is a chance that the receiver accumulates as many bytes based on bad COUNT field indication.

2.2 Bit-Oriented Protocols (HDLC)

A bit-oriented protocol views the frame as a collection of bits. The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol; SDLC was later standardized by the ISO as the High-Level Data Link Control (HDLC) protocol. HDLC frame format is given in figure 3.4.

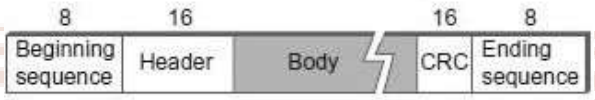

Figure 3.4: HDLC frame format

HDLC denotes the beginning and end of the frame by a bit sequence 01111110. The same sequence can appear anywhere in the body of the frame. Bit oriented protocols use the analog of DLE character, a technique known as *bit stuffing*. Bit stuffing in the HDLC protocol works as follows. On the sending side, any time five consecutive 1s have been transmitted from the body of the message, the sender inserts a 0 before transmitting the next bit. On the receiving side, should five consecutive 1s arrive, the receiver makes its decision based on the next bit it sees (i.e., the bit following the five 1s). If the next bit is a 0, it must have been stuffed, and so the receiver removes it. If the next bit is a 1, then one of two things is true: Either this is the end-of-frame marker or an error has been introduced into the bit stream. By looking at the next bit, the receiver can distinguish between these two cases. If it sees a 0 (i.e., the last 8 bits it has looked at are: 01111110), then it is the end-of-frame marker; if it sees a 1 (i.e., the last 8 bits it has looked at are: 01111111), then there must have been an error and the whole frame is discarded.

2.3 Clock-based Framing (SONET)

Clock- based framing is another approach to framing, and is represented by the Synchronous Optical Network (SONET) standard. SONET was first proposed by Bell Communications Research (Bellcore), and then developed under the American National Standards Institute (ANSI) for digital transmission over optical fiber; it has since been adopted by the ITU-T.

SONET has been for many years the dominant standard for long-distance transmission of data over optical networks.

SONET addresses both the framing problem and the encoding problem. It also addresses a problem that is very important for phone companies – the multiplexing of several low-speed links onto one high-speed link. STS-1 is the lowest speed SONET link, which runs at 51.84 Mbps. An STS-1 frame is shown in figure 3.5. It is arranged as 9 rows of 90 bytes each, and the first 3 bytes of each row are overhead, with the rest being available for data that is being transmitted over the link. The first 2 bytes of the frame contain a special bit pattern, and it is these bytes that enable the receiver to determine where the frame starts.

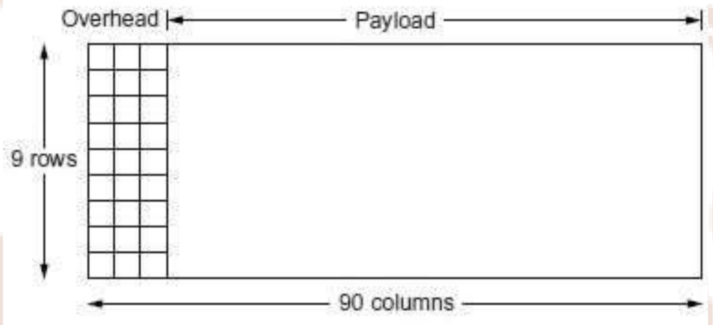

Figure 3.5: A SONET STS-1 Frame

However, since bit stuffing is not used, there is no reason why this pattern will not occasionally turn up in the payload portion of the frame. To guard against this, the receiver looks for the special bit pattern consistently, hoping to see it appearing once every 810 bytes, since each frame is 9×90 = 810 bytes long. When the special pattern turns up in the right place enough times, the receiver concludes that it is in sync and can then interpret the frame correctly.

