ST JOSEPH ENGINEERING COLLEGE MANGALURU-575028
(Accredited by NAAC with A+ Grade)
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
(UG Programme Accredited by National Board of Accreditation)

COMPUTER NETWORK LAB MANUAL

18ECL76

VII SEM

2022-23

ST JOSEPH ENGINEERING COLLEGE MANGALURU-575028
(Accredited by NAAC with A+ Grade)
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
(UG Programme Accredited by National Board of Accreditation)

COMPUTER NETWORK LAB MANUAL
18ECL76
VII SEM
2022-23

Lab In-charge:	Ms Padmini Bhat
Lab Faculty:	Ms Nandini Maninarayana
	Ms Reshma K J
	Ms Padmini Bhat
Lab Instructor:	Mr Avil Aaron Pinto

Ms Padmini Bhat
Lab In-Charge

Dr Dayakshini
HOD – ECE

CONTENTS

Vision

"To Excel in Electronics and Communication Engineering Education and Research, focusing on the needs of Industry and Society, with professional ethics"

Mission

- Provide opportunities to deserving students for quality professional education in the field of Electronics and Communication.
- Design and deliver curricula to meet the changing needs of industry through student centric learning methodologies to excel in their profession.
- Recruit, Nurture and Retain best faculty and technical manpower.
- Consolidate the state-of-art infrastructure and equipment for teaching and research activities.
- Promote all round personality development of the students through interaction with alumni, academia and industry.
- Strengthen the Educational Social Responsibilities of the institution.

PROGRAMME OUTCOMES

Engineering Graduate shall be able to

1. Apply knowledge of mathematics, science and engineering fundamentals, and Electronics and Communication Engineering for the solution of engineering problems.
2. Identify, formulate and solve engineering problems.
3. Design electronic systems, components or processes to meet desired specifications within realistic constraints of economic and environmental standards.
4. Design and conduct experiments, as well as to analyze and interpret data pertaining to electronic systems.
5. Use computer aided software tools and techniques for solving electronics and communication engineering problems.
6. Demonstrate awareness of contemporary engineering problems.
7. Apply engineering solutions in societal and environmental context.
8. Understand professional and ethical responsibility.
9. Function within multidisciplinary teams.
10. Communicate effectively in terms of system specifications within the team.
11. Demonstrate the understanding of management principles as applied to the specified work and apply this knowledge to manage the projects as a member and leader in a team.
12. Continue the education in self-learning mode.

PROGRAMME SPECIFIC OUTCOMES

Engineering Graduate shall be able to

1. Participate and succeed in competitive examinations.
2. Understand technological advances in industry through Industry interaction

Course Outcomes

CO No.	Course Outcomes (COs)	Bloom's Taxonomy Level	Target Attainment Level
18ECL76.1	Demonstrate and implement the data link layer framing and error detection methods	L4	2
18ECL76.2	Apply programming concepts to implement various Data link layer protocols	L4	2
18ECL76.3	Implement and analyse the different routing protocols	L4	2
18ECL76.4	Demonstrate the working of Link State routing and leaky bucket algorithms	L4	2
18ECL76.5	Analyse various network configuration using Network Simulator Tool	L4	2
18ECL76.6	Implement the concept of network communication to design an open-ended experiment using C++ programming or NS3	L4	2

Topic Learning Outcomes

Topic No.	Topic Learning Outcomes (TLOs)	COs mapped	Bloom's Taxonomy Level
4.1	Implement a point-to-point network with four nodes and duplex links between them. Analyse the network performance by setting the queue size and varying the bandwidth.	18ECL76.5	L4
4.2	Implement a four-node point to point network with links n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP between n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP.	18ECL76.5	L4
4.3	Implement Ethernet LAN using n (6-10) nodes. Compare the throughput by changing the error rate and data rate	18ECL76.5	L4
4.4	Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations.	18ECL76.5	L4
5.1	Implement ESS with transmission nodes in Wireless LAN and obtain the performance parameters.	18ECL76.5	L4
5.2	Implementation of Link state routing algorithm.	18ECL76.4	L4
1.1	Write a program for a HLDC frame to perform the following. i) Bit stuffing ii) Character stuffing.	18ECL76.1	L4
3.1	Write a program for a distance vector algorithm to find suitable path for transmission.	18ECL68.3	L4
3.2	Implement Dijkstra's algorithm to compute the shortest routing path.	18ECL68.3	L4
1.2	For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the	18ECL76.1	L4

	program for the cases a. Without error b. With error		
2.1	Implementation of Stop and Wait Protocol and Sliding Window Protocol	18ECL76.2	L4
2.2	Write a program for congestion control using leaky bucket algorithm.	18ECL76.4	L4

Syllabus

PART-A: Simulation experiments using NS3

1. Implement a point-to-point network with four nodes and duplex links between them. Analyse the network performance by setting the queue size and varying the bandwidth.
2. Implement a four-node point to point network with links n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP between n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP.
3. Implement Ethernet LAN using n (6-10) nodes. Compare the throughput by changing the error rate and data rate.
4. Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations.
5. Implement ESS with transmission nodes in Wireless LAN and obtain the performanceparameters.
6. Implementation of Link state routing algorithm.

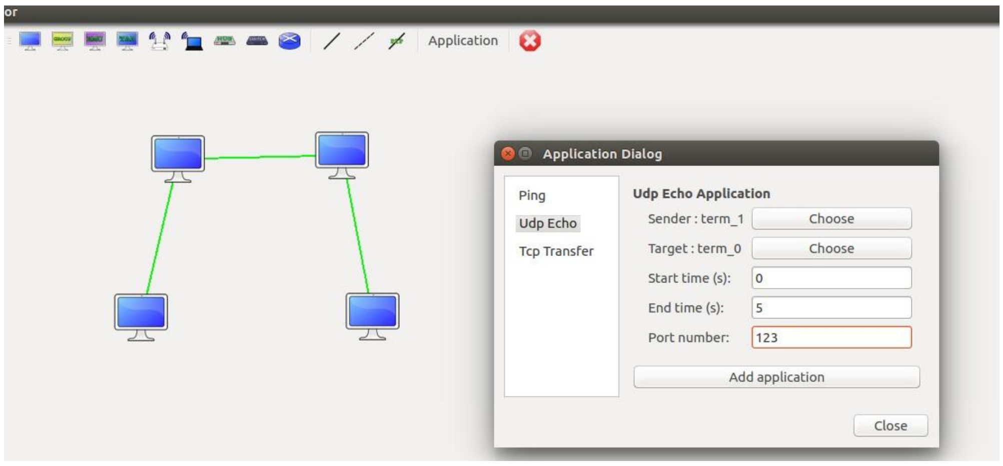
PART-B: Implement the following in C/C++

7. Write a program for a HLDC frame to perform the following.
   i) Bit stuffing   ii) Character stuffing.
8. Write a program for distance vector algorithm to find suitable path for transmission.
9. Implement Dijkstra’s algorithm to compute the shortest routing path.
10. For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the programfor the cases a. Without errorb. With error
11. Implementation of Stop and Wait Protocol and Sliding Window Protocol
12. Write a program for congestion control using leaky bucket algorithm.

Part A using NS3

Program 1: Implement a point–to–point network with 4 nodes and duplex links between them. Analyse the network performance by setting the queue size and varying the bandwidth.

Solution:
Draw the following topology using the steps given below



Udp Echo Application
Sender : term_1
Target : term_0
Start time (s): 0
End time (s): 5
Port number: 123

General Instructions

- Go to the home folder
- Create a folder with your USN number
- For each Program create a new folder inside the USN folder then right click and select open in terminal
- Type: sjecccn.sh
- Topology generator opens

Procedure

Step 1: Get 4 terminals (term_0, term_1, term_2, term_3) form the tool-bar onto the working window of the topology generator.

Step 2: Select the P2P link from the tool-bar and make the connect from term_0 to term_1, term_1 to term_2 and term_2 to term_3.

Step 3: Click on Application dialog and select Udp Echo, choose term_0 as the sender and term_1 as the receiver. Enter Start time, End time, Port number and then click on add application and close

Step 4: Similarly create for the other connections.

Step 5: From menu select Generate and then C++.

Step 6: Save the generated C++ code with .cc extension.

Step 7: Close the topology generator

Step 8: Blank text editor opens.

Step 9: Open the .cc file which was saved in the Step 6. Edit .cc, save and close the text editor

Program

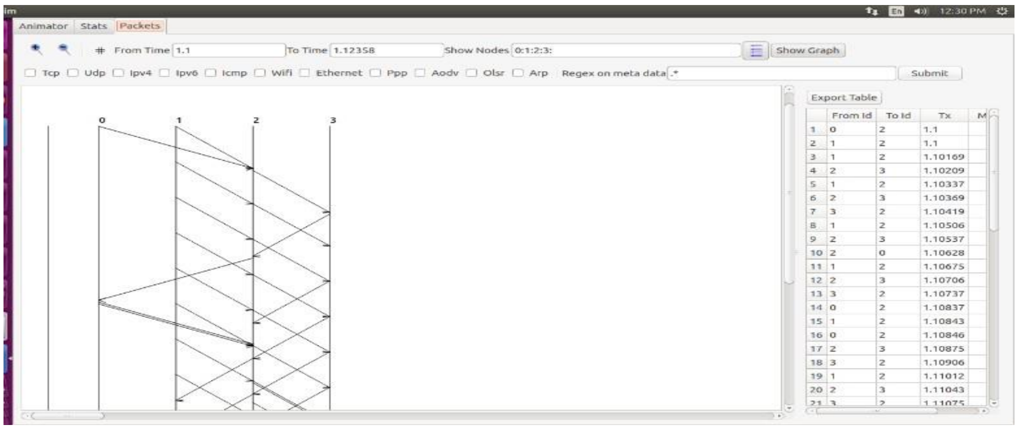
```
#include "ns3/core-module.h"
#include "ns3/global-route-manager.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/bridge-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("Firstscript");

int main (int argc, char *argv[])
{
  CommandLine cmd;
  cmd.Parse (argc, argv);

  /* Configuration. */
LogComponentEnable  ("UdpEchoClientApplication",LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication",LOG_LEVEL_INFO);
std::string animFile="exp1.xml";

  /* Build nodes. */
  NodeContainer term_0;
  term_0.Create (1);
```

```
NodeContainer term_1;
term_1.Create (1);
NodeContainer term_2;
term_2.Create (1);
NodeContainer term_3;
term_3.Create (1);

/*        Build        link.        */
PointToPointHelper p2p_p2p_0;
p2p_p2p_0.SetDeviceAttribute ("DataRate", DataRateValue (5000000));
p2p_p2p_0.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
PointToPointHelper p2p_p2p_1;
p2p_p2p_1.SetDeviceAttribute ("DataRate", DataRateValue (5000000));
p2p_p2p_1.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
PointToPointHelper p2p_p2p_2;
p2p_p2p_2.SetDeviceAttribute ("DataRate", DataRateValue (5000000));
p2p_p2p_2.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));

/* Build link net device container. */
NodeContainer all_p2p_0;
all_p2p_0.Add (term_1);
all_p2p_0.Add (term_0);
NetDeviceContainer ndc_p2p_0 = p2p_p2p_0.Install (all_p2p_0);
NodeContainer all_p2p_1;
all_p2p_1.Add (term_0);
all_p2p_1.Add (term_2);
NetDeviceContainer ndc_p2p_1 = p2p_p2p_1.Install (all_p2p_1);
NodeContainer all_p2p_2;
all_p2p_2.Add (term_2);
all_p2p_2.Add (term_3);
NetDeviceContainer ndc_p2p_2 = p2p_p2p_2.Install (all_p2p_2);

/* Install the IP stack. */
InternetStackHelper internetStackH;
internetStackH.Install (term_0);
internetStackH.Install (term_1);
internetStackH.Install (term_2);
internetStackH.Install (term_3);
```

```
/* IP assign. */
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.0.0.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_0 = ipv4.Assign (ndc_p2p_0);
ipv4.SetBase ("10.0.1.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_1 = ipv4.Assign (ndc_p2p_1);
ipv4.SetBase ("10.0.2.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_2 = ipv4.Assign (ndc_p2p_2);

/* Generate Route. */
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

/* Generate Application. */
uint16_t port_udpEcho_0 = 123;
UdpEchoServerHelper server_udpEcho_0 (port_udpEcho_0);
ApplicationContainer apps_udpEcho_0 = server_udpEcho_0.Install (term_0.Get(0));
apps_udpEcho_0.Start (Seconds (0.0));
apps_udpEcho_0.Stop (Seconds (5.0));
Time interPacketInterval_udpEcho_0 = Seconds (1.0);
UdpEchoClientHelper client_udpEcho_0 (iface_ndc_p2p_0.GetAddress(1), 123);
client_udpEcho_0.SetAttribute ("MaxPackets", UintegerValue (20));
client_udpEcho_0.SetAttribute ("Interval", TimeValue (interPacketInterval_udpEcho_0));
client_udpEcho_0.SetAttribute ("PacketSize", UintegerValue (1024));
apps_udpEcho_0 = client_udpEcho_0.Install (term_1.Get (0));
apps_udpEcho_0.Start (Seconds (0.1));
apps_udpEcho_0.Stop (Seconds (5.0));
uint16_t port_udpEcho_1 = 321;
UdpEchoServerHelper server_udpEcho_1 (port_udpEcho_1);
ApplicationContainer apps_udpEcho_1 = server_udpEcho_1.Install (term_2.Get(0));
apps_udpEcho_1.Start (Seconds (1.0));
apps_udpEcho_1.Stop (Seconds (10.0));
Time interPacketInterval_udpEcho_1 = Seconds (1.0);
UdpEchoClientHelper client_udpEcho_1 (iface_ndc_p2p_1.GetAddress(1), 321);
client_udpEcho_1.SetAttribute ("MaxPackets", UintegerValue (20));
client_udpEcho_1.SetAttribute ("Interval", TimeValue (interPacketInterval_udpEcho_1));
client_udpEcho_1.SetAttribute ("PacketSize", UintegerValue (1024));
apps_udpEcho_1 = client_udpEcho_1.Install (term_0.Get (0));
apps_udpEcho_1.Start (Seconds (1.1));
apps_udpEcho_1.Stop (Seconds (10.0));
```

```
 uint16_t port_udpEcho_2 = 456;
 UdpEchoServerHelper server_udpEcho_2 (port_udpEcho_2);
 ApplicationContainer apps_udpEcho_2 = server_udpEcho_2.Install (term_3.Get(0));
 apps_udpEcho_2.Start (Seconds (1.0));
 apps_udpEcho_2.Stop (Seconds (10.0));
 Time interPacketInterval_udpEcho_2 = Seconds (1.0);
 UdpEchoClientHelper client_udpEcho_2 (iface_ndc_p2p_2.GetAddress(1), 456);
 client_udpEcho_2.SetAttribute ("MaxPackets", UintegerValue (20));
 client_udpEcho_2.SetAttribute ("Interval", TimeValue (interPacketInterval_udpEcho_2));
 client_udpEcho_2.SetAttribute ("PacketSize", UintegerValue (1024));
 apps_udpEcho_2 = client_udpEcho_2.Install (term_2.Get (0));
 apps_udpEcho_2.Start (Seconds (1.1));
 apps_udpEcho_2.Stop (Seconds (10.0));

 /* Simulation. */
AnimationInterface anim(animFile);
Ptr<Node> n=term_0.Get(0);
anim.SetConstantPosition(n,200,200);
n=term_1.Get(0);
anim.SetConstantPosition(n,300,200);
n=term_2.Get(0);
anim.SetConstantPosition(n,250,100);
n=term_3.Get(0);
anim.SetConstantPosition(n,350,100);

 /* Pcap output. */
 /* Stop the simulation after x seconds. */
uint32_t stopTime = 11;
 Simulator::Stop (Seconds (stopTime));
 /* Start and clean simulation. */
 Simulator::Run ();
 Simulator::Destroy ();
}
```

Step 10: Netanim window opens, browse the exp1.xml file from the folder.

