

Spring 2016

Performance Improvement in Packet Buffers for High Bandwidth Routers

Pradeep Korlamanda
Governors State University

Rajitha Siripurapu
Governors State University

Satya Deepthi Vemula
Governors State University

Follow this and additional works at: <http://opus.govst.edu/capstones>

 Part of the [Computer Sciences Commons](#)

Recommended Citation

Korlamanda, Pradeep; Siripurapu, Rajitha; and Vemula, Satya Deepthi, "Performance Improvement in Packet Buffers for High Bandwidth Routers" (2016). *All Capstone Projects*. 205.
<http://opus.govst.edu/capstones/205>

For more information about the academic degree, extended learning, and certificate programs of Governors State University, go to http://www.govst.edu/Academics/Degree_Programs_and_Certifications/

Visit the [Governors State Computer Science Department](#)

This Project Summary is brought to you for free and open access by the Student Capstone Projects at OPUS Open Portal to University Scholarship. It has been accepted for inclusion in All Capstone Projects by an authorized administrator of OPUS Open Portal to University Scholarship. For more information, please contact opus@govst.edu.

Table of Contents

1	<i>Project Description</i>	3
1.1	Project Abstract	3
1.2	Existing System	3
1.3	Proposed System.....	3
2	<i>Introduction</i>	3
3	<i>Project Requirements</i>	4
3.1	Software Requirements.....	4
3.2	Hardware Requirements	4
4	<i>Project Design Description</i>	5
4.1	<i>System Design</i>	
4.1.1	Definition.....	5
4.1.2	UML is a language.....	5
4.1.3	UML Specifying.....	5
4.1.4	UML Visualization.....	5
4.1.5	UML Constructing.....	5
4.1.6	UML Documenting.....	5
4.2	Uses Of UML.....	7
4.2.1	Building Blocks Of UML.....	7
4.2.1.1	Things.....	7
4.2.1.2	Relationships.....	8
4.3	Usecase Diagrams.....	8
4.4	Class Diagram.....	9
4.5	Sequence Diagram.....	10
4.6	Collaboration Diagram.....	11
4.7	State Machine Diagram.....	12
4.8	Activity Diagram.....	13
4.9	Component Diagram.....	14
4.10	Deployment Diagram.....	15
5	<i>Sample Screens</i>	17
6	<i>Technologies Used</i>	25
6.1	Introduction to Java	25
6.2	Swings	28
7	<i>Sample Code</i>	30
8	<i>System Testing</i>	57
8.1	Types of Tests.....	57
8.2	Unit Testing.....	59
8.3	Integration Testing.....	59
8.4	Acceptance Testing.....	59
9	<i>Conclusion And Future Work</i>	60
10	References	60

1 Project Description

1.1 Project Abstract

High-speed routers rely on well-designed packet buffers that support multiple queues, provide large capacity and short response times. Some researchers suggested combined SRAM/DRAM hierarchical buffer architectures to meet these challenges. However, these architectures suffer from either large SRAM requirement or high time-complexity in the memory management. In this paper, we present scalable, efficient, and novel distributed packet buffer architecture. Two fundamental issues need to be addressed to make this architecture feasible: 1) how to minimize the overhead of an individual packet buffer; and 2) how to design scalable packet buffers using independent buffer subsystems. We address these issues by first designing an efficient compact buffer that reduces the SRAM size requirement by $(k - 1)/k$. Then, we introduce a feasible way of coordinating multiple subsystems with a load-balancing algorithm that maximizes the overall system performance. Both theoretical analysis and experimental results demonstrate that our load-balancing algorithm and the distributed packet buffer architecture can easily scale to meet the buffering needs of high bandwidth links and satisfy the requirements of scale and support for multiple queues.

1.1.1 Existing System:

The router buffer sizing is still an open issue. The traditional rule of thumb for Internet routers states that the routers should be capable of buffering $RTT \cdot R$ data, where RTT is a round-trip time for flows passing through the router, and R is the line rate. Many researchers claimed that the size of buffers in backbone routers can be made very small at the expense of a small loss in throughput. Focusing on the performance of individual TCP flows, researchers claimed that the output/input capacity ratio at a network link largely determines the required buffer size. If the output/input capacity ratio is lower than one, the loss rate follows a power-law reduction with the buffer size and significant buffering is needed.

1.1.2 Proposed System:

We devise a “traffic-aware” approach which aims to provide different services for different types of data streams. This approach further reduces the system overhead. Both mathematical analysis and simulation demonstrate that the proposed architecture together with its algorithm reduce the overall SRAM requirement significantly while providing guaranteed performance in terms of low time complexity, upper bounded drop rate, and uniform allocation of resources.

2 Project Requirements

2.1 Hardware Requirements

3 Project Design Description

3.1 System Design:

UML is a method for describing the system architecture in detail using the blueprint.

UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

UML is a very important part of developing objects oriented software and the software development process.

UML uses mostly graphical notations to express the design of software projects.

Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

3.1.1 Definition:

UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of the software system.

3.1.2 UML is a language:

It will provide vocabulary and rules for communications and function on conceptual and physical representation. So it is modeling language.

3.1.3 UML Specifying:

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

3.1.4 UML Visualization:

The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

3.1.5 UML Constructing:

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

3.1.6 UML Documenting:

UML provides variety of documents in addition raw executable codes.

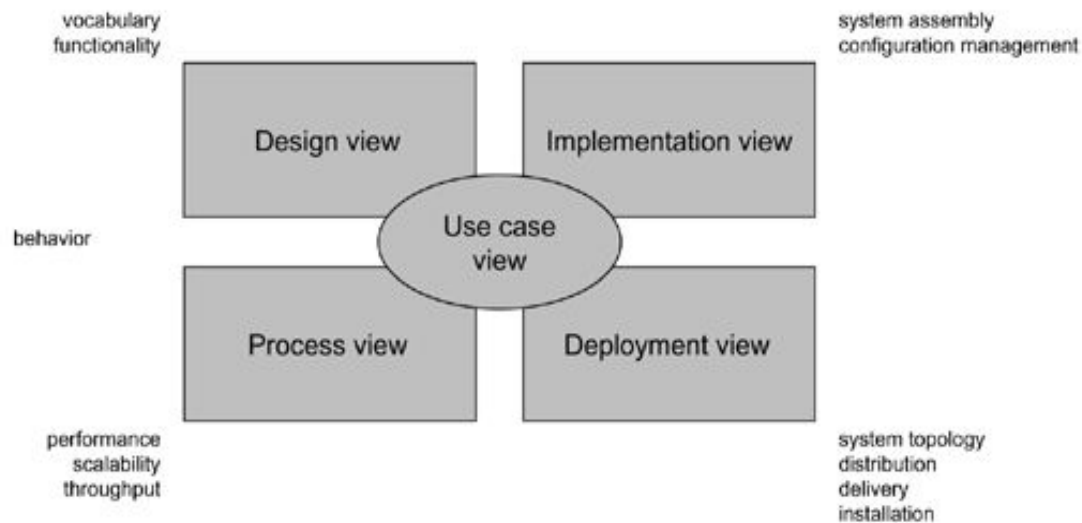


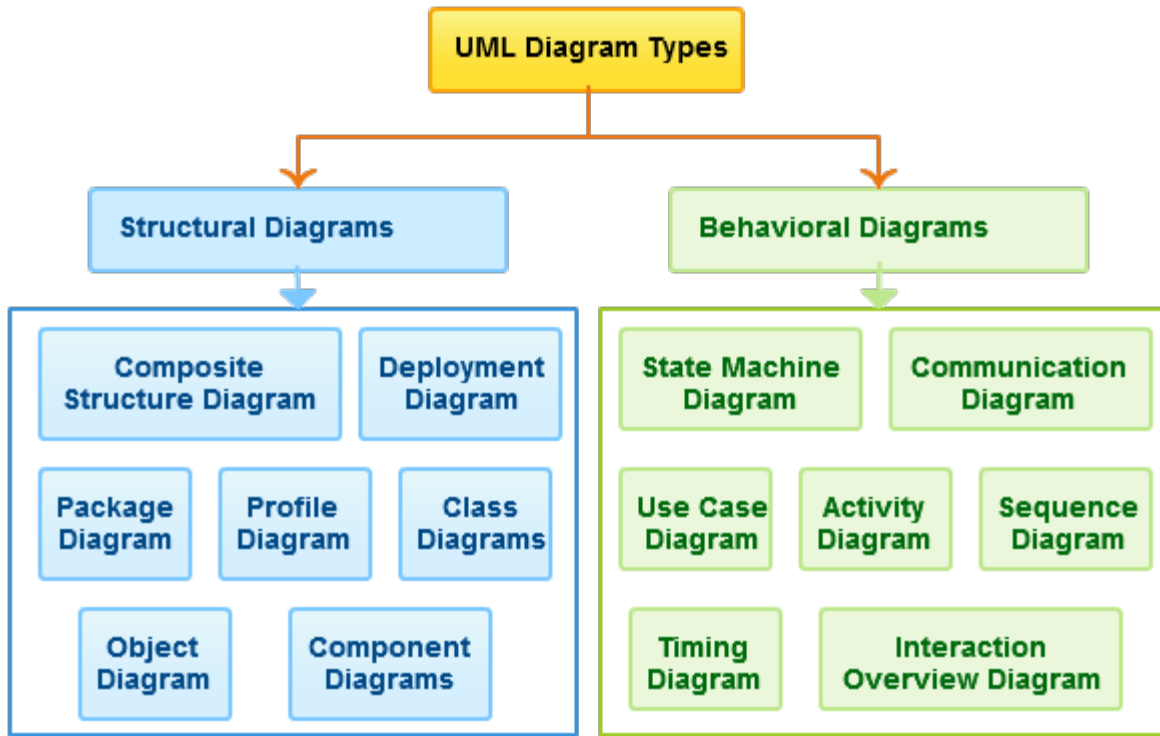
Figure 3.4 Modeling a System Architecture using views of UML

The use case view of a system encompasses the use cases that describe the behavior of the system as seen by its end users, analysts, and testers.

The *design view* of a system encompasses the classes, interfaces, and collaborations that form the vocabulary of the problem and its solution.

The *process view* of a system encompasses the threads and processes that form the system's concurrency and synchronization mechanisms.

The *implementation view* of a system encompasses the components and files that are used to assemble and release the physical system. The *deployment view* of a system encompasses the nodes that form the system's hardware topology on which the system executes.



[online diagramming & design] creately.com

3.2 Uses of UML :

The UML is intended primarily for software intensive systems. It has been used effectively for such domain as

- Enterprise Information System
- Banking and Financial Services
- Telecommunications
- Transportation
- Defense/Aerosp
- Retails
- Medical Electronics
- Scientific Fields
- Distributed Web

3.2.1 Building blocks of UML:

The vocabulary of the UML encompasses 3 kinds of building blocks

- Things
- Relationships
- Diagrams

3.2.1.1 Things:

Things are the data abstractions that are first class citizens in a model. Things are of 4 types

3.2.1.2 Relationships:

Relationships tie the things together. Relationships in the UML are

Dependency, Association, Generalization, Specialization

UML Diagrams:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).

There are two types of diagrams, they are:

Structural and Behavioral Diagrams

Structural Diagrams:-

The UML's four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. You can view the static parts of a system using one of the following diagrams. Structural diagrams consist of Class Diagram, Object Diagram, Component Diagram, Deployment Diagram.

Behavioral Diagrams :

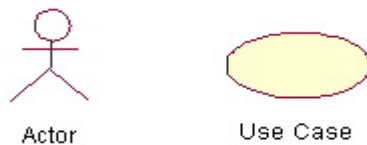
The UML's five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML's behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system.

Behavioral diagrams consist of

Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram

3.3 Use-Case diagram:

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.

