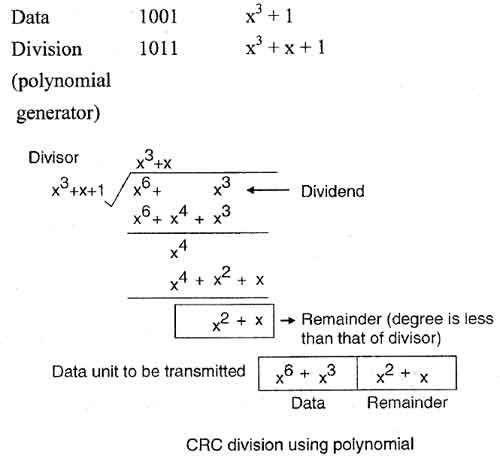
**Experiment 3. Write a program for error detecting code using CRC-CCITT (16 bits).**

CRC generator using polynomials

If we consider the data unit 1001 and divisor or polynomial generator 1011their polynomial representation is:



* Now string of n 0s (one less than that of divisor) is appended to data. Now data is 1001000 and its corresponding polynomial representation is x6 + x3.
* The division of x6+x3 by x3+x+ 1 is shown in fig.
* The polynomial generator should have following properties:

1. It should have at least two terms.
2. The coefficient of the term x0 should be 1.
3. It should not be divisible by x.
4. It should be divisible by x+ 1.

Some Standard Generator Polynomials are shown below:

|  |  |
| --- | --- |
| Name | Generator Polynomial |
| CRC – 8 | x8 + x2 + x + 1 |
| CRC – 10 | x10 + x9 + x5 +x4 + x2 + 1 |
| CRC - 16 | x16 + x12 + x5 + 1 |

**Program**

import java.util.Scanner; public class Crc {

static int data[],cs[];

static int g[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};

static int n, i, e,c,pos; static int N=17; static void xor() {

for(c=0;c<N;c++) cs[c]=((cs[c]==g[c])?0:1);

}

static void crc() {

for(i=0;i<N;i++) cs[i]=data[i];

do {

if(cs[0]==1) xor();

for(c=0;c<N-1;c++)

cs[c]=cs[c+1];

cs[c]=data[i++];

}while(i<=n+N-1);

}

public static void main(String[] args) { cs=new int[100];

Scanner br=new Scanner(System.in); System.out.println("Enter no of Data bits"); n=br.nextInt();

data=new int[100]; System.out.println("\nEnter the data bits : "); for(int i=0;i<n;i++)

data[i]=br.nextInt(); System.out.println("\n\nCRC Divisor : ");

for(int i=0;i<N;i++)

System.out.print(g[i]); for(i=n;i<n+N-1;i++)

data[i]=0; System.out.println("\n\nModified Data is : for(i=0;i<n+N-1;i++)

System.out.print(data[i]);

crc();

System.out.println("\n\nCRC Checksum is : "); for(int i=0;i<N-1;i++)

System.out.print(cs[i]); for(i=n;i<n+N-1;i++) data[i]=cs[i-n];

System.out.println("\n\nFinal Codeword is :"); for(i=0;i<n+N-1;i++)

System.out.print(data[i]); System.out.println("\n\nTest Error detection 0(yes) 1(no) ? : "); e=br.nextInt();

if(e==0) {

System.out.println("Enter position where error is to inserted : "); pos=br.nextInt();

data[pos]=(data[pos]==0)?1:0; System.out.println("\nErroneous data for(i=0;i<n+N-1;i++)

System.out.print(data[i]);

}

crc();

System.out.println("\n\nReceiver Checksum:"); for(int i=0;i<N;i++)

System.out.print(cs[i]); for(i=0;i<N-1;i++)

{

if(cs[i]!=0)

{

System.out.println("\n\nERROR in Received Codeword "); System.exit(0);

}

}

System.out.println("\nNo Error in Received Codeword");

}

}

/\*Output of cyclic redundancy check program

**OUTPUT:**

$ gedit Crc.java

$ javac Crc.java

$ java Crc

Enter no of Data bits 4

Enter the data bits :

