9.4.2 Questions

Q9-1. Distinguish between communication at the network layer and communication at the data-link layer.

Q9-1. Communication at the network layer is *host-to-host*; communication at the data-link layer is *node-to-node*.

The Network layer provides communication using the host-to-host transformation of the data packets. The Data-link layer provides communication using the node-to-node transformation of the data packets.

The difference between the communication at the Network layer and at the Data-link layer is given below.

Difference between the communication at the Network layer and at the Data-link layer:

Network layer	Data-link layer
It provides the communication using host-to-host delivery of the data.	It provides the communication using node-to-node delivery of the data.
It sends and receives the information within the same network with the single LAN.	It sends and receives the information through the multiple LAN networks.
It does not divide the network into subnetworks.	It divides the data link layer into sublayers.
It provides Packetizing to encapsulate the data, routing, and forwarding the data.	It provides framing, flow and error control, and error detection and correction.
It provides connectionless communication between source and destination.	Using LAN's, it provides connectionless communication between source and destination.
It uses logical addressing.	It uses physical addressing.
The issues in the network layer are security and congestion control.	It deals with the issues of point-to-point and broadcast process.

Q9-2. Distinguish between a point-to-point link and a broadcast link.

Broadcast and Point to point network:

It is a method where multiple parties can hear a single signal while transmitting a signal.

In point to point network communication happens between two nodes.

Difference between Point to Point and Broadcast networks:

Point to Point is a direct connection between the individual users. A packet is a message which transmits from source location to destination location. A good route within the network is necessary to transmit the packet. In this network unicasting takes place where transmission happens between one sender and one receiver. In broadcast link communication is shared by all the systems which are connected to that network. In this network the message is sent by one system and it is shared by all the other systems.

Q9-3. Can two hosts in two different networks have the same link-layer address? Explain.

Q9-3. Two hosts in two different networks can theoretically have the same link-layer address because a link-layer address has only local jurisdiction. However, the tendency is to avoid this for the future development of the Internet. Even today, manufacturers of network interface cards (NIC) use different set of link-layer addresses to make them distinguished.

NIC (Network interface cards) uses different sets of link layer addresses to distinguish between the two or more host with different.

Yes, the two hosts in two different networks can have the same link layer address to send or receive datagram between them as the link layer address is local.

Q9-4. Is the size of the ARP packet fixed? Explain.

ARP:

• ARP stands for Address Resolution Protocol.

• This helps to find the IP address of the destination host or next hop.

• Whenever user sends a packet from source host to destination host, it requires to identify the way to deliver this packet to the next hop first. For this, IP packet refer its routing table.

• Since, data link layer services are used by IP to get physical address of the next hop which is done by a protocol, called ARP.

Hence, the size of ARP packet is not fixed because it is variable and depends on the length of the physical and logical addresses used.

Q9-5. What is the size of an ARP packet when the protocol is IPv4 and the hardware is Ethernet?

Q9-5. ARP Packet Size = 2 + 2 + 1 + 1 + 2 + 6 + 4 + 6 + 4 = 28 bytes (Figure 9. 8).

ARP fields:

Hardware type = 16 bits

Protocol type = 16 bits

Hardware length = 8 bits

Protocol length = 8 bits

Operation = 16 bits

Sender h/w address = 48 bits (Ethernet source address)

Sender protocol address = 32 bits (IPv4 source address)

Target h/w address = 48 bits (Ethernet destination address)

Target protocol address = 32 bits (IPv4 destination address)

Converting each field in ARP packet into bytes then:

2 + 2 + 1 + 1 + 2 + 6 + 4 + 6 + 4 = 28 bytes

The packet has 48-bit fields for the sender hardware address and target hardware address and 32-bit fields for the corresponding sender and target protocol addresses. Therefore combine all fields of the ARP packet size in this case is 28 bytes.

Q9-6. Assume we have an isolated link (not connected to any other link) such as a private network in a company. Do we still need addresses in both the network layer and the data-link layer? Explain.

Each hop (router or host) should know its own link-layer address. The destination link-layer address is determined by using the Address Resolution Protocol

---

Q9-7. In Figure 9.9, why is the destination hardware address all 0s in the ARP request message?

Q9-7. Station A does not know the link-layer address of station B yet. It uses an all-zero address to define that this address is desired.

Q9-8. In Figure 9.9, why is the destination hardware address of the frame from A to B a broadcast address?

Q9-9. In Figure 9.9, how does system A know what the link-layer address of system B is when it receives the ARP reply?

Q9-9. The source hardware address defines the link-layer address of station B.

Q9-10. When we talkabout the broadcast address in a link, do we mean sending a message to all hosts and routers in the link or to all hosts and routers in the Internet? In other words, does a broadcast address have a local jurisdiction or a universal jurisdiction? Explain.

Yes, broadcast means sending to all hosts and routers. Broadcast has a universal jurisdiction

---

Q9-11. Why does a host or a router need to run the ARP program all of the time in the background?

Q9-11. A host does not know when another host sends an ARP request; it needs to be ready all of the time to respond to an ARP request.

Because address of other router or hosts is not static, so whenever a address is changed, router needs to update it to send packets to correct address.

Q9-12. Why does a router normally have more than one interface?

A router, by nature, is connected to two or more network. It needs an interface (port) for each network it is connected to

---

Q9-13. Why is it better not to change an end-to-end address from the source to the destination?

Q9-13. If an end-to-end address is changed during the packet journey, it is not guarantee that the packet arrives at its destination.

Q9-14. How many IP addresses and how many link-layer addresses should a router have when it is connected to five links?

A router needs to have a MAC address (link layer address) and IP address on each network interface (link). Hence there are 5 IP addresses and 5 link layer address.

9.4.3 Problems

P9-1. Assume we have an internet (a private small internet) in which all hosts are connected in a mesh topology. Do we need routers in this internet? Explain.

P9-1. Theoretically, we do not need IP addresses because the global communication is one to one. If a station has a packet to send to another station, it uses the link-layer address of the destination host (or even port number related to the destination) to send a packet. However, if the internet uses the TCP/IP protocol suite, then messages pass through the network layer and IP address come to the picture.

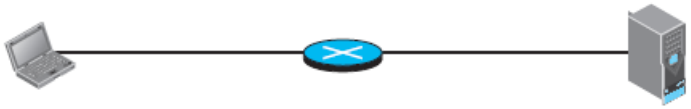
No.There is no need of routers in mesh topolgy.In mesh topology each node is connected to atleast two other nodes.This topology plays a vital role in internet.Network nodes can interact with each other directly without assistance of internet.One of the benefits of this topology is there is no centralised point so that no failure of system because even though one node fails the other nodes operates and communicates with each other through one or more intermediate nodes.Mesh topology is of two types full mesh and partial mesh topology where in full mesh topology each node in network is connected with all other nodes in network and in partial mesh topology one node can be connected with other nodes in network while others may connected to nodes from which the data is exchanged.

P9-2. In the previous problem, do we need both network and data-link layers?

No.As we are not using routers in our topolgy there is no need for both network and data link layers.Network layer involves the mechanism of addressing the packet to route to desination with the help of other layers such as data link ,physical layer.Data link layer is most good for node to node delivery of data. It forms frames from the packets of data that are received from network layer and gives it to physical layer. It also states the information which needs to be transmitted over the data. Error controlling is easy The data are then passed to physical.

P9-3. Explain why we do not need the router in Figure 9.15.

Figure 9.15 *Problem 9-3*

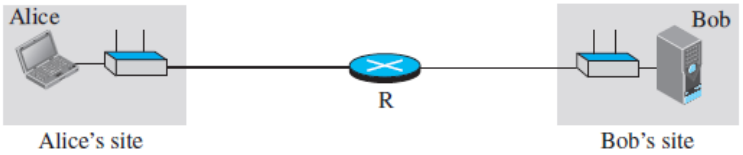


P9-3. A router is need when we have more than one paths for the packet to travel from the source to destination. In Figure 9.15 (in the text) there is only path in each direction. We need no router.

The main purpose of a router is to connect multiple networks and forward packets destined either for its own networks or other networks. Here in this figure only a laptop and a CPU needs to be connected which could be done through ethernet cable, there is no requirement of a router.or if its not a CPU, if it is a Server then also router is not needed as a router is required when multiple networks needs to be connected.

P9-4. Explain why we may need a router in Figure 9.16.

Figure 9.16 *Problem 9-4*



P9-5. Is the current Internet using circuit-switching or packet-switching at the data-link layer? Explain.