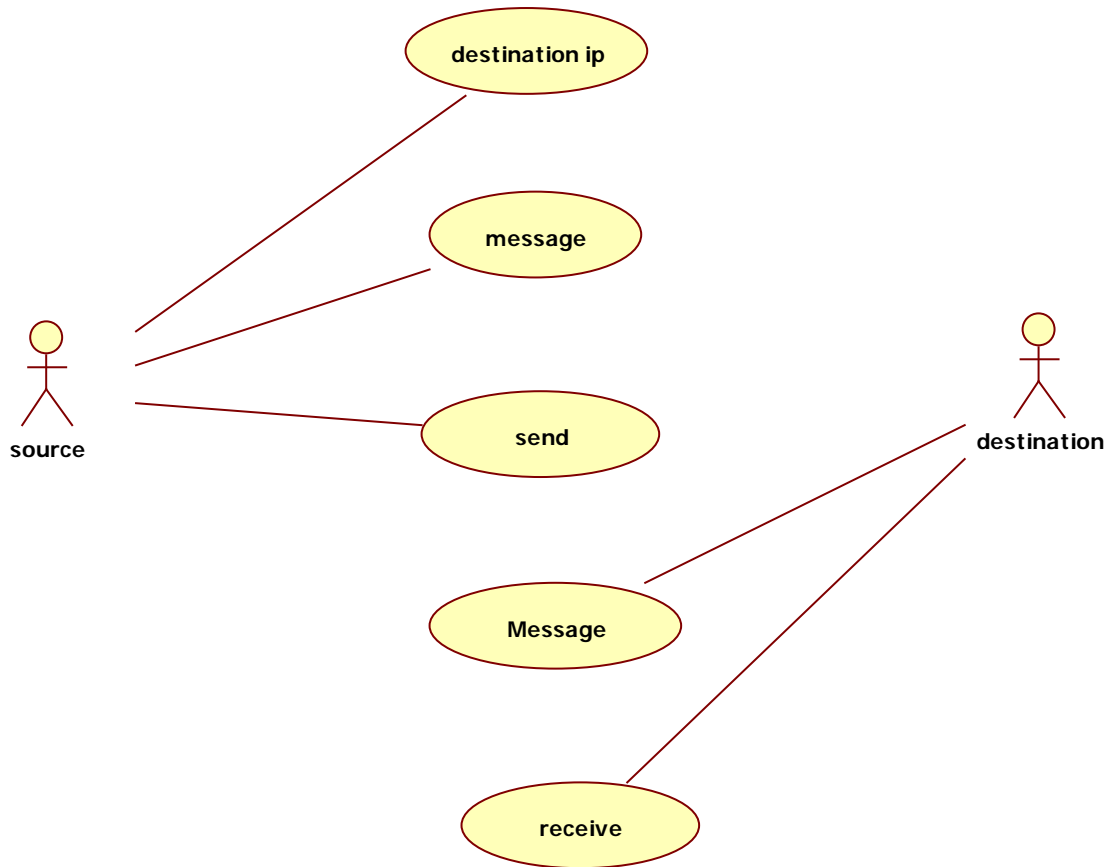


An actor represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

Contents:

- Use cases

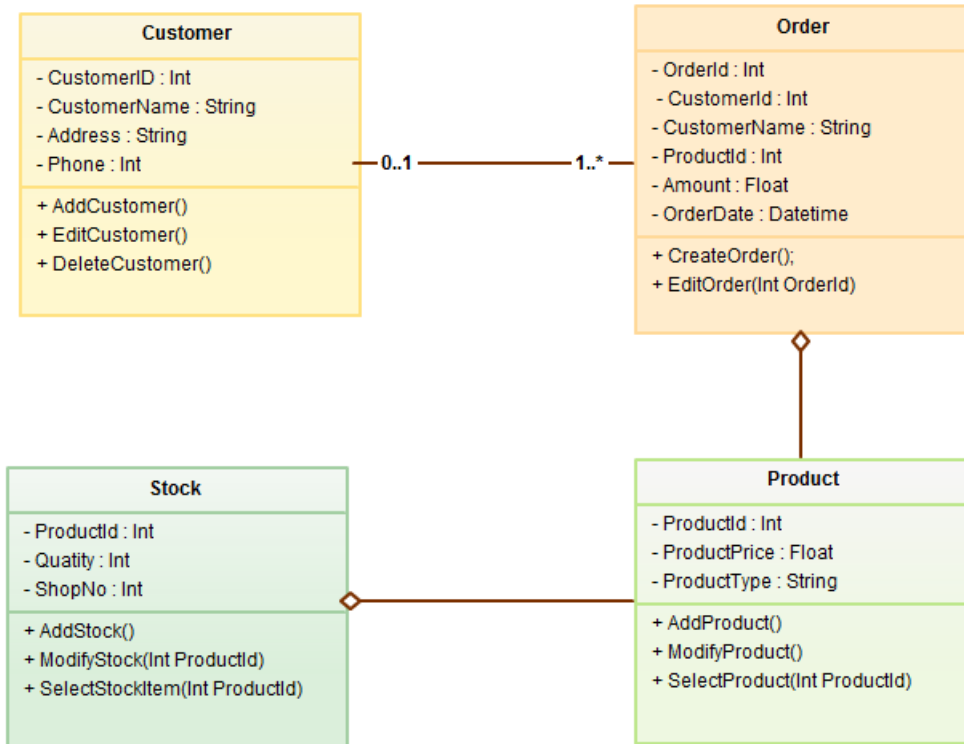
- Actors
- Dependency, Generalization, and association relationships
- System boundary



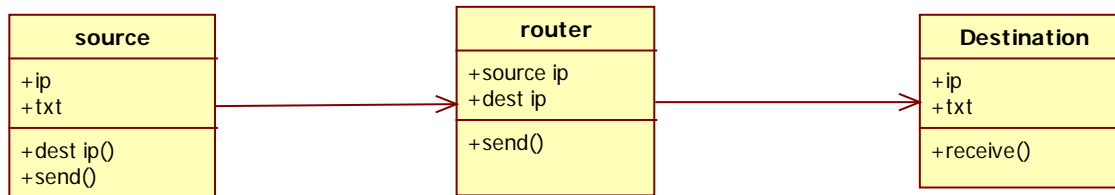
3.4 Class Diagram:

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design. Class diagrams are arguably the most used UML diagram type. It is the main building block of any object oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class. In most modeling tools a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom. In large systems with many classes related classes are grouped together to to create class diagrams. Different relationships between diagrams are show by different types of Arrows. Below is a image of a class diagram. Follow the link for more class diagram examples.

Class Diagram for Order Processing System

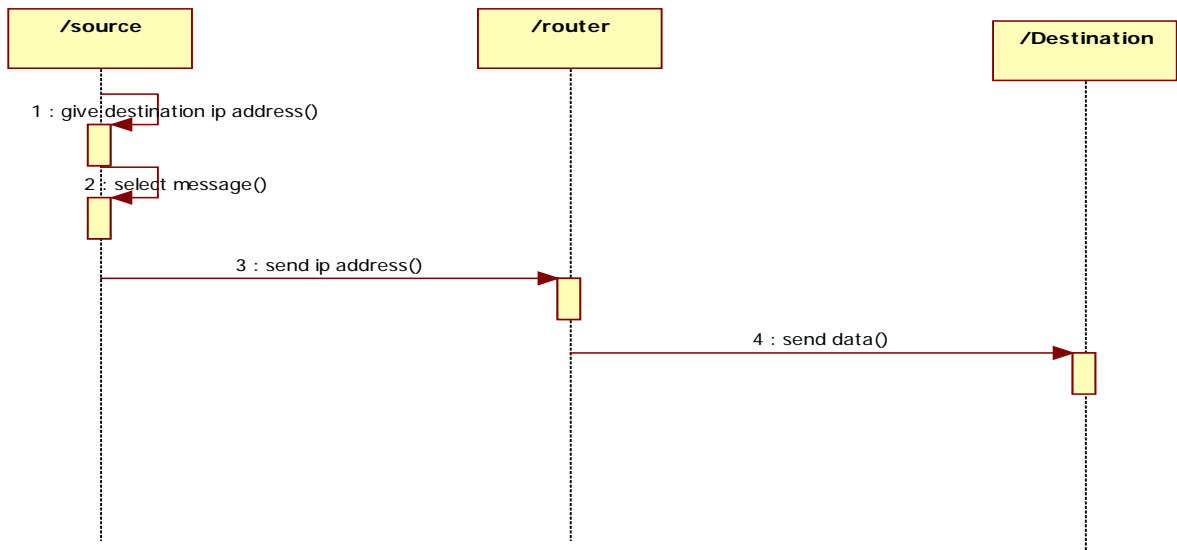
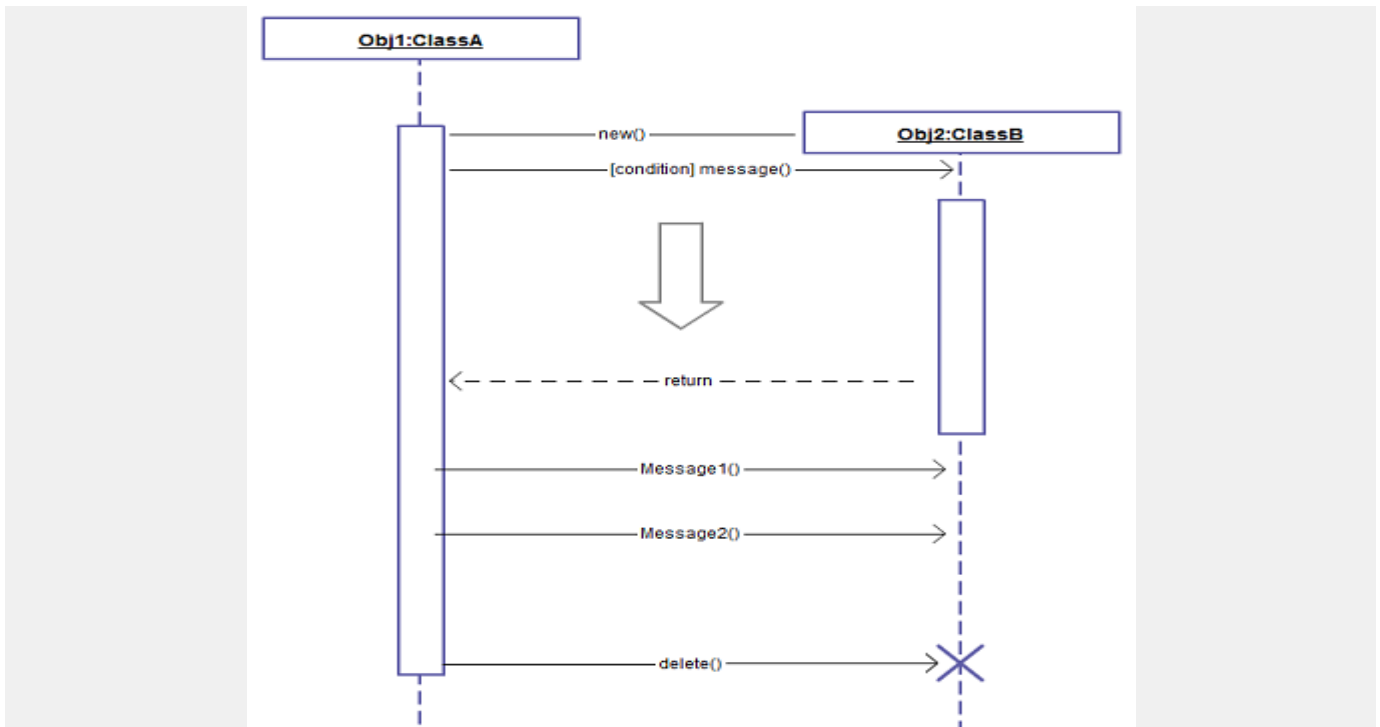


UML Class Diagram with Relationships



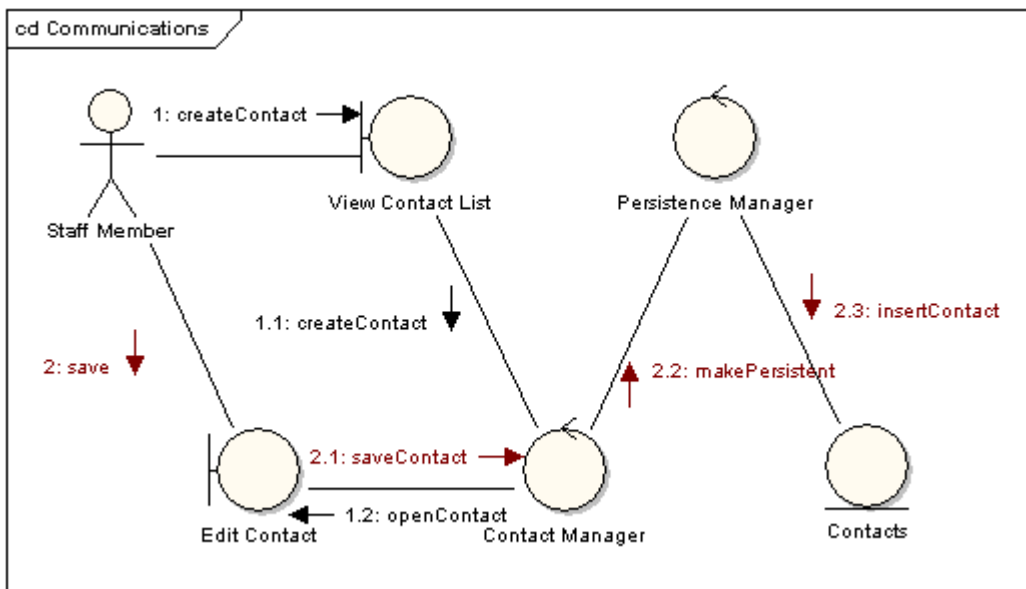
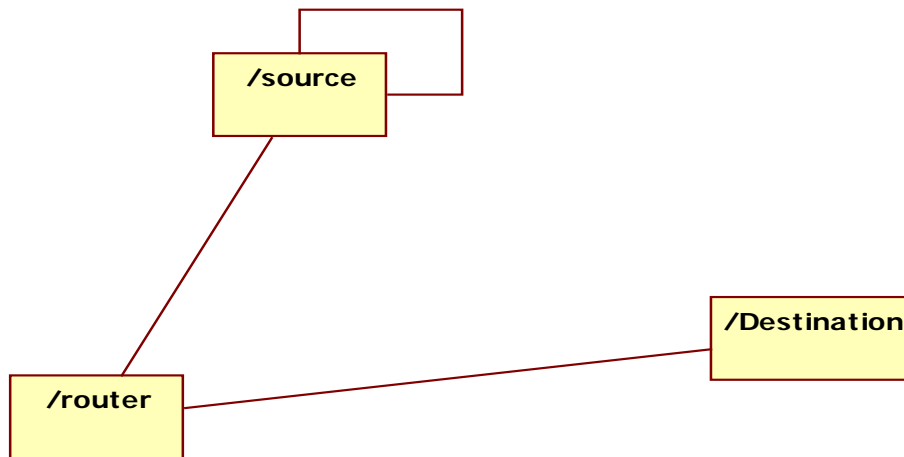
3.5 Sequence Diagram

Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows. This article explains the purpose and the basics of Sequence diagrams.



