Data Link Layer

Dr. Vijay Ukani

CSE Department, Institute of Technology, Nirma University

Error Control

Q. What is the hamming distance between 0111110000111011 0111111000011001?

Ans.

0111110000111011

xor 0111111000011001

0000001000100010

So distance =3

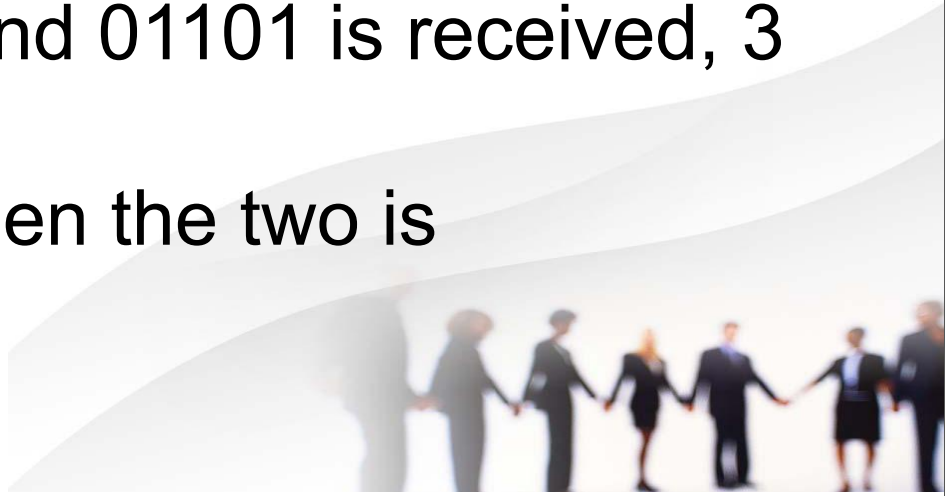
Error Control

Q. Discuss the relationship between the Hamming distance and errors occurring during transmission.

Ans. When a codeword is corrupted during transmission, the Hamming distance between the sent and received codewords is the number of bits affected by the error. In other words,the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission.

For example, if the codeword 00000 is sent and 01101 is received, 3 bits

Are in error and the Hamming distance between the two is d(00000,01101)=3.



Error Control

Q. What minimum distance between codes is required for error detection?

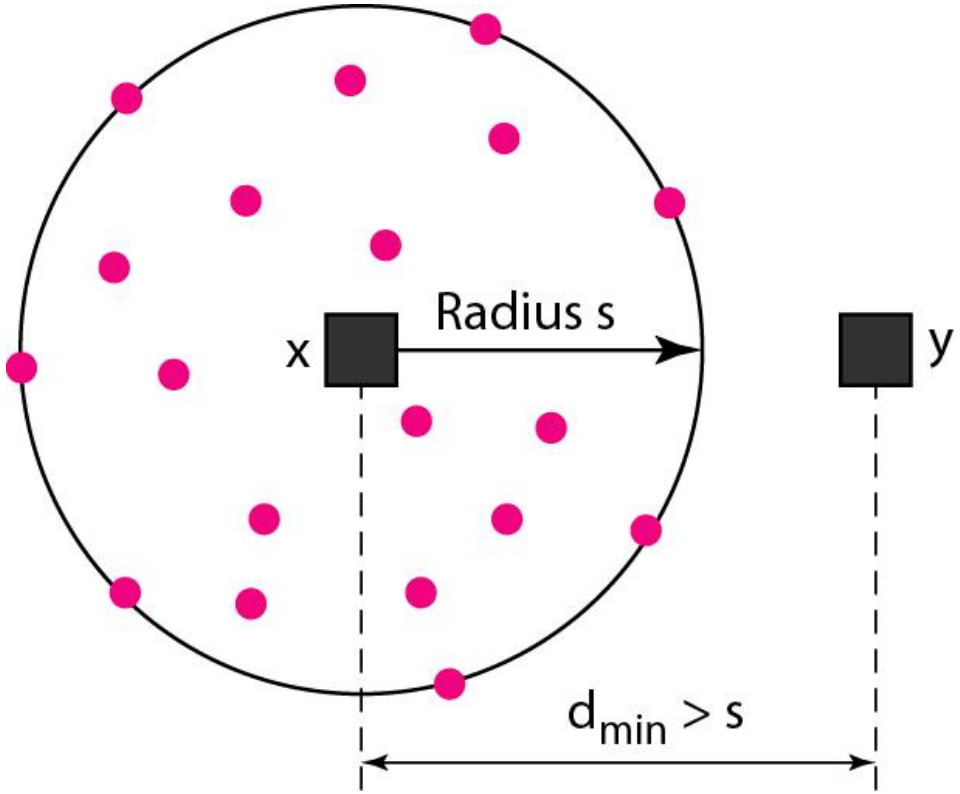
Ans. If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If our code is to **detect** upto s errors, the minimum distance between the valid codes must be **s+1**, so that the received codeword does not match a valid codeword.