P9-5. The current Internet is using packet-switching at the data-link layer. The source divides the data at the data-link layer into frames and each frame is independent.

P9-6. Assume Alice is travelling from 2020 Main Street in Los Angeles to 1432 American Boulevard in Chicago. If she is travelling by air from Los Angeles Airport to Chicago Airport,

a. find the end-to-end addresses in this scenario.

b. find the link-layer addresses in this scenario.

a. End-to-end addresses (the whole journey)
Source: 2020 Main Street, Los Angeles
Destination: 1432 American Boulevard, Chicago

b. **First Link**
Source: 2020 Main Street
Destination: Los Angeles Airport
Second Link
Source: Los Angeles Airport
Destination: Denver Airport
Third Link
Source: Denver Airport
Destination: Chicago Airport
Fourth Link
Source: Chicago Airport
Destination: 1432 American Boulevard

P9-7. In the previous problem, assume Alice cannot find a direct flight from the Los Angeles to the Chicago. If she needs to change flights in Denver,

a. find the end-to-end addresses in this scenario.

b. find the link-layer addresses in this scenario.

P9-7. We can think of one journey with four links in this case: home-to-airport, airport-to-airport, and airport-to-home

a. **End-to-end addresses** (the whole journey)
Source: 2020 Main Street, Los Angeles
Destination: 1432 American Boulevard, Chicago

b.

First Link
Source: 2020 Main Street
Destination: Los Angeles Airport

Second Link
Source: Los Angeles Airport
Destination: Denver Airport

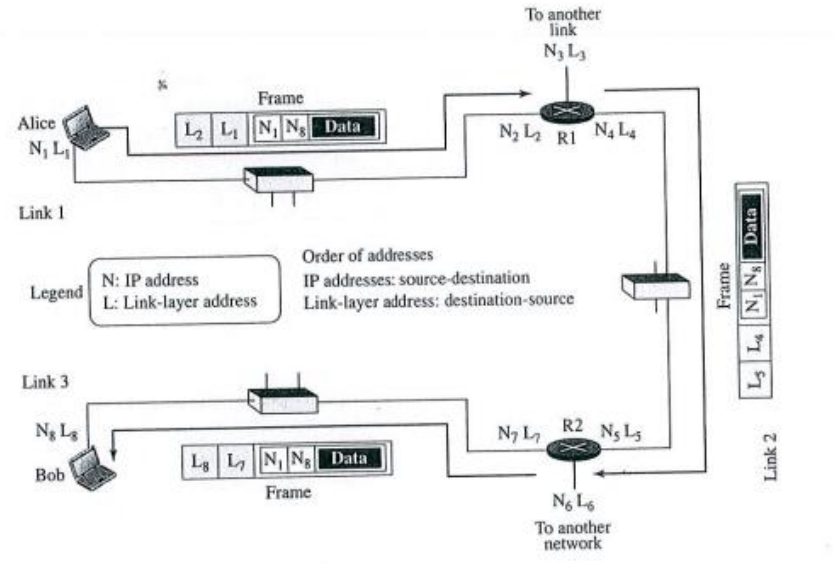
Third Link
Source: Denver Airport
Destination: Chicago Airport

Fourth Link
Source: Chicago Airport
Destination: 1432 American Boulevard

P9-8. When we send a letter using the services provided by the post office, do we use an end-to-end address? Does the post office necessarily use an end-to-end address to deliver the mail? Explain.

P9-9. In Figure 9.5, assume Link 2 is broken. How can Alice communicate with Bob?

Figure 9.5 *IP addresses and link-layer addresses in a small internet*



P9-9. The communication is impossible unless router R1 can reach router R2 using another path (not shown in the figure).

P9-10. In Figure 9.5, show the process of frame change in routers R1 and R2.

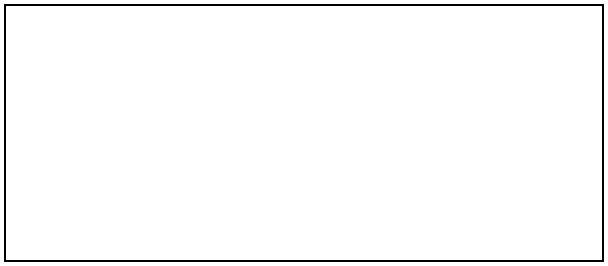
P9-11. In Figure 9.7, assume system B is not running the ARP program. What would happen?

P9-11. The packet cannot be delivered unless system A broadcast it and system B receive it. In this case, all stations receive the packet. Other stations should drop it.

P9-12. In Figure 9.7, do you think that system A should first check its cache for mapping from N2 to L2 before even broadcasting the ARP request?

P9-13. Assume the network in Figure 9.7 does not support broadcasting. What do you suggest for sending the ARP request in this network?

P9-13. Two approaches can be used. In the first approach, system A has a table to match the network-layer addresses to data-link addresses, it can use the table to find the data-link address of system B. In the second approach, system A has only the list of all data-link layer addresses, it can send unicast ARP packet to all stations to find out the one which matches the network-layer address. None of the approaches are practical because a host may change its data-link layer address without notice (by changing NIC as we see in Chapter 13). Some networks support tunneling, in which the network encapsulates a broadcast or multicast packet in a unicast packet and send them to all stations.



P9-14. In Figures 9.11 to 9.13, both the forwarding table and ARP are doing a kind of mapping. Show the difference between them by listing the input and output of mapping for a forwarding table and ARP.

The differences between input and output of all three figures are listed below:

Figure 9.11:
Input : Na Nb Data

Output L1 La Na Nb Data

Figure 9.12:
Input : L1 La Na Nb Data

Output L3 L2 Na Nb Data

Figure 9.13:
Input : L3 L2 Na Nb Data

Output Lb L4 Na Nb Data

P9-15. Figure 9.7 shows a system as either a host or a router. What would be the actual entity (host or router) of system A and B in each of the following cases:

a. If the link is the first one in the path?

b. If the link is the middle one in the path?

c. If the link is the last one in the path?

d. If there is only one link in the path (local communication)?

P9-15.

a. A: host B: router

b. A: router B: router

c. A: router B: host

d. A: host B: host

Q10-1. How does a single-bit error differ from a burst error?

Q10-1. In a *single-bit error* only one bit of a data unit is corrupted; in a *burst error* more than one bit is corrupted (not necessarily contiguous).

Single bit error:

Only one bit of the data is corrupted in a single bit error.

Burst error:

In burst error more than one bit is corrupted.

Q10-2. What is the definition of a linear block code?

Definition for Linear block code:

It is a block code in which exclusive-OR of any two codewords creates another codeword.

Definition for cyclic code:

It is a linear code with one extra property in which the cyclic shifting (rotation) of each codeword creates another codeword.

Q10-3. In a block code, a dataword is 20 bits and the corresponding codeword is 25 bits. What are the values of k, r, and n according to the definitions in the text? How many redundant bits are added to each dataword?

Q10-3. In this case, $k = 20$, $r = 5$, and $n = 20$. Five redundant bits are added to the dataword to create the corresponding codeword.

Given n=25, k=20

To find the redundant bits we have a formula which is shown below,

$$r = n - k$$

The redundant bits that needs to be added is,

$$r = n - k$$
$$r = 25 - 20$$
$$r = 5 \text{ bits}$$

We need to add 5 redundant bit to each data word.

Q10-4. In a codeword, we add two redundant bits to each 8-bit data word. Find the number of

a. valid codewords. **b.** invalid codewords.

In block coding process, k bits in data word makes 2^k possible combinations. By adding r bits as the redundant length then code word n becomes $k+r$, with this it is possible to have $2^{(n)}$ combinations to code word.

As block coding process is one to one, only 2^k code words are valid and 2^n-2^k code words will be invalid or illegal.

With length of data word k is 8 and no of redundant bits r is 2, considering above explanation

No of possible data words $2^8 = 256$

Length of code word n becomes 10 i.e. 8+2

Number of possible code words $2^{10} = 1024$

Number of **Valid code words** 2^8 = **256**

Number of **Invalid code words** 2^{10}-2^8 =**768**

Q10-5. What is the minimum Hamming distance?

Q10-5. The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Hamming distance: The hamming distance between two words is the number of differences between corresponding bits.

Example: If v_1=011011 and v_2=110001 then d (v_1, v2) = 3

Minimum Hamming distance: The minimum Hamming distance is the smallest hamming distance between all possible pairs in a set of words.

Example: If v_1=011011 and v_2=110001 then d (v_1, v2) = 3 and

If v_1=011011 and v_2=011000 then d (v_1, v2) = 2

So the minimum Hamming distance between two code words is 2, the Hamming distance of the code is 2.

