	Course Title: Network programming lab using JAVA and NS		
	CourseCode: 18CSL57	**No. of Credits: 0 : 0 :1 (L-T-P)**	**No. of lecture hours/week : 2**
	Exam Duration : 3 hours	**CIE+ SEE = 50+50=100**	

Course Objectives:	**Description**
	1. To understand and apply the basics of Java Programming. 2. To demonstrate some concepts of Networking using Java Programming. 3. To introduce network topologies using NS2 and check the performance of TCP and UDP protocols 4. To understand the creation of an Ethernet LAN by changing error rate and data rate to verify the throughput. 4. To understand and design wireless and wired network using NS2.

Unit No	**Syllabus Content**
	PART-A
1.	Write a Java program using synchronized threads to demonstrate producer-consumer concepts.
2.	Write a Java Swing program that consists of three tabs named Select Semester, Select Course and Select Electives. The "Select Semester" tab must contain four Buttons. The "Select Course" should contain a list of check boxes named with the courses such as Java, Compiler Design, and Machine Learning. "The Select Electives" tab should contain a drop down list of elective names of subjects. Hint: Swing application which uses, **i)** JTabbed Pane **ii)** Each tab should Jpanel which include any one component given below in each JPanel **iii)**CheckBox/List/RadioButton
3.	Design and implement a simple Client Server Application using RMI.
4.	Design and implement Client Server communication using TCP socket programming. (Client requests a file, Server responds to client with contents of that file which is then displayed on the screen by Client).
5.	Implement a JAVA Servlet Program to create a dynamic HTML web page. (user name and password should be accepted using HTML and displayed using a Servlet).
6.	Implement JDBC Application Program Using MySQL/ORACLE connectivity, develop a program to accept book information such as accession number, title, authors, edition and publisher from the stored table in the database. Perform the following: 1. Display of Table Contents.. 2..Insertion of Values to the Table.

	3.Update and Delete contents as and when required.
PART-B	
1	Simulate a 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.
2	Simulate an Ethernet LAN using n nodes (6-10), change error rate and data rate and compare throughput.
3	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.
4	Create a scenario and simulate to study the performance of Stop and Wait ARQ Protocol.
5	Simulate simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.

Course Outcomes	**Description**	**RBT Levels**
CO1	Design solutions using programming constructs in Java to create User interface.	**L4**
CO2	To Demonstrate the usage of Java networking concepts and creation of dynamic web pages.	**L5**
CO3	Apply and compare the performance of transport layer protocols.	**L4**
CO4	Analyze the working of LAN by inducing error model.	**L4**
CO5	Evaluate the parameters to be configured for wired and wireless communication.	**L5**

CO-PO Mapping	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**P06**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
CO1	**3**	**3**	**3**	**1**	**2**							
CO2	**3**	**3**	**3**	**1**	**2**							
CO3	**3**	**3**	**3**	**1**	**1**							
CO4	**3**	**3**	**3**	**1**	**2**							
CO5	**3**	**3**	**3**	**1**	**2**							

Strong -3 Medium -2 Weak -1

Instructions to Students:

Part-A: The programs formulated should be executed using Java Programming Language using eclipse IDE.

Part-B: The programs formulated should be executed using NS2 Simulation Software.	
COURSE COORDINATOR:	1.Dr.Mary Cherian 2.Dr.Smitha Shekar B 3.Prof Madhu B 4.Prof.Pushpaveni H P 5.Prof.Veena A

PART-A

JAVA

Introduction

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

Java is guaranteed to be **Write Once, Run Anywhere.**

Java is:

- **Object Oriented:** In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- **Platform independent:** Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by virtual Machine (JVM) on whichever platform it is being run.
- **Simple:** Java is designed to be easy to learn. If you understand the basic concept of OOP Java would be easy to master.
- **Secure:** With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architectural-neutral:** Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system.
- **Portable:** Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset.
- **Robust:** Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- **Multithreaded:** With Java's multithreaded feature it is possible to write programs that can do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.
- **Interpreted:** Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light weight process.

- **High Performance:** With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed:** Java is designed for the distributed environment of the internet.
- **Dynamic:** Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

Creating a simple Java Program

Hello World example:

```
class HelloWorld {
public static void main (String args[]) {
    System.out.println("Hello World! ");
    }
}
```

This program has two main parts:

- All the program is enclosed in a class definition—here, a class calledHello World.
- The body of the program (here, just the one line) is contained in a method (function)called main(). In Java applications, as in a C or C++ program, main() is the first method(function) that is run when the program is executed.

Compiling the above program :

- In Sun's JDK, the Java compiler is called javac.

```
javac HelloWorld.java
```

- When the program compiles without errors, a file called HelloWorld.class is created,in the same directory as the source file. This is the Java bytecode file.
- Then run that bytecode file using the Java interpreter. In the JDK, the Java interpreteris called simply java.

```
java HelloWorld
```

If the program was typed and compiled correctly, the output will be:

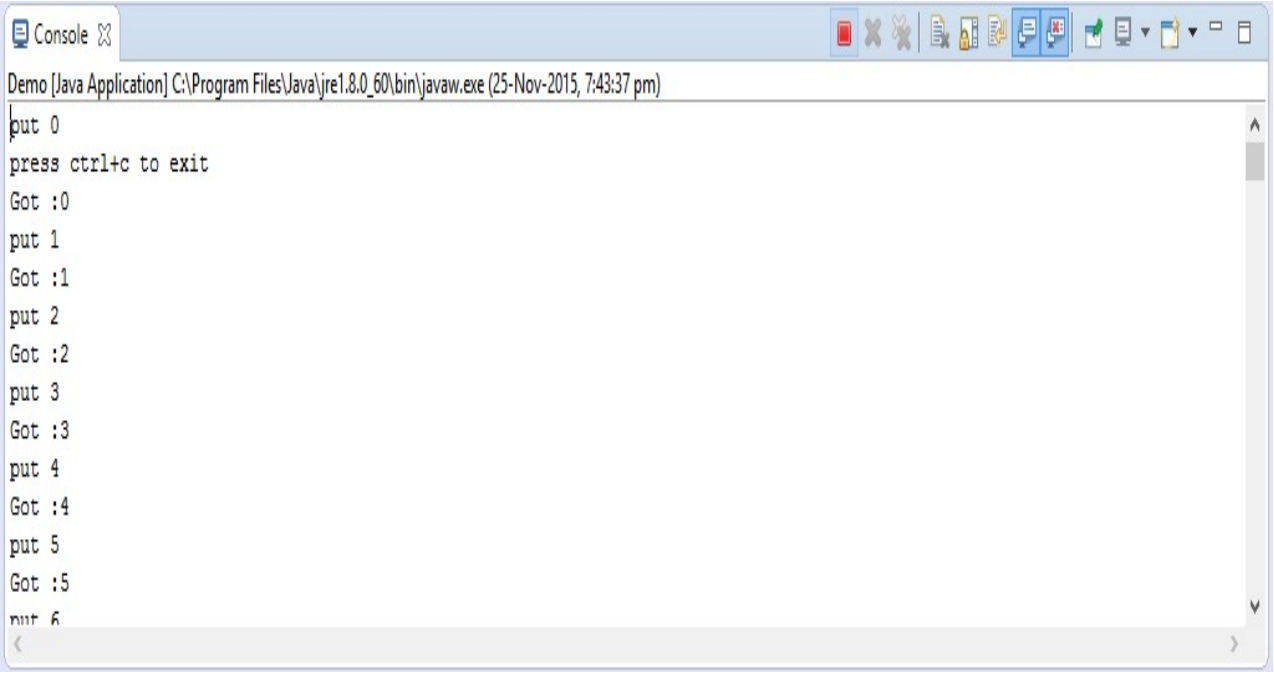
```
"Hello World!"
```

1) Write a Java program using synchronized threads which demonstrate producer-consumer concepts.

```
class Q {
   int n;
   boolean valueset = false;

   synchronized int get() {
      while (!valueset)
         try {
            wait();
         } catch (InterruptedException e) {
            System.out.println("Thread Interrupted");
         }
      System.out.println("Got :" + n);
      valueset = false;
      notify();
      return n;
   }

   synchronized void put(int n) {
      while (valueset)
         try {
            wait();
         } catch (InterruptedException e) {
            System.out.println("Thread interrupted");
         }
      this.n = n;
      valueset = true;
      System.out.println("put " + n);
      notify();
   }
}

class Producer implements Runnable {
   Q q;

   Producer(Q q) {
      this.q = q;
      new Thread(this, "Producer").start();
   }

   public void run() {
```

```
        int i = 0;
        while (true) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;

    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }

    public void run() {
        int i = 0;
        while (true) {
            q.get();
        }
    }
}

class Demo {
    public static void main(String args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("press ctrl+c to exit");
    }
}
```

Output


Console
Demo [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (25-Nov-2015, 7:43:37 pm)
put 0
press ctrl+c to exit
Got :0
put 1
Got :1
put 2
Got :2
put 3
Got :3
put 4
Got :4
put 5
Got :5

2) Write a Java Swing program that consists of three tabs named Select Semester, Select Course and Select Electives. The "Select Semester" tab must contain four Buttons. The "Select Course" should contain a list of check boxes named with the courses such as Java, Compiler Design, and Machine Learning. "The Select Electives" tab should contain a drop down list of elective names of subjects.
Hint: Swing application which uses,

i) JTabbed Pane

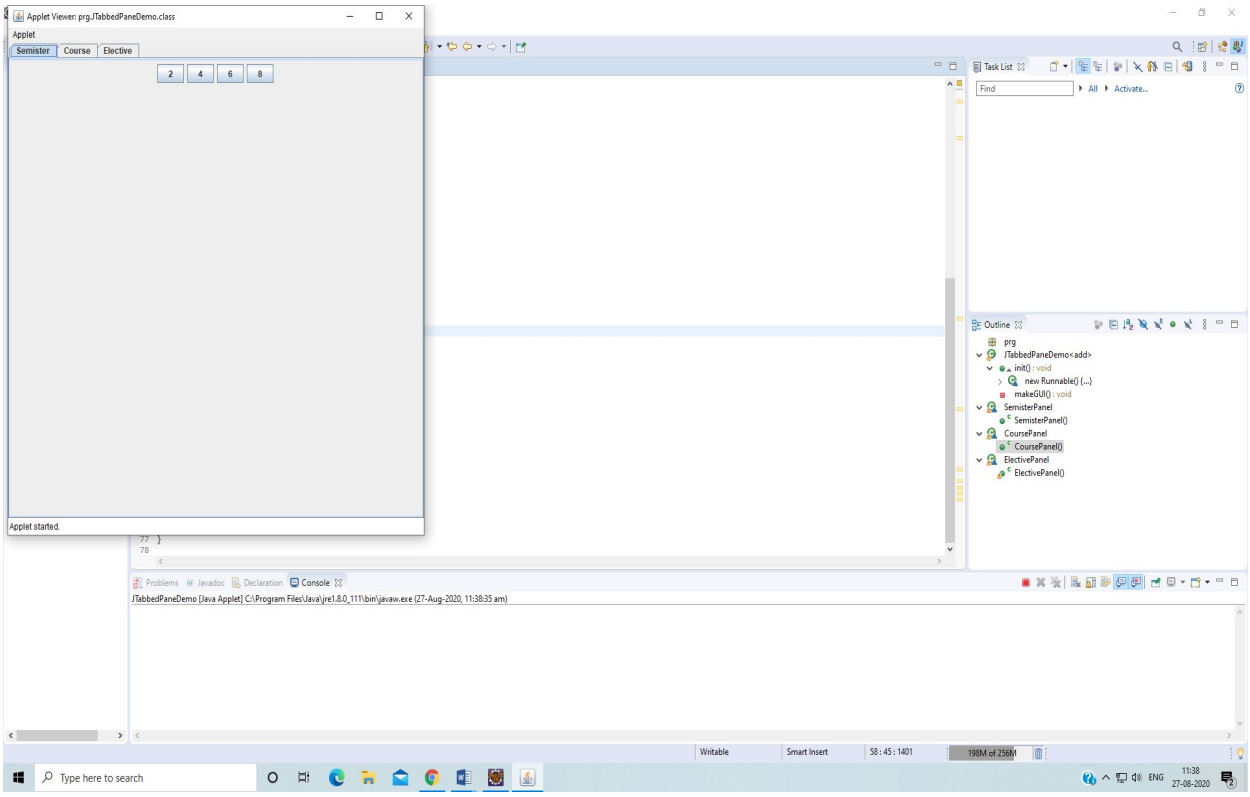
ii) Each tab should Jpanel which include any one component given below in each JPanel