1 0 0 1

CRC Divisor : 10001000000100001

Modified Data is : 10010000000000000000

CRC Checksum is : 1001000100101001

Final Codeword is : 10011001000100101001

Test Error detection 0(yes) 1(no) ? : 1

Receiver Checksum: 00000000000000000

No Error in Received Codeword Enter no of Data bits

4

Enter the data bits : 1 0 0 1

CRC Divisor : 10001000000100001

Modified Data is :

10010000000000000000

CRC Checksum is :

1001000100101001

Final Codeword is :

10011001000100101001

Test Error detection 0(yes) 1(no) ? :

0

Enter position where error is to inserted :

2

Erroneous data 10111001000100101001

Receiver Checksum:

00100000010000100

ERROR in Received Codeword

**EXPERIMENT 4**

**TITLE**

Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the throughput with respect to transmission of packets.

**AIM**

To understand how Extended Service Set is created and works by providing services to nodes in wireless LAN and analyzing the wireless traffic for determining packet drops.

**DESCRIPTION**

In this experiment, network simulator2 is used to create a IEEE 802.11 Wireless LAN consisting of mobile nodes and Extended Service set. An extended service set (ESS) is one or more interconnected basic service sets (BSSs) and their associated LANs. Each BSS consists of a single access point (AP) together with all wireless client devices (stations, also called STAs) creating a local or enterprise 802.11 wireless LAN (WLAN). Wireless mobile nodes

**INPUT**

* Two TCP traffic between pair of nodes
* Code to analyze the out.tr trace log file for determining the number of packets dropped.

**EXPECTED OUTPUT**

* Generate the out.tr trace log file, and
* Generate the out.nam network animation file depicting the IEEE 802.11 Wireless LAN.

**STEPS**

* Open NSG2select the type of connection -wireless scenario
* Create n nodes by clicking on the screen by selecting the node
* Click on agent tab
  + for the source node=>select agent type “UDP”

=>enter packet size in bytes

=>click on the source node and drag

* + for the destination node=>select agent type “NULL”

=>enter packet size in bytes

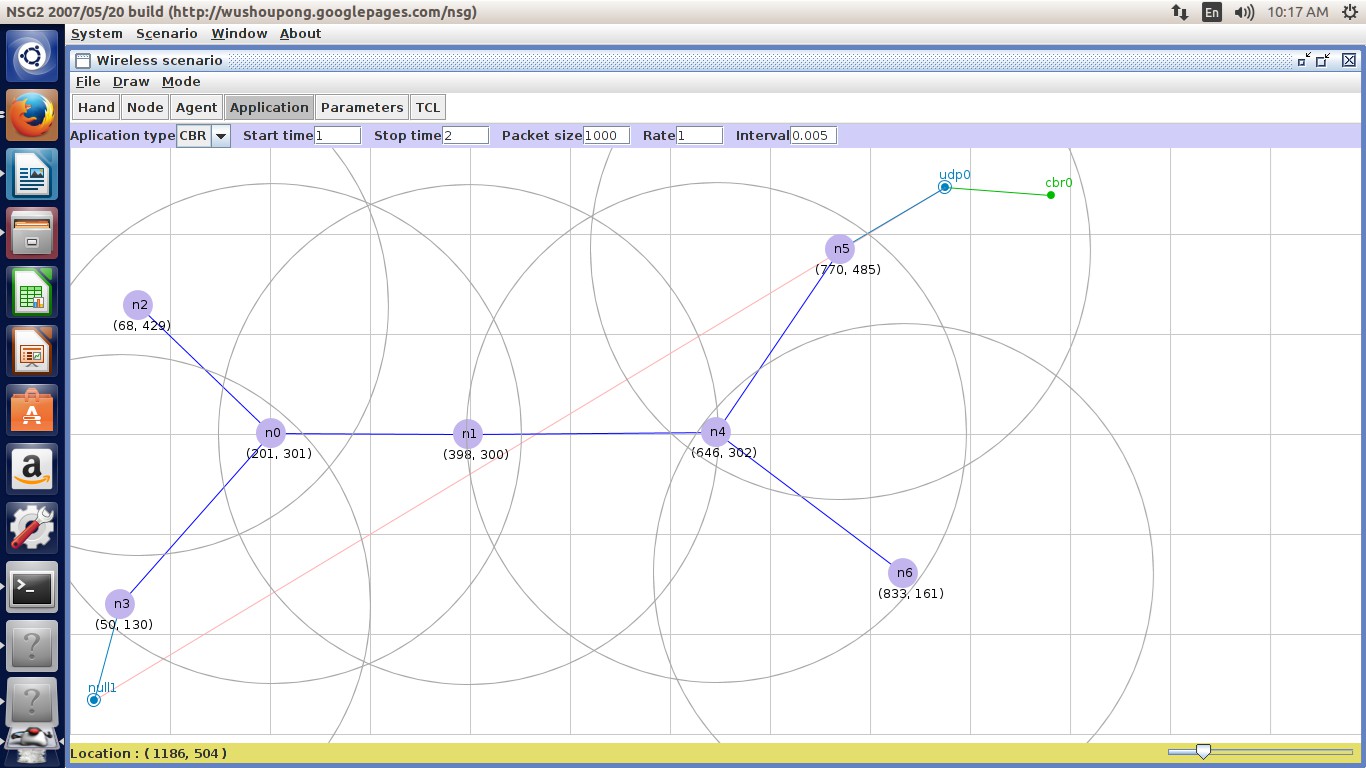
=>click on the destination node and drag

