||Jai Sri Gurudev ||
BGSKH Education Trust (R.) – A unit of Sri Adichunchanagiri Shikshana Trust(R.)
BGS College of Engineering and Technology (BGSCET)
(Approved by AICTE, New Delhi and Affiliated to VTU, Belagavi)
Adjacent to Mahalakshmi Metro Station, Mahalakshmipuram, West of Chord Road, Bengaluru -560 086, Karnataka

COMPUTER NETWORKS LABORATORY

For Fifth Semester B.E-2022 Batch [VTU/CBCS, 2022 syllabus]

Subject Code – BCS502

Prepared by:

Manjula L
Assistant professor, Dept. of AI & ML
BGSCET

Vision

"Creating Competent IT Professionals with Core Values for The Real World."

Mission

- Providing Students with a Sound Knowledge in IT Fundamentals.
- Exposing Students to Emerging Frontiers in various domains of IT enabling Continuous Learning.
- Promoting Excellence in Teaching, Training, Research and Consultancy.
- Developing Entrepreneurial acumen to venture into Innovative areas of IT.
- Imparting value based Professional Education with a sense of Social Responsibility.

COMPUTER NETWORKS LABORATORY-BCS502
INTERNAL EVALUATION SHEET

EVALUATION (MAX MARKS 50)			
TEST A	**REGULAR EVALUATION B**	**RECORD C**	**TOTAL MARKS A+B+C**
10	**5**	**10**	**25**

R1: REGULAR LAB EVALUATION WRITE UP RUBRIC (MAX MARKS 10)				
Sl. No.	**Parameters**	**Good**	**Average**	**Needs improvement**
a.	**Understanding of problem (3 marks)**	Clear understanding of problem statement while designing and implementing the program (3)	Problem statement is understood clearly but few mistakes while designing and implementing program (2)	Problem statement is not clearly understood while designing the program (1)
b.	**Writing program (4 marks)**	Program handles all possible conditions (4)	Average condition is defined and verified. (3)	Program does not handle possible conditions (1)
c.	**Result and documentation (3 marks)**	Meticulous documentation and all conditions are taken care (3)	Acceptable documentation shown (2)	Documentation does not take care all conditions (1)

R2: REGULAR LAB EVALUATION VIVA RUBRIC (MAX MARKS 10)					
Sl. No.	**Parameter**	**Excellent**	**Good**	**Average**	**Needs Improvement**
a.	**Conceptual understanding (10 marks)**	Answers 80% of the viva questions asked (10)	Answers 60% of the viva questions asked (7)	Answers 30% of the viva questions asked (4)	Unable to relate the concepts (1)

R3: REGULAR LAB PROGRAM EXECUTION RUBRIC (MAX MARKS 10)				
Sl. No.	**Parameters**	**Excellent**	**Good**	**Needs Improvement**
a.	**Design, implementation and demonstration (5 marks)**	Program follows syntax and semantics of C programming language. Demonstrates the complete knowledge of the program written (5)	Program has few logical errors, moderately demonstrates all possible concepts implemented in programs (3)	Syntax and semantics of C programming is not clear (1)
b.	**Result and documentation (5 marks)**	All test cases are successful, all errors are debugged with own practical knowledge and clear documentation according to the guidelines (5)	Moderately debugs the programs , few test case are unsuccessful and Partial documentation (3)	Test cases are not taken care , unable to debug the errors and no proper documentation (1)

R4: RECORD EVALUATION RUBRIC (MAX MARKS 10)					
Sl. No.	**Parameter**	**Excellent**	**Good**	**Average**	**Needs Improvement**
a.	**Documentation (10 marks)**	Meticulous record writing including program, comments and test cases as per the guidelines mentioned (20)	Write up contains program and test cases, but comments are not included (18)	Write up contains only program (15)	Program written with few mistakes (10)

TEST /LAB INTERNALS MARKS (MAX MARKS 5)

TEST #	Write up 5	Execution 10	Viva 5	Sign	Total 20	Avg. 40	Final 10
TEST-1							
TEST-2							

REGULAR LAB EVALUATION (MAX MARKS 5)

Lab program	Date of Execution	Additional programs	Write up (5)	Execution (5)	Viva (5)	Total 15	Teacher Signature
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
Total Marks		**105**				**5**	

Final Marks obtained from test (05) + regular evaluation (05)	10	Lab in charge : **HOD:**
Record(10)	10	
Total Marks Obtained		

PREFACE

We have developed this comprehensive laboratory manual on **COMPUTER NETWORKS LABORATORY-BCS502** with two primary objectives: To make the students comfortable with basic principles of problem solving and to train them in evolving as an efficient COMPUTER NETWORKS by strengthening their CN programming abilities.

This material provides students an exposure to problem solving approaches and solution to number of problems using Java and NS-3 programming. The problems discussed in this manual comprises of an algorithm, programming solution, alternative logic and extensive test cases. Viva questions, frequently appeared examination questions and practicing programming problems constitute an indispensable part of this material.

Our profound and sincere efforts will be fruitful only when students acquire the extensive knowledge by reading this manual and apply the concepts learnt apart from the requirements specified in COMPUTER NETWORKS LABORATORY-BCS502 as prescribed by VTU, Belagavi.

TABLE OF CONTENTS

INTRODUCTION TO NS3

NS3 Simulator Basics

- NS-3 is a network simulator
- Developed for network research and education
- Developed afterns-2
- ns-3 is written in C++
- Bindings in Python
- ns-3 uses the waf build system

Waf is a build automation tool designed to assist in the automatic compilation and installation of computer software. It is written inPython.

Waf features:

- Portable to Unix and non-Unix systems
- Lightweight
- Offers a Turing-complete programming language
- Support for standard targets: configure, build, clean, install, and uninstall
- Parallel builds
- Colored output and progress bar display
- Scripts are Python modules
- XML script front-end and a dedicated, easy-to-parse "IDE output" mode to ease the interaction with integrated development environments
- Modular configuration scheme with customizable command-line parsing
- Daemon mode for background recompilation
- Find source files intelligently (glob()-like) to ease script maintenance
- Support for global object cache to avoid unnecessary recompilations
- Support for unit tests run on programs at the end of builds

Waf supports:

- AC/C++preprocessor for computing dependencies simulation programs are C++executables or python scripts

Features

- It is a discrete event simulator
- Modular design / Opensource
- Actively developed (ContrastNS-2)
- Developed in C++. Python binding available.
- Live visualizer
- Logging facility for debugging

- Tracing facility for getting output
- Can be connected to a real network
- Direct Code Execution(DCE)

How to install ns3?
Download tarball from www.nsnam.orgthe recent release in your directory. Go to that directory and untar the tarball.

tar xvfj ns3.tar

It creates the directory for ns3 change to it.

cd ns3

For compiling and installing

./build.py --enable-examples --enable-tests

How to run ns3 script?
To test the installation copy one example available in the distribution to scratch directory and build and run the same using the commands below:

cd ns3.26
cp examples/tutorial/first.cc scratch/first.cc
./waf --run scratch/first

Steps in writing scripts

- Include necessary files
- Use appropriate namespace
- Set simulation time resolution(Optional)
- Enable logging for different modules(Optional)
- Create nodes
- Create net devices with MAC and PHY
- Attach Net devices to nodes and set interconnections
- Install protocol stack In nodes
- Set network address for interfaces
- Setup routing
- Install applications in nodes
- Setup tracing(Optional)
- Set application start and stop time
- Set simulation start time(Optional)
- Run simulation
- Release resources at end of simulation

Tutorial : First.cc

Simple point to point (Wired network) link between server and client is established here. This program is in your NS3 repository.(example/tutorial/first.cc)

Note : To know about NS3, you must have the base knowledge in c++ and OOPS concept.

1. ModuleIncludes

```
#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"
```

Each of the ns-3 include files is placed in a directory called ns3 (under the build directory) during the build process to help avoid include file name collisions. The ns3/core-module.h file corresponds to the ns-3 module you will find in the directorysrc/core in your downloaded release distribution. If you list this directory you will find a large number of header files. When you do a build, Waf will place public header files in an ns3 directory under the appropriate build/debug or build/optimized directory depending on your configuration. Waf will also automatically generate a module include file to load all of the public header files.

2. NS3Namespace

using namespace ns3;

The ns-3 project is implemented in a C++ namespace called ns3. This groupsall ns-3-related declarations in a scope outside the global namespace, which we hope will help with integration with other code. The C++ using statement introduces the ns-3 namespace into the current (global) declarative region. This is a fancy way of saying that after this declaration, you will not have to type ns3:: scope resolution operator before all of the ns-3 code in order to use it. If you are unfamiliar with namespaces, please consult almost any C++ tutorial and compare the ns3 namespace and usage here with instances of the std namespace and the using namespace std; statements you will often find in discussions of cout andstreams.

3. Set simulation timeresolution

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

This is just the declaration of the main function of your program (script). Just as in any C++ program, you need to define a main function that will be the first function run. There is nothing at all special here. Your ns3 script is just a C++program.