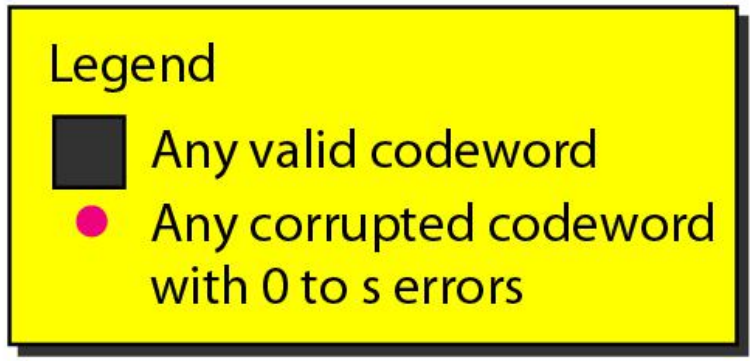
If the minimum distance between all valid codewords is **s+1**, the received codeword cannot be erroneously mistaken for another codeword.

Error Control

Q. What minimum distance between codes is required for error detection?

Ans.

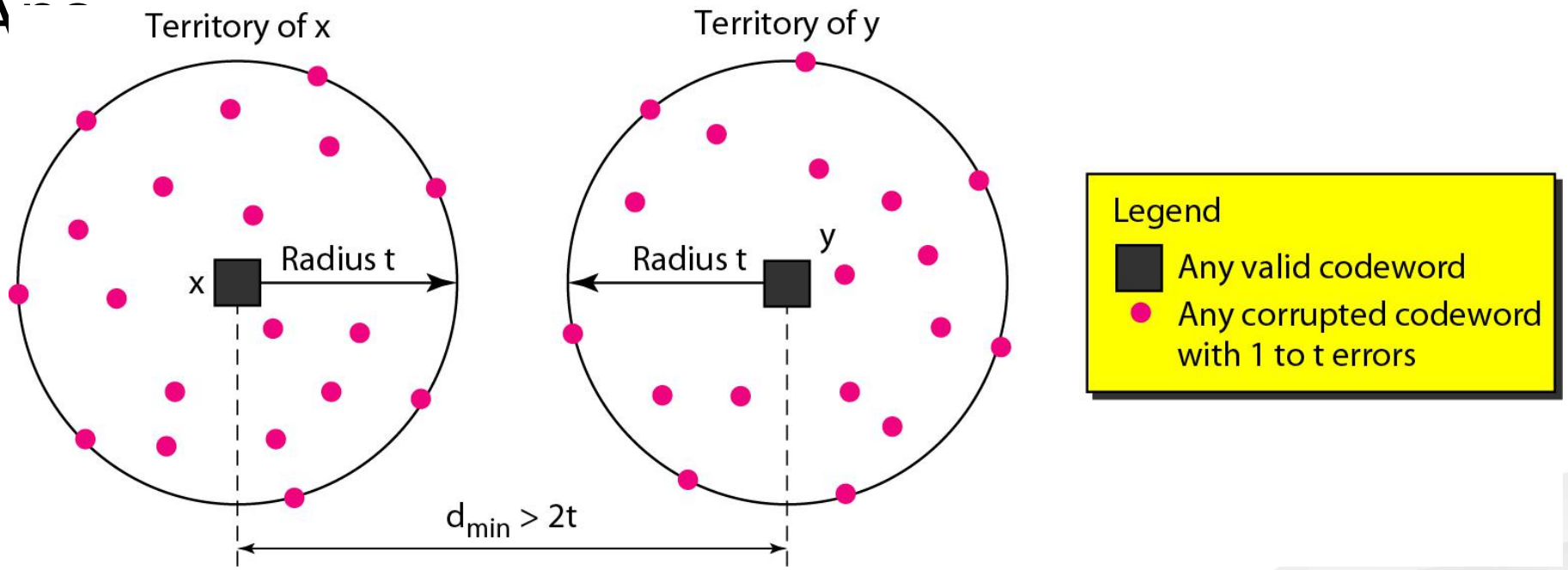




Error Control

Q. What is the minimum distance required for error correction?

A

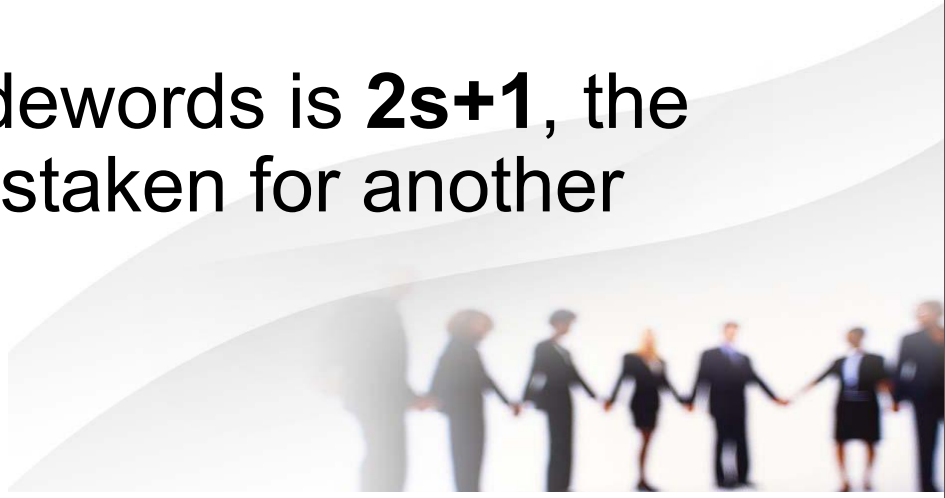


Error Control

Q. What minimum distance between the codes is required for error correction?