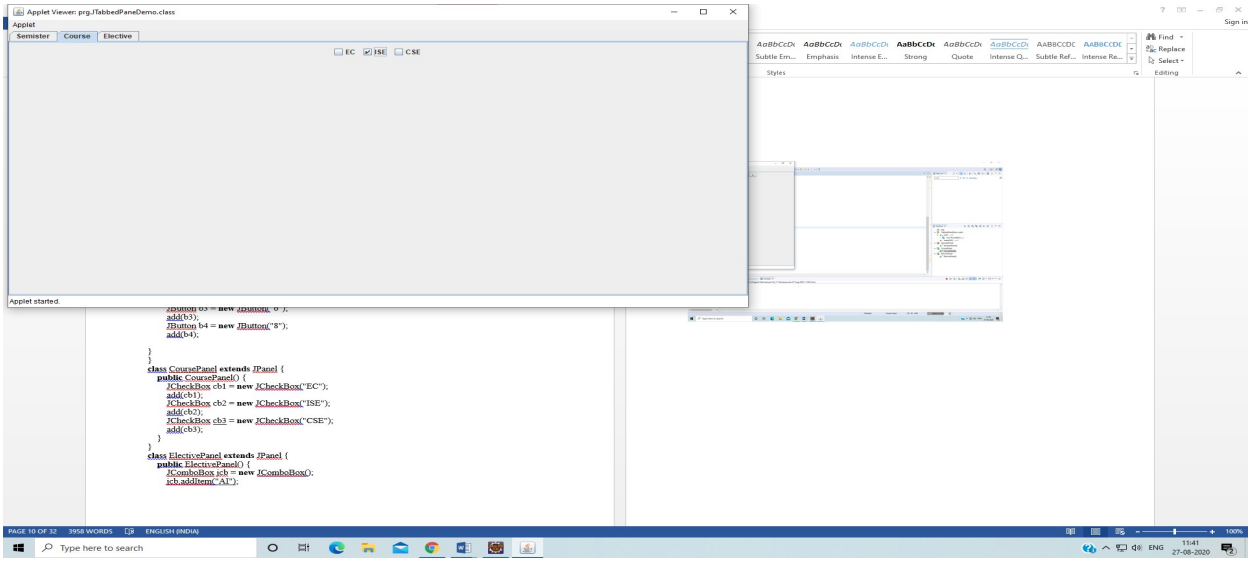
iii)CheckBox/List/RadioButton

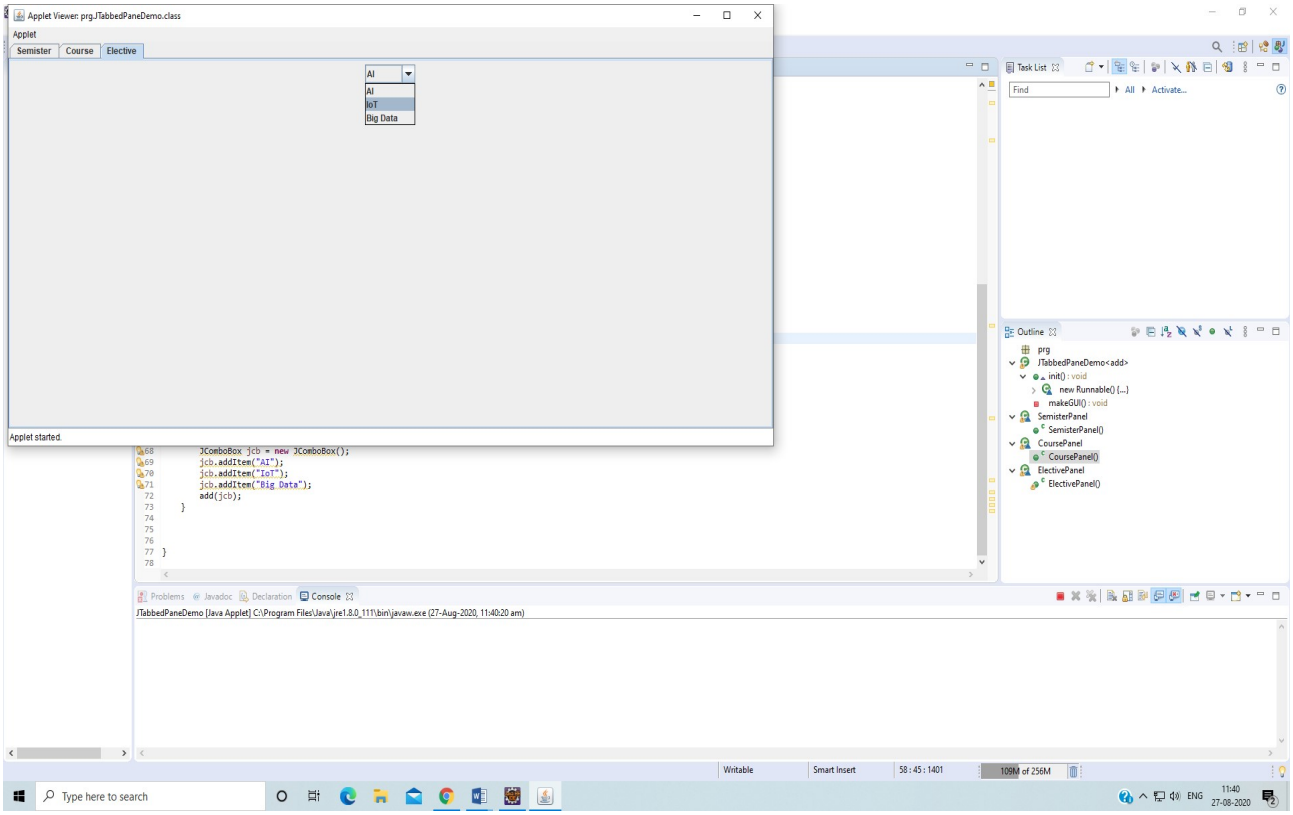
```
import java.awt.Color;
import java.awt.Graphics;

import javax.swing.*;

/*
<applet code="JTabbedPaneDemo" width=400 height=100>
</applet>
*/
public class JTabbedPaneDemo extends JApplet {
   public void init() {
      try {
         SwingUtilities.invokeAndWait(
               new Runnable() {
                  public void run() {
                     makeGUI();
                  }
               }
         );
```

```
        } catch (Exception exc) {
            System.out.println("Can't create because of " +exc);
        }
    }

    private void makeGUI() {

        JTabbedPane jtp = new JTabbedPane();
        jtp.addTab("Semister", new SemisterPanel());
        jtp.addTab("Course", new CoursePanel());
        jtp.addTab("Elective", new ElectivePanel());
        add(jtp);

    }
}

// Make the panels that will be added to the tabbed pane.
class SemisterPanel extends JPanel {
    public SemisterPanel() {
        JButton b1 = new JButton("2");
        add(b1);
        JButton b2 = new JButton("4");
        add(b2);
        JButton b3 = new JButton("6");
        add(b3);
        JButton b4 = new JButton("8");
        add(b4);

}
}
```

```
class CoursePanel extends JPanel {
    public CoursePanel() {
        JCheckBox cb1 = new JCheckBox("EC");
        add(cb1);
        JCheckBox cb2 = new JCheckBox("ISE");
        add(cb2);
        JCheckBox cb3 = new JCheckBox("CSE");
        add(cb3);
    }
}
class ElectivePanel extends JPanel {
    public ElectivePanel() {
        JComboBox jcb = new JComboBox();
        jcb.addItem("AI");
        jcb.addItem("IoT");
        jcb.addItem("Big Data");
        add(jcb);
    }



}
```


Applet Viewer: prg.JTabbedPaneDemo.class
Applet
Semister
Course
Elective
2
4
6
8
Applet started.
Task List
Find
All
Activate...
Outline
prg
JTabbedPaneDemo<add>
init() : void
new Runnable() {...}
makeGUI() : void
SemisterPanel
SemisterPanel()
CoursePanel
CoursePanel()
ElectivePanel
ElectivePanel()
Problems
Javadoc
Declaration
Console
JTabbedPaneDemo [Java Applet] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (27-Aug-2020, 11:38:35 am)
Writable
Smart Insert
58 : 45 : 1401
198M of 256M
Type here to search
ENG
11:38
27-08-2020


Applet Viewer: prg.JTabbedPaneDemo.class
Applet
Semister
Course
Elective
EC
ISE
CSE
Applet started.
add(b3);
JButton b4 = new JButton("8");
add(b4);
}
}
class CoursePanel extends JPanel {
public CoursePanel() {
JCheckBox cb1 = new JCheckBox("EC");
add(cb1);
JCheckBox cb2 = new JCheckBox("ISE");
add(cb2);
JCheckBox cb3 = new JCheckBox("CSE");
add(cb3);
}
}
class ElectivePanel extends JPanel {
public ElectivePanel() {
JComboBox jcb = new JComboBox();
jcb.addItem("A1");
PAGE 10 OF 32
3958 WORDS
ENGLISH (INDIA)
Type here to search
ENG
11:41
27-08-2020


Applet Viewer: prg.JTabbedPaneDemo.class
Applet
Semister
Course
Elective
AI
IoT
Big Data
Applet started.
JComboBox jcb = new JComboBox();
jcb.addItem("AI");
jcb.addItem("IoT");
jcb.addItem("Big Data");
add(jcb);
}
}
Task List
Find
All
Activate...
Outline
prg
JTabbedPaneDemo<add>
init() : void
new Runnable() {...}
makeGUI() : void
SemisterPanel
SemisterPanel()
CoursePanel
CoursePanel()
ElectivePanel
ElectivePanel()
Problems
Javadoc
Declaration
Console
JTabbedPaneDemo [Java Applet] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (27-Aug-2020, 11:40:20 am)
Writable
Smart Insert
58 : 45 : 1401
109M of 256M
Type here to search
ENG
11:40
27-08-2020

RMI (Remote Method Invocation)

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects stub and skeleton.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.
A **remote object** is an object whose method can be invoked from another JVM.

stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

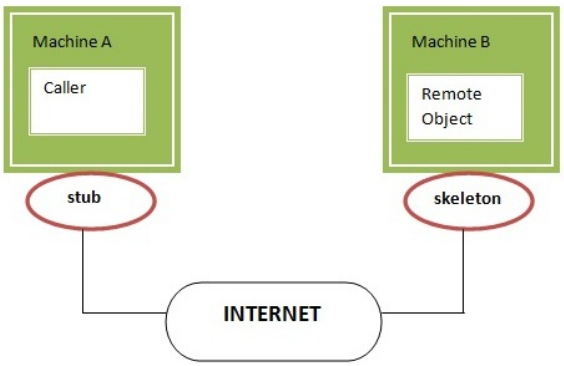
1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads the return value or exception, and
5. It finally, returns the value to the caller.

skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and

3. It writes and transmits (marshals) the result to the caller.



Steps to write the RMI program

1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

3) Design and implement a simple Client Server Application using RMI.

AddServerIntf.java

```
import java.rmi.*;
public interface AddServerIntf extends Remote {
   int add(int x, int y) throws RemoteException;
}
```

AddServerImpl.java

```
import java.rmi.*;
import java.rmi.server.*;
public class AddServerImpl extends UnicastRemoteObject implements AddServerIntf{
   public AddServerImpl() throws RemoteException {}
   public int add(int x, int y) throws RemoteException {
      return x+y;
   }
}
```

AddServer.java

```
import java.rmi.*;
public class AddServer {
   public static void main(String[] args) {
      try{
         AddServerImpl server = new AddServerImpl();
         Naming.rebind("registerme",server);
         System.out.println("Server is running...");
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

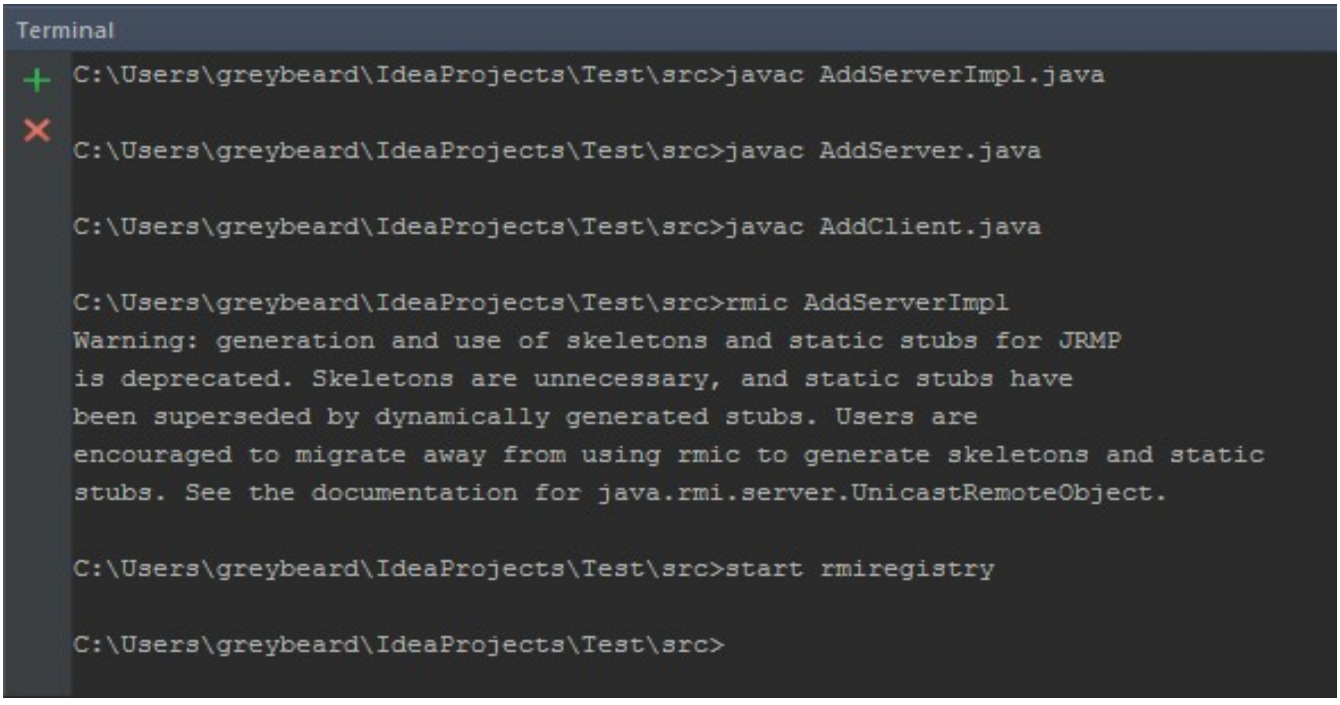
AddClient.java

```
import java.rmi.*;
public class AddClient {
   public static void main(String[] args) {
      try{
         AddServerIntf client = (AddServerIntf)Naming.lookup("registerme");
         System.out.println("First number is :" + args[0]);
         int x = Integer.parseInt(args[0]);
         System.out.println("Second number is :" + args[1]);
         int y = Integer.parseInt(args[1]);
         System.out.println("Sum =" + client.add(x,y));
      } catch (Exception e){
         System.out.println(e);
      }
   }
}
```

Output:

Open a terminal

Navigate to the src folder of your project



In another terminal (while previous one is still running)

Navigate to the src folder of your project

```
C:\Users\greybeard\IdeaProjects\Test\src>java AddServer
Server is running...
```

In third terminal (while previous both are still open)

Navigate to the src folder of your project

```
C:\Users\greybeard\IdeaProjects\Test\src>java AddClient 2 3
First number is :2
Second number is :3
Sum =5

