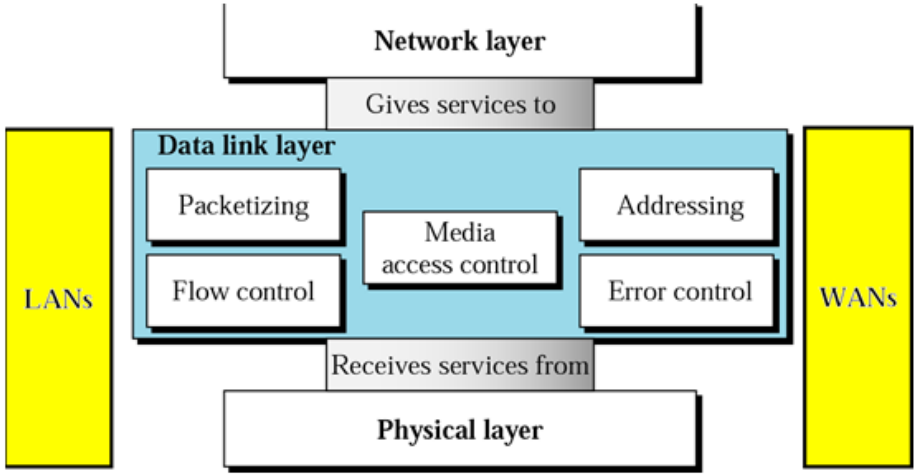
Overview of DLL

- The data link layer transform the physical layer, a raw transmission facility to a link responsible for node-to-node transmission
- Specific responsibilities of data link layer includes framing, addressing, flow control, error control and media access control



Design Issues in Data Link Layer

- Services provided
  - The data link layer act as a service interface to the network layer
  - The principle service is transferring data from network layer on sending machine to the network layer on destination machine
- Frame Synchronization
  - The source machine sends data in the form of blocks called frames to the destination machine
  - The starting and ending of each frame should be identified so that the frame can be recognized by the destination machine

Design Issues in Data Link Layer

- Flow control
  - Flow control is done to prevent the flow of data frame at the receiver end
  - The source machine must not send data frames at a rate faster than the capacity of destination machine to accept them
- Error control
  - Error control is done to prevent duplication of frames
  - The errors introduced during transmission from source to destination machines must be detected and corrected at the destination machine

Framing

- Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination
- The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing
- The data link layer on the other hand needs to pack bits into frames, so that each frame is distinguishable from another

Why whole message is not packed in single frame?

- Although the whole message could be packed in one frame, generally it is not normally done
- One reason is that a frame can be very large, making flow and error control very inefficient
- When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message
- When a message is divided into smaller frames, a single-bit error affects only that small frame

Parts of a Frame

- A frame has the following parts
  - Frame Header: It contains the source and the destination addresses
  - Payload field: It contains the message to be delivered
  - Trailer: It contains the error detection and error correction bits
  - Flag: It marks the beginning and end of the frame

Types of Framing

- Fixed size
  - In fixed-size framing, there is no need for defining the boundaries of the frames
  - The size itself can be used as a delimiter
- Drawback
  - It suffers from internal fragmentation if data size is less than frame size

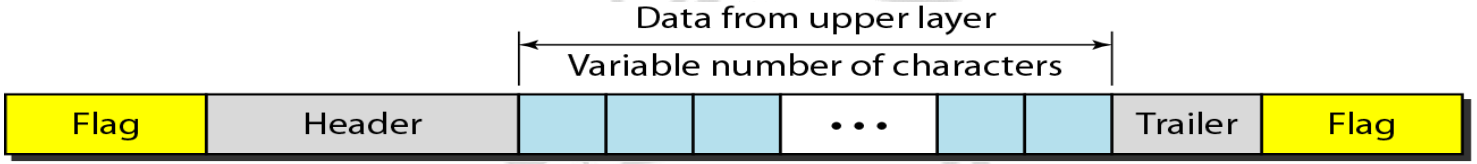
Types of Framing

- Variable-Size Framing
  - In variable-size framing, we need a way to define the end of the frame and the beginning of the next

- Historically, two approaches were used for this purpose
  - Character-oriented approach
  - Bit-oriented approach

Character-oriented approach

- To separate one frame from the next, an 8-bit (1-byte) flag is added at the beginning and end of a frame
- The flag composed of protocol-dependent special characters, which signals the start or end of a frame

