Program1

set ns [new Simulator]

set nf [open out.nam w]

$ns namtrace-all $nf

proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

$ns connect $tcp0 $sink0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

Program2:

set ns [new Simulator]

set nf [open out.nam w]

$ns namtrace-all $nf

proc finish {} {

global ns nf

$ns flush-trace

fileclose $nf

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns duplex-link $n3 $n4 1Mb 10ms DropTail

$ns duplex-link $n4 $n5 1Mb 10ms DropTail

$ns duplex-link $n5 $n0 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

$ns connect $tcp0 $sink0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

Program4

set ns [new Simulator]

$ns color 1 Blue

$ns color 2 Red

set nf [open out.nam w]

$ns namtrace-all $nf

proc finish {} {

global ns nf

$ns flush-trace

close $nf

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

$ns queue-limit $n2 $n3 10

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns duplex-link-op $n2 $n3 orient right

$ns duplex-link-op $n2 $n3 queuePos 0.5

set tcp [new Agent/TCP]

$tcp set class\_ 2

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type\_ FTP

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n3 $null

$ns connect $udp $null

$udp set fid\_ 2

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

$cbr set rate\_ 1mb

$cbr set random\_ false

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

$ns at 4.5 "$cbr stop"

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

$ns at 5.0 "finish"

puts "CBR packet size = [$cbr set packet\_size\_]"

puts "CBR interval = [$cbr set interval\_]"

$ns run

Program3

set ns [new Simulator]

set nf [open out.nam w]

$ns namtrace-all $nf

proc finish {} {

global ns nf

$ns flush-trace

fileclose $nf

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$n0 shape square

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n0 $n2 1Mb 10ms DropTail

$ns duplex-link $n0 $n3 1Mb 10ms DropTail

$ns duplex-link $n0 $n4 1Mb 10ms DropTail

$ns duplex-link $n0 $n5 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

$ns connect $tcp0 $sink0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

P1

#include <stdio.h>

#include <string.h>

int main() {

int i, length, count = 0;

char v1[8], v2[8];

printf("Write a Program to Find Hamming Distance\n");

// Input 1st String

printf("Enter 1st String : ");

scanf("%s", v1);

// Input 2nd String with validation

do {

printf("Enter 2nd String (should be of the same length as the 1st String) : ");

scanf("%s", v2);

} while (strlen(v1) != strlen(v2));

length = strlen(v1);

for (i = 0; i < length; i++) {

if (v1[i] != v2[i]) {

count++;

}

}

printf("\nHamming Distance : %d\n", count);

return 0;

}

P2

#include<stdio.h>

#include<string.h>

#define GEN\_POLY\_LENGTH strlen(gen\_poly)

char data[28];

char check\_value[28];

char gen\_poly[10];

int data\_length, i, j;

void XOR() {

for (j = 1; j < GEN\_POLY\_LENGTH; j++)

check\_value[j] = ((check\_value[j] == gen\_poly[j]) ? '0' : '1');

}

void receiver() {

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 < GEN\_POLY\_LENGTH - 1) && (check\_value[i] != '1'); i++);

if (i < GEN\_POLY\_LENGTH - 1)

printf("\nError detected\n\n");

else

printf("\nNo error detected\n\n");

}

void crc() {

for (i = 0; i < GEN\_POLY\_LENGTH; i++)

check\_value[i] = data[i];

do {

if (check\_value[0] == '1')

XOR();

for (j = 0; j < GEN\_POLY\_LENGTH - 1; j++)

check\_value[j] = check\_value[j + 1];

check\_value[j] = data[i++];

} while (i <= data\_length + GEN\_POLY\_LENGTH - 1);

}

int main() {

printf("\nEnter data to be transmitted: ");

scanf("%s", data);

printf("\nEnter the Generating polynomial: ");

scanf("%s", gen\_poly);

data\_length = strlen(data);

for (i = data\_length; i < data\_length + GEN\_POLY\_LENGTH - 1; i++)

data[i] = '0';

printf("\n----------------------------------------");

printf("\nData padded with n-1 zeros: %s", data);

printf("\n----------------------------------------");

crc();

printf("\nCRC or Check value is: %s", check\_value);

for (i = data\_length; i < data\_length + GEN\_POLY\_LENGTH - 1; i++)

data[i] = check\_value[i - data\_length];

printf("\n----------------------------------------");

printf("\nFinal data to be sent: %s", data);

printf("\n----------------------------------------\n");

// Calling the receiver function to check errors

receiver();

return 0;

}