Ans. If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If our code is to **correct** upto s errors, the minimum distance between the valid codes must be **2s+1**, so that the received codeword does not match a valid codeword and correct code can be matched from list of valid codes

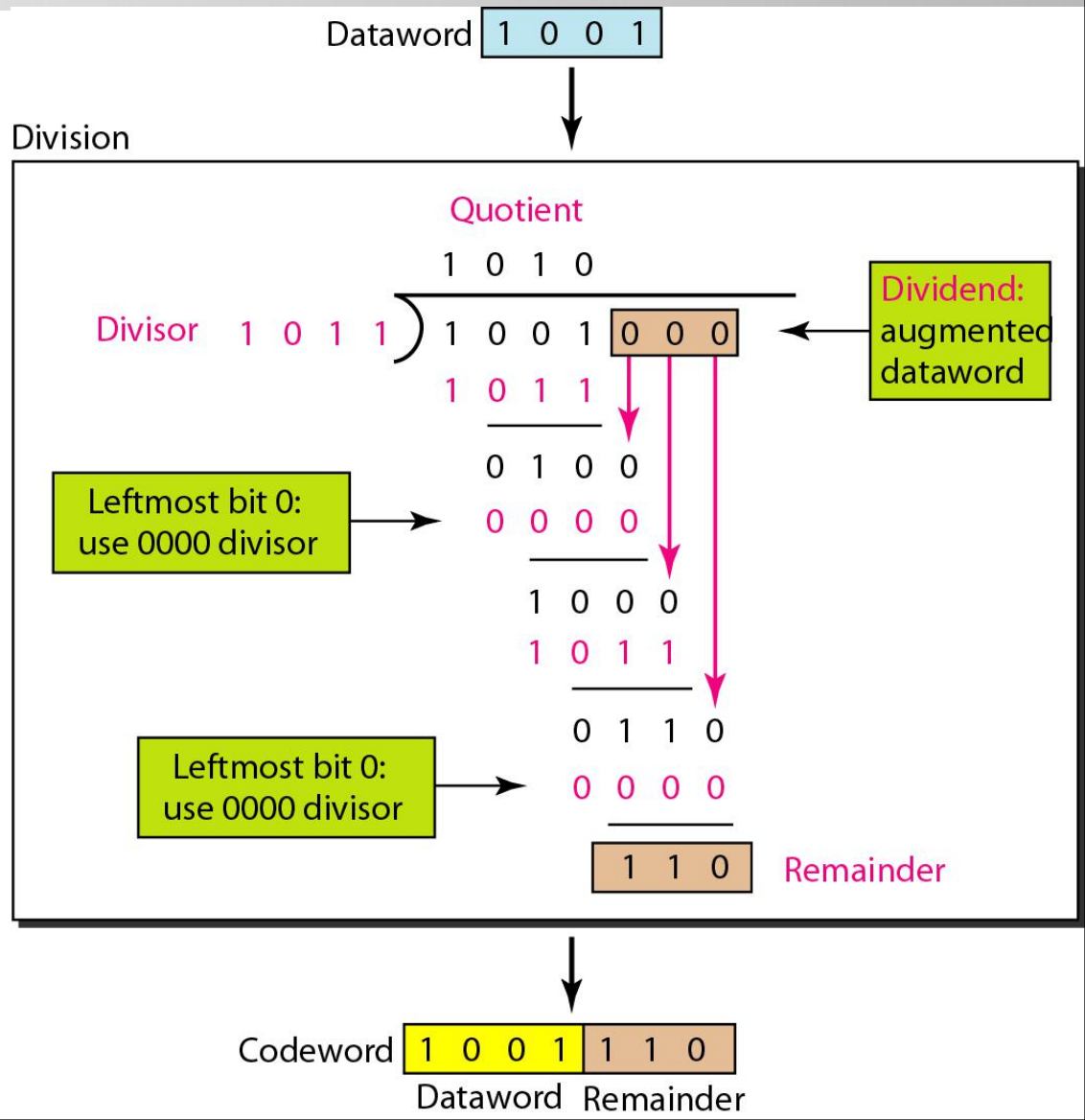
If the minimum distance between all valid codewords is **2s+1**, the received codeword cannot be erroneously mistaken for another codeword.