The next line sets the time resolution to one nanosecond, which happens to be the default value:

Time::SetResolution (Time::NS);

The resolution is the smallest time value that can be represented (as well as the smallest represent able difference between two time values). You can change the resolution exactly once. The mechanism enabling this flexibility is somewhat memory hungry, so once the resolution has been set explicitly we release the memory, preventing further updates. (If you don't set the resolution explicitly, it will default to one nanosecond, and the memory will be released when the simulation starts.)

4. Enable logging for differentmodules

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

this line declares a logging component called FirstScriptExample that allows you to enable and disable console message logging by reference to thename.

5. Createnodes

NodeContainer

nodes;nodes.Create (2);

The NodeContainer topology helper provides a convenient way to create, manage and access any Node objects that we create in order to run a simulation. The first line above just declares a NodeContainer which we call nodes. The second line calls the Create method on the nodes object and asks the container to create twonodes.

6. Create net devices with MAC andPHY

PointToPointHelperpointToPoint;

It instantiates a PointToPointHelper object on the stack. From a high-level perspective the next line,

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

Above line tells the PointToPointHelper object to use the value "5Mbps" (five megabits per second) as the "DataRate" when it creates a PointToPointNetDevice object. From a more detailed perspective, the string "DataRate" corresponds to what we call an Attribute of the PointToPointNetDevice.

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

It tells the PointToPointHelper to use the value "2ms" (two milliseconds) as the value of the transmission delay of every point to point channel it subsequentlycreates.

7. Attach Net devices to nodes and setinterconnections

NetDeviceContainerdevices;

devices = pointToPoint.Install (nodes);

The first line declares the device container mentioned above and the second does the heavy lifting. The Install method of the PointToPointHelper takes a NodeContainer as a parameter. Internally, a NetDeviceContainer is created. For each node in the NodeContainer (there must be exactly two for a point- to-point link) a PointToPointNetDevice is created and saved in the devicecontainer.

A PointToPointChannel is created and the two PointToPointNetDevices are attached. When objects are created by the PointToPointHelper, the Attributes previously set in the helper are used to initialize the corresponding Attributes in the created objects. After executing the pointToPoint.Install (nodes) call we will have two nodes, each with an installed point-to-point net device and a single point-to-point channel

between them. Both devices will be configured to transmit data at five megabits per second over the channel which has a two millisecond transmissiondelay.

8. Install protocol stack innodes

The only user-visible API is to set the base IP address and network mask to use when performing the actual address allocation (which is done at a lower level inside the helper).

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

It declares an address helper object and tell it that it should begin allocating IP addresses from thenetwork 10.1.1.0 using the mask 255.255.255.0 to define the allocatable bits. By default the addresses allocated will start at one and increase monotonically, so the first address allocated from this base will be 10.1.1.1, followed by 10.1.1.2, etc. The low level ns3 system actually remembers all of the IP addresses allocated and will generate a fatal error if you accidentally cause the same address to be generated twice (which is a very hard to debug error, by theway).

9. Set network address forinterfaces

Ipv4InterfaceContainer interfaces = address.Assign (devices);

It performs the actual address assignment. In ns-3 we make the association between an IP address and a device using an Ipv4Interface object. Just as we sometimes need a list of net devices created by a helper for future reference we sometimes need a list of Ipv4Interface objects. The Ipv4InterfaceContainer provides this functionality. Now we have a point-to-point network built, with stacks installed and IP addresses assigned. What we need at this point are applications to generatetraffic.

10. Setuprouting

UdpEchoServerHelperechoServer (9);

ApplicationContainerserverApps = echoServer.Install (nodes.Get (1));

The first line of code in the above snippet declares the UdpEchoServerHelper. As usual, this isn't the application itself, it is an object used to help us create the actual applications. One of our conventions is to place required Attributes in the helper constructor. In this case, the helper can't do anything useful unless it is provided with a port number that the client also knows about. Rather than just picking one and hoping it all works out, we require the port number as a parameter to the constructor. The constructor, in turn, simply does a SetAttribute with the passed value. If you want, you can set the "Port" Attribute to another value later usingSetAttribute.

Similar to many other helper objects, the UdpEchoServerHelper object has an Install method. It is the execution of this method that actually causes the underlying echo server application to be instantiated and attached to a node. Interestingly, the Install method takes a NodeContainter as a parameter just as the other

Install methods we have seen. This is actually what is passed to the method even though it doesn't look so in this case. There is a C++ implicit conversion at work here that takes the result of nodes.Get (1) (which returns a smart pointer to a node object — Ptr<Node>) and uses that in a constructor for an unnamed NodeContainer that is then passed to Install. If you are ever at a loss to find a particular method signature in C++ code that compiles and runs just fine, look for these kinds ofimplicit conversions.

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

Above lines cause the echo server application to Start (enable itself) at one second into the simulation and to Stop (disable itself) at ten seconds into the simulation. By virtue of the fact that we have declared a simulation event (the application stop event) to be executed at ten seconds, the simulation will last at least tenseconds.

11. Install applications innodes

UdpEchoClientHelperechoClient (interfaces.GetAddress (1), 9);

echoClient.SetAttribute ("MaxPackets", UintegerValue (1));

echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));

echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

For the echo client, however, we need to set five different Attributes. The first two Attributes are set during construction of the UdpEchoClientHelper. We pass parameters that are used (internally to the helper) to set the "RemoteAddress" and "RemotePort" Attributes in accordance with our convention to make required Attributes parameters in the helperconstructors.

The zeroth interface in the interfaces container is going to correspond to the IP address of the zeroth node in the nodes container. The first interface in the interfaces container corresponds to the IP address of the first node in the nodes container. So, in the first line of code (from above), we are creating the helper and telling it so set the remote address of the client to be the IP address assigned to the node on which the server resides. We also tell it to arrange to send packets to portnine.

The "MaxPackets" Attribute tells the client the maximum number of packets we allow it to send during thesimulation.

The "Interval" Attribute tells the client how long to wait between packets, and the "PacketSize" Attribute tells the client how large its packet payloads should be. With this particular combination of Attributes, we are telling the client to send one 1024-bytepacket.

12. Set application start and stoptime

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

First it will run the event at 1.0 seconds, which will enable the echo server application (this event may, in turn, schedule many other events). Then it will run the event scheduled for t=2.0 seconds which will start the echo client application. Again, this event may schedule many more events. The start event implementation in the echo client application will begin the data transfer phase of the simulation by sending a packet to theserver.

13. Runsimulation

Simulator::Run ();

When Simulator::Run is called, the system will begin looking through the list of scheduled events and executing them. Eventually, since we only send one packet (recall the MaxPackets Attribute was set to one), the chain of events triggered by that single client echo request will taper off and the simulation will go idle. Once this happens, the remaining events will be the Stop events for the server and the client. When these events are executed, there are no further events to process and Simulator::Run returns. The simulation is thencomplete.

14. Release resources at end ofsimulation

All that remains is to clean up. This is done by calling the global function Simulator::Destroy. As the helper functions (or low level ns-3 code) executed, they arranged it so that hooks were inserted in the simulator to destroy all of the objects that were created. You did not have to keep track of any of these objects yourself — all you had to do was to call Simulator::Destroy and exit. The ns-3 system took care of the hard part for you. The remaining lines of our first ns-3 script, first.cc, do just that:

Simulator::Destroy ();

1. **Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.**

Point-to-point network topology is a simple topology that displays the network of exactly two hosts (computers, servers, switches or routers) connected with a cable. Point-to-point topology is widely used in the computer networking and computer architecture. It is also used in the telecommunications systems when we speak about the communication connection of two nodes or endpoints.

Networktopology

```
      10.1.1.0       10.1.2.0
n0----------------------n1.......... n2
point-to-point
```

Inthisprogramwehavecreated3pointtopointnodesn0,n1,n2.Node n0 has IP address 10.1.1.1 and n3 has 10.1.2.2. Node n1 has 2 interfaces(10.1.1.2and10.1.2.1). OnOffHelper application is used to generatethetrafficatsourcenoden0.Packetsmovefromn0ton2via n1. Acknowledgment is sent from n2 to n0 via n1. Details of the flow(Number of packets sent, received and dropped) can be verified by using tracemetrics(lab1.trfile).

