P-1641

INTEGRATED CONGESTION MANAGEMENT FOR LAN

A PROJECT REPORT

Submitted by

IKRAM AHAMED.M	**71202205017**
PRABAKAR.N	**71202205028**
SARAVANA KUMAR N.S	**71202205041**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2006

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report " **INTEGRATED CONGESTION MANAGEMENT FOR LAN** " is the bonafide work of " **M.IKRAM AHAMED, N. PRABAKAR,N.S.SARAVANA KUMAR** " who carried out the project under my supervision.

SIGNATURE
Dr.G.Gopalsamy Ph.D.,
HEAD OF THE DEPARTMENT

Information technology
Kumaraguru College of Technology
Coimbatore-641006

SIGNATURE
Dr.G.Gopalsamy Ph.D.,
SUPERVISOR

Information technology
Kumaraguru College of Technology
Coimbatore-641006

The candidates with University Register Nos. **71202205017, 71202205028, 71202205041** were examined by us in the project viva-voice examination held on 26.04.06.

INTERNAL EXAMINER

EXTERNAL EXAMINER 26/4/06

DECLARATION

We

IKRAM AHAMED.M	**71202205017**
PRABAKAR.N	**71202205028**
SARAVANAKUMAR.N.S	**71202205041**

Declare that the project entitled "**INTEGRATED CONGESTION MANAGEMENT FOR LAN**", submitted in partial fulfillment to Anna University as the project work of Bachelor Of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and the guidance of **Dr.G.Gopalsamy, Ph.D, HOD.,** Department of Information Technology, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

Date: 26.04.06

[IKRAM AHAMED .M]

[PRABAKAR.N]

[SARAVANAKUMAR.N.S]

Project Guided by

[Dr.G.Gopalsamy Ph.D]

ACKNOWLEDGEMENT

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with. We express our deep gratitude to our Principal **Dr.K.K Padmanabhan B.Sc. (Engg).**, **M.Tech.**, **Ph.D.**, for ushering us in the path of triumph.

We are always thankful to **Dr.Gopalsamy Ph.D.**, our beloved guide and the Head of the Department, Information technology, whose consistent support and enthusiastic involvement helped us a great deal. As a token of our esteem and gratitude, we honor him assistance towards this cause.

We also thank our project coordinator **Prof.K.R.Baskaran, B.E., M.S.**, and our beloved class advisor **Mrs Sugandhi**, for their invaluable assistance.

We feel proud to pay our respectful thanks to our **Parents** for their enthusiasm and encouragement and also we thank our friends who have associated themselves to bring out this project successfully.

ABSTRACT

This project presents a novel framework for managing network congestion from an end-to-end perspective. Our work is motivated by trends in traffic patterns that threaten the long term stability of LAN network. These trends include use of independent concurrent flows by various applications and the increasing use of transport protocols and applications that do not adapt to congestion. We represent an end-system architecture centered around a Congestion Manager (CM) that ensures proper congestion behavior and allows applications to easily adapt to network congestion. Our framework integrates congestion management across all applications and transport protocols. The CM maintains congestion parameters and exposes an API to enable applications top learn about network characteristics, pass information to the CM and schedule data transmission. Internally it uses a scheduler to regulate transmissions.

TABLE OF CONTENTS

S.NO	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF ABBREVIATIONS	viii
1	**INTRODUCTION**	1
	1.1 Project scope	2
2	**LITERATURE REVIEW**	3
	2.1 Development environment	4
	2.1.1 Hardware specification	4
	2.1.2 Software specification	4
	2.1.3 Software study	4
	2.1.3.1 Introduction to java	4
	2.1.3.2 About JCreator	7
	2.1.3.3 Introduction to socket programming	11
	2.2 Congestion in internet	12
	2.2.1 Current situation	12
	2.2.2 Reasons for internet traffic	12
	2.3 What is congestion	13
	2.4 Proposed system	13