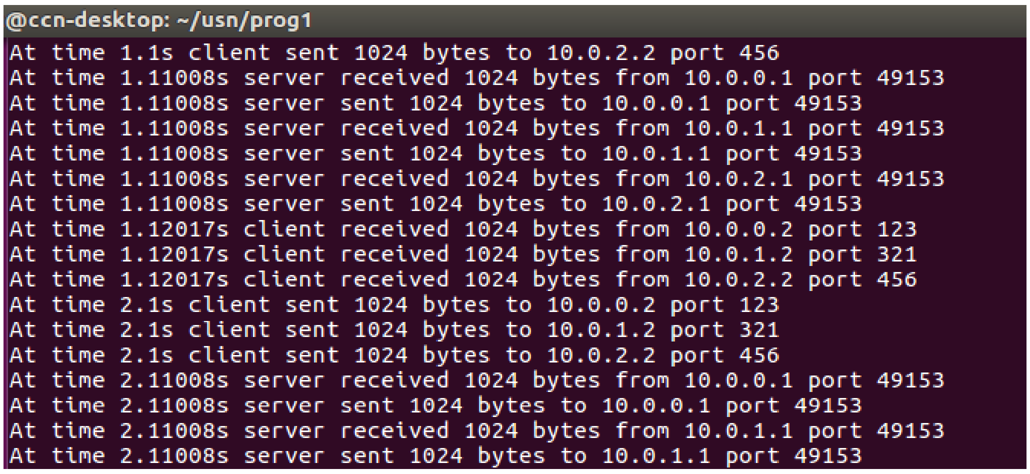


From Time 1.1 To Time 1.12358 Show Nodes 0:1:2:3:

☐ Tcp ☐ Udp ☐ Ipv4 ☐ Ipv6 ☐ Icmp ☐ Wifi ☐ Ethernet ☐ Ppp ☐ Aodv ☐ Olsr ☐ Arp Regex on meta data .*

	From Id	To Id	Tx	M
1	0	2	1.1	
2	1	2	1.1	
3	1	2	1.10169	
4	2	3	1.10209	
5	1	2	1.10337	
6	2	3	1.10369	
7	3	2	1.10419	
8	1	2	1.10506	
9	2	3	1.10537	
10	2	0	1.10628	
11	1	2	1.10675	
12	2	3	1.10706	
13	3	2	1.10737	
14	0	2	1.10837	
15	1	2	1.10843	
16	0	2	1.10846	
17	2	3	1.10875	
18	3	2	1.10906	
19	1	2	1.11012	
20	2	3	1.11043	
21	3	2	1.11075	

Step 11: Analyse the data using Netamin and terminal window

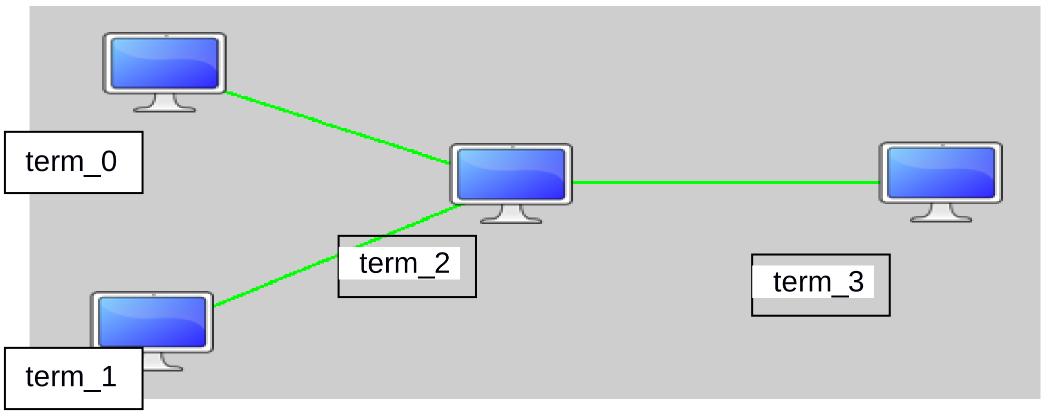


```
@ccn-desktop: ~/usn/prog1
At time 1.1s client sent 1024 bytes to 10.0.2.2 port 456
At time 1.11008s server received 1024 bytes from 10.0.0.1 port 49153
At time 1.11008s server sent 1024 bytes to 10.0.0.1 port 49153
At time 1.11008s server received 1024 bytes from 10.0.1.1 port 49153
At time 1.11008s server sent 1024 bytes to 10.0.1.1 port 49153
At time 1.11008s server received 1024 bytes from 10.0.2.1 port 49153
At time 1.11008s server sent 1024 bytes to 10.0.2.1 port 49153
At time 1.12017s client received 1024 bytes from 10.0.0.2 port 123
At time 1.12017s client received 1024 bytes from 10.0.1.2 port 321
At time 1.12017s client received 1024 bytes from 10.0.2.2 port 456
At time 2.1s client sent 1024 bytes to 10.0.0.2 port 123
At time 2.1s client sent 1024 bytes to 10.0.1.2 port 321
At time 2.1s client sent 1024 bytes to 10.0.2.2 port 456
At time 2.11008s server received 1024 bytes from 10.0.0.1 port 49153
At time 2.11008s server sent 1024 bytes to 10.0.0.1 port 49153
At time 2.11008s server received 1024 bytes from 10.0.1.1 port 49153
At time 2.11008s server sent 1024 bytes to 10.0.1.1 port 49153
```

Program 2: Simulate a four node point-to-point network with the links connected as follows:

n0–n2,n1–n2 and n2–n3. Apply TCP agent between n0-n3 and UDP between n1- n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP /UDP.

Solution:
Step 1: Configure the following network



Step 2: Click on Application dialog and select Udp Echo, choose term_1 as the sender and term_3 as the receiver. Enter Start time, End time, Port number and then click on add application and close

Step 3: Again, click on **Application** dialog and select TcpTransfer by specifying term_0 as the sender and term_3 as the receiver. Enter Start time, End time, Port number and click on Add Application and close.

Step 4: Generate C++ code and save the file in terms of .cc Close the topology generator window.

Step 5: Open the .cc file in the blank text editor. Edit .cc, save and close the text editor

Program

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"
```

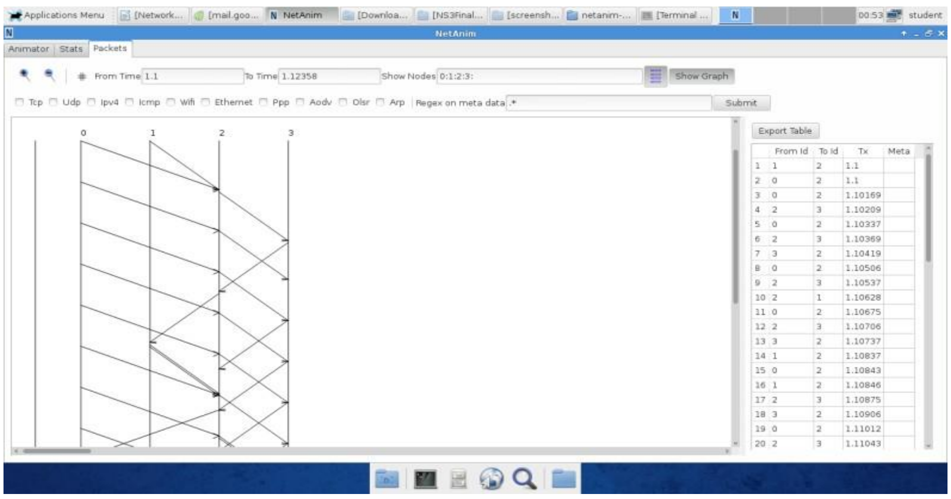
```
using namespace ns3;
NS_LOG_COMPONENT_DEFINE("Firstscript");

int main(int argc, char *argv[])
{
 CommandLine cmd;
 cmd.Parse (argc, argv);

 /* Configuration. */
Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (1024));
Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("5Mbps"));

std::string animFile = "exp2.xml";

 /* Build nodes. */
 NodeContainer term_0;
 term_0.Create (1);
 NodeContainer term_1;
 term_1.Create (1);
 NodeContainer term_2;
 term_2.Create (1);
 NodeContainer term_3;
 term_3.Create (1);

 /*      Build      link.      */
 PointToPointHelper p2p_p2p_0;
 p2p_p2p_0.SetDeviceAttribute ("DataRate", DataRateValue (5000000));
 p2p_p2p_0.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
 PointToPointHelper p2p_p2p_1;
 p2p_p2p_1.SetDeviceAttribute ("DataRate", DataRateValue (5000000));
 p2p_p2p_1.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
 PointToPointHelper p2p_p2p_2;
 p2p_p2p_2.SetDeviceAttribute ("DataRate", DataRateValue (5000000));
 p2p_p2p_2.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));

 /* Build link net device container. */
 NodeContainer all_p2p_0;
 all_p2p_0.Add (term_0);
```

```
all_p2p_0.Add (term_2);
NetDeviceContainer ndc_p2p_0 = p2p_p2p_0.Install (all_p2p_0);
NodeContainer all_p2p_1;
all_p2p_1.Add (term_2);
all_p2p_1.Add (term_3);
NetDeviceContainer ndc_p2p_1 = p2p_p2p_1.Install (all_p2p_1);
NodeContainer all_p2p_2;
all_p2p_2.Add (term_1);
all_p2p_2.Add (term_2);
NetDeviceContainer ndc_p2p_2 = p2p_p2p_2.Install (all_p2p_2);

/* Install the IP stack. */
InternetStackHelper internetStackH;
internetStackH.Install (term_0);
internetStackH.Install (term_1);
internetStackH.Install (term_2);
internetStackH.Install (term_3);

/* IP assign. */
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.0.0.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_0 = ipv4.Assign (ndc_p2p_0);
ipv4.SetBase ("10.0.1.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_1 = ipv4.Assign (ndc_p2p_1);
ipv4.SetBase ("10.0.2.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_2 = ipv4.Assign (ndc_p2p_2);

/* Generate Route. */
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

/* Generate Application. */
uint16_t port_udpEcho_0 = 9;
UdpEchoServerHelper server_udpEcho_0 (port_udpEcho_0);
ApplicationContainer apps_udpEcho_0 = server_udpEcho_0.Install (term_3.Get(0));
apps_udpEcho_0.Start (Seconds (1.0));
apps_udpEcho_0.Stop (Seconds (10.0));
Time interPacketInterval_udpEcho_0 = Seconds (0.001);
UdpEchoClientHelper client_udpEcho_0 (iface_ndc_p2p_1.GetAddress(1), 9);
client_udpEcho_0.SetAttribute ("MaxPackets", UintegerValue (20));
```

```
  client_udpEcho_0.SetAttribute ("Interval", TimeValue (interPacketInterval_udpEcho_0));
  client_udpEcho_0.SetAttribute ("PacketSize", UintegerValue (1024));
  apps_udpEcho_0 = client_udpEcho_0.Install (term_1.Get (0));
  apps_udpEcho_0.Start (Seconds (1.1));
  apps_udpEcho_0.Stop (Seconds (10.0));
  uint16_t port_tcp_0 = 8;
  Address sinkLocalAddress_tcp_0 (InetSocketAddress (Ipv4Address::GetAny (), port_tcp_0));
  PacketSinkHelper sinkHelper_tcp_0 ("ns3::TcpSocketFactory", sinkLocalAddress_tcp_0);
  ApplicationContainer sinkApp_tcp_0 = sinkHelper_tcp_0.Install (term_3);
  sinkApp_tcp_0.Start (Seconds (1.0));
  sinkApp_tcp_0.Stop (Seconds (10.0));
  OnOffHelper clientHelper_tcp_0 ("ns3::TcpSocketFactory", Address ());
clientHelper_tcp_0.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
clientHelper_tcp_0.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
  ApplicationContainer clientApps_tcp_0;
  AddressValue remoteAddress_tcp_0 (InetSocketAddress (iface_ndc_p2p_1.GetAddress (1),
port_tcp_0));
  clientHelper_tcp_0.SetAttribute ("Remote", remoteAddress_tcp_0);
  clientApps_tcp_0.Add (clientHelper_tcp_0.Install (term_0));
  clientApps_tcp_0.Start (Seconds (1.1));
  clientApps_tcp_0.Stop (Seconds (10.0));

  /* Simulation.  */
AnimationInterface anim (animFile);
Ptr<Node> n = term_0.Get (0);
anim.SetConstantPosition (n, 100, 100);
n = term_1.Get (0);
anim.SetConstantPosition (n, 100, 200);
n = term_2.Get (0);
anim.SetConstantPosition (n, 200, 150);
n = term_3.Get (0);
anim.SetConstantPosition (n, 300, 150);