Q10-6. If we want to be able to detect two-bit errors, what should be the minimum Hamming distance?

Hamming distance is the shortest distance between two equal length data bits, in which the corresponding data bits differ. This can be achieved by XOR the two data bits and measuring number of 1s in result.

The minimum hamming distance is s+1. Where s is number of errors can occur during transmission.

For example: Codeword 1= 0110001 and

Codeword2= 1010100

Now C1 XOR C2 is 1100001 in this case Hamming distance is 3.

Is 4 not 3

Q10-7. A category of error detecting (and correcting) code, called the ***Hamming code,*** is a code in which $d_{min} = 3$. This code can detect up to two errors (or correct one single error). In this code, the values of n, k, and r are related as: $n = 2^r - 1$ and $k = n - r$. Find the number of bits in the dataword and the codewords if r is 3.

Q10-7. We have $n = 2^r - 1 = 7$ and $k = n - 3 = 7 - 3 = 4$. A dataword has four bits and a codeword has seven bits. Although it is not asked in the question, we give the datawords and valid codewords below. Note that the minimum distance between the two valid codewords is 3.

Data	*Code*	*Data*	*Code*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

Let n = code word bits

r= redundant bits

k=data word

$n=2^r-1$

k=n-r

Here the value of r=3

$n= 2^3-1$

n=8-1

n=7

k=7-3=4

Hence data word bits = 4

Code word bits =7.

Q10-8. In CRC, if the dataword is 5 bits and the codeword is 8 bits, how many 0s need to be added to the dataword to make the dividend? What is the size of the remainder? What is the size of the divisor?

The size of the dataword = 5 bits

The size of the codeword = 8 bits

The size of the code word and dividend are same. Hence, three 0's need to be added to the dataword to make the dividend.

$$\begin{aligned}\text{The size of the remainder} &= \text{size of the codeword - size of the dataword}\\ &= 8 - 5\\ &= 3\end{aligned}$$

$$\begin{aligned}\text{The size of the divisor} &= \text{size of the remainder} + 1\\ &= 3 + 1\\ &= 4\end{aligned}$$

Therefore, the size of the remainder and size of the divisor are 3 and 4 respectively.

Q10-9. In CRC, which of the following generators (divisors) guarantees the detection of a single bit error?

a. 101 **b.** 100 **c.** 1

Q10-9.

a. The generator has three bits (more than required). Both the rightmost bit and leftmost bits are 1s; it can detect all single-bit errors.

b. This cannot be used as a generator: the rightmost bit is 0.

c. This cannot be used as a generator; it has only one bit.

a)

101

First let us convert the bits into polynomials

Then 101 becomes x^2+1

So if you divide any polynomial of x^n /x^2+1 then always will get the remainder.

b)

100

Again converting into polynomial

It becomes x^3

Any polynomial of x^n/x^3 =0 when n>=3

That means x^n is divisible by x^3

c)

1

Now convert to polynomial it becomes x^0

$X^0=1$

Any polynomial of $x^n/1=x^n$

Q10-10. In CRC, which of the following generators (divisors) guarantees the detection of an odd number of errors?

a. 10111 **b.** 101101 **c.** 111

Odd number of errors can be detected by using generator evenly divided by binary 11

a)

10111_2 cannot be divided by 11_2

Convert the binary numbers into decimals (for better understanding)

Then 10111_2 becomes 23

And 11_2 becomes 3

It is understood that if you divide 23/3

The remainder is 2 because 3 is a prime number.

b)

101101_2 can be divided by 11_2

Similarly 101101_2 are 45 in decimals.

And 11 are 3 in decimals.

45/3 = 15.

The remainder is 0.

c)

111_2 cannot be divided by 11_2

Here 111_2 is 7 in decimals.

11_2 is 3.

The remainder is 1.

Hence 101101 can detect odd number of errors.

Q10-11. In CRC, we have chosen the generator 1100101. What is the probability of detecting a burst error of length

a. 5? **b.** 7? **c.** 10?

Q10-11. In this case $r = 7 - 1 = 6$.

a. The length of the error is L = 5, which means $L \leq r$. All burst errors of this size will be detected.

b. The length of the error is L = 7, which means $L = r + 1$. This CRC will detect all burst errors of this size with the probability $1 - (0.5)^5 \approx 0.9688$. Almost 312 out of 10,000 errors of this length may be passed undetected.

c. The length of the error is L = 10, which means $L > r$. This CRC will detect all burst errors of this size with the probability $1 - (0.5)^6 \approx 0.9844$. Almost 156 out of 10,000 errors of this length may be passed undetected. Although the length of the burst error is increased, the probability of errors being passed undetected is decreased.

By selecting length of generator as R (where length of crc will be R-1) .

With this is it possible to detect all errors if L <= r where L is no of error.

If L = r+1 then probability is $1-0.5^{r-1}$

If L > r+1 then probability is $1-0.5^{r}$

A)

r is 6 and L is 5 hence it possible to detect all errors.

B)

r is 6 and L is 7 hence probability is 0.96875

C)

r is 6 and L is 10 hence probability is 0.984375

Q10-12. Assume we are sending data items of 16-bit length. If two data items are swapped during transmission, can the traditional checksum detect this error? Explain.

The drawback of traditional checksum is it cannot detect error if you change the order of data items, or swapped. Because the traditional checksum only adds all the data items and the results are complimented.

So in which ever order the data item is it cannot defect the error.

Q10-13. Can the value of a traditional checksum be all 0s (in binary)? Defend your answer.

Q10-13. The value of a checksum can be all 0s (in binary). This happens when the value of the sum (after wrapping) becomes all 1s (in binary).

Q10-14. Show how the Fletcher algorithm (Figure 10.18) attaches weights to the data items when calculating the checksum.

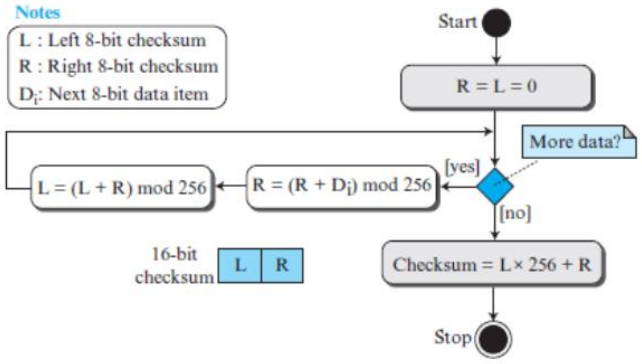


Figure 5.18 *Algorithm to calculate an 8-bit Fletcher checksum*

In Fletcher algorithm weights of the data is attached by using the position of the data.

There are two types of algorithms 8bit and 16bit.

8 bit contains 16 bit checksum.

16 bit contains 32 bit checksum.

This algorithm uses modulo 256 (2^8), where data is divided by 256 for 8bit 65536 for 16bit and reminder is kept. It uses two accumulators L and R for calculating checksum.

First data added to content of R then this will be divided by 256 or 65536 based on checksum length to get reminder kept and use it as (R). Then add R to L and calculate modulo 256 and use it as L, repeat this for all the data. once all the data is completed multiply L with 256 or 65536 and add R to it to get final checksum value

In this process each data is added with a weight which is associated with it is position.

Q10-15. Show how the Adler algorithm (Figure 10.19) attaches weights to the data items when calculating the checksum.

Q10-15. The following shows that L is the weighted sum of the data items.

Beginning	$R = 1$	$L = 0$
Iteration 1	$R = 1 + D_1$	$L = L + R = 0 + 1 + D1$
Iteration 2	$R = R + D_2 = 1 + D_1 + D_2$	$L = L + R = 2 + 2D_1 + D_2$
…	…	…
Iteration n	$R = 1 + D_1 + D_2 + \ldots + D_n$	$L = n + nD_1 + \ldots + D_n$

10.7.3 Problems

P10-1. What is the maximum effect of a 2-ms burst of noise on data transmitted at the following rates?

a. 1500 bps **b.** 12 kbps **c.** 100 kbps **d.** 100 Mbps

P10-1. We have (vulnerable bits) = (data rate) × (burst duration). The last example shows how a noise of small duration can affect a large number of bits if the data rate is high.

a. vulnerable bits = $(1500) \times (2 \times 10^{-3})$ = 3 bits

b. vulnerable bits = $(12 \times 10^{3}) \times (2 \times 10^{-3})$ = 24 bits

c. vulnerable bits = $(100 \times 10^{3}) \times (2 \times 10^{-3})$ = 200 bits

d. vulnerable bits = $(100 \times 10^{6}) \times (2 \times 10^{-3})$ = 200,000 bits

a)

Given data:

Data rate = 1500bps

Burst duration = 2-ms

Maximum effected bits = (data rate) × (burst duration)

= $(1,500) \times (2 \times 10^{-3})$

= 3bits

b)

Given data:

Data rate = 12kbps

Burst duration = 2-ms

Maximum effected bits = (data rate) × (burst duration)

= $(12 \times 10^{3}) \times (2 \times 10^{-3})$

= 24bits

c)

Given data:

Data rate = 100kbps

Burst duration = 2-ms

Maximum effected bits = (data rate) × (burst duration)

= $(100\times10^{3})\times(2\times10^{-3})$

= 200bits

d)

Given data:

Data rate = 100Mbps

Burst duration = 2-ms

Maximum effected bits = (data rate) × (burst duration)

= $(100\times10^{6})\times(2\times10^{-3})$

= 200000bits

Note: It shows how a noise of small duration can affect so many bits if the data rate is high.