	2.4.1 Creating congestion	13
	2.4.2 Controlling congestion	13
3	**SYSTEM STUDY**	**15**
	3.1 Server module	15
	3.2 Server Interface module	15
	3.3 Client module	15
	3.4 Client Interface module	15
	3.5 CPU module	16
	3.5 File Transfer	17
4	**SYSTEM TESTING**	**18**
	4.1 Testing objective	18
	4.2 Testing principle	18
	4.3 Levels of testing	19
	4.3.1 Unit testing	19
	4.3.2 Integration testing	20
	4.3.1 Validation testing	21
5	**FUTURE ENHANCEMENT**	**22**
6	**CONCLUSION**	**23**
7	**APPENDIX**	**24**
8	**REFERENCES**	**54**

LIST OF ABBREVIATIONS

TCP – Transmission Control Protocol

CM – Congestion Manager

UDP – User Datagram Protocol

IP – Internet Protocol

INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

As the internet users are increasing day-by-day in the information age, we have no doubt in the scope of our project which aims at managing the congestion in the Internet. It will have a huge impact among the users of internet.

Integrated congestion management will be able to manage the congestion in the inter network independent of any protocol (TCP/UDP).The major factor for the occurrence of congestion is over utilization of the CPU. Most of the internet connections are characterized by short TCP connection due to which the internet is unable to solve the congestion problems. The download accelerators also have contributed to the congestion in the network.

The last reason is that real time applications are forced to use UDP for ease of performance of that application sacrificing the congestion control mechanisms embedded in TCP. All these add up to the congestion of the network. Our project mainly deals with the management of congestion occurring due to the over utilization of the CPU.

LITERATURE REVIEW

2.1 DEVELOPMENT ENVIRONMENT

2.1.1 HARDWARE SPECIFICATION

- Processor - Pentium IV 2.6 GHz
- System RAM - 512 MB
- Hard Disk - 40 GB
- Mouse - Microsoft PS/2 mouse
- Keyboard - Windows keyboard
- Monitor - Color monitor

2.1.2 SOFTWARE SPECIFICATION

- Languages used - Java, Visual C++
- Operating system - Windows 9x/ME/2000/NT/Xp
- IDE - Jcreator

2.1.3 SOFTWARE STUDY

2.1.3.1 INTRODUCTION TO JAVA:

Java is an object-oriented programming language with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets. Because of its rich set of API"s, similar to Macintosh and Windows. And its platform independence, Java can also be thought of as a platform in itself. Java also has standard libraries for doing mathematics.

Java is a general purpose programming language with a number of features that make the language well suited for use on the World Wide Web. Small Java applications are called Java applets and can be downloaded from a web server and run on your computer by a Java-compatible Web browser, such as Netscape Navigator or Microsoft Internet Explorer:

The inventors of Java wanted a design a language, which could offer solution to some of the problems encountered in modern programming. They wanted the language to be not only reliable, portable and distributed.

Although the above appears to be a list of buzzwords, they aptly describe the full potential of the language. These features have made java the first application language of the World Wide Web. Java will also become the primer language for general-purpose stand-alone applications.

Properties of java:

1 Compiled and interpreted
2 Platform-independent and portable
3 Robust and secure
4 Multi Threading and interactive
5 Dynamic and extensible

Compiled and interpreted:

Usually a computer language is either compiled or interpreted. Java combines both these approaches thus making Java a two-stage system. First, Java compiler translates source code into what is known as byte code instructions.

Platform independent and portable:

The most significant contribution of Java over languages is its portability. Java programs can be easily moved from one computer system to another, anywhere anytime. Changes and upgrades in operating systems, processors and system resources will not force any changes in Java programs. This is the reason why java has become a popular language for programming on Internet, which inter connects different kinds of systems worldwide. We can download a Java applet from a remote computer on to our local system via Internet an extension of the user's basic system providing practically unlimited number of accessible applets and applications.

Robust and secure:

Security becomes an important issue for a language that is used for programming on Internet. Threat of virus and abuse of resources is everything. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointers in Java ensures that programs cannot gain access to memory locations without proper authorization.

Distributed:

Java is designed as a distributed language for creating applications on networks. It has the ability to share both data programs. Java applications can open and access remote objects on Internet as easily as they can in a local system. This enables multiple programmers at multiple remote locations to collaborate and work together on a single project.

Multithreaded:

Multithreaded means handling multiple tasks simultaneously. Java supports multithreading programs. This means that we need not wait for the

application to finish one task before beginning another. For example, we can listen to an audio clip time download an applet from a distance computer. This feature greatly improves the interactive performance of graphical applications.

