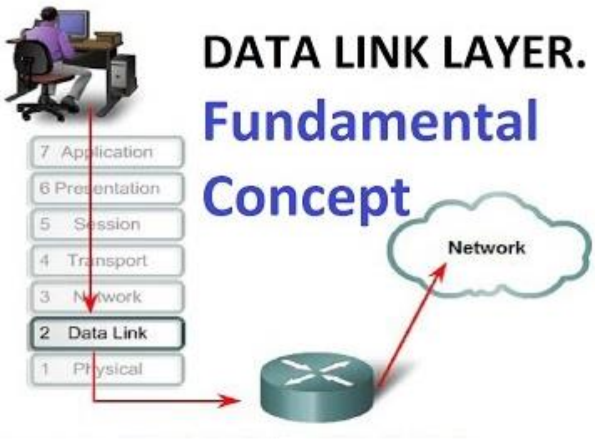
MODULE-3: Data Link Layer

VIT | Vidyalankar Institute of Technology

Accredited A+ by NAAC

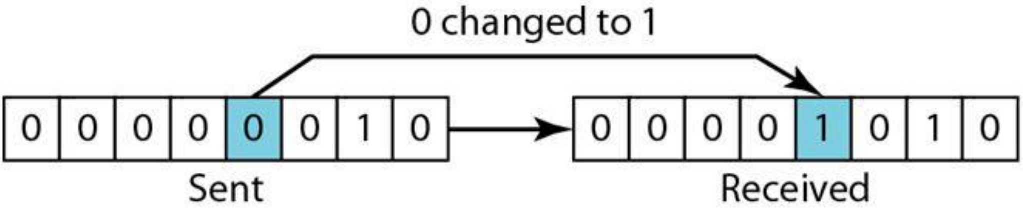


Prepared by Prof. Amit K. Nerurkar

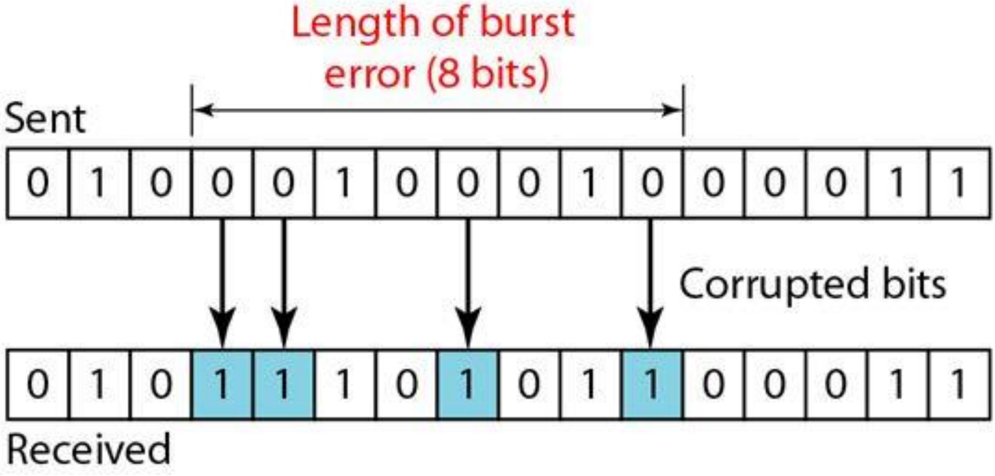
Error Detection

Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference. This interference can change the shape of the signal.

Single-Bit Error: The term single-bit error means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.



Burst Error: The term burst error means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.