Error Control

Q. Suppose 1001 is to be transmitted with generator polynomial 1011. Compute CRC code and verify the reception

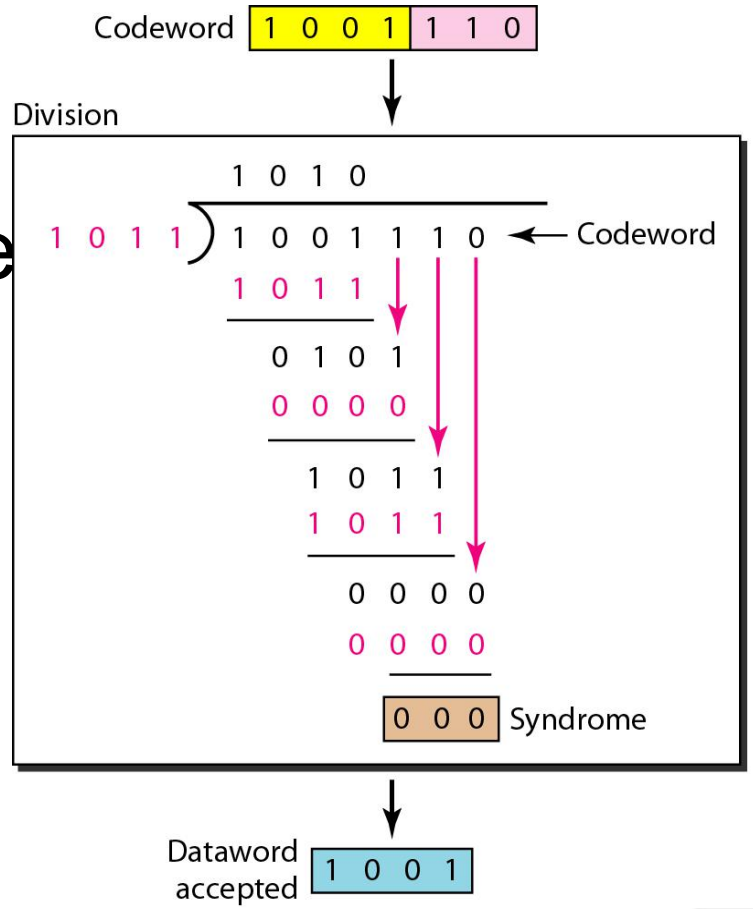
Ans. Sender

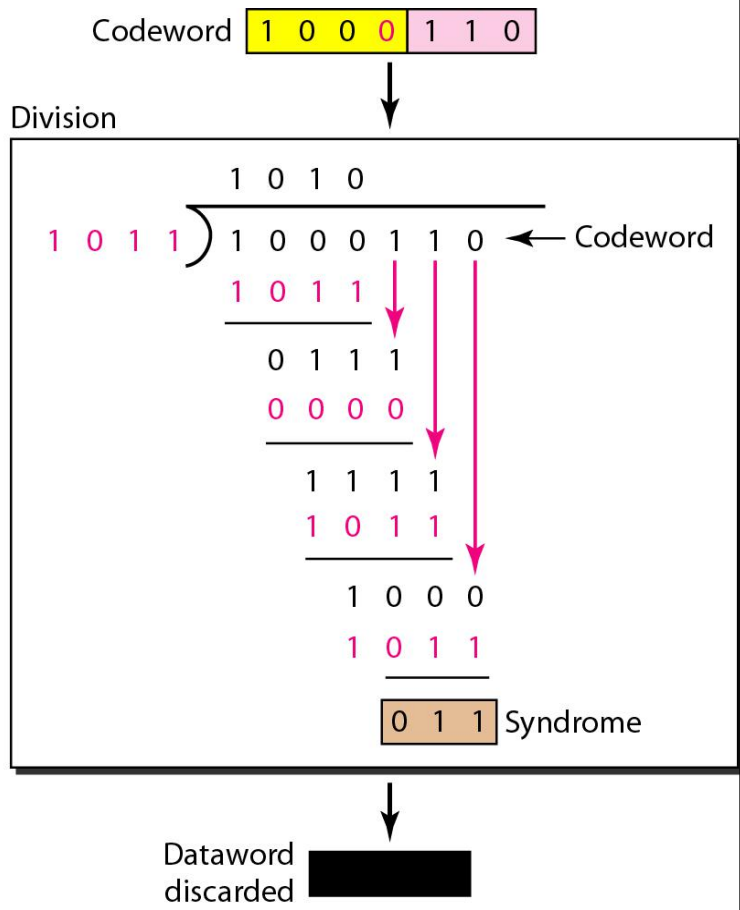


Error Control

Q. Suppose 1001 is to be transmitted with