Dynamic and Extensible:

Java is a dynamic language. Java is capable of dynamically linking in new class libraries methods and objects. Java can also determine the type of class through a query, making it possible to either dynamically link or abort the program, depending on the response.

Advantages of JAVA:

Java has gained enormous popularity since it first appeared. Its rapid ascension and wide acceptance can be traced to its design and programming features, particularly in its promise that we can write a program once, and run it anywhere. Java was chosen as the programming language for network computers (NC) and has been perceived as a universal front end for the enterprise database. As stated in Java language white paper by Sun Microsystems: Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neural, portable, multithreaded and dynamic.

2.1.3.2 ABOUT JCreator

JCreator is a powerful IDE for Java technologies that provides more power at your fingertips than all the ordinary IDEs combined. What makes JCreator a cut above the rest? Just look at the following features:

- Manage projects with ease in the interface that is much like Microsoft Visual Studio.
- Define your own color schemes for unlimited ways to organize your code.

- Unlike most IDEs, JCreator wraps around your existing projects and allows you to use different JDK profiles.
- Get down to writing code quickly with our project templates.
- Our class browser makes viewing your project a breeze.
- Debug with an easy, intuitive interface. No need for silly DOS prompts!
- Walk through our wizards and cut to the chase of writing your project, quickly and easily.
- You don't have to spend valuable time on Classpath configurationJCreator does it all for you.
- Customize our user interface the way that you like it.
- Set up your own run-time environments to run your application as an applet, in a JUnit environment or in a DOS window.
- JCreator has lower system requirements, yet faster speed, than all those other IDEs.

Blast from the Past

JCreator is backwards-compatible with all previous version of the JDKs published by Sun Microsystems. With one click, you can toggle between different JDK profiles. Each profile defines the JDK home path and any additional files for the classpath, such as the following:

- JDK 1.5.0
- JDK 1.5.0 - J2EE
- JDK 1.4.2

- JDK 1.2.2
- JDK 1.0.2

JCreator is faster and more efficient than Java-based IDEs. This powerful tool is written entirely in a native Windows language. And, on top of that, JCreator is professionally designed to meet Microsoft Windows interface guidelines: you can work quickly and efficiently with the intuitive Windows interface.

Features Of JCreator v3

	LE	Pro
The capabilities of the editor		
Selection margin with line numbers JCreator gives you the option of viewing line numbers in the selection margin.	x	x
Instant colour syntax highlighting With syntax highlighting, you can easily distinguish between keywords, methods, comments and plain text. All colors can be customized to your preferences. The syntax parser works with syntax configuration files which you can modify to your needs. The parse handles up to four keyword lists and up to four	x	x

embedded languages. A new feature is to expor the settings to xml based files and to print the document in colors.		
Unlimited Undo/Redo capability	x	x
Visible display of tabs and spaces	x	x
Code Folding You can collapse logical hierarchical sections o your code as you are working in other parts of t file. This allows quick navigation within the source and provides and overview of the class methods. A folding point can be any block of code: class, module, def, if, else, for, and so forth.		
A powerful search and replace engine for single and multiple documents The result of the search will be displayed in the output view	x	x
Auto-indent When you begin a new line, auto-indent positio	x	x

your cursor at the same initial position as the te: on the previous line.		
Word completion With the word-completion function you can complete half typed words. If they are already mentioned somewhere in the text.	x	x
External file modification detection JCreator will automatically ask to reload a file when it has been altered outside the editor.		
Customizable keys		
Bookmarking capabilities		
Format text capabilities Change cases, spaces to tabs, comments etc.		
Advanced edit commands Advanced edit commands for selecting and deleting text.		
Code Templates		

The Code Templates are abbreviations you may expand to strings while editing the code. They are expanded in the text by entering the abbreviations and pressing either Space,Enter o Tab. You can also trigger the code template popup-list using the shortcut defined in the edit menu.

Go to last change option
The cursor will move to the position were you made the last change.

2.1.3.3 INTRODUCTION TO SOCKET PROGRAMMING

A socket is a communication end point, or an object through which a Socket application sends or receives packets of data across a network. Two sockets types are available:

i. Stream sockets
   Stream sockets provide for a data flow without record boundaries a stream of bytes. Streams are guaranteed to be delivered and to be correctly sequenced and unduplicated.

ii. Datagram sockets
   Datagram sockets support a record-oriented data flow that is not guaranteed to be delivered and may not be sequenced as sent or unduplicated.