C:\Users\greybeard\IdeaProjects\Test\src>
```

SOCKET PROGRAMMING

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

The **java.net.Socket** class represents a socket, and the **java.net.ServerSocket** class provides a mechanism for the server program to listen for clients and establish connections with them.

The following steps occur when establishing a TCP connection between two computers using sockets:

- The server instantiates a **ServerSocket** object, denoting which port number communication is to occur on.
- The server invokes the **accept()** method of the **ServerSocket** class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- On the server side, the **accept()** method returns a reference to a new socket on the server that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an **OutputStream** and an **InputStream**. The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.

TCP is a two-way communication protocol, so data can be sent across both streams at the same time. There are following useful classes providing complete set of methods to implement sockets.

ServerSocket Class Methods:

The **java.net.ServerSocket** class is used by server applications to obtain a port and listen for client requests

One of the four ServerSocket constructors are shown below:

public ServerSocket(int port) throws IOException Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.

If the ServerSocket constructor does not throw an exception, it means that your application has successfully bound to the specified port and is ready for client requests.

Here are some of the common methods of the ServerSocket class:

Sl.No	Methods with Description
1	**public int getLocalPort()** Returns the port that the server socket is listening on. This method is useful if you passed in 0 as the port number in a constructor and let the server find a port for you.
2	**public Socket accept() throws IOException** Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the setSoTimeout() method. Otherwise, this method blocks indefinitely
3	**public void setSoTimeout(int timeout)** Sets the time-out value for how long the server socket waits for a client during the accept().
4	**public void bind(SocketAddress host, int backlog)** Binds the socket to the specified server and port in the SocketAddress object. Use this method if you instantiated the ServerSocket using the no-argument constructor.

When the ServerSocket invokes accept(), the method does not return until a client connects. After a client does connect, the ServerSocket creates a new Socket on an unspecified port and returns a reference to this new Socket. A TCP connection now exists between the client and server, and communication can begin.

Socket Class Methods:

The **java.net.Socket** class represents the socket that both the client and server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the **accept()** method.

The Socket class has five constructors that a client uses to connect to a server. One of them is shown below:

public Socket(String host, int port) throws UnknownHostException, IOException.

This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.

When the Socket constructor returns, it does not simply instantiate a Socket object but it actually attempts to connect to the specified server and port.

Some methods of interest in the Socket class are listed here. Notice that both the client and server have a Socket object, so these methods can be invoked by both the client and server.

Sl.No.	Methods with Description
1	**public int getPort()** Returns the port the socket is bound to on the remote machine.
2	**public SocketAddress getRemoteSocketAddress()** Returns the address of the remote socket.
3	**public InputStream getInputStream() throws IOException** Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
4	**public OutputStream getOutputStream() throws IOException** Returns the output stream of the socket. The output stream is connected to the input stream of the

	remote socket
5	**public void close() throws IOException** Closes the socket, which makes this Socket object no longer capable of connecting again to any server

4) Design and implement Client Server communication using socket programming (Client requests a file, Server responds to client with contents of that file which is then display on the screen by Client).

Client.java

```
import java.net.*;
import java.io.*;
public class Client {
   public static void main(String[] args) {
      Socket client = null;
      BufferedReader br = null;
      try {
         System.out.println(args[0] + " " + args[1]);
         client = new Socket(args[0],Integer.parseInt(args[1]));
      } catch (Exception e){}
      DataInputStream input = null;
      PrintStream output = null;
      try {
         input = new DataInputStream(client.getInputStream());
         output = new PrintStream(client.getOutputStream());
         br = new BufferedReader(new InputStreamReader(System.in));
         String str = input.readLine();  //get the prompt from the server
         System.out.println(str);
         String filename = br.readLine();
         if (filename!=null){
            output.println(filename);
         }
         String data;
         while ((data=input.readLine())!=null) {
            System.out.println(data);
         }
         client.close();
      } catch (Exception e){
         System.out.println(e);
      }
   }
}
```

Server.java

```
import java.net.*;
import java.io.*;


public class Server {
   public static void main(String[] args) {
      ServerSocket server = null;
      try {
         server = new ServerSocket(Integer.parseInt(args[0]));
      } catch (Exception e) {
      }
      while (true) {
         Socket client = null;
         PrintStream output = null;
         DataInputStream input = null;
         try {
            client = server.accept();
         } catch (Exception e) {
            System.out.println(e);
         }
         try {
            output = new PrintStream(client.getOutputStream());
            input = new DataInputStream(client.getInputStream());
         } catch (Exception e) {
            System.out.println(e);
         }
         //Send the command prompt to client
         output.println("Enter the filename :>");
         try {
            //get the filename from client
            String filename = input.readLine();
            System.out.println("Client requested file :" + filename);
            try {
               File f = new File(filename);
               BufferedReader br = new BufferedReader(new FileReader(f));
               String data;
               while ((data = br.readLine()) != null) {
                  output.println(data);
               }
            } catch (FileNotFoundException e) {
               output.println("File not found");
            }
            client.close();
         } catch (Exception e) {
            System.out.println(e);
         }
      }
   }
}
```

Output

Create a file called testfile.txt in the folder where Client.java and Server.java is located. Add some content.

Open two terminals

Navigate to the src folder of your project

```
C:\Users\greybeard\IdeaProjects\Test\src>javac Server.java
Note: Server.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\greybeard\IdeaProjects\Test\src>java Server 4000
Client requested file :testfile.txt
```

```
C:\Users\greybeard\IdeaProjects\Test\src>javac Client.java
Note: Client.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\greybeard\IdeaProjects\Test\src>java Client localhost 4000
localhost 4000
Enter the filename :>
testfile.txt
Hello
How are you?