P10-2. Assume that the probability that a bit in a data unit is corrupted during transmission is p. Find the probability that x number of bits are corrupted in an n-bit data unit for each of the following cases.

a. $n = 8, x = 1, p = 0.2$

b. $n = 16, x = 3, p = 0.3$

c. $n = 32, x = 10, p = 0.4$

Given: the chance that one bit gets corrupted during a transmission is 'p'.

a) Given n=8, x=1, p=0.2;

Therefore the probability that 1 bit error occurs during the transmission of n bits of data unit is similar to the probability of occurrence of 'k' successes (corruption of a bit) during 'n' trials (transmission of n bits) and the probability of success is given by 'p':

= 0.336

• Therefore the probability of a bit getting corrupted is 0.336

b) Given n=16,x=3,p=0.3;

Therefore the probability that 1 bit error occurs during the transmission of n bits of data unit is similar to the probability of occurrence of 'k' successes (corruption of a bit) during 'n' trials (transmission of n bits) and the probability of success is given by 'p':

$$\frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

$$\frac{16!}{3!(13)!} 0.3^3 (1-0.3)^{16-3} = 0.14$$

• Therefore the probability of 3 bits getting corrupted is 0.14

c) Given n=32,x=10,p=0.4;

Therefore the probability that 1 bit error occurs during the transmission of n bits of data unit is similar to the probability of occurrence of 'k' successes (corruption of a bit) during 'n' trials (transmission of n bits) and the probability of success is given by 'p':

$$\frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

$$\frac{32!}{10!(22)!} 0.4^{10} (1-0.4)^{32-10} = 0.084$$

• Therefore the probability of 10 bits getting corrupted is 0.084

P10-3. Exclusive-OR (XOR) is one of the most used operations in the calculation of codewords. Apply the exclusive-OR operation on the following pairs of patterns. Interpret the results.

a. (10001) ⊕ (10001) **b.** (11100) ⊕ (00000) **c.** (10011) ⊕ (11111)

P10-3. The following shows the results. In the interpretation, 0 means a word of all 0 bits, 1 means a word of all 1 bits, and ~X means the complement of X.

a. `(10001) ⊕ (10001) = (00000)` **Interpretation: X ⊕ X → 0**

b. `(11100) ⊕ (00000) = (11100)` **Interpretation: X ⊕ 0 → X**

c. `(10011) ⊕ (11111) = (01100)` **Interpretation: X ⊕ 1 → ~X**

According to the XOR table below, if only one of the data word is '1' then the result will be '1' or else the data word will be '0' in other case.

0	0	0
0	1	1
1	0	1
1	1	0

a)

(10001) ⊕ (10001)

```
    1 0 0 0 1
 ⊕  1 0 0 0 1
--------------
    0 0 0 0 0
--------------
```

The result is 00000

b)

$(11100) \oplus (00000)$

```
    1 1 1 0 0
 ⊕  0 0 0 0 0
 -------------
    1 1 1 0 0
 -------------
```

The result is 11100

c)

$(10011) \oplus (11111)$

```
    1 0 0 1 1
 ⊕  1 1 1 1 1
 -------------
    0 1 1 0 0
 -------------
```

The result is 01100

P10-4. In Table 10.1, the sender sends dataword 10. A 3-bit burst error corrupts the codeword. Can the receiver detect the error? Defend your answer.

Assume the sender encodes the dataword 10 as 101 and sends it to the receiver. This codeword will be changed to 010 if a 3-bit burst error occurs. This pattern is not one of the valid codewords. So, the receiver detects the error and discards the received pattern.

P10-5. Using the code in Table 10.2, what is the dataword if each of the following codewords is received?

a. 01011 **b.** 11111 **c.** 00000 **d.** 11011

P10-5. Answers are given below:

a. error **b.** error **c.** 0000 **d.** 1101

Block coding:

- In block coding message is divided into blocks.
- The size of each block is k bits.
- These blocks are known as **Datawords**.
- For each block, r redundant bits are added.
- After adding redundant bits block length is n = k + r
- The n bits block is known as **Codewords**.

Process of finding dataword from the given codeword from the table given below:

- Codewords are generally used for error detection.
- The sender encodes the Dataword with a Codeword and sends it to the receiver.
- The receiver receives the Codeword and checks whether the Codeword is present in the table or not.
- If the received Codeword is not in the table, it means an error has occurred.
- Assume that there is only one bit error, implement the following strategy.

o Compare the received Codeword with each Codeword in the table.

o If any difference between the two Codewords is one, then the corresponding Codeword is Dataword.

o If the difference between the Codeword is more than one, then it is an error.

Given data:

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

a)

Received codeword is 01011

• Compare the received codeword 01011 with other codewords in the table.

• The received codeword matches with the codeword in the second row.

• Therefore, the corresponding dataword is 01.

b)

Received codeword is 11111

• Compare the received codeword 11111 with the other codewords in the table.

• The codeword in the fourth row 11110 and the received codeword 11111 have a difference of one bit.

• The received Codeword matches with Codeword in the fourth row.

• Therefore, the corresponding dataword is 11.

c)

Received codeword is 00000

• Compare the received codeword 00000 with other codewords in the table.

• The received codeword matches with the codeword in the first row.

• Therefore, the corresponding dataword is 00.

d)

Received codeword is 11011

• Compare the received codeword 11011 with other codewords in the table.

• The received codeword does not match with the any codeword in the table.

• The difference between the received codeword and the codewords in the table is more than one.

• Therefore, the received codeword is an error.

P10-6. Prove that the code represented by the following codewords is not linear. You need to find only one case that violates the linearity.

{(00000), (01011), (10111), (11111)}

A code is said to be linear when the XOR operation between any two codeword's in the given code results in a valid codeword belonging to the code.

• To find the linearity of the code, an XOR operation is performed on any two codeword's from the given code:

Given code :{ 00000, 01011, 10111, 11111}

01011

XOR 10111

11100

• Therefore as the result is not a valid codeword or in other words the result is not belonging to the set of codeword's given, the given code is not linear.

P10-7. What is the Hamming distance for each of the following codewords?

a. d (10000, 00000)
b. d (10101, 10000)
c. d (00000, 11111)
d. d (00000, 00000)

P10-7. The following shows the result. Part d shows that the Hamming distance between a word and itself is 0.

a. d (10000, 00000) = 1
b. d (10101, 10000) = 2
c. d (00000, 11111) = 5
d. d (00000, 00000) = 0

a) Given codeword d(10000,00000)

Apply the exclusive-OR operation to find hamming distance of the given codeword. The exclusive-OR operation is:

$$\begin{array}{r} 10000 \\ \underline{00000} \\ \oplus\underline{10000} \end{array}$$

After performing exclusive-OR operation, we get result (10000) and then we identify number of one's in that result is treated as a hamming distance. Here we have only 1 one in this result. So, the hamming distance of this codeword is 1.

$$\boxed{d(10000,00000)=1}$$

b) Given codeword d (10101, 10000)

Apply the exclusive-OR operation to find hamming distance of the given codeword. The exclusive-OR operation is:

$$\begin{array}{r} 10101 \\ \underline{10000} \\ \oplus\underline{00101} \end{array}$$

After performing exclusive-OR operation, we get result (00101) and then we identify number of one's in that result is treated as a hamming distance. Here we have 2 one's in this result. So, the hamming distance of this codeword is 2.

$$\boxed{d(10101,10000)=2}$$

c) Given codeword d(11111,11111)

Apply the exclusive-OR operation to find hamming distance of the given codeword. The exclusive-OR operation is:

$$\begin{array}{r} 11111 \\ \underline{11111} \\ \oplus\underline{00000} \end{array}$$

After performing exclusive-OR operation, we get result (00000) and then we identify number of one's in that result is treated as a hamming distance. Here we don't have 1's in this result. So, the hamming distance of this codeword is 0.

$$\boxed{d(11111,11111)=0}$$

d) Given codeword d (000,000)

Apply the exclusive-OR operation to find hamming distance of the given codeword. The exclusive-OR operation is:

$$\begin{array}{r} 000 \\ \underline{000} \\ \oplus\underline{000} \end{array}$$

After performing exclusive-OR operation, we get result (000) then we identify number of one's in that result is treated as a hamming distance. Here we don't have 1's in this result. So, the hamming distance of this codeword is 0.

$$\boxed{d(000,000)=0}$$

P10-8. Although it can be formally proved that the code in Table 10.3 is both linear and cyclic, use only two tests to partially prove the fact:

a. Test the cyclic property on codeword 0101100.

b. Test the linear property on codewords 0010110 and 1111111.