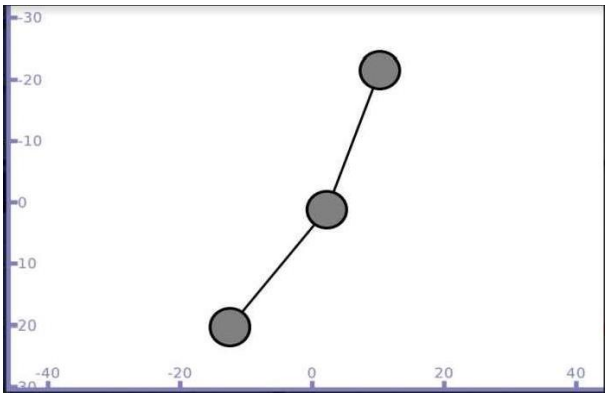
Program:

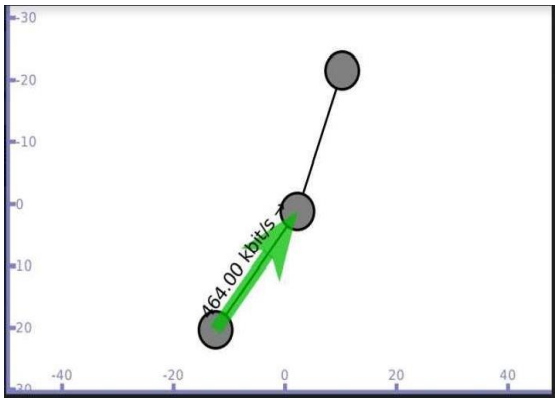
```
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/traffic-control-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("FirstScriptExample");

int main(int argc, char* argv[])
{
   double simulationTime = 10; // seconds     // add from traffic_control.cc
   CommandLine cmd(_FILE_);
   cmd.Parse(argc, argv);

   Time::SetResolution(Time::NS);
```

```
LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);

NodeContainer nodes;  //create nodes
nodes.Create(3);


//Vary the Bandwidth and delay and check the Packet drop
PointToPointHelper pointToPoint;                              //point to point link
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));


 NetDeviceContainer devices01;
devices01 = pointToPoint.Install(nodes.Get(0),nodes.Get(1));    // add Device interfaces

NetDeviceContainer devices12;
devices12 = pointToPoint.Install(nodes.Get(1),nodes.Get(2));

InternetStackHelper stack;
stack.Install(nodes);

Ipv4AddressHelper address1;
address1.SetBase("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces01 = address1.Assign(devices01);


Ipv4AddressHelper address2;
address2.SetBase("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces12 = address2.Assign(devices12);

Ipv4GlobalRoutingHelper::PopulateRoutingTables();     //add from third.cc

uint16_t port = 7;
Address localAddress(InetSocketAddress(Ipv4Address::GetAny(), port));
PacketSinkHelper packetSinkHelper("ns3::TcpSocketFactory", localAddress);
ApplicationContainer sinkApp = packetSinkHelper.Install(nodes.Get(2));

sinkApp.Start(Seconds(0.0));
sinkApp.Stop(Seconds(simulationTime + 0.1));

uint32_t payloadSize = 1448;
Config::SetDefault("ns3::TcpSocket::SegmentSize", UintegerValue(payloadSize));

OnOffHelper onoff("ns3::TcpSocketFactory", Ipv4Address::GetAny());
onoff.SetAttribute("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));
onoff.SetAttribute("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));
onoff.SetAttribute("PacketSize", UintegerValue(payloadSize));
```

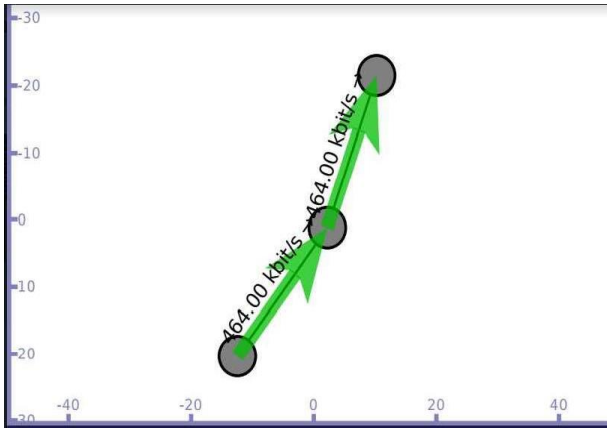
```
    onoff.SetAttribute("DataRate", StringValue("50Mbps")); // bit/s
    ApplicationContainer apps;

    InetSocketAddress rmt(interfaces12.GetAddress(1), port);
    rmt.SetTos(0xb8);
    AddressValue remoteAddress(rmt);
    onoff.SetAttribute("Remote", remoteAddress);
    apps.Add(onoff.Install(nodes.Get(0)));
    apps.Start(Seconds(1.0));
    apps.Stop(Seconds(simulationTime + 0.1));

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

    Simulator::Stop(Seconds(simulationTime + 5));
    Simulator::Run();

    Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier>(flowmon.GetClassifier());
    std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats();
    std::cout << std::endl << "* Flow monitor statistics *" << std::endl;
    std::cout << "  Tx Packets/Bytes:   " << stats[1].txPackets << " / " << stats[1].txBytes
          << std::endl;
    std::cout << "  Offered Load: "
          << stats[1].txBytes * 8.0 /
               (stats[1].timeLastTxPacket.GetSeconds() -
                stats[1].timeFirstTxPacket.GetSeconds()) /
               1000000
          << " Mbps" << std::endl;
    std::cout << "  Rx Packets/Bytes:   " << stats[1].rxPackets << " / " << stats[1].rxBytes
          << std::endl;

    Simulator::Run();
    Simulator::Destroy();
    return 0;
}
```

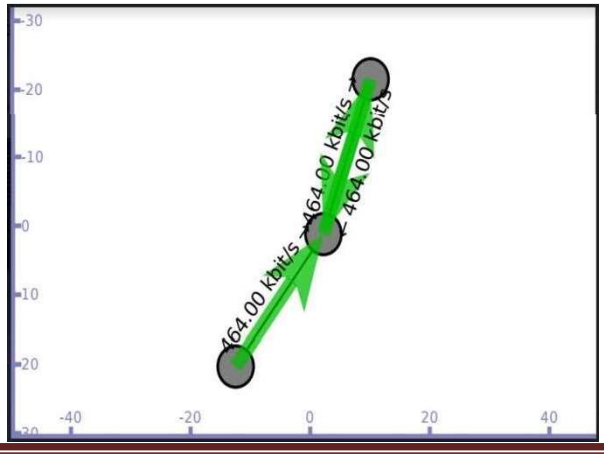
Output:





Acknowledgment sent fromn2





Plot the comparison graph

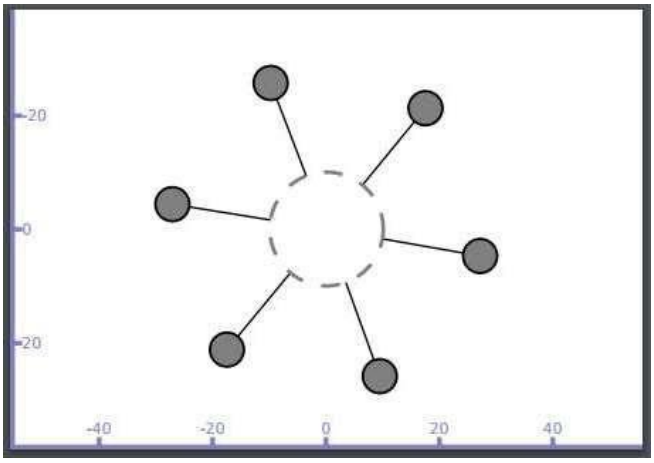
2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

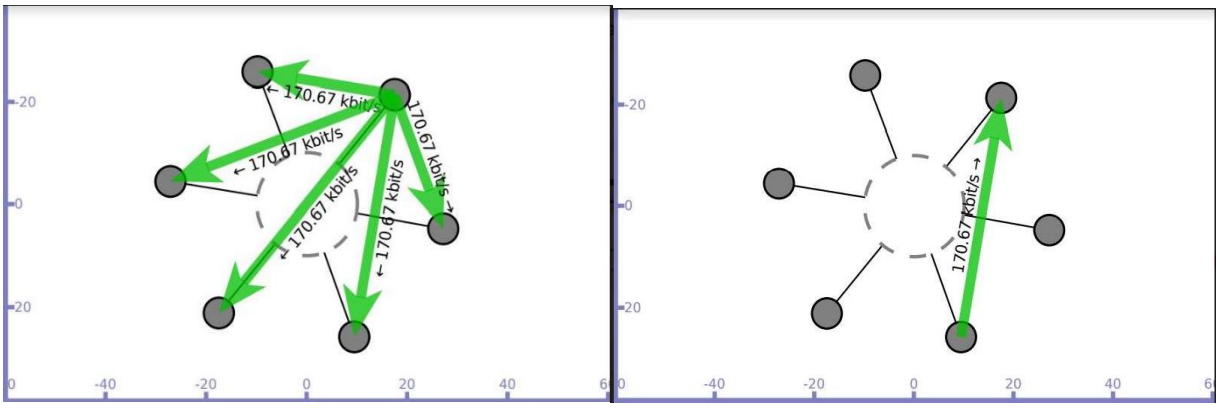
Program:

```
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/internet-apps-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("SecondScriptExample");

int
main(int argc, char* argv[])
{
   bool verbose = true;
   uint32_t nCsma = 6;
   Time interPacketInterval{Seconds(1.0)};
   uint32_t size{56};
   uint32_t count{5};
   double error_p = 0.25;

   CommandLine cmd(_FILE_);
   cmd.AddValue("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
   cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);

   cmd.Parse(argc, argv);

   if (verbose)
   {
      LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
      LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
   }

   nCsma = nCsma == 0 ? 1 : nCsma;


   NodeContainer csmaNodes;
   csmaNodes.Create(nCsma);

   Ptr<UniformRandomVariable> uv = CreateObject<UniformRandomVariable>();
```

```
//introduce error module from tcp-variants-comparision.cc
   uv->SetStream(50);
   RateErrorModel error_model;
   error_model.SetRandomVariable(uv);
   error_model.SetUnit(RateErrorModel::ERROR_UNIT_PACKET);
   error_model.SetRate(error_p);

   CsmaHelper csma;
   csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
   csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));
   csma.SetDeviceAttribute("ReceiveErrorModel", PointerValue(&error_model));

   NetDeviceContainer csmaDevices;
   csmaDevices = csma.Install(csmaNodes);

   InternetStackHelper stack;
   stack.Install(csmaNodes);

    Ipv4AddressHelper address;
   address.SetBase("10.1.2.0", "255.255.255.0");
   Ipv4InterfaceContainer csmaInterfaces;
   csmaInterfaces = address.Assign(csmaDevices);


    // Create Ping application and installing on node A
    //Add from Pingexample.cc

   PingHelper pingHelper(csmaInterfaces.GetAddress(1, 0));
   pingHelper.SetAttribute("Interval", TimeValue(interPacketInterval));
   pingHelper.SetAttribute("Size", UintegerValue(size));
   pingHelper.SetAttribute("Count", UintegerValue(count));
   ApplicationContainer apps = pingHelper.Install(csmaNodes.Get(0));
   apps.Start(Seconds(1));
   apps.Stop(Seconds(50));


   Simulator::Run();
   Simulator::Destroy();
   return 0;
}
```