C:\Users\greybeard\IdeaProjects\Test\src>
```

SERVLET PROGRAMMING

Servlet technology is used to create web application (resides at server side and generates dynamic web page).

They are modules of Java code that run in a server application.

The advantages of using Servlets over traditional CGI programs are:

1. **Better performance:** because it creates a thread for each request not process.
2. **Portability:** because it uses java language.
3. **Robust:** Servlets are managed by JVM so no need to worry about momory leak, garbage collection etc.
4. **Secure:** because it uses java language.

Life cycle of a servlet

The life cycle of a servlet is controlled by the container in which the servlet has been deployed.

When a request is mapped to a servlet, the container performs the following steps:

1. If an instance of the servlet does not exist, the web container:
    a. Loads the servlet class
    b. Creates an instance of the servlet class
    c. Initializes the servlet instance by calling the init method. Initialization is covered inInitializing a Servlet
2. Invokes the service method, passing a request and response object.
3. If the container needs to remove the servlet, it finalizes the servlet by calling the servlet'sdestroy method.

Servlet API

- The **javax.servlet** and **javax.servlet.http** packages represent interfaces and classes for servlet api.
- The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only

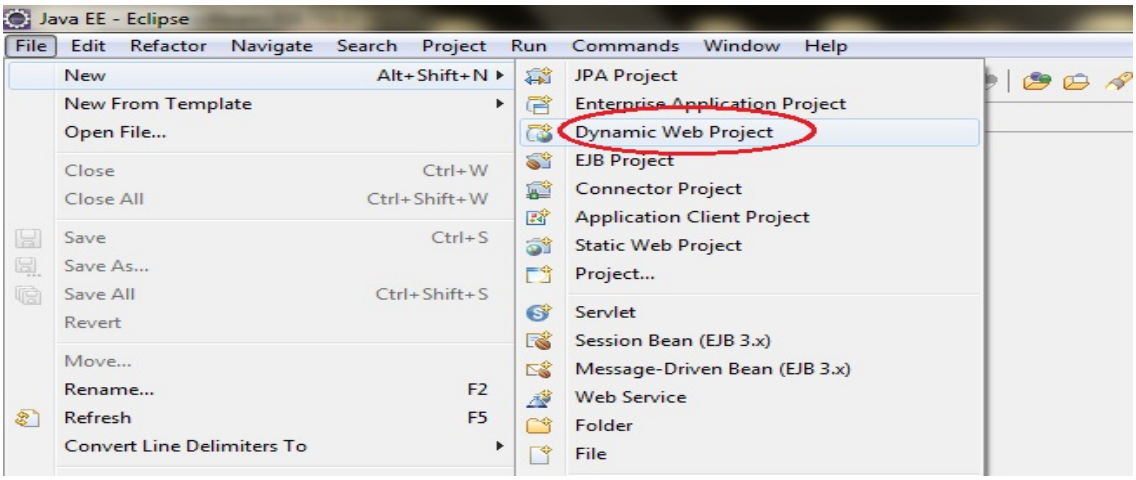
javax.servlet package

- The javax.servlet package contains a number of classes and interfaces that describeand define the contracts between a servlet class and the runtime environment provided for aninstance of such a class by a conforming servlet container.
- The Servlet interface is the central abstraction of the servlet API.
- All servlets implement this interface either directly, or more commonly, by extending aclass that implements the interface.
- The two classes in the servlet API that implement the Servlet interface are GeneriISErvletand HttpServlet .
- For most purposes, developers will extend HttpServlet to implement their servlets whileimplementing web applications employing the HTTP protocol.
- The basic Servlet interface defines a service method for handling client requests. Thismethod is called for each request that the servlet container routes to an instance of a servlet.

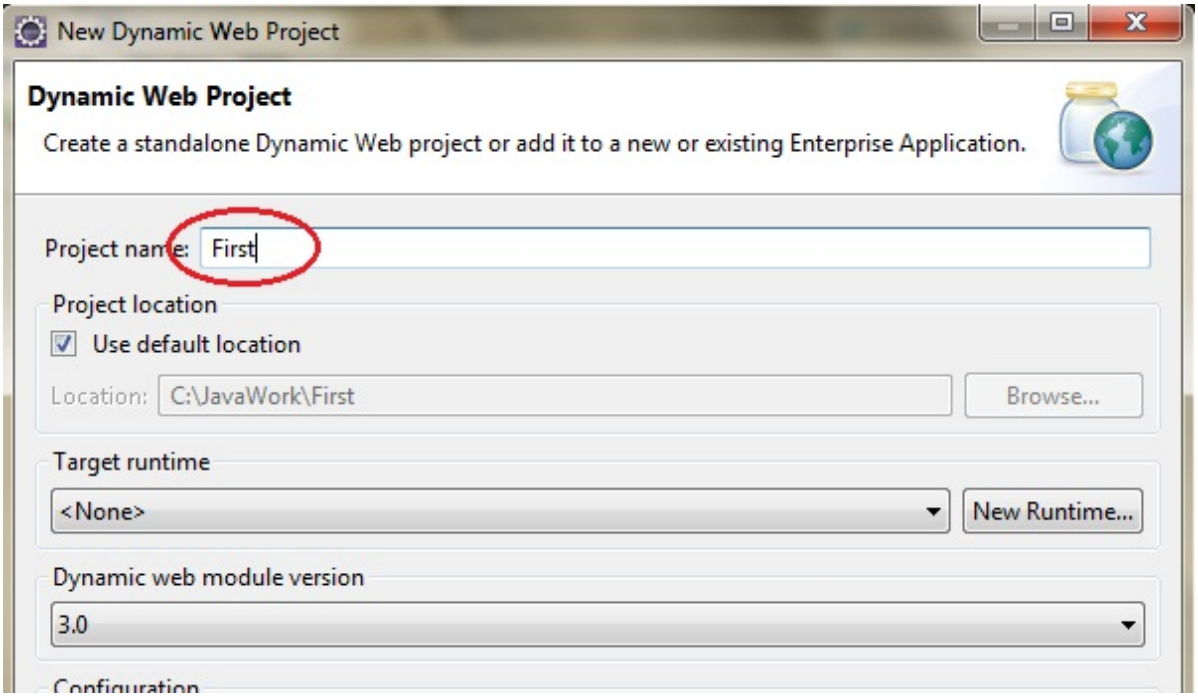
Running Servlet Programs in Eclipse EE

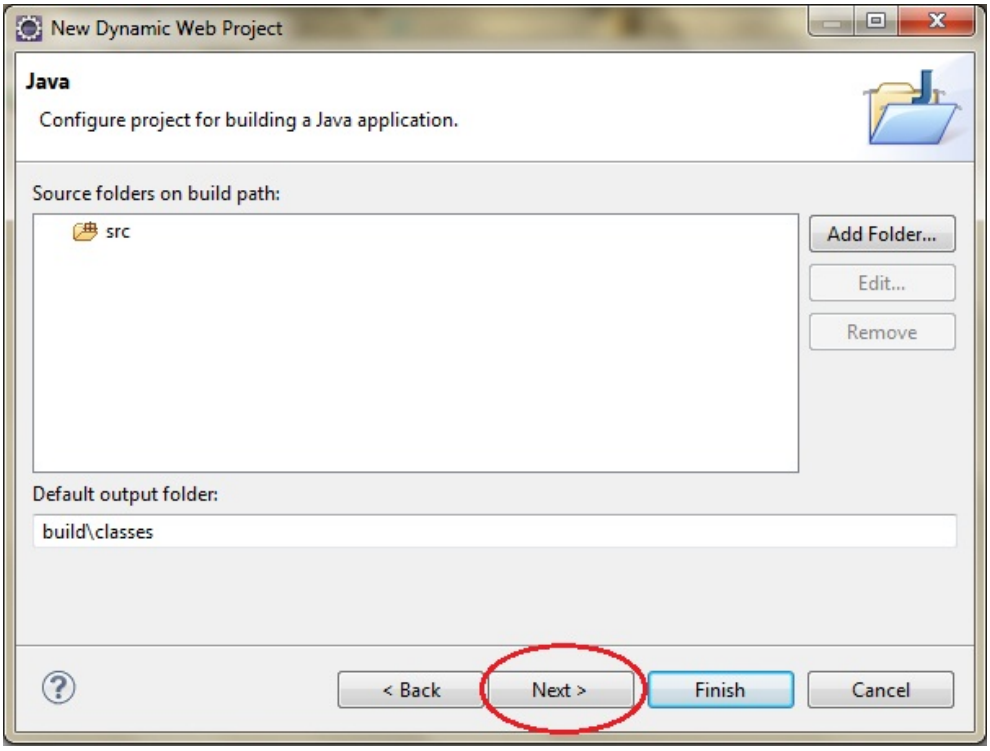
To create a Servlet application in Eclipse IDE you will need to follow the following steps:

Step 1. Goto **File ->New ->Dynamic Web Project**

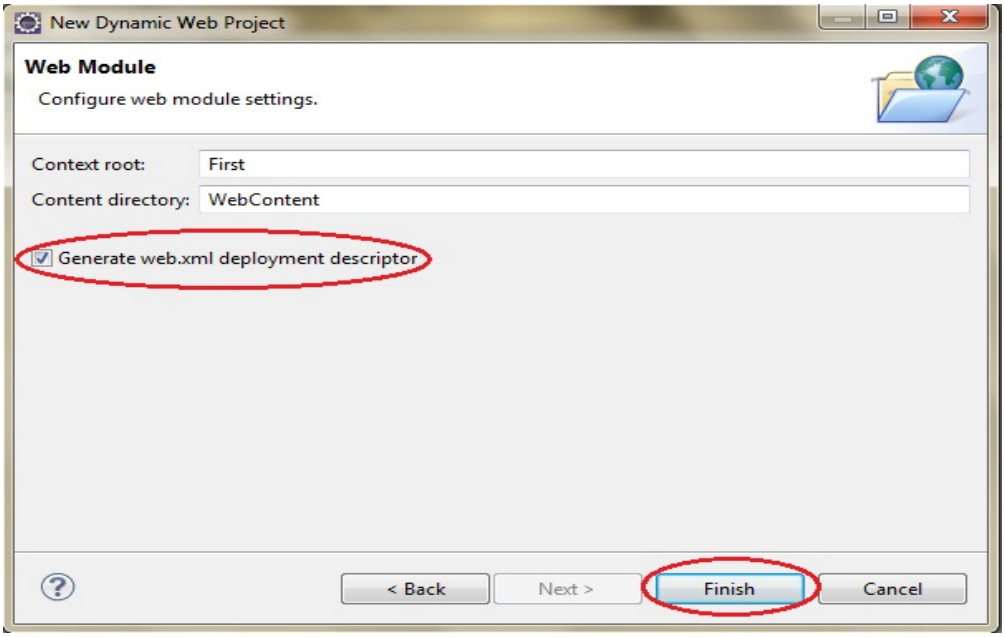


Step 2. **Give a Name to your Project and click Next**

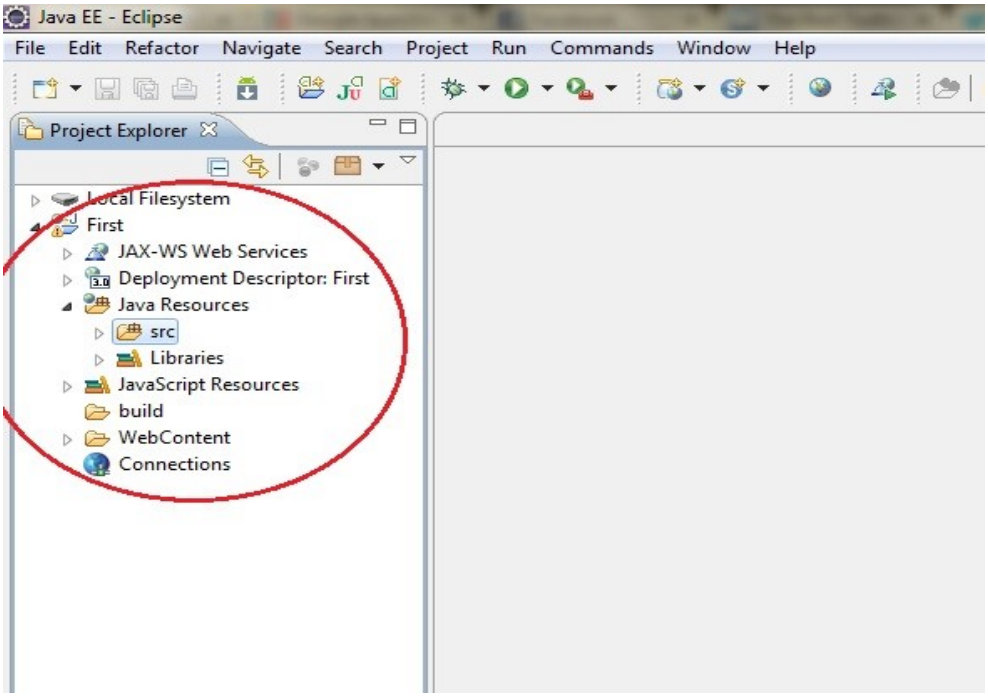




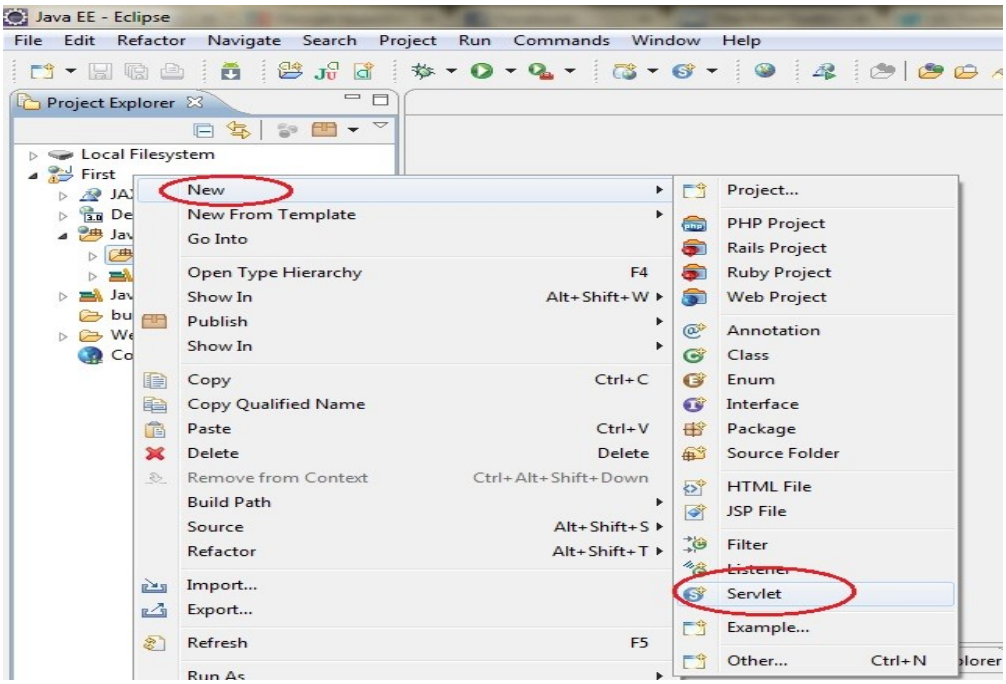
Step 3. **Check Generate web.xml Deployment Descriptor and click Finish**



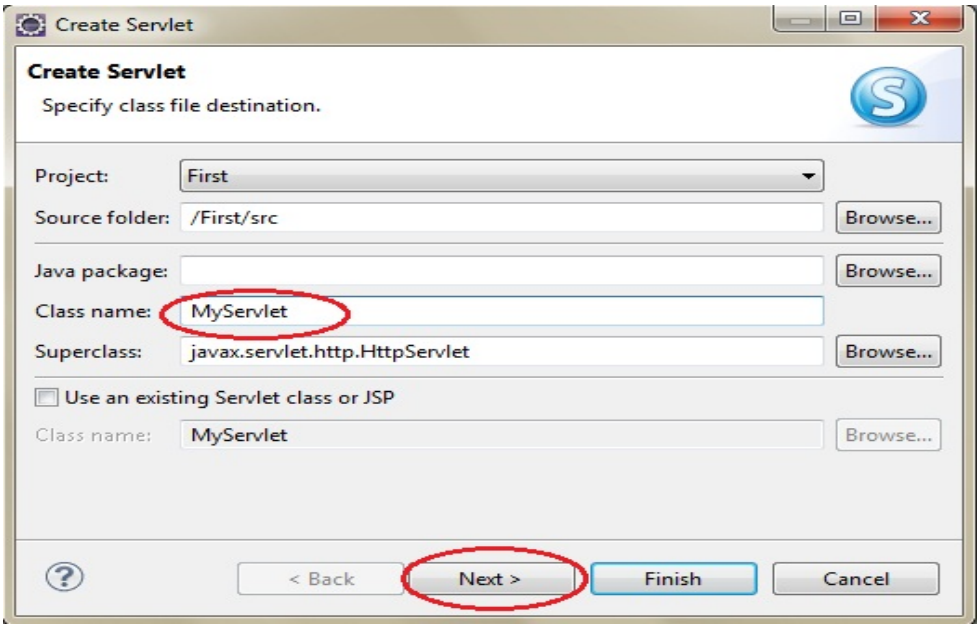
Step 4. **Now, the complete directory structure of your Project will be automatically created by Eclipse IDE**



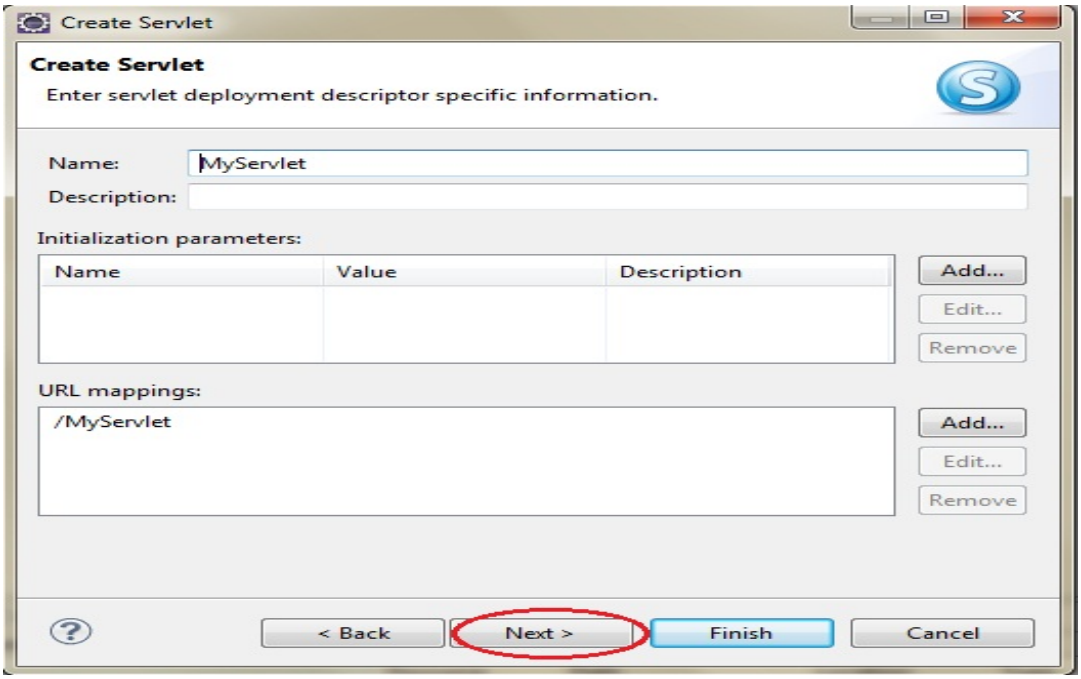
Step 5. **Click on First project, go to Java Resources ->src. Right click on src select New ->Servlet**



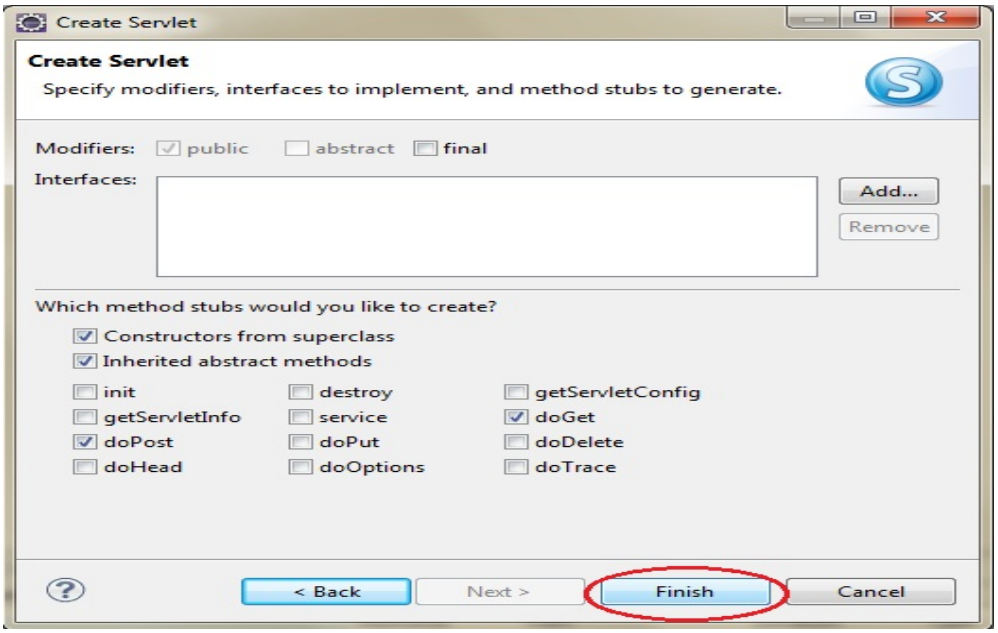
Step 6. **Give Servlet class name and click Next**



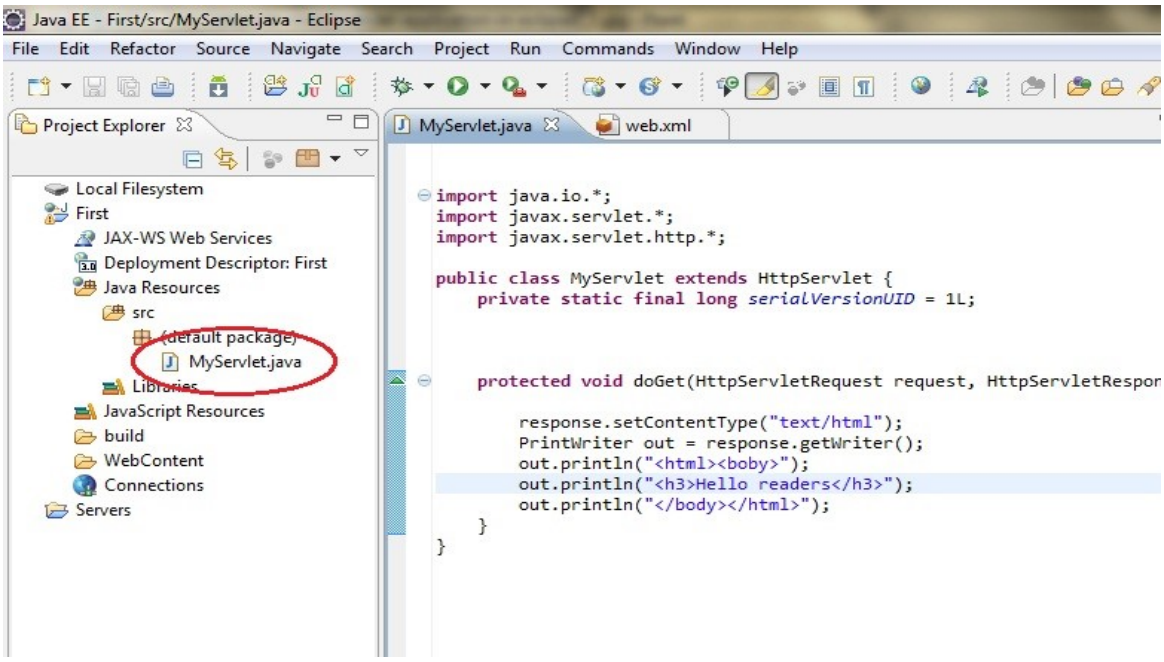
Step 7. **Give your Servlet class a Name of your choice.**



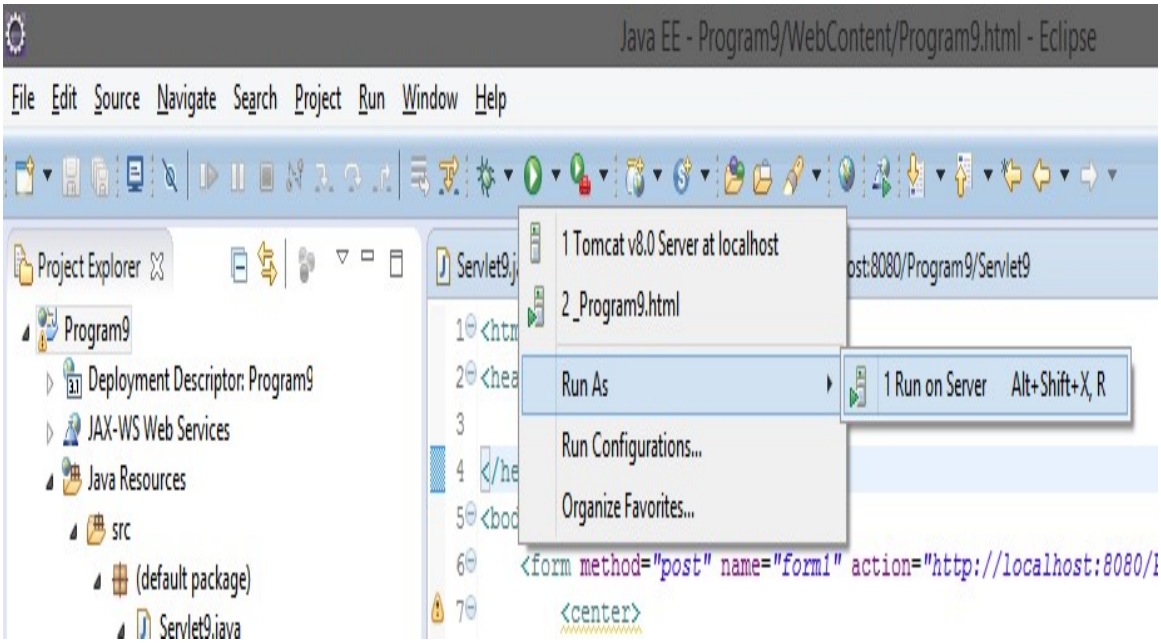
Step 8. **Leave everything else to default and click Finish**



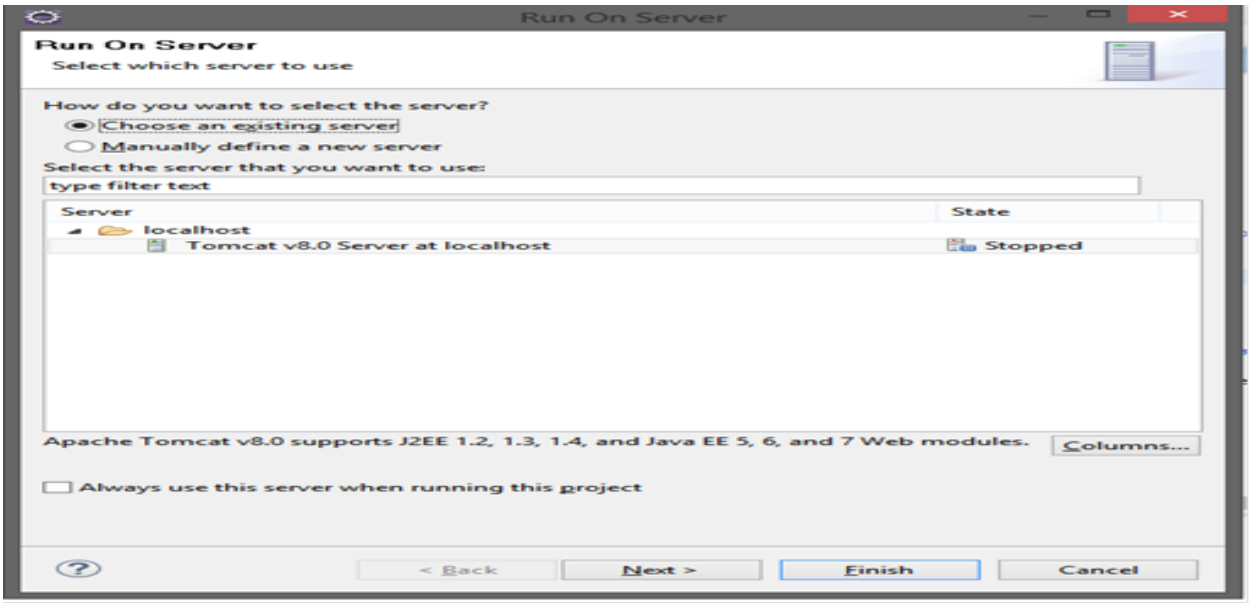
Step 9. **Now your Servlet is created, write the code inside it.**



Step 10. **Now all you have to do is Start the server and run the application.**



Step 11. **Select the existing Tomcat server and click finish**



5) Implement a JAVA Servlet Program to implement a dynamic HTML using Servlet (user name and password should be accepted using HTML and displayed using a Servlet).

Create a new servlet named Servlet9 in the project (as shown in the steps above from Page 37) and then type the following code in it

Servlet9.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/Servlet9")
publicclass Servlet9 extends HttpServlet {

        protectedvoid doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                String str = request.getParameter("uname");
                String str1 = request.getParameter("pname");
                out.println("<html>");
                out.println("<body>");
                out.println("Username is :" + str + "<br/>");
                out.println("Password is :" + str1);
                out.println("</body>");
                out.println("</html>");          }

}
```