Given data:

Datawords	Codewords	Datawords	Codewords
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

a) Cyclic property: In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

For given codeword is 0101100 and we cyclically left-shift, then 0010110 is also a codeword.

b) Linear property: In linear code, the exclusive-OR of any two valid codewords creates valid codeword.

For given codewords 0010110 and 1111111 then perform exclusive-OR operation on these codes words below:

$$\begin{array}{r} 0010110 \\ 1111111 \\ \oplus\ 1101001 \end{array}$$

Therefore, the linear property is satisfied because we performing exclusive-OR operation on given codes then we can create given codeword.

P10-9. Referring to the CRC-8 in Table 5.4, answer the following questions:

a. Does it detect a single error? Defend your answer.

b. Does it detect a burst error of size 6? Defend your answer.

c. What is the probability of detecting a burst error of size 9?

d. What is the probability of detecting a burst error of size 15?

P10-9. The CRC-8 is 9 bits long, which means $r = 8$.

a. It has more than one bit and the rightmost and leftmost bits are 1s; it can detect a single-bit error.

b. Since $6 \leq 8$, a burst error of size 6 is detected.

c. Since $9 = 8 + 1$, a burst error of size 9 is detected most of the time; it may be left undetected with probability $(1/2)^{r-1}$ or $(1/2)^{8-1} \approx 0.008$.

d. Since $15 > 8 + 1$, a burst error of size 15 is detected most of the time; it may be left undetected with probability $(1/2)^{r}$ or $(1/2)^{8} \approx 0.004$.

P10-10. Assuming even parity, find the parity bit for each of the following data units.

a. 1001011 **b.** 0001100 **c.** 1000000 **d.** 1110111

a) Given data unit is 1001011

Assume even parity then find the even parity. It is simpler to count the number of 1s and make them even by adding a 0 or a 1.

Here the number of ones in given data unit is 4. So, the parity bit 0 and codeword is 01001011.

Therefore, the parity bit for 1001011 is 0.

b) Given data unit is 0001100

Assume even parity then find the even parity. It is simpler to count the number of 1s and make them even by adding a 0 or a 1.

Here, the number of ones in given data unit is 2. So, the parity bit 0 and codeword is 00001100.

Therefore, the parity bit for 0001100 is 0.

c) Given data unit is 1000000

Assume even parity then find the even parity. It is simpler to count the number of 1s and make them even by adding a 0 or a 1.

Here the number of ones in given data unit is 1. So, add one more one to data unit to make even parity. Therefore, the parity bit 1 and codeword is 1000000.

Therefore, the parity bit for 1000000 is 1.

d) Given data unit is 1110111

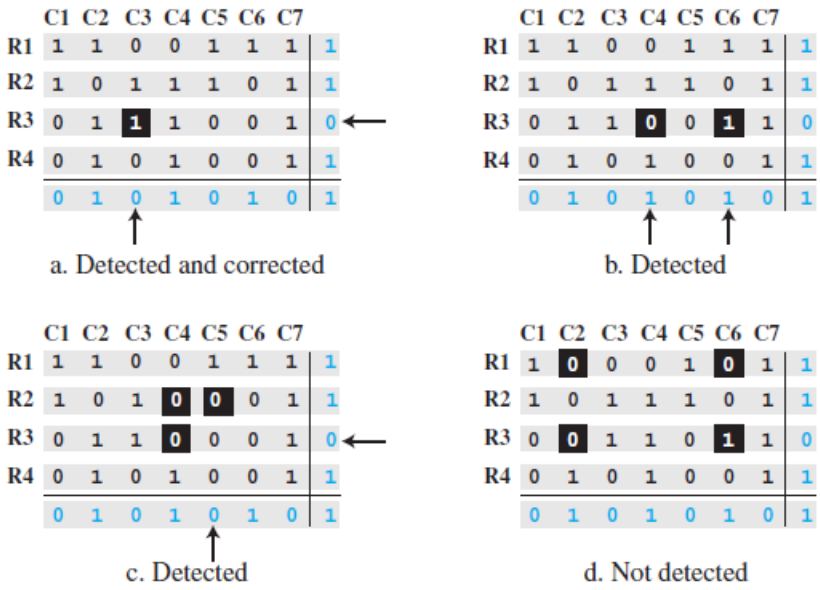
Assume even parity then find the even parity. It is simpler to count the number of 1s and make them even by adding a 0 or a 1.

Here, the number of ones in given data unit is 6. So, the parity bit 0 and codeword is 01110111

Therefore the parity bit for 1110111 is 1.

P10-11. A simple parity-check bit, which is normally added at the end of the word (changing a 7-bit ASCII character to a byte), cannot detect even numbers of errors. For example, two, four, six, or eight errors cannot be detected in this way. A better solution is to organize the characters in a table and create row and column parities. The bit in the row parity is sent with the byte, the column parity is sent as an extra byte (Figure 10.23).

Figure 10.23 *P10-11*



Show how the following errors can be detected:

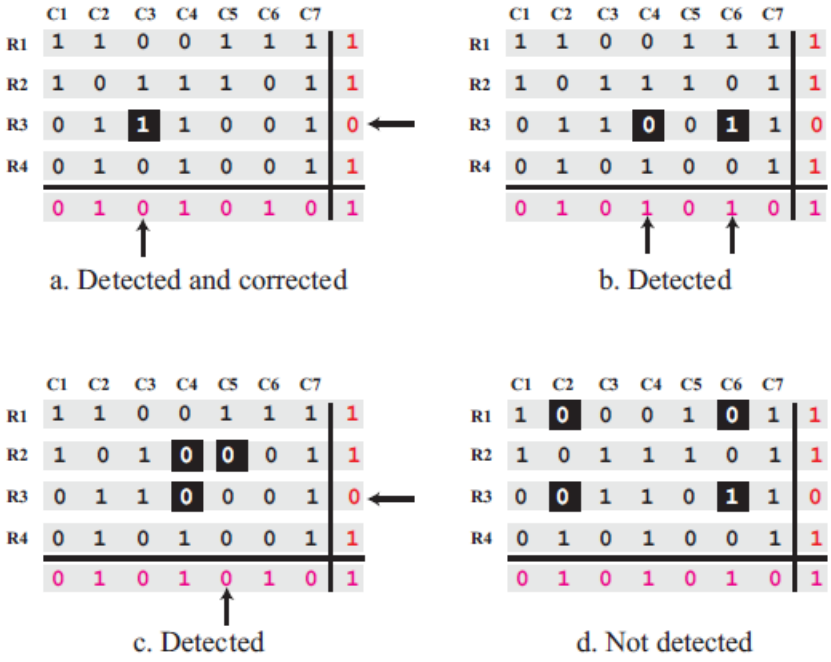
a. An error at (R3, C3).

b. Two errors at (R3, C4) and (R3, C6).

c. Three errors at (R2, C4), (R2, C5), and (R3, C4).

d. Four errors at (R1, C2), (R1, C6), (R3, C2), and (R3, C6).