SONET supports the multiplexing of multiple low-speed links in the following way. A given SONET link runs at one of a finite set of possible rates, ranging from 51.84 Mbps (STS-1) to 2488.32 Mbps (STS-48), and beyond. All of these rates are integer multiples of STS-1. The significance for framing is that a single SONET frame can contain sub-frames for multiple lower-rate channels. Another feature is that each frame is 125µs long. This means that at STS-1 rates, a SONET frame is 810 bytes long, while at STS-3 rates, it is 2430 (3x810=2430) bytes long. This also means that three STS-1 frames fit exactly in a single STS-3 frame. Similarly, STS-N frame can be thought of as consisting of N STS-1 frames.

Although it is accurate to view an STS-N signal as being used to multiplex N STS-1 frames, the payload from these STS-1 frames can be linked together to form a larger STS-N payload; such a link is denoted STS-Nc (for *concatenated*). Following figure 3.6 depicts concatenation in the case of three STS-1 frames being concatenated into a single STS-3c frame.

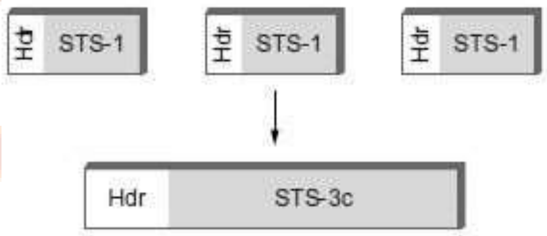

Figure 3.6: Three STS-1 frames multiplexed onto one STS-3c frame

The significance of a SONET link being designated as STS-3c rather than STS-3 is that, in STS-3c, the user of the link can view it as a single 155.25- Mbps pipe, whereas an STS-3 should really be viewed as three 51.84-Mbps links that happen to share a fiber.

Self-Assessment Questions - 1

1. Framing is the process of breaking down a stream of bits into smaller portions called _________ .
2. A frame header contains ___________ .
3. is represented by older protocols such as BISYNC, PPP and DDCMP.
4. DDCMP is the abbreviation of __________ .
5. The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a ____________ protocol.
6. SDLC was later standardized by the ISO as High-Level Data Link Control (HDLC) protocol. State true or false.

   (a) True (b) False
7. Clock- based framing is represented by the ___________ standard.
8. SONET is the abbreviation of ___________.

3. ERROR DETECTION AND CORRECTION

Error detection and correction are the two major issues to be dealt with when transmitting data over a network. There are two basic strategies for dealing with errors of data transmission. Both the strategies add redundant information to the data that is sent. One strategy is to include enough redundant information to enable the receiver to derive what the transmitted data must have been. This strategy uses *error-correcting* codes. Another strategy is to include only enough redundancy to allow the receiver to find out that an error has occurred and have it request a retransmission. This strategy uses *error-detecting codes*. The use of error-correcting codes is often referred to as *FEC (Forward Error Correction)*. We will discuss both error-correcting codes and error-detecting codes in the following sections.

3.1 Error Correcting Codes

There are four different types of error correcting codes. They are: *Hamming codes*, *Binary convolutional codes*, *Reed-Solomon* codes and *Low-Density Parity Check Codes*. All of these codes add redundancy to the information that is sent.

Hamming Codes: A frame consists of 'm' data bits and 'r' redundant bits. In a *systematic code*, the *m* data bits are sent directly, along with the check bits, rather than being encoded themselves before they are sent. In a *linear code*, the *r* check bits are computed as a linear function of the *m* data bits. Let total length of a block be n *(n=m + r)*. This is known as *(n, m)* code. An *n-bit* unit containing data and check bits is referred to as an *n bit* codeword. The code rate, or simply rate, is the fraction of the codeword that carries information that is not redundant, or *m/n*.

Before discussing error correction, it is necessary to understand what error is. Consider two codewords that may be transmitted or received- Say, 10001001 and 10110001. From these codewords, it is possible to determine how many corresponding bits differ. In this case, 3 bits differ. To determine how many bits differ, XOR the two codewords and count the number of 1 bit in the result. That is,

10001001
<u>10110001</u>
00111000

The number of bit positions in which two codewords differ is called the ***Hamming distance***. Its significance is that if two codewords are a Hamming distance d apart, it will require single-bit errors to convert one into the other.