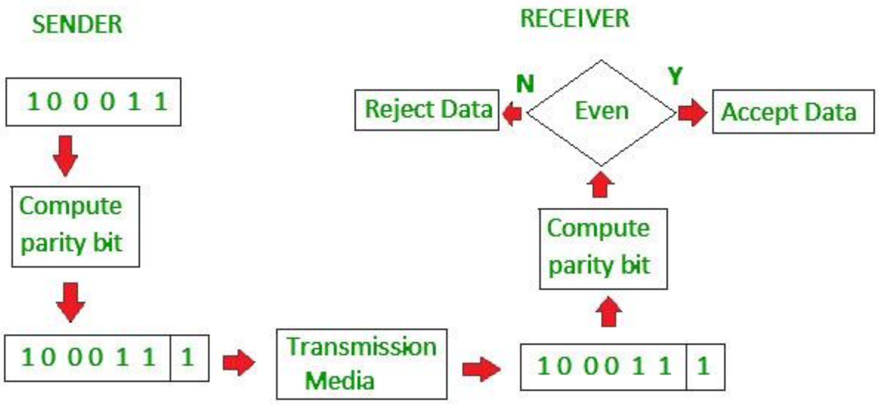
Error detection

Error detection involves checking whether any error has occurred or not. The number of error bits and the type of error does not matter. There are three main techniques for detecting errors in frames:

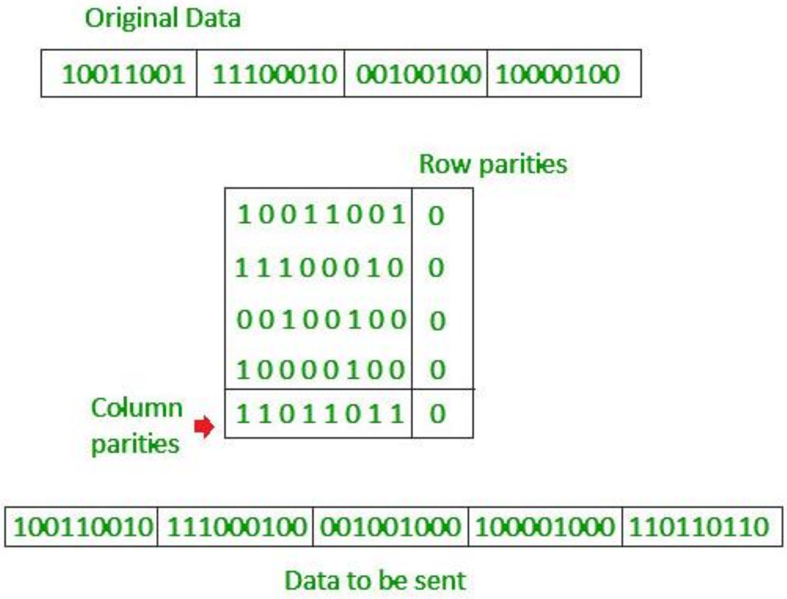
1. Parity Check
2. Checksum
3. Cyclic Redundancy Check (CRC).

1. Parity Check

The parity check is done by adding an extra bit, called parity bit to the data to make a number of 1s either even in case of even parity or odd in case of odd parity.



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.



10011001	11100010	00100100	10000100

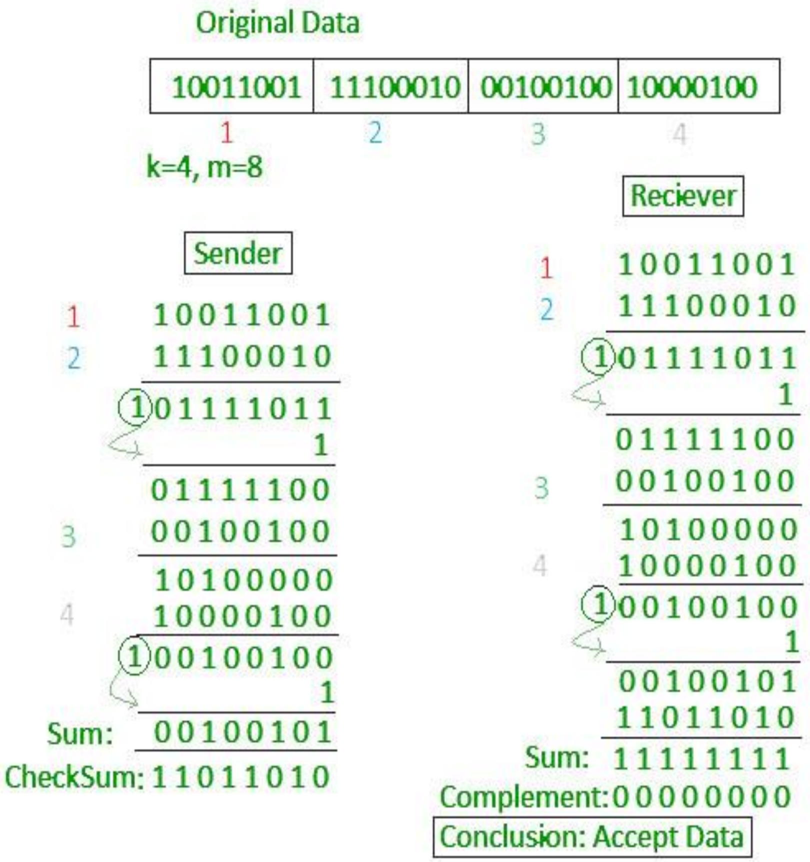
10011001	0
11100010	0
00100100	0
10000100	0
11011011	0