Under WebContent, create a new html file, Program9.html

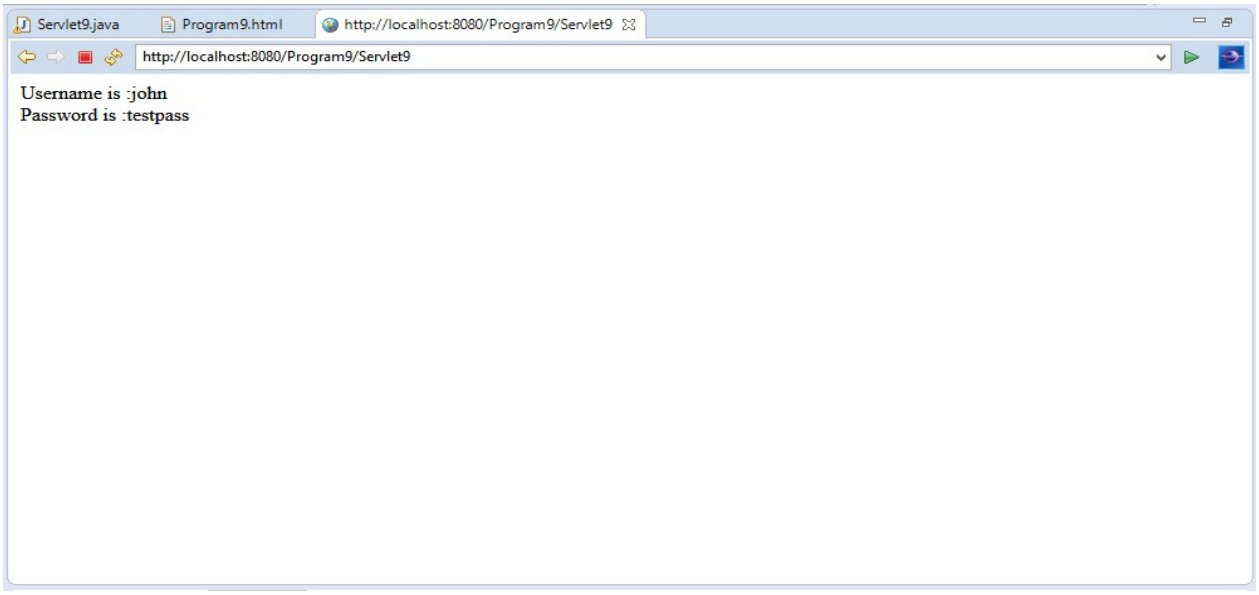
```
<html>
<head>
<title>Program 9</title>
</head>
<body bgcolor=orange>
<form method="post" name="form1"
action="http://localhost:8080/ProjectName/ServletClassName">
<center>
<b><br/><br/>
        Enter Username : <input type="text" name="uname" size="10"/>
```

```
<br/>
        Enter Password : <input type="password" name="pname" size="10"/>
<br/><br/>
<input type="button" value="Submit" onclick="submit()"/>
</center>
<script type="text/javascript">
function validate(){
if(document.form1.uname.value =="" || document.from1.pname.value ==""){
alert("Fields cannot be blank");
return;
            }

        }
</script>
</form>
</body>
</html>
```

In the above html file, replace **ProjectName** and **ServletClassName** with your respective project and filename

Output


Servlet9.java
Program9.html
Program 9
http://localhost:8080/Program9/Program9.html
Enter Username :
john
Enter Password :
Submit


Servlet9.java
Program9.html
http://localhost:8080/Program9/Servlet9
http://localhost:8080/Program9/Servlet9
Username is :john
Password is :testpass

JDBC(JAVA DATABASE CONNECTION)

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

Java Database Connectivity

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows:

- Register the Driver class
- Create connection
- Create statement
- Execute queries
- Close connection

Java Database Connectivity can be connected using Oracle or MySql

Java Database Connectivity with Oracle

To connect java application with the oracle database, we need to follow 5 following steps. In this example, we are using Oracle 10g as the database. So we need to know following information for the oracle database:

1. **Driver class:** The driver class for the oracle database is **oracle.jdbc.driver.OracleDriver**.

2. **Connection URL:** The connection URL for the oracle10G database is **jdbc:oracle:thin:@localhost:1521:xe** where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and XE is the Oracle service name. You may get all these information from the tnsnames.ora file.
3. **Username:** The default username for the oracle database is system.
4. **Password:** It is the password given by the user at the time of installing the oracle database.

Note: Download Load the ojdbc14. jar file

paste the ojdbc14.jar file in jre/lib/ext folder

Java Database Connectivity with MySQL

To connect Java application with the MySQL database, we need to follow 5 following steps.

In this example we are using MySql as the database. So we need to know following informations for the mysql database:

1. **Driver class:** The driver class for the mysql database is **com.mysql.jdbc.Driver**.
2. **Connection URL:** The connection URL for the mysql database is **jdbc:mysql://localhost:3306/Program** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sonoo is the database name. We may use any database, in such case, we need to replace the sonoo with our database name.
3. **Username:** The default username for the mysql database is **root**.
4. **Password:** It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

Note: Download the mysqlconectonr.jar file

Paste the mysqlconectonr.jar file in jre/lib/ext folder

6. Implement JDBC Application Program Using MySQL/ORACLE connectivity, by developing a program to accept book information such as Book-ID, Title, Authors, Edition and Publisher from the table already stored in the database, to perform the following transactions.

1. Display of Table Contents.

2. Insertion of Values to the Table.

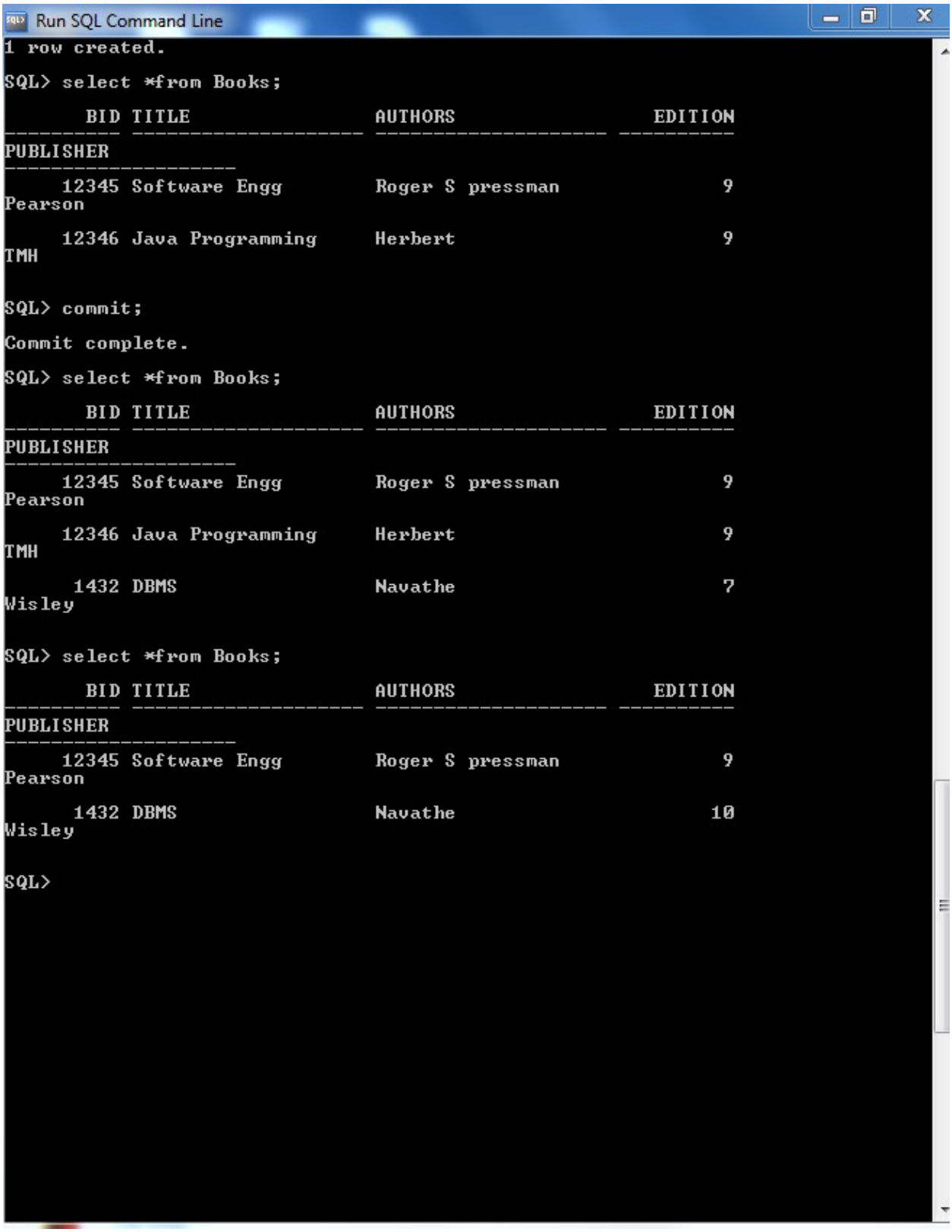
3. Update and Delete contents as and when required

```
import java.sql.*;
class prg{
public static void main(String args[]){

try{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","javanetwork","tiger"
);

Statement stmt=con.createStatement();
//String sql1=("insert into Books values(12,'Java Programming','Herbert',9,'TMH')");
//String sql2=("update Books set edition=9 where Bid=1432");
String sql3=("Delete from Books where Bid=12345");
String sql=("select * from Books");
//ResultSet rs2=stmt.executeQuery(sql1);
//ResultSet rs3=stmt.executeQuery(sql2);
ResultSet rs4=stmt.executeQuery(sql3);
ResultSet rs=stmt.executeQuery(sql);


while(rs.next())
{
System.out.println(rs.getInt(1)+ " " +rs.getString(2)+" "+rs.getString(3)+" "+rs.getInt(4)+"
"+rs.getString(5) );

}
con.close();

}catch(Exception e)
{

System.out.println(e);
```

```
}
}
}
```