  /* Pcap output. */
  /* Stop the simulation after x seconds. */
uint32_t stopTime = 11;
  Simulator::Stop (Seconds (stopTime));
  /* Start and clean simulation. */
  Simulator::Run ();
  Simulator::Destroy ();

}
```

Step 6: Netanim window opens, browse the exp1.xml file from the folder.

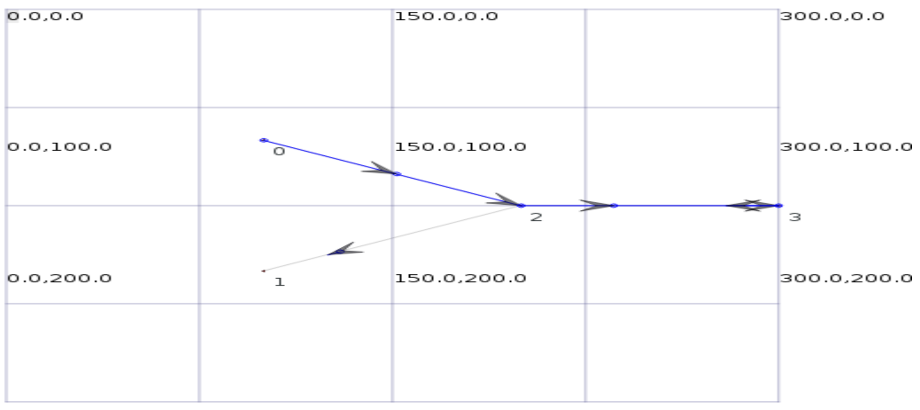
Step 7: Analyse the data using Netamin and terminal window



From Time 1.1 To Time 1.12358 Show Nodes 0:1:2:3:

☐ Tcp ☐ Udp ☐ Ipv4 ☐ Icmp ☐ Wifi ☐ Ethernet ☐ Ppp ☐ Aodv ☐ Olsr ☐ Arp Regex on meta data .*

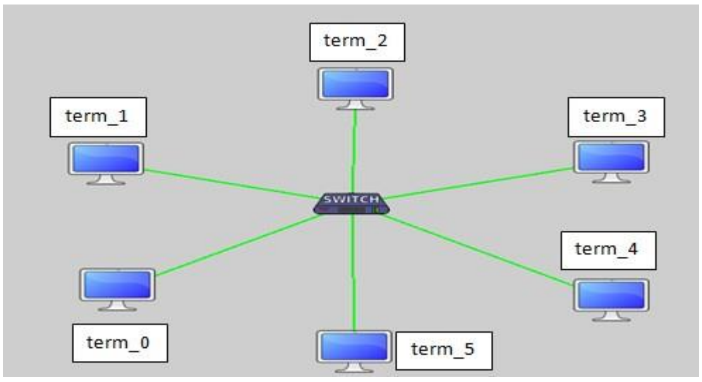
	From Id	To Id	Tx	Meta
1	1	2	1.1	
2	0	2	1.1	
3	0	2	1.10169	
4	2	3	1.10209	
5	0	2	1.10337	
6	2	3	1.10369	
7	3	2	1.10419	
8	0	2	1.10506	
9	2	3	1.10537	
10	2	1	1.10628	
11	0	2	1.10675	
12	2	3	1.10706	
13	3	2	1.10737	
14	1	2	1.10837	
15	0	2	1.10843	
16	1	2	1.10846	
17	2	3	1.10875	
18	3	2	1.10906	
19	0	2	1.11012	
20	2	3	1.11043	



Program 3: Implement an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare throughput.

Solution

Step 1: Configure the following network



Step 2: Again, click on Application dialog and select Udp Echo by specifying term_0 as the sender and term_3 as the receiver. Enter Start time, End time, Port number and click on Add Application and close.

Step 3: Generate C++ code and save the file in terms of .cc Close the topology generator window.

Step 4: Open the .cc file in the blank text editor. Edit .cc, save and close the text editor

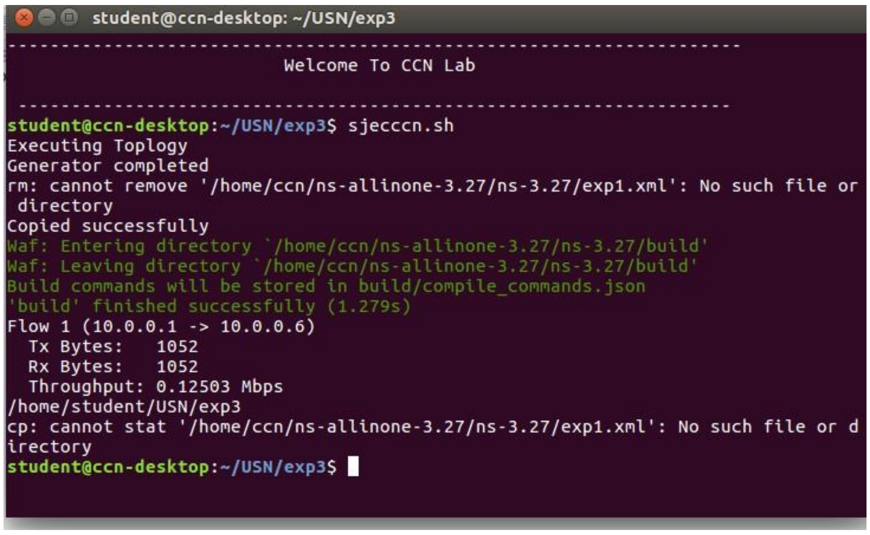
Program

```
#include "ns3/core-module.h"
#include "ns3/global-route-manager.h"
#include "ns3/bridge-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/applications-module.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("EthernetLANExample");

int main(int argc, char *argv[])
```

```
{
 CommandLine cmd;
 cmd.Parse (argc, argv);

 /* Configuration. */
 /* Build nodes. */
 NodeContainer term_0;
 term_0.Create (1);
 NodeContainer term_1;
 term_1.Create (1);
 NodeContainer term_2;
 term_2.Create (1);
 NodeContainer term_3;
 term_3.Create (1);
 NodeContainer term_4;
 term_4.Create (1);
 NodeContainer term_5;
 term_5.Create (1);
 NodeContainer bridge_0;
 bridge_0.Create (1);

 /* Build link. */
 CsmaHelper csma_bridge_0;
 csma_bridge_0.SetChannelAttribute ("DataRate", DataRateValue (100000000));
 csma_bridge_0.SetChannelAttribute ("Delay",  TimeValue (MilliSeconds (10)));

 /* Build link net device container. */
 NodeContainer all_bridge_0;
 all_bridge_0.Add (term_0);
 all_bridge_0.Add (term_1);
 all_bridge_0.Add (term_2);
 all_bridge_0.Add (term_3);
 all_bridge_0.Add (term_4);
 all_bridge_0.Add (term_5);
 NetDeviceContainer terminalDevices_bridge_0;
NetDeviceContainer  BridgeDevices_bridge_0;
for (int i = 0; i < 6; i++)
 {
  NetDeviceContainer link = csma_bridge_0.Install(NodeContainer(all_bridge_0.Get(i),
bridge_0));
  terminalDevices_bridge_0.Add (link.Get(0));
  BridgeDevices_bridge_0.Add (link.Get(1));
 }
 BridgeHelper bridge_bridge_0;
 bridge_bridge_0.Install (bridge_0.Get(0), BridgeDevices_bridge_0);
 NetDeviceContainer ndc_bridge_0 = terminalDevices_bridge_0;

 /* Install the IP stack. */
 InternetStackHelper internetStackH;
 internetStackH.Install (term_0);
 internetStackH.Install (term_1);
 internetStackH.Install (term_2);
 internetStackH.Install (term_3);
 internetStackH.Install (term_4);
 internetStackH.Install (term_5);

 /* IP assign. */
 Ipv4AddressHelper ipv4;
 ipv4.SetBase ("10.0.0.0", "255.255.255.0");
 Ipv4InterfaceContainer iface_ndc_bridge_0 = ipv4.Assign (ndc_bridge_0);
 /* Generate Route. */
 Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
 /* Generate Application. */
 uint16_t port_udpEcho_0 = 8;
 UdpEchoServerHelper server_udpEcho_0 (port_udpEcho_0);
 ApplicationContainer apps_udpEcho_0 = server_udpEcho_0.Install (term_5.Get(0));
 apps_udpEcho_0.Start (Seconds (1.0));
 apps_udpEcho_0.Stop (Seconds (10.0));
 Time interPacketInterval_udpEcho_0 = Seconds (1.0);
 UdpEchoClientHelper client_udpEcho_0 (iface_ndc_bridge_0.GetAddress(5), 8);
 client_udpEcho_0.SetAttribute ("MaxPackets", UintegerValue (10));
 client_udpEcho_0.SetAttribute ("Interval", TimeValue (interPacketInterval_udpEcho_0));
 client_udpEcho_0.SetAttribute ("PacketSize", UintegerValue (1024));
 apps_udpEcho_0 = client_udpEcho_0.Install (term_0.Get (0));
 apps_udpEcho_0.Start (Seconds (1.1));
 apps_udpEcho_0.Stop (Seconds (10.0));
```

```
FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();

  NS_LOG_INFO("RUN SIMULATION");
  Simulator::Stop (Seconds(9.0));
  Simulator::Run ();

monitor->CheckForLostPackets ();

  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>
(flowmon.GetClassifier ());
std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();
for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i !=
stats.end (); ++i)
    {
            Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
if ((t.sourceAddress=="10.0.0.1" && t.destinationAddress == "10.0.0.6"))
      {
          std::cout << "Flow "<< i->first << " (" << t.sourceAddress << " -> " <<
t.destinationAddress << ")\n";
          std::cout <<" Tx Bytes: " << i->second.txBytes << "\n";
          std::cout <<" Rx Bytes: " << i->second.rxBytes << "\n";
          std::cout <<"  Throughput: " << i->second.rxBytes * 8.0 /
(I>second.timeLastRxPacket.GetSeconds() -
i>second.timeFirstTxPacket.GetSeconds())/1024/1024  << " Mbps\n";
       }
     }
   Simulator::Destroy ();
return 0;
}
```

Step 6: Close the Netanim window and analyse the data in the terminal window.

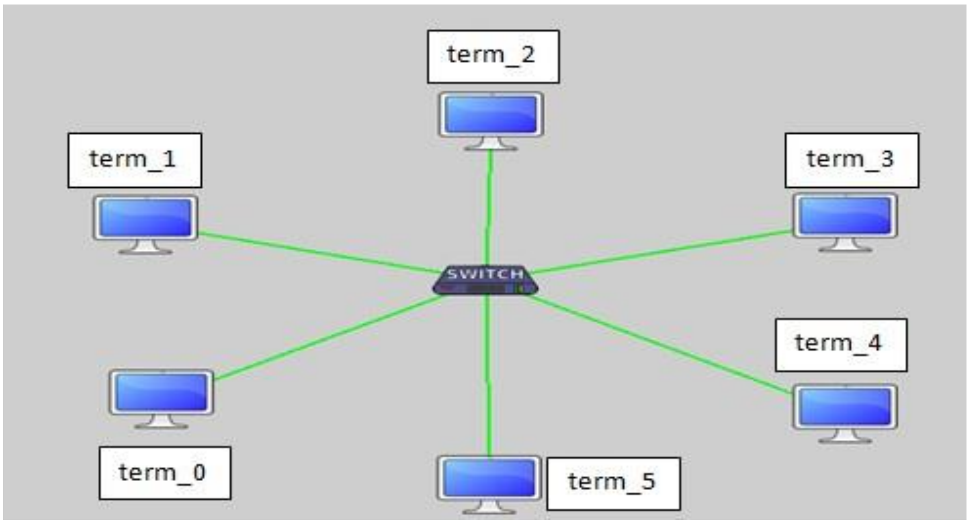


```
student@ccn-desktop: ~/USN/exp3
--------------------------------------------------------------------
                          Welcome To CCN Lab

 ------------------------------------------------------------------
student@ccn-desktop:~/USN/exp3$ sjecccn.sh
Executing Toplogy
Generator completed
rm: cannot remove '/home/ccn/ns-allinone-3.27/ns-3.27/exp1.xml': No such file or
 directory
Copied successfully
Waf: Entering directory `/home/ccn/ns-allinone-3.27/ns-3.27/build'
Waf: Leaving directory `/home/ccn/ns-allinone-3.27/ns-3.27/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.279s)
Flow 1 (10.0.0.1 -> 10.0.0.6)
  Tx Bytes:   1052
  Rx Bytes:   1052
  Throughput: 0.12503 Mbps
/home/student/USN/exp3
cp: cannot stat '/home/ccn/ns-allinone-3.27/ns-3.27/exp1.xml': No such file or d
irectory
student@ccn-desktop:~/USN/exp3$
```

Program 4: Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window for different source / destination.

Solution

Step 1: Configure the following network



Step 2: Set up **Tcp Transfer** by specifying term_1 as the sender and term_5 as the receiver.

Step 3: Generate C++ code and close

Step 4: Edit the .cc , save and close

Step 5: Close the Netanim window and analyse the data in the terminal window

Program

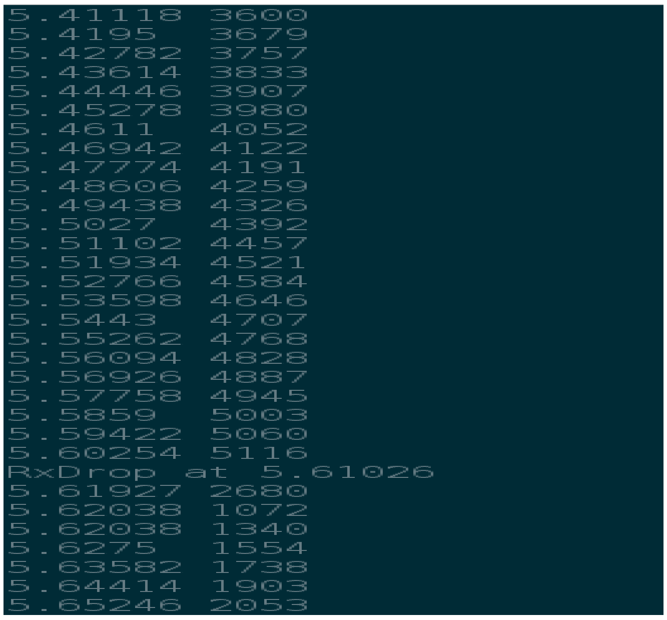
```
#include <fstream>
#include <iostream>
#include <string>
#include <cassert>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/csma-module.h"
#include "ns3/bridge-module.h"

using namespace ns3;
```

```
NS_LOG_COMPONENT_DEFINE ("FifthExample");

classMyApp : public Application
{
public:

MyApp ();
virtual ~MyApp();

void Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t nPackets,
DataRate dataRate);

private:
virtual void StartApplication (void);
virtual void StopApplication (void);

voidScheduleTx (void);
voidSendPacket (void);

Ptr<Socket>m_socket;
Address       m_peer;
uint32_t      m_packetSize;
uint32_t      m_nPackets;
DataRate       m_dataRate;
EventId        m_sendEvent;
bool           m_running;
uint32_t      m_packetsSent;
};

MyApp::MyApp () : m_socket (0),
m_peer (),
m_packetSize (0),
m_nPackets (0),
m_dataRate (0),
m_sendEvent (),
m_running (false),
m_packetsSent (0)
{
}

MyApp::~MyApp()
{
m_socket = 0;
}

Void MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t
nPackets, DataRate dataRate)
{
m_socket =  socket;
m_peer = address;
m_packetSize = packetSize;
m_nPackets = nPackets;
m_dataRate = dataRate;
}

void MyApp::StartApplication (void)
{
m_running = true;
m_packetsSent = 0;
m_socket->Bind ();
m_socket->Connect (m_peer);
SendPacket ();
}

Void MyApp::StopApplication (void)
{
m_running = false;

if (m_sendEvent.IsRunning ())
   {
     Simulator::Cancel (m_sendEvent);
   }

if (m_socket)
   {
m_socket->Close ();
   }
```

```
}

void MyApp::SendPacket (void)
{
Ptr<Packet> packet = Create<Packet> (m_packetSize);
m_socket->Send (packet);

if (++m_packetsSent<m_nPackets)
   {
ScheduleTx ();
   }
}

void MyApp::ScheduleTx (void)
{
if (m_running)
   {
     Time tNext (Seconds (m_packetSize * 8 / static_cast<double> (m_dataRate.GetBitRate
())));
m_sendEvent = Simulator::Schedule (tNext, &MyApp::SendPacket, this);
   }
}

static void CwndChange (uint32_t oldCwnd, uint32_t newCwnd)
{
  NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << "\t" <<newCwnd);
}

static void RxDrop (Ptr<const Packet> p)
{
  NS_LOG_UNCOND ("RxDrop at "<< Simulator::Now ().GetSeconds ());
}

int main(int  argc, char *argv[])
{
CommandLine cmd;
cmd.Parse (argc, argv);

  /* Configuration. */
  /* Build nodes. */
NodeContainer term_0;
term_0.Create (1);
NodeContainer term_1;
term_1.Create (1);
NodeContainer term_2;
term_2.Create (1);
NodeContainer term_3;
term_3.Create (1);
NodeContainer term_4;
term_4.Create (1);
NodeContainer term_5;
term_5.Create (1);
NodeContainer bridge_0;
bridge_0.Create (1);

  /* Build link. */
CsmaHelper csma_bridge_0;
  csma_bridge_0.SetChannelAttribute ("DataRate", DataRateValue (10000000));
  csma_bridge_0.SetChannelAttribute ("Delay",  TimeValue (NanoSeconds (6560)));

  /* Build link net device container. */
NodeContainer all_bridge_0;
all_bridge_0.Add (term_0);
all_bridge_0.Add (term_1);
all_bridge_0.Add (term_2);
all_bridge_0.Add (term_3);
all_bridge_0.Add (term_4);
all_bridge_0.Add (term_5);
NetDeviceContainer terminalDevices_bridge_0;
NetDeviceContainer BridgeDevices_bridge_0;
for (inti = 0; i< 6; i++)
  {
NetDeviceContainer link = csma_bridge_0.Install(NodeContainer(all_bridge_0.Get(i),
bridge_0));
   terminalDevices_bridge_0.Add (link.Get(0));
   BridgeDevices_bridge_0.Add (link.Get(1));
  }
```

```
BridgeHelper bridge_bridge_0;
  bridge_bridge_0.Install (bridge_0.Get(0), BridgeDevices_bridge_0);
NetDeviceContainer ndc_bridge_0 = terminalDevices_bridge_0;

  /* Install the IP stack. */
InternetStackHelper internetStackH;
internetStackH.Install (term_0);
internetStackH.Install (term_1);
internetStackH.Install (term_2);
internetStackH.Install (term_3);
internetStackH.Install (term_4);
internetStackH.Install (term_5);

Ptr<RateErrorModel>em = CreateObject<RateErrorModel> ();
em->SetAttribute ("ErrorRate", DoubleValue (0.00001));
ndc_bridge_0.Get (5)->SetAttribute ("ReceiveErrorModel", PointerValue (em));

  /* IP assign. */
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.0.0.0", "255.255.255.0");
  Ipv4InterfaceContainer iface_ndc_bridge_0 = ipv4.Assign (ndc_bridge_0);

  /* Generate Route. */
  //Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  /* Generate Application. */
  uint16_t port_tcp_0 = 8;
  //Address sinkLocalAddress_tcp_0 (InetSocketAddress (Ipv4Address::GetAny (),
port_tcp_0));
 Address sinkLocalAddress_tcp_0 (InetSocketAddress (iface_ndc_bridge_0.GetAddress (5),
port_tcp_0));

  //PacketSinkHelper sinkHelper_tcp_0 ("ns3::TcpSocketFactory", sinkLocalAddress_tcp_0);
PacketSinkHelper sinkHelper_tcp_0 ("ns3::TcpSocketFactory", InetSocketAddress
(Ipv4Address::GetAny (), port_tcp_0));

ApplicationContainer sinkApp_tcp_0 = sinkHelper_tcp_0.Install (term_5);
  sinkApp_tcp_0.Start (Seconds (0.0));
  sinkApp_tcp_0.Stop (Seconds (20.0));
  /*OnOffHelper clientHelper_tcp_0 ("ns3::TcpSocketFactory", Address ());
clientHelper_tcp_0.SetAttribute ("OnTime",
StringValue("ns3::ConstantRandomVariable[Constant=1]"));
clientHelper_tcp_0.SetAttribute ("OffTime",
StringValue("ns3::ConstantRandomVariable[Constant=0]"));
ApplicationContainer clientApps_tcp_0;
AddressValue remoteAddress_tcp_0 (InetSocketAddress (iface_ndc_bridge_0.GetAddress
(5), port_tcp_0));
  clientHelper_tcp_0.SetAttribute ("Remote", remoteAddress_tcp_0);
  clientApps_tcp_0.Add (clientHelper_tcp_0.Install (term_1));
  clientApps_tcp_0.Start (Seconds (1.1));
  clientApps_tcp_0.Stop (Seconds (20.0));*/

Ptr<Socket> ns3TcpSocket = Socket::CreateSocket (term_1.Get (0),
TcpSocketFactory::GetTypeId ());
  ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow", MakeCallback
(&CwndChange));

Ptr<MyApp> app = CreateObject<MyApp> ();
app->Setup (ns3TcpSocket, sinkLocalAddress_tcp_0, 1040, 1000, DataRate ("1Mbps"));
  term_1.Get (0)->AddApplication (app);
app->SetStartTime (Seconds (1.));
app->SetStopTime (Seconds (20.));

  ndc_bridge_0.Get (5)->TraceConnectWithoutContext ("PhyRxDrop", MakeCallback
(&RxDrop));

  /* Simulation. */
  /* Pcap output. */
  /* Stop the simulation after x seconds. */
  //uint32_t stopTime = 21;
  //Simulator::Stop (Seconds (stopTime));
  /* Start and clean simulation. */
  Simulator::Run ();
  Simulator::Destroy ();
}
```

Output

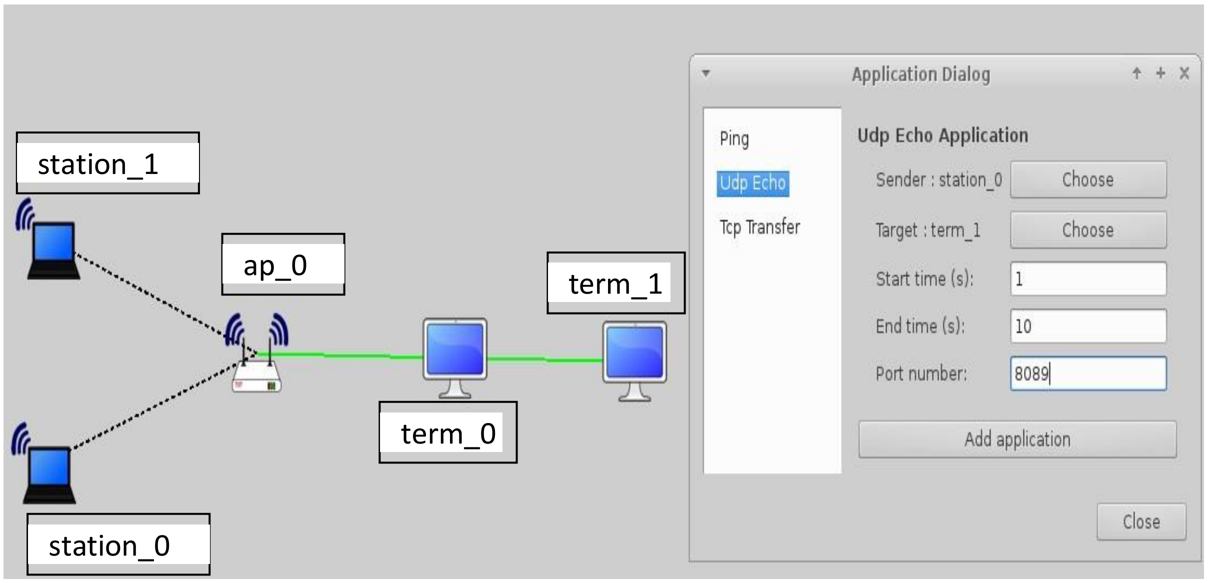


```
5.41118 3600
5.4195  3679
5.42782 3757
5.43614 3833
5.44446 3907
5.45278 3980
5.4611  4052
5.46942 4122
5.47774 4191
5.48606 4259
5.49438 4326
5.5027  4392
5.51102 4457
5.51934 4521
5.52766 4584
5.53598 4646
5.5443  4707
5.55262 4768
5.56094 4828
5.56926 4887
5.57758 4945
5.5859  5003
5.59422 5060
5.60254 5116
RxDrop at 5.61026
5.61927 2680
5.62038 1072
5.62038 1340
5.6275  1554
5.63582 1738
5.64414 1903
5.65246 2053
```

Program 5: Implement ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.

Solution

Step 1: Configure the following network



Udp Echo Application

Sender : station_0

Target : term_1

Start time (s): 1

End time (s): 10

Port number: 8089

Step 2: Set up **Udp Echo** by specifying **station_0** as the sender and **term_1** as the receiver and add application and close.

Step 3: Generate C++ code and close

Step 4: Edit the .cc , save and close

Step 5: In the Netanim window open the exp1.xml and run.

Step 6: Analyse the data

Program

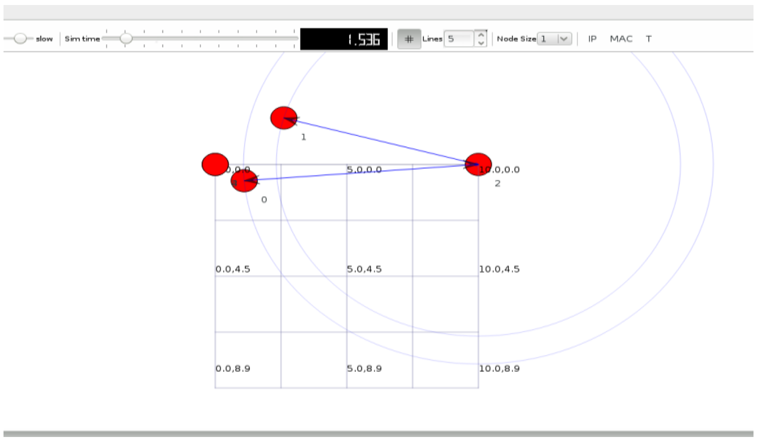
```
#include "ns3/core-module.h"
#include "ns3/global-route-manager.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/bridge-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"
```

```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("WifiScriptExample");

int main(int argc, char *argv[])
{
CommandLine cmd;
cmd.Parse (argc, argv);

  /* Configuration. */
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

std::stringanimFile="wifiudp.xml";

  /* Build nodes. */
NodeContainer station_0;
station_0.Create (1);
NodeContainer station_1;
station_1.Create (1);
NodeContainer ap_0;
ap_0.Create (1);
NodeContainer term_0;
term_0.Create (1);
NodeContainer term_1;
term_1.Create (1);

  /* Build link. */
YansWifiPhyHelper wifiPhy_ap_0 = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel_ap_0 = YansWifiChannelHelper::Default ();
wifiPhy_ap_0.SetChannel (wifiChannel_ap_0.Create ());
PointToPointHelper p2p_p2p_0;
  p2p_p2p_0.SetDeviceAttribute ("DataRate", DataRateValue (100000000));
p2p_p2p_0.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (10)));
PointToPointHelper p2p_p2p_1;
  p2p_p2p_1.SetDeviceAttribute ("DataRate", DataRateValue (100000000));
  p2p_p2p_1.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (10)));

  /* Build link net device container. */
```

```
NodeContainer all_ap_0;
NetDeviceContainer ndc_ap_0;
Ssid ssid_ap_0 = Ssid("ns-3-ssid");
WifiHelper wifi_ap_0 = WifiHelper::Default ();
NqosWifiMacHelper wifiMac_ap_0 = NqosWifiMacHelper::Default ();

  wifi_ap_0.SetRemoteStationManager ("ns3::AarfWifiManager");
 /*wifiMac_ap_0.SetType ("ns3::NqapWifiMac",
   "Ssid", SsidValue (ssid_ap_0),
   "BeaconGeneration", BooleanValue (true),
   "BeaconInterval", TimeValue (Seconds (2.5)));*/

wifiMac_ap_0.SetType ("ns3::StaWifiMac","Ssid",
SsidValue (ssid_ap_0),"ActiveProbing",
BooleanValue (false));

  ndc_ap_0.Add (wifi_ap_0.Install (wifiPhy_ap_0, wifiMac_ap_0, station_0));
  ndc_ap_0.Add (wifi_ap_0.Install (wifiPhy_ap_0, wifiMac_ap_0, station_1));

