Program No. 3: Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

Soln:

Tool Command Language (TCL) Script

```
set ns [new Simulator]
set mytrace [open example3.tr w]
$ns trace-all $mytrace
set myNAM [open example3.nam w]
$ns namtrace-all $myNAM
set n0 [$ns node]
$n0 color “magenta”
$n0 label “src1”
set n1 [$ns node]
set n2 [$ns node]
$n2 color “magenta”
$n2 color “src2”
set n3 [$ns node]
$n3 color “blue”
$n3 label “dest2”
set n4 [$ns node]
set n5 [$ns node]
$n5 color “blue”
$n5 label “dest1”
$ns make-lan “$n0 $n1 $n2 $n3 $n4” 100Mbps 100ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n4 $n5 1Mbps 1ms DropTail
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001
set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5
$ns connect $tcp0 $sink5
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp2 $sink3
```

```
set file1 [open file1.tr w]
$tcp0 attach $file1
set file2 [open file2.tr w]
$tcp2 attach $file2
$tcp0 trace cwnd_
$tcp2 trace cwnd_
proc finish { } {
global ns mytrace myNAM
$ns flush-trace
close $mytrace
close $myNAM
exec nam example3.nam &
exit 0
}
$ns at 0.1 “$ftp0 start”
$ns at 5.0 “$ftp0 stop”
$ns at 7.0 “$ftp0 start”
$ns at 0.2 “$ftp2 start”
$ns at 8.2 “$ftp2 stop”
$ns at 14.0 “$ftp0 stop”
$ns at 10.0 “$ftp2 start”
$ns at 15.0 “$ftp2 stop”
$ns at 16.0 “finish”
$ns run
```

Acknowledgement file

```
BEGIN  {
}
{
if ($6 == “cwnd_”)
printf(“%f\t%f\t\n”, $1, $7);
}
END {
}
```

Program 4. Write a program for error detecting code using CRC-CCITT (16-bits).

Soln:

```
// Include headers
#include<stdio.h>
#include<string.h>
// length of the generator polynomial
#define N strlen(gen_poly)
// data to be transmitted and received
char data[28];
// CRC value
char check_value[28];
// generator polynomial
char gen_poly[10];
// variables
int data_length,i,j;
// function that performs XOR operation
int XOR()
{
   // if both bits are the same, the output is 0
   // if the bits are different the output is 1
   for(j = 1;j < N; j++)
   check_value[j] = (( check_value[j] == gen_poly[j])?'0':'1');

}
// Function to check for errors on the receiver side
int receiver(){
```

```
// get the received data
    printf("Enter the received data: ");
    scanf("%s", data);
    printf("\n-----------------------------\n");
    printf("Data received: %s", data);
// Cyclic Redundancy Check
    crc();
// Check if the remainder is zero to find the error
    for(i=0;(i<N-1) && (check_value[i]!='1');i++);
        if(i<N-1)
            printf("\nError detected\n\n");
        else
            printf("\nNo error detected\n\n");
}
int crc( ){
    // initializing check_value
    for(i=0;i<N;i++)
        check_value[i]=data[i];
    do{
    // check if the first bit is 1 and calls XOR function
        if(check_value[0]=='1')
            XOR();
// Move the bits by 1 position for the next computation
        for(j=0;j<N-1;j++)
            check_value[j]=check_value[j+1];
        // appending a bit from data
        check_value[j]=data[i++];
    }while(i<=data_length+N-1);
// loop until the data ends
}
int main()
{
    // get the data to be transmitted
    printf("\nEnter data to be transmitted: ");
    scanf("%s",data);
    printf("\n Enter the Generating polynomial: ");
    // get the generator polynomial
    scanf("%s",gen_poly);
    // find the length of data
    data_length=strlen(data);
    // appending n-1 zeros to the data
    for(i=data_length;i<data_length+N-1;i++)
        data[i]='0';
    printf("\n----------------------------------------");
// print the data with padded zeros
    printf("\n Data padded with n-1 zeros : %s",data);
    printf("\n----------------------------------------");
// Cyclic Redundancy Check
    crc();
// print the computed check value
    printf("\nCRC or Check value is : %s",check_value);
// Append data with check_value(CRC)
    for(i=data_length;i<data_length+N-1;i++)
        data[i]=check_value[i-data_length];
    printf("\n--------------------------------------");
// printing the final data to be sent
    printf("\n Final data to be sent : %s",data);
    printf("\n--------------------------------------\n");
// Calling the receiver function to check errors
    receiver();
        return 0;
}
```