* Create TCP connections between source(n5) and destination(n3)
* Connect source to destination node by dragging UDP to NULL (Virtual connection)
* Click on application tab =>select application type(CBR)

=>enter start time and stop time

* Click on parameter tab =>enter simulation time, trace file and nam file
* Click on TCL tab to generate code
* Save the code with the file name with extension .tcl
* Edit the highlighted code
* Run the code in terminal by typing **ns filename.tcl**
* It would generate an animated topology window where transmission of packets between nodes can be viewed
* Now press the play button in topology window and the simulation begins

**NETWORK TOPOLOGY**



**Figure 10:NSG2 code generator screen**

**PROGRAM**

**# This script is created by NSG2 beta1**

**# <**[**http://wushoupong.googlepages.com/nsg>**](http://wushoupong.googlepages.com/nsg)

**#===================================**

**# Simulation parameters setup #===================================**

**set val(chan) Channel/WirelessChannel; # channel type**

**set val(prop) Propagation/TwoRayGround; # radio-propagation model set val(netif) Phy/WirelessPhy; # network interface type**

**set val(mac) Mac/802\_11; # MAC type**

**set val(ifq) Queue/DropTail/PriQueue; # interface queue type set val(ll) LL ;# link layer type**

**set val(ant) Antenna/OmniAntenna ;# antenna model set val(ifqlen) 50 ;# max packet in ifq**

**set val(nn) 6 ;# number of mobilenodes set val(rp) DSDV ;# routing protocol**

**set val(x) 810 ;# X dimension of topography**

**set val(y) 600 ;# Y dimension of topography**

**set val(stop) 10.0 ;# time of simulation end**

**#===================================**

**# Initialization #===================================**

**#Create a ns simulator set ns [new Simulator]**

**#Setup topography object**

**set topo [new Topography]**

**$topo load\_flatgrid $val(x) $val(y) create-god $val(nn)**

**#Open the NS trace file**

**set tracefile [open out.tr w]**

**$ns trace-all $tracefile**

**#Open the NAM trace file set namfile [open out.nam w]**

**$ns namtrace-all $namfile**

**$ns namtrace-all-wireless $namfile $val(x) $val(y) set chan [new $val(chan)];#Create wireless channel**

**#===================================**

**# Mobile node parameter setup #===================================**

**$ns node-config -adhocRouting $val(rp) \**

**-llType $val(ll) \**

**-macType $val(mac) \**

**-ifqType $val(ifq) \**

**-ifqLen $val(ifqlen) \**

**-antType $val(ant) \**

**-propType $val(prop) \**

**-phyType $val(netif) \**

**-channel $chan \**

**-topoInstance $topo \**

**-agentTrace ON \**

**-routerTrace ON \**

**-macTrace ON \**

**-movementTrace ON**

**#===================================**

**# Nodes Definition #===================================**

**#Create 6 nodes set