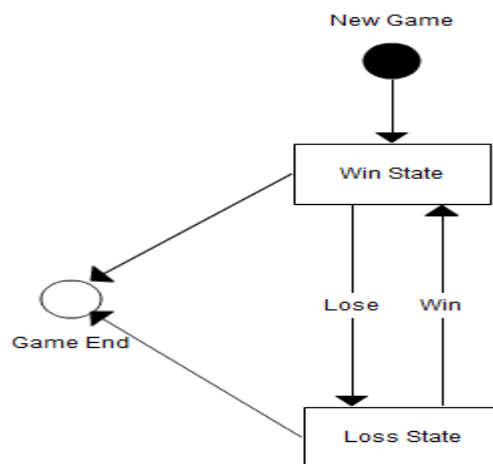
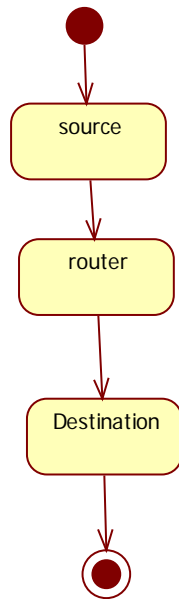
3.6 Collaboration diagram

Communication diagram was called collaboration diagram in UML 1. It is similar to sequence diagrams but the focus is on messages passed between objects. The same information can be represented using a sequence diagram and different objects. [Click here to understand the differences using an example.](#)



3.7 State machine diagrams

State machine diagrams are similar to activity diagrams although notations and usage changes a bit. They are sometimes known as state diagrams or state chart diagrams as well. These are very useful to describe the behavior of objects that act differently according to the state they are at the moment. Below State machine diagram shows the basic states and actions.



State Machine diagram in UML, sometime referred to as State or State chart diagram

3.8 Activity diagram:

Activity Diagram:

Activity diagrams describe the workflow behavior of a system. Activity diagrams are similar to state diagrams because activities are the state of doing something. The diagrams describe the state of activities by showing the sequence of activities performed. Activity diagrams can show activities that are conditional or parallel.

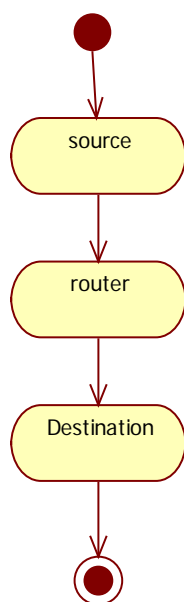
3.8.1 How to Draw: Activity Diagrams

Activity diagrams show the flow of activities through the system. Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities. A fork is used when multiple activities are occurring at the

same time. The diagram below shows a fork after activity1. This indicates that both activity2 and activity3 are occurring at the same time. After activity2 there is a branch. The branch describes what activities will take place based on a set of conditions. All branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch. After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.

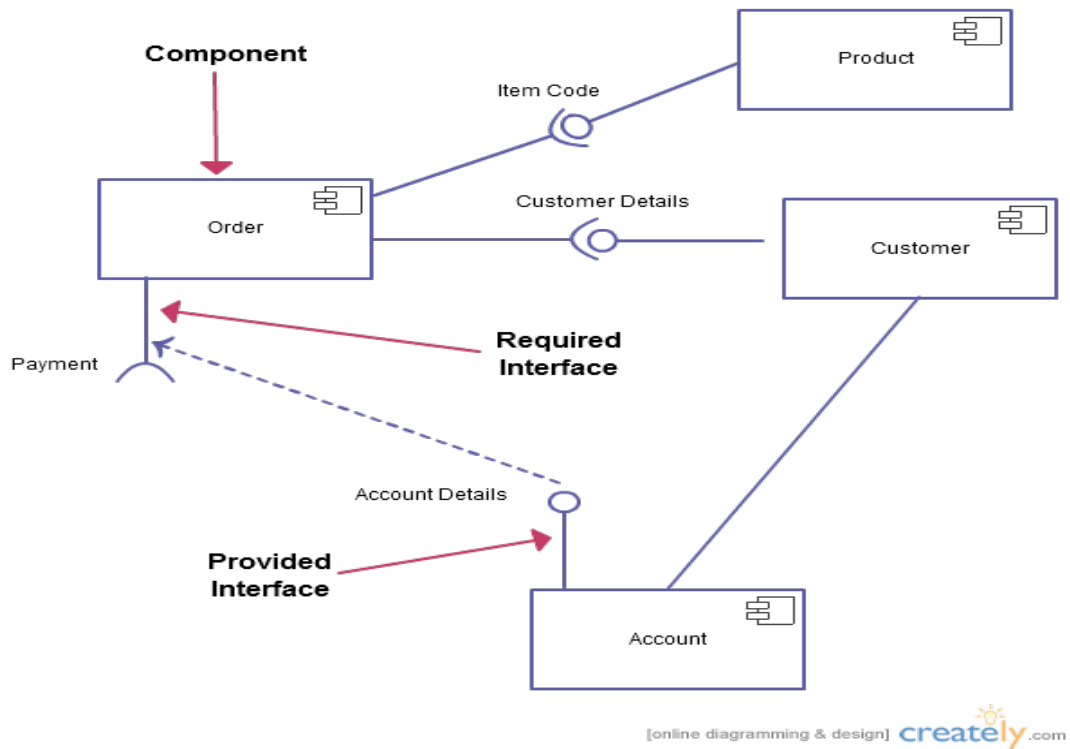
3.8.2 When to Use: Activity Diagrams

Activity diagrams should be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams. The main reason to use activity diagrams is to model the workflow behind the system being designed. Activity Diagrams are also useful for: analyzing a use case by describing what actions need to take place and when they should occur; describing a complicated sequential algorithm; and modeling applications with parallel processes.



3.9 Component diagram

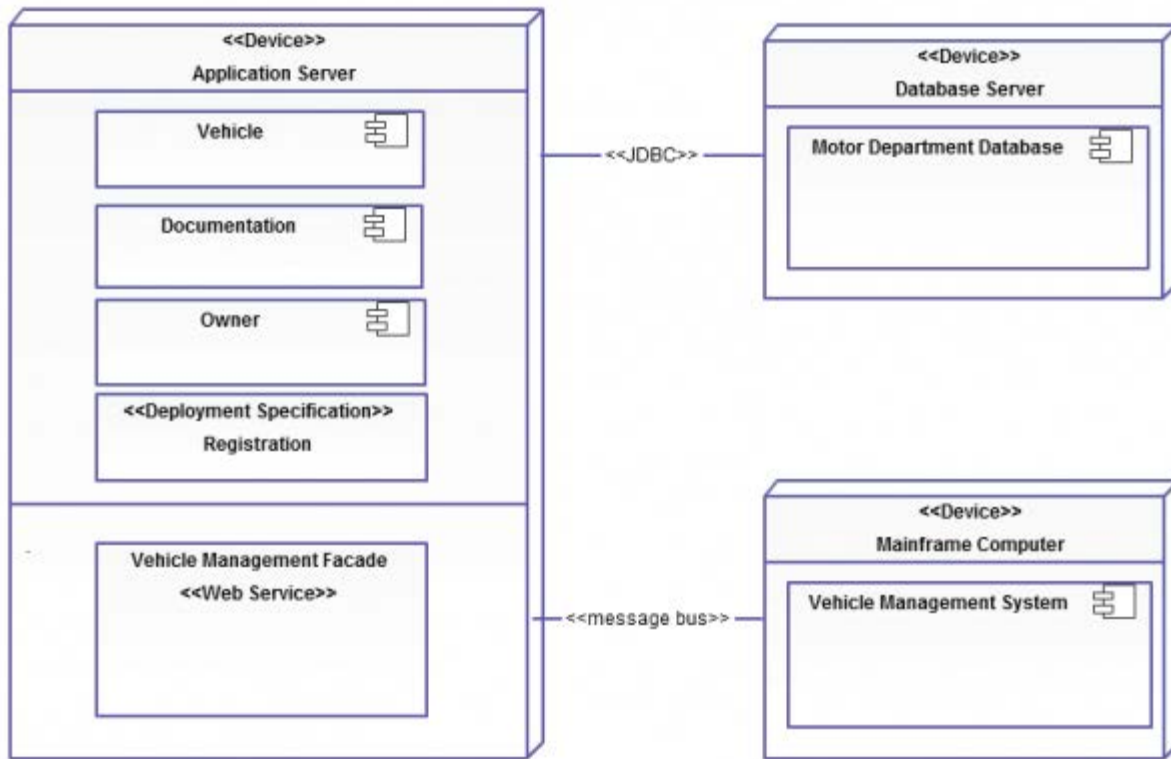
A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems that has many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. Below images shows a component diagram.



3.10 Deployment Diagram

A deployment diagrams shows the hardware of your system and the software in those hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.