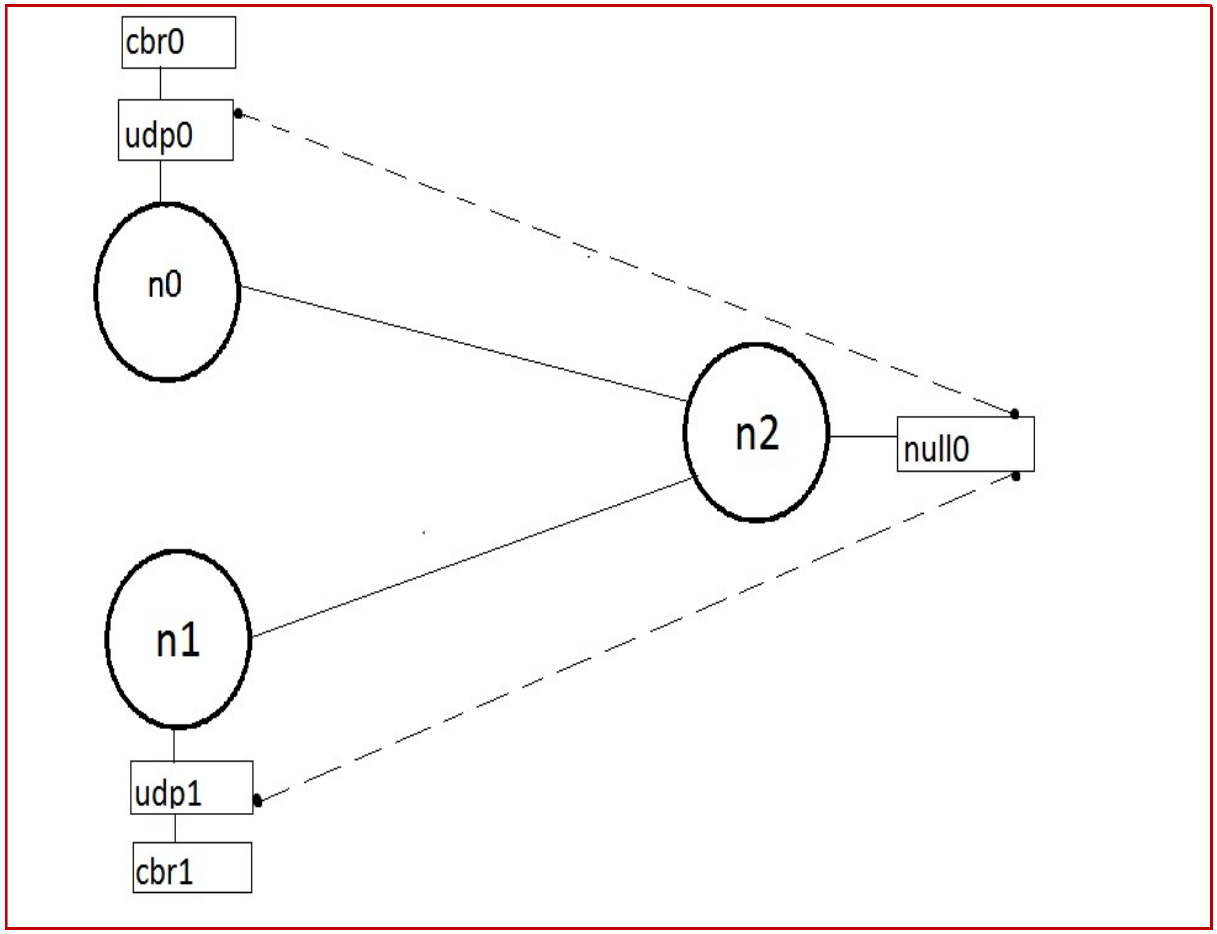
Note:

1. **Create the table Books using SQL Command Prompt or MySql Command Prompt.**
2. **Write an JDBC Program using Ecllipse IDE.**


Run SQL Command Line
1 row created.
SQL> select *from Books;
BID TITLE AUTHORS EDITION
PUBLISHER
12345 Software Engg Roger S pressman 9
Pearson
12346 Java Programming Herbert 9
TMH
SQL> commit;
Commit complete.
SQL> select *from Books;
BID TITLE AUTHORS EDITION
PUBLISHER
12345 Software Engg Roger S pressman 9
Pearson
12346 Java Programming Herbert 9
TMH
1432 DBMS Navathe 7
Wisley
SQL> select *from Books;
BID TITLE AUTHORS EDITION
PUBLISHER
12345 Software Engg Roger S pressman 9
Pearson
1432 DBMS Navathe 10
Wisley
SQL>

PART-B

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



Program(lab1.tcl)

```
set ns [ new Simulator ]

set tf [ open lab1.tr w ]

$ns trace-all $tf

set nf [ open lab1.nam w ]
```

```
$ns namtrace-all $nf

# The below code is used to create the nodes.

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

#This is used to give color to the flow of packets.

$ns color 1 "red"

$ns color 2 "blue"

$n0 label "Source/udp0"

$n1 label "Source/udp1"

$n2 label "destination"

#providing the link

$ns duplex-link $n0 $n2 10kb 100ms DropTail

$ns duplex-link $n1 $n2 10kb 10ms DropTail

# set the queue size b/w the nodes

$ns set queue-limit $n0 $n2  5

$ns set queue-limit $n1 $n2  5

set udp0   [new Agent/UDP]

set udp1  [new Agent/UDP]

$ns attach-agent $n0 $udp0

$ns attach-agent $n1 $udp1
```

```
set cbr0  [new Application/Traffic/CBR]

set cbr1 [new Application/Traffic/CBR]

$cbr0 attach-agent $udp0

$cbr1 attach-agent $udp1

set null0  [new Agent/Null]

$ns attach-agent $n2 $null0
```

#The below code sets the udp0 packets to red and udp1 packets to blue color

```
$udp0 set class_  1

$udp1 set class_  2
```

#The below code is used to connect the agents.

```
$ns connect $udp0 $null0

$ns connect $udp1 $null0
```

#The below code is used to set the packet size to 500

```
$cbr0 set packetSize_  500Mb

$cbr1 set packetSize_  500Mb
```

#The below code is used to set the interval of the packets,

```
$cbr0 set interval_ 0.01

$cbr1 set interval_ 0.01

proc finish  { }  {

global  ns n f  tf

$ns flush-trace
```

```
exec nam lab1.nam &

close $tf

close $nf

exit 0

}

$ns at 0.1 "$cbr0 start"

$ns at 0.1 "$cbr1 start"

$ns at 9.5 "$cbr0 stop"

$ns at 10.0 "$cbr1 stop"

$ns at 10.0 "finish"

$ns run
```

AWK Script (lab1.awk)

```
BEGIN{
count=0;
}
{
if($1=="d")
count++ ;
}
END{
printf("The Total no of Packets Dropped due to Congestion : %d\n\n", count)
}
```

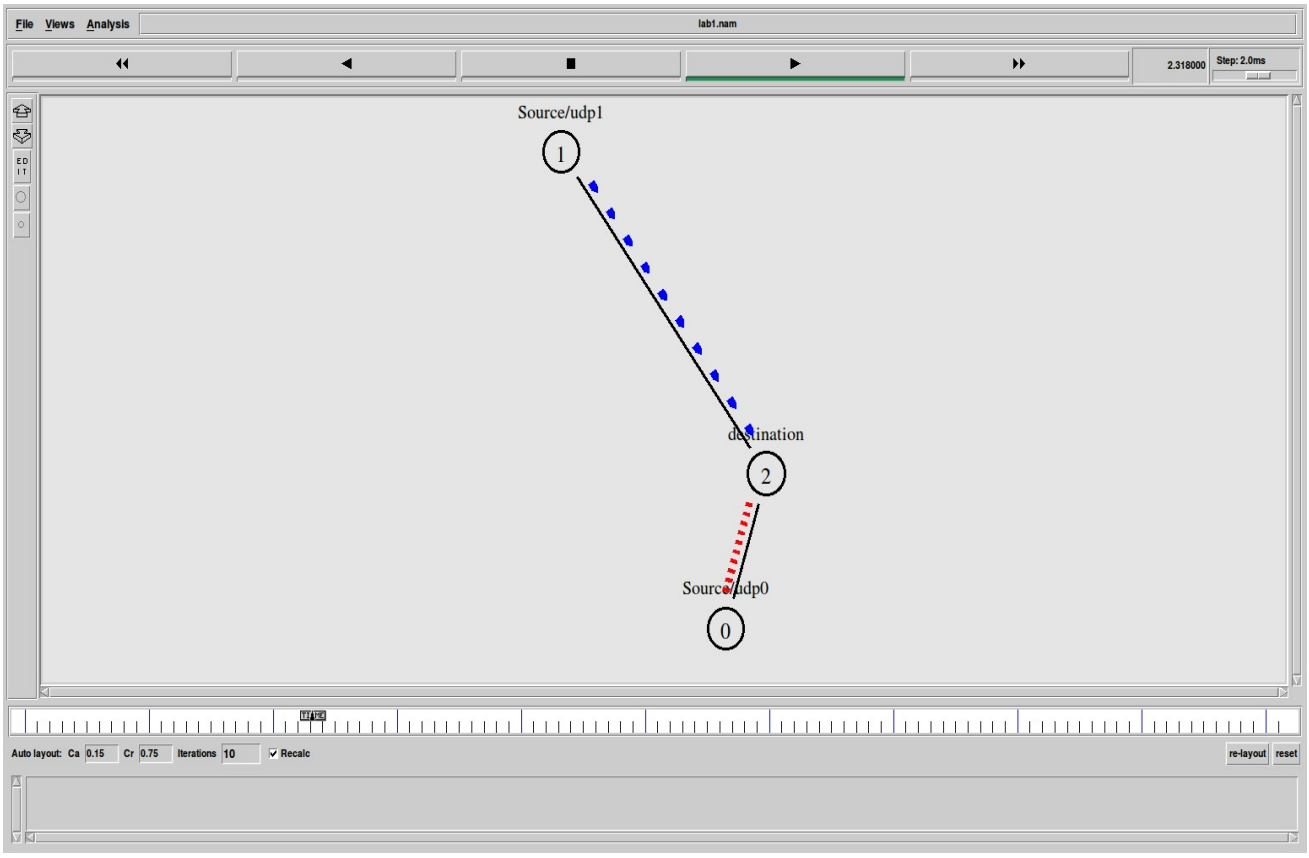
Output:

ns lab1.tcl

awk –f lab1.awk lab1.tr

The Total no of packets Dropped due to congestion:4560

Screenshot



Note:

1. Find the total number of packets dropped by changing the bandwidth as shown below in the table.

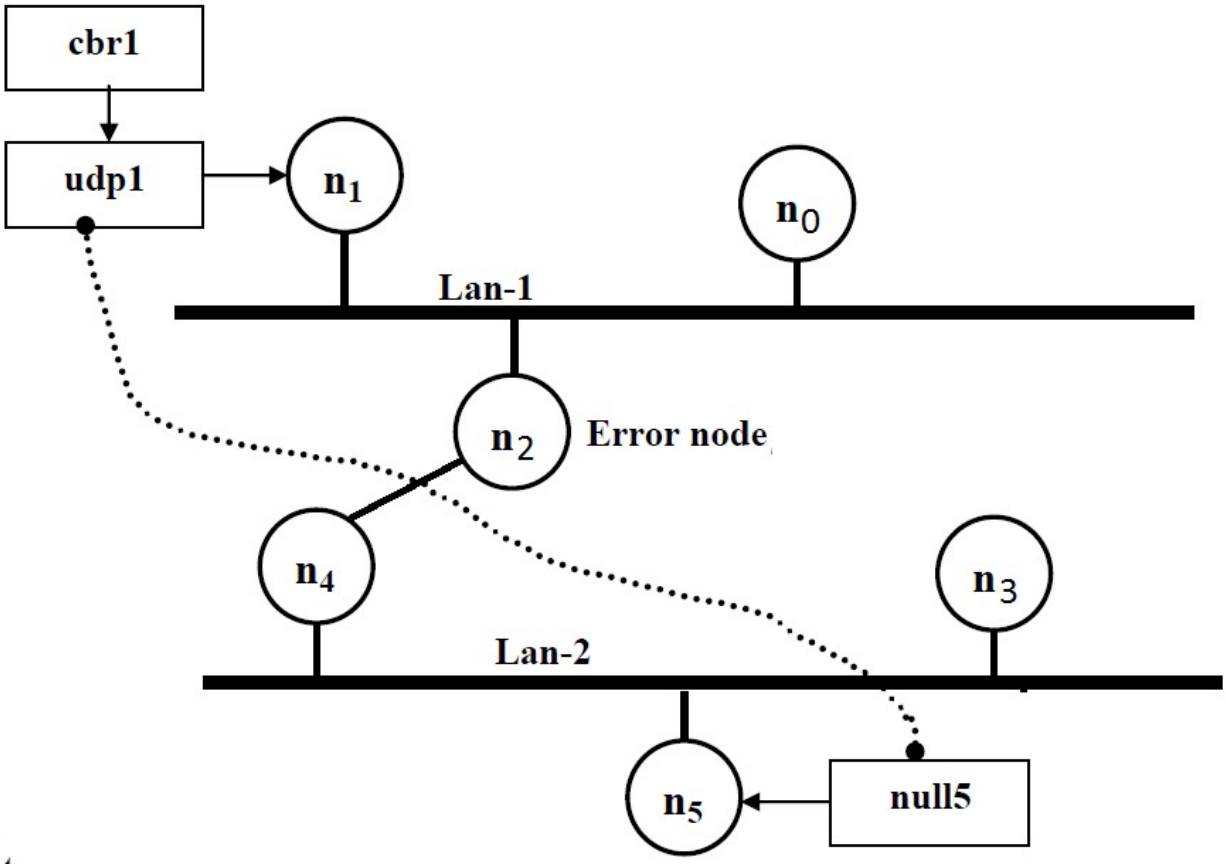
Bandwidth	Queue size	Packets dropped
10kb	5	
100kb	5	
1Mb	5	

2. Find the total number of packets dropped by changing the queue size as shown below in the table.

Bandwidth	Queue size	Packets dropped
100kb	3	
100kb	5	
100kb	10	

2. Simulate an Ethernet lan using n nodes (6-10), change the error rate and data rate and compare the throughput.

Design:



Note:

1. The lan can be created by using the command:

$ns make-lan“$n0 $n1 $n2” 100Mb 10ms LL Queue/DropTail Mac/802_3

2. The Error between the nodes n2 and n4 can be added as follows:

set err [new ErrorModel]

$ns lossmodel $err $n2 $n4

$err set rate_ 0.1 # used to set error rate.