Output:





3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

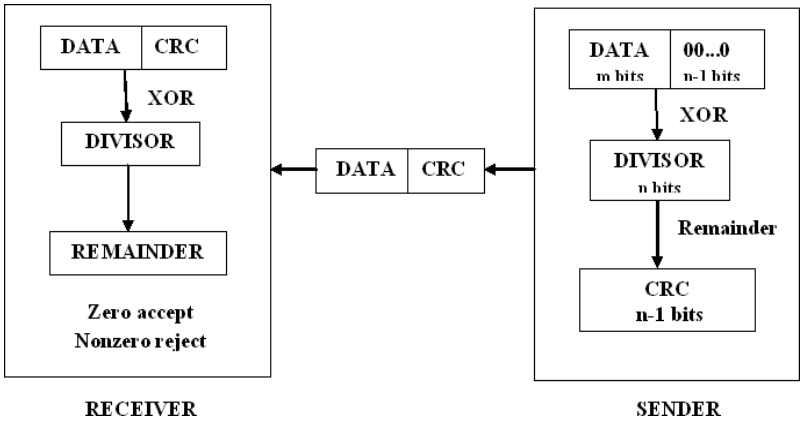
examples/tcp/tcp-variants-comparison.cc Source File

make tracing= true

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

The cyclic redundancy check (CRC) is a technique used to detect errors in digital data. CRC is a hash function that detects accidental changes to raw computer data commonly used in digital telecommunications networks and storage devices such as hard disk drives. This technique was invented by W. Wesley Peterson in 1961 and further developed by the CCITT (International Telegraph and Telephone Consultative Committee). Cyclic redundancy checks are quite simple to implement in hardware and can be easily analyzed mathematically. It is one of the better techniques in detecting common transmission errors.

As explained above cyclic redundancy check is also applied to storage devices like hard disks. In this case, check bits are allocated to each block in the hard disk. When a corrupt or incomplete file is read by the computer, the cyclic redundancy error is reported. This could be from another storage device or from CD/DVDs. The common reasons for errors include system crashes, incomplete or corrupt files, or files with lots of bugs.



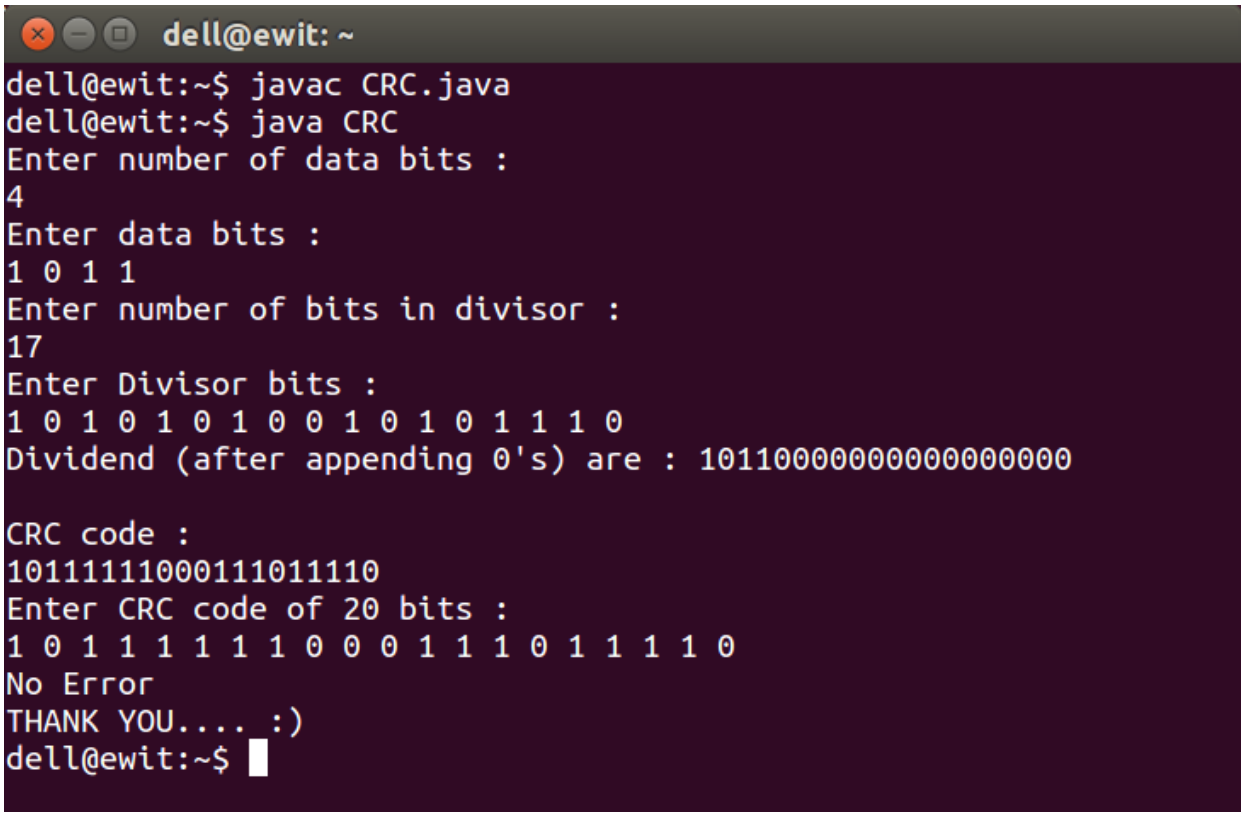
Steps:

- First, a string ofn-1 0sisappendedtothe data unit.
- Thenumberof0sisonelessthanthenumberofbitsinthedivisor which is n bits.
- Then the newly elongated data unit is divided by the divisor using a process called binary division (XOR).
- TheremainderisCRC.TheCRCreplacestheappended0sattheend of the data unit.
- The data unit arrives at the receiver first, followed by the CRC.
- The receiver treats whole string as the data unit and divides it by the same divisor that was used to find the CRC remainder.
- If the remainder is 0 then the data unit is error free. Otherwise it having some error and it must be discarded.

```
import java.io.*;
import java.util.*;
class CRC
{
	public static void main(String a[]) throws IOException
	{
		Scanner sc=new Scanner(System.in);
		int[] message;
		int[] gen;
		int[] app_message;
		int[] rem;
		int[] trans_message;
		int message_bits,gen_bits,total_bits;

		System.out.println("Enter no bits in mwssage:");
		message_bits=sc.nextInt();

		message=new int [message_bits];
		System.out.println("\nEnter message bits:");
		for(int i=0; i<message_bits;i++)
			message[i]= sc.nextInt();

		System.out.println("\nEnter number of bits in gen:");
		gen_bits= sc.nextInt();

		gen=new int[gen_bits];
		System.out.println("\nEnter gen bits:");
		for(int i=0;i<gen_bits;i++)
			gen[i]= sc.nextInt();

		total_bits=message_bits+gen_bits-1;

		app_message=new int[total_bits];
		rem=new int[total_bits];
		trans_message=new int[total_bits];

		for(int i=0;i<message.length;i++)
			app_message[i]=message[i];

		System.out.println("\nMessage bits are:");
		for(int i=0;i<message_bits;i++)
			System.out.print("\t"+message[i]);

		System.out.println("\nGenerators bits are:");
		for(int i=0;i<gen_bits;i++)
			System.out.print("\t"+gen[i]);
```

```
		System.out.println("\nAppended message is:");
		for(int i=0;i<app_message.length;i++)
			System.out.print("\t"+app_message[i]);

		for(int j=0;j<app_message.length;j++)
			rem[j]=app_message[j];

		rem=computecrc(app_message,gen,rem);

		for(int i=0;i<app_message.length;i++)
			trans_message[i]=(app_message[i]^rem[i]);

		System.out.println("\nTransmitted message from the transmitter is:");
		for(int i=0;i<trans_message.length;i++)
			System.out.print("\t"+trans_message[i]);

		System.out.println("\nEnter received message of"+total_bits+"bits at receiver end:");
		for(int i=0;i<trans_message.length;i++)
			trans_message[i]= sc.nextInt();

		System.out.println("\nReceived message is:");
		for(int i=0;i<trans_message.length;i++)
			System.out.print("\t"+trans_message[i]);

		for(int j=0;j<trans_message.length;j++)
			rem[j]=trans_message[j];

		rem=computecrc(trans_message,gen,rem);
		for(int i=0;i<rem.length;i++)
		{
			if(rem[i]!=0)
			{
				System.out.println("\nThere is error in the received message");
				break;
			}
			if(i==rem.length-1)
				System.out.println("\nThere is no erron in the received mesage!!");
		}
	}
	static int[] computecrc(int app_message[],int gen[], int rem[])
	{
		int current=0;
		while(true)
		{
			for(int i=0;i<gen.length;i++)
				rem[current+i]=(rem[current+i]^gen[i]);
			while(rem[current]==0 && current!=rem.length-1)
```

```
                    current++;
                if((rem.length-current)<gen.length)
                    break;
            }
            return rem;
        }
}
```



5. Develop a program to implement a sliding window protocol in the data link layer.

The sliding window protocol is a well-known technique that plays a significant role in ensuring reliable and orderly data exchange between a sender and a receiver. In this section, we will delve into the concept of the sliding window protocol and demonstrate how to implement it using Java.

Understanding the Sliding Window Protocol

The sliding window protocol is a communication protocol used to manage the flow control and reliability of data transmission over a network. It allows the sender to transmit a specified number of packets, known as the window size, without waiting for an acknowledgment from the receiver for each packet. This approach enhances efficiency by minimizing the communication overhead.

The protocol employs two primary components: the sender's sliding window and the receiver's sliding window. The sender's window keeps track of the packets that have been sent but not yet acknowledged, while the receiver's window tracks the expected sequence of packets to receive. As acknowledgments are received, both windows slide forward, allowing for the continuous flow of data.

Implementation of the Sliding Window Protocol

To implement the sliding window protocol in Java, we will create a simplified example of a sender and a receiver using sockets for communication. We will assume a reliable connection, so the focus will be on the sliding window mechanism.

Server Program

```
import   java.io.*;
import java.net.*;
import java.util.Random;
public class SlidingWindowServer  {
private static final int PORT = 12345;
private static final int WINDOW_SIZE = 4;
private static final double PACKET_LOSS_RATE = 0.1; // 10% packet loss
public static void main(String[] args) {
try (ServerSocket serverSocket = new ServerSocket(PORT)) {
System.out.println("Server is running and waiting for connections...");
try (Socket clientSocket = serverSocket.accept();
BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {
System.out.println("Client connected.");
String line;
int base = 0;
while ((line = in.readLine()) != null) {
if (simulatePacketLoss()) {
System.out.println("Packet " + line + " lost during transmission.");
} else {
System.out.println("Received packet: " + line);
// Simulate acknowledgment
if (base < WINDOW_SIZE) {
out.println("ACK: " + base);
base++;
}
}
}
```

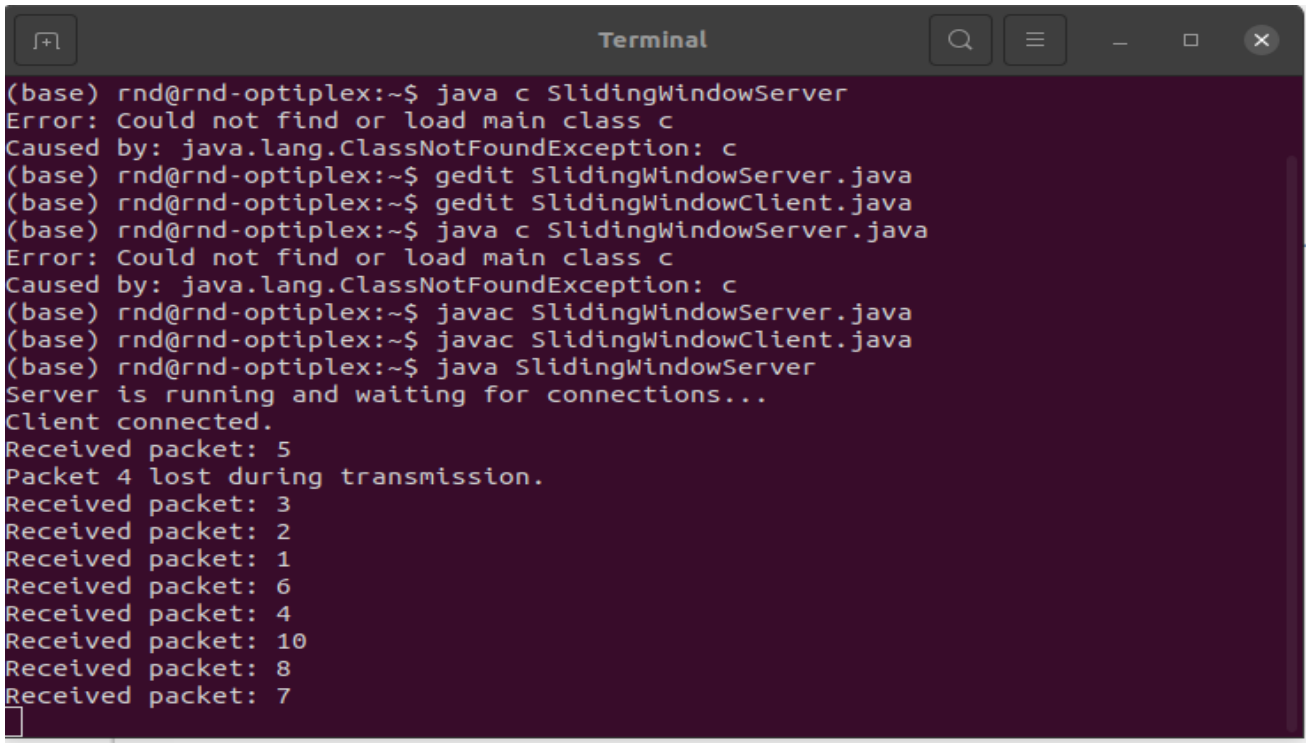
```
}
} catch (IOException e) {
e.printStackTrace();
}
}
private static boolean simulatePacketLoss() {
Random random = new Random();
return random.nextDouble() < PACKET_LOSS_RATE;
}
}
```