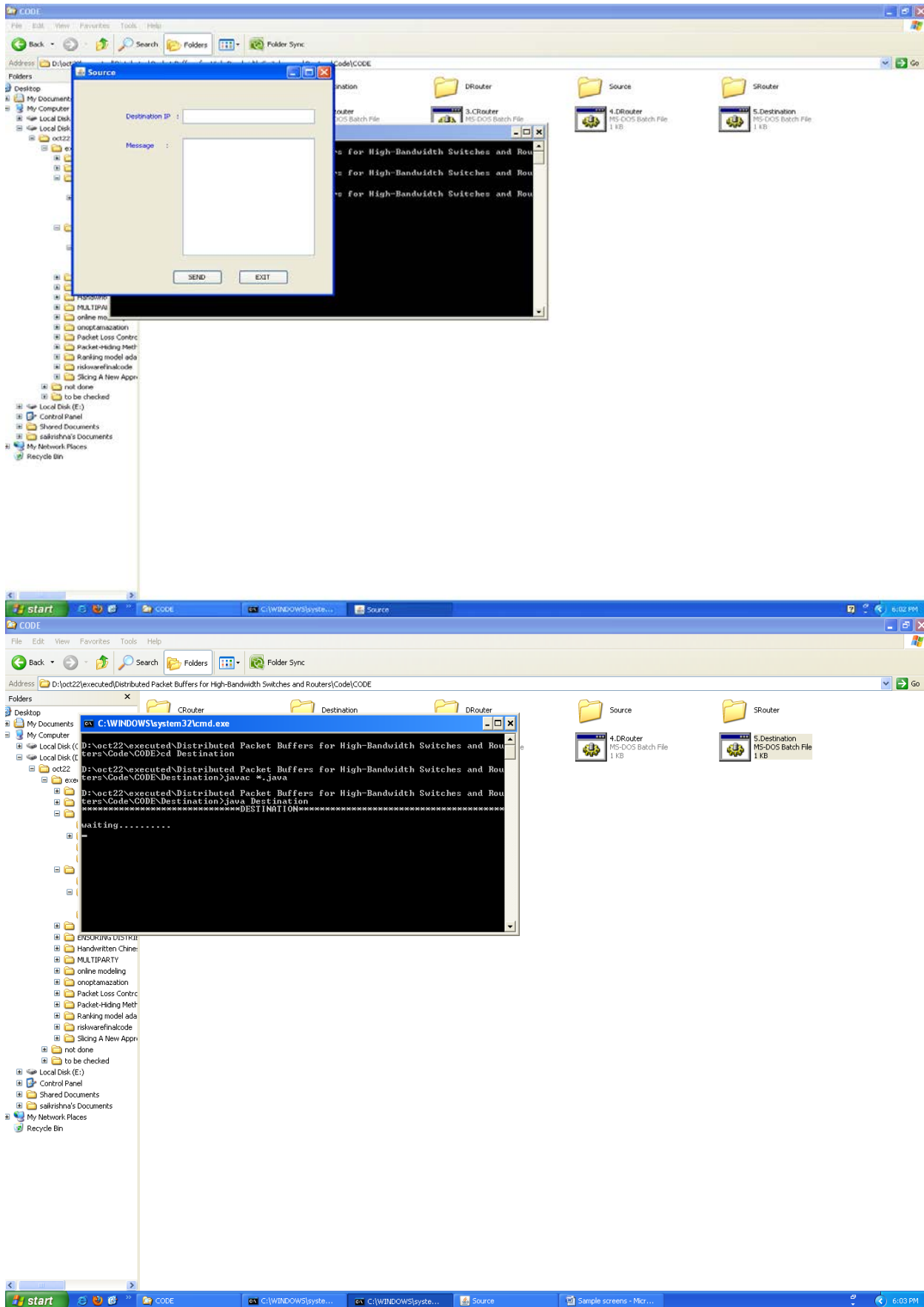


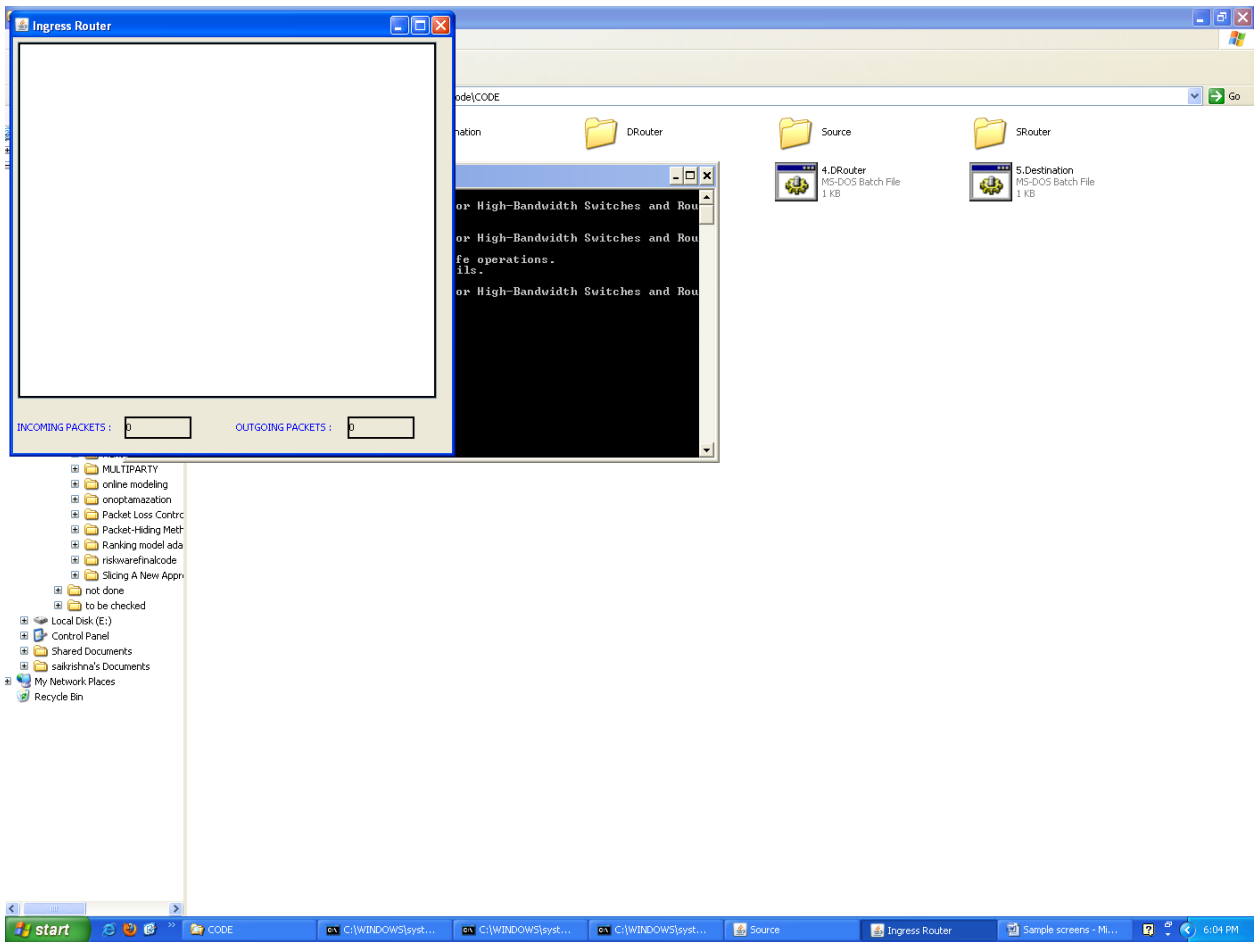


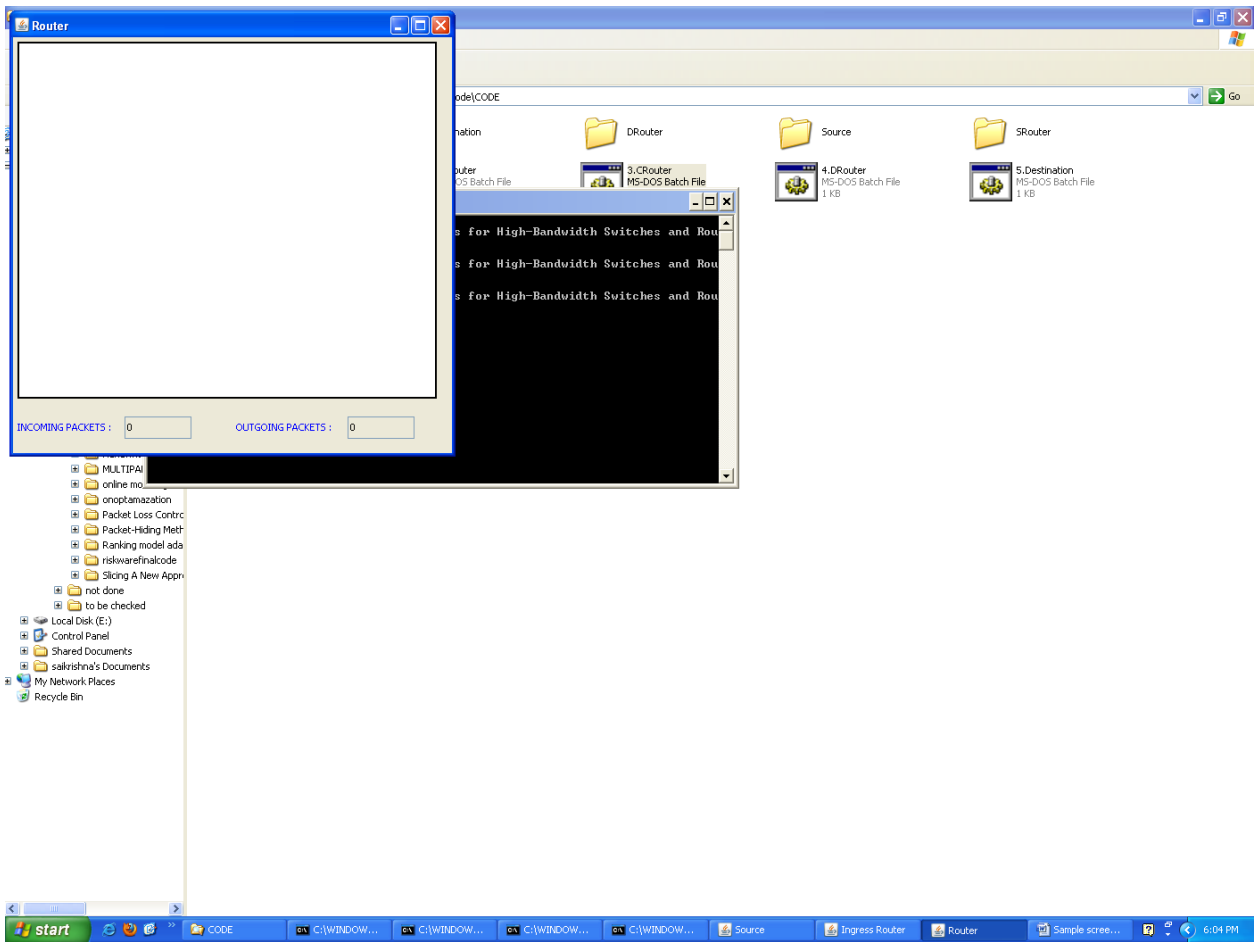
[online diagramming & design] creately.com

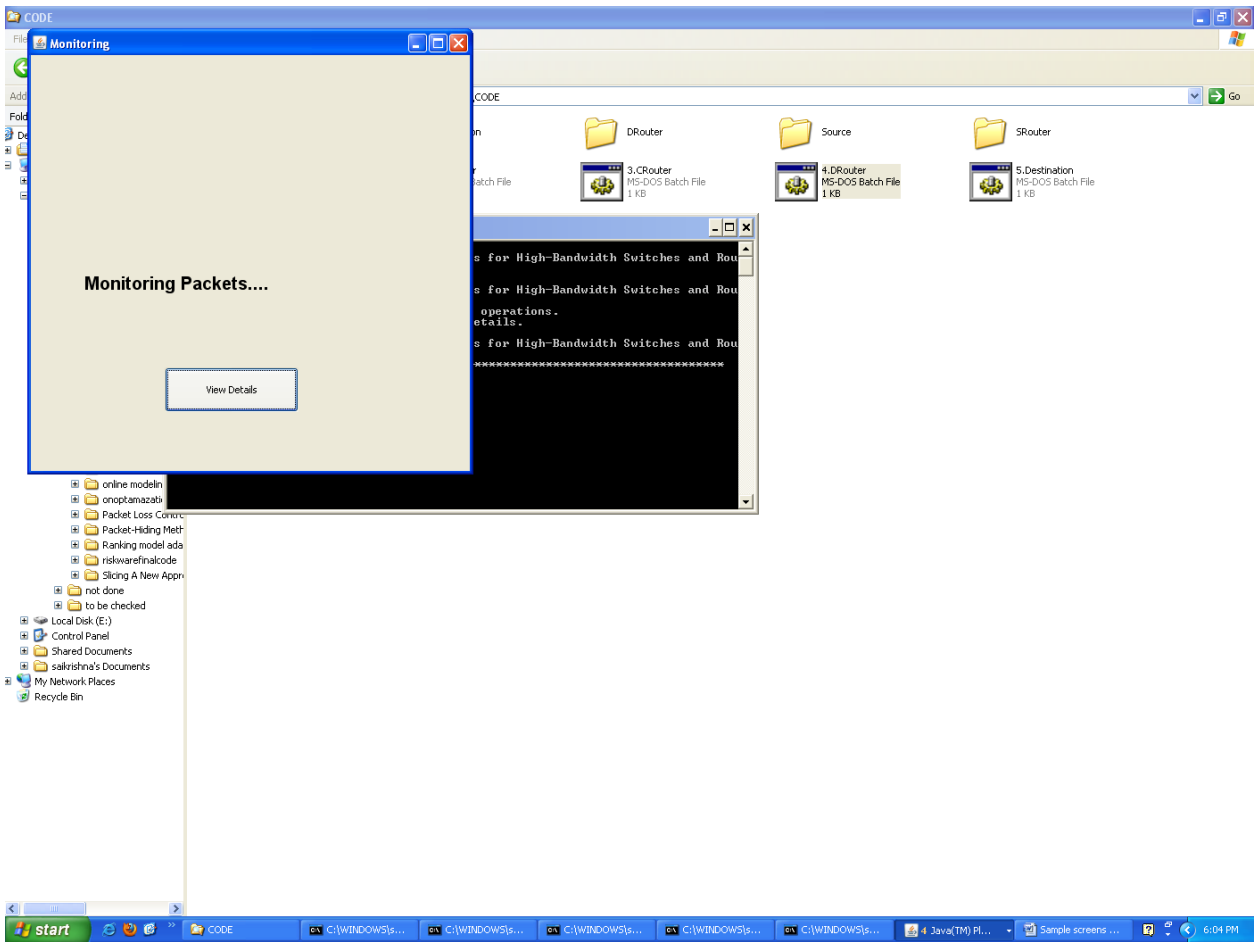
UML Deployment Diagram (Click on the image to use it as a template)

