CHAPTER: 1

INTRODUCTION

1. INTRODUCTION

1.1 Brief Information of Project

Modern IP network services provide for the simultaneous digital transmission of voice, video, and data. These services require congestion control protocols and algorithms which can solve the packet loss parameter can be kept under control. Congestion control is therefore, the cornerstone of packet switching networks. It should prevent congestion collapse, provide fairness to competing flows and optimize transport performance indexes such as throughput, delay and loss. The literature abounds in papers on this subject; there are papers on high-level models of the flow of packets through the network, and on specific network architecture .

Despite this vast literature, congestion control in telecommunication networks struggles with two major problems that are not completely solved. The first one is the time-varying delay between the control point and the traffic sources. The second one is related to the possibility that the traffic sources do not follow the feedback signal. This latter may happen because some sources are silent as they have nothing to transmit. Originally designed for a cooperative environment. It is still mainly dependent on the TCP congestion control algorithm at terminals, supplemented with load shedding at congestion links. This model is called the Terminal Dependent Congestion Control case.

Core-Stateless Fair Queuing (CSFQ) set up an open- loop control system at the network layer, which inserts the label of the flow arrival rate onto the packet header at edge routers and drops the packet at core routers based on the rate label if congestion happens. CSFQ is the first to achieve approximate fair bandwidth allocation among flows with O(1) complexity at core routers.

According to Cache Logic report, P2P traffic was 60% of all the internet traffic in 2004, of which Bit-Torrent was responsible for about 30% of the above, although the report generated quite a lot of discussions around the real numbers. In networks with P2P traffic, CSFQ can provide fairness to competing flows, but unfortunately it is not what end-users and operators really want. Token-Based Congestion Control (TBCC) restricts the total token resource

consumed by an end-user. So, no matter how many connections the end-user has set up, it cannot obtain extra bandwidth resources when TBCC is used.

In this paper a new and better mechanism for congestion control with application to Packet Loss in networks with P2P traffic is proposed. In this new method the edge and the core routers will write a measure of the quality of service guaranteed by the router by writing a digital number in the Option Field of the datagram of the packet. This is called a token. The token is read by the path routers and interpreted as its value will give a measure of the congestion especially at the edge routers. Based on the token number the edge router at the source, thus reducing the congestion on the path. In Token-Limited Congestion Control (TLCC), the inter-domain router restricts the total output token rate to peer domains. When the output token rate exceeds the threshold, TLCC will decreases the Token-Level of output packets, and then the output token rate will decrease.

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 n company strength. Once these things r 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 r 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.

Background concepts:

Congestion control of the best-effort service in the Internet was originally designed for a cooperative environment. It is still mainly dependent on the TCP congestion control algorithm at terminals, supplemented with load shedding at congestion links. This model is called the Terminal Dependent Congestion Control case. Although routers equipped with Active Queue

Management such as RED can improve transport performance, they are neither able to prevent congestion collapse nor provide fairness to competing flows.

In order to enhance fairness in high speed networks, Core-Stateless Fair Queuing (CSFQ) set up an open- loop control system at the network layer, which inserts the label of the flow arrival rate onto the packet header at edge routers and drops the packet at core routers based on the rate label if congestion happens. CSFQ is the first to achieve approximate fair bandwidth allocation among flows with O(1) complexity at core routers.

In networks with P2P traffic, CSFQ can provide fairness to competing flows, but unfortunately it is not what end-users and operators really want. Token-Based Congestion Control(TBCC) restricts the total token resource consumed by an end-user. So, no matter how many connections the end-user has set up, it cannot obtain extra bandwidth resources when TBCC is used.

The Self-Verifying CSFQ tries to expand CSFQ across the domain border. It randomly selects a flow, re-estimates the flow's rate, and checks whether the re-estimated rate inconsistent with the label on the flow's packet. Consequently Self-Verifying CSFQ will put a heavy load on the border router and makes the weighted CSFQ null and void.

In this paper a new and better mechanism for congestion control with application to Packet Loss in networks with P2Ptraffic is proposed. In this new method the edge and the core routers will write a measure of the quality of service guaranteed by the router by writing a digital number in the Option Field of the datagram of the packet. This is called a token. The token is read by the path routers and interpreted as its value will give a measure of the congestion especially at the edge routers. Based on the token number the edge router at the source's edge point will shape the traffic generated by the source, thus reducing the congestion on the path. In Token-Limited Congestion Control (TLCC) [9], the inter-domain router restricts the total output token rate to pee r domains. When the output token rate exceeds the threshold, TLCC will decreases the Token-Level of output packets, and then the output token rate will decrease.

Similarly to CSFQ and TBCC, TLCC uses also the iterative algorithm to estimate the congestion level of its output link, and requires a long period of time to reach a stable state. With bad parameter configuration, TLCC may cause the traffic to fall into an oscillated process. The window size of TCP flows will always increase when acknowledge packets are received, and the congestion level will increase at the congested link. At congestion times many flows will lose their packets. Then, the link will be idle and the congestion level will decrease. The two steps may be repeated alternately, and then the congestion control system will never reach stability. To solve the oscillation problem, the Stable Token-Limited Congestion Control (STLCC) is introduced. It integrates the algorithms of TLCC and XCP altogether. In STLCC, the output rate of the sender is controlled according to the algorithm of XCP, so there is almost no packet lost at the congested link. At the same time, the edge router allocates all the access token resource to the incoming flows equally. When congestion happens, the incoming token rate increases at the core router, and then the congestion level of the congested link will also increase. Thus STLCC can measure the congestion level analytically, allocate network resources according to the access link, and further keep the congestion control system stable.

CHAPTER: 3

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 Introduction

Systems analysis is the study of sets of interacting entities, including computer systems analysis. This field is closely related to requirements analysis or operations research. It is also an explicit formal inquiry carried out to help someone (referred to as the decision maker) identify a better course of action and make a better decision than he might otherwise have made.

The section 3.2 explains about the description of the existing system and what are the problems of the existing system. Section 3.3 explains about the proposed system and advantages of the proposed system. In 3.4 explains about the Sample architecture of the system. In the section 3.5 it describes about entire modules of the system. Similarly 3.6 and 3.7 are discussing about the feasibility study and requirement specifications.

3.2 Existing System

In the existing system, the sender sends the packets without the intermediate station. The data packets has been losses many and time is wasted. Retransmission of data packets is difficulty.

PROBLEM IN EXISTING SYSTEM:

1) TCP(Transaction Control Protocol) is insufficient to handle congestion control. This protocol neither able to prevent congestion collapse nor provide fairness to competing flows.

2) CSFQ(Core-Stateless Fair Queuing) was designed as an open-loop controller to provide the fair best effort service for supervising the per-flow bandwidth consumption and has become helpless when the P2P flows started to dominate the traffic of the Internet**.**

3) TBCC(Token Based Congestion Control) restricts the total tokens consumed by the end user. But it cannot obtain extra bandwidth resources when TBCC used.