n0 [$ns node]**

**$n0 set X\_ 270**

**$n0 set Y\_ 408**

**$n0 set Z\_ 0.0**

**$ns initial\_node\_pos $n0 20 set n1 [$ns node]**

**$n1 set X\_ 503**

**$n1 set Y\_ 404**

**$n1 set Z\_ 0.0**

**$ns initial\_node\_pos $n1 20 set n2 [$ns node]**

**$n2 set X\_ 710**

**$n2 set Y\_ 416**

**$n2 set Z\_ 0.0**

**$ns initial\_node\_pos $n2 20 set n3 [$ns node]**

**$n3 set X\_ 621**

**$n3 set Y\_ 191**

**$n3 set Z\_ 0.0**

**$ns initial\_node\_pos $n3 20 set n4 [$ns node]**

**$n4 set X\_ 402**

**$n4 set Y\_ 176**

**$n4 set Z\_ 0.0**

**$ns initial\_node\_pos $n4 20 set n5 [$ns node]**

**$n5 set X\_ 249**

**$n5 set Y\_ 174**

**$n5 set Z\_ 0.0**

**$ns initial\_node\_pos $n5 20**

**#===================================**

**# Configure mobile nodes #===================================**

**$ns at 1.5 "$n1 setdest 390.0 460.0 40.0"**

**$ns at 1.5 "$n4 setdest 472.0 510.0 50.0"**

**$ns at 1.5 "$n5 setdest 523.0 570.0 40.0" #===================================**

**# Agents Definition #===================================**

**#Setup a TCP connection set tcp0 [new Agent/TCP]**

**$ns attach-agent $n0 $tcp0**

**set sink1 [new Agent/TCPSink]**

**$ns attach-agent $n5 $sink1**

**$ns connect $tcp0 $sink1**

**$tcp0 set packetSize\_ 1500**

**#Setup a UDP connection set udp2 [new Agent/UDP]**

**$ns attach-agent $n2 $udp2 set null3 [new Agent/Null]**

**$ns attach-agent $n3 $null3**

**$ns connect $udp2 $null3**

**$udp2 set packetSize\_ 1500**

**#===================================**

**# Applications Definition #===================================**

**#Setup a FTP Application over TCP connection set ftp0 [new Application/FTP]**

**$ftp0 attach-agent $tcp0**

**$ns at 1.0 "$ftp0 start"**

**$ns at 2.0 "$ftp0 stop”**

**#Setup a CBR Application over UDP connection set cbr1 [new Application/Traffic/CBR]**

**$cbr1 attach-agent $udp2**

**$cbr1 set packetSize\_ 1000**

**$cbr1 set rate\_ 1.0Mb**

**$cbr1 set random\_ null**

**$ns at 1.0 "$cbr1 start"**

**$ns at 2.0 "$cbr1 stop"**

**#===================================**

**# Termination #===================================**

**#Define a 'finish' procedure proc finish {} {**

**global ns tracefile namfile**

**$ns flush-trace close $tracefile close $namfile**

**exec nam out.nam & exit 0**

**}**

**for {set i 0} {$i < $val(nn) } { incr i } {**

**$ns at $val(stop) "\$n$i reset"**

**}**

**$ns at $val(stop) "$ns nam-end-wireless $val(stop)"**

**$ns at $val(stop) "finish"**

**$ns at $val(stop) "puts \"done\" ; $ns halt"**

**$ns run**

**AWK Script to determine the performance with respect to transmission of packets:**

BEGIN{

count=0 pack=0 time=0

}

{

if($1=="r"&&$3=="\_3\_"&&$4=="AGT")

{

}

} END{

count++ pack=pack+$8 time=$2

printf("The Throughtput from n0 to n3: %f Mbps", ((count\*pack\*8)/(time\*1000000)));