P10-11. The following shows the errors and how they are detected.



a. In the case of one error, it can be detected and corrected because the two affected parity bits can define where the error is.

b. Two errors can definitely be detected because they affect two bits of the column parity.The receiver knows that the message is somewhat corrupted (although not where). It discards the whole message.

c. Three errors are detected because they affect two parity bits, one of the column parity and one of the row parity. The receiver knows that the message is somewhat corrupted (although not where). It discards the whole message.

d. The last case cannot be detected because none of the parity bits are affected.

a)

From the figure 5.88

An error at (R3, C3)

When there is corruption at R3, C3 checksum for row 3 and column 3 will not match, with this it is possible to find the corruption.

b)

Two errors at (R3, C4) and (R3, C6)

When there is an error in R3, C4 and R3, C6, here after corruption in Row3 parity is still same but parity for column 4 and column 6 will not match with this it is possible to find the corruption.

c)

Three errors at (R2, C4), (R2, C5), and (R3, C4)

With the errors in R3, C4 and R2, C4 parity for column will not corrupt. But this can be found by corruption parity of Row 2 and row 3 as described below.

With the errors in R2, C4 and R2, C5 parity for Row 2 will corrupt.

With the errors in R3, C4 parity for Row 3 will corrupt.

d)

Four errors at (R1, C2), (R1, C6), (R3, C2), and (R3, C6)

It is not possible to detect these errors all these errors will not have any impact parity

P10-12. Given the dataword 101001111 and the divisor 10111, show the generation of the CRC codeword at the sender site (using binary division).

Data word is 101001111

Divisor or polynomial 10111

In this case length of codeword will be length of data word + length of divisor -1

To calculate CRC for the Data 1010001111 first we need to add 4 zeros (length of divisor -1)

For ease of calculating CRC we will convert data in to polynomial representation.Give weight age to the 1s in data from left to right starting from 0 and increment by 1.

1010011110000 will become $x^{12}+x^{10}+x^7+x^6+x^5+x^4$

10111 will become $x^4+x^2+x^1+1$

As we know the multiplication $X^y * X^z = X^{(y+z)}$

Now we divide $x^{12}+x^{10}+x^7+x^6+x^5+x^4$ with $x^4+x^2+x^1+1$

$x^4+x^2+x^1+1)\ x^{12}+x^{10}+ x^7+x^6+x^5+x^4\ (x^8+x^5+x^4+x^2+x^1+x^0$ i.e. $x^0 = 1$

$x^{12}+x^{10}+x^9+x^8$

$x^9+x^8+x^7+x^6+x^5+x^4$

$x^9+ x^7+x^6+x^5$

$x^8+ x^4$

$x^8+ x^6+x^5+x^4$

---

x^6+x^5

$x^6+ x^4+x^3+x^2$

$x^5+x^4+x^3+x^2$

$x^5+ x^3+x^2+x^1$

$x^4 + x^1$

$x^4 + x^2 + x^1 + x^0$ i.e. $x^0 = 1$

$x^2 + 1 = 101$

Sender side crc code will be 0101

P10-13. Apply the following operations on the corresponding polynomials:

a. $(x^3 + x^2 + x + 1) + (x^4 + x^2 + x + 1)$

b. $(x^3 + x^2 + x + 1) - (x^4 + x^2 + x + 1)$

c. $(x^3 + x^2) \times (x^4 + x^2 + x + 1)$

d. $(x^3 + x^2 + x + 1) / (x^2 + 1)$

P10-13.

a. $(x^3 + x^2 + x + 1) + (x^4 + x^2 + x + 1) = x^4 + x^3$

b. $(x^3 + x^2 + x + 1) - (x^4 + x^2 + x + 1) = x^4 + x^3$

c. $(x^3 + x^2) \times (x^4 + x^2 + x + 1) = x^7 + x^6 + x^5 + x^2$

d. $(x^3 + x^2 + x + 1) / (x^2 + 1) = x + 1$ (remainder is 0)

P10-14. Answer the following questions:

a. What is the polynomial representation of 101110?

b. What is the result of shifting 101110 three bits to the left?

c. Repeat part b using polynomials.

d. What is the result of shifting 101110 four bits to the right?

e. Repeat part d using polynomials.

a) Given bit stream is 101110 then the bit stream can be represented in polynomial is

$x^5 + x^3 + x^2 + x$

Note: Negative powers are deleted.

b) Given bit stream 101110

A binary pattern is often shifted a number of bits to the right or left. Shifting to the left means adding extra 0s as rightmost bits. For given bit stream 101110 then performing three bit left shift operation the result is 101110000 .here, three 0s are added to the right.

c) Given bit stream is 101110000 then the bit stream can be represented in polynomial is

$x^8 + x^6 + x^5 + x^4$

Note: Negative powers are deleted.

d) Given bit stream 101110

A binary pattern is often shifted a number of bits to the right or left. Shifting to the right means deleting some right most bits. For given bit stream 101110 then performing four bit right shift operation the result is 10 .here four bits are deleted to the right.

e) Given bit stream is 10 then the bit stream can be represented in polynomial is x

Note: Negative powers are deleted.

P10-15. Which of the following CRC generators guarantee the detection of a single bit error?

a. $x^3 + x + 1$ **b.** $x^4 + x^2$ **c.** 1 **d.** $x^2 + 1$

P10-15. To detect single bit errors, a CRC generator must have at least two terms and the coefficient of x^0 must be nonzero.

a. $x^3 + x + 1 \rightarrow$ It meets both critic.

b. $x^4 + x^2 \rightarrow$ It meets the first criteria, but not the second.

c. $1 \rightarrow$ It meets the second criteria, but not the first.

d. $x^2 + 1 \rightarrow$ It meets both criteria.

Single bit property: To detect single bit errors, a CRC generator has more than one term and the coefficient of x^0 is 1.

a) Given CRC generator $x^3 + x + 1$

This generator has more than one term and the coefficient of x^0 is 1.

So, the CRC generator $x^3 + x + 1$ is satisfying both criteria.

b) Given CRC generator $x^4 + x^2$

This generator has more than one term but is hasn't the coefficient of x^0 is 1.

So, the CRC generator $x^4 + x^2$ is satisfying the first criteria, but not satisfying the second.

c) Given CRC generator 1

This generator has not more than one term but is has the coefficient of x^0 is 1.

So, the CRC generator 1 is satisfying the second criteria, but not satisfying the first.

d) Given CRC generator $x^2 + 1$

This generator has more than one term and the coefficient of x^0 is 1.

So, the CRC generator $x^2 + 1$ is satisfying both criteria.

P10-16. Referring to the CRC-8 polynomial in Table 10.7, answer the following questions:

a. Does it detect a single error? Defend your answer.

b. Does it detect a burst error of size 6? Defend your answer.

c. What is the probability of detecting a burst error of size 9?

d. What is the probability of detecting a burst error of size 15?

Given that CRC-8 polynomial $x^8 + x^2 + x + 1$

a) Given CRC-8 polynomial detect single bit error because it has more than one term and the coefficient of x^0 is 1.So, it can detect a single-bit error.

b) Given CRC-8 polynomial detect burst error because the polynomial is of degree 8, which means that the number of check bits is 8. It will detect all burst errors of size 8 or less.

c) Burst errors of size 9 are detected most of the time, but they slip by with probability

(1/2)r −1 or (1/2)8−1≈ 0.008. This means 8 out of 1000 burst errors of size 9 are not here undetected.

d) Burst errors of size 15 are detected most of the time, but they slip by with probability

(1/2)r or (1/2)8 ≈ 0.004. This means 4 out of 1000 burst errors of size 15 are not here undetected.

P10-17. Referring to the CRC-32 polynomial in Table 10.4, answer the following questions:

a. Does it detect a single error? Defend your answer.

b. Does it detect a burst error of size 16? Defend your answer.

c. What is the probability of detecting a burst error of size 33?

d. What is the probability of detecting a burst error of size 55?

P10-17. This generator is $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

a. It has more than one term and the coefficient of x^0 is 1. It detects all single-bit error.

b. The polynomial is of degree 32, which means that the number of checkbits (remainder) r = 32. It will detect all burst errors of size 32 or less.

c. Burst errors of size 33 are detected most of the time, but they are slip by with probability $(1/2)^{r-1}$ or $(1/2)^{32-1} \approx 465 \times 10^{-12}$. This means **465 out of 10^{12}** burst errors of size 33 are left undetected.

d. Burst errors of size 55 are detected most of the time, but they are slipped with probability $(1/2)^r$ or $(1/2)^{32} \approx 233 \times 10^{-12}$. This means **233 out of 10^{12}** burst errors of size 55 are left undetected.

P10-18. Assume a packet is made only of four 16-bit words $(A7A2)_{16}$, $(CABF)_{16}$, $(903A)_{16}$, and $(A123)_{16}$. Manually simulate the algorithm in Figure 10.17 to find the checksum.