3.3 Proposed System

Modern IP network services provide for the simultaneous digital transmission of voice, video, and data. These services require congestion control protocols and algorithms which can solve the packet loss parameter can be kept under control. Congestion control is therefore, the cornerstone of packet switching networks. It should prevent congestion collapse, provide fairness to competing flows and optimize transport performance indexes such as throughput, delay and loss. The literature abounds in papers on this subject; there are papers on high-level models of the flow of packets through the network, and on specific network architecture.

ADVANTAGES OF PROPOSED SYSTEM:

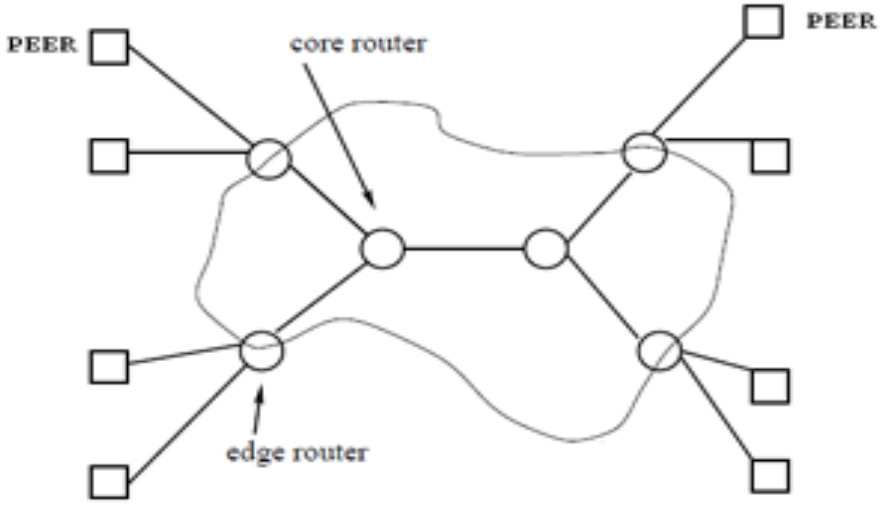
1)Stable Token-Limited Congestion Control (STLCC) is introduced as new protocols which appends inter-domain congestion control to TBCC and make the congestion control system to be stable.

2) STLCC is able to shape output and input traffic at the inter-domain link with O(1) complexity.

3) STLCC produces a congestion index, pushes the packet loss to the network edge and improves the network performance.

4) It is deployable in the Internet without any IP protocols modifications and preserves also the packet datagram.

ARCHITECTURE



3.3 SAMPLE ARCHITECTURE

3.4 Modules Description

Network Congestion:

- Congestion occurs when the number of packets being transmitted through the network approaches the packet handling capacity of the network
- Congestion control aims to keep number of packets below level at which performance falls off dramatically

Stable Token Limit Congestion Control (STLCC):

STLCC is able to shape output and input traffic at the inter-domain link with O(1) complexity. STLCC produces a congestion index, pushes the packet loss to the network edge and improves the network performance. To solve the oscillation problem, the Stable Token-Limited Congestion Control (STLCC) is introduced. It integrates the algorithms of TLCC and XCP altogether. In STLCC, the output rate of the sender is controlled according to the algorithm of XCP, so there is almost no packet lost at the congested link. At the same time, the edge router allocates all the access token resource to the incoming flows equally. When congestion happens, the incoming token rate increases at the core router, and then the congestion level of the congested link will also increase. Thus STLCC can measure the congestion level analytically, allocate network resources according to the access link, and further keep the congestion control system stable.

Token:

In this paper a new and better mechanism for congestion control with application to Packet Loss in networks with P2P traffic is proposed. In this new method the edge and the core routers will write a measure of the quality of service guaranteed by the router by writing a digital number in the Option Field of the datagram of the packet. This is called a token. The token is read by the path routers and interpreted as its value will give a measure of the congestion especially at the edge routers. Based on the token number the edge router at the source, thus reducing the congestion on the path.

Core Router:

A core router is a router designed to operate in the Internet Backbone or core. To fulfill this role, a router must be able to support multiple telecommunications interfaces of the highest speed in use in the core Internet and must be able to forward IP packets at full speed on all of them. It must also support the routing protocols being used in the core. A core router is distinct from an edge routers.

Edge Router:

Edge routers sit at the edge of a backbone network and connect to core routers. The token is read by the path routers and interpreted as its value will give a measure of the congestion especially at the edge routers. Based on the token number the edge router at the source, thus reducing the congestion on the path.

3.5 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.

3.6 Requirements Specifications

3.6.1 Functional Requirements

The Scope of the Work

At the beginning a series of protocols like TCP,CSFQ are used to controlling the network congestion. But these are become helpless when the P2P flows started to dominate the traffic of the internet.

To solve this congestion problem we are using the STLCC protocol. It is combination of XCP and TLCC. This is deployable in the internet without any IP protocol modification.

The scope of the product

1. STLCC pushes the packet loss to the network edge and improves the network performance.
2. It provides fair bandwidth to the end user without any congestion.
3. It easily deployable to the present internet.

Functional requirements

- **Description of data to be entered into the system:**

The sender can send the text file through peer. It is divided into packets.
The receiver receives those packets based on token number.

- **Description of operation of each screen:**

The sender sends or receiver receives the data through its Edge routers with help of the Core router.

- **Description of work flow:**

In this first the user sends their data through the peer. At this data is divided into different number of packet. Each packet has token number. After that it is passed to Edge router with the help of the Core router. At receiver side the other user who wants the data receive the data.

- **Who can enter the data to the system:**

On the sender can send the data to the system.

- **Description of Output:**

Only the receiver receives the data. The data is collected in the form of packets. These packets are arranged in the order based on the token number.

3.6.2 Minimum Hardware Requirements

- Hard Disk : 40 GB
- RAM : 256 MB

3.6.3 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.

The section 4.2 mainly describes about the UML diagrams that describes about the diagrams that are used to represent the project. The section 4.2.1 describes about the use case diagram. Similarly the sections 4.2.2, 4.2.3, 4.2.4, and 4.2.5 describes about class, sequence, activity and data flow diagrams respectively.

4.2 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.

4.2.1 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.

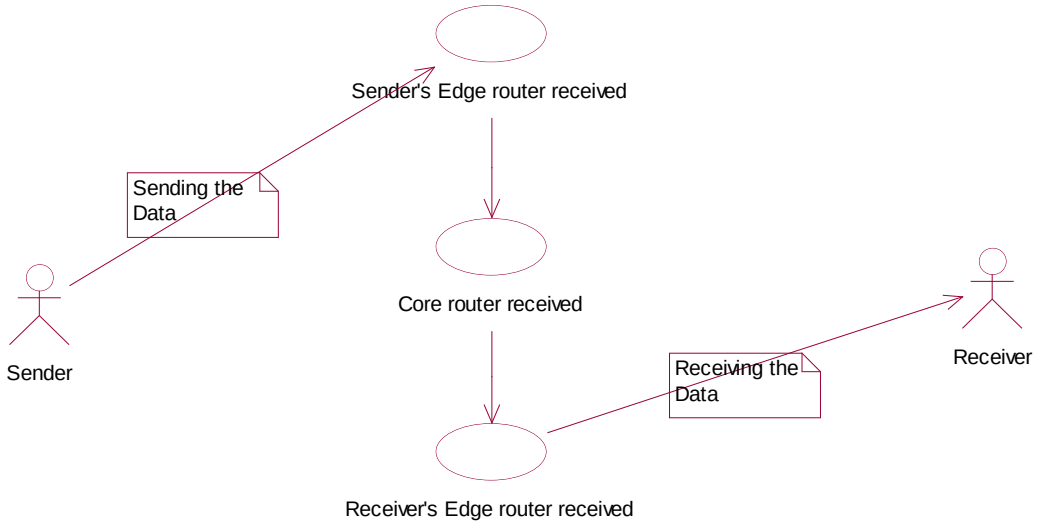


Fig 4.2.1 Use Case Diagram

4.2.2 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.