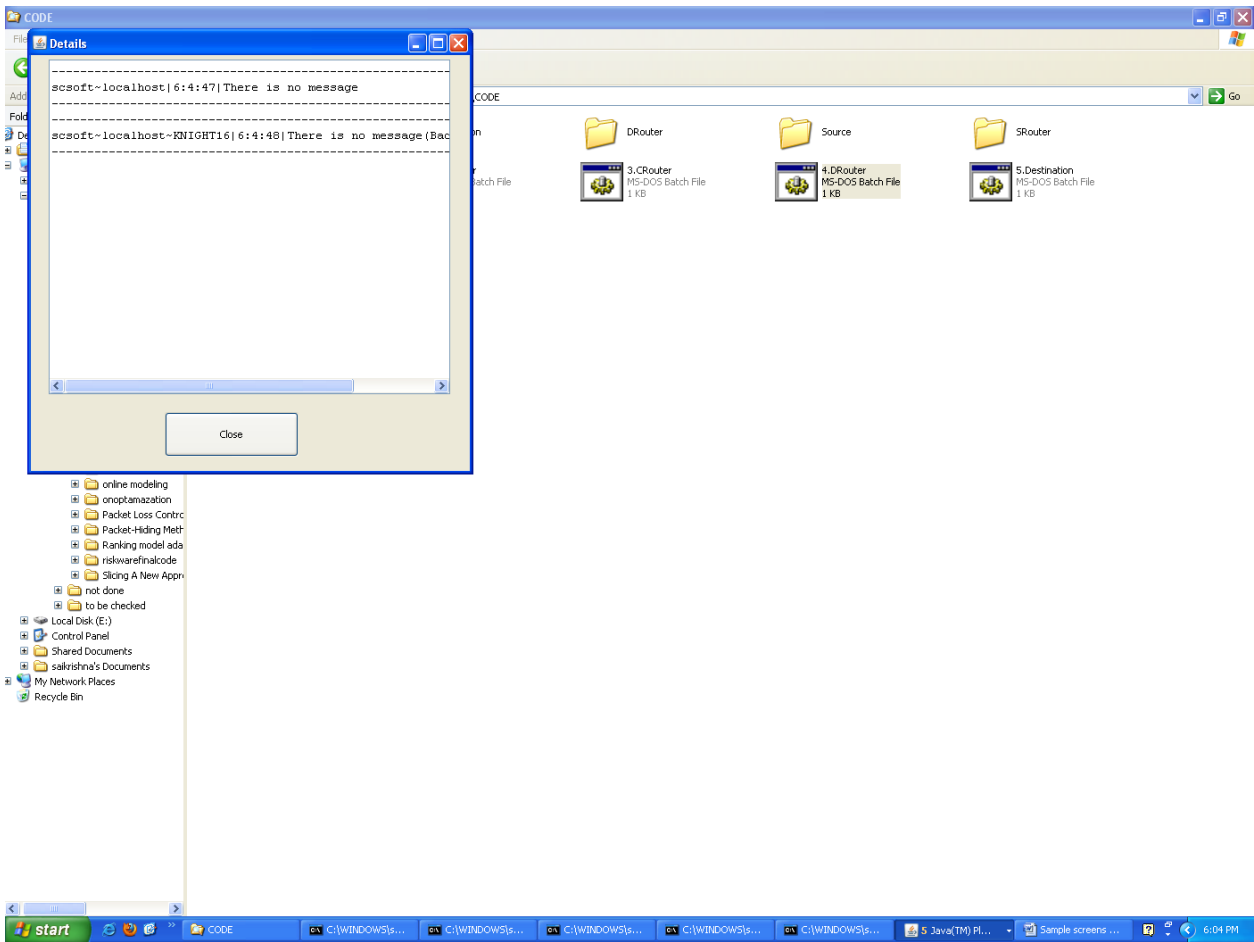
4 Sample Screens:

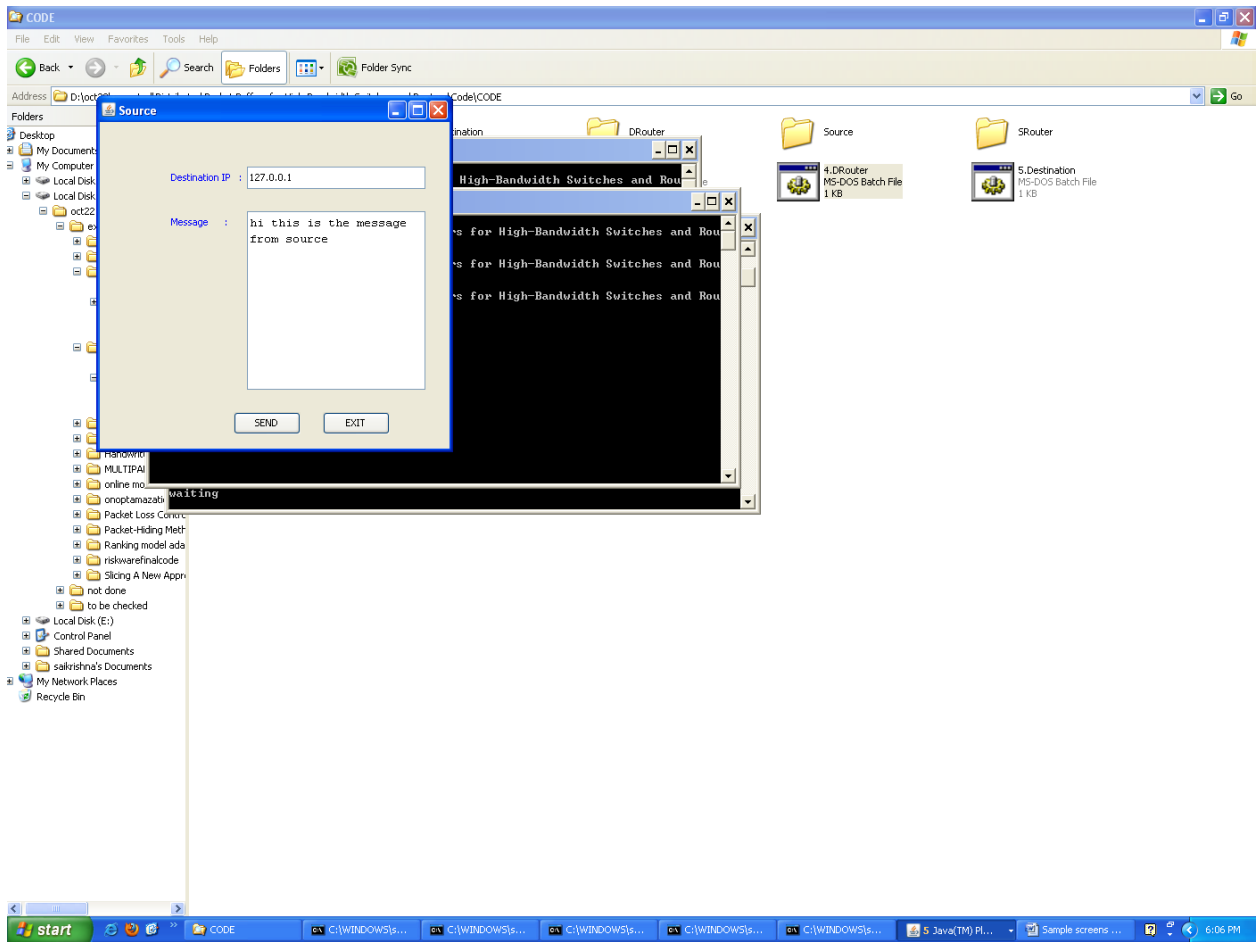


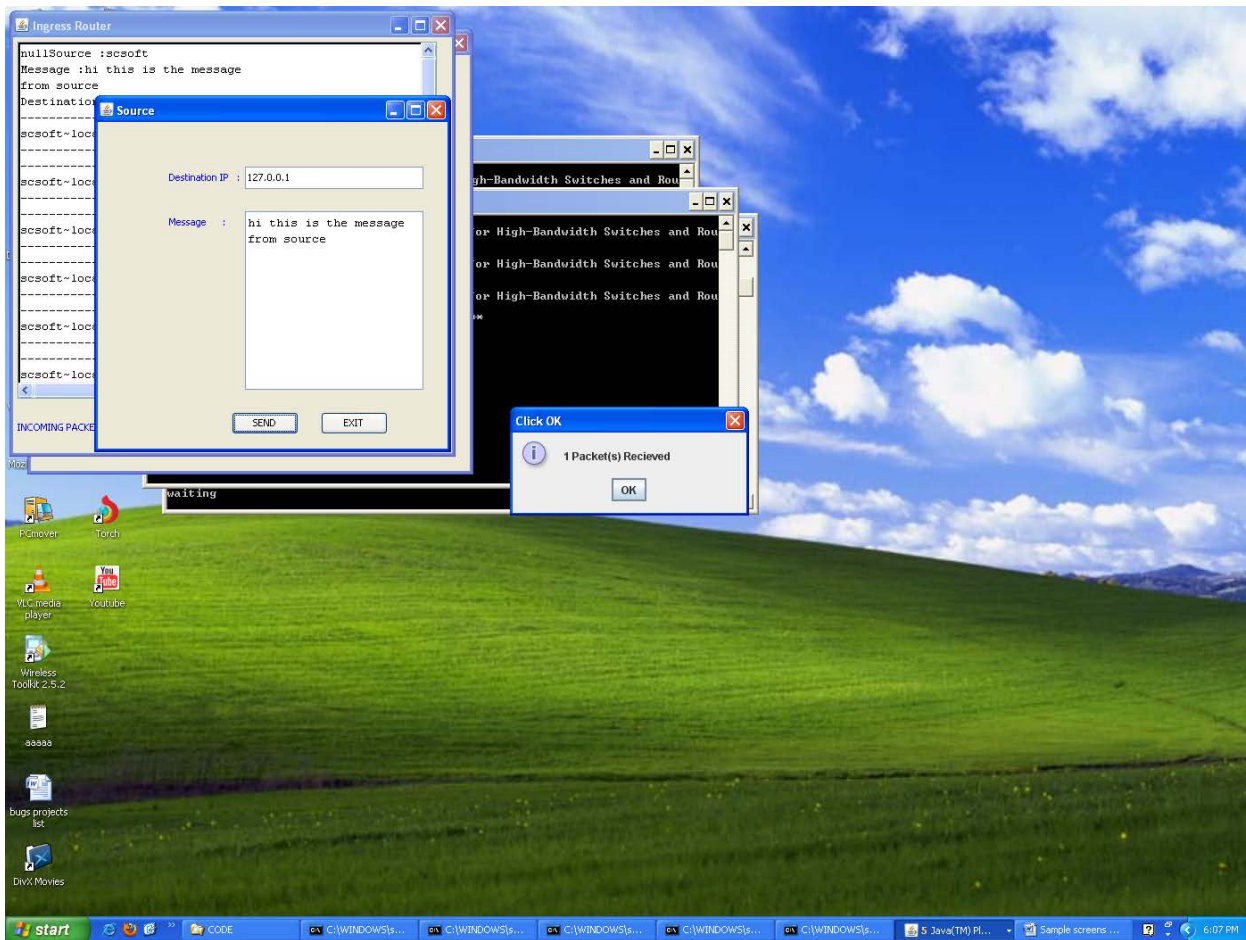


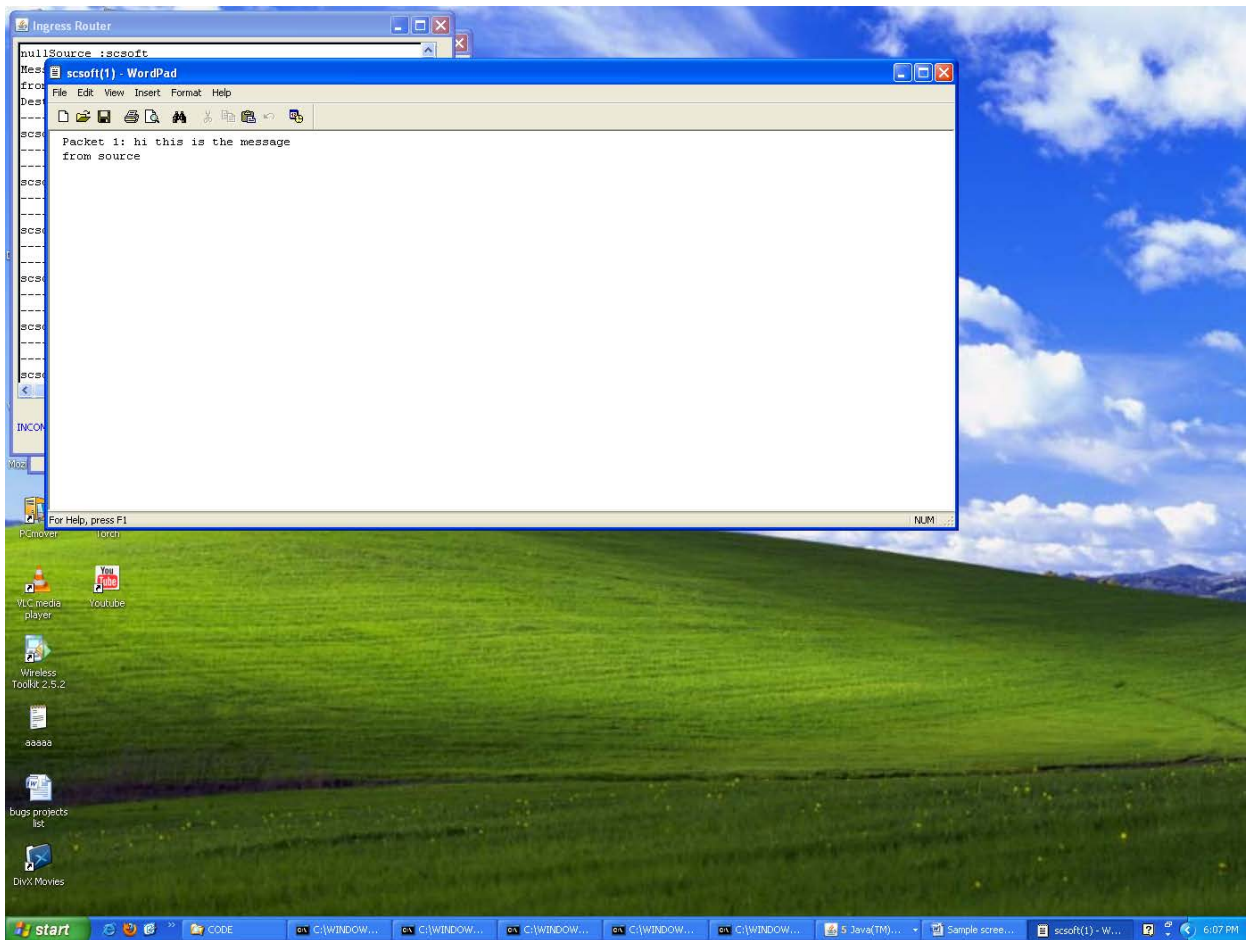


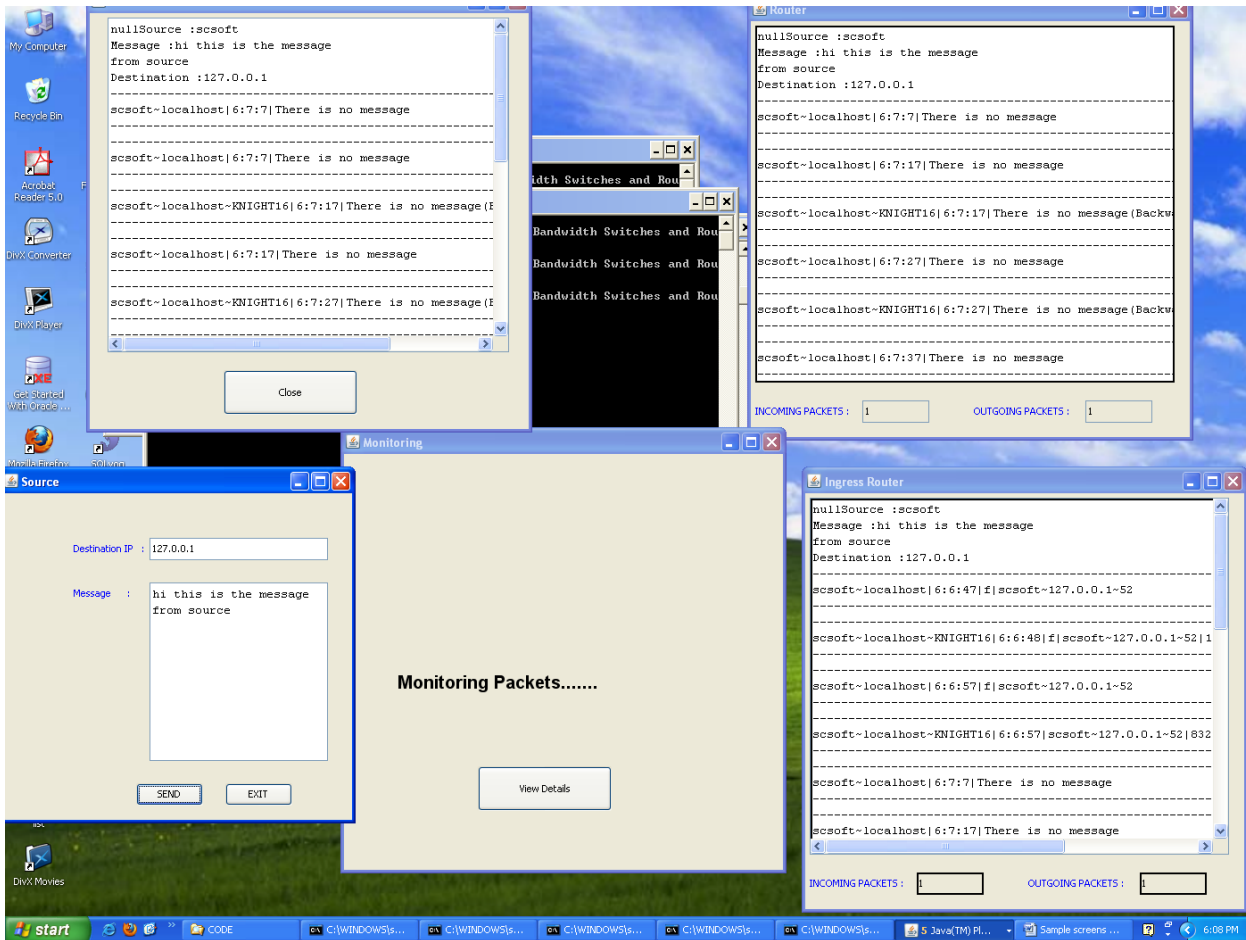












5 Technologies Used:

5.1 Introduction To Java:

Java has been around since 1991, developed by a small team of Sun Microsystems developers in a project originally called the Green project. The intent of the project was to develop a platform-independent software technology that would be used in the consumer electronics industry. The language that the team created was originally called Oak.

The first implementation of Oak was in a PDA-type device called Star Seven (*7) that consisted of the Oak language, an operating system called GreenOS, a user interface, and hardware. The name *7 was derived from the telephone sequence that was used in the team's office and that was dialed in order to answer any ringing telephone from any other phone in the office.

Around the time the First Person project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The World Wide Web, a name applied to the Internet's millions of linked HTML documents was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical Web browser called Mosaic, developed by ncSA. The browser simplified Web browsing by combining text and

graphics into a single interface to eliminate the need for users to learn many confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called Java. Soon after, the Java compiler was rewritten in the Java language from its original C code, thus proving that Java could be used effectively as an application language. Sun introduced Java in May 1995 at the Sun World 95 convention.

Web surfing has become an enormously popular practice among millions of computer users. Until Java, however, the content of information on the Internet has been a bland series of HTML documents. Web users are hungry for applications that are interactive, that users can execute no matter what hardware or software platform they are using, and that travel across heterogeneous networks and do not spread viruses to their computers. Java can create such applications.

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

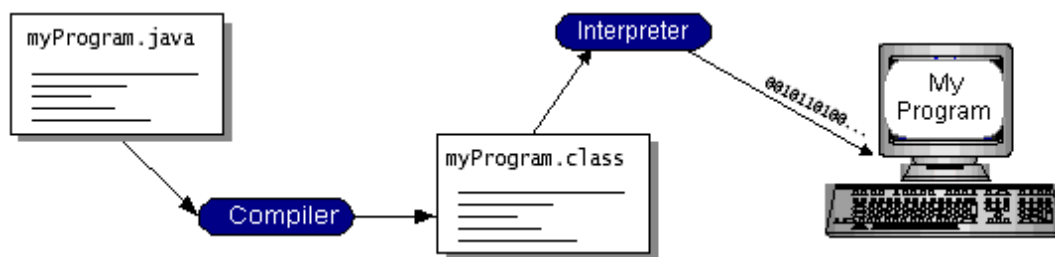


Figure 4.1: Working Of Java

You can think of Java bytecodes as the machine code instructions for the java virtual machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java

bytecodes help make “write once, run anywhere” possible. You can compile your program into bytecodes on any platform that has a Java compiler. The bytecodes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

The Java Platform:

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

The java virtual mechine (Java VM)

The java application programming interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do?, highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

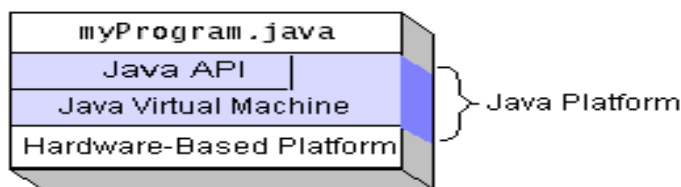


Figure 4.2: The Java Platform

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time bytecode compilers can bring performance close to that of native code without threatening portability.

Working Of Java:

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data and functions. When the interpreter executes a class, it looks for a particular method by the name of **main**, which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv[] of C), and is declared as a static method.

To output text from the program, iexecute the **println** method of **System. out**, which is java’s output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user’s program.

5.2 Swing:

Introduction To Swing:

Swing contains all the components. It’s a big library, but it’s designed to have appropriate complexity for the task at hand – if something is simple, you don’t have to write much code but as you try to do more your code becomes increasingly complex. This means an easy entry point, but you’ve got the power if you need it.

Swing has great depth. This section does not attempt to be comprehensive, but instead introduces the power and simplicity of Swing to get you started using the library. Please be aware that what you see here is intended to be simple. If you need to do more, then Swing can probably give you what you want if you’re willing to do the research by hunting through the online documentation from Sun.

Benefits Of Swing:

Swing components are Beans, so they can be used in any development environment that supports Beans. Swing provides a full set of UI components. For speed, all the components are lightweight and Swing is written entirely in Java for portability.

Swing could be called “orthogonality of use;” that is, once you pick up the general ideas about the library you can apply them everywhere. Primarily because of the Beans naming conventions.

Keyboard navigation is automatic – you can use a Swing application without the mouse, but you don’t have to do any extra programming. Scrolling support is effortless – you simply wrap your component in a JScrollPane as you add it to your form. Other features such as tool tips typically require a single line of code to implement.

Swing also supports something called “pluggable look and feel,” which means that the appearance of the UI can be dynamically changed to suit the expectations of users working under different platforms and operating systems. It’s even possible to invent your own look and feel.

Domain Description:

Data mining involves the use of sophisticated data analysis tools to discover previously unknown, valid patterns and relationships in large data sets. These tools can include statistical models, mathematical algorithms, and machine learning methods (algorithms that improve their performance automatically through experience, such as neural networks or decision trees). Consequently, data mining consists of more than collecting and managing data, it also includes analysis and prediction.

Data mining can be performed on data represented in quantitative, textual, or multimedia forms. Data mining applications can use a variety of parameters to examine the data. They include association (patterns where one event is connected to another event, such as purchasing a pen and purchasing paper), sequence or path analysis (patterns where one event leads to another event, such as the birth of a child and purchasing diapers), classification (identification of new patterns, such as coincidences between duct tape purchases and plastic sheeting purchases), clustering (finding and visually documenting groups of previously unknown facts, such as geographic location and brand preferences), and forecasting

(discovering patterns from which one can make reasonable predictions regarding future activities, such as the prediction that people who join an athletic club may take exercise classes)

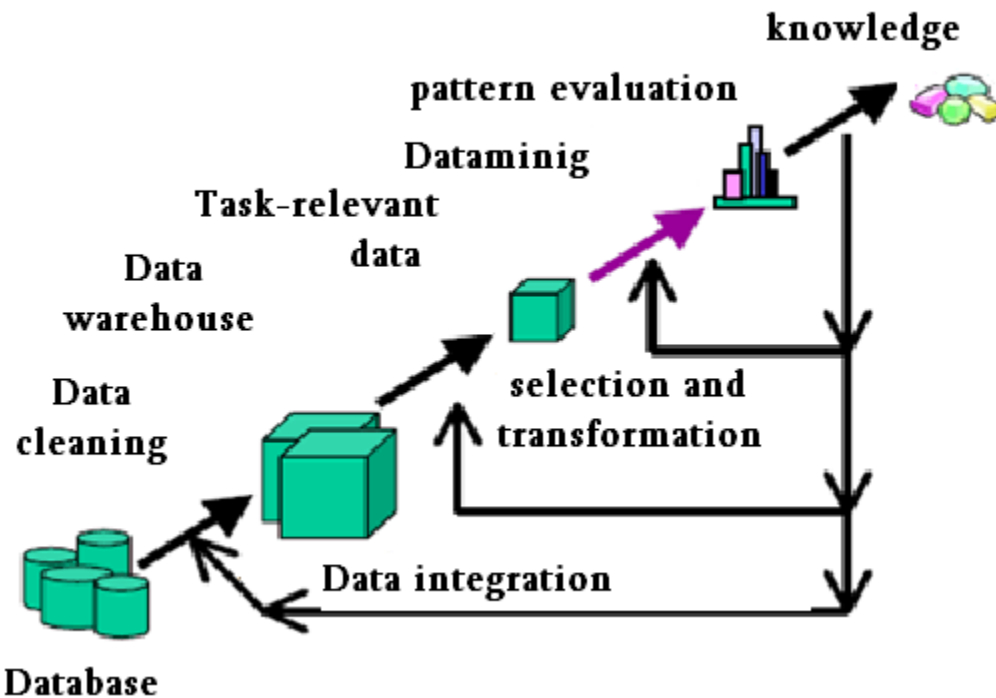


Figure 4.3 knowledge discovery process

Data Mining Uses:

Data mining is used for a variety of purposes in both the private and public sectors.

- Industries such as banking, insurance, medicine, and retailing commonly use data mining to reduce costs, enhance research, and increase sales. For example, the insurance and banking industries use data mining applications to detect fraud and assist in risk assessment (e.g., credit scoring).
- Using customer data collected over several years, companies can develop models that predict whether a customer is a good credit risk, or whether an accident claim may be fraudulent and should be investigated more closely.
- The medical community sometimes uses data mining to help predict the effectiveness of a procedure or medicine.
- Pharmaceutical firms use data mining of chemical compounds and genetic material to help guide research on new treatments for diseases. Retailers can use information collected through affinity programs (e.g., shoppers' club cards, frequent flyer points, contests) to assess the effectiveness of product selection and placement decisions, coupon offers, and which products are often purchased together.