In case of a 4-digit data, number of redundant/parity bits required is 3 (This can be calculated using the equation $2^p \geq m+p+1$, where m is number of data bits and p is the number of parity bits). Consider an example of a 4-bit data, the code is composed of four information bits and three parity bits. The left-most bit is designated as bit 1, next is bit 2 and so on…

bit 1, bit 2, bit 3, bit 4, bit 5, bit 6, bit 7

In this example, the parity bits are located in the positions that are numbered corresponding to ascending powers of 2 (1, 2, 4, 8…) as shown in Table 3.1.

Table 3.1: Assignment of Parity Bit values

Bit Designation	P_1,	P_2,	M_1,	P_3,	M_2,	M_3,	M_4
Bit Position	1	2	3	4	5	6	7
Binary position number	001	010	011	100	101	110	111

Here, P_n designates a particular parity bit and M_n shows information bit. Note that, the binary position number of parity bit P_1 has a 1 for its right most digit. So, this parity bit checks for all bit positions, including itself that have 1's in the same location in the binary position numbers. Therefore, parity bit P1 checks bit positions 1, 3, 5 and 7. Similarly, binary position number of parity bit P2 has a 1 for its middle bit, it checks all bit positions including itself, that have 1s in this same position. Therefore, P_2 checks bits 2, 3, 6 and 7. Similarly P_3 is in binary position 4 and has 1 for the leftmost bit, so it checks bit position 4, 5, 6 and 7.

Let's discuss one example, consider the data 1001 (information bits) being transmitted using even parity. Number of parity bits required is 3 (Let p=3, then $2_p \geq m+p+1$, i.e., $2^3=8$ and m+p+1=4+3+1=8. So, three parity bits are sufficient) and total code bits = 4+3=7. Next step is to determine the parity bits. Bits P_1 checks bit positions 1, 3, 5 and 7 and must be a 0 for there to be an even number of 1s in this group. Bit P_2 checks bit positions 2, 3, 6 and 7 and must be a 0 for there to be an even number of 1s in this group. Bit P_3 checks bit positions 4, 5, 6 and 7 and must be a 1 for there to be an even number of 1s in this group. These bits are entered in table 3.2 and the resulting combined code is 0011001.

Table 3.2: Representation of bits in case of information bits 1001

Bit Designation	P_1	P_2	M_1	P_3	M_2	M_3	M_4
Bit Position	1	2	3	4	5	6	7
Binary Position No.	001	010	011	100	101	110	111
Information Bits			1		0	0	1
Parity bits	0	0		1			

Here, the codeword is 0011001. Suppose, there is an error occurred during transmission and the codeword received is 0010001. The receiver doesn't know what was transmitted and must look for proper parities to determine if the code is correct. Designate any error that has occurred in transmission if even parity is used. Table 3.3 shows the bit position table of received code.

Table 3.3: Bit Position table of Received code

Bit Designation	P_1	P_2	M_1	P_3	M_2	M_3	M_4
Bit Position	1	2	3	4	5	6	7
Binary Position No.	001	010	011	100	101	110	111
Received code	0	0	1	0	0	0	1

Bit P_1 checks positions 1, 3, 5 and 7. There are two 1s in this group, so parity check is good. Bit P_2 checks positions 2, 3, 6 and 7. There are two 1s in this group. Parity check is good. Bit P3 checks positions 4, 5, 6 and 7. There is one 1 in this group. Parity check is bad. So, the error position code is 100 (binary four). This says that bit in position 4 is in error. It is a 0 and should be a 1. The corrected code is 0011001, which agrees with the transmitted code.

Binary convolutional codes: In convolutional code, an encoder processes a sequence of input bits and generates a sequence of output bits. There is no natural message size or encoding boundary as in a block code. The output depends on the current and previous input bits. That is, the encoder has memory. The number of previous bits on which the output depends is called the constraint length of the code. Convolutional codes are specified in terms of their rate and constraint length. Figure 3.7 shows an example of convolutional code.