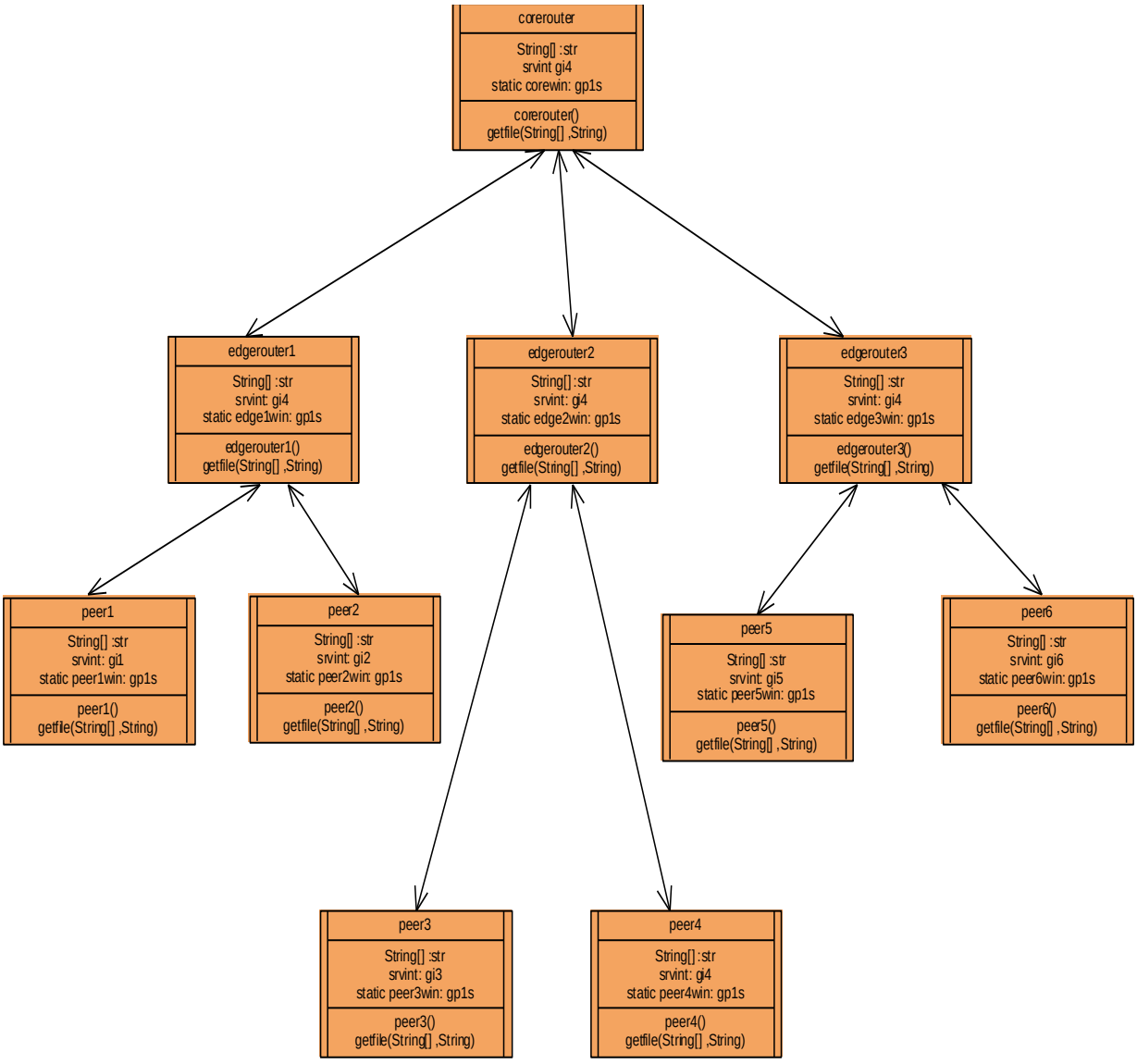


Fig 4.2.2 Class Diagram

4.2.3 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.

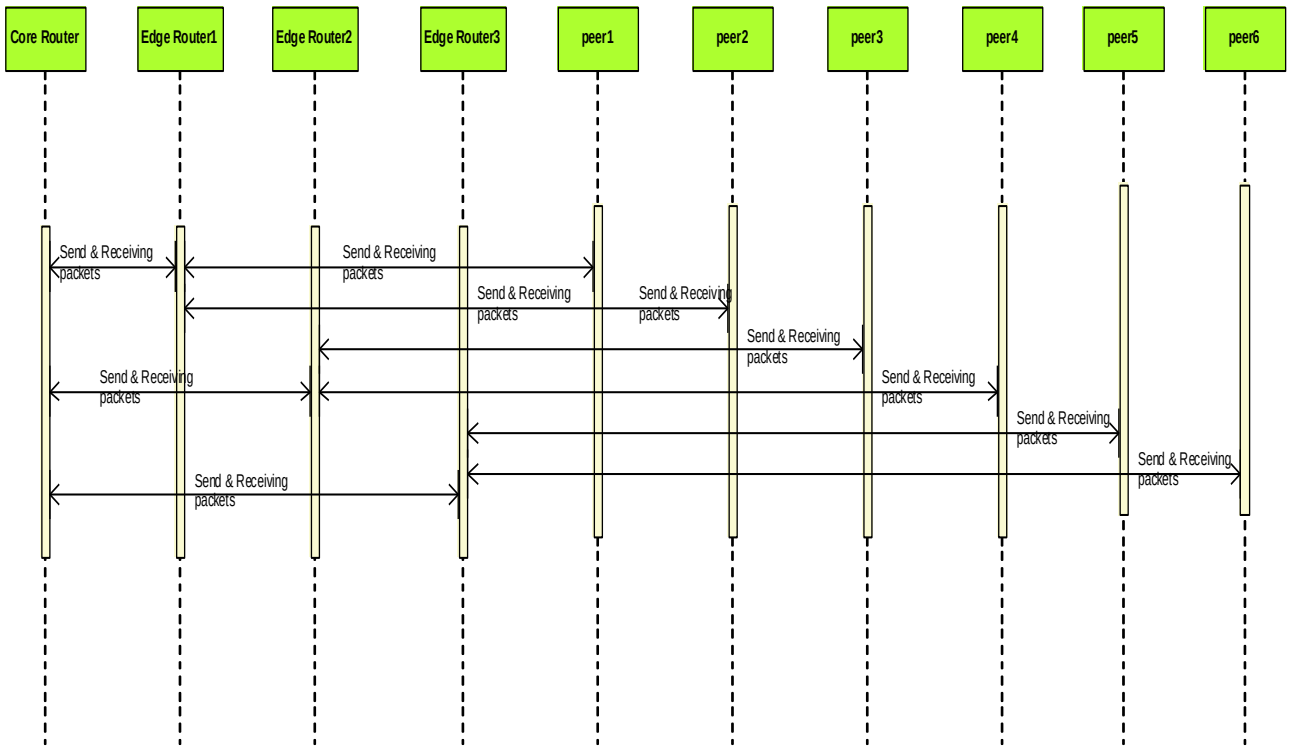


Fig.4.2.3 Sequence diagram

4.2.4 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 then 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.

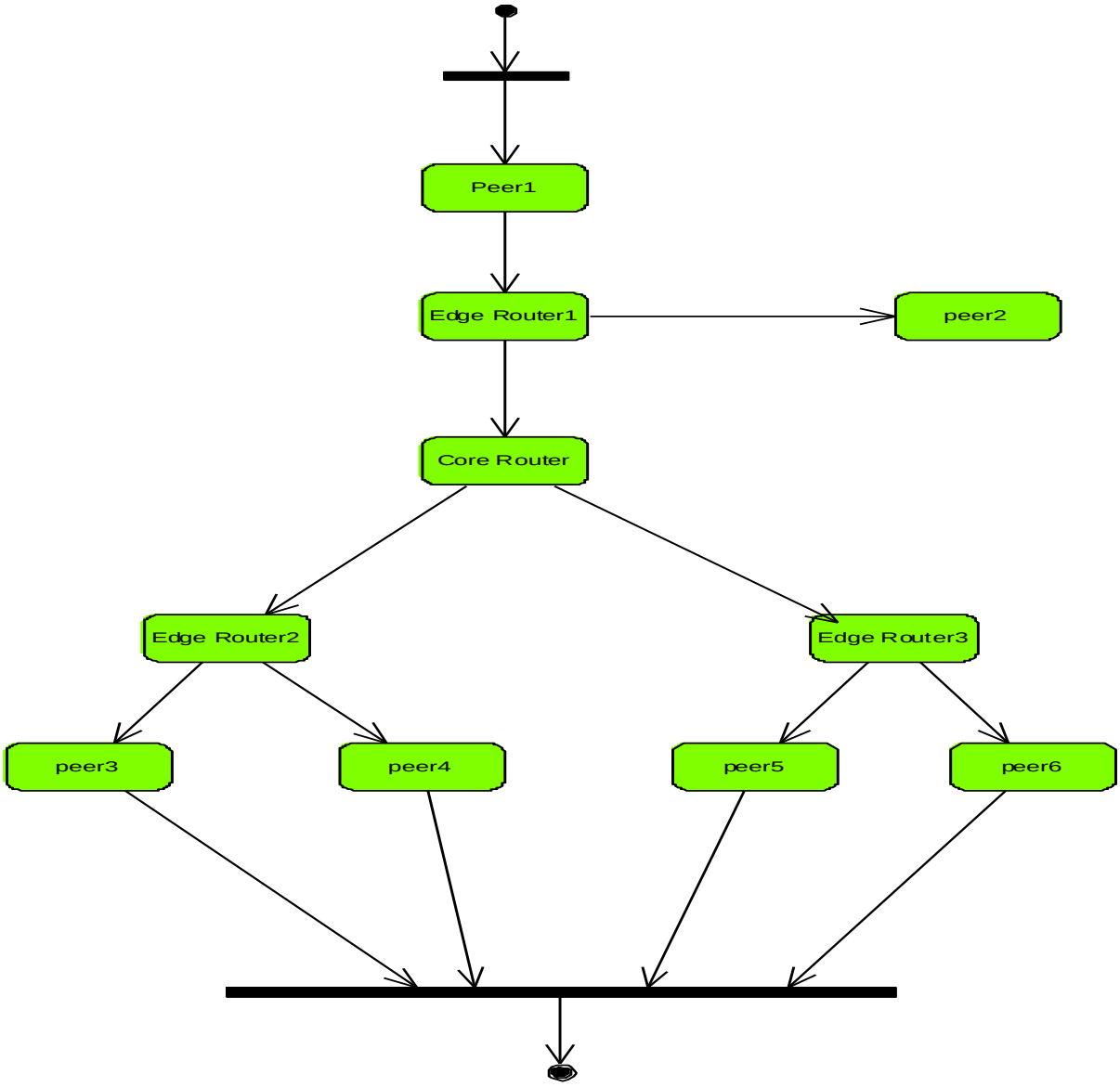


Fig.4.2.4 Activity Diagram

4.2.5 Data Flow Diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on these data, and the output data is generated by the system.

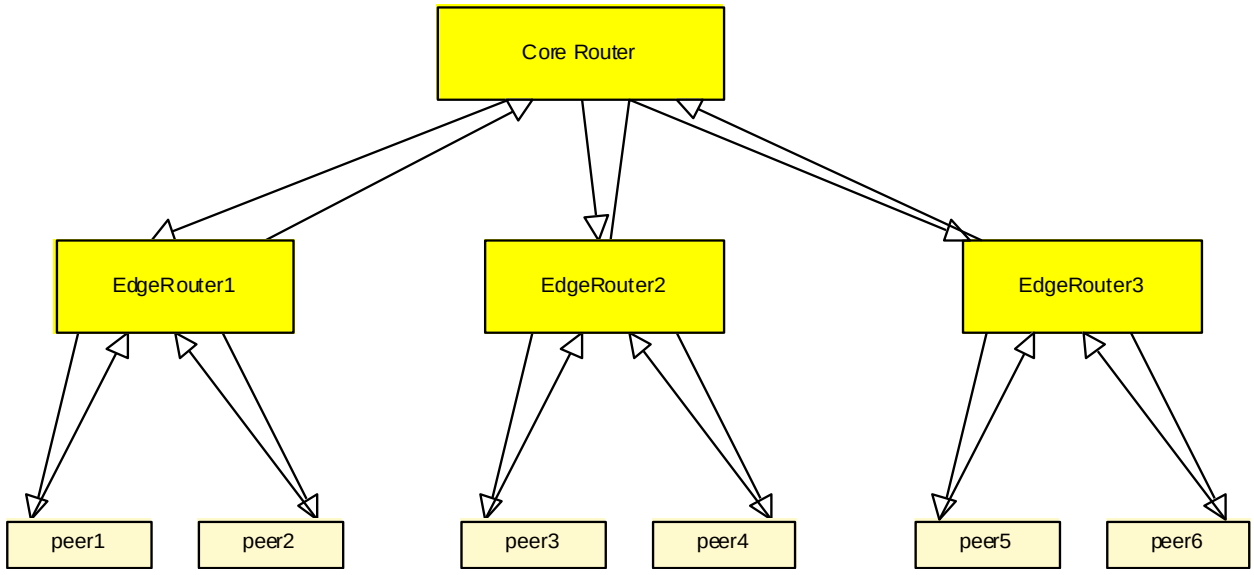


Fig.4.2.5 Data Flow Diagram

CHAPTER: 5

TECHNOLOGY DESCRIPTION

5. TECHNOLOGY DESCRIPTION

Java Technology

Java technology is both a programming language and a platform.

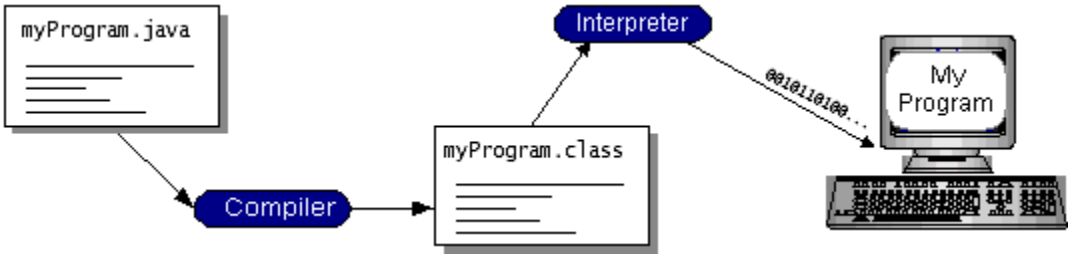
The Java Programming Language

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.



5.1 Running of java program

You can think of Java byte codes 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 byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes 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.

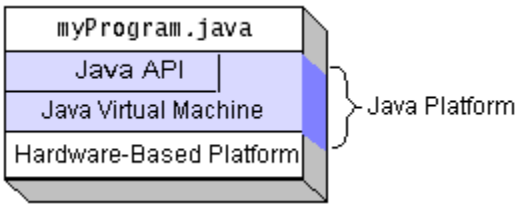
The Java platform has two components:

- The Java Virtual Machine (Java VM)
- The Java Application Programming Interface (Java API)

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.



5.2 Java internal structure

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 byte code compilers can bring performance close to that of native code without threatening portability.

The most common types of programs written in the Java programming language are applets and applications. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized

program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**:

  Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

- **Applets**:

  The set of conventions used by applets.

- **Networking**:

  URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.

- **Internationalization**:

  Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.

- **Security**:

  Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.

- **Software components**:

  Known as JavaBeans, can plug into existing component architectures.

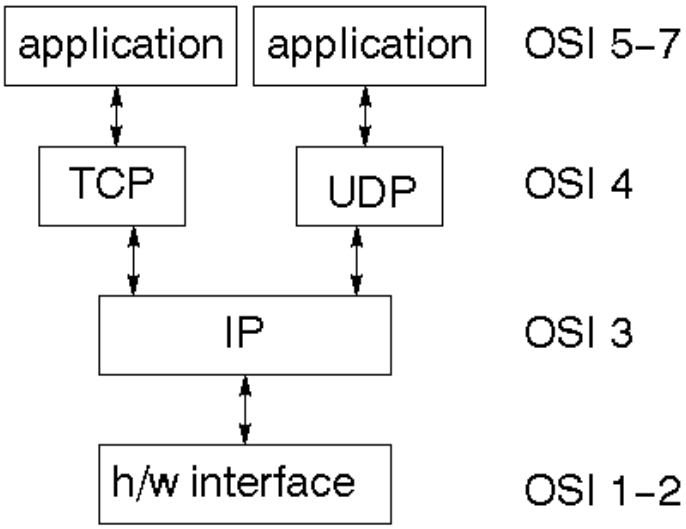
- **Object serialization**:

  Allows lightweight persistence and communication via Remote Method Invocation (RMI).

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



5.3 TCP/IP stack

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

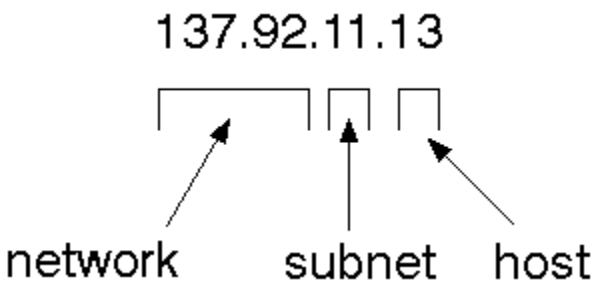
Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address



The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be AF_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

JFree Chart

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types.

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

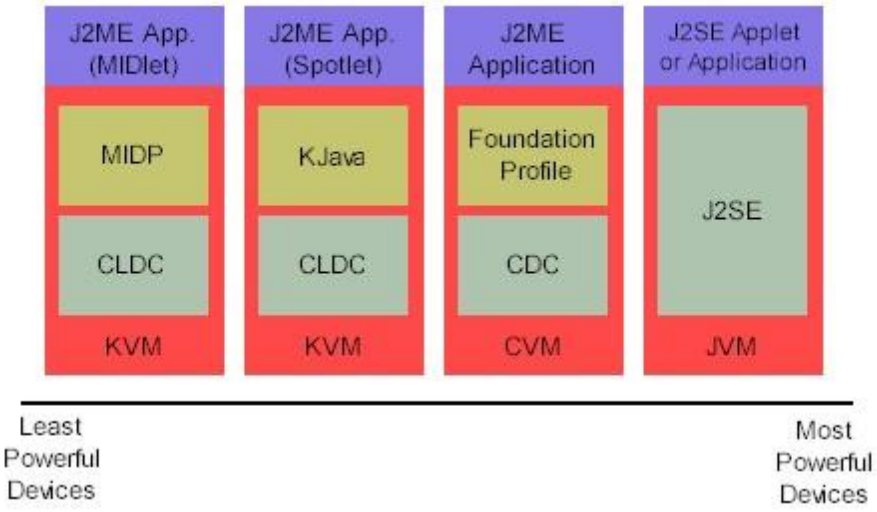
4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

J2ME (Java 2 Micro edition):-

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

1. General J2ME architecture



5.4 J2ME architecture

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

2. Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role preverification plays in this process.

3. Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

- Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.
- Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.
- Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

4. Configurations overview

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

- Connected Limited Device Configuration (CLDC) is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.
- Connected Device Configuration (CDC) is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

5. J2ME profiles

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

Profile 1: KJava

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

- java.lang
- java.io
- java.util
- javax.microedition.io
- javax.microedition.lcdui
- javax.microedition.midlet
- javax.microedition.rms

CHAPTER: 6

TESTING

6. 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.

TYPES OF TESTS

6.1 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.

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.

6.2 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 satisfaction, 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.

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.

6.3 Functional testing

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.

6.4 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.

6.5 System Testing

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 in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is 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.