6 *Sample Code:*

//CRouter.java

```
import javax.swing.*;
import java.net.*;
import java.io.*;
import java.awt.*;
import javax.swing.UIManager;

class Router_Frame
{
    JFrame Ing_fra;
    Container cp1;
    JTextArea Ing_data;
    JTextField InTxt,OutTxt;
    JLabel InPac,OutPac;
    static String instring="";
    static int length=0;
    static int length1=0;
    static String s="";
    static String s1="";
    String fin="";
    static String dest="";
    static String sou="";
    static int inp=0,outp=0;
    JScrollPane jsp;
    ObjectInputStream in;
    OutputStream out;
    int readcnt=0;
    boolean header=false;
    static String text;
    static String text1;
    char chstr[]=new char[512];
    Socket soc;
```

```
String inf="com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
```

```
Router_Frame()
```

```
{
```

```
}
```

```
public void dis_ing_fra()
```

```
{
```

```
    Ing_fra=new JFrame("Router");
```

```
    cp1=Ing_fra.getContentPane();
```

```
    cp1.setLayout(null);
```

```
    Ing_data=new JTextArea();
```

```
    InPac=new JLabel("INCOMING PACKETS :");
```

```
    OutPac=new JLabel("OUTGOING PACKETS :");
```

```
    InTxt=new JTextField();
```

```
    OutTxt=new JTextField();
```

```
    InTxt.setText("0");
```

```
    OutTxt.setText("0");
```

```
    InPac.setForeground(Color.blue);
```

```
    OutPac.setForeground(Color.blue);
```

```
    Ing_data.setEditable(false);
```

```
    InTxt.setEditable(false);
```

```
    OutTxt.setEditable(false);
```

```
    Ing_data.setBorder(BorderFactory.createEtchedBorder(Color.black,Color.black));
```

```
    //InTxt.setBorder(BorderFactory.createEtchedBorder(Color.black,Color.black));
```

```
    //OutTxt.setBorder(BorderFactory.createEtchedBorder(Color.black,Color.black));
```

```
    //jsp=new JScrollPane(Ing_data);
```

```
    //cp1.add(jsp);
```

```
    cp1.add(Ing_data);
```

```
    cp1.add(InPac);
```

```
    cp1.add(InTxt);
```

```
    cp1.add(OutPac);
```

```
    cp1.add(OutTxt);
```

```

//jsp.setBounds(5,5,470,400);
Ing_data.setBounds(5,5,470,400);
InPac.setBounds(5,425,130,25);
InTxt.setBounds(125,425,75,25);
OutPac.setBounds(250,425,150,25);
OutTxt.setBounds(375,425,75,25);
Ing_fra.setBounds(5,5,500,500);
Ing_fra.setVisible(true);
Ing_fra.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
try
{
    UIManager.setLookAndFeel(inf);
    SwingUtilities.updateComponentTreeUI(Ing_fra);
}
catch(Exception e)
{
    //e.printStackTrace();
}
}

public void dis_ing_data()
{
    try
    {
        System.out.println("Waiting...");
        ServerSocket ss=new ServerSocket(7799);
        while(true)
        {
            soc=ss.accept();
            System.out.println("connected...");
            display();
        }
    }
    catch(IOException e)

```

```

        {
            e.printStackTrace();
        }
    }

    public void add()
    {
        text+="Source :"+sou+"\nMessage :"+s.trim()+"\nDestination :"+dest+"\n";
        Ing_data.setText(text);
    }

    public void display()
    {
        try
        {
            ObjectInputStream ois=new ObjectInputStream(soc.getInputStream());
            instring=(String) ois.readObject();
            length=instring.length();
            int fin=instring.indexOf('|');
            if(fin>0)
            {
                int last=instring.lastIndexOf('|');
                header=true;
                Ing_data.append("-----\n");
                Ing_data.append(instring+"\n");
                Ing_data.append("-----
"+ "\n");

                Socket soc1=new Socket("localhost",7700);
                ObjectOutputStream oos=new
ObjectOutputStream(soc1.getOutputStream());
                oos.writeObject(instring);

            }
            else

```



```

        {
            header=false;
        }
    if(length>0 && (!header))
    {
        Socket soc1=new Socket("localhost",7700);
        ObjectOutputStream oos=new
ObjectOutputStream(soc1.getOutputStream());
        oos.writeObject(instring);
        oos.flush();
        outp++;
        OutTxt.setText(outp+"");
    for(int l=0;l<length;l++)
        {
            if((instring.charAt(l)=='\n')
                {
                    dest=instring.substring(0,l);
                    s1=instring.substring(l+1,length);
                    l=length+1;
                    length1=s1.length();
                }
            }
        for(int l=0;l<length1;l++)
        {
            if((s1.charAt(l)=='\n')
                {
                    sou=s1.substring(0,l);
                    if((s1.substring((s1.length()-4),s1.length()).equals("null"))
                        {
                            s=s1.substring(l+1,length1-4);
                        }
                    else
                        {
                            s=s1.substring(l+1,length1);

```

```

        }
        l=length1+1;
    }
}
    add();
    inp++;
    InTxt.setText(inp+"");
}
}
catch(ClassNotFoundException e)
{
    e.printStackTrace();
}
catch(IOException e1)
{
    e1.printStackTrace();
}
}
}

class CRouter
{
    public static void main(String args[])
    {
        Router_Frame obj=new Router_Frame();
        obj.dis_ing_fra();
        Back b=new Back(obj);
        obj.dis_ing_data();
    }
}

//Back.java
import java.io.*;
import java.net.*;

```

```
class Back extends Thread
```

```
{
```

```
    Router_Frame rf;
```

```
    ServerSocket ss;
```

```
    Back(Router_Frame obj1)
```

```
    {
```

```
        rf=obj1;
```

```
        start();
```

```
    }
```

```
    public void run()
```

```
    {
```

```
        try
```

```
        {
```

```
            ss=new ServerSocket(888);
```

```
        }
```

```
        catch(Exception e)
```

```
        {
```

```
            e.printStackTrace();
```

```
        }
```

```
        while(true)
```

```
        {
```

```
            try
```

```
            {
```

```
                Socket soc=ss.accept();
```

```
                ObjectInputStream ois=new ObjectInputStream(soc.getInputStream());
```

```
                String instring=(String) ois.readObject();
```

```
                rf.Ing_data.append("-----\n");
```

```
                rf.Ing_data.append(instring+"\n");
```

```
                rf.Ing_data.append("-----"+"n");
```

```
                Socket soc1=new Socket("localhost",777);
```

```

        ObjectOutputStream oos=new
ObjectOutputStream(soc1.getOutputStream());
        oos.writeObject(instring);
        Thread.sleep(1000);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    }
}

```

//Destination.java

```

import java.net.*;
import java.io.*;
import java.awt.*;
import javax.swing.*;

```

class Destination

```

{
    public static void main(String args[]) throws IOException
    {

```

```

System.out.println("*****DESTINATION*****
*****");

```

```

        String instring=" ";

```

```

        String s=" ";

```

```

        int length=0;

```

```

        String dest=" ";

```

```

        int coun=0,i=1;

```

```

            int count=0;

```

```

            int initial=0;

```

```

            FileWriter fos;

```

```

        try

```

```

{
    ServerSocket sock1 = new ServerSocket(7718);

    while(true)
    {
        System.out.println("waiting.....");
        Socket insocket1 = sock1.accept();
        System.out.println("connected sucessfully.....");
        try
        {
            ObjectInputStream ois=new ObjectInputStream(insocket1.getInputStream());
            instring=(String) ois.readObject();
                }
            catch(ClassNotFoundException e)
            {
                e.printStackTrace();
            }
            length=instring.length();
            int st=instring.indexOf(' ');
            int end=instring.lastIndexOf(' ');
            dest=instring.substring(st+1,end);
            String dest1=dest+"("+"i+"";
                if((instring.substring((instring.length()-
4),instring.length())).equals("null"))
                {
                    coun=1;
                    count++;
                    s=instring.substring(end+1,length-4);
                }
                else
                {
                    s=instring.substring(end+1,length);
                    count++;
                }

```

```

        s="Packet "+count+": "+s;
//byte data[]=s.getBytes();
if(initial==0)
{
fos = new FileWriter(dest1+".txt",false);
}
else
{
        fos = new FileWriter(dest1+".txt",true);
}
BufferedWriter bw = new BufferedWriter(fos);
bw.write(s);
bw.newLine();
bw.newLine();
bw.newLine();
bw.close();
fos.close();

        initial++;

if(coun==1)
{ System.out.println("finished");

        JOptionPane.showMessageDialog((Component) null,count+"
Packet(s) Recieved ", "Click OK",JOptionPane.INFORMATION_MESSAGE);

        Runtime r = Runtime.getRuntime();
        Process p = null;
p = r.exec("C:\\Program Files\\Windows NT\\Accessories\\wordpad.exe"+"
"+dest1+".txt");

        coun=0;
        i++;
        initial=0;
        count=0;

}
}
}

```

```

catch(UnknownHostException e)
{
    e.printStackTrace();
}
}
}

```

//Drouter.java

```

import java.net.*;
import java.io.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.util.*;
import javax.swing.UIManager;

```

```

class Get implements Runnable

```

```

{
    Tim1 ti=new Tim1();
    Thread t1;
    static String instring=" ";
    static int length=0;
    static int length1=0;
    static String s=" ";
    static String s1=" ";
    static String dest=" ";
    static String sou=" ";
    boolean header=false;
    long tim;
    long avg_rate;
    Socket soc;
    Vector sour=new Vector();
    Vector des=new Vector();
    Vector leng=new Vector();
    Vector virtual_time=new Vector();

```

```
String instring1="";
static TSW ts;
static JFrame jf1;
static JTextArea jta;
static JButton jb1;
static JLabel jl1;
static String text;
static Container cp1;
String find,sub1;
int fir,sec;
static long CT;
int last,check;
```

```
Get()
```

```
{
}
```

```
Get(Socket insocket1)
```

```
{
    t1=new Thread(this,"Get");
    soc=insocket1;
    t1.start();
}
```

```
public void dis_Sec_Frame()
```

```
{
    jf1 =new JFrame("Details");
    jb1=new JButton("Close");
    jta=new JTextArea();
    jta.setEditable(false);
    JScrollPane jsp=new JScrollPane(jta);
    cp1=jf1.getContentPane();
    cp1.setLayout(null);
    cp1.add(jb1);
```



```

        jb1.addActionListener(new AL());
        cp1.add(jsp);
        jsp.setBounds(20,5,450,375);
        jb1.setBounds(150,400,150,50);
        jf1.setBounds(25,25,500,500);
    }
class AL implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource()==jb1)
        {
            jf1.setVisible(false);
        }
    }
}

public void show()
{
    jf1.setVisible(true);
}

public void add()
{
    text+="Source :"+sou+"\nMessage :"+s.trim()+"\nDestination :"+dest+"\n";
    jta.setText(text);
}

public void run()
{
    try
    {
        ObjectInputStream ois=new ObjectInputStream(soc.getInputStream());
        instring=(String) ois.readObject();
    }
}

```

```

length=instring.length();
header=true;
CT=ti.calculateTime();
int fin=instring.indexOf('|');
    if(fin>0)
        {
            int last=instring.lastIndexOf('|');
            fir=instring.indexOf('|');
            sec=instring.indexOf('|',fir+1);
            int thi=instring.indexOf('|',sec+1);
            if(thi<0)
                {
                    Thread.sleep(10000);
                    jta.append("-----
\n");
                    jta.append(instring+"\n");
                    jta.append("-----
"+"
");
                    Back b=new Back(instring,fir,sec,this);
                }
            if(thi>0)
                {
                    Thread.sleep(10000);
                    jta.append("-----
\n");
                    jta.append(instring);
                    jta.append("\n-----
"+"
");
                    instringl=instring.substring(0,sec)+"|";
                    long
LT=Long.parseLong(instring.substring(last+1,instring.length()));
                    tim=(CT-LT);
                    ts=new TSW(tim);
                    avg_rate=length/tim;

```

```

find=instring.substring(sec,thi+1);
if(find.equals("|f"))
{
    String sub=instring.substring(thi+1,instring.length());
    do
    {
        int beg=sub.indexOf('~');
        if(beg>0)
        {
            int first=sub.indexOf('~',beg+1);
            check=sub.indexOf('|',first+1);
            if(check<0)
            {
                last=sub.length();
            }
            else
            {
                last=check;
            }
            sour.add(sub.substring(0,beg));
            des.add(sub.substring(beg+1,first));
            leng.add(sub.substring(first+1,last));

instring1+=sub.substring(0,beg)+"~"+sub.substring(beg+1,first)+"~"+sub.substring(first+1,last)+"
|";

            if(check>0)
            {
                sub=sub.substring(check+1,sub.length());
            }
            if(!leng.isEmpty())
            {
                long tim1=(
Long.parseLong((leng.get(leng.size()-1)).toString()))*tim;