generator polynomial 1011. Compute CRC code and verify the reception

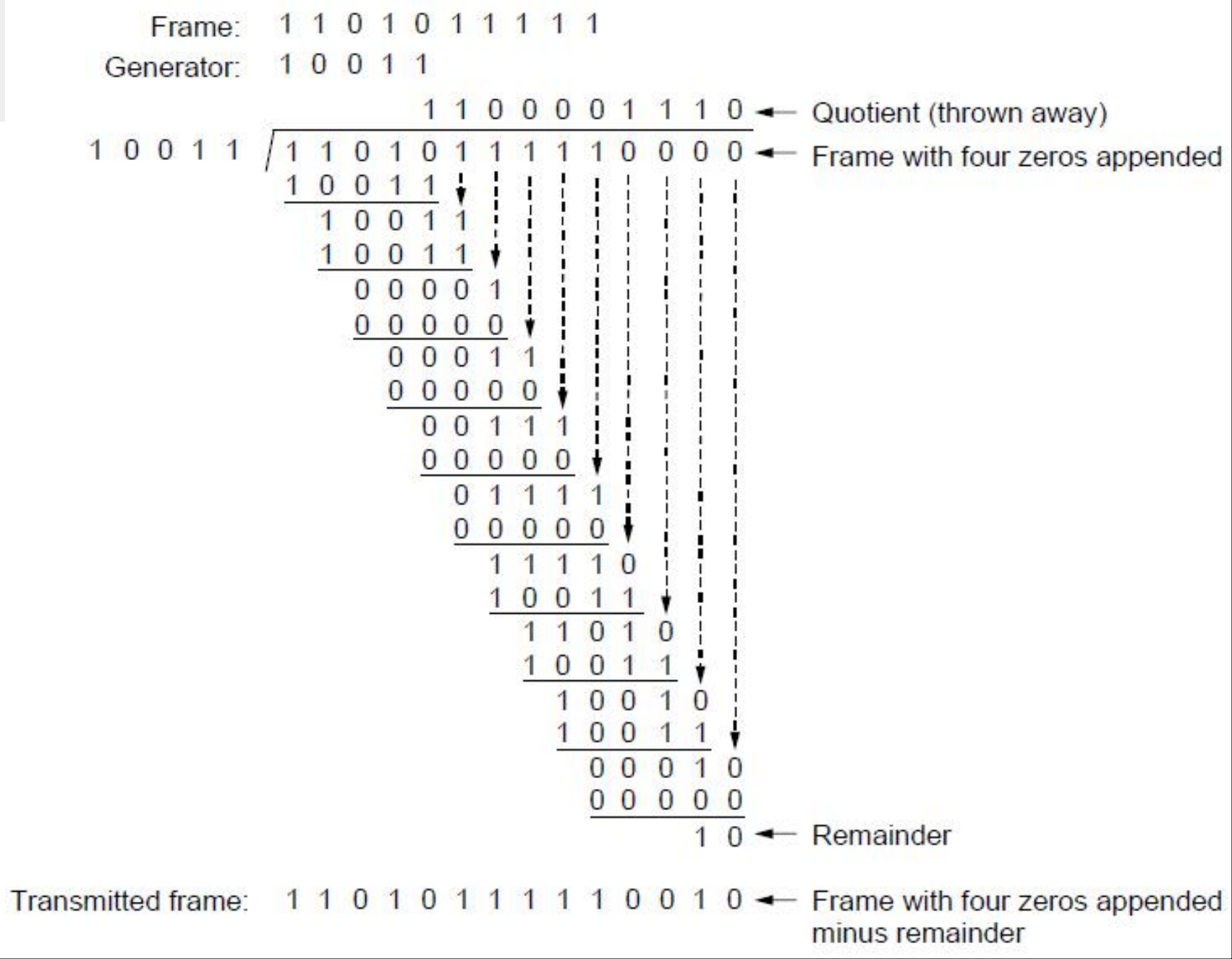
Ans. Receiver





Error Control

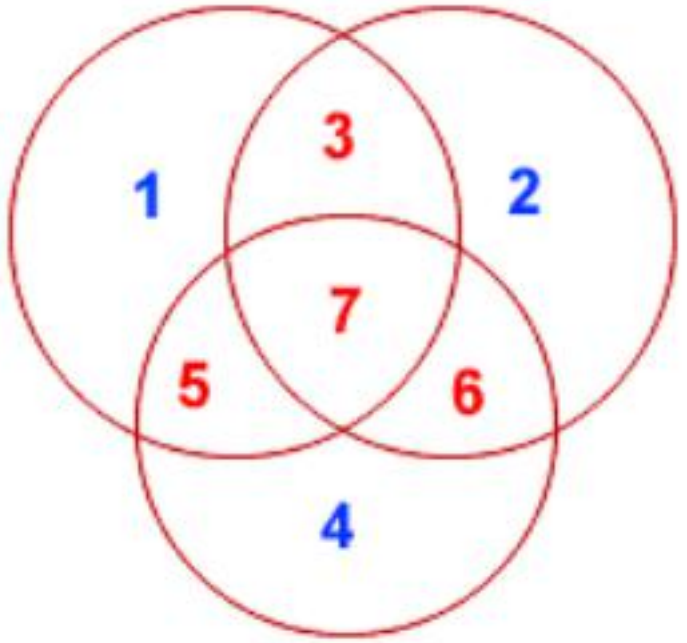
Q. Suppose 1101011111 is to be transmitted with generator polynomial 10011. Compute CRC code