CHAPTER: 7

SCREEN SHOTS

7. SCREEN SHOTS

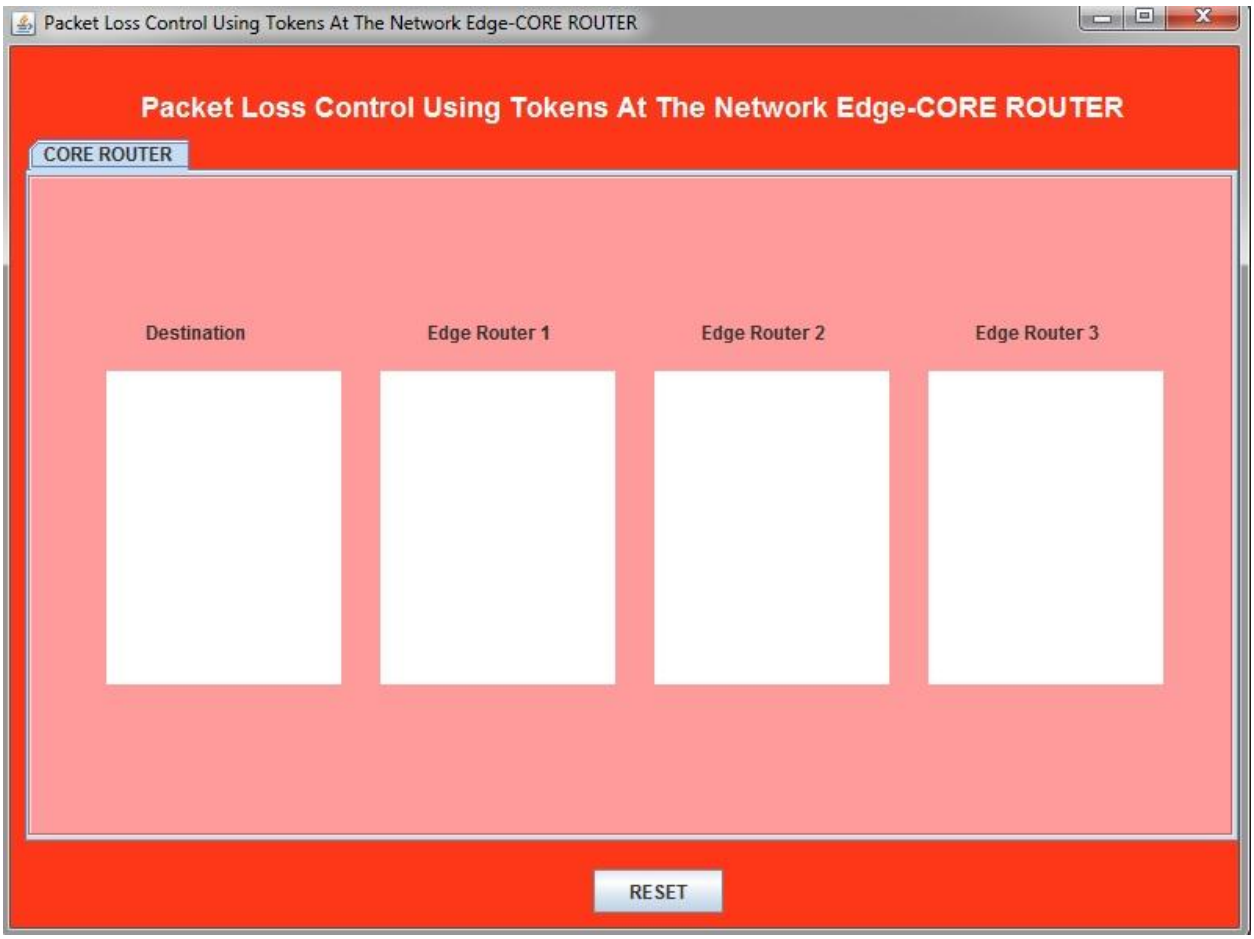


Fig 7.1: Screen shot of Core Router

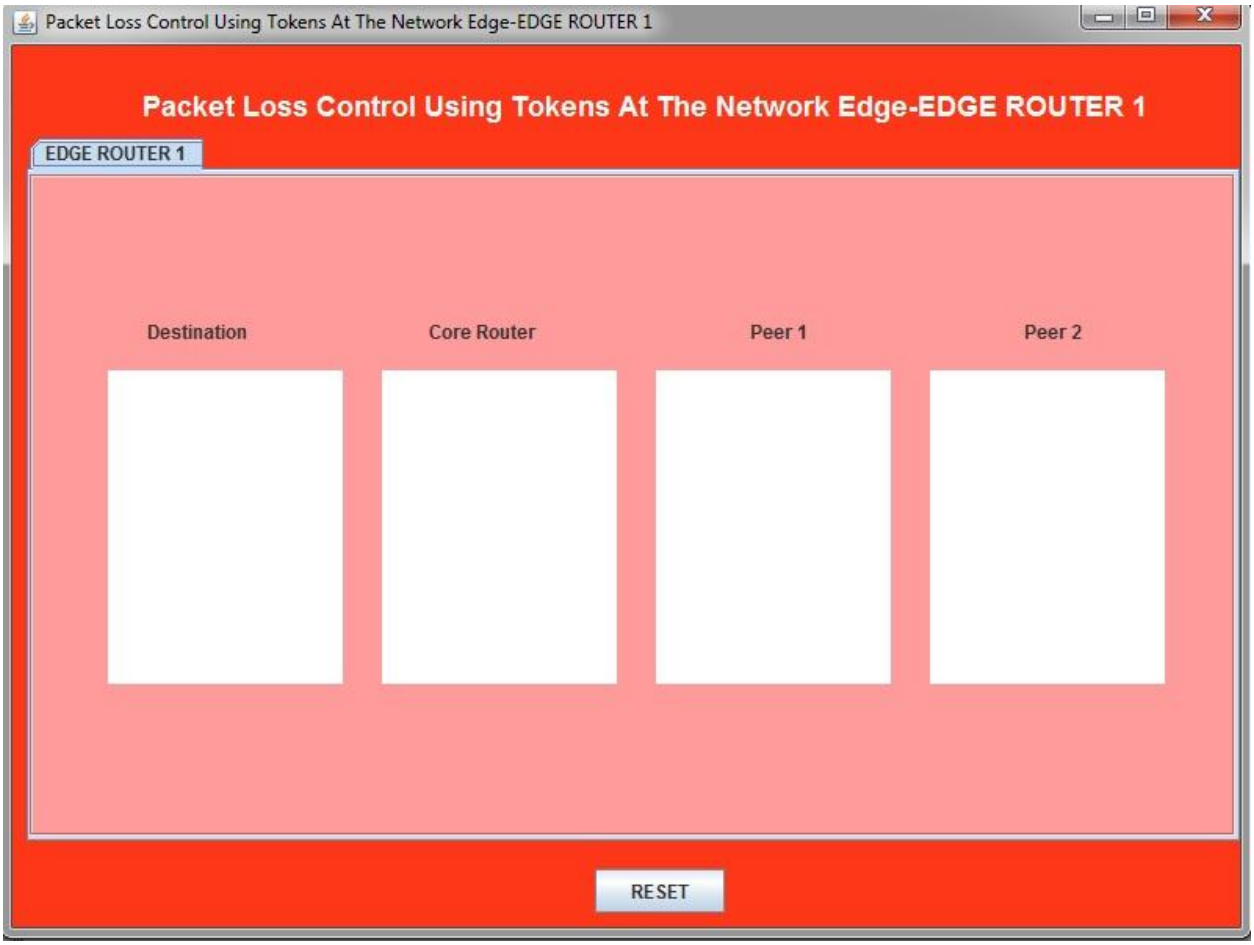


Fig 7.2: Screen shot of Edge Router

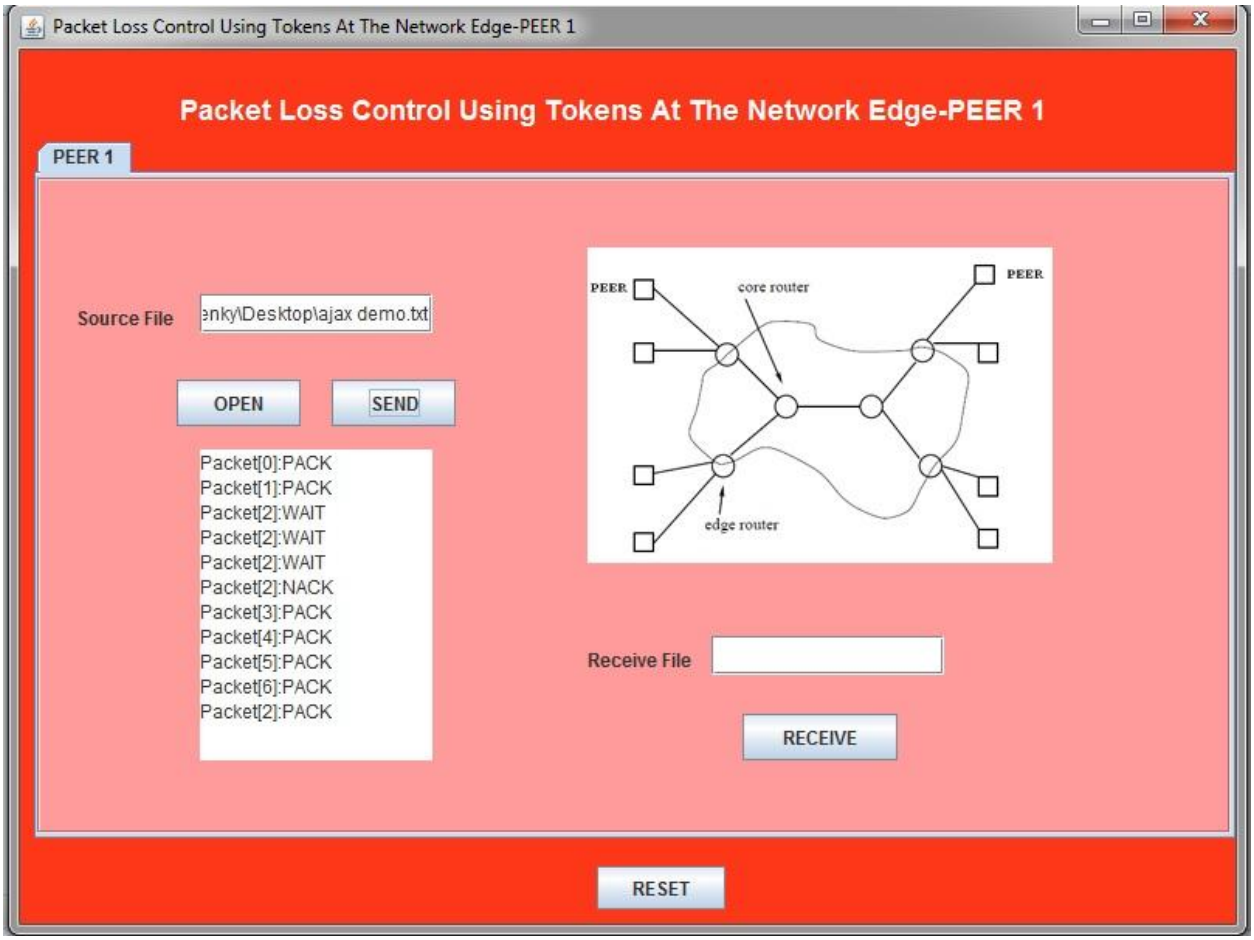


Fig 7.3: Screen shot of Peer After Sending Data

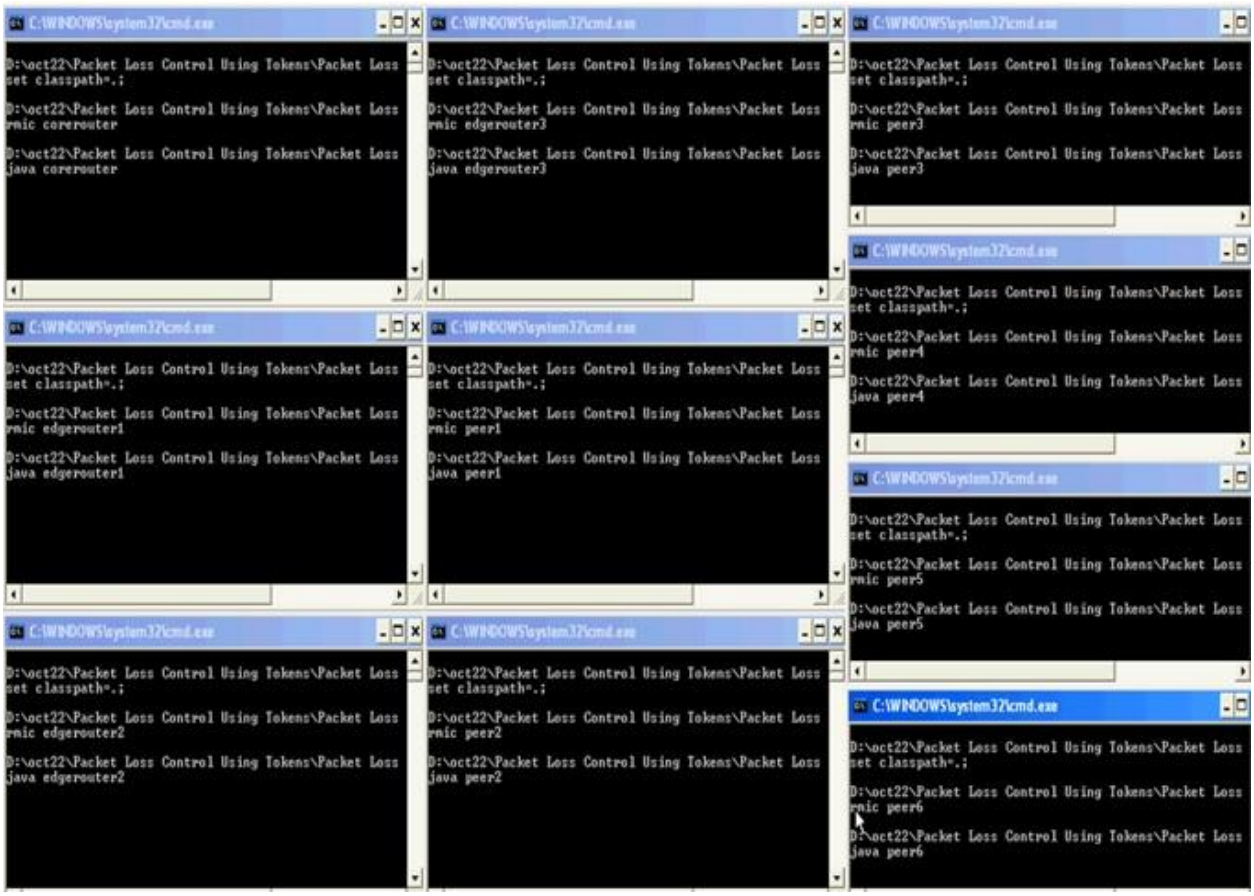


Fig 7.4: Screen shot of Execution

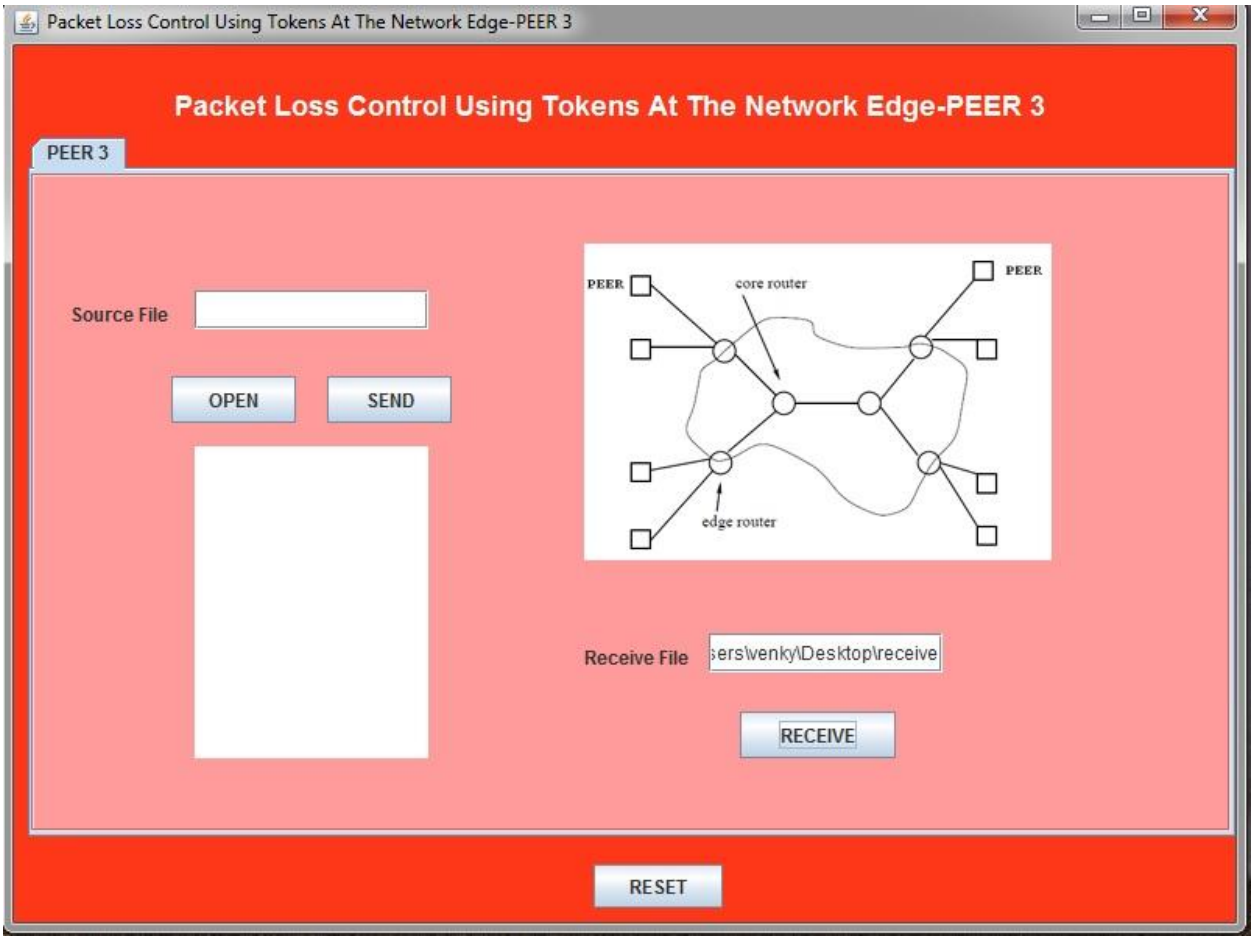


Fig 7.5: Screen shot of Peer After Receiving Data

CHAPTER: 8

CONCLUSION

8. CONCLUSION

This paper is organized as follows. In the architecture of Token-Based Congestion Control (TBCC), which provides fair bandwidth allocation to end-users in the same domain will be introduced. It evaluates two congestion control algorithms CSFQ and TBCC. STLCC is presented and the simulation is designed to demonstrate its validity. The Unified Congestion Control Model