```

```
virtual_time.add(tim1+"");
```

```
instring1+=virtual_time.get(virtual_time.size()-1)+"|";  
    }
```

```
    }
```

```
    else
```

```
    {
```

```
        check=-1;
```

```
    }
```

```
    }while(check>0);
```

```
    Back b=new Back(instring1,fir,sec,this);
```

```
    }
```

```
    else
```

```
    {
```

```
    }
```

```
    }
```

```
    }
```

```
    else
```

```
    {
```

```
        header=false;
```

```
    }
```

```
    if(length>0 && (!header))
```

```
    {
```

```
        try
```

```
        {
```

```
            ts.TimeSlide(length,CT);
```

```
        }
```

```
        catch(Exception e){e.printStackTrace();}
```

```
        for(int l=0;l<length;l++)
```

```
        {
```

```
            if((instring.charAt(l))=='' )
```

```
            {
```

```

        dest=instring.substring(0,l);
        s1=instring.substring(l+1,length);
        l=length+1;
        length1=s1.length();
    }
}
for(int l=0;l<length1;l++)
{
    if((s1.charAt(l))=='')
    {
        sou=s1.substring(0,l);
        if((s1.substring((s1.length()-4),s1.length())).equals("null"))
        {
            s=s1.substring(l+1,length1-4);
        }
        else
        {
            s=s1.substring(l+1,length1);
        }
        l=length1+1;
    }
}
add();
Socket soc1=new Socket(dest,7718);
ObjectOutputStream oos=new ObjectOutputStream(soc1.getOutputStream());
oos.writeObject(instring);
oos.flush();
}
}
catch(ClassNotFoundException e)
{ }
catch(IOException e1)
{ }
catch(Exception e)

```

```

        {
    }
}

```

```

class Fir_Frame extends Get implements Runnable,ActionListener
{
    String Monlab="Monitoring Packets";
    int count=0;
    Thread t3;
    JFrame jf;
    Container cp;
    JPanel jp;
    JLabel jl;
    JButton jb;
    String inf="com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
    Fir_Frame()
    {
        dis_Fir_Frame();
        dis_Sec_Frame();
        t3=new Thread(this,"Mon");
        t3.start();
    }
    public void dis_Fir_Frame()
    {
        jf=new JFrame("Monitoring");
        jl=new JLabel(Monlab);
        jb=new JButton("View Details");
        cp=jf.getContentPane();
        jl.setFont(new Font("Times-Roman",Font.BOLD,20));
        cp.setLayout(null);
        cp.add(jl);
        cp.add(jb);
    }
}

```

```

        jb.addActionListener(this);
        jl.setBounds(60,5,500,500);
jb.setBounds(150,350,150,50);
jf.setVisible(true);
jf.setBounds(25,25,500,500);
try
    {
        UIManager.setLookAndFeel(inf);
        SwingUtilities.updateComponentTreeUI(jf);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==jb)
    {
        show();
    }
}

public void run()
{
    while(true)
    {
        Monlab=Monlab+".";
        jl.setText(Monlab);
        count++;
        if(count==11)
        {
            Monlab="Monitoring Packets";
            count=0;
        }
    }
}

```

```

else
{
try
{
t3.sleep(500);
}
catch(InterruptedException e)
{
e.printStackTrace();
}
}
}
}
}

```

```

class DRouter
{
public static void main(String args[]) throws IOException,InterruptedException
{
System.out.println("*****Egress
Router*****");
try
{
ServerSocket socket = new ServerSocket(7700);
Fir_Frame ff=new Fir_Frame();
while(true)
{
System.out.println("waiting");
Socket insocket1 = socket.accept();
System.out.println("connected");
Get g1=new Get(insocket1);
}
}
}
}

```



```

        catch(UnknownHostException e)
        {
            System.out.println("UHE");
        }
    }
}
//Tim.java
import java.util.*;

class Tim
{
    public String calculateTime()
    {
        Calendar c;
        int hr,min,sec;
        c=Calendar.getInstance();
        hr=c.get(Calendar.HOUR);
        min=c.get(Calendar.MINUTE);
        sec=c.get(Calendar.SECOND);
        String str=hr+":"+min+": "+sec;
        return str;
    }
}

```

```

//TSW.java
import java.io.*;
import java.net.*;
class TSW
{
    static long winlength=1;
    static long avg_rate=0;
    static long T_Front=0;

```

```

static long Byte_in_TSW=0;
static long New_Byte=0;
static long Packet_Size=0;
static Tim1 t=new Tim1();

```

```

TSW()

```

```

{

}

```

```

TSW(long avg_rate)

```

```

{
    avg_rate=avg_rate;
    //System.out.println("Const");
}

```

```

public void TimeSlide(long Packet_Size,long now)

```

```

{
    Packet_Size=Packet_Size;
    Byte_in_TSW=avg_rate*winlength;
    New_Byte=Byte_in_TSW+Packet_Size;
    avg_rate=New_Byte*(now-T_Front+winlength);
    System.out.println("hai :"+Math.abs(avg_rate));
    String avg=Math.abs(avg_rate)+"";
    long avg1=Long.parseLong(avg.substring(0,14));
    T_Front=now;
    try
    {
        Socket soc=new Socket("localhost",3333);
        PrintWriter pw=new PrintWriter(soc.getOutputStream(),true);
        pw.println(avg1);
    }
    catch(Exception e){e.printStackTrace();}
}

```

```

    }
}
//Source.java
import java.net.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

class SouFrame implements ActionListener
{
    static JButton jb1;
    static JButton jb2;
    static JLabel jl1,jl2;
    static JTextField jtf;
    static JTextArea jtfa;
    static JScrollPane jsp;
    static JFrame jf;
    String dest_addr=" ";
    String Msg=" ";
    String conc=" ";
    String fin=" ";
    int st=0;
    int end=48;
    int len1=0;
    Socket soc;
    int split=0;
    //InetAddress in1=InetAddress.getLocalHost();

    SouFrame() throws IOException
    {
        CreateFrame();
    }
}

```

```

public void CreateFrame()
{
    jf=new JFrame("Source");
        try
        {
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
                SwingUtilities.updateComponentTreeUI(jf);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    Container cp=jf.getContentPane();
    jl1=new JLabel("Destination IP  :");
    jtf=new JTextField();
    jl2=new JLabel("Message  :");
    jtf1=new JTextArea(10,10);
    jb1=new JButton("SEND");
    jb2=new JButton("EXIT");
    jsp=new JScrollPane(jtf1);
    cp.setLayout(null);
    cp.add(jl1);
    cp.add(jtf);
    cp.add(jl2);
    cp.add(jsp);
    cp.add(jb1);
    jl1.setForeground(Color.blue);
    jl2.setForeground(Color.blue);
    jb1.addActionListener(this);
    cp.add(jb2);
    jb2.addActionListener(this);
    jl1.setBounds(80,50,195,25);

```

```

    jtf.setBounds(165,50,200,25);
    jl2.setBounds(80,100,195,25);
    jsp.setBounds(165,100,200,200);
    jb1.setBounds(150,325,75,25);
    jb2.setBounds(250,325,75,25);
    jf.setVisible(true);
    jf.setBounds(100,100,400,400);
    jf.validate();
    jf.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}

```

```

public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==jb1)
    {
        try
        {
            SendPacket();
        }
        catch(IOException e1)
        {
            JOptionPane.showMessageDialog((Component) null,"IngressMachine is
Not Ready To Data Transfer","Click OK",JOptionPane.ERROR_MESSAGE);
        }
    }
    else
    {
        System.exit(0);
    }
}
}

```

```

public void SendPacket() throws IOException
{
    try

```

```

    { InetAddress in1=InetAddress.getLocalHost();
    dest_addr=jtf.getText();
      Msg=jtf1.getText();
      if(((dest_addr.trim()).length())>0)
      {
          if(((Msg.trim()).length())>0)
          {
System.out.println("*****"+jtf.getText()+"*****");
          soc=new Socket("localhost",7788);
          OutputStream out = soc.getOutputStream();
          st=0;
          end=48;
          conc=dest_addr+" "+in1.getHostName()+" ";
          byte buffer[]=Msg.getBytes();
          int len=buffer.length;
          len1=len;
          if(len<=48)
          {
              fin=conc+Msg+"\n"+"null";
              byte buffer1[]=fin.getBytes();
              out.write(buffer1);
          }
          else
          {
              fin=conc+Msg.substring(st,end)+"\n";
              byte buffer2[]=fin.getBytes();
              out.write(buffer2);
              Thread.sleep(1000);
              while(len1>48)
              {
                  len1-=48;
                  if(len1<=48)
                  {

```



```

        }
        catch(InterruptedException e1)
        {
            }
        }
    }
}

class Source
{
    public static void main(String args[])throws IOException
    {
        SouFrame sf=new SouFrame();
    }
}

```

7 *System Testing:*

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.1 **TYPES OF TESTS**

Unit testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing:

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

7.2 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

7.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Test Case Specifications:

Testcase number	Testcase	Input	Expected output	Obtained output
1	Send ip	Destination ip	Send data to destination	sent
2	Send message	Send message to destination	Send data to dest	sent

8 Conclusion and Future Work:

Building packet buffers based on a hybrid SRAM/DRAM architecture while introducing minimum overhead is the major issue discussed in this paper. To distinctly increase the throughput and storage capacity of a packet buffer, a parallel mechanism using multiple DRAM chips should be deployed. Our analysis shows that previous algorithms make very little effects in exploring the advantage of parallel DRAMs leading to the requirement of large size SRAM and high time complexity in memory management. In this paper, we present a novel packet buffer architecture by using both fast batch load scheme and a hierarchal distributed structure. It reduces the requirement of SRAM size greatly. Both mathematical analysis and simulation results indicate that the proposed architecture provides guaranteed performance in terms of the low time complexity, short access delay, and upper bounded drop rate, when a small speedup is provided.

9 References

1. A. Willig, A Short Introduction to Queueing Theory, Telecom Networks Group, pp. 19-27, 1999.
2. B. Agrawal and T. Sherwood, "Virtually Pipelined Network Memory," Proc. IEEE/ACM 39th Ann. Int'l Symp. Microarchitecture (Micro '06), pp. 197-207, Dec. 2006.
3. B.S. Arnaud, "Scaling Issues on Internet Networks," <http://www.canet3.net/library/papers/scaling.pdf>, 2001.
4. Cisco, "Cisco Carrier Router System," <http://www.cisco.com/en/US/products/ps5763/index.html>, 2011.
- [5] D. Lin, M. Hamdi, and J. Muppala, "Designing Packet Buffers in High Bandwidth Switches and Routers," Proc. Int'l Conf. High Performance Switching and Routing (HPSR '10), pp. 32-37, June 2010.
- [6] D. Lin, M. Hamdi, and J. Muppala, "Designing Packet Buffers Using Random Round Robin," Proc. IEEE GlobeCom '10, pp. 1-5, Dec. 2010.

- [7] D. Lin and M. Hamdi, "Two-Stage Fair Queuing Using Budget Round-Robin," Proc. IEEE Int'l Conf. Comm. (ICC '10), pp. 1-5, May 2010.
- [8] DRAMeXchange, <http://www.dramexchange.com/#dram>, 2011.
- [9] F. Wang and M. Hamdi, "Scalable Router Memory Architecture Based on Interleaved DRAM," Proc. Workshop High Performance Switching and Routing (HPSR '06), pp. 6-10, May 2006.
- [10] F. Wang, M. Hamdi, and J. Muppala, "Using Parallel DRAM to Scale Router Buffers," IEEE Trans. Parallel and Distributed Systems, vol. 20, no. 5, pp. 710-724, May 2009.
- [11] G. Appenzeler, I. Keslassy, and N. McKeown, "Sizing Router Buffers," ACM SIGCOMM Computer Comm. Rev., vol. 34, no. 4, pp. 281-292, Oct. 2004.
- [12] G. Shrimali and N. McKeown, "Building Packet Buffers with Interleaved Memories," Proc. Workshop High Performance Switching and Routing (HPSR '05), pp. 1-5, May 2005.
- [13] H. Wang and B. Lin, "Block-Based Buffer with Deterministic Packet Departure," Proc. Int'l Conf. High Performance Switching and Routing (HPSR '10), pp. 38-43, June 2010.
- [14] H. Wang, H. Zhao, B. Lin, and J. Xu, "Design and Analysis of a Robust Pipelined Memory System," Proc. IEEE INFOCOM '10, pp. 1-9, Mar. 2010.
- [15] J. Corbal, R. Espasa, and M. Valero, "Command Vector Memory Systems: High performance at Low Cost," Proc. Int'l Conf. Parallel Architectures and Compilation Techniques, pp. 68-77, Oct. 1998.
- [16] J. Garcia, J. Corbal, L. Cerda, and M. Valero, "Design and Implementation of High-Performance Memory Systems for Future Packet Buffers," Proc. IEEE/ACM 36th Ann. Int'l Symp. Microarchitecture (Micro '03), pp. 372-384, Dec. 2003.
- [17] J. Garcia, M. March, L. Cerda, J. Corbal, and M. Valero, "A DRAM/SRAM Memory Scheme for Fast Packet Buffers," IEEE Trans. Computers, vol. 55, no. 5, pp. 588-602, May 2006.
- [18] J. Kleinberg and E. Tardos, Algorithm Design, pp. 758-760. Prentice Hall, 2006.
- [19] Juniper E Series Router, <http://juniper.net/products/eseries/>, 2011.
- [20] K.G. Coffman and A.M. Odlyzko, "Is There a Moore's Law for Data Traffic?," Handbook of Massive Data Sets, pp. 47-93, Kluwer, 2002.