The Hamming Code

Consider a message having four data bits (D) which is to be transmitted as a 7-bit codeword by adding three error control bits. This would be called a (7,4) code. The three bits to be added are three EVEN Parity bits (P), where the parity of each is computed on different subsets of the message bits as shown below.

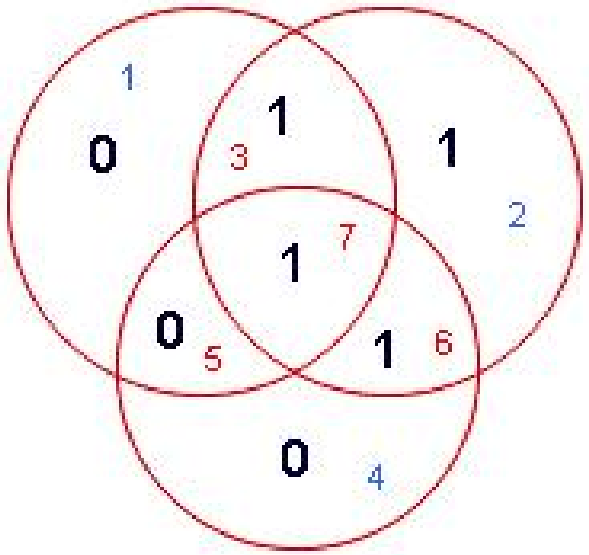
7	6	5	4	3	2	1	
D	D	D	P	D	P	P	7-BIT CODEWORD
D	-	D	-	D	-	P	(EVEN PARITY)
D	D	-	-	D	P	-	(EVEN PARITY)
D	D	D	P	-	-	-	(EVEN PARITY)



The Hamming Code

- For example, the message 1101 would be sent as 1100110, since:

7	6	5	4	3	2	1	
1	1	0	0	1	1	0	7-BIT CODEWORD
1	-	0	-	1	-	**0**	(EVEN PARITY)
1	1	-	-	1	**1**	-	(EVEN PARITY)
1	1	0	**0**	-	-	-	(EVEN PARITY)



The Hamming Code

- It may now be observed that if an error occurs in any of the seven bits, that error will affect different combinations of the three parity bits depending on the bit position.

- For example, suppose the above message 1100110 is sent and a single bit error occurs such that the codeword 1110110 is received:

transmitted message		received message
1 1 0 0 1 1 0	---------->	1 1 1 0 1 1 0
BIT: 7 6 5 4 3 2 1		BIT: 7 6 5 4 3 2 1

The above error (in bit 5) can be corrected by examining which of the three parity bits was affected by the bad bit:

The Hamming Code

7	6	5	4	3	2	1			
1	1	1	**0**	1	**1**	**0**	7-BIT CODEWORD		
1	-	1	-	1	-	0	(EVEN PARITY)	**NOT!**	**1**
1	1	-	-	1	1	-	(EVEN PARITY)	**OK!**	**0**
1	1	1	0	-	-	-	(EVEN PARITY)	**NOT!**	**1**

Framing

Q. The following data fragment occurs in the middle of a data stream for which the byte-stuffing algorithm described in the text is used: A B ESC C ESC FLAG FLAG D. What is the output after stuffing?

Ans.

After stuffing, we get

A B **ESC** ESC C **ESC** ESC **ESC** FLAG **ESC** FLAG D

- Complete frame with delimiter

FLAG A B **ESC** ESC C **ESC** ESC **ESC** FLAG **ESC** FLAG D FLAG

Framing

Q. A bit string, 01111101111101111110, needs to be transmitted at the data link layer. What is the string actually transmitted after bit stuffing?

Ans.