 /*wifiMac_ap_0.SetType ("ns3::NqstaWifiMac",
   "Ssid", SsidValue  (ssid_ap_0),
   "ActiveProbing", BooleanValue (false));*/

  wifiMac_ap_0.SetType ("ns3::ApWifiMac","Ssid",SsidValue (ssid_ap_0));

ndc_ap_0.Add (wifi_ap_0.Install (wifiPhy_ap_0, wifiMac_ap_0, ap_0 ));

MobilityHelper mobility_ap_0;

 mobility_ap_0.SetPositionAllocator ("ns3::GridPositionAllocator",
                          "MinX", DoubleValue (0.0),
                          "MinY", DoubleValue (0.0),
                          "DeltaX", DoubleValue (5.0),
                          "DeltaY", DoubleValue (10.0),
                          "GridWidth", UintegerValue (3),
                          "LayoutType", StringValue ("RowFirst"));

  mobility_ap_0.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                        "Bounds",
RectangleValue (Rectangle (-50, 50, -50, 50)));
 mobility_ap_0.Install (station_0);
 mobility_ap_0.Install (station_1);
 mobility_ap_0.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
 mobility_ap_0.Install (ap_0);
 mobility_ap_0.Install(all_ap_0);

NodeContainer all_p2p_0;
 all_p2p_0.Add (term_0);
 all_p2p_0.Add (ap_0);
NetDeviceContainer ndc_p2p_0 = p2p_p2p_0.Install (all_p2p_0);
NodeContainer all_p2p_1;
all_p2p_1.Add (term_0);
all_p2p_1.Add (term_1);
NetDeviceContainer ndc_p2p_1 = p2p_p2p_1.Install (all_p2p_1);

  /* Install the IP stack. */
InternetStackHelperinternetStackH;
internetStackH.Install (station_0);
internetStackH.Install (station_1);
internetStackH.Install (ap_0);
internetStackH.Install (term_0);
internetStackH.Install (term_1);

  /* IP assign. */
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.0.0.0", "255.255.255.0");
  Ipv4InterfaceContainer iface_ndc_ap_0 = ipv4.Assign (ndc_ap_0);
  ipv4.SetBase ("10.0.1.0", "255.255.255.0");
  Ipv4InterfaceContainer iface_ndc_p2p_0 = ipv4.Assign (ndc_p2p_0);
  ipv4.SetBase ("10.0.2.0", "255.255.255.0");
  Ipv4InterfaceContainer iface_ndc_p2p_1 = ipv4.Assign (ndc_p2p_1);

  /* Generate Route. */
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  /* Generate Application. */
  uint16_t port_udpEcho_0 = 8;
UdpEchoServerHelper server_udpEcho_0 (port_udpEcho_0);
```

```
ApplicationContainer apps_udpEcho_0 = server_udpEcho_0.Install (term_1.Get(0));
apps_udpEcho_0.Start (Seconds (1.0));
apps_udpEcho_0.Stop (Seconds (10.0));
Time interPacketInterval_udpEcho_0 = Seconds (1.0);
UdpEchoClientHelper client_udpEcho_0 (iface_ndc_p2p_1.GetAddress(1), 8);
client_udpEcho_0.SetAttribute ("MaxPackets", UintegerValue (1));
client_udpEcho_0.SetAttribute ("Interval", TimeValue (interPacketInterval_udpEcho_0));
client_udpEcho_0.SetAttribute ("PacketSize", UintegerValue (1024));
apps_udpEcho_0 = client_udpEcho_0.Install (station_0.Get (0));
apps_udpEcho_0.Start (Seconds (1.1));
apps_udpEcho_0.Stop (Seconds (10.0));

 /* Simulation. */
AnimationInterfaceanim(animFile);
 /* Pcap output. */
 /* Stop the simulation after x seconds. */
uint32_tstopTime = 11;
 Simulator::Stop (Seconds (stopTime));
 /* Start and clean simulation. */
 Simulator::Run ();
 Simulator::Destroy ();
}
```

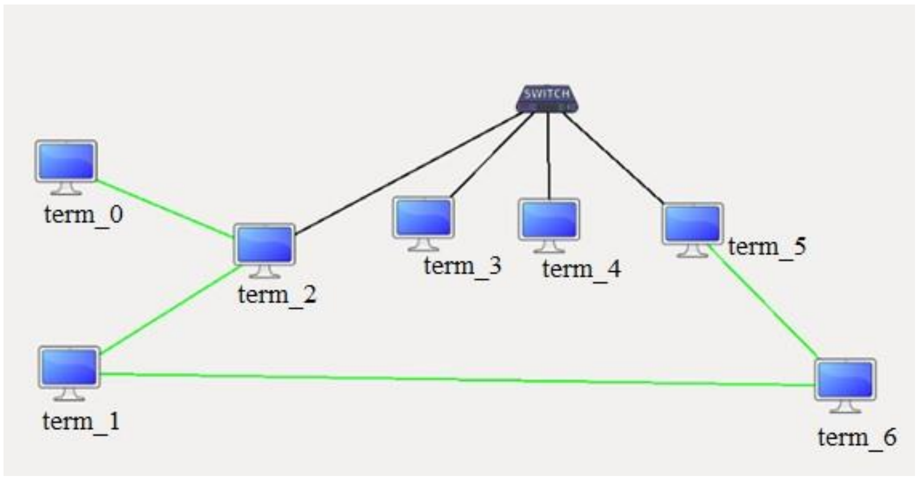
Output



Program 6: Implement Link state routing algorithm

Solution:

Step 1: Configure the following network



Step 3: Generate C++ code and close

Step 4: Edit the .cc , save and close

Step 5: In the Netanim window open the exp1.xml and run.

Step 6: Analyse the data

Program

```
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>
#include "ns3/core-module.h
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("DynamicGlobalRoutingExample");
int main(int argc, char *argv[])
{
```

```
CommandLine cmd;
cmd.Parse (argc, argv);
Config::SetDefault ("ns3::Ipv4GlobalRouting::RespondToInterfaceEvents", BooleanValue
(true));
/* Configuration. */

/* Build nodes. */
NodeContainer term_0;
term_0.Create (1);
NodeContainer term_1;
term_1.Create (1);
NodeContainer term_2;
term_2.Create (1);
NodeContainer term_3;
term_3.Create (1);
NodeContainer term_4;
term_4.Create (1);
NodeContainer term_5;
term_5.Create (1);
NodeContainer term_6;
term_6.Create (1);

/*      Build      link.      */
CsmaHelper csma_hub_0;
csma_hub_0.SetChannelAttribute ("DataRate", DataRateValue (5000000));
csma_hub_0.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
PointToPointHelper p2p_p2p_0;
p2p_p2p_0.SetDeviceAttribute  ("DataRate",  DataRateValue  (5000000));
p2p_p2p_0.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (10)));


/* Build link net device container. */
NodeContainer all_hub_0;
all_hub_0.Add (term_2);
all_hub_0.Add (term_3);
all_hub_0.Add (term_4);
all_hub_0.Add (term_5);
NetDeviceContainer ndc_hub_0 = csma_hub_0.Install (all_hub_0);
NodeContainer all_p2p_0;
```

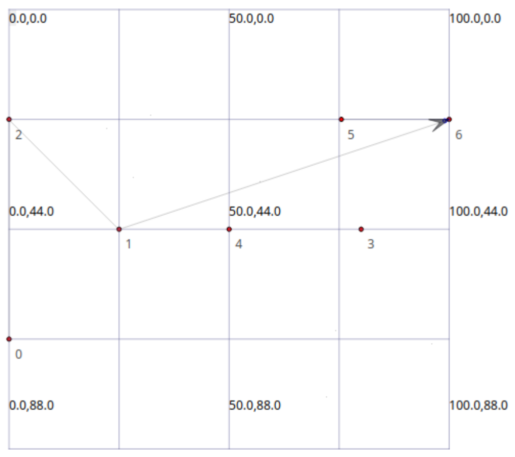
```
all_p2p_0.Add (term_0);
all_p2p_0.Add (term_2);
NetDeviceContainer ndc_p2p_0 = p2p_p2p_0.Install (all_p2p_0);
NodeContainer all_p2p_1;
all_p2p_1.Add (term_1);
all_p2p_1.Add (term_2);
NetDeviceContainer ndc_p2p_1 = p2p_p2p_0.Install (all_p2p_1);
NodeContainer all_p2p_2;
all_p2p_2.Add (term_5);
all_p2p_2.Add (term_6);
NetDeviceContainer ndc_p2p_2 = p2p_p2p_0.Install (all_p2p_2);
NodeContainer all_p2p_3;
all_p2p_3.Add (term_1);
all_p2p_3.Add (term_6);
NetDeviceContainer ndc_p2p_3 = p2p_p2p_0.Install (all_p2p_3);

/* Install the IP stack. */
InternetStackHelper internetStackH;
internetStackH.Install (term_0);
internetStackH.Install (term_1);
internetStackH.Install (term_2);
internetStackH.Install (term_3);
internetStackH.Install (term_4);
internetStackH.Install (term_5);
internetStackH.Install (term_6);

/* IP assign. */
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.250.1.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_hub_0 = ipv4.Assign (ndc_hub_0);
ipv4.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_0 = ipv4.Assign (ndc_p2p_0);
ipv4.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_1 = ipv4.Assign (ndc_p2p_1);
ipv4.SetBase ("10.1.3.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_2 = ipv4.Assign (ndc_p2p_2);
ipv4.SetBase ("172.16.1.0", "255.255.255.0");
Ipv4InterfaceContainer iface_ndc_p2p_3 = ipv4.Assign (ndc_p2p_3);
```

```
  /* Generate Route. */
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  //NS_LOG_INFO ("Create Applications.");
  uint16_t port = 9; // Discard port (RFC 863)
  OnOffHelper onoff ("ns3::UdpSocketFactory",
                     InetSocketAddress (iface_ndc_p2p_3.GetAddress (1), port));
onoff.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onoff.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute ("DataRate", StringValue ("2kbps"));
onoff.SetAttribute ("PacketSize", UintegerValue (50));

  ApplicationContainer apps = onoff.Install (term_1.Get(0));
  apps.Start (Seconds (1.0));
  apps.Stop (Seconds (10.0));

  // Create a second OnOff application to send UDP datagrams of size
  // 210 bytes at a rate of 448 Kb/s
  OnOffHelper onoff2 ("ns3::UdpSocketFactory",
                     InetSocketAddress (iface_ndc_p2p_2.GetAddress (1), port));
onoff2.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onoff2.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
onoff2.SetAttribute ("DataRate", StringValue ("2kbps"));
onoff2.SetAttribute ("PacketSize", UintegerValue (50));

  ApplicationContainer apps2 = onoff2.Install (term_1.Get(0));
  apps2.Start (Seconds (11.0));
  apps2.Stop (Seconds (16.0));

  // Create an optional packet sink to receive these packets
  PacketSinkHelper sink ("ns3::UdpSocketFactory",
   Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
  apps = sink.Install (term_6.Get(0));
  apps.Start (Seconds (1.0));
  apps.Stop (Seconds (10.0));
```

```
PacketSinkHelper sink2 ("ns3::UdpSocketFactory",
Address (InetSocketAddress (Ipv4Address::GetAny (), port)));
apps2 = sink2.Install (term_6.Get(0));
apps2.Start (Seconds (11.0));
apps2.Stop (Seconds (16.0));


AsciiTraceHelper ascii;
Ptr<OutputStreamWrapper> stream = ascii.CreateFileStream ("dynamic-global
routing.tr");
p2p_p2p_0.EnableAsciiAll (stream);
csma_hub_0.EnableAsciiAll (stream);
internetStackH.EnableAsciiIpv4All (stream);

p2p_p2p_0.EnablePcapAll ("dynamic-global-routing");
csma_hub_0.EnablePcapAll ("dynamic-global-routing", false);

Ptr<Node> n1 = term_1.Get(0);
Ptr<Ipv4> ipv41 = n1->GetObject<Ipv4> ();
// The first ifIndex is 0 for loopback, then the first p2p is numbered 1,
// then the next p2p is numbered 2
uint32_t ipv4ifIndex1 = 2;

Simulator::Schedule (Seconds (2),&Ipv4::SetDown,ipv41, ipv4ifIndex1);
Simulator::Schedule (Seconds (4),&Ipv4::SetUp,ipv41, ipv4ifIndex1);

Ptr<Node> n6 = term_6.Get(0);
Ptr<Ipv4> ipv46 = n6->GetObject<Ipv4> ();
// The first ifIndex is 0 for loopback, then the first p2p is numbered 1,
// then the next p2p is numbered 2
uint32_t ipv4ifIndex6 = 2;
Simulator::Schedule (Seconds (6),&Ipv4::SetDown,ipv46, ipv4ifIndex6);
Simulator::Schedule (Seconds (8),&Ipv4::SetUp,ipv46, ipv4ifIndex6);

Simulator::Schedule (Seconds (12),&Ipv4::SetDown,ipv41, ipv4ifIndex1);
Simulator::Schedule (Seconds (14),&Ipv4::SetUp,ipv41, ipv4ifIndex1);
Ipv4GlobalRoutingHelper g;
```