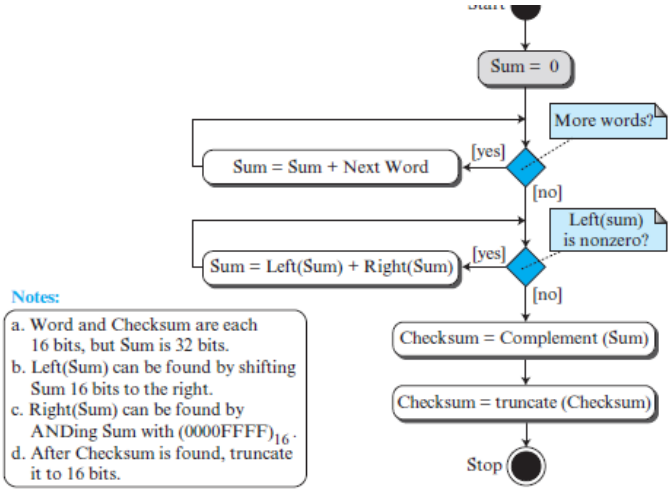


Figure 5.17 *Algorithm to calculate a traditional checksum*

P5-15. The following shows the steps:

a. We first add the numbers to get (0002A3BE)16. This corresponds to the first loop in Figure 5.17.

b. We extract the leftmost four digits, (0002)16, and the rightmost four digits, (A3BE)16, and add them together to simulate the second loop in Figure 5.17. The result is (A3C0)16. We stop here because the result does not create a carry.

c. Finally, we complement the result to get the checksum as (5C3F)16.

P10-19. Traditional checksum calculation needs to be done in one's complement arithmetic. Computers and calculators today are designed to do calculations in two's complement arithmetic. One way to calculate the traditional checksum is to add the numbers in two's complement arithmetic, find the quotient and remainder of dividing the result by 2^{16}, and add the quotient and the remainder to get the sum in one's complement. The checksum can be found by subtracting the sum from $2^{16} - 1$. Use the above method to find the checksum of the following four numbers: 43,689, 64,463, 45,112, and 59,683.

P10-19. The following shows the steps:

a. We first add the numbers in two's complement to get 212,947.

b. We divide the above result by 65,536 (or 2^{16}). The quotient is 3 and the remainder is 16,339. The sum of the quotient and the remainder is 16,342.

c. Finally, we subtract the sum from 65,535 (or $2^{16} - 1$), simulating the complement operation, to get 49,193 as the checksum.

P10-20. This problem shows a special case in checksum handling. A sender has two data items to send: $(4567)_{16}$ and $(BA98)_{16}$. What is the value of the checksum?

P5-17. The sum in this case is (FFFF)16 and the checksum is (0000)16. The problem shows that the checksum can be all 0s in hexadecimal. It can be all Fs in the hexadecimal only if all data items are all 0s, which makes no sense.

P10-21. Manually simulate the Fletcher algorithm (Figure 10.18) to calculate the checksum of the following bytes: $(2B)_{16}$, $(3F)_{16}$, $(6A)_{16}$, and $(AF)_{16}$. Also show that the result is a weighted checksum.

P10-21.

a. We calculate R and L values in each iteration of the loop and then concatenate L and R to get the checksum. All calculations are in hexadecimal and modulo 256 or $(FF)_{16}$. Note that R needs to be calculated before L in each iteration ($L = L_{previous} + R$).

Initial values:	`R = 00`	`L = 00`
Iteration 1:	`R = 00 + 2B = 2B`	`L = 00 + 2B = 2B`
Iteration 2:	`R = 2B + 3F = 6A`	`L = 2B + 6A = 95`
Iteration 3:	`R = 6A + 6A = D4`	`L = 95 + D4 = 69`
Iteration 4:	`R = D4 + AF = 83`	`L = 69 + 83 = EC`
Checksum = **EC83**		

b. The L and R values can be calculated as shown below (D_i is the corresponding bytes), which shows that L is the weighted sum of bytes.

$$R = D_1 + D_2 + D_3 + D_4 = 2B + 3F + 6A + AF = 83$$
$$L = 4 \times D_1 + 3 \times D_2 + 2 \times D_3 + 1 \times D_4 = EC$$

P10-22. Manually simulate the Adler algorithm (Figure 10.19) to calculate the checksum of the following words: $(FBFF)_{16}$ and $(EFAA)_{16}$. Also show that the result is a weighted checksum.

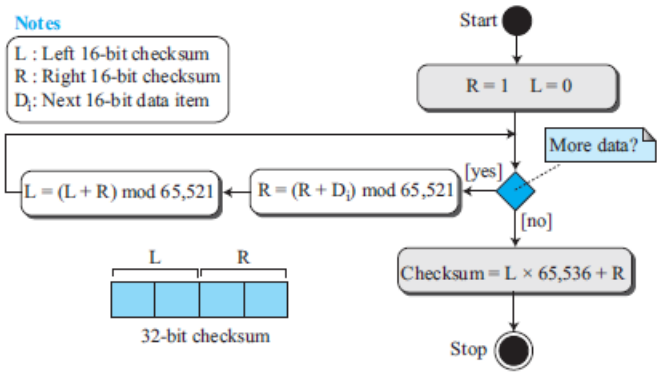


Figure 5.19 *Algorithm to calculate an Adler checksum*

Data in hex	Data in dec	R initial	Data + R in	R new = (D	L initial	L new = L +	L new = (L	CRC in DEC	CRC in HEX
FBFF	64511	1	64512	64512	0	64512	64512	4.23E+09	FC00FC00
EFAA	61354	64512	125866	60345	64512	124857	59336	3.89E+09	E7C3E389

The above table shows the Adler algorithm to find the checksum of 32 bit. For 16 bit data CRC calculated will be 32Bit.

According to Adler algorithm

Initial value for L=0 and R=1

Step 1: Add R to $Data_n$, where n is byte number of given data 1 to n and calculate mod 65521, keep it as new R value.

Step 2: Add L to the Calculated R Value step 1 and calculate mod 65521, keep it as new L value

Repeat Step 1 and Step 2 for all the data bytes.

Step 3: Now calculate final CRC first by multiplying new L value with 65536, and then by adding new R value to it.

In this process each position is given a different weight, which can be seen in the above calculation by observing the L and R values.

P10-23. One of the examples of a weighted checksum is the ISBN-10 code we see printed on the back cover of some books. In ISBN-10, there are 9 decimal digits that define the country, the publisher, and the book. The tenth (rightmost) digit is a checksum digit. The code, $D_1D_2D_3D_4D_5D_6D_7D_8D_9C$, satisfies the following.

$$[(10 \times D_1) + (9 \times D_2) + (8 \times D_3) + \ldots + (2 \times D_9) + (1 \times C)] \bmod 11 = 0$$

In other words, the weights are 10, 9, . . .,1. If the calculated value for C is 10, one uses the letter X instead. By replacing each weight w with its complement in modulo 11 arithmetic $(11 - w)$, it can be shown that the check digit can be calculated as shown below.

$$C = [(1 \times D_1) + (2 \times D_2) + (3 \times D_3) + \ldots + (9 \times D_9)] \bmod 11$$

Calculate the check digit for ISBN-10: **0-07-296775-C**.

P10-23. We use modulo-11 calculation to find the check digit:

$$C = (1 \times 0) + (2 \times 0) + (3 \times 7) + (4 \times 2) + (5 \times 9) + (6 \times 6) + (7 \times 7) + (8 \times 7) + (9 \times 5) \bmod 11 = 7$$

P10-24. An ISBN-13 code, a new version of ISBN-10, is another example of a weighted checksum with 13 digits, in which there are 12 decimal digits defining the book and the last digit is the checksum digit. The code, $D_1D_2D_3D_4D_5D_6D_7D_8D_9D_{10}D_{11}D_{12}C$, satisfies the following.

$$[(1 \times D_1) + (3 \times D_2) + (1 \times D_3) + \ldots + (3 \times D_{12}) + (1 \times C)] \bmod 10 = 0$$

In other words, the weights are 1 and 3 alternately. Using the above description, calculate the check digit for ISBN-13: **978-0-07-296775-C**.

P5-23. We first calculate the sum modulo 10 of all digits. We then let the check digit to be 10 - sum. In this way, when the check digit is added to the sum, the result is 0 modulo 10.