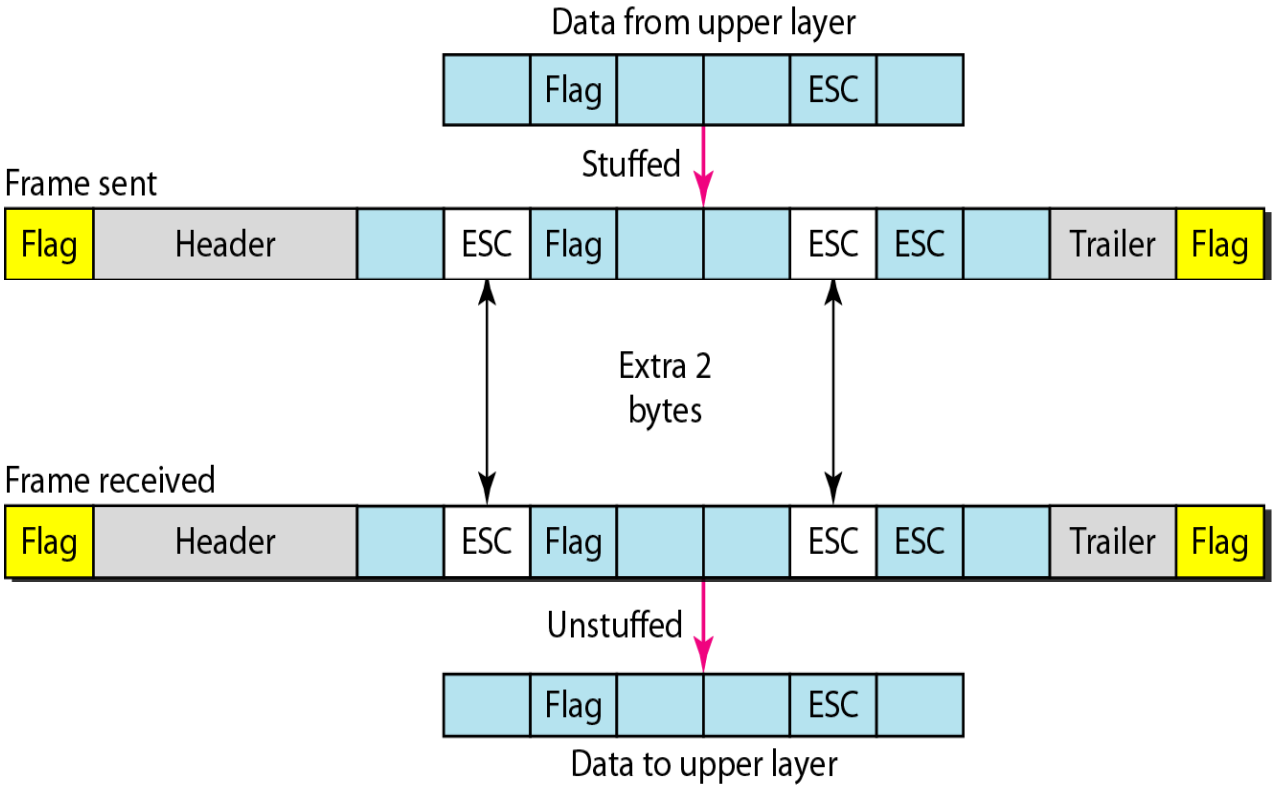


- Problem???

Character-oriented protocols

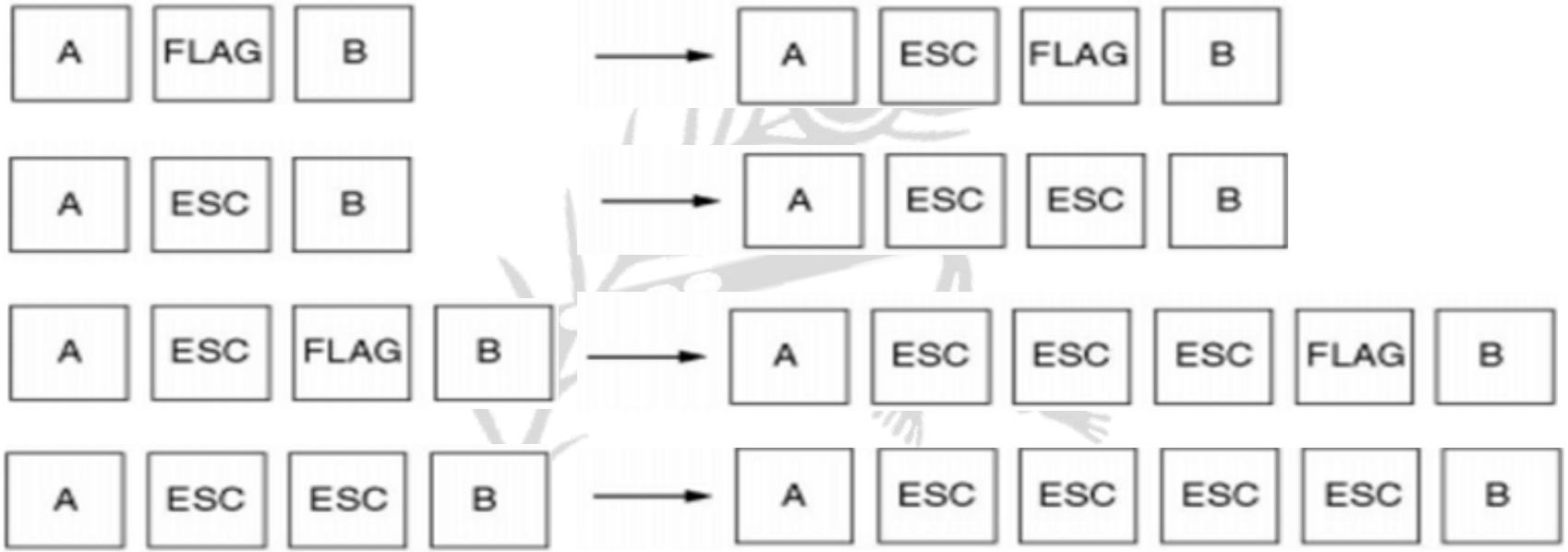
- In byte stuffing, a special byte is added to the data section of the frame when there is a character with the same pattern as the flag
- The data section is stuffed with an extra byte
- This byte is usually called the escape character (ESC), which has a predefined bit pattern
- Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag

Character-oriented protocols



Character-oriented protocols

- Eg:



Character-oriented protocols

- Eg:
  - Considering “f” as flag delimiter and “e” as DLE (escape) character, Apply the byte stuffing on the single frame data

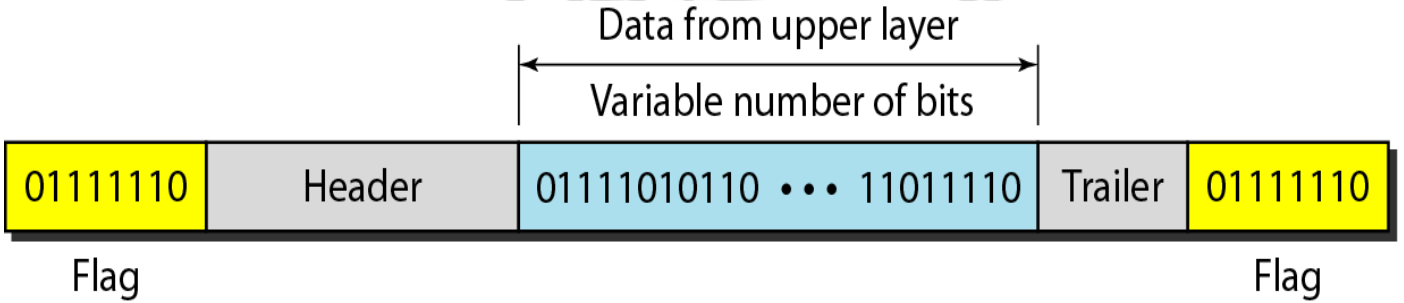
five and four

Solution

f efivee and efour f

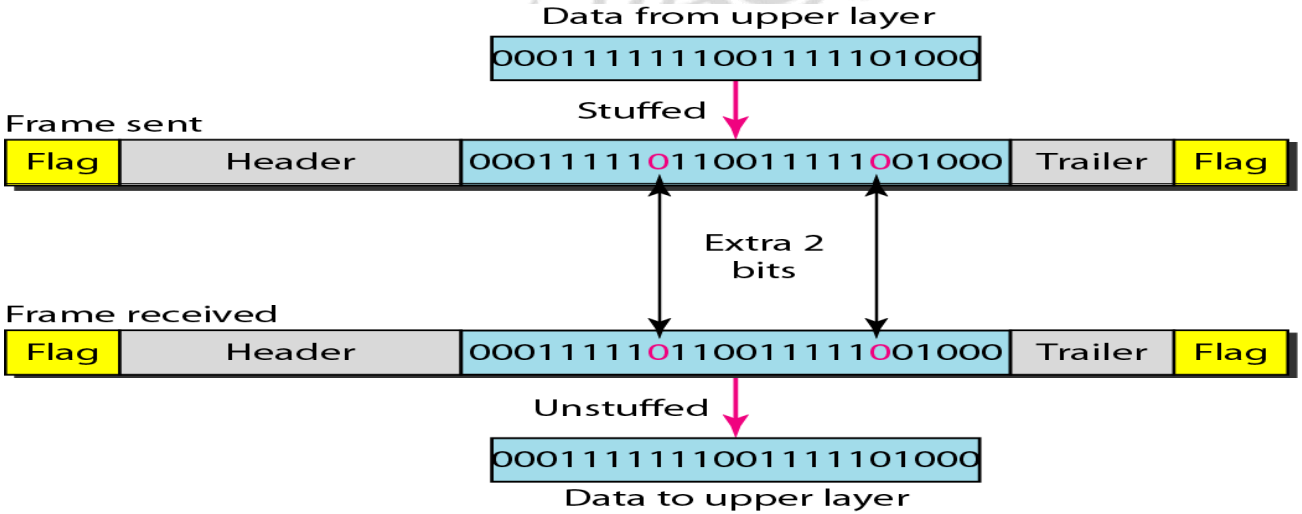
Bit-oriented protocols

- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer
- However, in addition to headers and trailers, we still need a delimiter to separate one frame from the other.
- Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame



Bit-oriented protocols

- The data may contain the special bit-pattern from the upper layer
- Bit stuffing is the process of adding one extra 0 whenever five consecutive 1's are received, so that the receiver does not mistake the pattern 01111110 for a flag



Bit-oriented protocols

- Eg:
  - Assume, we send a frame of 2 8-bit characters: char1 (01111111) and char2 (01111101)
- Sender rule
  - if 5 1s in data, then add (stuff) a 0
- On the line, we'll send

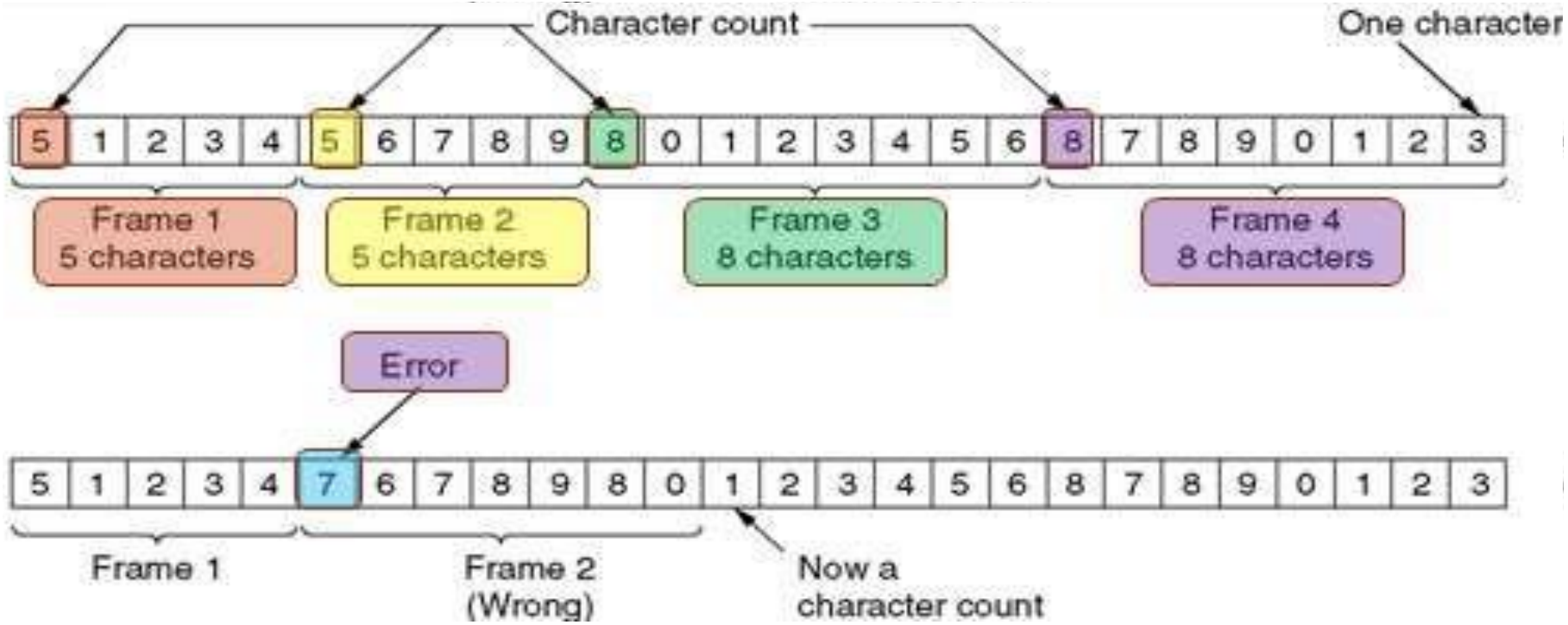
  01111110 011111011 011111001 01111110
- Receiver rule
  - if a 0 occurs after 5 1s, then remove it

  01111110 011111-11 011111-01 01111110

Framing

- The framing method uses a field in the header to specify the number of characters in the frame
- When the data link layer at the destination sees the character count, it knows how many characters to follow and hence where the end of the frame
- The trouble with this algorithm is that the count can be corrupted by a transmission error

Framing


Character count
One character
5 1 2 3 4 5 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3
Frame 1
5 characters
Frame 2
5 characters
Frame 3
8 characters
Frame 4
8 characters
Error
5 1 2 3 4 7 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3
Frame 1
Frame 2
(Wrong)
Now a
character count

ERROR DETECTION AND CORRECTION

Error Detection and Correction

- Data can be corrupted during transmission
- Some applications may be required to detect and correct the errors
- There are many reasons such as noise, cross-talk etc., which is induced in data to get corrupted during transmission
- Data-link layer uses some error control mechanism to ensure that frames (data bit streams) are transmitted with certain level of accuracy

Error

- A condition when the receiver's information does not match with the sender's information
- During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from sender to receiver
- This means a 0 may change to 1 or vice-versa
- Note: Error Detecting Codes (Implemented either at Data link layer or Transport Layer of OSI Model)

Types of Error

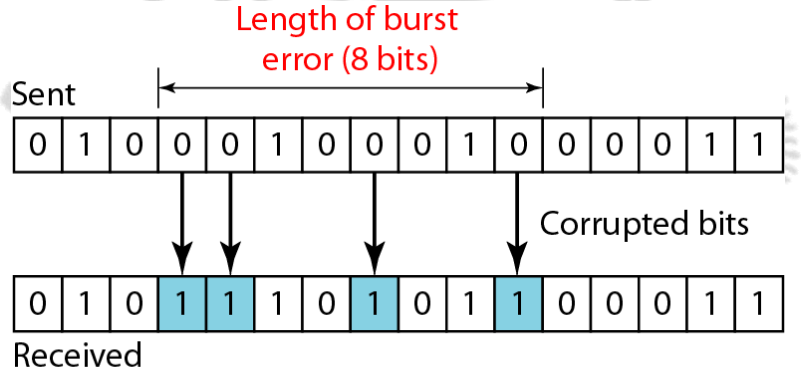
- Single bit error
  - In a frame, there is only one bit, anywhere, which is corrupt

- Multiple bits error
  - Frame is received with more than one bits corrupted

- Burst error
  - Frame contains more than1 consecutive bits corrupted

Burst Error

- The term burst error means that two or more consecutive bits in the data unit have changed from 1 to 0 or from 0 to 1
- Burst errors does not necessarily mean that the errors occur in consecutive bits, the length of the burst is measured from the first corrupted bit to the last corrupted bit Some bits in between may not have been corrupted
- NOTE: To detect or correct errors, we need to send extra (redundant) bits with data



Error control mechanism

- Error detection
- Error correction

Error control mechanism

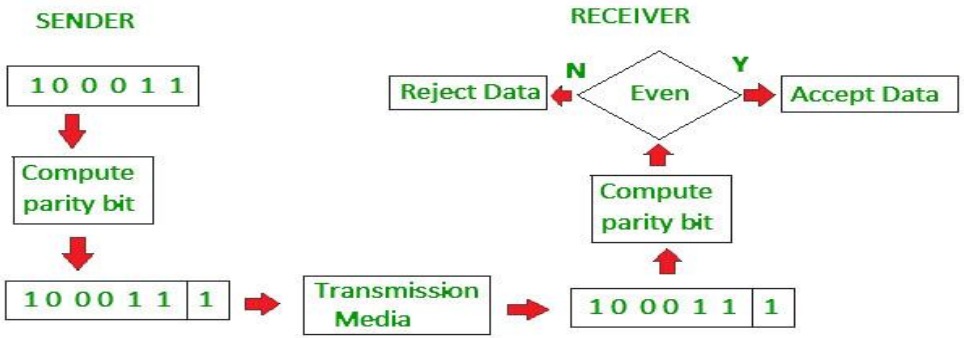
- Error detection:
  - Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted
  - To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if any error has occurred during transmission of the message
  - Basic approach used for error detection is the use of redundancy bits, where additional bits are added to facilitate detection of errors

- Some popular techniques for error detection are:
  - Simple parity check
  - Two-dimensional parity check
  - Checksum
  - Cyclic redundancy check

Error control mechanism

Simple Parity check

- Blocks of data from the source are subjected to a check bit or parity bit generator, where a parity of:
  - 1 is added to the block if it contains odd number of 1's, and
  - 0 is added if it contains even number of 1's
- This scheme makes the total number of 1's even, that is why it is called even parity checking



Error control mechanism

Two Parity check

- Parity check bits are calculated for each row, which is equivalent to a simple parity check bit
- Parity check bits are also calculated for all columns, then both are sent along with the data
- At the receiving end these are compared with the parity bits calculated on the received data

Error control mechanism

Original Data

10011001	11100010	00100100	10000100

	Row parities
10011001	0
11100010	0
00100100	0
10000100	0
Column parities → 11011011	0

100110010	111000100	001001000	100001000	110110110

Data to be sent

Error control mechanism

Check Sum

- Error detecting technique
- Applied to a message of any length
- Mostly used at network and transport layer rather to data link layer

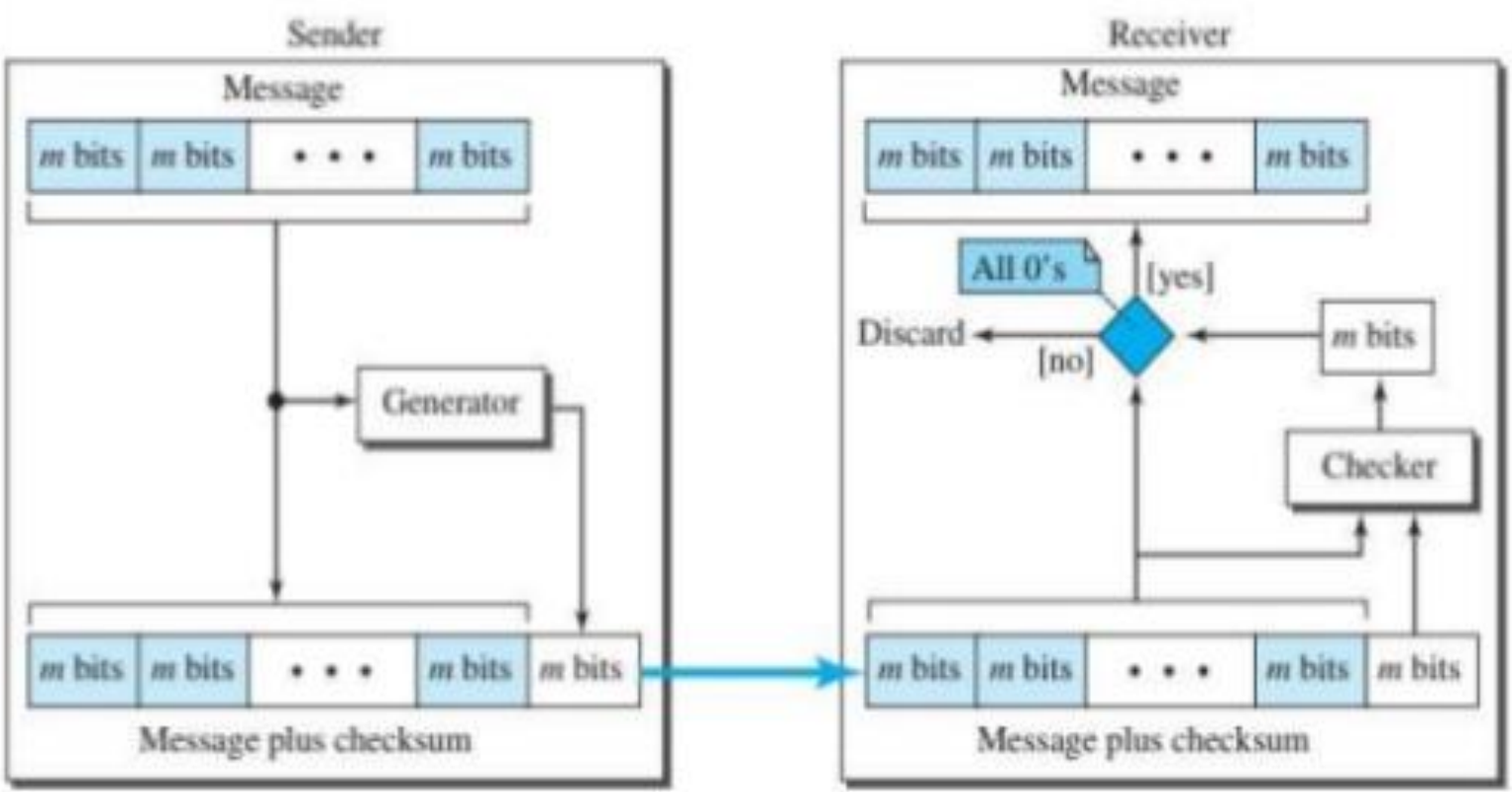
Error control mechanism

Check Sum - Approach

- At the source, the message is divided into m-bits unit
- The generator than creates a checksum (an extra m-bit unit)

- At the destination, the checker creates a new checksum from the combination of message and sent checksum
- If checksum is all 0s
  - Message is accepted
  - Else
  - Message is discarded

Error control mechanism



Error control mechanism

- Eg:
  - if the set of numbers is (7, 11, 12, 0, 6)
  - we send (7, 11, 12,0,6,36), where 36 is the sum of the original numbers
  - The receiver adds the five numbers and compares the result with the sum
  - If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum
  - Otherwise, there is an error somewhere and the data are not accepted

Error control mechanism

- Eg:
  - Suppose that the sender wants to send 4 frames each of 8 bits, where the frames are 11001100, 10101010, 11110000 and 11000011

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011
Sum:	11111111
Complement:	00000000
Hence accept frames.	

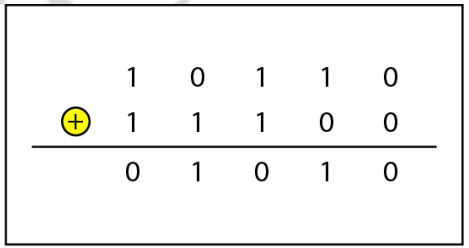
Error control mechanism

Cyclic Redundancy Check (CRC)

- CRC is a method of detecting accidental changes/errors in the communication channel
- CRC is used in networks such as LANs and WANs
- CRC involves binary division of the data bits being sent by a predetermined divisor agreed upon by the communicating system
- The divisor is generated using polynomials so, CRC is also called polynomial code checksum

a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.



c. Result of XORing two patterns

Error control mechanism

Cyclic Redundancy Check (CRC) - Approach

- Sender Side
  - The binary data is first augmented by adding k-1 zeros in the end of the data
  - Use modulo-2 binary division to divide binary data by the key and store remainder of division
  - Append the remainder at the end of the data to form the encoded data and send the same

- n : Number of bits in data to be sent from sender side
- k : Number of bits in the key obtained from generator polynomial

Error control mechanism

Cyclic Redundancy Check (CRC) - Approach

- Receiver Side
  - Perform modulo-2 division again and if the remainder is 0, then there are no errors

- Modulo 2 Division: The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers.
- Instead of subtraction, we use XOR here

Error control mechanism

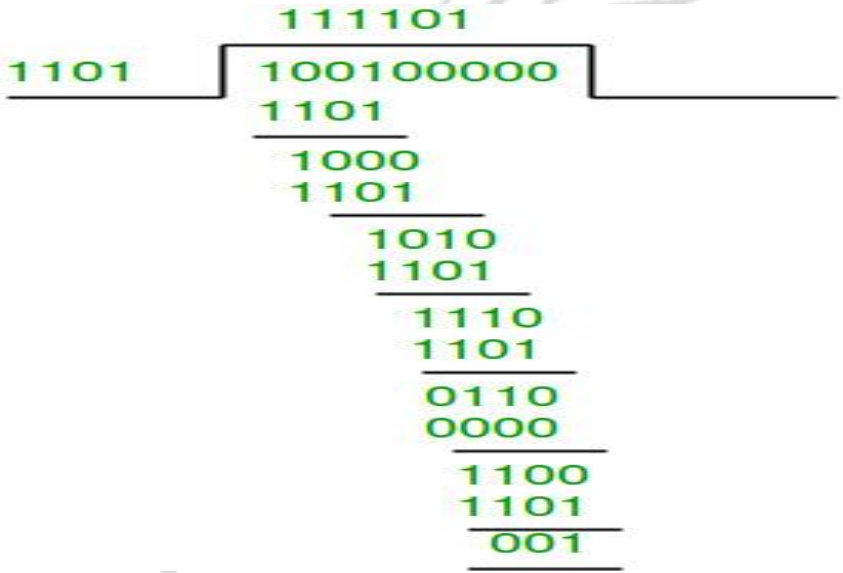
Cyclic Redundancy Check (CRC) - Approach

- Case - 01: Remainder = 0
- If the remainder is zero
- Receiver assumes that no error occurred in the data during the transmission
- Receiver accepts the data

- Case - 02: Remainder ≠ 0
- If the remainder is non-zero
- Receiver assumes that some error occurred in the data during the transmission
- Receiver rejects the data and asks the sender for retransmission

Error control mechanism

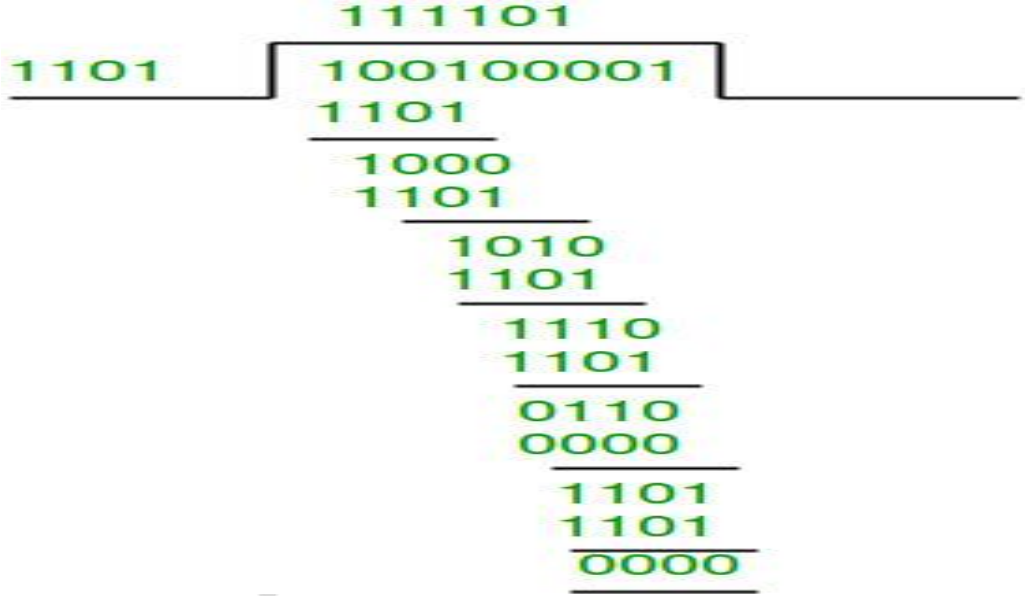
- Eg:
  - Data word to be sent - 100100 Key - 1101 [Or generator polynomial $x^3 + x^2 + 1$]



- The remainder is 001 and hence the encoded data sent is 100100001

Error control mechanism

- Code word received at the receiver side 100100001



- The remainder is all zeros. Hence, the data received has no error

Hamming Code

- Hamming code is a set of error-correction codes that can be used to detect and correct the errors
- Hamming codes are linear block code
- Commonly 7-bit hamming code is used

Hamming Code

- Redundant bits
  - Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer
- The number of redundant bits can be calculated using the following formula:
  - $2^r \geq m + r + 1$
  - where, r = redundant bit, m = data bit
- Suppose the number of data bits are 7, then the number of redundant bits can be calculated using:
- $2^4 \geq 7 + 4 + 1$
- Thus, the number of redundant bits = 4

Hamming Code

- General algorithm
  - Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc)
  - All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc)
  - Each data bit is included in a unique set of parity bits, as determined its bit position in binary form

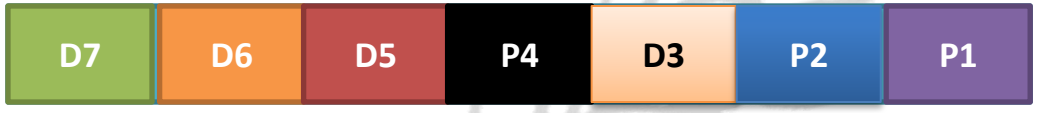
Hamming Code

- Selection of parity bits

- Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7 etc)
- Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7 etc)
- Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4,5,6,7 etc)

Hamming Code

- Eg:
  - A bit word 1011 is to be transmitted construct even parity 7 bit hamming code.
- Step-1



- P1 = 1,3,5,7 = [P1,111] as nos. of 1's are odd so P1 will be set to 1, [1111]
- P2 = 2,3,6,7 = [P2,101] as nos. of 1's are even so P2 will be set to 0, [0101]
- P4 = 4,5,6,7 = [P4,101] as nos. of 1's are even so P3 will be set to 0, [0101]
- So, required 7 bit hamming code is 1010101

Detection of Error in Hamming Code

- At Receiver end decoded data is checked for the following groups;
- Group-1[1,3,5,7]
- Group-2[2,3,6,7]
- Group-3[4,5,6,7]
- If all group contains even parity then there is no error

Error Correction in Hamming code

- Eg:
  - Codeword received 1011011,assume even parity. State whether received codeword is correct or wrong. Locate the bit which has error

Solution

- 1(D7) 0(D6) 1(D5) 1(P4) 0(D3) 1(P2) 1(P1)
- Check for error for even parity
- Data at 1,3,5,7=1011
- Data at 2,3,6,7=1001
- Data at 4,5,6,7=1101

- Write Error : P4 P2 P1
- 1 0 1
- 101=5, so error is in 5th bit that means 5th bit is 1,must be 0 to be corrected

Hamming distance

- The Hamming distance between two words is the number of differences between corresponding bits
- Let us find the Hamming distance between two pairs of words
  - The hamming distance d(000, 011) is 2

$$000 \oplus 011 \text{ is } 011 \text{ (two 1s)}$$

  - The hamming distance d(10101, 11110) is 3

$$10101 \oplus 11110 \text{ is } 01011 \text{ (three 1s)}$$

Hamming distance

- The minimum hamming distance is the smallest hamming distance between all possible pairs in a set of words

Hamming distance

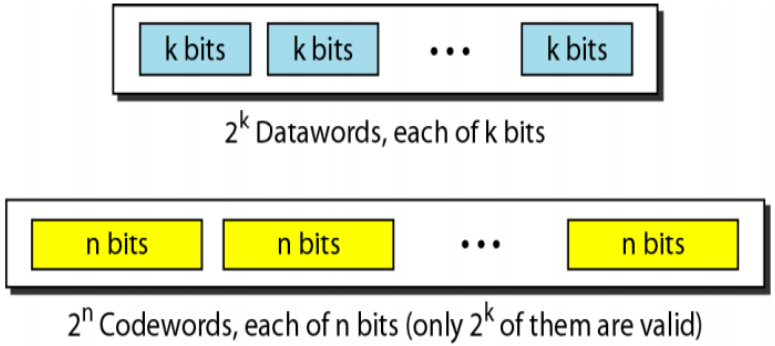
- The minimum hamming distance is the smallest hamming distance between all possible pairs in a set of words
- Eg:
  - Find the minimum hamming distance

$d(00000, 01011) = 3$	$d(00000, 10101) = 3$	$d(00000, 11110) = 4$
$d(01011, 10101) = 4$	$d(01011, 11110) = 3$	$d(10101, 11110) = 3$

The d_{min} in this case is 3.

Block coding

- In block coding, we divide our message into blocks, each of k bits, called datawords
- We add r redundant bits to each block to make the length n = k + r
- The resulting n-bit blocks are called codewords
- Block coding is normally referred as mb/nb coding
- It replaces each m-bit group with n-bit group



Process of error detection in block coding