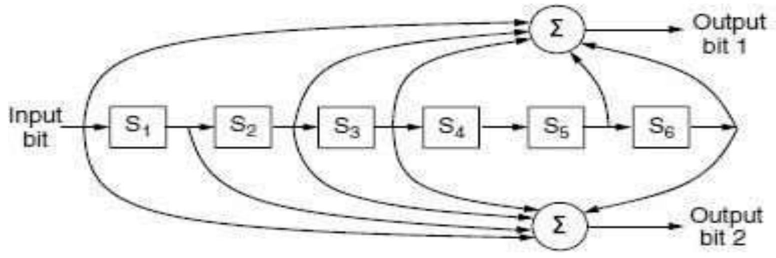

Figure 3.7: Example of Convolutional code

In figure 3.7, each input bit on the left-hand side produces two output bits on the right-hand side that are XOR sums of the input and internal state. Since it deals with bits and performs linear operations, this is a binary, linear convolutional code. Since one input bit produces two output bits, the code rate is ½. It is not systematic since none of the output bits is simply the input bit.

Reed-Solomon Code: This is the third kind of error correcting code. Like Hamming codes, Reed-Solomon codes are linear block codes, and they are often systematic too. Unlike Hamming codes, which operate on individual bits, Reed-Solomon codes operate on *m* bit symbols.

Reed-Solomon codes are based on the fact that every *n* degree polynomial is uniquely determined by n + 1 points. For example, a line having the form ax + b is determined by two points. Extra points on the same line are redundant, which is helpful for error correction. Imagine that we have two data points that represent a line and we send those two data points plus two check points chosen to lie on the same line. If one of the points is received in error, we can still recover the data points by fitting a line to the received points. Three of the points will lie on the line, and one point, the one in error, will not. By finding the line we correct the error.

Reed-Solomon codes are actually defined as polynomials that operate over finite fields, but they work in a similar manner. For m bit symbols, the codewords are $2m-1$ symbols long. A popular choice is to make m = 8 so that symbols are bytes. A codeword is then 255 bytes long. The 255 bytes codeword is widely used; it adds 32 redundant symbols to 233 data symbols so it is denoted as (255,233) code. Decoding and error correction is done with an algorithm developed by Berlekamp and Massey that can efficiently perform the fitting task for moderate-length codes. Reed-Solomon codes are widely used in practice because of their strong error-correction properties, particularly for burst errors. They are used for DSL, data over cable, satellite communications, CDs, DVDs, and Blu-ray discs.

Low-Density Parity Check (LDPC) code: LDPC codes are linear block codes that were invented by Robert Gallagher in his doctoral thesis. In an LDPC code, each output bit is formed from only a fraction of the input bits. This leads to a matrix representation of the code that has a low density of 1s, hence the name for the code. The received codewords are decoded with an approximation algorithm that iteratively improves on a best fit of the received data to a legal codeword. This algorithm corrects errors. LDPC codes are practical for large block sizes and have excellent error-correction abilities that outperform many other codes. For this reason, they are rapidly being included in new protocols. They are part of the standard for digital video broadcasting, 10 Gbps Ethernet, power-line networks, and the latest version of 802.11.

3.2 Error Detecting Codes

Error-correcting codes are used on wireless links, which are very noisy and error prone when compared to optical fibers. However, over fiber or high-quality copper, error rate is much lower, so error detection and retransmission are usually more efficient. There are three different error detecting codes. They are: *Parity, Checksums, and Cyclic Redundancy Checks (CRCs).*

Parity: Consider the first error-detecting code, in which a *single parity bit* is appended to the data. The parity bit is chosen so that the number of 1 bit in the codeword is even or odd. For example, when 1011010 is sent in even parity, a bit is added to the end to make it 10110100. With odd parity 1011010 becomes 10110101.

One difficulty with this scheme is that a single parity bit can only reliably detect a single bit error in the block. If the block is badly garbled by a long burst error, the probability that the error will be detected is only 0.5, which is hardly acceptable. The odds can be improved considerably if each block to be sent is regarded as a rectangular matrix n bits wide and k bit high. Now, if we compute and send one parity bit for each row, up to *k bit* errors will be reliably detected as long as there is at most one error per row.