C = [(**1** x 9)+ (**3** x 7) + (**1** x 8) + (**3** x 0) + (**1** x 0) + (**3** x 7) + (**1** x 2) + (**3** x 9) + (**1** x 6) + (**3** x 7) + (**1** x 7) + (**3** x 5)] mod 10 = 137 mod $10 = 7 \rightarrow$ **C** = $10 - 7 = 3$

P10-25. In the interleaving approach to FEC, assume each packet contains 10 samples from a sampled piece of music. Instead of loading the first packet with the first 10 samples, the second packet with the second 10 samples, and so on, the sender loads the first packet with the odd-numbered samples of the first 20 samples, the second packet with the even-numbered samples of the first 20 samples, and so on. The receiver reorders the samples and plays them. Now assume that the third packet is lost in transmission. What will be missed at the receiver site?

P10-25. The receiver misses samples 21, 23, 25, 27, 29, 31, 33, 35, 37, and 39. However, the even-numbered samples are received and played. There may be some glitches in the audio, but that passes immediately.

P10-26. Assume we want to send a dataword of two bits using FEC based on the Hamming distance. Show how the following list of datawords/codewords can automatically correct up to a one-bit error in transmission.

$00 \rightarrow 00000$ $01 \rightarrow 01011$ $10 \rightarrow 10101$ $11 \rightarrow 11110$

When data words of two bits are to be sent the sender and receiver use the following table for encoding and decoding the data.

As k=2 , 2^k= 4 codeword's can be formed.

DataWord	Code Word
00	00000
01	01011
10	10101
11	11110

a) The codeword contains the data word and the redundant bits, when the user transfers a codeword 00000. '00' is the data word and the remaining bits are the redundant bits.

• Suppose the original codeword is '00000', and the received codeword is 00001.

• The receiver compares the received codeword with the table entries, as no matching is found the receiver understands an error has occurred.

• The receiver compares the received codeword to identify the corrupted bit.

• The receiver checks for the one bit difference between the codeword received and between the codeword's in the table.

• Only the first entry '00000' has one bit difference so the receiver corrects the error.

b) The codeword contains the data word and the redundant bits, when the user transfers a codeword 01011. '01' is the data word and the remaining bits are the redundant bits.

• Suppose the original codeword is '01011', and the received codeword is 01001.

• The receiver compares the received codeword with the table entries, as no matching is found the receiver understands an error has occurred.

• The receiver compares the received codeword to identify the corrupted bit.

• The receiver checks for the one bit difference between the codeword received and between the codeword's in the table.

• Only the second entry '01011' has one bit difference so the receiver corrects the error.

c) The codeword contains the data word and the redundant bits, when the user transfers a codeword 10101. ‘10’ is the data word and the remaining bits are the redundant bits.

• Suppose the original codeword is ‘10101’, and the received codeword is 10100.

• The receiver compares the received codeword with the table entries, as no matching is found the receiver understands an error has occurred.

• The receiver compares the received codeword to identify the corrupted bit.

• The receiver checks for the one bit difference between the codeword received and between the codeword’s in the table.

• Only the third entry ‘10100’ has one bit difference so the receiver corrects the error.

d) The codeword contains the data word and the redundant bits, when the user transfers a codeword 11110. ‘11’ is the data word and the remaining bits are the redundant bits.

• Suppose the original codeword is ‘11110’, and the received codeword is 11111.

• The receiver compares the received codeword with the table entries, as no matching is found the receiver understands an error has occurred.

• The receiver compares the received codeword to identify the corrupted bit.

• The receiver checks for the one bit difference between the codeword received and between the codeword’s in the table.

• Only the fourth entry ‘11110’ has one bit difference so the receiver corrects the error.

P10-27. Assume we need to create codewords that can automatically correct a one-bit error. What should the number of redundant bits (r) be, given the number of bits in the dataword (k)? Remember that the codeword needs to be $n = k + r$ bits, called C(n, k). After finding the relationship, find the number of bits in r if k is 1, 2, 5, 50, or 1000.

P10-27. The redundant bits in this case need to find $(n + 1)$ different states because the corruption can be in any of the n bits or in no bits (no corruption). A set of r bits can define 2^r states. This means that we need to have the following relationship: $2^r \geq n + 1$. We need to solve the equation for each value of k using trial and error to find the minimum value of r.

a. If $k = 1$, then $r = 2$ and $n = 3$ because $(2^2 \geq 3 + 1)$, which means C(3, 1).

b. If $k = 2$, then $r = 3$ and $n = 5$ because $(2^3 \geq 5 + 1)$, which means C(5, 1).

c. If $k = 5$, then $r = 4$ and $n = 9$ because $(2^4 \geq 9 + 1)$, which means C(9, 5).

d. If $k = 50$, then $r = 6$ and $n = 56$ because $(2^6 \geq 56 + 1)$, which means C(56, 50).

e. If $k = 1000$, then $r = 10$ and $n = 1010$ because $2^{10} \geq 1010 + 1$, which means C(1010, 1000).

P10-28. In the previous problem we tried to find the number of bits to be added to a dataword to correct a single-bit error. If we need to correct more than one bit, the number of redundant bits increases. What should the number of redundant bits (r) be to automatically correct one or two bits (not necessarily contiguous) in a dataword of size k? After finding the relationship, find the number of bits in r if k is 1, 2, 5, 50, or 1000.

• In encoding using hamming distance, r redundant bits are added to the data word of 'k' bits long for error correction, the relation between the codeword and the data word is n = k+r.

• Given that n=k+r, where 'n' is the total number of bits in the code word. 'k' is the data bits, and 'r' is the redundant bits added.

• The number of redundant bits required to identify one bit error can be determined by the following equation based on hamming distance. $k \le 2^r - r - 1$

That is 2 $2^r \ge n+1$, $2^r \ge n_{C_0} + n_{C_1}$ (combination of '0' bit errors and '1' bit error)

• From the above equation the number of redundant bits required to identify one bit and 2 bit errors can be determined by the following equation based on hamming distance

$$2^r \ge n_{C_0} + n_{C_1} + n_{c_2} \quad 2^r \ge n+1+\frac{n(n-1)}{2}$$

• When k=1 means n=r+1, by trial and error method select the value of r which satisfies the condition.

$$2^r \ge n+1+\frac{n(n-1)}{2},$$

when r= 4, 2^4>=(4+1+1+5(5-1)/2)= 16 the condition is satisfied. Therefore when k=1 the minimum number of redundant bits are r= 4.

• When k=2 means n=r+2, by trial and error method select the value of r which satisfies the condition.

$$2^r \ge n+1+\frac{n(n-1)}{2},$$

When r=5, 2^5>= (5+2)+1 + 21=29, therefore when k=2 the minimum number of redundant bits required are r=5.

• When k=5 means n=r+5, by trial and error method select the value of r which satisfies the condition.

$$2^r \geq n+1+\frac{n(n-1)}{2},$$

When r=7, 2^7>=(7+5)+1+66=79, therefore when k=5 the minimum number of redundant bits required are r=7.

• When k=50 means n=r+50, by trial and error method select the value of r which satisfies the condition.

$$2^r \geq n+1+\frac{n(n-1)}{2},$$

When r=11, 2^{11}>= (11+50)+1+ 61*30=1892, therefore when k=50 the minimum number of redundant bits required are r=11.

• When k=1000 means n=r+1000, by trial and error method select the value of r which satisfies the condition.

$$2^r \geq n+1+\frac{n(n-1)}{2},$$

When r=19, 2^{19}>=518671, therefore when k=1000 the minimum number of redundant bits required are r = 19.

P10-29. Using the ideas in the previous two problems, we can create a general formula for correcting any number of errors (m) in a codeword of size (n). Develop such a formula. Use the combination of n objects taking x objects at a time.

P10-29. If we need to correct m bits in an n bit codeword, we need to think about the combination of n objects taking no object at a time or $Com(n, 0)$, which means the state of no error, the combination of n objects taking one object at a time or $Com(n, 1)$, which means the state of one-bit error, the combination of n objects taking two objects at a time or $Com(n, 2)$, which means the state of two-bit error, and so on. We can have the following relationship between the value of r (number of redundant bits) and the value of m (the number of errors) we need to correct.

$$2^r \geq Com(n, m) + Com(n, m-1) + \ldots + Com(n, 1) + Com(n, 0)$$

P10-30. In Figure 10.22, assume we have 100 packets. We have created two sets of packets with high and low resolutions. Each high-resolution packet carries on average 700 bits. Each low-resolution packet carries on average 400 bits. How many extra bits are we sending in this scheme for the sake of FEC? What is the percentage of overhead?

P8-26. Sending only high-resolution packets means sending only 700 bits per packet, or 70,000 bits. Sending both high-resolution and low-resolution means sending 1100 bits per packet or 110,000 bits. We are sending 40,000 extra bits. The overhead is $(110{,}000 - 70{,}000) / 70{,}000 \approx 57$ percent.