```
Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper> ("dynamic-
global-routing.routes", std::ios::out);
g.PrintRoutingTableAllAt (Seconds (12), routingStream);
AnimationInterface anim("exp1.xml");
Ptr<Node> n=term_0.Get(0);
anim.SetConstantPosition(term_0.Get(0),0.0,75.0);
anim.SetConstantPosition(term_1.Get(0),25.0,50.0);
anim.SetConstantPosition(term_2.Get(0),0.0,25.0);
anim.SetConstantPosition(term_3.Get(0),80.0,50.0);
anim.SetConstantPosition(term_4.Get(0),50.0,50.0);
anim.SetConstantPosition(term_5.Get(0),75.5,25.0);
anim.SetConstantPosition(term_6.Get(0),100.0,25.0);
NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
Simulator::Destroy ();
NS_LOG_INFO ("Done.");
}
```

Output



Part B Programs using C/C++

Program 1

Aim: To write a C++ program to perform Character Stuffing and Bit Stuffing

Software/ OS: g++, Ubuntu (Linux platform)

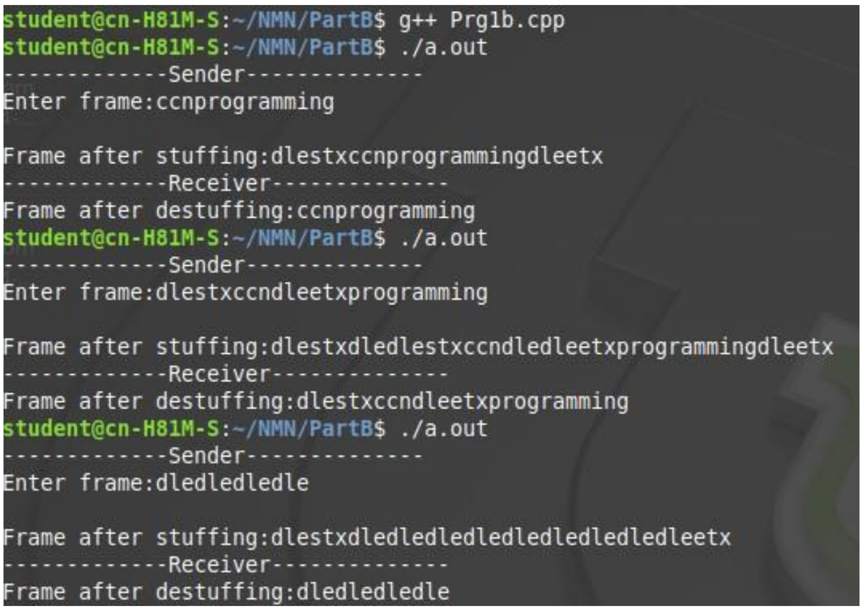
Theory: In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII. The header, which normally carries the source and destination addresses and other control information, and the trailer, which carries error detection or error correction redundant bits, are also multiples of 8 bits. To separate one frame from the next, an 8-bit (l-byte) flag is added at the beginning and the end of a frame. The flag, composed of protocol-dependent special characters, signals the start or end of a frame
The flag composed of protocol dependent special character sequence DLESTX and ends with the sequence DLEETX (where DLE is Data Link Escape, STX is Start of TeXt and ETX is End of TeXt.) When any part of the information is similar to the character used for the flag, the receiver thinks it has reached the end of the frame. To fix this problem, a technique called character stuffing is used. The sender's data link layer inserts an ASCII DLE character just before the DLE character in the data. The receiver's data link layer removes this DLE before this data is given to the network layer.

Program

```
#include<iostream>
using namespace std;
#include<string.h>
int main()
{
int i=0,j=0;
char frame[20],stuffframe[50]="\0";
char dframe[20]="\0",recframe[50]="\0";

cout<<"...............Sender...............\n";
cout<<"Enter frame:";
cin>>frame;
strcpy(stuffframe,"dlestx");
j=strlen("dlestx");
for(i=0;i<strlen(frame);i++)
{
if(frame [i]=='d' && frame [i+1]=='l' && frame [i+2]=='e')
{
```

```
stuffframe [j++]='d';
stuffframe [j++]='l';
stuffframe [j++]='e';
stuffframe [j++]= frame[i++];
stuffframe [j++]= frame [i++];

stuffframe [j++]= frame [i];
}
else
stuffframe [j++]= frame [i];
}
strcat(stuffframe,"dleetx");
cout<<"\nFrame after stuffing:"<<stuffframe;
cout<<"\n------------Receiver------------";
j=0;
strcpy(recframe,stuffframe);
for(i=0;i<strlen(recframe);i++)
{
if(recframe[i]=='d'&&recframe[i+1]=='l'&&recframe[i+2]=='e'&&recframe[i+3]=='e'&&recfra
me[i+4]=='t'&&recframe[i+5]=='x')
i=i+6;
if(recframe[i]=='d'&&recframe[i+1]=='l'&&recframe[i+2]=='e'&&recframe[i+3]=='s'&&recfra
me[i+4]=='t'&&recframe[i+5]=='x')
i=i+6;
if (recframe[i]=='d'&&recframe[i+1]=='l'&&recframe[i+2]=='e')
i=i+3;
dframe[j++]=recframe[i];
}
dframe[j]='\0';
cout<<"\nFrame after destuffing:" <<dframe<<endl;
return 0;
}
```

Output



```
student@cn-H81M-S:~/NMN/PartB$ g++ Prg1b.cpp
student@cn-H81M-S:~/NMN/PartB$ ./a.out
-------------Sender-------------
Enter frame:ccnprogramming

Frame after stuffing:dlestxccnprogrammingdleetx
-------------Receiver-------------
Frame after destuffing:ccnprogramming
student@cn-H81M-S:~/NMN/PartB$ ./a.out
-------------Sender-------------
Enter frame:dlestxccndleetxprogramming

Frame after stuffing:dlestxdledlestxccndledleetxprogrammingdleetx
-------------Receiver-------------
Frame after destuffing:dlestxccndleetxprogramming
student@cn-H81M-S:~/NMN/PartB$ ./a.out
-------------Sender-------------
Enter frame:dledledledle

Frame after stuffing:dlestxdledledledledledledledledleetx
-------------Receiver-------------
Frame after destuffing:dledledledle
```

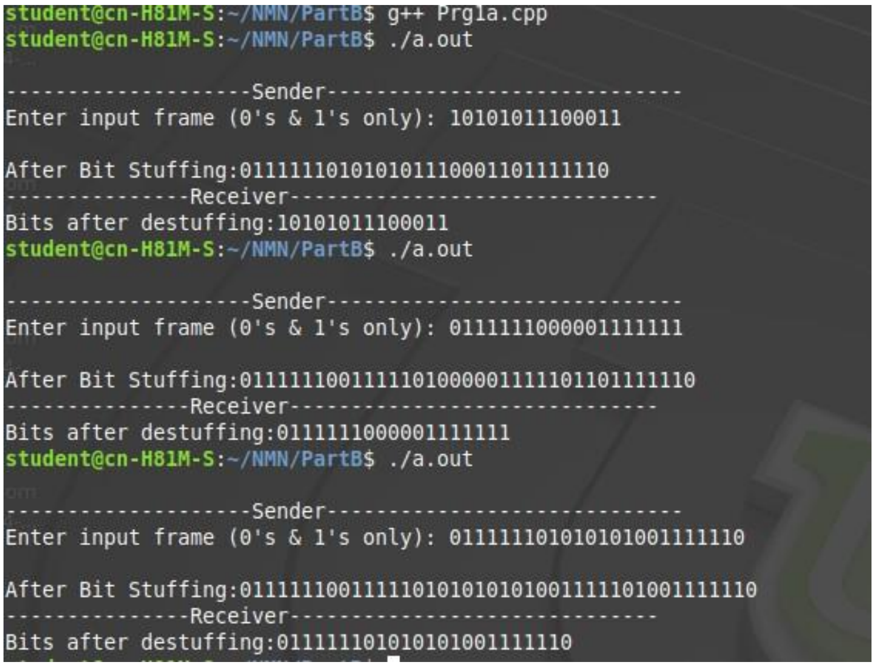
Bit Stuffing: In bit-oriented framing, the data section of a frame is a sequence of bits. If the flag patterns appear in the data, 1 single bit is stuffed to prevent the pattern from looking like a flag. Here each frame begins and ends with a special bit pattern, 01111110. Whenever the data stream contains five consecutive 1s in the data, the data link layer stuffs a 0 bit into the outgoing bit stream. This strategy is known as bit stuffing. Reverse mechanism is known as bit de-stuffing

Program

```
#include<iostream>
#include <string.h>
using namespace std;

int main()
{
int i,j,count=0;
char frame[100],stufframe[100]="\0";
char dframe[100]="\0",recframe[100]="\0";
cout<<"\n.........................Sender....................................";
cout<<"\nEnter input frame (0's & 1's only): ";
cin>>frame;
strcpy(stufframe,"01111110");
j=strlen(stufframe);
for(i=0;frame[i]; i++)
{
```

```
if(frame[i]=='1')
count++;
else
count=0;
stufframe[j++]=frame[i];
if(count==5)
{
stufframe[j++]='0';
count=0;
}
}
strcat(stufframe,"01111110");
cout<<"\nAfter Bit Stuffing:"<<stufframe;
cout<<"\n..............Receiver.............................";
strcpy(recframe,stufframe);
j=0;
for(i=0;i<strlen(recframe);i++)
{
if(recframe[i]=='0'&&recframe[i+1]=='1'&&recframe[i+2]=='1'&&recframe[i+3]=='1'&&recfame[i+4]=='1'&&recframe[i+5]=='1'&&recframe[i+6]== '1' &&recframe[i+7]=='0')
i=i+8;
dframe[j++]=recframe[i];
if(recframe[i]=='1'&&recframe[i+1]=='1'&&recframe[i+2]=='1'&&recframe[i+3]=='1'&&recframe[i+4]=='1')
{
i=i+1;
dframe[j++]=recframe[i++];
dframe[j++]=recframe[i++];
dframe[j++]=recframe[i++];
dframe[j++]=recframe[i++];
}
}
cout<<"\nBits after destuffing:"<<dframe<<endl;
return 0;
}
```

Output



```
student@cn-H81M-S:~/NMN/PartB$ g++ Prg1a.cpp
student@cn-H81M-S:~/NMN/PartB$ ./a.out

--------------------Sender-----------------------------
Enter input frame (0's & 1's only): 10101011100011

After Bit Stuffing:0111111010101011100011011111110
---------------Receiver------------------------------
Bits after destuffing:10101011100011
student@cn-H81M-S:~/NMN/PartB$ ./a.out

--------------------Sender-----------------------------
Enter input frame (0's & 1's only): 0111111000001111111

After Bit Stuffing:01111110011111010000011111011011111110
---------------Receiver------------------------------
Bits after destuffing:0111111000001111111
student@cn-H81M-S:~/NMN/PartB$ ./a.out

--------------------Sender-----------------------------
Enter input frame (0's & 1's only): 0111111010101010011111110

After Bit Stuffing:011111100111110101010101001111101001111110
---------------Receiver------------------------------
Bits after destuffing:0111111010101010011111110
```

Program 2

Aim: To write a C++ program for distance vector algorithm to find suitable path for transmission and analyses the output for different number of nodes and distance matrix

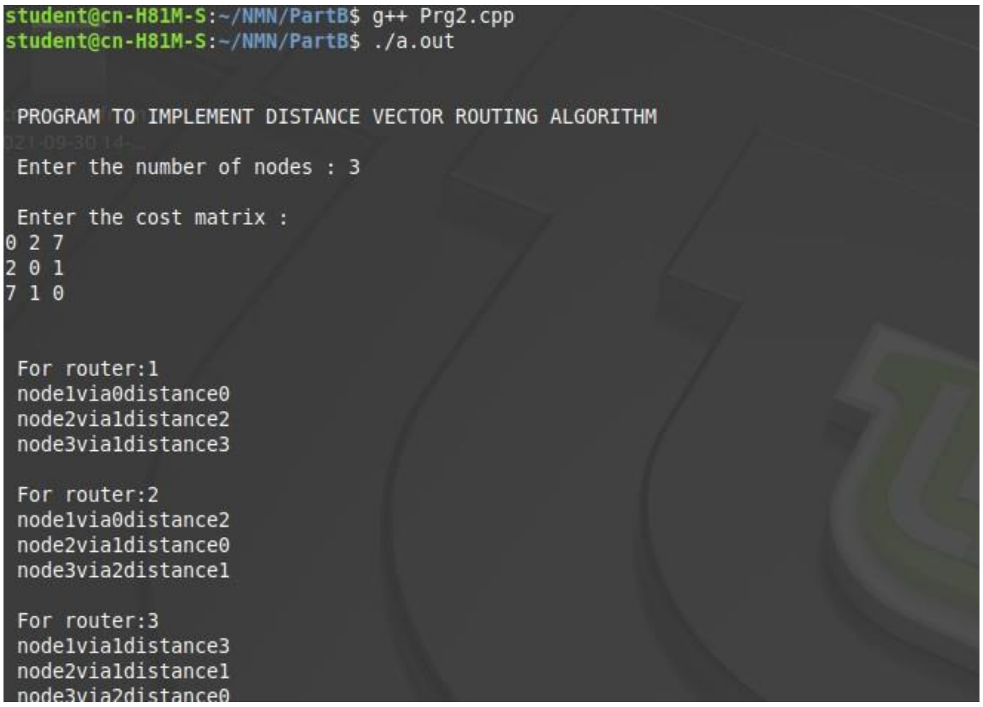
Software/ OS: g++, Ubuntu (Linux platform)

Theory: In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distances to every node. The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing). We can think of nodes as the cities in an area and the lines as the roads connecting them. A table can show a tourist the minimum distance between cities

Program

```
#include<iostream>
using namespace std;
struct node
{
int dist[10];
unsigned from[10];
}DVR[10];
int main()
{
cout<<"\n\n PROGRAM TO IMPLEMENT DISTANCE VECTOR ROUTING ALGORITHM ";
int costmat[10][10];
int nodes, i, j, k;
cout<<"\n\n Enter the number of nodes : ";
cin>>nodes;
cout<<"\n Enter the cost matrix : \n" ;
for(i = 0; i < nodes; i++)
{
for(j = 0; j < nodes; j++)
{
cin>>costmat[i][j];
costmat[i][i] = 0;
DVR[i].dist[j] = costmat[i][j];
DVR[i].from[j] = j;
}
}
for(i = 0; i < nodes; i++)
for(j = i+1; j < nodes; j++)
for(k = 0; k < nodes; k++)
if(DVR[i].dist[j] > costmat[i][k] + DVR[k].dist[j])
{
DVR[i].dist[j] = DVR[i].dist[k] + DVR[k].dist[j];
DVR[j].dist[i] = DVR[i].dist[j];
DVR[i].from[j] = k;
DVR[j].from[i] = k;
}
for(i = 0; i < nodes; i++)
{
cout<<"\n\n For router:"<<i+1;
for(j = 0; j < nodes; j++)
cout<<"\t\n node"<<j+1<<"via"<<DVR[i].from[j]<<"distance"<<DVR[i].dist[j];
}
cout<<endl;
return 0;
}
```

Output