which is the abstract model of CSFQ, Re-feedback and STLCC. The simple version of STLCC is proposed, which can be deployed on the current Internet. Finally, conclusions will be given. To inter-connect two TBCC domains, the inter-domain router is added to the TBCC system. To support the SKA arrangement, the inter-domain router should limit its output token rate to the rate of the other domains and police the incoming token rate from peer domains.

CHAPTER: 9

BIBILOGRAPHY

9. BIBLIOGRAPHY

[1] Andrew S. Tanenbaum, Computer Networks, Prentice-Hall International, Inc.

[2] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance, ACM/IEEE Transactions on Networking, August 1993.

[3] Ion Stoica, Scott Shenker, Hui Zhang, "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks", In Proc. of SIGCOMM, 1998.

[4] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In Proc. of SIGCOMM, 2004.

[5] Zhiqiang Shi, Token-based congestion control: Achieving fair resource allocations in P2P networks, Innovations in NGN: Future Network Services, 2008. K-INGN 2008. First ITU-T Kaleidoscope Academic Conference.

[6] I. Stoica, H. Zhang, S. Shenker Self-Verifying CSFQ, in Proceedings of INFOCOM, 2002.

[7] Bob Briscoe, Policing Congestion Response in an Internetwork using Refeedback, In Proc. ACM SIGGCOMM05, 2005,

[8] Bob Briscoe,Re-feedback:Freedom with Accountability for Causing Congestion in a Connectionless Internetwork.

[9] Zhiqiang Shi, Yuansong Qiao, Zhimei Wu, Congestion Control with the Fixed Cost at the Domain Border, Future Computer and Communication (ICFCC),2010.

[10] Dina Katabi, Mark Handley, and Charles Rohrs, "Internet Congestion Control for Future High Bandwidth-Delay Product Environments."ACM Sigcomm 2002, August 2002.