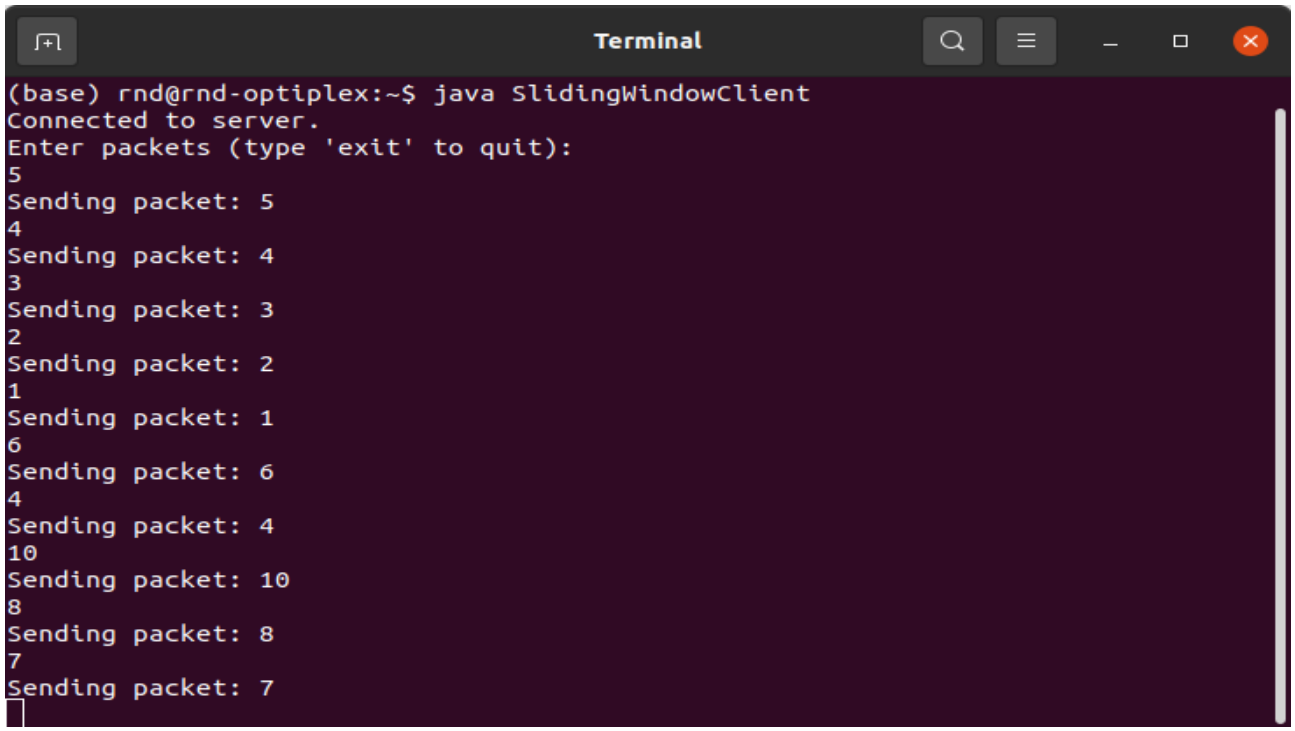
Client Program

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
import java.util.Random;
public class SlidingWindowClient {
private static final String SERVER_ADDRESS = "localhost";
private static final int PORT = 12345;
private static final int WINDOW_SIZE = 4;
private static final double PACKET_LOSS_RATE = 0.1; // 10% packet loss
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
try (Socket socket = new Socket(SERVER_ADDRESS, PORT);
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) {
System.out.println("Connected to server.");
System.out.println("Enter packets (type 'exit' to quit):");
int base = 0;
```

```
while (true) {
if (base >= WINDOW_SIZE) {
// Wait for acknowledgment
String ack = in.readLine();
if (ack != null) {
System.out.println("Received " + ack);
base++;
}
}
String packet = scanner.nextLine();
if (packet.equalsIgnoreCase("exit")) {
break;
}
if (simulatePacketLoss()) {
System.out.println("Packet " + packet + " lost during transmission.");
} else {
System.out.println("Sending packet: " + packet);
out.println(packet);
}
}
} catch (IOException e) {
e.printStackTrace();
}
}
private static boolean simulatePacketLoss() {
Random random = new Random();
return random.nextDouble() < PACKET_LOSS_RATE;
}
}
```

Snapshot:


Terminal
(base) rnd@rnd-optiplex:~$ java c SlidingWindowServer
Error: Could not find or load main class c
Caused by: java.lang.ClassNotFoundException: c
(base) rnd@rnd-optiplex:~$ gedit SlidingWindowServer.java
(base) rnd@rnd-optiplex:~$ gedit SlidingWindowClient.java
(base) rnd@rnd-optiplex:~$ java c SlidingWindowServer.java
Error: Could not find or load main class c
Caused by: java.lang.ClassNotFoundException: c
(base) rnd@rnd-optiplex:~$ javac SlidingWindowServer.java
(base) rnd@rnd-optiplex:~$ javac SlidingWindowClient.java
(base) rnd@rnd-optiplex:~$ java SlidingWindowServer
Server is running and waiting for connections...
Client connected.
Received packet: 5
Packet 4 lost during transmission.
Received packet: 3
Received packet: 2
Received packet: 1
Received packet: 6
Received packet: 4
Received packet: 10
Received packet: 8
Received packet: 7


Terminal
(base) rnd@rnd-optiplex:~$ java SlidingWindowClient
Connected to server.
Enter packets (type 'exit' to quit):
5
Sending packet: 5
4
Sending packet: 4
3
Sending packet: 3
2
Sending packet: 2
1
Sending packet: 1
6
Sending packet: 6
4
Sending packet: 4
10
Sending packet: 10
8
Sending packet: 8
7
Sending packet: 7

6. Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.

Steps:

Input : Graph and a source vertex *src* .

Output: Shortest distance to all vertices from *src*. If there is a negative weight cycle, then shortest distances are not calculated, negative weight cycle is reported.

1. This step initializes distances from source to all vertices as in finite and distance to source itself as 0. Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.

2. This step calculates shortest distances. Do following |V|-1 times where |V| is the number of vertices in given graph.

a) Do following for each edge u-v.

If dist[v]>dist[u]+weigh to f edge uv, then update
list[v]. dist[v] = dist[u] + weight of edge uv

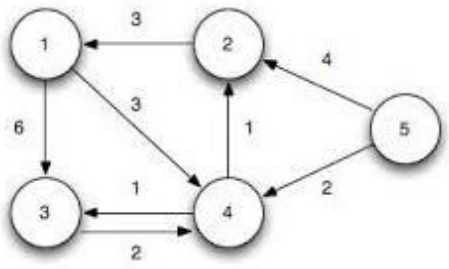
3. Thisstepreportsifthereisanegativeweightcycleingraph.Dofollowingfor each edge u-v.

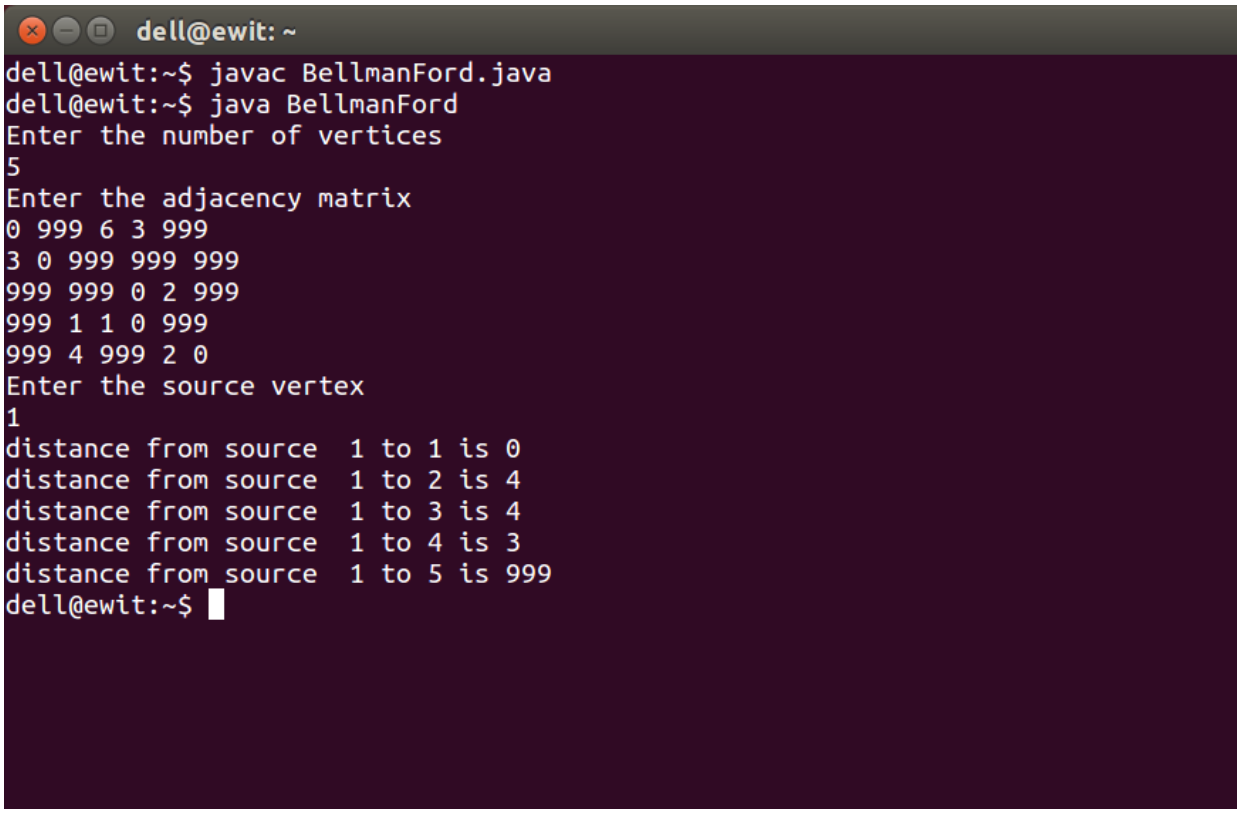
If dist[v] >dist[u] + weight of edge uv, then
"Graph contains negative weight cycle".

The idea of step 3 is, step 2 guarantees shortest distances if graph doesn't contain negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle.