```
student@cn-H81M-S:~/NMN/PartB$ g++ Prg2.cpp
student@cn-H81M-S:~/NMN/PartB$ ./a.out


 PROGRAM TO IMPLEMENT DISTANCE VECTOR ROUTING ALGORITHM

 Enter the number of nodes : 3

 Enter the cost matrix :
0 2 7
2 0 1
7 1 0


 For router:1
 node1via0distance0
 node2via1distance2
 node3via1distance3

 For router:2
 node1via0distance2
 node2via1distance0
 node3via2distance1

 For router:3
 node1via1distance3
 node2via1distance1
 node3via2distance0
```

Program 3

Aim: Write a C++ program to implement Dijkstra's algorithm to compute the shortest path.

Software/ OS: g++, Ubuntu (Linux platform)

Theory: Dijkstra's algorithm, conceived by Dutch computer scientist Edsger Dijkstra in 1956 and published in 1959, is a graph search algorithm that solves the single source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. This algorithm is often used in routing as a subroutine in other graph algorithms, or in GPS Technology.

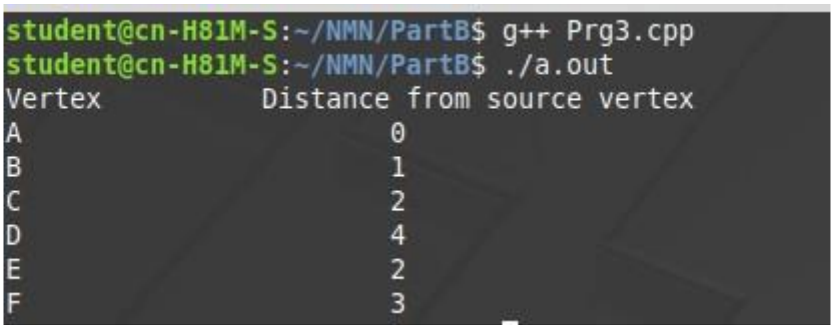
For a given source vertex in the graph, the algorithm finds the path with lowest cost between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined.

Step 1: Select any vertex as starting vertex.
Step 2: Make the distance for all other vertices as INF_MAX
Step 3: From the starting vertex, visit all its adjacent/ neighbouring vertices and write their cost/value.
Step 4: Now that we have finished visiting all the adjacent vertices, we make the source vertex as visited.
Step 5: Choose another vertex, such that, the distance from source vertex is minimum.
Step 6: Now write down the cost for reaching it's adjacent nodes form the starting node, such that the distance should be minimum.
Step 7: Repeat till all the nodes have been visited.

Program

```
#include<iostream>
#include<climits>
using namespace std;

int miniDist(int distance[], bool Tset[]) // finding minimum distance
{
  int minimum=INT_MAX,ind;

  for(int k=0;k<6;k++)
  {
    if(Tset[k]==false && distance[k]<=minimum)
    {
      minimum=distance[k];
      ind=k;
    }
  }
  return ind;
}

void DijkstraAlgo(int graph[6][6],int src) // adjacency matrix
{
  int distance[6]; // // array to calculate the minimum distance for each node
  bool Tset[6];// boolean array to mark visited and unvisited for each node


  for(int k = 0; k<6; k++)
  {
    distance[k] = INT_MAX;
    Tset[k] = false;
  }

  distance[src] = 0;  // Source vertex distance is set 0

  for(int k = 0; k<6; k++)
  {
    int m=miniDist(distance,Tset);
    Tset[m]=true;
    for(int k = 0; k<6; k++)
    {
      // updating the distance of neighbouring vertex
      if(!Tset[k] && graph[m][k] && distance[m]!=INT_MAX &&
distance[m]+graph[m][k]<distance[k])
        distance[k]=distance[m]+graph[m][k];
    }
  }
  cout<<"Vertex\t\tDistance from source vertex"<<endl;
  for(int k = 0; k<6; k++)
  {
        char str=65+k;
        cout<<str<<"\t\t\t"<<distance[k]<<endl;
    }
}

int main()
{
    int graph[6][6]={
        {0, 1, 2, 0, 0, 0},
        {1, 0, 0, 5, 1, 0},
        {2, 0, 0, 2, 3, 0},
        {0, 5, 2, 0, 2, 2},
        {0, 1, 3, 2, 0, 1},
        {0, 0, 0, 2, 1, 0}};
    DijkstraAlgo(graph,0);
    return 0;
}
```

Output



```
student@cn-H81M-S:~/NMN/PartB$ g++ Prg3.cpp
student@cn-H81M-S:~/NMN/PartB$ ./a.out
Vertex          Distance from source vertex
A                       0
B                       1
C                       2
D                       4
E                       2
F                       3
```

Program 4

Aim: For the given data, use CRC-CCITT polynomial to implement CRC. Verify the program for the cases. i) Without error. ii) With error

Software/ OS: g++, Ubuntu (Linux platform)

Theory: This Cyclic Redundancy Check (CRC) is the most powerful, easy to implement technique and CRC is based on binary division. In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number. At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted. A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.

The generalized technique can be explained as follows. If a k bit message is to be transmitted, the transmitter generates an r-bit sequence, known as Frame Check Sequence (FCS) so that the (k+r) bits are actually being transmitted. Now this r-bit FCS is generated by dividing the original number, appended by r zeros, by a predetermined number. This number, which is (r+1) bit in length, can also be considered as the coefficients of a polynomial, called Generator Polynomial. The remainder of this division process generates the r-bit FCS. On receiving the packet, the receiver divides the (k+r) bit frame by the same predetermined number and if it produces no remainder, it can be assumed that no error has occurred during the transmission

Commonly used divisor polynomials are:

- CRC-16 = $X^{16} + X^{15} + X^2 + 1$
- CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$
- CRC-32 = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + 1$

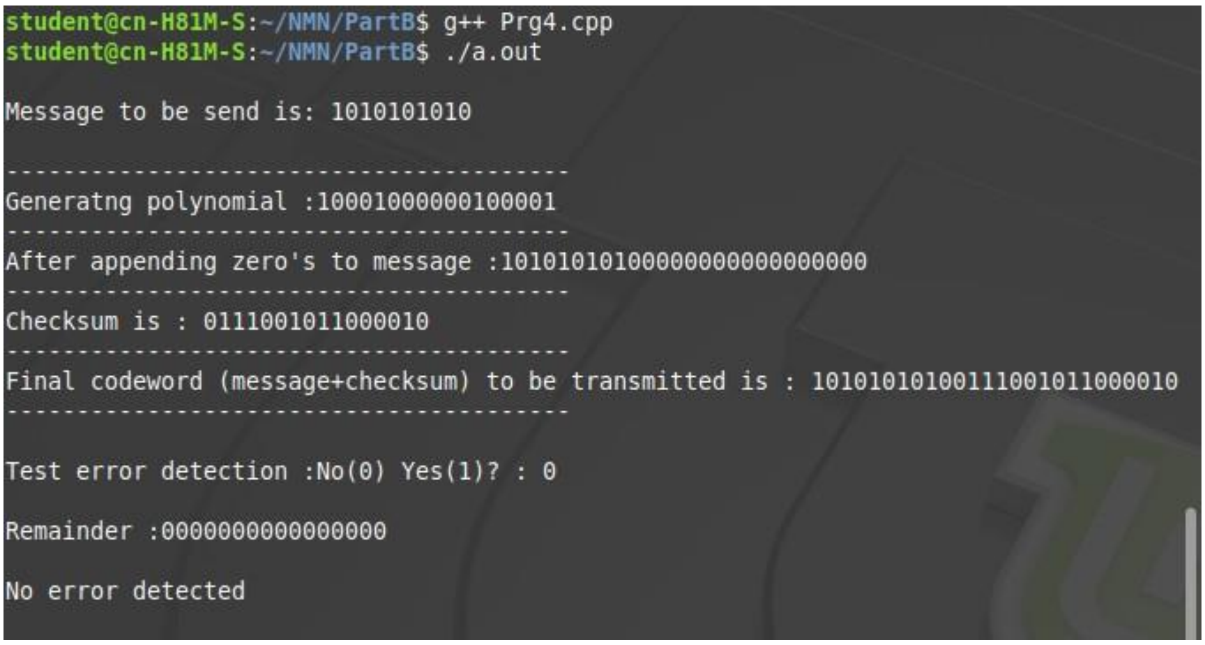
Program

```
#include<iostream>
using namespace std;
#include<string.h>
void xor1();
void crc();
char msg[28],cs[28],g[]="10001000000100001";
int msglen,i,j;
int main()
{
int genlen,err;
```

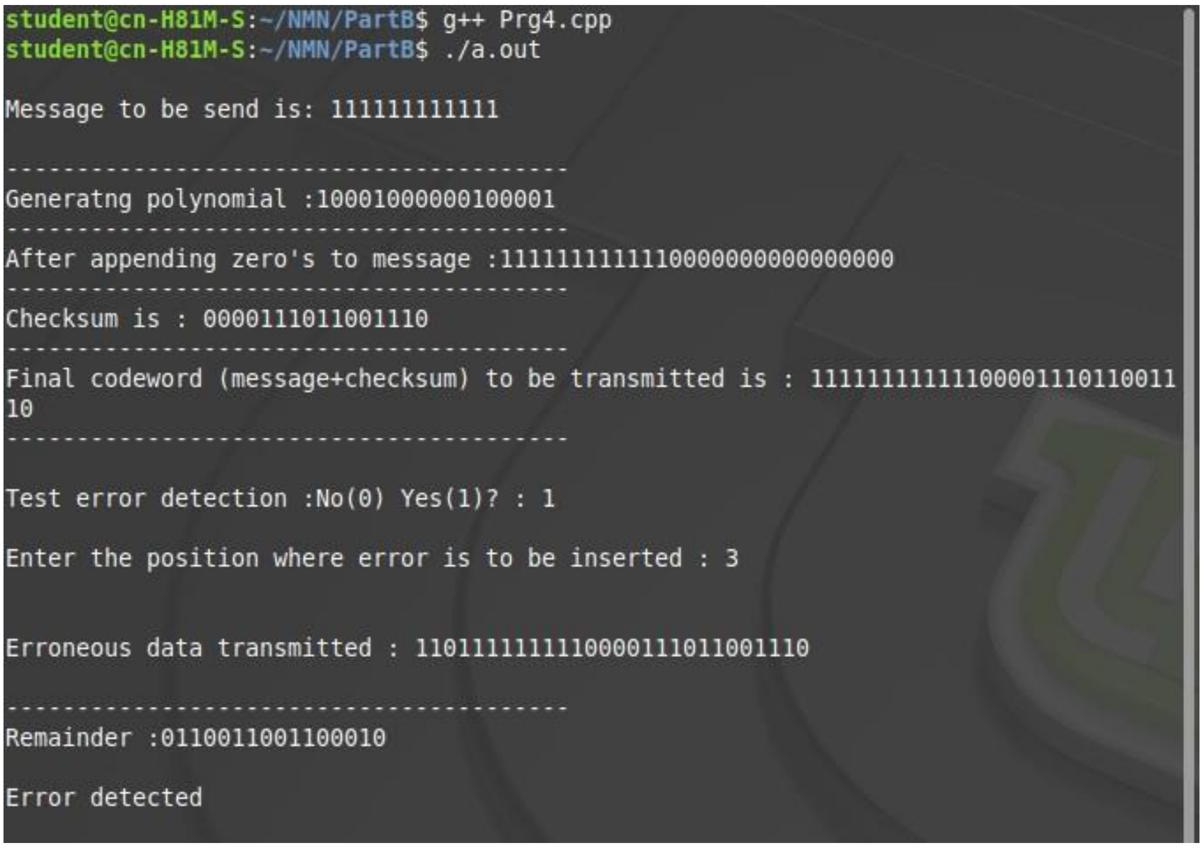
```
cout<<"\nMessage to be send is: ";
cin>>msg;
cout<<"\n--------------------------";
cout<<"\nGeneratng polynomial :"<<g;
msglen=strlen(msg);
genlen=strlen(g);
for(i=msglen;i<msglen+genlen-1;i++)//appending n-1 zeros
msg[i]='0';
cout<<"\n--------------------------";
cout<<"\nAfter appending zero's to message :"<<msg;
cout<<"\n--------------------------";
crc();
printf("\nChecksum is : %s",cs);
for(i=msglen;i<msglen+genlen-1;i++) //data+checksum
msg[i]=cs[i-msglen];
cout<<"\n--------------------------";
cout<<"\nFinal codeword (message+checksum) to be transmitted is : "<<msg;
cout<<"\n--------------------------";
cout<<"\n\nTest error detection :No(0) Yes(1)? : ";
cin>>err;
if(err==1)
{
do
{
cout<<"\nEnter the position where error is to be inserted : ";
cin>>err;
} while(err==1 || err>msglen+genlen-1);
msg[err-1]=(msg[err-1]=='0')?'1':'0';
cout<<"\n\nErroneous data transmitted : "<<msg<<endl;
cout<<"\n--------------------------";
}
crc();
cout<<"\nRemainder :"<<cs;
for(i=0;(i<genlen-1) && (cs[i]!='1');i++); //if 1 is encounter than the for loop will terminate
if(i<genlen-1)
cout<<"\n\nError detected\n\n";
else
cout<<"\n\nNo error detected\n\n";
cout<<"\n--------------------------\n";
```

```
return 0;
}
void xor1()
{
for(j = 1;j<strlen(g); j++)
cs[j] = (( cs[j] == g[j])?'0':'1');
}
void crc(){
for(i=0;i<strlen(g);i++)
cs[i]=msg[i];
do{
if(cs[0]=='1') // if first bit is 1 do xor, otherwise directly go for shifting
xor1();
for(j=0;j< strlen(g)-1;j++)
cs[j]=cs[j+1]; //shifting
cs[j]=msg[i++]; //droping next bit for division
}while(i<=msglen+strlen(g)-1);
}
```

Output:



```
student@cn-H81M-S:~/NMN/PartB$ g++ Prg4.cpp
student@cn-H81M-S:~/NMN/PartB$ ./a.out

Message to be send is: 1010101010

----------------------------------------
Generatng polynomial :10001000000100001
----------------------------------------
After appending zero's to message :10101010100000000000000000
----------------------------------------
Checksum is : 0111001011000010
----------------------------------------
Final codeword (message+checksum) to be transmitted is : 10101010100111001011000010
----------------------------------------

Test error detection :No(0) Yes(1)? : 0

Remainder :0000000000000000

No error detected
```



```
student@cn-H81M-S:~/NMN/PartB$ g++ Prg4.cpp
student@cn-H81M-S:~/NMN/PartB$ ./a.out

Message to be send is: 111111111111

----------------------------------------
Generatng polynomial :10001000000100001
----------------------------------------
After appending zero's to message :1111111111110000000000000000
----------------------------------------
Checksum is : 0000111011001110
----------------------------------------
Final codeword (message+checksum) to be transmitted is : 1111111111110000111011001110
10
----------------------------------------

Test error detection :No(0) Yes(1)? : 1

Enter the position where error is to be inserted : 3


Erroneous data transmitted : 1101111111110000111011001110

----------------------------------------
Remainder :0110011001100010

Error detected
```

Program 5

Aim: To implement Stop and Wait Protocol and Sliding Window Protocol for different number of frames/packets and analyses the output

Software/ OS: g++, Ubuntu (Linux platform)

Theory: Stop and Wait protocol is the simplest form of flow control where a sender transmits a data frame. After receiving the frame, the receiver indicates its willingness to accept another frame by sending back an ACK frame acknowledging the frame just received. The sender must wait until it receives the ACK frame before sending the next data frame.

This is sometimes referred to as ping-pong behaviour, request/reply is simple to understand and easy to implement, but not very efficient.

Program:

```
#include <iostream>
using namespace std;
#include <unistd.h>
#define RTT 4
#define TIMEOUT 4
#define TOT_FRAMES 7
enum {NO,YES} ACK;
int main()
{
int wait_time, i;
ACK=YES;
for(i=1;i<=TOT_FRAMES;)
{
if (ACK==YES && i!=1)
{
printf("\nSENDER: ACK for Frame %d Received.\n",i-1);
}
printf("\nSENDER: Frame %d sent.\nWaiting for ACK...\n",i);
ACK=NO;
wait_time= rand()% 4+1;
if (wait_time==TIMEOUT)
{
printf("SENDER: TIMEOUT \nACK not received for Frame %d",i);
printf("\n\nResending Frame %d...",i);
}
else
{
sleep(RTT/2);
printf("\nRECEIVER: Frame %d received, ACK sent\n",i);
printf("-------------------------------------------");
ACK=YES;
sleep(RTT/2);
i++;
}
}
return 0;
}
```

Output

```
student@cn-H81M-S:~/NMN/PartA$ g++ Prg5.cpp
student@cn-H81M-S:~/NMN/PartA$ ./a.out

SENDER: Frame 1 sent.
Waiting for ACK...
SENDER: TIMEOUT
ACK not received for Frame 1

Resending Frame 1...
SENDER: Frame 1 sent.
Waiting for ACK...

RECEIVER: Frame 1 received, ACK sent
-------------------------------------------
SENDER: ACK for Frame 1 Received.

SENDER: Frame 2 sent.
Waiting for ACK...

RECEIVER: Frame 2 received, ACK sent
-------------------------------------------
SENDER: ACK for Frame 2 Received.

SENDER: Frame 3 sent.
Waiting for ACK...
```

```
SENDER: TIMEOUT
ACK not received for Frame 3

Resending Frame 3...
SENDER: Frame 3 sent.
Waiting for ACK...

RECEIVER: Frame 3 received, ACK sent
-------------------------------------------
SENDER: ACK for Frame 3 Received.

SENDER: Frame 4 sent.
Waiting for ACK...
SENDER: TIMEOUT
ACK not received for Frame 4

Resending Frame 4...
SENDER: Frame 4 sent.
Waiting for ACK...

RECEIVER: Frame 4 received, ACK sent
-------------------------------------------
SENDER: ACK for Frame 4 Received.

SENDER: Frame 5 sent.
Waiting for ACK...

RECEIVER: Frame 5 received, ACK sent
-------------------------------------------
SENDER: ACK for Frame 5 Received.

SENDER: Frame 6 sent.
Waiting for ACK...

RECEIVER: Frame 6 received, ACK sent
-------------------------------------------
SENDER: ACK for Frame 6 Received.

SENDER: Frame 7 sent.
```

```
Waiting for ACK...

RECEIVER: Frame 7 received, ACK sent
student@cn-H81M-S:~/NMN/PartA$
```

Sliding Window Protocol: With the use of multiple frames for a single message, the stop-and-wait protocol does not perform well. Only one frame at a time can be in transit. In stop-and-wait flow control, if a > 1, serious inefficiencies result. Efficiency can be greatly improved by allowing multiple frames to be in transit at the same time. Efficiency can also be improved by making use of the full-duplex line. To keep track of the frames, sender station sends sequentially numbered frames. Since the sequence number to be used occupies a field in the frame, it should be of limited size. If the header of the frame allows k bits, the sequence numbers range from 0 to 2k – 1. Sender maintains a list of sequence numbers that it is allowed to send (sender window). The size of the sender's window is at most 2k – 1. The sender is provided with a buffer equal to the window size. Receiver also maintains a window of size 2k – 1. The receiver acknowledges a frame by sending an ACK frame that includes the sequence number of the next frame expected. This also explicitly announces that it is prepared to receive the next N frames, beginning with the number specified. This scheme can be used to acknowledge multiple frames. It could receive frames 2, 3, 4 but withhold ACK until frame 4 has arrived. By returning an ACK with sequence number 5, it acknowledges frames 2, 3, 4 in one go. The receiver needs a buffer of size 1.

Program
```
#include<iostream>
using namespace std;
#include <unistd.h>
#include <stdlib.h>

#define RTT 5
int main()
{
int window_size,i,f,frames[50];
printf("Enter window size: ");
scanf("%d",&window_size); // read window size
printf("\nEnter number of frames to transmit: ");
scanf("%d",&f); // read no. of frames
printf("\nEnter %d frames: ",f);
for(i=1;i<=f;i++)
scanf("%d",&frames[i]); //read frame values
printf("\nAfter sending %d frames at each stage sender waits for ACK",window_size);
printf("\nSending frames in the following manner ...\n\n");
for(i=1;i<=f;i++)
{
if(i%window_size!=0) // collect the frames to fit in window
{
printf("\n Sending Frame no %d ",frames[i]);
}
else
{
printf("\n Sending Frame no %d ",frames[i]); // send the frames
printf("\n SENDER:waiting for ACK for %d frame ...\n\n",i);
sleep(RTT/2); // wait for RTT/2
printf("RECEIVER: %d Frame Received, ACK Sent \n",i);
printf("..................................................\n");
sleep(RTT/2); // wait for RTT/2
printf("SENDER:ACK received for the frame%d, sending next frames\n",i);
}
}
if(f%window_size!=0) // send the left over frames
{
printf("\nSENDER:waiting for ACK...\n");
sleep(RTT/2);// wait for RTT/2
printf("\nRECEIVER:Frames Received, ACK Sent\n");
printf("..........................................................\n");
sleep(RTT/2); // wait for RTT/2
printf("\n SENDER:ACK received.");
}
return 0;
}
```

Output

```
student@cn-H81M-S:~/NMN/PartB$ g++ Prg6.cpp
student@cn-H81M-S:~/NMN/PartB$ ./a.out
Enter window size: 3

Enter number of frames to transmit: 10

Enter 10 frames: 1 2 3 4 5 6 7 8 9 10

After sending 3 frames at each stage sender waits for ACK
Sending frames in the following manner....

 Sending Frame no 1
 Sending Frame no 2
 Sending Frame no 3
 SENDER:waiting for ACK for 3 frame ...

RECEIVER: 3 Frame Received, ACK Sent

SENDER:ACK received for the frame3, sending next frames

 Sending Frame no 4
 Sending Frame no 5
 Sending Frame no 6
 SENDER:waiting for ACK for 6 frame ...

RECEIVER: 6 Frame Received, ACK Sent

SENDER:ACK received for the frame6, sending next frames

 Sending Frame no 7
 Sending Frame no 8
 Sending Frame no 9
 SENDER: waiting for ACK for 9 frame ...

RECEIVER: 9 Frame Received, ACK Sent

SENDER: ACK received for the frame9, sending next frames

Sending Frame no 10
SENDER: waiting for ACK...

RECEIVER: Frames Received, ACK Sent
________________________________________
```

Program 6

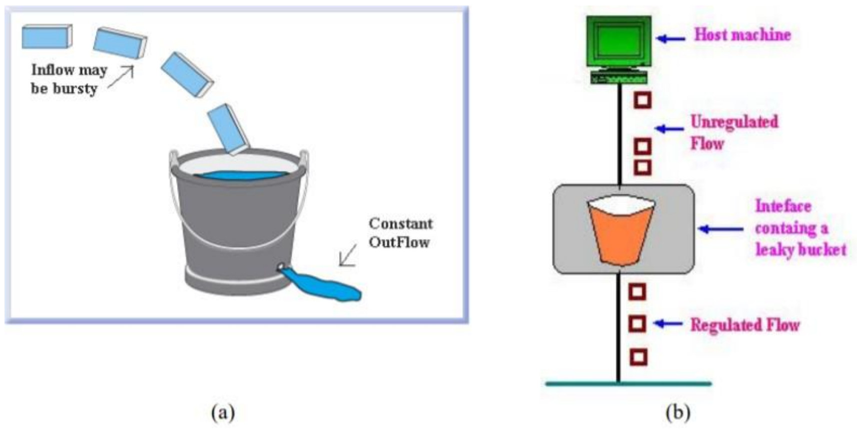
Aim: To write a program for congestion control using leaky bucket algorithm and analyse the output for different output rate and bucket size

Software/ OS:gcc, Ubuntu (Linux platform)

Theory: Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant. This scenario is depicted in figure (a). Once the bucket is full, any additional water entering it spills over the sides and is lost. The same idea of leaky bucket can be applied to packets, as shown in figure (b). Conceptually each network interface contains a leaky bucket. And the following steps are performed:

- When the host has to send a packet, the packet is thrown into the bucket.
- The bucket leaks at a constant rate, meaning the network interface transmits packets at constant rate.
- Bursty traffic is converted to a uniform traffic by the leaky bucket.
- In practice the bucket is a finite queue that outputs at a finite rate.

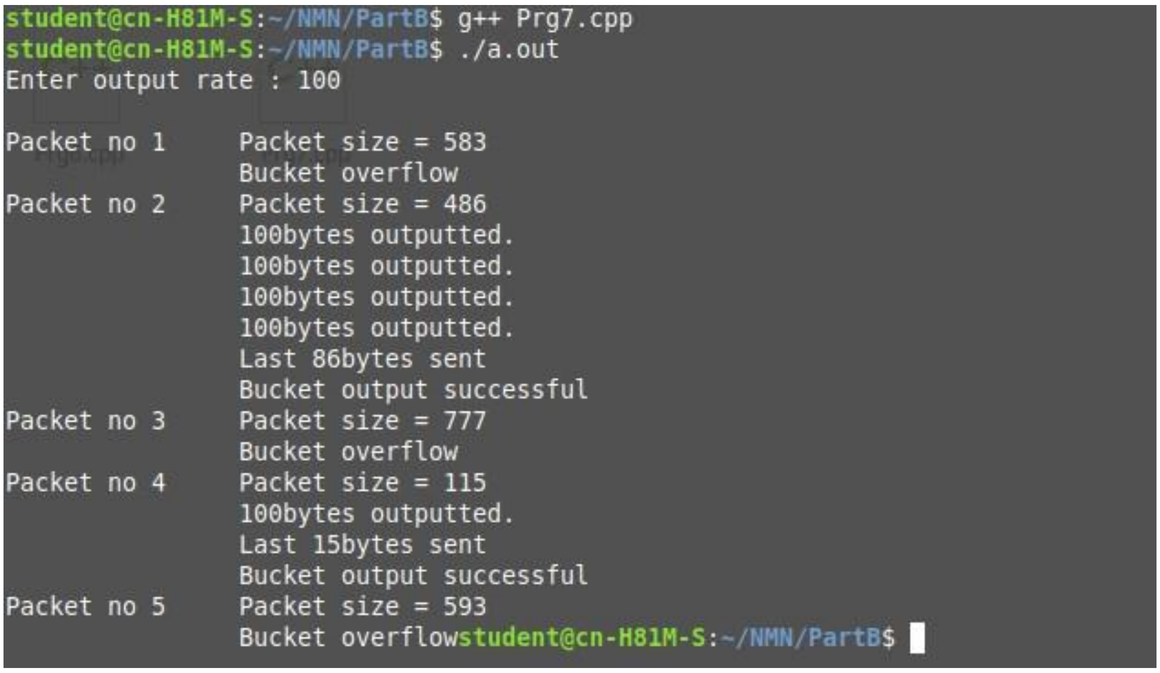
This arrangement can be simulated in the operating system or can be built into the hardware. Implementation of this algorithm is easy and consists of a finite queue. Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded.



Program
```
#include<iostream>
using namespace std;
#include <unistd.h>
```

```
#define bucketSize 512
void bktInput(int a,int b)
{
if(a>bucketSize)
   cout<<"\n\t\tBucket overflow";
else {
   sleep(2);
while(a>b){
   cout<<"\n\t\t"<<b<<"bytes outputted.";
   a-=b;
sleep(2);
}
if (a>0)
cout<<"\n\t\tLast "<<a<<"bytes sent\t";
cout<<"\n\t\tBucket output successful";
}
}
int main()
{
   int op, pktSize;

   cout<<"Enter output rate : "; cin>>op;
   for(int i=1;i<=5;i++)
   {
     sleep(4);
     pktSize=rand()%800;
     cout<<"\nPacket no "<<i<<"\tPacket size = "<<pktSize;
     bktInput(pktSize,op);
   }
return 0;
}
```

Output


```
student@cn-H81M-S:~/NMN/PartB$ g++ Prg7.cpp
student@cn-H81M-S:~/NMN/PartB$ ./a.out
Enter output rate : 100

Packet no 1     Packet size = 583
                Bucket overflow
Packet no 2     Packet size = 486
                100bytes outputted.
                100bytes outputted.
                100bytes outputted.
                100bytes outputted.
                Last 86bytes sent
                Bucket output successful
Packet no 3     Packet size = 777
                Bucket overflow
Packet no 4     Packet size = 115
                100bytes outputted.
                Last 15bytes sent
                Bucket output successful
Packet no 5     Packet size = 593
                Bucket overflowstudent@cn-H81M-S:~/NMN/PartB$ 
```