The output is 011111011111**0**011111**0**10

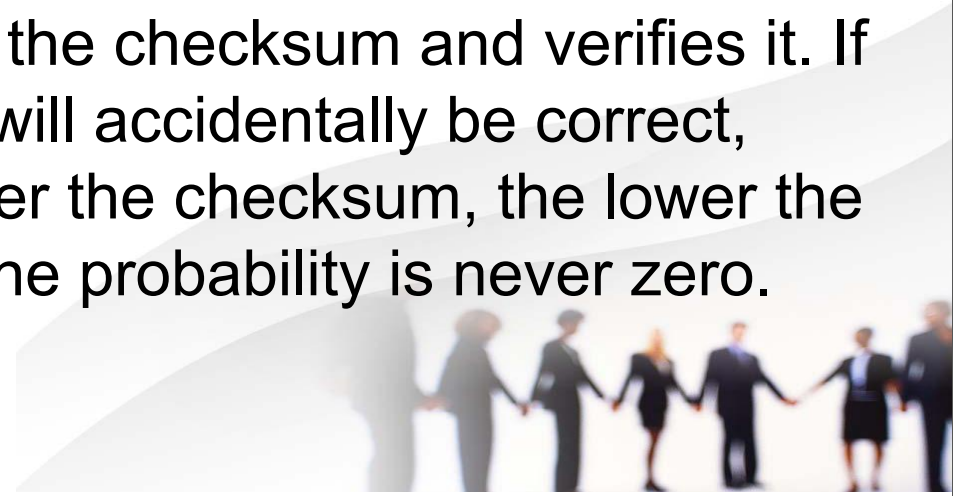
Framing

Q. When bit stuffing is used, is it possible for the loss, insertion, or modification of a single bit to cause an error not detected by the checksum? If not, why not? If so, how? Does the checksum length play a role here?

Ans.

It is possible.

Suppose that the original text contains the bit sequence 01111110 as data. After bit stuffing, this sequence will be rendered as 011111010. If the second 0 is lost due to a transmission error, what is received is 01111110, which the receiver sees as end of frame. It then looks just before the end of the frame for the checksum and verifies it. If the checksum is 16 bits, there is 1 chance in 2^{16} that it will accidentally be correct, leading to an incorrect frame being accepted. The longer the checksum, the lower the probability of an error getting through undetected, but the probability is never zero.



Equations

- R = Transmission rate
- S = Signal Speed
- D = Distance between sender and receiver
- T = Time to create one frame
- F = Number of bits in a frame
- N = Number of data bits in a frame
- A = Number of bits in an acknowledgement

➢ Time needed to transmit one frame = F/R

➢ The receiver receives the last bit of the first frame at time = T + F/R + D/S

➢ The time required for the sender to receive the ack is = T + A/R + D/S

➢ Time elapses between sending two data frames = [2*(T + D/S)] + [(F+A)/R]

➢ Amount of time a frame is actually in transit is = F/R + D/S

Stop-N-Wait Protocol

A system uses the Stop-N-Wait protocol. If each packet carries 1000 bits of data, how long does it take to send 1 million bits of data if the distance between the sender and receiver is 5000km and propagation speed is $2*10^8$m. Ignore transmission, waiting and processing delays. We assume no data or control frame is damaged or lost.

- Number of packets = $10^6/10^3$ = 1000
- $2 * 10^8$m……..1 sec
- $5000 * 10^3$m…….. 25ms; RTT = 2 *25 =50ms
- Time to send 1 packet = 50ms
- Time to send 1000 packet = 50sec

If the distance between A and B is 4000km, how long does it take computer A to receive ACK for a packet? Use the speed of light for propagation speed and assume time between receiving and sending ACK is zero.

- Speed of light = $3 * 10^8$ m/s
- $3 * 10^8$ m …… 1 sec
- $4000 * 10^3$m…….. = 1/75
- RTT = 2/75 = 26.7ms
- Total time to receive ACK = Transmission time + RTT = 26.7

Total Time

Calculate the total time required to transfer a 1.5 MB file in the following cases, assuming RTT of 80ms, a packet size of 1KB and an initial 2 x RTT of “handshaking” before it is sent.

Total Time

The b/w is 10Mbps, and the data packets can be sent continuously.

- We will count the transfer as completed when the last data bit arrives at its destination
- Total time = 2 RTT + Transmission Time + Propagation Time
- 1.5MB = 12,582,912 bits
- Total Time = 2 x 80 ms + (12,582,912)/(10,000,000) bps + RTT/2
- Total Time = 1.458 secs (approx)

Total Time

The b/w is 10Mbps, but after we finish sending each data packet, we must wait one RTT before sending the next

- To the above, we add the time for 1499 RTTs for a total of 1.46 + 119.92 = 121.38 secs

Total Time

The link allows infinitely fast transmits, but limits bandwidth such that only 20 packets can be sent per RTT

- This is 74.5 RTTs, plus the two initial RTTS, for 6.12 secs

Total Time

Assuming zero transmit time as in previous case, but during the first RTT, we can send one packet, during the 2^{nd} RTT we can send 2 packets, during the 3^{rd} we can send $4 = 2^{(3-1)}$ and so on

- Right after the handshaking is done, we send one packet. One RTT after handshaking, we send 2 packets. At n RTTs past the handshaking, we have sent $1+2+4+... + 2^n = 2^{(n+1)} – 1$. At n =10, we have thus been able to send all 1500 pkts, the last batch arrives 0.5 RTT later. Total time is 2 + 10.5 RTTs or 1 sec.

Link Utilization

If the bandwidth of the line is 1 mbps, propagation delay is 20ms and packet size is 1kB then what will be the link utilization for a stop and wait protocol?