}

**OUTPUT:**

$awk –f exp4.awk out.tr

The Throughtput from n0 to n3: 12.52 Mbps

**OR**

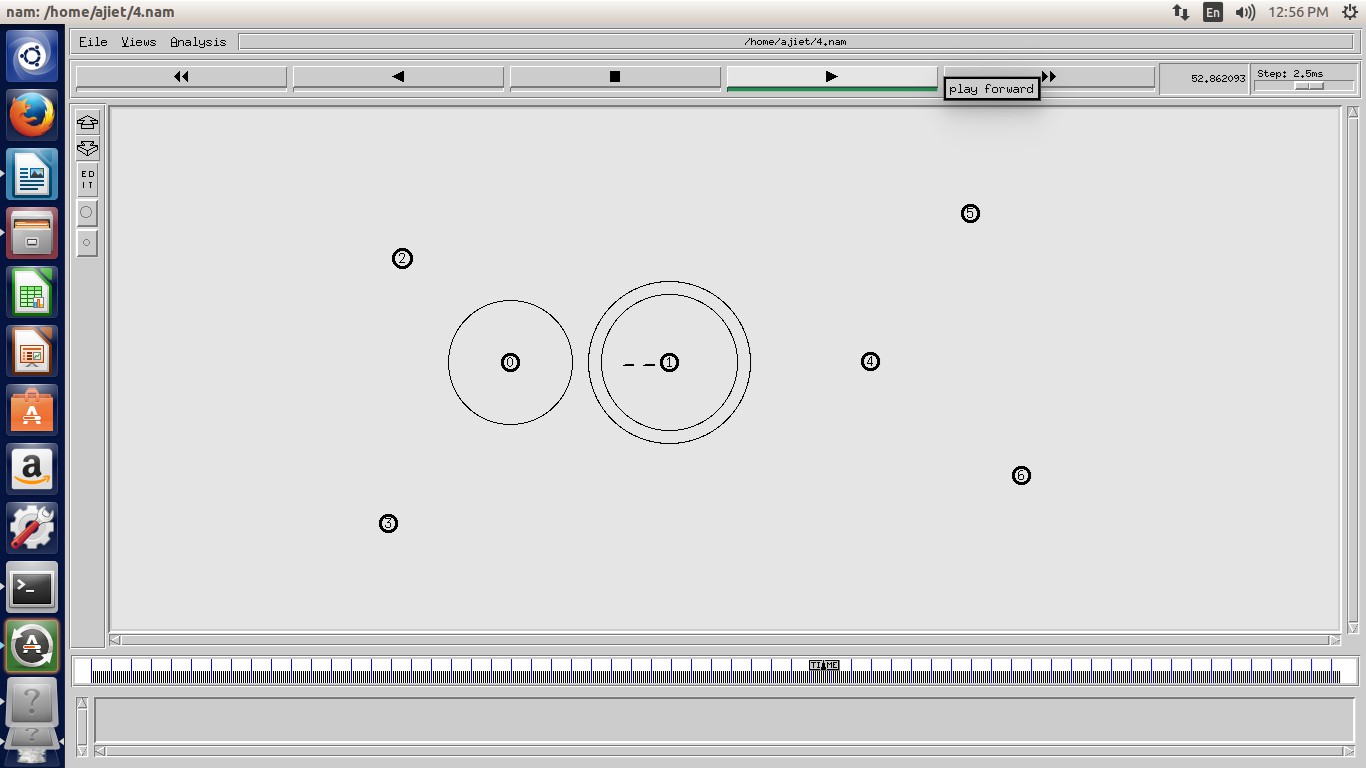
To find the number of packets sent:

**Grep “^s” out.tr | grep –c “\_5\_ AGT”**

To find the number of packets sent:

**Grep “^r” out.tr | grep –c “\_3\_ AGT”**

**RESULT**



**Figure 11: NAM output animation**



**Figure 11: Graph showing performance with respect to transmission of packets**

**Experiment 5: Write a program to find the shortest path between vertices using bellman- ford algorithm.**

Bellman-Ford algorithm solves the single-source shortest-path problem in the general case in which edges of a given digraph can have negative weight as long as G contains no negative cycles.

This algorithm, like Dijkstra's algorithm uses the notion of edge relaxation but does not use with greedy method. Again, it uses d[u] as an upper bound on the distance d[u, v] from u to v. The algorithm progressively decreases an estimate d[v] on the weight of the shortest path from the source vertex s to each vertex v in V until it achieve the actual shortest-path. The algorithm returns Boolean TRUE if the given digraph contains no negative cycles that are reachable from source vertex s otherwise it returns Boolean FALSE.

