CHAPTER: 1

INTRODUCTION

1. INTRODUCTION

1.1 Brief Information of Project

The Internet has been transformed from a special purpose network to a ubiquitous platform for a wide range of everyday communication services. The demands on Internet reliability and availability have increased accordingly. A disruption of a link in central parts of a network has the potential to affect hundreds of thousands of phone conversations or TCP connections, with obvious adverse effects. The ability to recover from failures has always been a central design goal in the Internet. IP networks are intrinsically robust, since IGP routing protocols like OSPF are designed to update the forwarding information based on the changed topology after a failure. This re-convergence assumes full distribution of the new link state to all routers in the network domain. When the new state information is distributed, each router individually calculates new valid routing tables.

This network-wide IP re-convergence is a time consuming process, and a link or node failure is typically followed by a period of routing instability. During this period, packets may be dropped due to invalid routes. This phenomenon has been studied in both IGP and BGP context, and has an adverse effect on real-time applications. Events leading to a re-convergence have been shown to occur frequently. Much effort has been devoted to optimizing the different steps of the convergence of IP routing, i.e., detection, dissemination of information and shortest path calculation, but the convergence time is still too large for applications with real time demands. A key problem is that since most network failures are short lived, too rapid triggering of the re-convergence process can cause route flapping and increased network instability.

The IGP convergence process is slow because it is *reactive* and *global*. It reacts to a failure after it has happened, and it involves all the routers in the domain. In this paper we present a new scheme for handling link and node failures in IP networks. Multiple Routing Configuration (MRC) is a *proactive* and *local* protection mechanism that allows recovery in the range of milliseconds. MRC allows packet forwarding to continue over preconfigured alternative next-hops immediately after the detection of the failure. Using MRC as a first line of defense against network failures, the normal IP convergence process can be put on hold. This process is then initiated only as a consequence of non-transient failures. Since no global re-routing is performed, fast failure detection mechanisms like fast hellos or hardware alerts can be used to trigger MRC without compromising network stability.

Objective of the project

The main idea of MRC is to use the network graph and the associated link weights to produce a small set of backup network configurations. The link weights in these backup configurations are manipulated so that for each link and node failure, and regardless of whether it is a link or node failure, the node that detects the failure can safely forward the incoming packets towards the destination on an alternate link. MRC assumes that the network uses shortest path routing and destination based hop-by-hop forwarding. The shifting of traffic to links bypassing the failure can lead to congestion and packet loss in parts of the network. This limits the time that the proactive recovery scheme can be used to forward traffic before the global routing protocol is informed about the failure, and hence reduces the chance that a transient failure can be handled without a full global routing re-convergence. Ideally, a proactive recovery scheme should not only guarantee connectivity after a failure, but also do so in a manner that does not cause an unacceptable load distribution.

This requirement has been noted as being one of the principal challenges for pre calculated IP recovery schemes. With MRC, the link weights are set individually in each backup configuration. This gives great flexibility with respect to how the recovered traffic is routed. The backup configuration used after a failure is selected based on the failure instance, and thus we can choose link weights in the backup configurations that are well suited for only a subset of failure instances.

1.2 Overview of the project

MRC is based on building a small set of backup routing configurations that are used to route recovered traffic on alternate paths after a failure. The backup configurations differ from the normal routing configuration in that link weights are set so as to avoid routing traffic in certain parts of the network. We observe that if all links attached to a node are given sufficiently high link weights, traffic will never be routed through that node. The failure of that node will then only affect traffic that is sourced at or destined for the node itself. Similarly, to exclude a link (or a group of links) from taking part in the routing, we give it infinite weight. The link can then fail without any consequences for the traffic. Our MRC approach is threefold. First, we create a set of backup configurations, so that every network component is excluded from packet forwarding in one configuration. Second, for each configuration, a standard routing algorithm like OSPF is used to calculate configuration specific shortest paths and create forwarding tables in each router, based on the

configurations. The use of a standard routing algorithm guarantees loop-free forwarding within one configuration. Finally, we design a forwarding process that takes advantage of the backup configurations to provide fast recovery from a component failure.

In our approach, we construct the backup configurations so that for all links and nodes in the network, there is a configuration where that link or node is not used to forward traffic. Thus, for any single link or node failure, there will exist a configuration that will route the traffic to its destination on a path that avoids the failed element. Also, the backup configurations must be constructed so that all nodes are reachable in all configurations, i.e., there is a valid path with a finite cost between each node pair. Shared Risk Groups can also be protected, by regarding such a group as a single component that must be avoided in a particular configuration. We formally describe MRC and how to generate configurations that protect every link and node in a network.

CHAPTER: 2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Introduction

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

Although many randomized asynchronous protocols have been designed throughout the years , only recently one implementation of a stack of randomized multicast and agreement protocols has been reported, SINTRA. These protocols are built on top of a binary consensus protocol that follows a Rabin-style approach, and in practice terminates in one or two communication steps. The protocols, however, depend heavily on public-key cryptography primitives like digital and threshold signatures. The implementation of the stack is in Java and uses several threads. RITAS uses a different approach, Ben-Or-style, and resorts only to fast cryptographic operations such as hash functions.

Randomization is only one of the techniques that can be used to circumvent the FLP impossibility result. Other techniques include failure detectors, partial synchrony and distributed wormholes. Some of these techniques have been employed in the past to build other intrusion-tolerant protocol suites.

2.2 Definition of terms:

MRC

MRC is strictly connectionless, and assumes only destination based hop-by-hop forwarding. MRC is based on keeping additional routing information in the routers, and allows packet forwarding to continue on an alternative output link immediately after the detection of a failure.

Multiple Routing Configurations as an approach to achieve fast recovery in IP networks. MRC is based on providing the routers with additional routing configurations, allowing them to forward packets along routes that avoid a failed component. MRC guarantees recovery from any single node or link failure in an arbitrary bi-connected network.

By calculating backup configurations in advance, and operating based on locally available information only, MRC can act promptly after failure discovery.

Background concepts:

This network-wide IP re-convergence is a time consuming process, and a link or node failure is typically followed by a period of routing instability. During this period, packets may be dropped due to invalid routes. This phenomenon has been studied in both IGP and BGP context, and has an adverse effect on real-time applications. Events leading to a re-convergence have been shown to occur frequently. Much effort has been devoted to optimizing the different steps of the convergence of IP routing, i.e., detection, dissemination of information and shortest path calculation, but the convergence time is still too large for applications with real time demands. A key problem is that since most network failures are short lived, too rapid triggering of the re-convergence process can cause route flapping and increased network instability.

The IGP convergence process is slow because it is *reactive* and *global*. It reacts to a failure after it has happened, and it involves all the routers in the domain. In this paper we present a new scheme for handling link and node failures in IP networks. Multiple Routing Configurations (MRC) is a *proactive* and *local* protection mechanism that allows recovery in the range of milliseconds. MRC allows packet forwarding to continue over preconfigured alternative next-hops immediately after the detection of the failure. Using MRC as a first line of defense against network failures, the normal IP convergence process can be put on hold. This process is then initiated only as a consequence of non-transient failures. Since no global re-routing is performed, fast failure detection mechanisms like fast hellos or hardware alerts can be used to trigger MRC without compromising network stability.

Advantage:

- MRC guarantees recovery from any single link or node failure, which constitutes a large majority of the failures experienced in a network.
- MRC makes no assumptions with respect to the root cause of failure, e.g., whether the packet forwarding is disrupted due to a failed link or a failed router.
- MRC is to use the network graph and the associated link weights to produce a small set of backup network configuration.

CHAPTER: 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Existing System

The Internet has been transformed from a special purpose network to an ubiquitous platform for a wide range of everyday communication services. The demands on Internet reliability and availability have increased accordingly. A disruption of a link in central parts of a network has the potential to affect hundreds of thousands of phone conversations or TCP connections, with obvious adverse effects. The ability to recover from failures has always been a central design goal in the Internet. IP networks are intrinsically robust, since IGP routing protocols like OSPF are designed to update the forwarding information based on the changed topology after a failure. This re-convergence assumes full distribution of the new link state to all routers in the network domain

3.2 Proposed System

To assure fast recovery from link and node failures in IP networks, we present a new recovery scheme called Multiple Routing Configurations (MRC). Our proposed scheme guarantees recovery in all single failure scenarios, using a single mechanism to handle both link and node failures, and without knowing the root cause of the failure. MRC is strictly connectionless, and assumes only destination based hop-by-hop forwarding. MRC is based on keeping additional routing information in the routers, and allows packet forwarding to continue on an alternative output link immediately after the detection of a failure.

3.3 Feasibility Study

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time.

System analysis is conducted with the following objectives

- Identify the user needs
- Evaluate the system concept for feasibility
- Perform technical and economic feasibility

- Allocate functions to hardware, software, people, databases& other system elements.
- Establish cost schedule constraints.

There are aspects in the feasibility study portion of the preliminary investigation:

- Technical feasibility
- Operational feasibility
- Economical feasibility

Technical Feasibility

It is the most difficult area to access because objectives, functions performance are somewhat hazy, anything seems to be possible if right assumptions are made. The considerations that are normally associated with technical feasibility include

Technology:

The proposed system will generate many kinds of reports depending on the requirements. By automating all these activities the work is done effectively and in time. There is also quick and good response for each operation.

Operational Feasibility

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project: Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

- Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
- Have the user been involved in the planning and development of the project?
- Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project

- Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

Economical Feasibility

The Economic Feasibility is generally the bottom line considerations for most systems. It is an obvious fact that the computerization of the project is economically advantageous.Firstly it will increase the efficiency and decrease the man-hour required to achieve the necessary result. Secondly it will provide timely and up to date to the administrative and individual departments. Since all the information is available with in a few seconds the system performance will be substantially increased.

3.4 Requirements Specifications

3.4.1 Minimum Hardware Requirements

- Hard Disk : 40 GB
- RAM : 256 MB

3.4.2 Software Requirements

- Operating system : Windows XP
- Front End : Java, Swing

CHAPTER: 4

DESIGN ANALYSIS

4. DESIGN ANALYSIS

4.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

4.2 Modules Description

Client Module:

- This module is used to send the data to server through routers
- It will provide user friendly interface to send the data to the required destination

Router Module:

- This are placed in between server and client to transfer the data.
- Whenever client sends the data to the server it will pass through any one router.
- If the router is failed the data will be transferred through another router to reduce the system failure.

Server Module:

- It will receive the data send by the client which came from the active router.
- It can have any number of clients.

4.3 UML Diagrams

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

- User Model View
    i. This view represents the system from the user's perspective.
    ii. The analysis representation describes a usage scenario from the end-users perspective.
- Structural model view
    i. In this model the data and functionality are arrived from inside the system.
    ii. This model view models the static structures.

- Behavioral Model View

  It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- Implementation Model View

  In this the structural and behavioral as parts of the system are represented as they are to be built.

- Environmental Model View

  In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

- UML Analysis modeling, this focuses on the user model and structural model views of the system.
- UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view. Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

4.3.1 Use Case Diagram

Use case describes the behavior of a system. It is used to structure things in a model. It contains multiple scenarios, each of which describes a sequence of actions that is clear enough for outsiders to understand.

An actor represents a coherent set of roles that users of a system play when interacting with the use cases of the system. An actor participates in use cases to accomplish an overall purpose. An actor can represent the role of a human, a device, or any other systems.

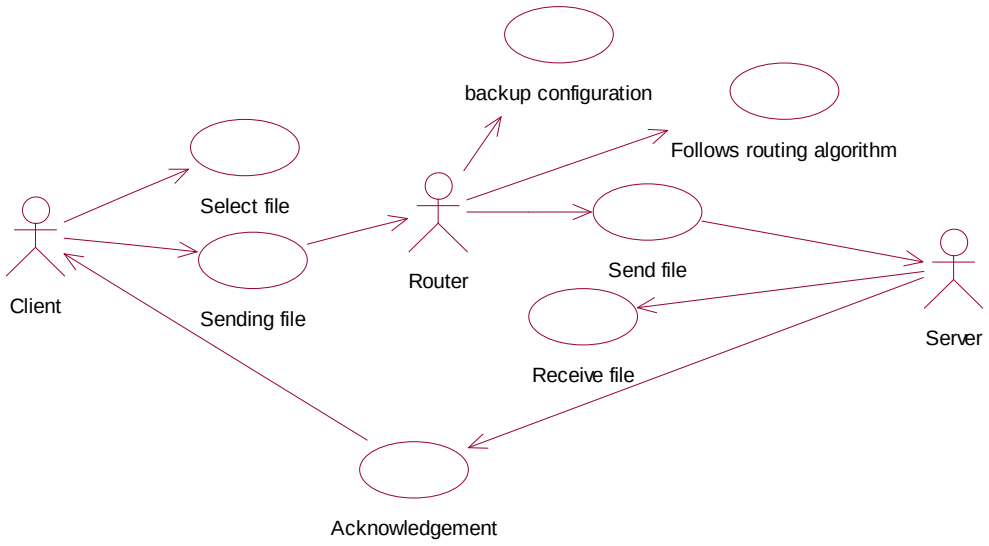


Fig.4.3.1. Use case Diagram

Template for Clients

Participating Actor : Clients

Flow of Events **:** 1. Select the file

2. Send file to the routers.

Entry Condition **:** Sensor sends the file.

Exit Condition **:** Routers receives the file.

Template for Routers

Participating Actor : Routers

Flow of Events **:** 1. Routers connected to the server.

2 .Routers receives the file, acknowledge of the file and Also file size of the file.

Entry Condition **:** Routers connected to the clients.

Exit Condition **:** Routers receives the file.

Template for Server

Participating Actor : Server

Flow of Events **:** 1.Receive the file from routers.

2. Receive the file size.

Entry Condition **:** Server connected to the routers.

Exit Condition **:** Server receives the file.

4.3.2 Sequence Diagram

This diagram is simple and visually logical, so it is easy to see the sequence of the flow of control. It also clearly shows concurrent processes and activations in a design.

Object: Object can be viewed as an entity at a particular point in time with a specific value and as a holder of identity that has different values over time. Associations among objects are not shown. When you place an object tag in the design area, a lifeline is automatically drawn and attached to that object tag.

Actor: An actor represents a coherent set of roles that users of a system play when interacting with the use cases of the system. An actor participates in use cases to accomplish an overall purpose. An actor can represent the role of a human, a device, or any other systems.

Message: A message is a sending of a signal from one sender object to other receiver object(s). It can also be the call of an operation on receiver object by caller object. The arrow can be labeled with the name of the message (operation or signal) and its argument values

Duration Message: A message that indicates an action will cause transition from one state to another state.

Self Message: A message that indicates an action will perform at a particular state and stay there.

Create Message: A message that indicates an action that will perform between two states.

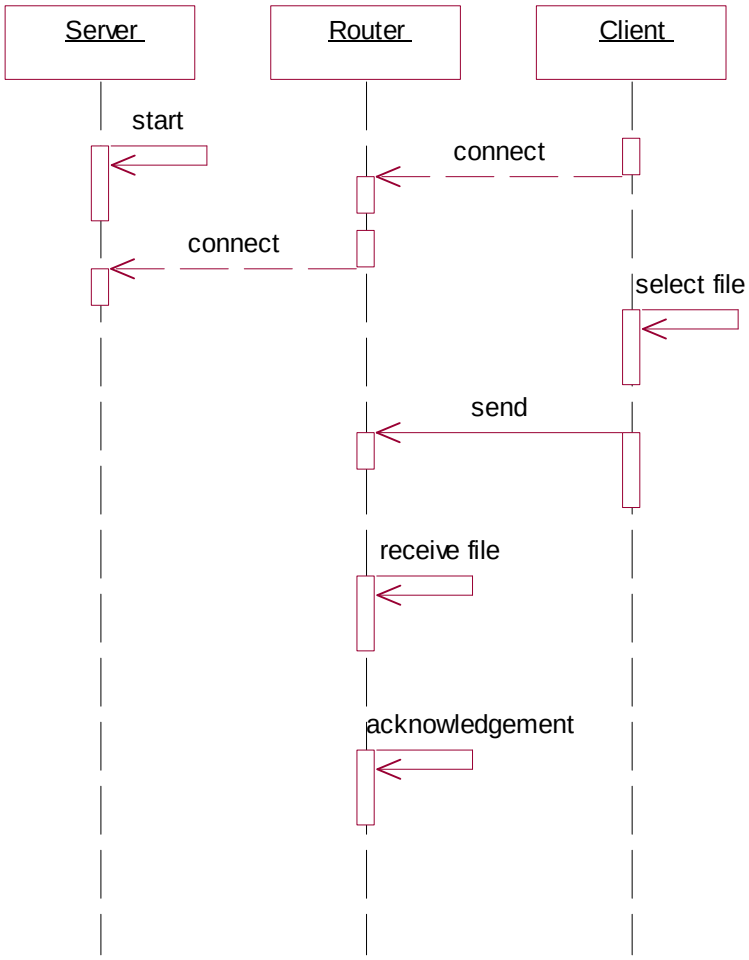


Fig.4.3.2.1 Sequence diagram for router failure condition

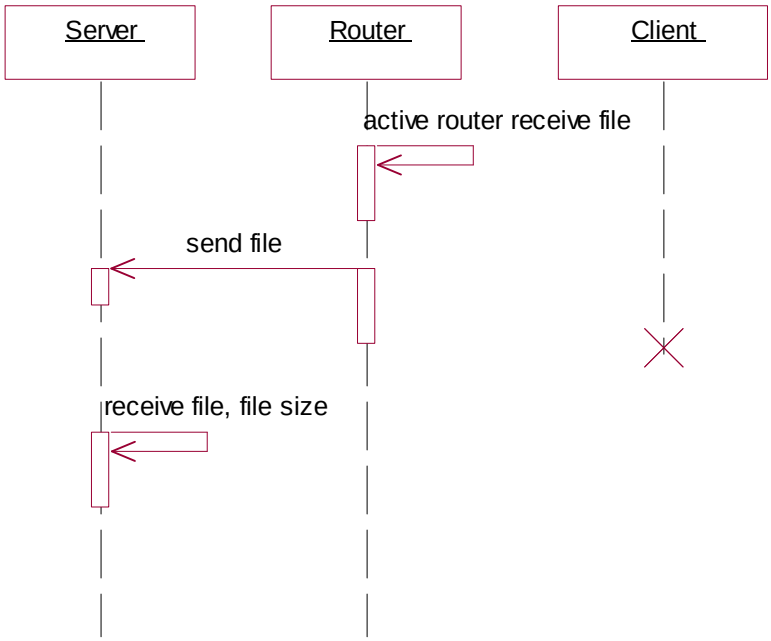


Fig.4.3.2.2 Sequence diagram for active router sending data

4.3.3 Collaboration diagram

Like the other Behavioral diagrams, Collaboration diagrams model the interactions between objects. This type of diagram is a cross between an object diagram and a sequence diagram. Unlike the Sequence diagram, which models the interaction in a column and row type format, the Collaboration diagram uses the free-form arrangement of objects as found in an Object diagram. This makes it easier to see all interactions involving a particular object.

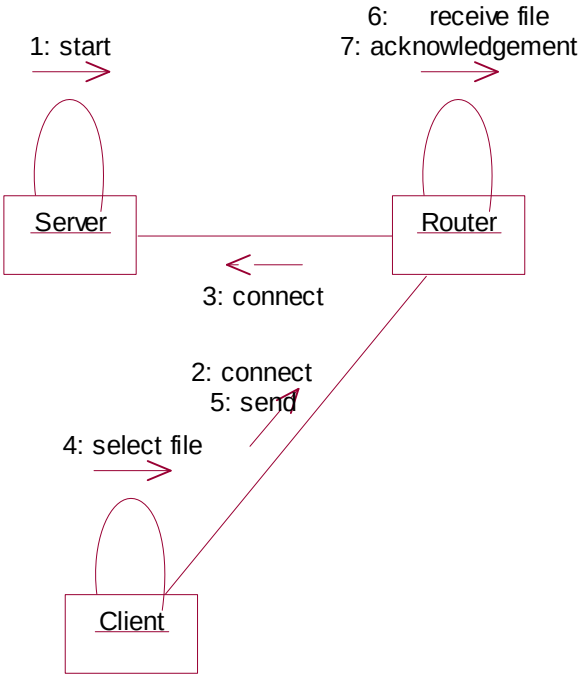


Fig.4.3.3.1 Collaboration Diagram for router failure condition

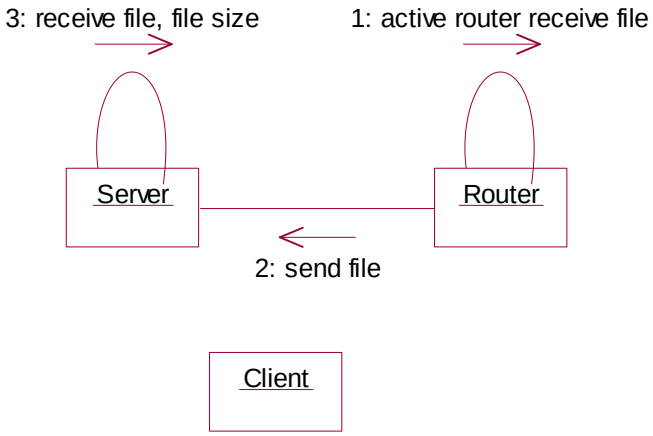


Fig.4.3.3.2 Collaboration Diagram for router sending data

4.3.4 Activity diagram

This shows the flow of events within the system. The activities that occur within a use case or within an objects behavior typically occur in a sequence .an activity diagram is designed to be simplified look at what happens during an operations or a process.

Each activity is represented by a rounded rectangle the processing within an activity goes to compilation and than an automatic transmission to the next activity occurs. An arrow represents the transition from one activity to the next. An activity diagram describes a system in terms of activities. Activities are the state that represents the execution of a set of operations. These are similar to flow chart diagram and dataflow.

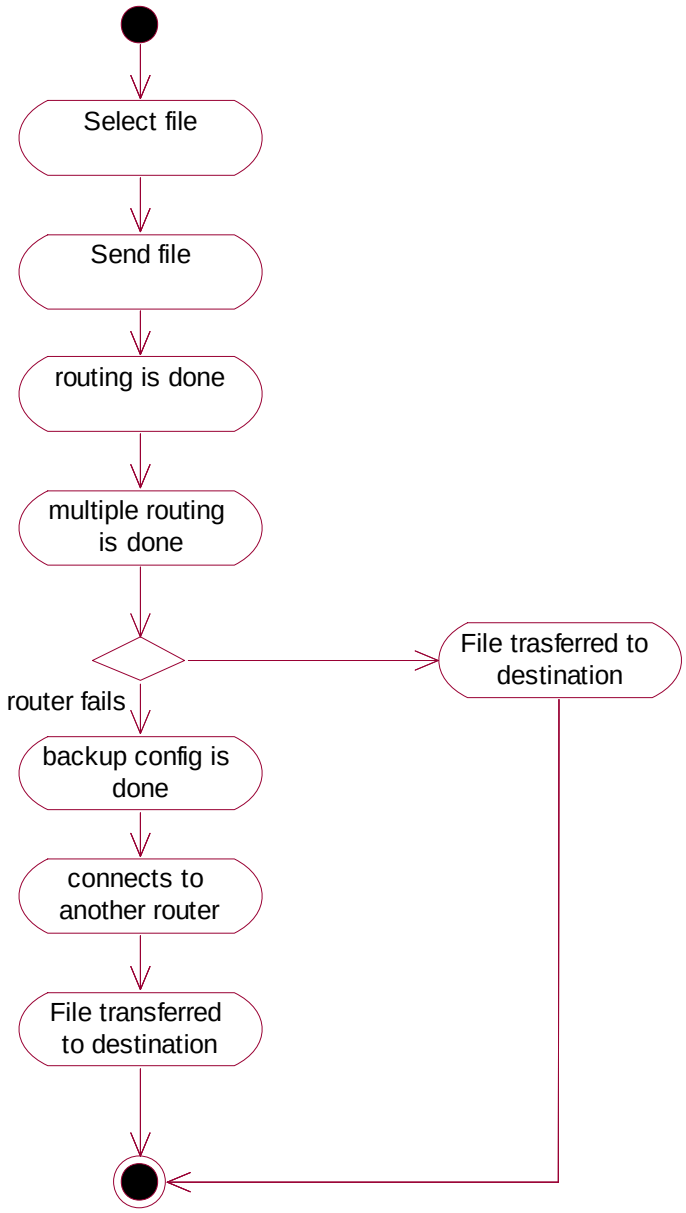


Fig.4.3.4 Activity Diagram

4.3.5 Class Diagram

Class: A Class is a description for a set of objects that shares the same attributes, and has similar operations, relationships, behaviors and semantics.

Generalization: Generalization is a relationship between a general element and a more specific kind of that element. It means that the more specific element can be used whenever the general element appears. This relation is also known as specialization or inheritance link.

Realization: Realization is the relationship between a specialization and its implementation. It is an indication of the inheritance of behavior without the inheritance of structure.

Association: Association is represented by drawing a line between classes. Associations represent structural relationships between classes and can be named to facilitate model understanding. If two classes are associated, you can navigate from an object of one class to an object of the class.

Aggregation: Aggregation is a special kind of association in which one class represents as the larger class that consists of a smaller class. It has the meaning of “has-a” relationship.

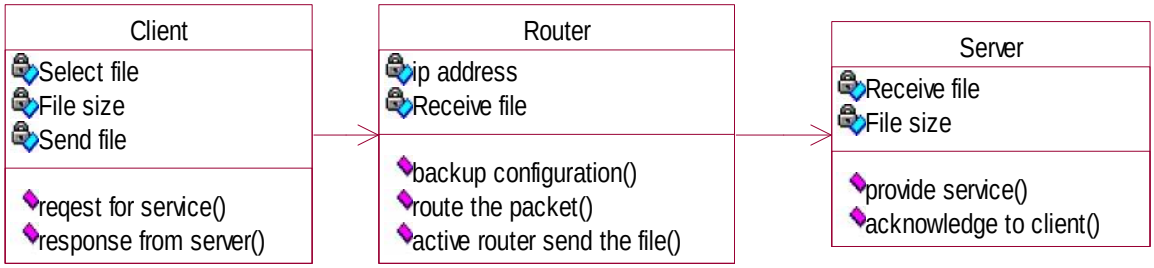


Fig 4.3.5 Class Diagram

CHAPTER: 5

TECHNOLOGY DESCRIPTION

5. TECHNOLOGY DESCRIPTION

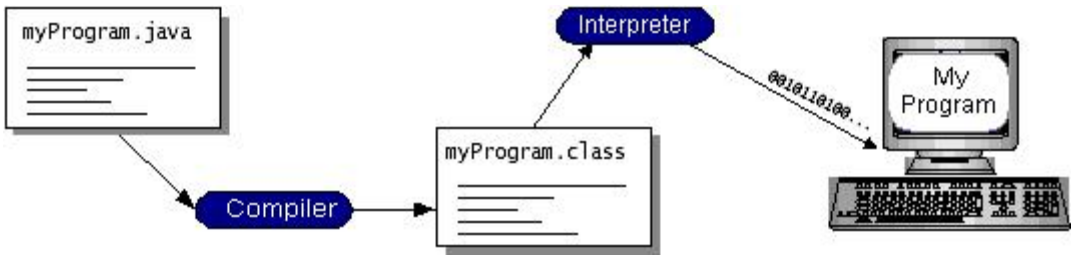
5.1 Introduction to Java

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

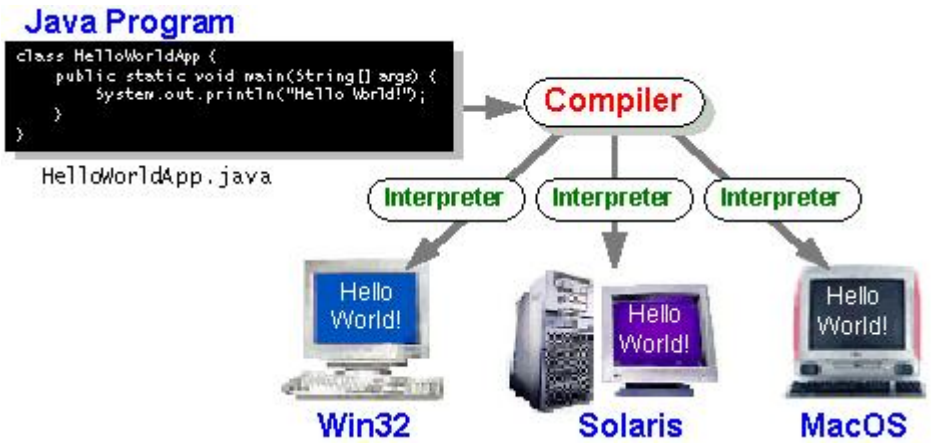
Finally, Java is to Internet programming where C was to system programming.

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.



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

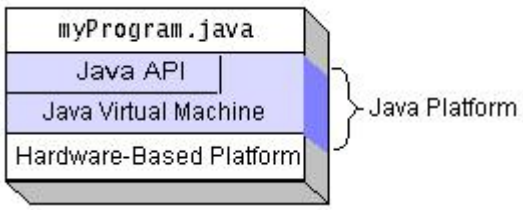
A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from 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 Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

It is 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.



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.

Features of Java:

Security

Every time you that you download a "normal" program; you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both of these concerns by providing a "firewall" between a networked application and your computer. When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

Portability

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed .As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java's solution to these two problems is both elegant and efficient.

The Byte code

The key that allows the Java to solve the security and portability problem is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to execute by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code. Translating a

Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it. Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

Java Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

The above picture shows the development process a typical Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called **JAVA**. The Java compiler produces a file called a. class file, which contains the byte code. The class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

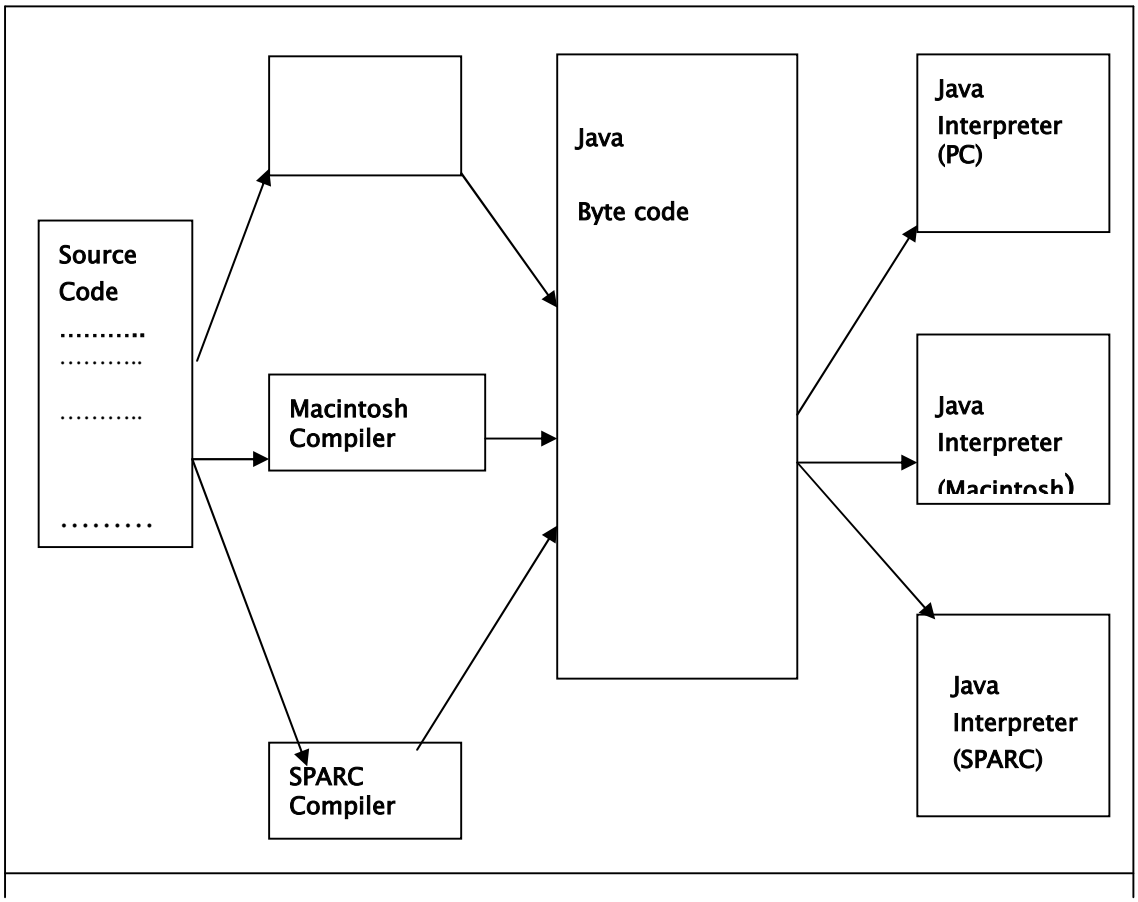
Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of Code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

Compiling and interpreting Java Source Code



During run-time the Java interpreter tricks the byte code file into thinking that it is running on a Java Virtual Machine. In reality this could be a Intel Pentium Windows 95 or Sun SARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

Simple

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the objects oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

Object-Oriented

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

Robust

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time. Java virtually eliminates the problems of memory management and de-allocation, which is completely automatic. In a well-written Java program, all run time errors can –and should –be managed by your program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change. Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be

a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program, and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow.

CHAPTER: 6

SAMPLE CODE

6. SAMPLE CODE

Sample code for server:

```
public class server {
public JLabel la1= new JLabel ("SERVER    ");
public JLabel la2= new JLabel("Received File         ");
public JLabel la3= new JLabel("File Size              :    ");
public JLabel la4= new JLabel("Source IPAddress  :    ");
public JTextArea t1= new JTextArea("");
public JTextField c1= new JTextField("");
public JTextField c2= new JTextField("");
public Font l=  new Font ("Times New roman" , Font.BOLD, 18);
public JScrollPane sc=new JScrollPane();
public Font l1=  new Font ("Times New roman" , Font.BOLD, 30 );
public JFrame jf;
public Container c;
server()
{
jf = new JFrame("Server");
c = jf.getContentPane();
c.setLayout(null);
jf.setSize(1024,736);
c.setBackground( new Color(236,216,234));
la1.setBounds(430, 50, 200, 35);
la2.setBounds(150, 200, 200, 35);
la4.setBounds(150, 130, 200, 50);
//t1.setBounds(150, 250, 400, 250);
la3.setBounds(150, 550, 200, 35);
c1.setBounds(320, 550, 150, 35);
c2.setBounds(340, 140, 150, 35);
c1.setForeground(new Color(30, 30,98));
c1.setFont(l);
```

```
sc.setBounds(100,300,300,200);
t1.setColumns(20);
t1.setRows(10);
t1.setForeground(new Color(160,35,163));
t1.setFont(l);
sc.setViewportView(t1);
c.add(la1);
c.add(la2);
c.add(la3);
//c.add(la4);
c.add(sc);
c.add(c1);
//c.add(c2);
la1.setFont(l1);
la2.setFont(l);
la3.setFont(l);
la4.setFont(l);
jf.show();
jf.addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent win) {
System.exit(0);
}
});

int[] ports = new int[] { 5000 };

for (int i = 0; i < 1; i++) {
Thread t = new Thread(new PortListener(ports[i]));
t.setName("Listener-" + ports[i]);
t.start();

}
}
```

```
public static void main (String args[])
{
new server();

}
```

Sample code for Client 1:

```
public class client1 implements ActionListener
{
public JButton b1= new JButton("Browse");
public JButton b2= new JButton("Split");
public JButton b3= new JButton("Send");
public JLabel la1= new JLabel("Select the file :");
public JLabel la2= new JLabel("File path         :");
public JLabel la3= new JLabel("File Size (Bits)   :");
public JLabel la4= new JLabel("CLIENT 1");
public JLabel c1= new JLabel();
public JTextField c2= new JTextField("");
public JTextArea t2= new JTextArea("");
public JScrollPane sc=new JScrollPane();
public JScrollPane sc1=new JScrollPane();
public JTextArea t1= new JTextArea("");
public Font l = new Font("Times New roman" , Font.BOLD , 18);
public Font l2 = new Font("Times New roman" , Font.BOLD , 16);
public Font l1 = new Font("Times New roman" , Font.BOLD, 30);
public JFrame jf;
public Container c;
client1()
{
jf = new JFrame("Client1");
c = jf.getContentPane();
c.setLayout(null);
jf.setSize(1024,736);
```

```
c.setBackground( new Color(236,216,234));
b1.setBounds(300,110,100,35);
b1.setFont(l);
b1.setForeground(new Color(10,70,198));
b2.setBounds(700,400,100,35);
b3.setBounds(375,600,100,35);
b3.setFont(l);
b3.setForeground(new Color(10,70,198));
la1.setBounds(100,100,150,50);
la2.setBounds(100,150,150,50);
la3.setBounds(100,200,150,50);
la4.setBounds(420,0,150,50);
c1.setBounds(300,160,400,35);
c1.setFont(l2);
c1.setForeground(new Color(120,0,0));
c2.setBounds(300,210,100,35);

c2.setForeground(new Color(30,70,98));
c2.setFont(l);
sc1.setBounds(400,300,100,200);
//b1.setBorderPainted(false);
sc.setBounds(100,300,300,200);
t1.setColumns(20);
t1.setRows(10);
t1.setForeground(Color.BLUE);
t1.setFont(l);
sc.setViewportView(t1);
t2.setBounds(100,300,300,200);
t2.setColumns(20);
t2.setRows(10);

sc1.setViewportView(t2);
c.add(b1);
c.add(b3);
```

```
  c.add(la1);
  c.add(la2);
  c.add(la3);
  c.add(la4);
  c.add(c1);
  c.add(c2);
  c.add(sc,BorderLayout.CENTER);
  la1.setFont(l);
  la2.setFont(l);
  la3.setFont(l);
  la4.setFont(l1);
  b1.addActionListener(this);
  b2.addActionListener(this);
  b3.addActionListener(this);
  jf.show();
  jf.addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent win) {
  System.exit(0);
}
}
```

CHAPTER: 7

TESTING

7. TESTING

7.1 Testing Concepts

Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior of the system.

- A component is a part of the system that can be isolated for testing. A component can be an object, a group of objects, or one or more subsystems.
- A fault, also called bug or defect, is a design or coding mistake that may cause abnormal component behavior.
- An error is a manifestation of a fault during the execution of the system.
- A failure is a deviation between the specification of a component and its behavior. A failure is triggered by one or more errors.
- A test case is a set of inputs and expected results that exercise a component with the purpose of causing failures and detecting faults.

Testing activities:

- Inspecting a component, this finds faults in an individual component through the manual inspection of its source code.
- Unit testing, which finds faults by isolating an individual component using test stubs and drivers and by exercising the components using a test case.
- Integration testing, which finds faults by integrating several components together.
- System testing, which focuses on the complete system, its functional and nonfunctional requirements and its target environment.

Testing is the phase where the errors remaining from all the previous phases must be detected. Hence, testing performs a very critical role for quality assurance and for ensuring the reliability of software. Testing of designed software consists of providing the software with a set of test outputs and observing if the software behaves as expected. If the software fails to behave as expected, then the conditions under which a failure occurs are needed for debugging and correction.

The following terms are some commonly used terms associated with testing.

Error

The term error is used in two different ways. It refers to the discrepancy between a computed, observed, or measured value and true, specified, or theoretically correct value. Error is also used to refer to human action those results in software containing a defect or fault. This definition is quite general and encompasses all the phases.

Fault

Fault is a condition that causes a system to fail in performing its required function. In other words a fault is an incorrect intermediate state that may have been entered during program execution

Failure

Failure is the inability of the system or component to perform a required function according to its specifications. In other words a failure is a manifestation of error. But the mere presence of an error may not cause a failure. Presence of an error implies that a failure must have occurred, and the observation of a failure implies that a fault must be present in the system. However, the presence of a fault does not imply that a failure must occur. A test case is the triplet [i, s, o], where i stands for the data input to the system, s is the state of the system at which the data is input, and o is the expected output of the system .A test suite is the set of all test cases with which a given software product is to be tested.

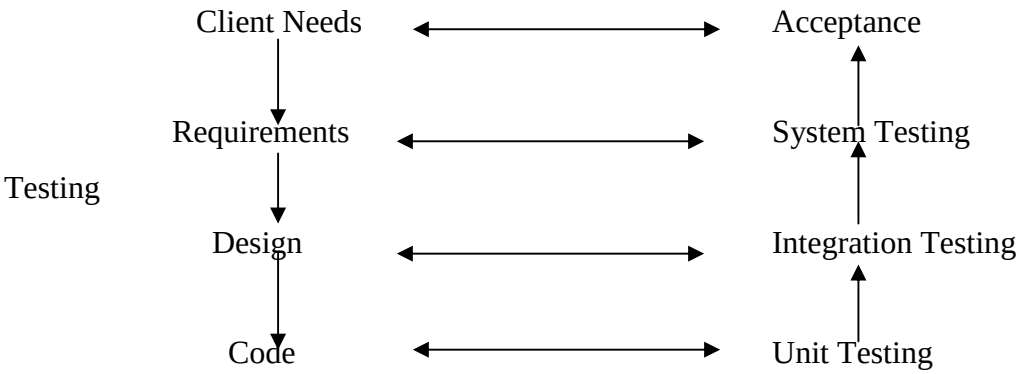
Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say,

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as yet undiscovered error.
- A good test case is one that has a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

Levels of Testing

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are



Unit testing:

Unit testing focuses verification effort on the smallest unit of software i.e. the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

In this project "Evaluation of Employee Performance" each service can be thought of a module. There are so many modules like Executive, Debit Card, Credit Cards, Performance, and Bills. Each module has been tested by giving different sets of inputs (giving wrong Debit card Number, Executive code) when developing the module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user.

Integration Testing:

After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

In this project 'Evaluation of Employee Performance', the main system is formed by integrating all the modules. When integrating all the modules I have checked whether the integration effects working of any of the services by giving different combinations of inputs with which the two services run perfectly before Integration.

System Testing

Here the entire software system is tested. The reference document for this process is the requirements document, and the goals to see if software meets its requirements. Here entire ‘Evaluation of Employee Performance’ has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

Acceptance Testing

Acceptance Test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system; the internal logic of program is not emphasized.

In this project ‘Evaluation of Employee Performance’s have collected some data and tested whether project is working correctly or not.

Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once. The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied.

White Box Testing

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors.I tested step wise every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing. I have generated a list of test cases, sample data. Which is used to check all possible combinations of execution paths through the code at every module level.

Black Box Testing

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

7.2 Sample Test Case Specification

Test case id	Test case Name	Input	Expected output	Observed Output	Result
T1	Router B	Client sends File to the server.	Router B shows a message of”data successfully received”.	Router B shows a message of”data successfully received”.	Pass
T2	Router C	Client sends File to the server.	This 'Router C' failure physical problem occurred. Therefore this data transfer using sub path	This 'Router C' failure physical problem occurred. Therefore this data transfer using sub path	Pass
T3	Server	No input	. The server receive blank file and file size is 0.	The server can’t receive blank file and file size is 0.	Fail

Table: 7.2 Test cases

7.2.1: Test case screens

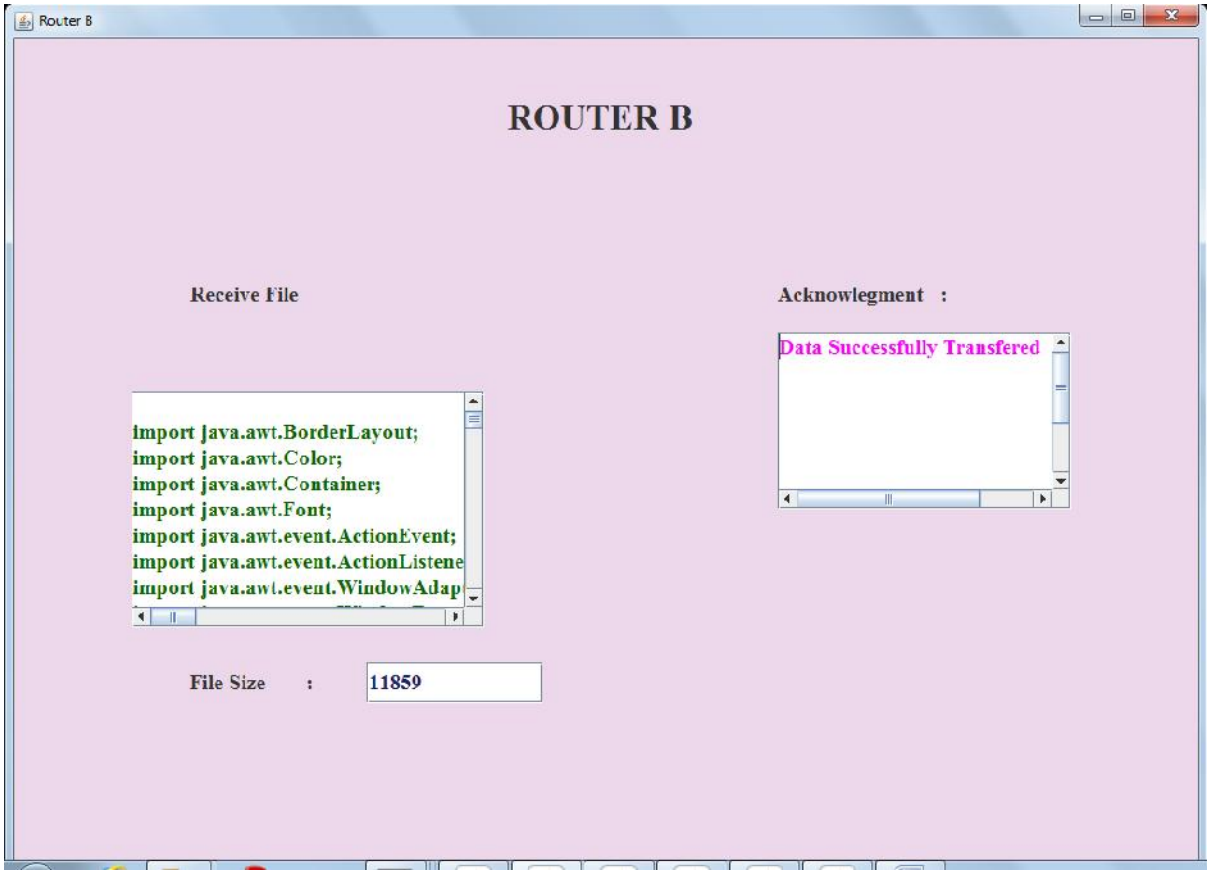


Fig 7.2.1: Test screen for Router B

Description: The Router B receives the file and size of the file. It can also Show an acknowledgement of "data successfully transferred".

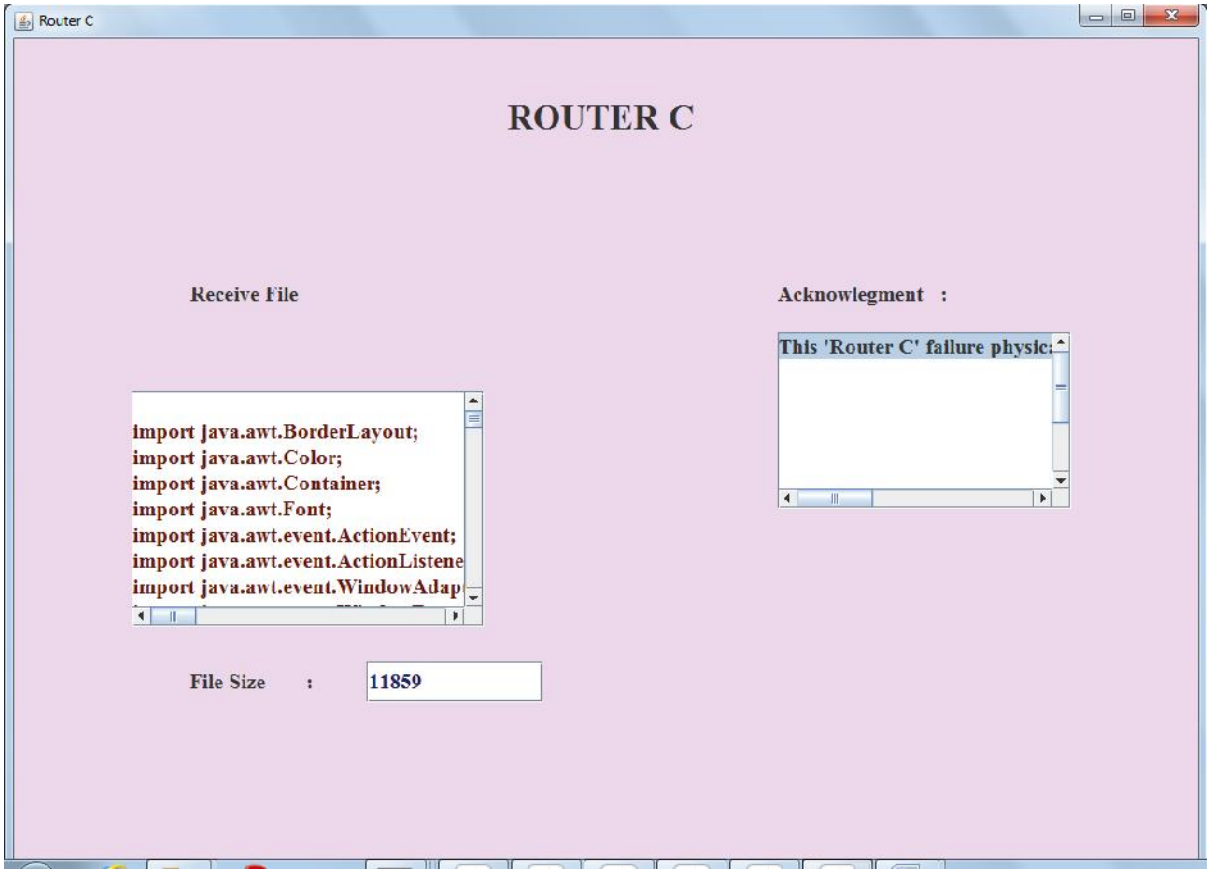


Fig 7.2.2: Test screen for Router C

Description: In this test screen Router C receives the file from client1 and it can show an acknowledgement of “This 'Router C' failure physical problem occurred. Therefore this data transfer using sub path”. It can show file size in terms of bits.

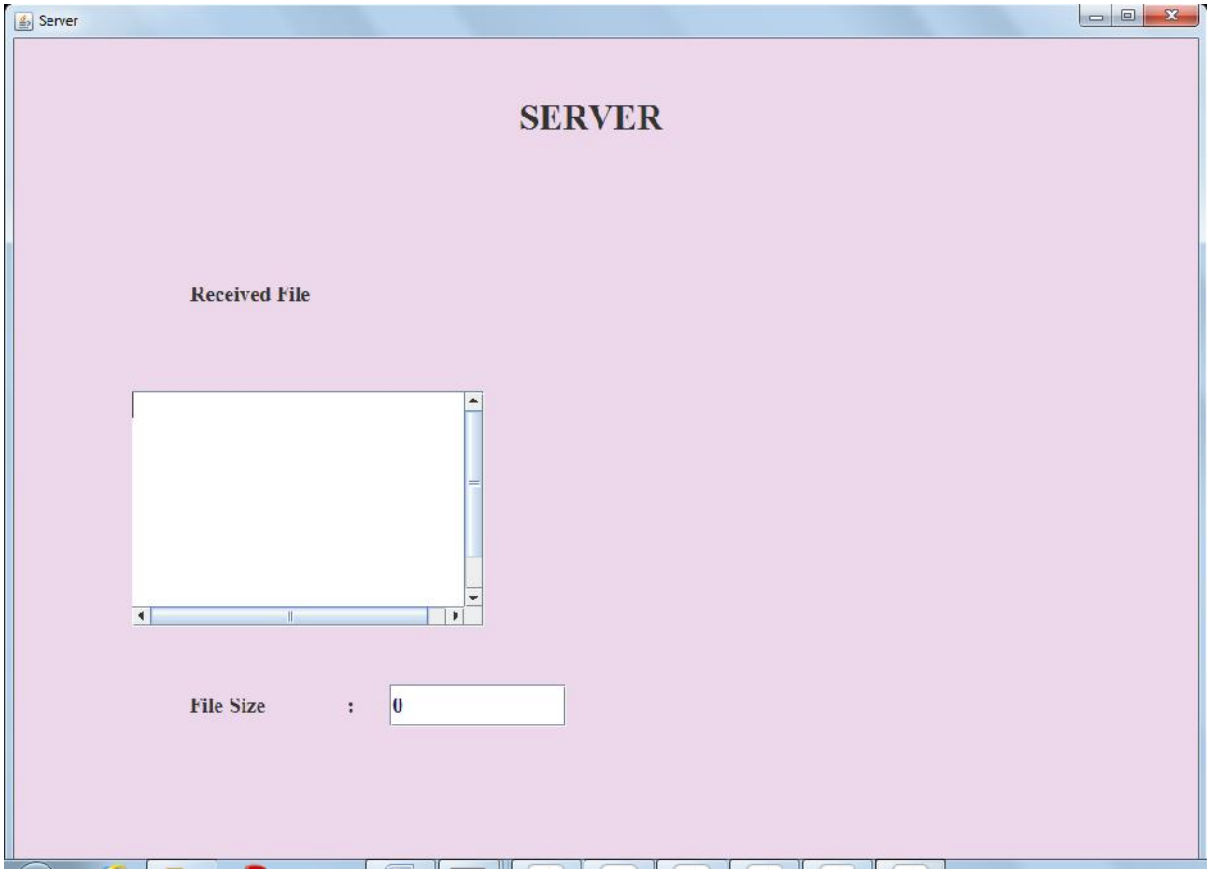


Fig 7.2.3: Test screen for server

Description: The client1 can't send any file to the server. The server receive blank file and file size is 0. The server receives the file and file size from client by using routers. The client1 send file to the server using routers. The server displays the file data and file size.

CHAPTER: 8
SCREEN SHOTS

8. SCREEN SHOTS

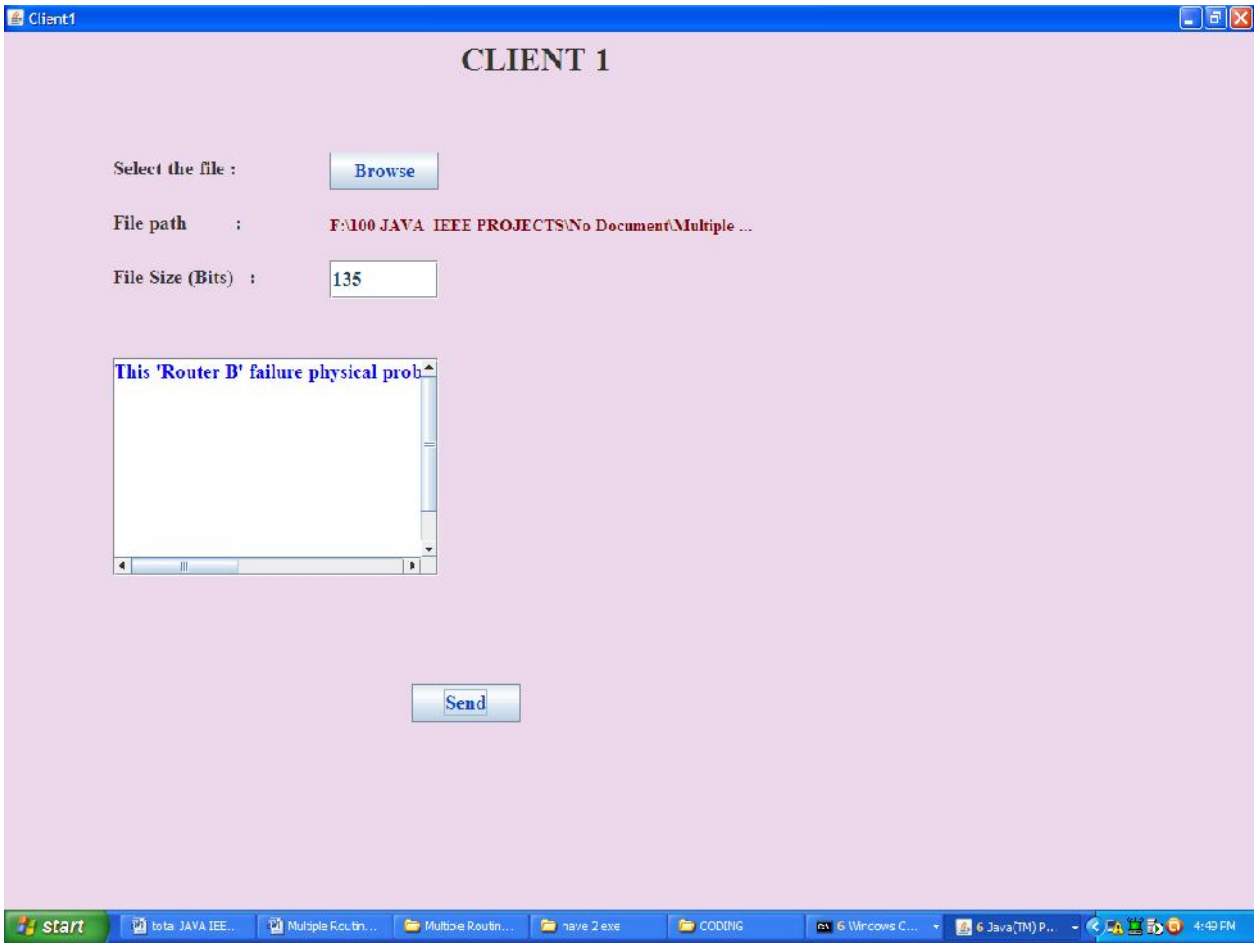


Fig 8.1: Screen shot for Client 1

In the client1, the client sends data file and size of the file to the server by using routers. It can show which router is physically problem. The file path can be displayed to the client1.

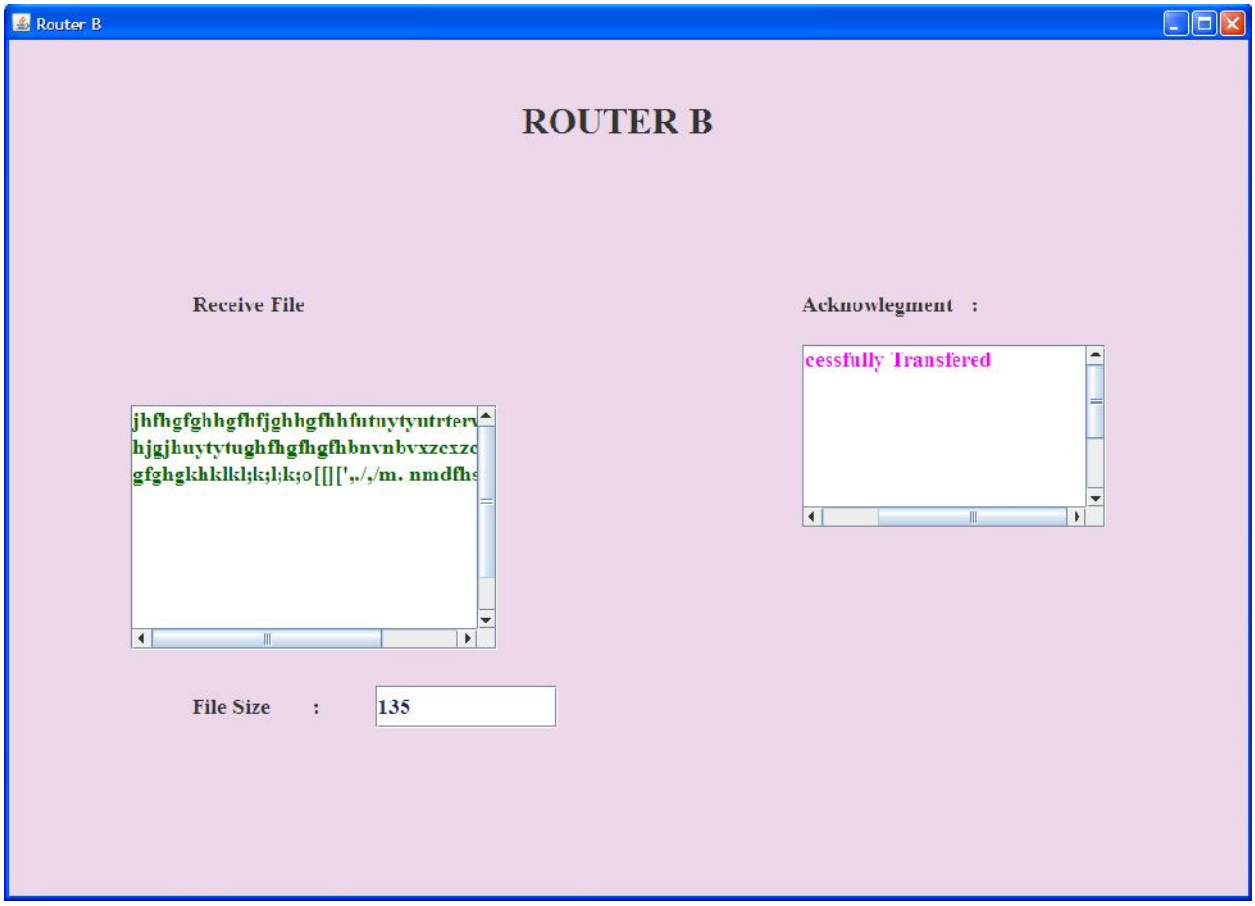


Fig 8.2: Screen shot for Router B

In this screen shot the Router B receive the file and size of the file. It can also Show an acknowledgement of "data successfully transferred".

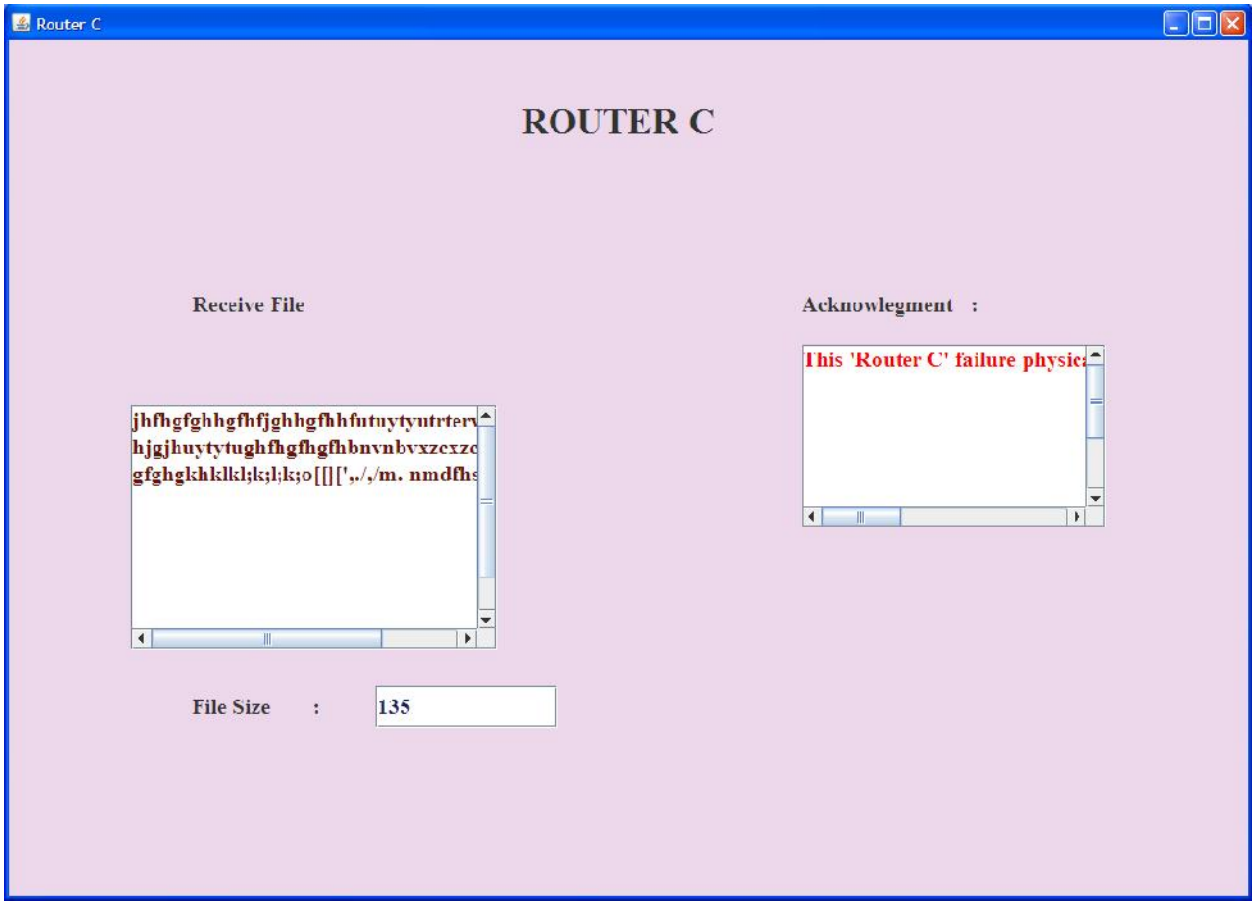


Fig.8.3: Screen shot for Router C

This screen shot receive the file and file size. It shows an acknowledgement of "This 'Router C' failure physical problem occurred. Therefore this data transfer using subpath".

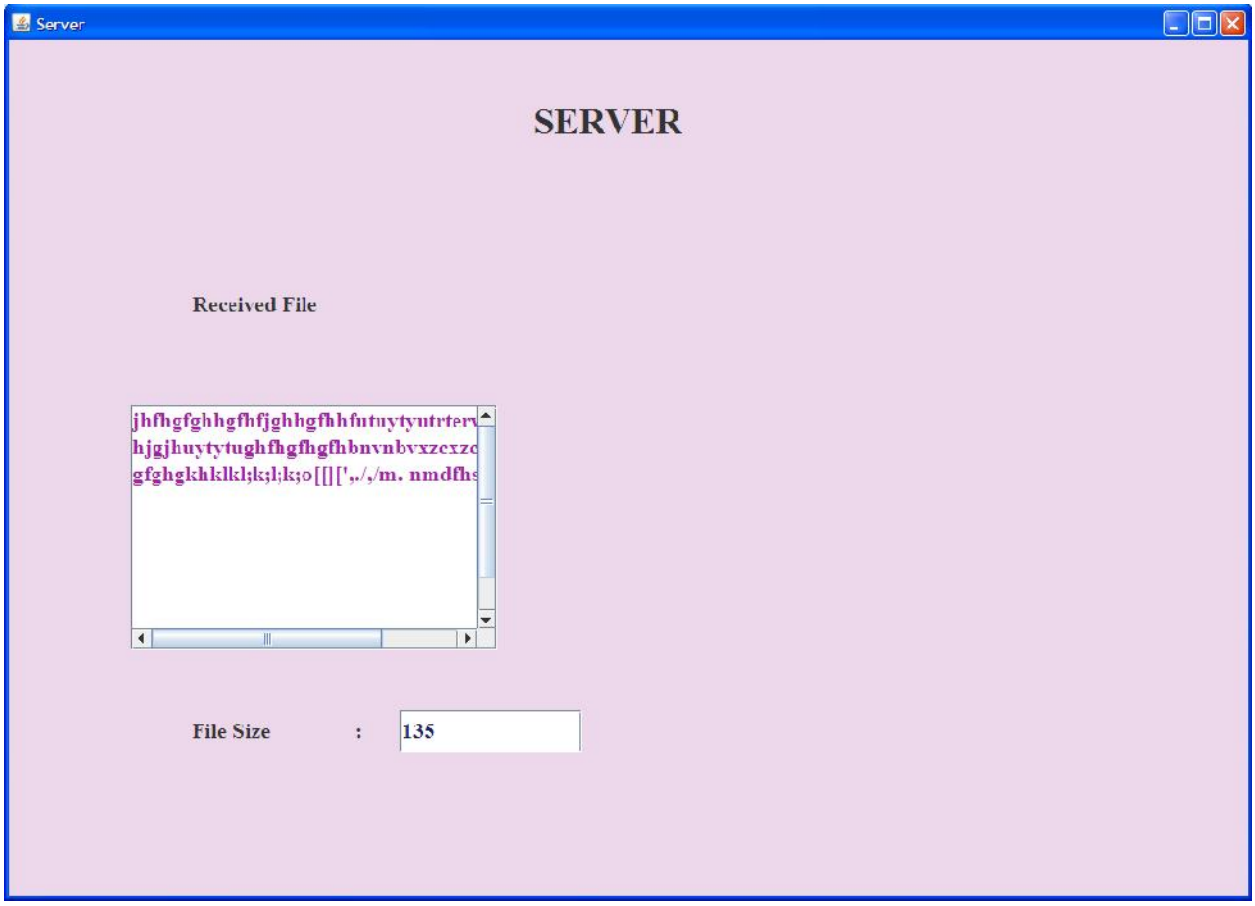


Fig.8.4: Screen shot for Server

In this screen shot the server receives the file and file size from client by using routers. The client1 send file to the server using routers. The server displays the file data and file size.

CHAPTER: 9

CONCLUSION

9: CONCLUSION

This project presented Multiple Routing Configurations as an approach to achieve fast recovery in IP networks. MRC is based on providing the routers with additional routing configurations, allowing them to forward packets along routes that avoid a failed component. MRC guarantees recovery from any single node or link failure in an arbitrary bi-connected network. By calculating backup configurations in advance, and operating based on locally available information only, MRC can act promptly after failure discovery.

MRC operates without knowing the root cause of failure, i.e., whether the forwarding disruption is caused by a node or link failure. This is achieved by using careful link weight assignment according to the rules we have described. The link weight assignment rules also provide basis for the specification of a forwarding procedure that successfully solves the last hop problem.

The performance of the algorithm and the forwarding mechanism has been evaluated using simulations. This project shown that MRC scales well: 3 or 4 backup configurations is typically enough to isolate all links and nodes in our test topologies. MRC backup path lengths are comparable to the optimal backup path lengths—MRC backup paths are typically zero to two hops longer. This project evaluated the effect MRC has on the load distribution in the network while traffic is routed in the backup configurations, and we have proposed a method that minimizes the risk of congestion after a link failure if we have an estimate of the demand matrix. In the network, this approach gave a maximum link load after the worst case link failure that was even lower than after a full IGP re-convergence on the altered topology. MRC thus achieves fast recovery with a very limited performance penalty.

CHAPTER: 10

BIBILOGRAPHY

10: BIBLIOGRAPHY

Books Referred:

- Professional Java Network Programming
- Java Complete Reference
- Data Communications and Networking, by Behrouz A Forouzan.
- Computer Networking: A Top-Down Approach, by James F. Kurose.

Websites:

- http://www.networkcomputing.com/
- http://www.java2s.com/
- http://java.sun.com

APPENDIX

Abbreviations Used

API	:	Application Programming Interface
BGP	:	Border Gateway Protocol
FLP	:	Fast Link Pulse
GUI	:	Graphical User Interface
IGP	:	Interior Gateway Protocol
IP	:	Internet Protocol
JIT	:	Just In Time
JVM	:	Java Virtual Machine
LAN	:	Local Area Network
MRC	:	Multiple Routing Configuration
ODBC	:	Open Data Base Connectivity
OSPF	:	Open Shotest Path first
TCP	:	Transmission Control Protocol
UML	:	Unified Modeling Language