100110010	111000100	001001000	100001000	110110110

2. Checksum

In this error detection scheme, the following procedure is applied

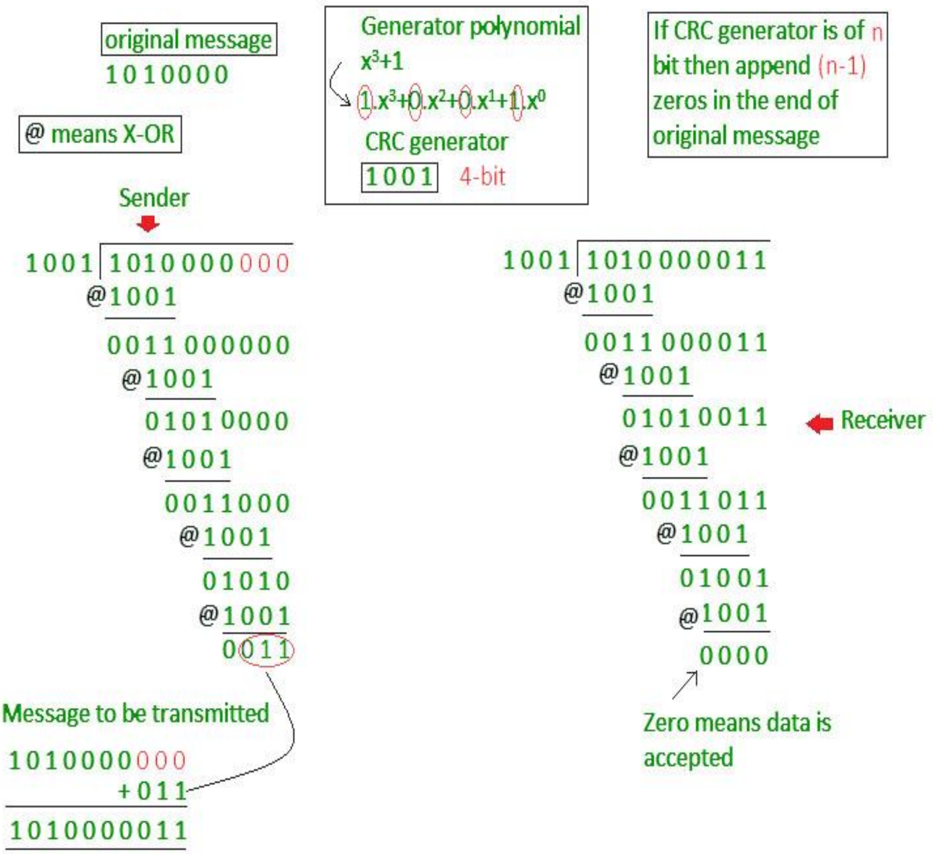
- Data is divided into fixed sized frames or segments.
- The sender adds the segments using 1's complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.
- The receiver adds the incoming segments along with the checksum using 1's complement arithmetic to get the sum and then complements it.
- If the result is zero, the received frames are accepted; otherwise, they are discarded.



3. Cyclic Redundancy Check (CRC).

Cyclic Redundancy Check (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.

- Here, the sender performs binary division of the data segment by the divisor. It then appends the remainder called CRC bits to the end of the data segment. This makes the resulting data unit exactly divisible by the divisor.
- The receiver divides the incoming data unit by the divisor. If there is no remainder, the data unit is assumed to be correct and is accepted. Otherwise, it is understood that the data is corrupted and is therefore rejected.



For more details visit: http://www2.ic.uff.br/~michael/kr1999/5-datalink/5_02-ec.htm

Error Detection and Correction

Error correction techniques find out the exact number of bits that have been corrupted and as well as their locations.

Hamming Code

Hamming code is a set of error-correction codes that can be used to **detect and correct the errors** that can occur when the data is moved or stored from the sender to the receiver. It is **technique developed by R.W. Hamming for error correction**.

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

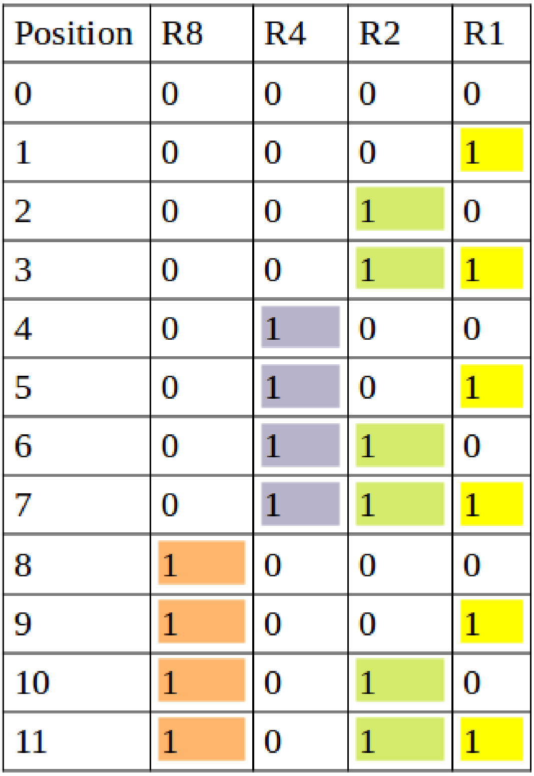
Suppose the number of data bits is 7, then the number of redundant bits can be calculated using:
= $2^4 \geq 7 + 4 + 1$
Thus, the number of redundant bits= 4

General Algorithm of Hamming code –
The Hamming Code is simply the use of extra parity bits to allow the identification of an error.

1. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
2. All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
3. All the other bit positions are marked as data bits.
4. Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.
   a. Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7, 9, 11, etc).
   b. 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, 10, 11, etc).
   c. Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4–7, 12–15, 20–23, etc).

d. Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit bits (8–15, 24–31, 40–47, etc).
e. In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.

5. Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is odd.
6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

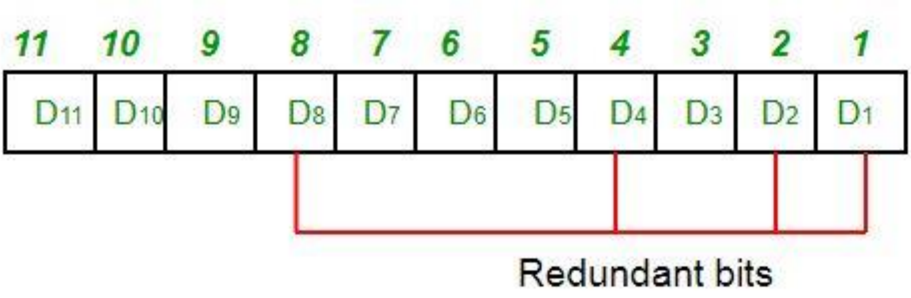


Position	R8	R4	R2	R1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

Determining the position of redundant bits –
These redundancy bits are placed at the positions which correspond to the power of 2.
As in the above example:

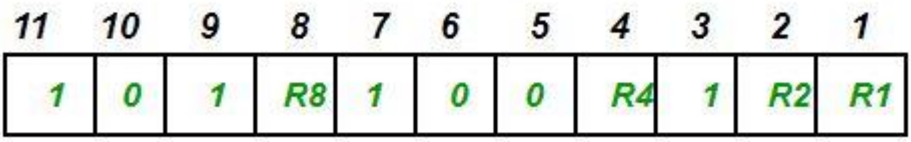
1. The number of data bits = 7

2. The number of redundant bits = 4
3. The total number of bits = 11
4. The redundant bits are placed at positions corresponding to power of 2- 1, 2, 4, and 8



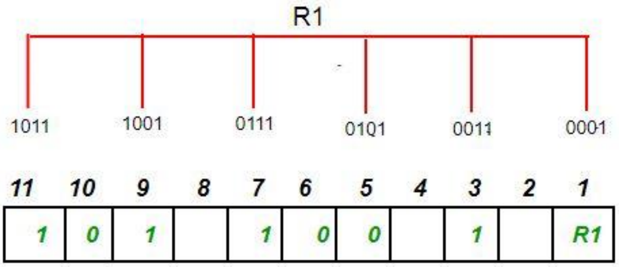
11	10	9	8	7	6	5	4	3	2	1
D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1

Suppose the data to be transmitted is 1011001, the bits will be placed as follows:



11	10	9	8	7	6	5	4	3	2	1
1	0	1	R8	1	0	0	R4	1	R2	R1

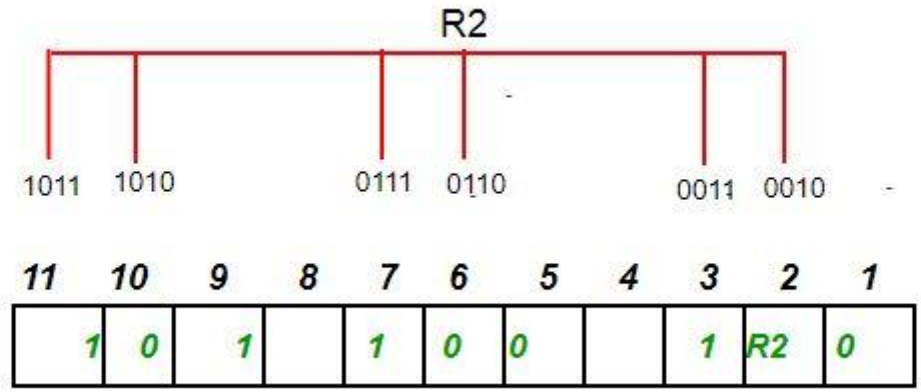
1. R1 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the least significant position.
R1: bits 1, 3, 5, 7, 9, 11



11	10	9	8	7	6	5	4	3	2	1
1	0	1		1	0	0		1		R1

To find the redundant bit R1, we check for even parity. Since the total number of 1’s in all the bit positions corresponding to R1 is an even number the value of R1 (parity bit’s value) = 0

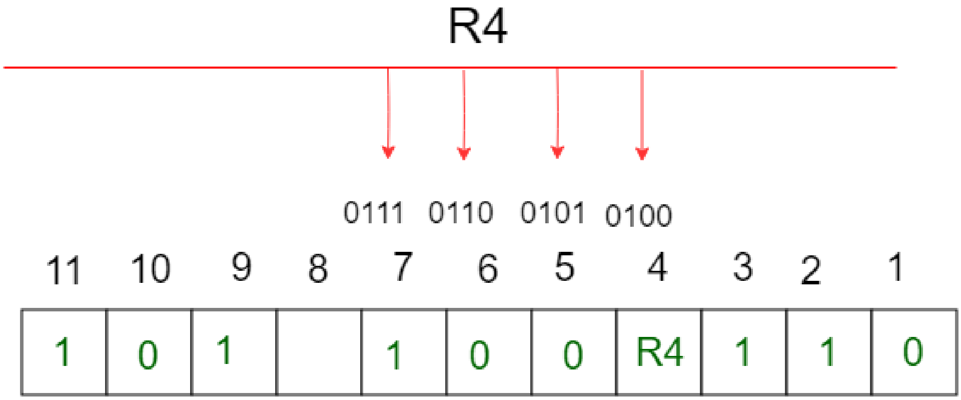
2. R2 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the second position from the least significant bit.
R2: bits 2,3,6,7,10,11



11	10	9	8	7	6	5	4	3	2	1
1	0	1		1	0	0		1	R2	0

To find the redundant bit R2, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R2 is odd the value of R2(parity bit's value)=1

3.R4 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the third position from the least significant bit.
R4: bits 4, 5, 6, 7

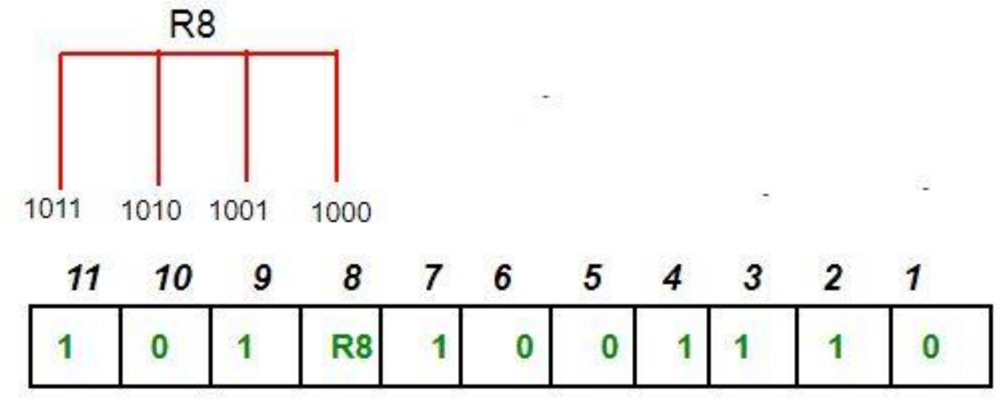


11	10	9	8	7	6	5	4	3	2	1
1	0	1		1	0	0	R4	1	1	0

To find the redundant bit R4, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R4 is odd the value of R4(parity bit's value) = 1

4. R8 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit.

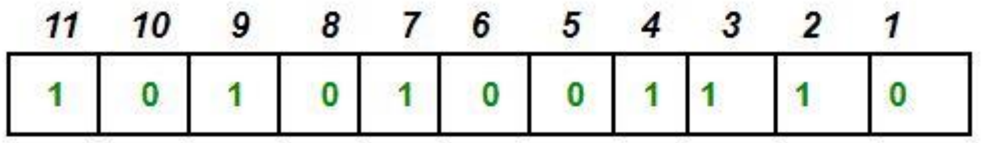
R8: bit 8,9,10,11



11	10	9	8	7	6	5	4	3	2	1
1	0	1	R8	1	0	0	1	1	1	0

To find the redundant bit R8, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R8 is an even number the value of R8(parity bit's value)=0.

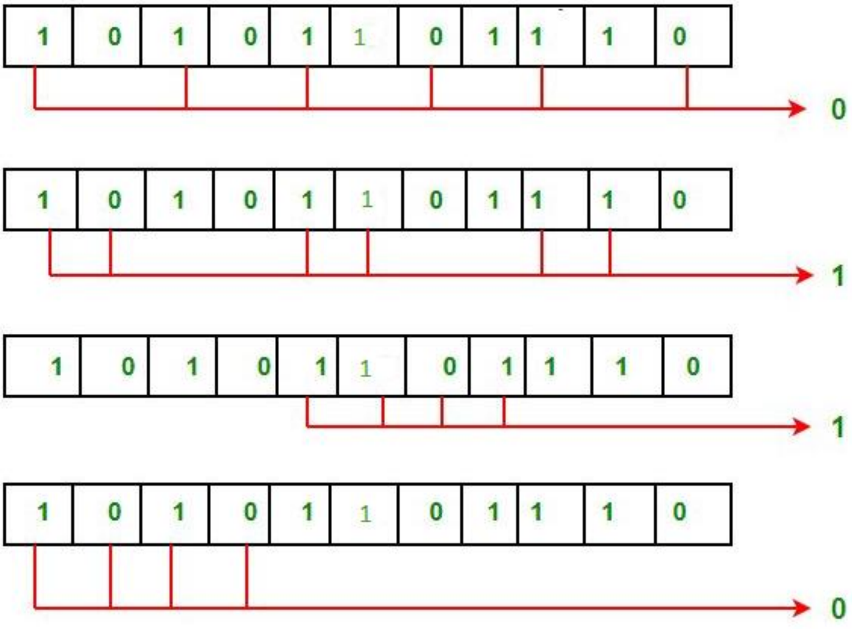
Thus, the data transferred is:



11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	1	0	0	1	1	1	0

Error detection and correction –

Suppose in the above example the 6th bit is changed from 0 to 1 during data transmission, then it gives new parity values in the binary number:



The bits give the binary number as 0110 whose decimal representation is 6. Thus, the bit 6 contains an error. To correct the error the 6th bit is changed from 1 to 0.

Video

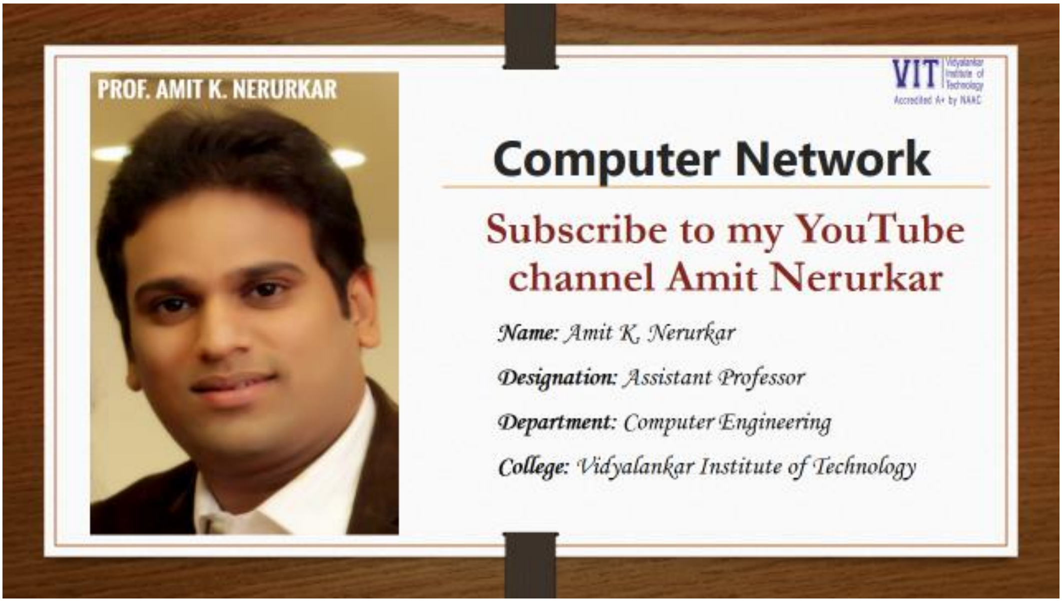
Hamming Code Generation Example with Even Parity

* HAMMING CODE GENERATION *
• Construct (7,4) hamming code for the message (1000)
Consider even parity

References

1. TCP/IP Protocol Suite by Fourozan
2. Computer Networks by Tanenbaum
3. http://www.myreadingroom.co.in/notes-and-studymaterial/68-dcn/797-types-of-errors.html
4. https://www.tutorialspoint.com/error-detection-and-correction-in-data-link-layer
5. https://www.geeksforgeeks.org/error-detection-in-computer-networks/

Subscribe to my YouTube Channel Amit Nerurkar



PROF. AMIT K. NERURKAR

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

Computer Network

Subscribe to my YouTube channel Amit Nerurkar

Name: Amit K. Nerurkar

Designation: Assistant Professor

Department: Computer Engineering

College: Vidyalankar Institute of Technology

Subjects Taught by Amit K. Nerurkar

1. C programming
2. Data Structure
3. Computer Network
4. Network Security
5. Artificial Intelligence
6. Soft Computing
7. Distributed Systems
8. Internet of Things
9. Linux Administration
10. Database Management System