BELLMAN-FORD (G, w, s)

1. INITIALIZE-SINGLE-SOURCE (G, s)
2. for each vertex i = 1 to V[G] - 1 do
3. for each edge (u, v) in E[G] do
4. RELAX (u, v, w)
5. For each edge (u, v) in E[G] do
6. if d[u] + w(u, v) < d[v] then
7. return FALSE
8. return TRUE
9. ​

Asymptotic complexity:

* Average case (random data): O(|V ||E|)
* Worst case: O(|V ||E|)

Conclusion:

Thus, the Bellman-Ford algorithm runs in O(E) time.

**Program**

import java.util.\*; class DVT

{

public static void main(String args[])

{

int dist[][]=new int[20][20];

int from[][]=new int[20][20]; int costmat[][]=new int[10][10]; int i,j,k,nodes;

Scanner s=new Scanner(System.in);

System.out.println("\nEnter the number of nodes :"); nodes=s.nextInt();

System.out.println("\nEnter the cost matrix :\n"); for(i=1;i<=nodes;i++)

{

for( j=1;j<=nodes;j++)

{

costmat[i][j]=s.nextInt(); costmat[i][i]=0; dist[i][j]=costmat[i][j]; from[i][j]=j;

}

}

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

{

for( j=1;j<=nodes;j++)

{

for( k=1;k<=nodes;k++)

{

if((dist[i][j])>dist[i][k]+dist[k][j])

{

dist[i][j]=dist[i][k]+dist[k][j]; from[i][j]=k;

}

}

}

}

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

{

System.out.println("\n\nFrom Router Node :"+i); System.out.println("\nDesti Node\tNext-Hop\tdistance\n"); for( j=1;j<=nodes;j++)

{

System.out.println(j +"\t \t "+from[i][j]+" \t\t"+dist[i][j]);

}

}

System.out.println("\n\n");

}

}

Output:

$javac BellmanFord.java

$java BellmanFord

**Enter the number of nodes : 3**

**Enter the cost matrix :**

**0 1 2**

**1 0 7**

**2 7 0**

**From Router Node :1**

|  |  |  |
| --- | --- | --- |
| **Desti Node** | **Next-Hop** | **distance** |
| **1** | **1** | **0** |
| **2** | **2** | **1** |
| **3** | **3** | **2** |

**From Router Node :2**

|  |  |  |
| --- | --- | --- |
| **Desti Node** | **Next-Hop** | **distance** |
| **1** | **1** | **1** |
| **2** | **2** | **0** |
| **3** | **1** | **3** |

**From Router Node :3**

|  |  |  |
| --- | --- | --- |
| **Desti Node** | **Next-Hop** | **distance** |
| **1** | **1** | **2** |
| **2** | **1** | **3** |
| **3** | **3** | **0** |

**Enter the number of nodes : 4**

**Enter the cost matrix :**

**0 99 6 3**

**99 0 99 2**

**6 99 0 1**

**3 2 1 0**

**From Router Node :1**

|  |  |  |
| --- | --- | --- |
| **Desti Node** | **Next-Hop** | **distance** |
| **1** | **1** | **0** |
| **2** | **4** | **5** |
| **3** | **4** | **4** |
| **4** | **4** | **3** |

**From Router Node :2**

|  |  |  |
| --- | --- | --- |
| **Desti Node** | **Next-Hop** | **distance** |
| **1** | **4** | **5** |
| **2** | **2** | **0** |
| **3** | **4** | **3** |
| **4** | **4** | **2** |

**From Router Node :3**

|  |  |  |
| --- | --- | --- |
| **Desti Node** | **Next-Hop** | **distance** |
| **1** | **4** | **4** |
| **2** | **4** | **3** |
| **3** | **3** | **0** |
| **4** | **4** | **1** |

**From Router Node :4**

|  |  |  |
| --- | --- | --- |
| **Desti Node** | **Next-Hop** | **distance** |
| **1** | **1** | **3** |

|  |  |  |
| --- | --- | --- |
| **2** | **2** | **2** |
| **3** | **3** | **1** |
| **4** | **4** | **0** |