```
import java.util.Scanner;
public class BellmanFord
{
	private int D[];
	private int NoV;
	public static final int MAX_VALUE = 999;

	public BellmanFord(int NoV)
	{
		this.NoV = NoV;
		D = new int[NoV + 1];
	}
	public void BellmanFordEvaluation(int source, int A[][])
	{
		for (int node = 1; node <= NoV; node++)
		{
			D[node] = MAX_VALUE;
		}
		D[source] = 0;
		for (int node = 1; node <= NoV - 1; node++)
		{
			for (int i = 1; i <= NoV; i++)
			{
				for (int j = 1; j <= NoV; j++)
				{
					if (A[i][j] != MAX_VALUE)
					{
						if (D[j] > D[i] + A[i][j])
							D[j] = D[i]+ A[i][j];
					}
				}
			}
		}
		for (int i = 1; i <= NoV; i++)
		{
			for (int j = 1; j <= NoV; j++)
			{
				if (A[i][j] != MAX_VALUE)
				{
					if (D[j] > D[i]+ A[i][j])
					{
						System.out.println("The Graph contains negative
egde cycle");
						return;
					}
```

```
                                }
                        }
                }
                for (int vertex = 1; vertex <= NoV; vertex++)
                {
                        System.out.println("distance of source  " + source + " to "+ vertex + " is "
+ D[vertex]);
                }
        }

        public static void main(String... arg)
        {
                int NoV = 0;
                int source;
                Scanner scanner = new Scanner(System.in);

                System.out.println("Enter the number of vertices");
                NoV = scanner.nextInt();
                int A[][] = new int[NoV + 1][NoV + 1];

                System.out.println("Enter the adjacency matrix");
                for (int i = 1; i <= NoV; i++)
                {
                        for (int j = 1; j <= NoV; j++)
                        {
                                A[i][j] = scanner.nextInt();
                        }
                }

                System.out.println("Enter the source vertex");
                source = scanner.nextInt();
                BellmanFord bellmanford = new BellmanFord(NoV);
                bellmanford.BellmanFordEvaluation(source, A);
                scanner.close();
        }
}
```




dell@ewit: ~
dell@ewit:~$ javac BellmanFord.java
dell@ewit:~$ java BellmanFord
Enter the number of vertices
5
Enter the adjacency matrix
0 999 6 3 999
3 0 999 999 999
999 999 0 2 999
999 1 1 0 999
999 4 999 2 0
Enter the source vertex
1
distance from source 1 to 1 is 0
distance from source 1 to 2 is 4
distance from source 1 to 3 is 4
distance from source 1 to 4 is 3
distance from source 1 to 5 is 999
dell@ewit:~$

7. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

Server.java

```
import java.net.*;
import java.io.*;
public class TCPS
{
	public static void main(String[] args) throws Exception
	{
		ServerSocket sersock=new ServerSocket(4000);
		System.out.println("Server ready for connection");
		Socket sock=sersock.accept();
		System.out.println("Connection Is successful and waiting for chatting");
		InputStream istream=sock.getInputStream();
		BufferedReader fileRead=new BufferedReader(new InputStreamReader(istream));
		String fname=fileRead.readLine();
		BufferedReader ContentRead=new BufferedReader(new FileReader(fname));
		OutputStream ostream=sock.getOutputStream();
		PrintWriter pwrite=new PrintWriter(ostream,true);
		String str;
		while((str=ContentRead.readLine())!=null)
		{
			pwrite.println(str);
		}
		sock.close();
		sersock.close();
		pwrite.close();
		fileRead.close();
		ContentRead.close();
	}
}
```

Client.java

```
import java.net.*;
import java.io.*;
public class TCPC
{
	public static void main(String[] args) throws Exception
	{
		Socket sock=new Socket("127.0.01",4000);
		System.out.println("Enter the filename");
		BufferedReader keyRead=new BufferedReader(new InputStreamReader(System.in));
		String fname=keyRead.readLine();
		OutputStream ostream=sock.getOutputStream();
		PrintWriter pwrite=new PrintWriter(ostream,true);
```

```
            pwrite.println(fname);
            InputStream istream=sock.getInputStream();
            BufferedReader socketRead=new BufferedReader(new InputStreamReader(istream));
            String str;
            while((str=socketRead.readLine())!=null)
            {
                  System.out.println(str);
            }
      pwrite.close();
      socketRead.close();
      keyRead.close();
      }
}
```

8. Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.

4.

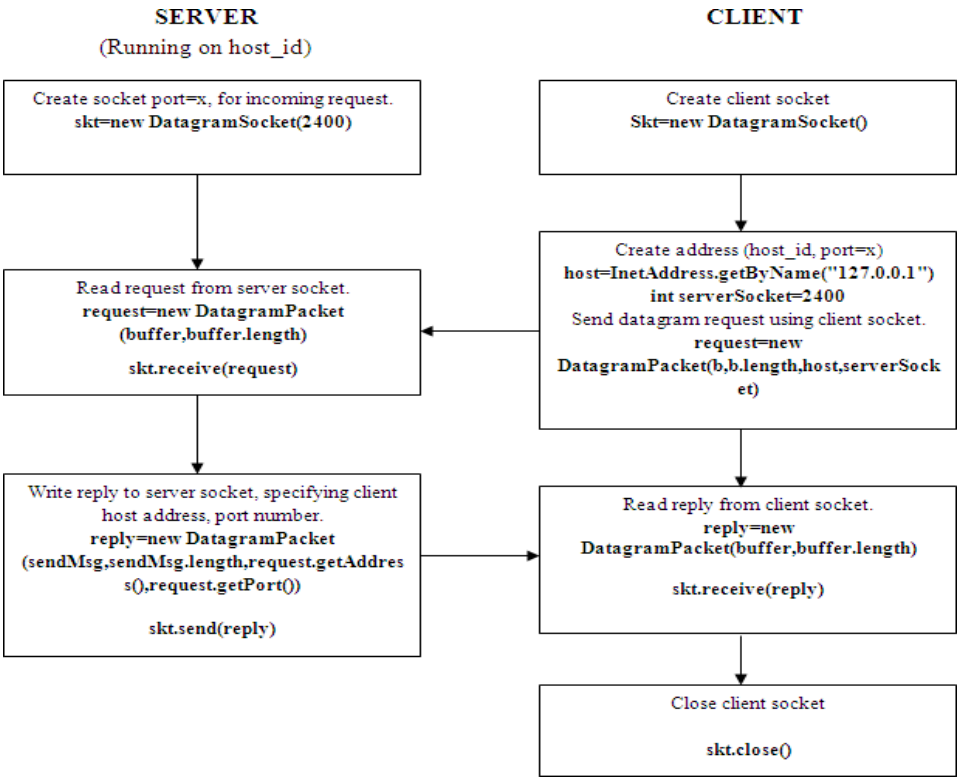


Fig 8.1:UDP client/server communication flow.

Methods and description

α. **DatagramSocket(intport) throws SocketEeption:** it creates a datagram socket and binds it with the given Port Number.

β. **DatagramPacket(byte[]buffer,intlength):** it creates a datagram packet. This constructor is used to receive the packets.

χ. **DatagramPacket(byte[] buffer, int length, InetAddress address, int port):**it creates a datagram packet. This constructor is used to send the packets.

```
//UDP Server Source code:
import java.io.*;
importjava.net.*;
importjava.util.*;
public class udps
{
publicstaticvoid main(String[]args)
{
DatagramSocketskt=null;
Scannersc=newScanner(System.in); try
{
skt=newDatagramSocket(2400); byte[] buffer=new
byte[1000]; while(true)
{
DatagramPacketrequest=newDatagramPacket(buffer,buffer.length);
skt.receive(request);
Stringmessage=sc.nextLine();
byte[]sendMsg=message.getByte
s(); DatagramPacket reply=new
DatagramPacket(sendMsg,sendMsg.length,request.getAddress(),request.getPort());
skt.send(reply);
}
}
catch(Exceptionex)
{
}
}
}
```

UDP Client Source code:

```
import java.io.*;
importjava.net.*;
publicclassudpc
{
publicstaticvoid main(String[]args)
{
DatagramSocketskt;
try
{
skt=new DatagramSocket();
Stringmsg="textmessage";
byte[]b=msg.getBytes();
InetAddresshost=InetAddress.getByName("127.0.0.1");
intserverSocket=2400;
DatagramPacketrequest=newDatagramPacket(b,b.length,host,serverSocket);
skt.send(request);
byte[]buffer=newbyte[1000];
```

```
DatagramPacketreply=newDatagramPacket(buffer,buffer.length);
skt.receive(reply);
System.out.println("clientrecieved:"+newString(reply.getData()));
skt.close();
}
catch(Exceptionex){}
}
}
```

9. Develop a program for a simple RSA algorithm to encrypt and decrypt the data.

RSA is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys (one is public key and another is private key). This is also called public key cryptography, because one of the m can be given to everyone. The other key must be kept private. It is based on the fact that finding the factors of an integer is hard (the factoring problem). RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman, who first publicly described it in 1978.

Following are the steps of RSA algorithm.

Key Generation

1. Generate two large prime numbers **p** and **q**, such that **p!=q**.
2. Let **n=p*q.**
3. Let **t=(p-1)*(q-1)**
4. Choose a small numbere ,co-prime tot, with **GCD(t,e)=1**and **1<e<t.**
5. Find d, such that **d*emodt=1**. Publish **e** and **n** as **public key**. Keep **d** and **t** as **secret key**.

Encryption

Cipher=(Message)^{e}modn

Decryption

Message=(Cipher)^{d}modn