3. The throughput can analyzed by changing the data rate and error rate as shown below.

i. First, fix the data rate to 0.001 and vary the error rate, as below.Tabulate the throughput.
ii. Secondly, fix the error rate to 0.1 and vary the data rate, as below. Tabulate the throughput.

Example:

Error rate	Data rate	Throughput
0.1	0.001	
0.2	0.001	
0.3	0.001	
0.4	0.001	

Error rate	Data rate	Throughput
0.1	0.1	
0.1	0.01	
0.1	0.001	
0.1	0.0001	

Program(lan.tcl)

```
set ns  [new Simulator]
set  tf  [open lan.tr w]
$ns trace-all $tf
set nf  [open lan.nam w]
$ns namtrace-all $nf
set n0  [$ns node]
set n1  [$ns node]
set n2  [$ns node]
set n3  [$ns node]
set n4  [$ns node]
set n5  [$ns node]
$ns color 1 "blue"
$n1 label  “Source"
$n2  label  "Error node"
$n5  label  "Destination"
```

#The below code is used to create two Lans (Lan1 and Lan2).

```
$ns make-lan "$n0 $n1 $n2" 10Mb 10ms LL Queue/DropTail  Mac/802_3
$ns make-lan "$n3 $n4 $n5" 10Mb 10ms LL Queue/DropTail  Mac/802_3
```

 # connect node n2 and n4

```
$ns duplex-link $n2 $n4 100Mb 10ms DropTail
$ns duplex-link-op $n2 $n4 color "green"
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
set null5 [new Agent/Null]
$ns attach-agent $n5 $null5
```

```
$ns connect $udp1 $null5

#data rate- change this to change the data rate
$cbr1 set packetSize_  1000
$cbr1 set interval_  0.001
$udp1 set class_  1

# The below code is used to add an error model between the nodes n2 and n4.
set err  [new ErrorModel]
$ns lossmodel $err $n2 $n4
$err set rate_   0.1
proc finish {}  {
        global ns tf nf
        $ns flush-trace
        exec nam lan.nam&
        close $tf
        close $nf
        exit 0
}
$ns at 0.1 "$cbr1 start"
$ns at 6.0 "finish"
$ns run
```

AWK Script (lan.awk)

```
BEGIN{
        pkt=0;
        time=0;
}
```

```
{
        if($1=="r"&& $9=="1.0" && $10=="5.0")
        {
                pkt=pkt+$6;
                time=$2;
        }
}
END{
        printf("Throughput: %f Mbps \n\n",(pkt/time)*(8/1000000));
}
```

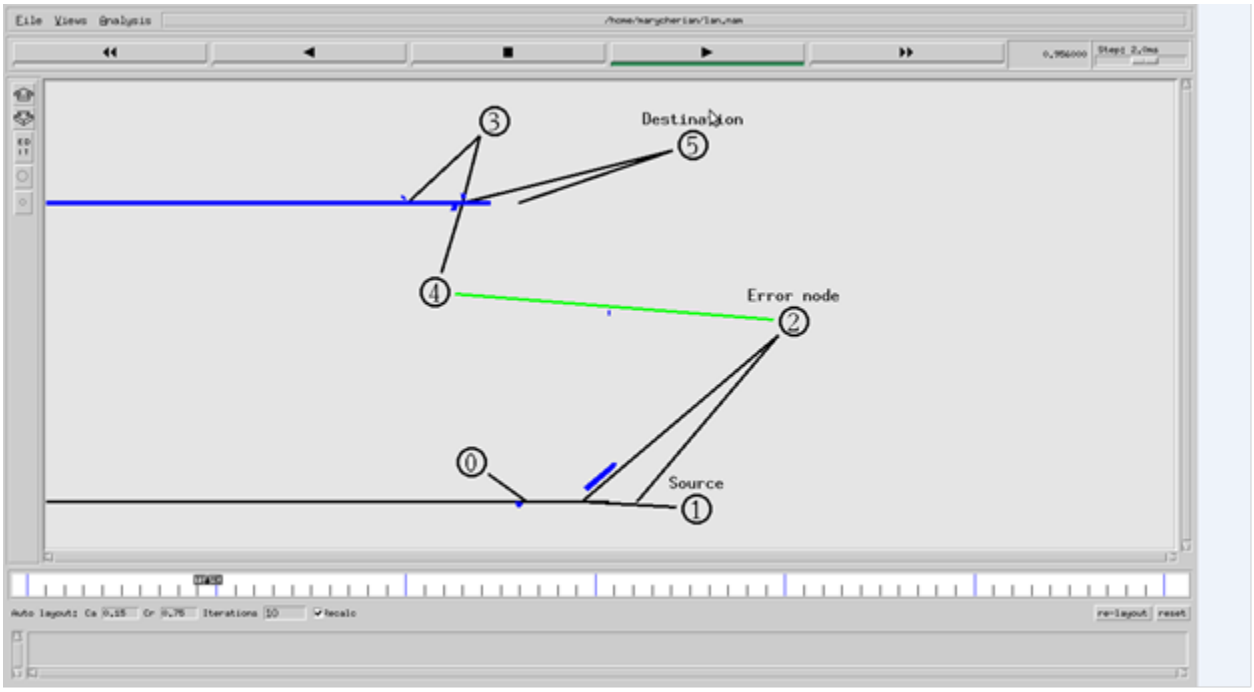
Execution of the program

Type the following commands in the terminal window for executing the programs.

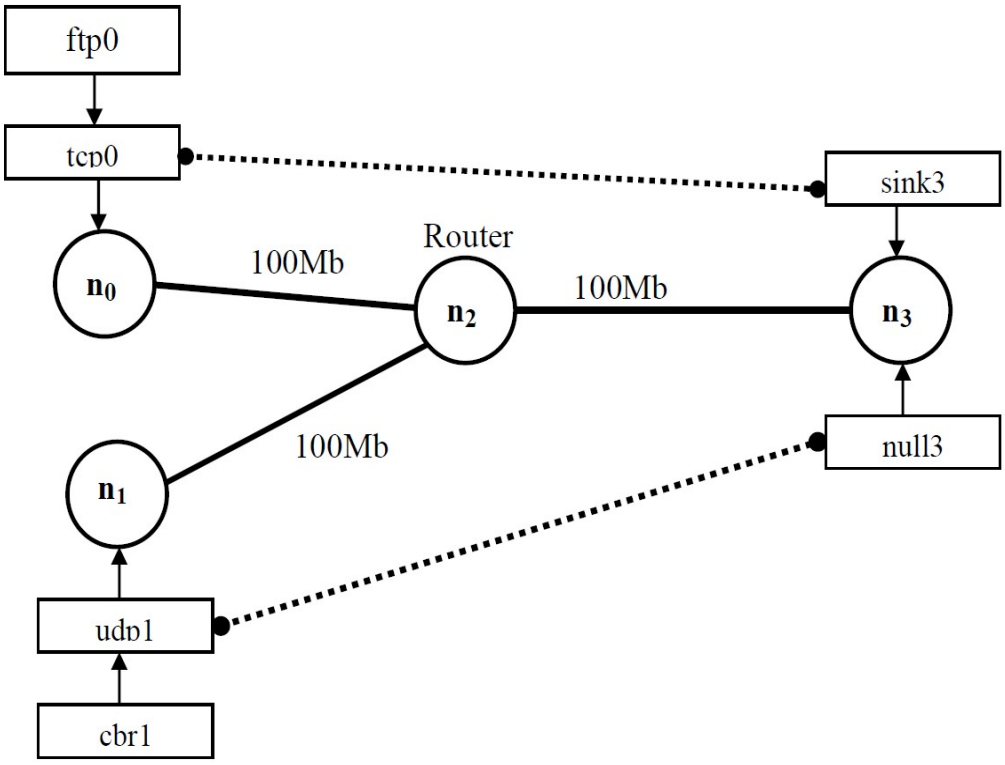
ns lan.tcl
awk –f lan.awk lan.tr

Output:

Throughput : 90.3Mbps
Snapshot:


File Views Analysis
Destination
5
3
4
Error node
2
0
Source
1
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset

3. Simulate a four node point –to- point network with links connected as follows: n0-n2, n1-n2 and n2-n3. Apply TCP agents between n0-n3 and UDP agents between n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP.



Example:

Note:

1. Find the total number of packets sent by TCP and UDP by changing the bandwidth 100Mb, 200Mb in the topology and the data rate 0.01, 0.001 as shown below in the table.

BANDWIDTH	PACKET INTERVAL	TCP PACKETS SENT	UDP PACKETS SENT
100Mb	0.01		
100Mb	0.001		

200Mb	0.01		
200Mb	0.001		

PROGRAM (lab3.tcl)

```
set ns [new Simulator]
set  tf  [open lab3.tr w]
$ns trace-all $tf
set nf  [open lab3.nam w]
$ns namtrace-all $nf
set  n0  [$ns node]
set  n1  [$ns node]
set  n2  [$ns node]
set  n3  [$ns node]

# The below code is used to set the color and name's to the nodes.
$ns color 1 "red"
$ns color 2 "blue"
$n0 label "Source/TCP"
$n1 label "Source/UDP"
$n2 label "Router"
$n3 label "Destination"
$ns duplex-link$n0 $n2 100Mb 1ms DropTail
$nsduplex-link $n1 $n2 100Mb 1ms DropTail
$ns duplex-link $n2 $n3 100Mb 1ms DropTail

# The below code is used to set the color and labels to the links.
$ns duplex-link-op $n0 $n2 color "green"
$ns duplex-link-op  $n0 $n2 label "from 0-2"
$ns duplex-link-op  $n1 $n2 color "green"
$ns duplex-link-op  $n1 $n2 label "from 1-2"
$ns duplex-link-op  $n2 $n3 color "green"
```

$ns **duplex-link-op** $n2 $n3 label "from 2-3"

 # The below code is used create TCP and UDP agents and the
 # traffic ftp &cbr respectively.

set tcp0 **[new Agent/TCP]**
$ns attach-agent $n0 $tcp0
set ftp0 **[new Application/FTP]**
$ftp0 attach-agent $tcp0
set sink3 **[new Agent/TCPSink]**
$ns attach-agent $n3 $sink3
set udp1 **[new Agent/UDP]**
$ns attach-agent $n1 $udp1
set cbr1 **[new Application/Traffic/CBR]**
$cbr1 attach-agent $udp1
set null3 **[new Agent/Null]**
$ns attach-agent $n3 $null3

#The below code is used to set the packet size of ftp and udp.
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.001

#The below code is used set the data rate
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.001

#This code is used give a color red->tcp and blue ->udp.
$tcp0 set class_ 1
$udp1 set class_ 2

 # The below code is used connect the agents.

$ns connect $tcp0 $sink3

$ns connect $udp1 $null3

proc finish { } {

global ns nf tf

$ns flush-trace

exec nam lab3.nam **&**

close $nf

close $tf

exit 0

}

$ns at 0.1 "$cbr1 start"

$ns at 0.2 "$ftp0 start"

$ns at 9.5 “$cbr1 stop"

$ns at **9.5 “**$ **ftp0** stop"

$ns at 10.0 "finish"

$ns run

AWK Script (lab3.awk)

BEGIN{

tcp=0;

udp=0;

}

{

if($1=="r"&&$3=="2"&&$4=="3"&& $5=="tcp")

tcp++;

if($1=="r"&&$3=="2"&&$4=="3"&&$5=="cbr")

udp++;

```
}
END{
printf("\n Total number of packets sent by TCP : %d\n",tcp);
printf("\n Total number of packets sent by UDP : %d\n",udp);
}
```

Execution of the program

Type the following commands in the terminal window for executing the programs.