Another method which provides better protection against burst errors is known as interleaving. In this method, we can compute the parity bits over the data in a different order than the order in which the data bits are transmitted. In this case, we will compute a parity bit for each of the n columns and send all the data bits as 'k' rows, sending the rows from top to bottom and the bits in each row from left to right in the usual manner. At the last row, we send the n parity bits. This transmission order is shown in figure 3.8 for n=7 and k=7.

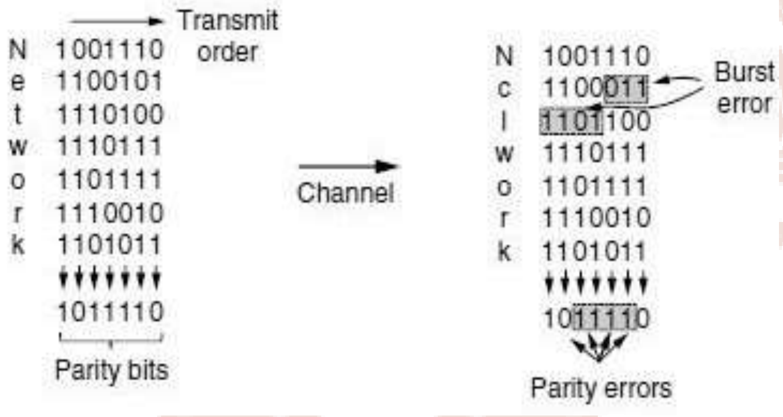

Figure 3.8: Interleaving of parity bits to detect a burst error.

Interleaving is a general technique to convert a code that detects isolated errors into a code that detects burst errors.

Checksum: The second kind of error detecting code, known as the checksum is closely related to groups of parity bits. Checksum means a group of check bits associated with a message. A group of parity bits is one example of a checksum. However, there are other, stronger checksums based on a running sum of the data bits of the message. The checksum is usually placed at the end of the message, as the complement of the sum function. This way, errors may be detected by summing the entire received codeword, both data bits and checksum. If the result comes out to be zero, no error has been detected.

Cyclic Redundancy Check (CRC): Third method of error detecting code is CRC which is also known as a polynomial code. This is a stronger kind of error-detecting code which is in widespread use at the link layer. Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only. A *k-bit* frame is regarded as the coefficient list for a polynomial with k terms, ranging from x^{k-1} *to* x^{0} . Such a polynomial is said to be of degree *k – 1*. The high-order (leftmost) bit is the coefficient of x^{k-1}, the next bit is the coefficient of $x^{k-2,}$ and so on.

For example, 110001 has 6 bits and thus represents a six-term polynomial with coefficients 1, 1, 0, 0, 0, and 1: $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0$. When the polynomial code method is employed, the sender and receiver must agree upon a *generator polynomial*, G(x), in advance. Both the high- and low order bits of the generator must be 1. To compute the CRC for some frame with m bits corresponding to the polynomial M(x), the frame must be

longer than the generator polynomial. The idea is to append a CRC to the end of the frame in such a way that the polynomial represented by the check summed frame is divisible by G(x). When the receiver gets the check summed frame, it tries dividing it by G(x). If there is a remainder, there has been a transmission error.

Self-Assessment Questions - 2

9. The use of error-correcting codes is often referred to as ___________ .
10. The number of bit positions in which two codewords differ is called_________ .
11. In _________ , an encoder processes a sequence of input bits and generates a sequence of output bits.
12. _____________are linear block codes that were invented by Robert Gallagher in his doctoral thesis.
13. Cyclic Redundancy Check is also known as ________ .

4. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- Framing is the process of breaking down a stream of bits into smaller portions called frames.
- Different protocols for addressing framing problems are: Byte-oriented Protocols, Bit-oriented Protocols, and clock-based framing.
- A byte-oriented approach is represented by older protocols such as the Binary Synchronous Communication (BISYNC) protocol, and the Digital Data Communication Message Protocol (DDCMP) used in Digital Equipment Corporation's DECNET.
- A bit-oriented protocol views the frame as a collection of bits.
- The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol.
- Clock-based framing is another approach to framing, and is represented by the Synchronous Optical Network (SONET) standard.
- Error detection and correction are the two major issues to be dealt with when transmitting data over a network.
- There are four different error correcting codes. They are: Hamming codes, Binary convolutional codes, Reed-Solomon codes and Low-Density Parity Check Codes.
- There are three different error detecting codes. They are: Parity, Checksums, Cyclic Redundancy Checks (CRCs).

5. TERMINAL QUESTIONS

1. Differentiate between Byte-oriented protocols and Bit-oriented protocols.
2. Write short notes on Clock-based Framing.
3. Describe Hamming codes in detail.
4. Compare Binary convolutional codes, Reed-Solomon codes and Low-Density Parity Check (LDPC) code.
5. Explain different error detecting codes.

6. ANSWERS

Self-Assessment Questions

1. Frames
2. Addressing information
3. Byte-oriented protocols
4. Digital Data Communication Message Protocol
5. Bit-oriented
6. (a) True
7. SONET
8. Synchronous Optical Network
9. Forward Error Correction
10. Hamming distance
11. Convolutional Code
12. LDPC (Low Density Parity Check)
13. Polynomial Code

Terminal Questions

1. Byte-oriented approach is represented by older protocols such as the Binary Synchronous Communication (BISYNC) protocol developed by IBM in the late 1960s, and the Digital Data Communication Message Protocol (DDCMP) used in Digital Equipment Corporation's DECNET. A bit-oriented protocol views the frame as a collection of bits. The Synchronous Data Link Control (SDLC) protocol developed by IBM is an example of a bit-oriented protocol; SDLC was later standardized by the ISO as the High-Level Data Link Control (HDLC) protocol. (Refer section 2.1 and 2.2 for more details).
2. Clock-based framing is another approach to framing, and is represented by the Synchronous Optical Network (SONET) standard. SONET was first proposed by Bell Communications Research (Bellcore), and then developed under the American National Standards Institute (ANSI) for digital transmission over optical fiber; it has since been adopted by the ITU-T. (Refer section 2.3 for more details)

3. A frame consists of 'm' data bits and 'r' redundant bits. In a *systematic code*, the *m* data bits are sent directly, along with the check bits, rather than being encoded themselves before they are sent. In a *linear code*, the *r check* bits are computed as a linear function of the *m data* bits. (Refer section 3.1 for more details).
4. In convolutional code, an encoder processes a sequence of input bits and generates a sequence of output bits. There is no natural message size or encoding boundary as in a block code. Like Hamming codes, Reed-Solomon codes are linear block codes, and they are often systematic too. Unlike Hamming codes, which operate on individual bits, Reed-Solomon codes operate on *m bit* symbols. LDPC codes are linear block codes that were invented by Robert Gallagher in his doctoral thesis. In an LDPC code, each output bit is formed from only a fraction of the input bits. (Refer section 3.1 for more details).
5. Error-correcting codes are used on wireless links, which are very noisy and error prone when compared to optical fibers. However, over fiber or high-quality copper, error rate is much lower, so error detection and retransmission are usually more efficient. There are three different error detecting codes. They are: *Parity, Checksums, and Cyclic Redundancy Checks.* (Refer section 3.2 for more details).

References:

- Andrew S Tanenbaum, David J. Wetherall, *"Computer Networks,"* Fifth edition.
- Larry L. Peterson, Bruce S. Davie, *"Computer Networks- a Systems Approach,"* Fifth edition.
- James F. Kurose, Keith W. Ross, *"Computer Networking-A top-down approach,"* Sixth edition.
- Behrouz A.Forouzan, Sophia Chung Fegan, *"Data Communication and Networking,"* Fourth edition.
- William Stallings, *"Computer Networking with Internet Protocols and Technology,"* Third edition.