```
importjava.util.Scanner;
public class RSA
{
        publicstaticintp,q,n,t,flag,msg,m,temp;
        public staticint e[]=new int[100];
        public staticint d[]=new int[100];
        public static int prime( intpr)
        {
                inti;

                Doublea=(Math.sqrt(p
                r)); m=a.intValue();

                for(i=2;i<=m;i++)
                {
                        if(pr%i==0)
                                return0;
                }

                return1;

        }
        publicstaticvoidce()
        {
                intk=0;

                for(inti=2;i<t;i++)
                {
                        if(t%i==0)
                                conti

                        nue;

                        flag=prime(i

                        );

                        if(flag==1&&i!=p&&i!=q)
                        {

                                e[k]=i;
                                flag=cd(e[k]);
                                if(flag>0)
                                {
                                        d[k]=fl
                                        ag;
                                        k++;
```

```
                    }
                    if(k==99)
                break;
                }
        }
}


publicstaticintcd( intx)
{
        intk=1;

        while(true)
        {
                k=k+t;

                if(k%x==0)
                        return(k/x);
        }

}

publicstaticvoid encrypt()
{

        intpt,ct,key=e[0],k;

        pt=msg;
        k=1;

        for(intj=0;j<key;j++)
        {
                k=k*pt;
                k=k%n;
        }

        ct=k;
        temp
        =ct;
        System.out.println("\nTHEENCRYPTEDMESSAGEIS:"+ct);

}

Publicstatic void decrypt()
{
        intpt,ct,key=d[0],k;
        ct=temp;
```

```
k=1;

                for(intj=0;j<key;j++)
                {
                        k=k
                        *ct;
                        k=k
                        %n;
                }

                pt=k;

               System.out.println("\nTHEDECRYPTEDMESSAGEIS:"+pt);
        }

        publicstaticvoidmain(Stringargs[])
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("ENTERFIRSTPRIMENUMBER"
                ); p=sc.nextInt();
                flag=prime(p);
                if(flag==0)
                {
                        System.out.println("WRONGINPU
                        T"); System.exit(1);
                }
                System.out.println("ENTERANOTHERPRIMENUMBER");
                q=sc.nextInt();
                flag=prime(q);
                if(flag==0||p==q)
                {
                        System.out.println("WRONGINPU
                        T"); System.exit(1);
                }

                System.out.println("ENTERMESSAG
                E"); msg=sc.nextInt();
                n=p*q;
                t=(p-1)*(q-1);

                ce();

                System.out.println("POSSIBLEVALUESOFeANDdARE");
                for (int i=0;i< m-1;i++)
                        System.out.printf("\n%d\t%d",e[i],d[i]);

                encrypt();
```

```
                    decrypt();
                }
        }
```

10. Develop a program for congestion control using a leaky bucket algorithm.

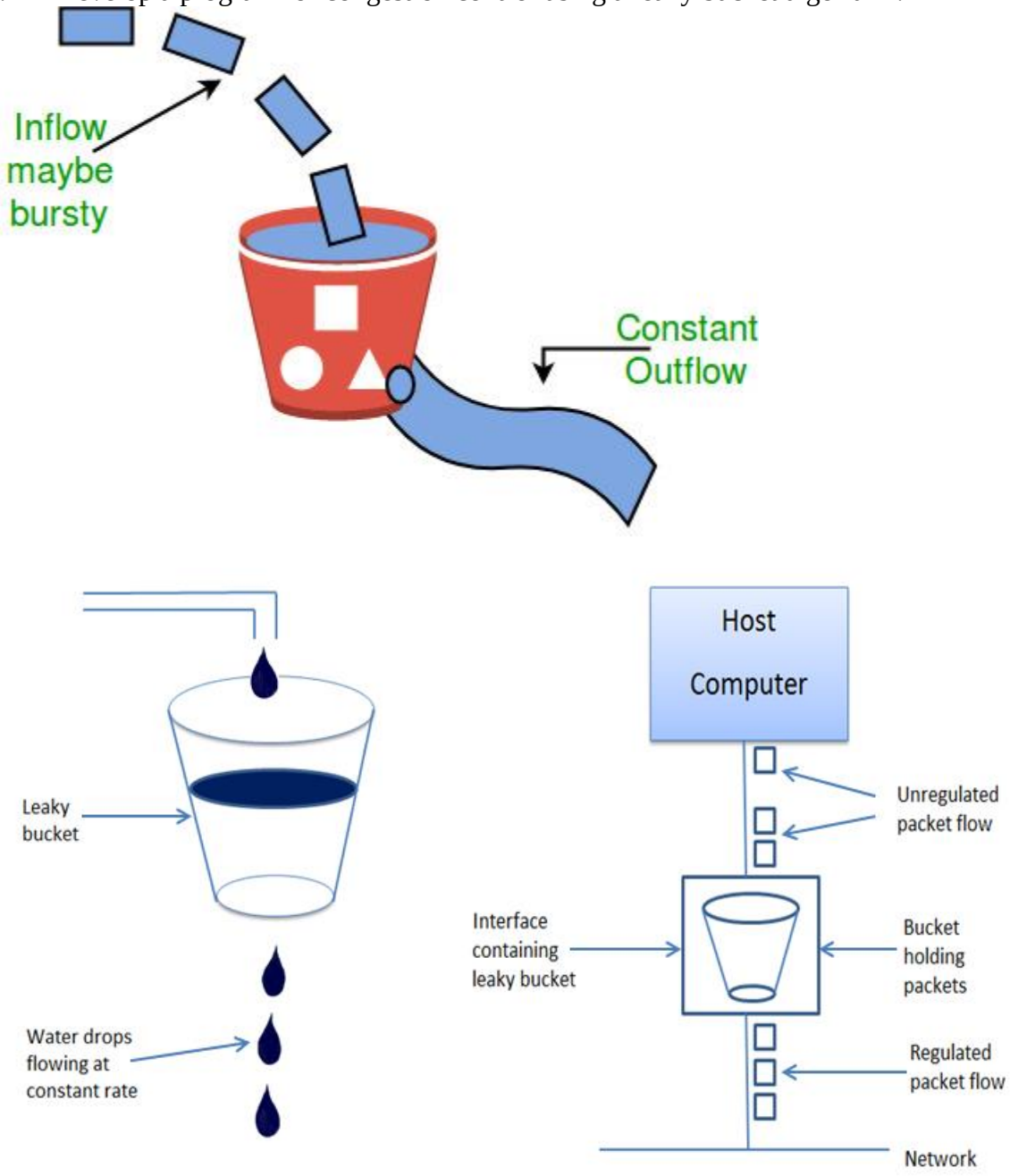


Fig: Leaky Bucket Algorithm

What is Leaky Bucket

- There is a bucket with a hole at the bottom.
- Water will be pouring into the bucket from the top.
- Water flows out at a constant rate irrespective of input of water into the bucket.

Lets apply the same concept to our network bucket.

- Tap → Host Computer
- Water flow → unregulated packet flow.
- Bucket → Interface

- Result:
- Packet Size beyond capacity → Dropped
- Packet Size within the capacity →Accept
-
- Example:
- Initially the bucket is empty : Remaining = 0
- Bucket Capacity = 4
- Rate = 3

i	A[i]	Accept by Bucket	Sent	Remaining
1	2	2	2	0
2	4	4	3	1
3	1	1	2	0
4	5	Dropped	0	0
5	3	3	3	0

Code:

```
import java.util.Scanner;
import java.lang.*;
public class LeakyBucket
{
    public static void main(String[] args)
     {
            int i;
            int a[]=new int[20];
            int buck_rem=0,buck_cap=4,rate=3,sent,recv;
            Scanner in = new Scanner(System.in);
            System.out.println("Enter the number of packets");
            int n = in.nextInt();
            System.out.println("Enter the packets");
            for(i=1;i<=n;i++)
                    a[i]= in.nextInt();
            System.out.println("Clock \t packet size \t accept \t sent \t remaining");
             for(i=1;i<=n;i++)
               {
                    if(a[i]!=0)
                    {
                            if(buck_rem+a[i]>buck_cap)
                                    recv=-1;
                            else
```

```
                        {
                                recv=a[i];
                                buck_rem+=a[i];
                        }
                }
                else
                        recv=0;

        if(buck_rem!=0)
        {
                if(buck_rem<rate)

                {
                        sent=buck_rem;
                        buck_rem=0;
                }
                else
                {
                        sent=rate;
                        buck_rem=buck_rem-rate;
                }
        }
        else
                sent=0;
        if(recv==-1)
                System.out.println(+i+ "\t\t" +a[i]+ "\t dropped \t" + sent +"\t"
        +buck_rem);
                else System.out.println(+i+ "\t\t" +a[i] +"\t\t" +recv +"\t" +sent + "\t"
        +buck_rem);
        }
    }
}
```