ns lab2.tcl

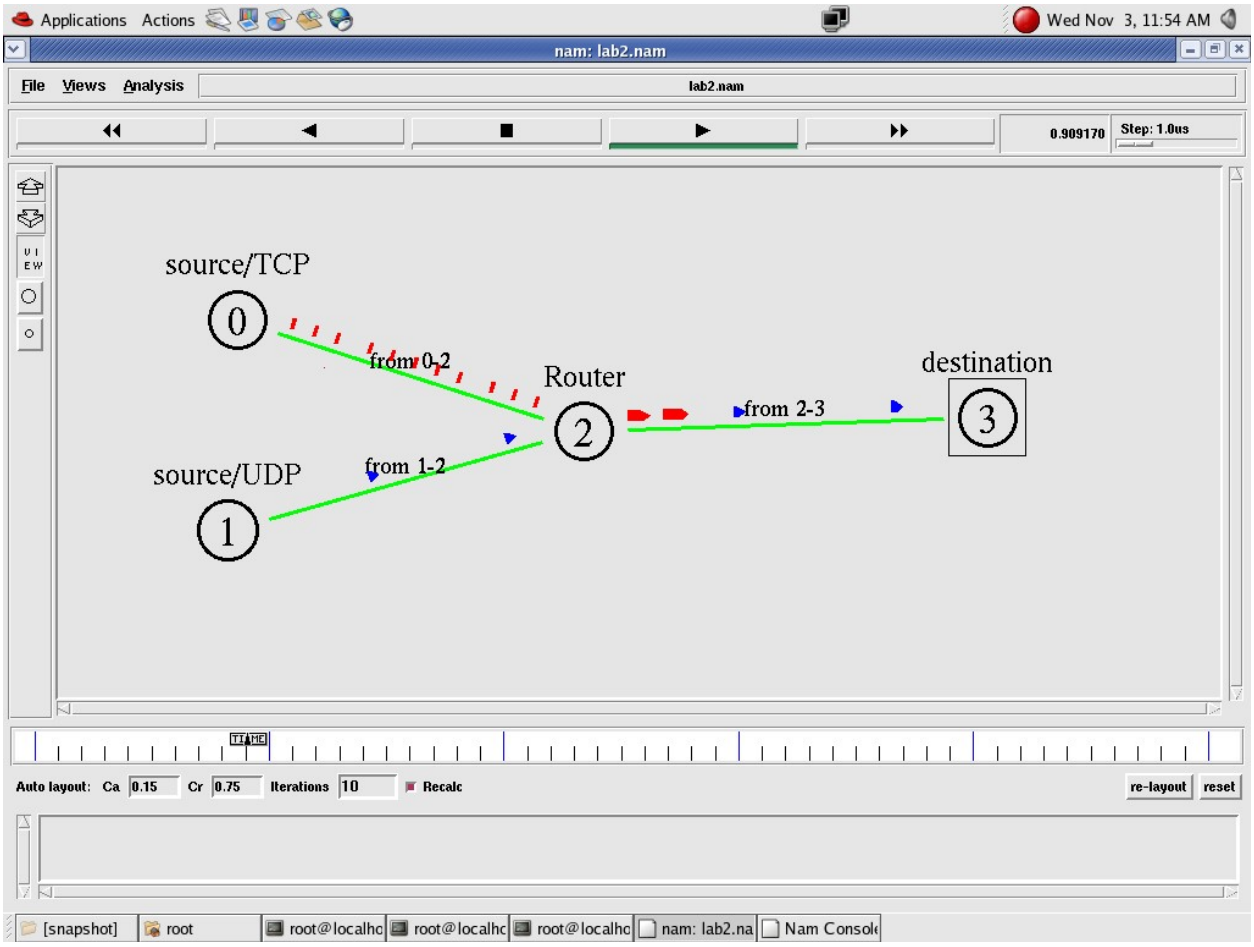
awk –f lab3.awk lab3.tr

Output:

Total number of packets sent by TCP : 1200

Total number of packets sent by UDP : 4500

Screenshot



4 To create scenario and study the performance of Stop and Wait ARQ Protocol through simulation.

#window-The upper bound on the advertised window for the TCP connection.

#**maxcwnd -** The upper bound on the congestion window for the TCP connection. Set to zero to ignore. (This is the default.)

#Agents are used to separate protocol states from nodes. They are always associated with nodes.

An agent has a name, which is a unique identifier of the agent. It is shown as a square with its name inside, and a line link the square to its associated node. The add agent and monitor agent are commands that support agent tracing.

#orient – The orientation of the link (up, down, right, left , right up, right down, left up, left down)

#nam_tracevar_true : It is format for TCP variable, required by nam visualization of TCP.

 # detach-agent - Detach the agent of type agent from the node

 # trace- annotate: creates trace file. **It will provide the information about the packet transfer event at the bottom of the simulation window**

Program(stop.tcl)

```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
$ns color 1 "red"
$ns at 0.0 "$n0 label Sender"
$ns at 0.0 "$n1 label Receiver"
set nf  [open stop.nam w]
$ns namtrace-all $nf
```

```
set tf [open stop.tr w]
$ns trace-all $tf
$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail
$ns duplex-link-op $n0 $n1 orient right
$ns queue-limit $n0 $n1 10
Agent/TCP set nam_tracevar_ true
set tcp  [new Agent/TCP]
$tcp set window_ 1
$tcp set maxcwnd_ 1
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns add-agent-trace $tcp tcp
$ns monitor-agent-trace $tcp
$tcp tracevar cwnd_
$tcp set class_ 1
$ns at 0.1 "$ftp start"
$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"
$ns at 3.5 "finish"
$ns at 0.0 "$ns trace-annotate \"Stop and Wait with normal operation\""
$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""
$ns at 0.11 "$ns trace-annotate \"Send Packet_0\""
$ns at 0.35 "$ns trace-annotate \"Receive Ack_0\""
$ns at 0.56 "$ns trace-annotate \"Send Packet_1\""
$ns at 0.79 "$ns trace-annotate \"Receive Ack_1\""
$ns at 0.99 "$ns trace-annotate \"Send Packet_2\""
$ns at 1.23 "$ns trace-annotate \"Receive Ack_2 \""
$ns at 1.43 "$ns trace-annotate \"Send Packet_3\""
```

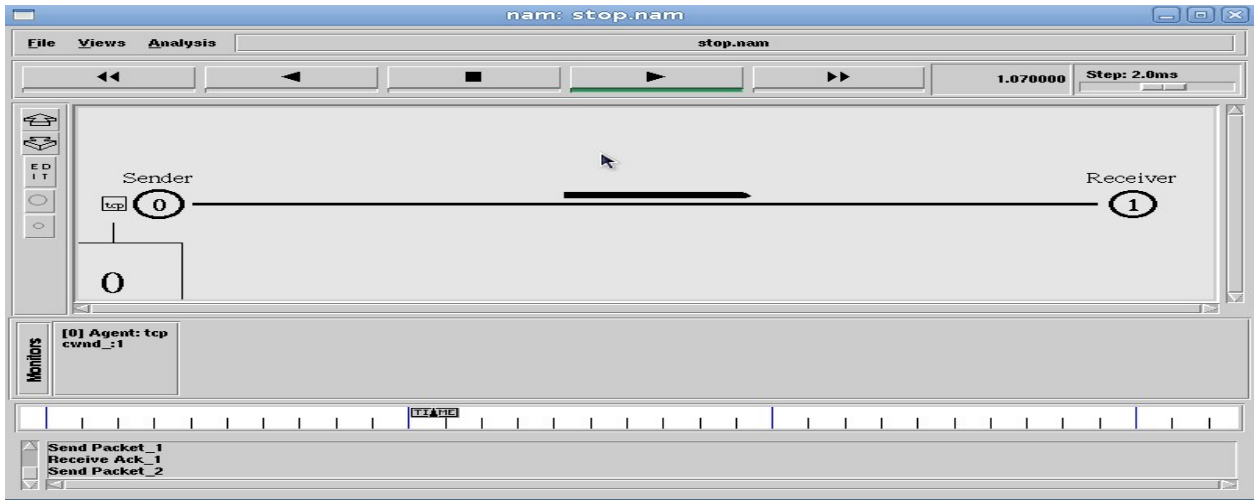
```
$ns at 1.67 "$ns trace-annotate \"Receive Ack_3\""
$ns at 1.88 "$ns trace-annotate \"Send Packet_4\""
$ns at 2.11 "$ns trace-annotate \"Receive Ack_4\""
$ns at 2.32 "$ns trace-annotate \"Send Packet_5\""
$ns at 2.55 "$ns trace-annotate \"Receive Ack_5 \""
$ns at 2.75 "$ns trace-annotate \"Send Packet_6\""
$ns at 2.99 "$ns trace-annotate \"Receive Ack_6\""
$ns at 3.1 "$ns trace-annotate \"FTP stops\""
proc finish { } {
global ns nf
$ns flush-trace
close $nf
puts "running nam..."
exec nam stop.nam &
exit 0
}
$ns run
```

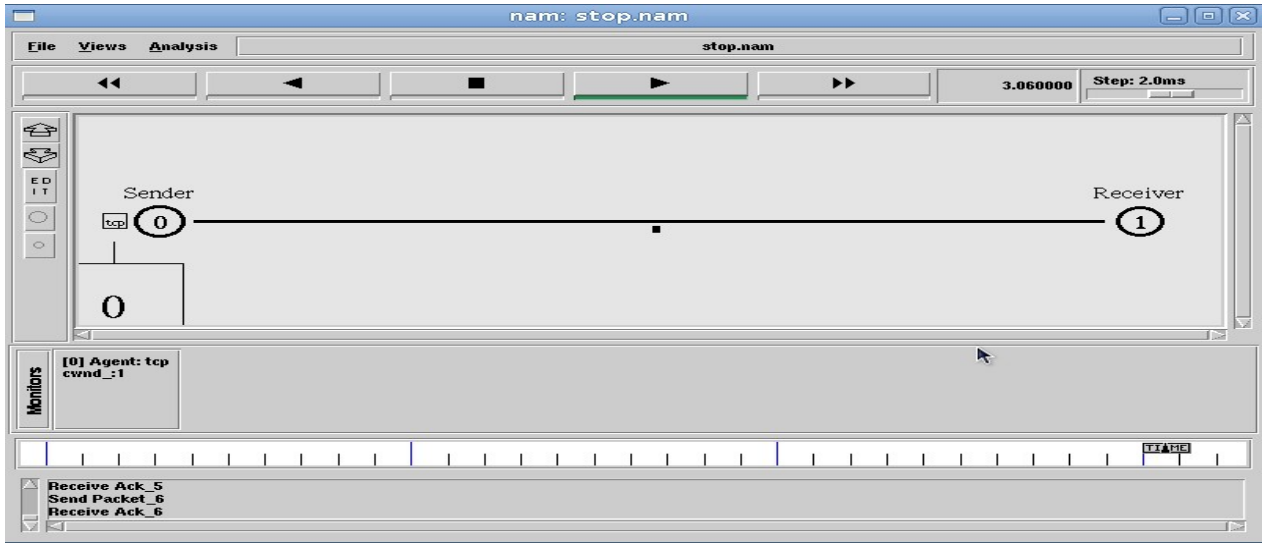
Execution of the program

Type the following commands in the terminal window for executing the programs.

ns stop.tcl

Snapshot :


nam: stop.nam
File
Views
Analysis
stop.nam
1.070000
Step: 2.0ms
Sender
Receiver
tcp
0
1
0
Monitors
[0] Agent: tcp
cwnd_:1
TIME
Send Packet_1
Receive Ack_1
Send Packet_2


nam: stop.nam
File
Views
Analysis
stop.nam
3.060000
Step: 2.0ms
Sender
Receiver
tcp
0
1
0
Monitors
[0] Agent: tcp
cwnd_:1
TIME
Receive Ack_5
Send Packet_6
Receive Ack_6

5. Simulate simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.
Simulation Parameters:

Area **:**700m *700m

Simulation Time **:**250 simulation sec

Wireless nodes **:**4

Routing Protocol **:**DSDV (Destination Sequenced Distance

vector)

Interface queue Type **:**Queue/DropTail

MAC **:**802.11

Application **:**FTP

Antenna **:**Omni antenna

(wireless.tcl)

```
set ns [new Simulator]
set tf [open wireless.tr w]
$ns trace-all $tf
set topo [new Topography]
$topo load_flatgrid 700 700
set nf  [open wireless.nam w]
$ns namtrace-all-wireless  $nf  700 700
$ns node-config      -adhocRouting DSDV\
                          -llType LL\
                          -ifqType Queue/DropTail\
                          -macType Mac/802_11\
                          -ifqLen 50 \
                          -phyTypePhy/WirelessPhy\
                          -channelType Channel/WirelessChannel\
                          -propType Propagation/TwoRayGround\
                          -antType Antenna/OmniAntenna\
```

```
                    -topoInstance $topo\
                    -agentTrace ON\
                   -routerTrace ON
create-god  4
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$n0 label "tcp0"
$n1 label "sink1"
$n2 label "tcp1"
$n3 label "sink2"
```

#The below code is used to give the initial node positions.

```
$n0 set  X_  50
$n0 set  Y_  50
$n0 set  Z_ 0
$n1 set  X_ 200
$n1 set  Y_ 200
$n1 set  Z_ 0
$n2 set  X_ 400
$n2 set  Y_ 400
$n2 set  Z_ 0
$n3 set  X_  600
$n3 set Y_  600

$n3 set Z_  0
$ns at 0.1  "$n0 setdest 50 50 15"
$ns at 0.1  "$n1  setdest 200 200 25"
$ns at 0.1  "$n2 setdest 400 400 25"
$ns at 0.1  "$n3 setdest 600 600 25"
```

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0  [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink1  [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns connect $tcp0 $sink1
set tcp1  [new Agent/TCP]
$ns attach-agent $n2 $tcp1
set ftp1  [new Application/FTP]
$ftp1 attach-agent $tcp1
set sink2  [new Agent/TCPSink]
$ns attach-agent $n3 $sink2
$ns connect $tcp1 $sink2
$ns at 5 "$ftp0 start"
$ns at 10 "$ftp1 start"
```

#The below code is used to give node movements .

```
$ns  at 100  "$n2  setdest  500 500 25"

proc finish {} {
globalnf ns tf
$ns flush-trace
exec nam wireless.nam&
close $tf
exit 0
}
$ns at 250 "finish"
$ns run
```

AWK Script (wireless.awk)

```
BEGIN{
        count1=0
        count2=0
        pack1=0
        pack2=0
        time1=0
        time2=0
}
{
        if($1=="r" && $3=="_1_" && $4=="AGT")
        {
                count1++
                pack1=pack1+$8
                time1=$2
        }
        if($1=="r" && $3=="_3_" && $4=="AGT")
        {
                count2++
                pack2=pack2+$8
```

```
                time2=$2
        }
}
END{
        printf("The Throughput  from n0 to n1:
%fMbps\n",((count1*pack1*8)/(time1*1000000)))

        printf(" The Throughput  from n2 to n3:
%fMbps\n",((count2*pack2*8)/(time2*1000000)))
}
```

Execution of the program

Type the following commands in the terminal window for executing the programs.

ns wireless.tcl
awk –f wireless.awk wireless.tr

Output:

The Throughput from n0 to n1: 5444Mbps

The Throughput from n2 to n3: 345Mbps

Snapshot :

Snap Shot: