**Computer Networks Lab**

**List of Programs**

1) Implement the data link layer framing methods such as character stuffing and bit stuffing.

2) Write a C program to develop a DNS client server to resolve the given hostname.

3) Implement on a data set of characters the three CRC polynomials – CRC-12, CRC-16 and CRC-CCIP.

4) Implement Dijkstra’s algorithm to compute the Shortest path in a graph.

5) Take an example subnet graph with weights indicating delay between nodes. Now obtain Routing table art each node using distance vector routing algorithm

6) Take an example subnet of hosts. Obtain broadcast tree for it.

7) Write a client-server application for chat using UDP

8) Implement programs using raw sockets (like packet capturing and filtering)

9) Write a C program to perform sliding window protocol.

10) Get the MAC or Physical address of the system using Address Resolution Protocol.

11) Simulate the Implementing Routing Protocols using border gateway protocol(BGP)

12) Simulate the OPEN SHORTEST PATH FIRST routing protocol based on the cost assigned to the path.

1. Implement the data link layer framing methods such as character stuffing and bit stuffing.

**Character stuffing**

#include<stdio.h>

#include<string.h>

main()

{

char a[30], fs[50] = " ", t[3], sd, ed, x[3], s[3], d[3], y[3];

int i, j, p = 0, q = 0;

clrscr();

printf("Enter characters to be stuffed:");

scanf("%s", a);

printf("\nEnter a character that represents starting delimiter:");

scanf(" %c", &sd);

fflush(stdin);

printf("\nEnter a character that represents ending delimiter:");

scanf(" %c", &ed);

x[0] = s[0] = s[1] = sd;

x[1] = s[2] = '\0';

y[0] = d[0] = d[1] = ed;

d[2] = y[1] = '\0';

strcat(fs, x);

for(i = 0; i < strlen(a); i++)

{

t[0] = a[i];

t[1] = '\0';

if(t[0] == sd)

strcat(fs, s);

else if(t[0] == ed)

strcat(fs, d);

else

strcat(fs, t);

}

strcat(fs, y);

printf("\n After stuffing:%s", fs);

getch();

}

Input:

Enter characters to be stuffed: goodday

Enter a character that represents starting delimiter: g

Enter a character that represents ending delimiter: y

Output: gggooddayyy

**B) Bit stuffing:**

#include<stdio.h>

#include<string.h>

void main()

{

int a[20],b[30],i,j,k,count,n;

clrscr();

printf("Enter frame size (Example: 8):");

scanf("%d",&n);

printf("Enter the frame in the form of 0 and 1 :");

for(i=0; i<n; i++)

scanf("%d",&a[i]);

i=0;

count=1;

j=0;

while(i<n)

{

if(a[i]==1)

{

b[j]=a[i];

for(k=i+1; a[k]==1 && k<n && count<5; k++)

{

j++;

b[j]=a[k];

count++;

if(count==5)

{

j++;

b[j]=0;

count=0;

}

i=k;

}

}

else

{

b[j]=a[i];

}

i++;

j++;

}

printf("After Bit Stuffing :");

for(i=0; i<j; i++)

printf("%d",b[i]);

getch();

}

**Input:**

Enter frame size (Example: 8):12

Enter the frame in the form of 0 and 1 :

0

1

1

1

1

1

1

1

1

1

1

0

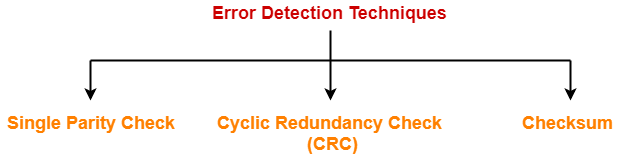
**Output:** After Bit Stuffing :

01111101111100

**Ex3: Error Detection**

|  |
| --- |
| Error detection is a technique that is used to check if any error occurred in the data during the transmission. |

Some popular error detection methods are-



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

In this article, we will discuss about Cyclic Redundancy Check (CRC).

**Cyclic Redundancy Check-**

* Cyclic Redundancy Check (CRC) is an error detection method.
* It is based on binary division.

**CRC Generator-**

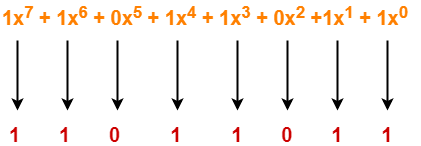
* CRC generator is an algebraic polynomial represented as a bit pattern.
* Bit pattern is obtained from the CRC generator using the following rule-

|  |
| --- |
| The power of each term gives the position of the bit and the coefficient gives the value of the bit. |

**Example-**

Consider the CRC generator is x7 + x6 + x4 + x3 + x + 1.

The corresponding binary pattern is obtained as-



 Thus, for the given CRC generator, the corresponding binary pattern is 11011011.

**Properties Of CRC Generator-**

The algebraic polynomial chosen as a CRC generator should have at least the following properties-

**Rule-01:**

* It should not be divisible by x.
* This condition guarantees that all the burst errors of length equal to the length of polynomial are detected.

**Rule-02:**

* It should be divisible by x+1.
* This condition guarantees that all the burst errors affecting an odd number of bits are detected.

**Important Notes-**

If the CRC generator is chosen according to the above rules, then-

* CRC can detect all single-bit errors
* CRC can detect all double-bit errors provided the divisor contains at least three logic 1’s.
* CRC can detect any odd number of errors provided the divisor is a factor of x+1.
* CRC can detect all burst error of length less than the degree of the polynomial.
* CRC can detect most of the larger burst errors with a high probability.

**Steps Involved-**

Error detection using CRC technique involves the following steps-

**Step-01: Calculation Of CRC At Sender Side-**

At sender side,

* A string of n 0’s is appended to the data unit to be transmitted.
* Here, n is one less than the number of bits in CRC generator.
* Binary division is performed of the resultant string with the CRC generator.
* After division, the remainder so obtained is called as **CRC**.
* It may be noted that CRC also consists of n bits.

**Step-02: Appending CRC To Data Unit-**

At sender side,

* The CRC is obtained after the binary division.
* The string of n 0’s appended to the data unit earlier is replaced by the CRC remainder.

**Step-03: Transmission To Receiver-**

* The newly formed code word (Original data + CRC) is transmitted to the receiver.

**Step-04: Checking at Receiver Side-**

At receiver side,

* The transmitted code word is received.
* The received code word is divided with the same CRC generator.
* On division, the remainder so obtained is checked.

The following two cases are possible-

**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.

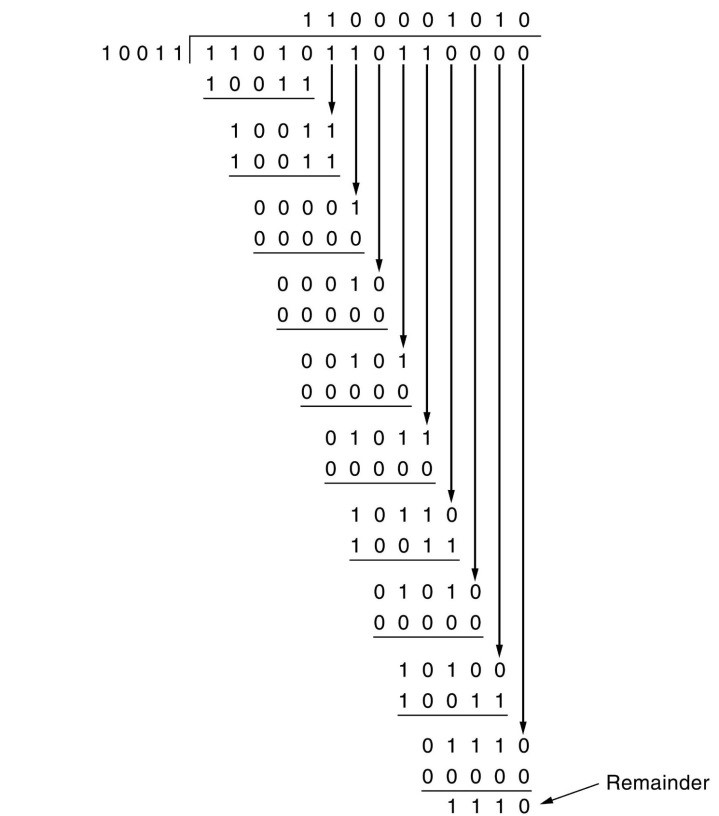
**PRACTICE PROBLEMS BASED ON CYCLIC REDUNDANCY CHECK (CRC)-**

**Example:** A bit stream 1101011011 is transmitted using the standard CRC method. The generator polynomial is x4+x+1. What is the actual bit string transmitted?

Solution-

* The generator polynomial G(x) = x4 + x + 1 is encoded as 10011.
* Clearly, the generator polynomial consists of 5 bits.
* So, a string of 4 zeroes is appended to the bit stream to be transmitted.
* The resulting bit stream is 1101011011**0000**.

Now, the binary division is performed as-



 From here, CRC = 1110.

Now,

* The code word to be transmitted is obtained by replacing the last 4 zeroes of 1101011011**0000**with the CRC.
* Thus, the code word transmitted to the receiver = 1101011011**1110**.

**Implement on a data set of characters the three CRC polynomials – CRC-12, CRC-16 and CRC-CCIP.**

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i, j,k = 0;

int flag = 1, a[16], g[16], r[20],div[16], n, m;

system("clear");

printf("Enter the degree of generator : ");

scanf("%d", &n);

printf("\nEnter the generator : \n");

for(i= 0; i<=n; i++)

scanf("%d", &g[i]);

printf("\nEnter the degree of frame : ");

scanf("%d", &m);

printf("Enter the frame : \n");

for(i = 0; i<=m; i++)

scanf("%d", &a[i]);

if(m<n || (g[0] && g[n]) == 0)

{

printf("Not a proper generator \n");

exit(0);

}

for(i = m+1; i<=m+n; i++)

a[i] = 0;

for(j= 0; j<=n; j++)

r[j] = a[j];

for(i= n; i<=m+n;i++)

{

if(i>n)

{

for(j = 0; j<n; j++)

r[j] = r[j+1];

r[j] = a[i];

}

if(r[0])

div[k++] = 1;

else

{

div[k++] = 0;

continue;

}

for(j= 0; j<=n; j++)

r[j] = r[j] ^ g[j] ;

}

printf("\nQuotient is : ");

for(j= 0; j<k; j++)

printf("%d ", div[j]);

printf("\nReminder is : ");

for(i= 1; i<=n; i++)

printf("%d ", r[i]);

printf("\nTransmitted frame is : ");

for(i = m+1,j= 1; i<=m+n;i++, j++)

a[i] = r[j];

for(i= 0; i<=m+n; i++)

printf("%d ", a[i]);

printf("\n");

printf("\nEnter the degree of frame : ");

scanf("%d", &m);

printf("Enter the frame : \n");

for(i = 0; i<=m; i++)

scanf("%d", &a[i]);

for(j = 0; j<=n; j++)

r[j] = a[j];

k = 0;

for(i= n; i<=m;i++)

{

if(i>n)

{

for(j = 0; j<n; j++)

r[j] = r[j+1];

r[j] = a[i];

}

if(r[0])

div[k++] = 1;

else

{

div[k++] = 0;

continue;

}

for(j= 0; j<=n; j++)

r[j] = r[j] ^ g[j] ;

}

printf("\nQuotient is : ");

for(j= 0; j<k; j++)

printf("%d ", div[j]);

printf("\nReminder is : ");

for(i= 1; i<=n; i++)

printf("%d ", r[i]);

for(i= 1; i<=n; i++)

{

if(r[i])

flag = 0;

}

if(flag)

printf("\n No Error\n");

else

printf("\nError");

return 0;

}