- $1 * 10^6$ bits …… 1 sec
- $8 * 10^3$ bits ……8 ms
- RTT = 2*20ms = 40ms.
- $U = T_{trans}/(T_{trans}+RTT) *100 = 800/48 = 16.667$ %

Stop-N-Wait Protocol

If bit rate is 10kbps, propagation delay is 40ms then for what frame size does stop-n-wait protocol give efficiency of 50%?

Ans

Link utilization = (1/(1 + 2BD)) and BD = bandwidth-delay product / frame size

- ½ = 1/(1+2a) where a = T_{prop}/T_{trans}, a=40/T_t
- T_t = 80 ms
- 1 sec…. bits transferred = 10 * 10^3
- 80 ms…. Bits transferred = 800bits

Stop-N-Wait Protocol

The distance from earth to a distant planet is approximately 9×10^{10} m. What is the channel utilization if a stop-and-wait protocol is used for frame transmission on a 64 Mbps point-to-point link? Assume that the frame size is 32 KB and the speed of light is 3×10^{8} m/s.

Ans.

Link utilization = (1/(1 + 2BD)) and BD = bandwidth-delay product / frame size

delay = $(9 \times 10^{10}) / (3 \times 10^{8}) = 300$ sec

bandwidth-delay product = 64 × 300 = 19. 2 Gbit

BD = 19200000 / 256 = 75000 frames.

So, link utilization is 6.67×10^{-4} %

Stop-N-Wait Protocol

In the previous problem, suppose a sliding window protocol is used instead. For what send window size will the link utilization be 100%? You may ignore the protocol processing times at the sender and the receiver.

Ans.

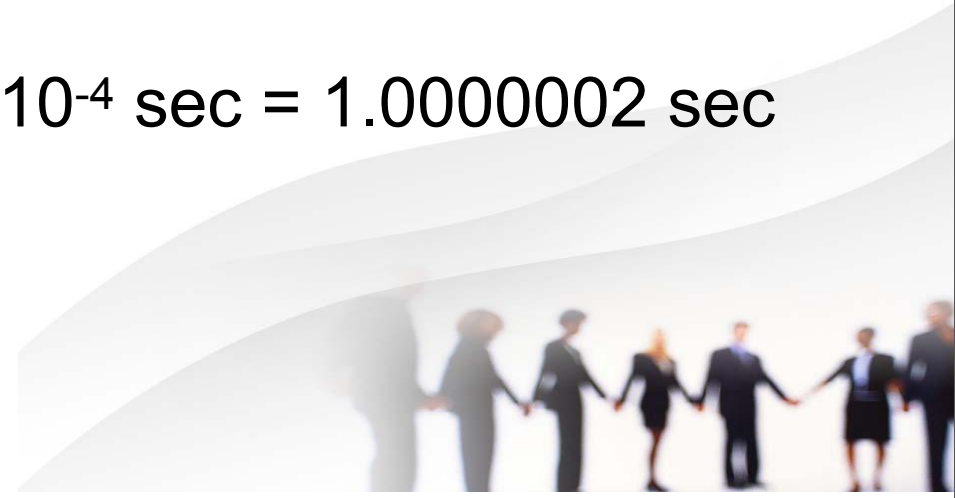
For a send window size w, link utilization is w/(1 + 2BD).

So, for 100% link utilization, w = 150001.

Stop and wait protocol

In stop and wait ARQ a station A sends 1000 bit frame to station B at 10 mbps. After sending a frame how much time computer will idle if size of ACK is 2 bit and RTT is 1 sec.

- To send 1000 bit requires $(1000/10*10^{6})$sec = 10^{-4} sec
- Ack takes $2/10^{7} = 2 * 10^{-7}$ sec
- Total time after sending the frame= $1 + 0.002*10^{-4}$ sec = 1.0000002 sec



Go-Back-N sliding window protocol

A 3000-km-long T1 trunk is used to transmit 64-byte frames using protocol 5. If the propagation speed is 6 μsec/km, how many bits should the sequence numbers be?

- The propagation time is 18 ms.
- At T1 speed, which is 1.536 Mbps (excluding the 1 header bit), a 64-byte frame takes 0.300 msec
- The first frame fully arrives 18.3 msec after its transmission was started. The acknowledgement takes another 18 msec to get back
- So first Ack is received at 36.3 msec
- The transmitter needs to have enough window space to keep going for 36.3 msec.
- A frame takes 0.3 ms, so it takes 121 frames to fill the pipe. Seven-bit sequence numbers are needed.

Go-Back-N sliding window protocol

Two stations, A & B connected via point-to-point link and exchange frames using go-back-n sliding window protocol with window size 7. Sequence number is of 3 bits. Station A sends frame 0 to 6. B receives them in order but frame 4 was damaged. So what will be the buffer frames in the current window of A?

- 4,5,6,7,0,1,2.