Sockets are highly useful in at least three communication contexts:

i. Client/Server models.

ii. Peer to peer scenarios such as chat applications.

iii. Making remote procedure calls by having the receiving application interpret a message as a function call.

2.2 CONGESTION IN INTERNET

2.2.1 CURRENT SITUATION

In our present network we have to cope a great deal with the increasing traffic which is unprecedented. We no longer will be able to control the traffic with our existing protocols efficiently. So irrelevant of the protocol used we would not be able to achieve a congestion free network without considering the following parameters.

- CPU Usage
- Bandwidth Utilization
- Application

2.2.2 REASONS FOR INTERNET TRAFFIC

These confirm to the above passage indicating the congestion

- CPU Usage is an important factor for network congestion because of the uninterrupted flow of data making it an overload for the processor there by in the process losing critical information
- 10 years then internet connections were characterized by long TCP connections enabling the network to react to congestion promptly. Because of the traffic we are encountering today we are unable to provide the network with the time that it needs to overcome a potential loss of data.

- In the current era of net users downloading information, which according to a study conducted by NASSCOM has increased twofold, Download accelerators are no new commodity to internet users. Download accelerators increase the probability of congestion by disabling the congestion control mechanisms in the protocol. So our algorithm is protocol independent.
- This feature discourages the use of inappropriate protocol for the purpose of controlling congestion

2.3 WHAT IS CONGESTION ?

Congestion is a state where we lose essential data due to the over utilization of bandwidth and CPU. This occurs because of the overflow of data that cannot be handled by the receiver judiciously.

2.4 PROPOSED SYSTEM

2.4.1 CREATING CONGESTION

In order to control congestion we have to create it and we do it by a method that has been tested and proven to its maximum capability. Because of its huge credentials we have altered it a little to cater to our requirements.

It is necessary to analyze the bandwidth and CPU utilization. We were able to come up with a solution for generating the type of data that would produce congestion by utilizing the entire bandwidth and CPU usage.

We are going to generate a matrix of data that is of huge proportions to cater to our requirements. By sending this type of data we achieve our target of creating the congestion. This will provide the matrix with full CPU

utilization which will enable congestion to tell itself. It can be controlled by controlling the CPU utilization .This task mentioned above is the objective of our project.

2.4.2 CONTROLLING CONGESTION

The aim of our project that we have undertaken is to provide our fellow internet users with minimum problems during their usage of this service. In doing so we have come up with a smart procedure to control the demon out of the congestion.

In order to do this we have to efficiently interact with the Operating system. This will pose a huge problem because it is no easy job to tell the OS to do certain things in certain ways for our purpose because Operating system holds our neck instead of the other way around.

This brings into light the scheduling algorithms that are of most importance. Scheduling the CPU requires some precise calculations in order to send various types of data in various ways according to the application that we are carrying out as our objective.

The time we apportion the processor for different tasks need to be Of varying degree because different task requires different time intervals. But it can be efficiently apportioned only when we know the appropriate time required for every task. So this requires prediction which achieved to the degree of the standard that is set by the Operating system designers will prove to be very useful to the task of CPU scheduling. The method which we use is one of its kinds where we reduce the burden of CPU by allocating some fixed time intervals.

We do this by stopping the file transfer for a very small time interval (milliseconds).This time interval is used for the purpose of processor operations. This can be varied according to the level of time interval that would increase the accuracy of our project.

SYSTEM STUDY

3. SYSTEM STUDY

3.1 SERVER MODULE

Server module deals with connection management and enumerates the number of clients that are currently connected to the server.
It also overlooks the transaction carried out between the clients. We should be able to configure the port no and the IP address for the clients that are to be connected. It will be able to terminate any or all of the clients that in its control.

3.2 SERVER INTERFACE MODULE

Server interacts with the clients through the server interface module. It consists of the functions necessary to design the GUI for the purpose of interaction with the user. It also consists of the coding for the events that should be carried out for every button clicks.

3.3 CLIENT MODULE

Client module deals with connection management with the server .We would also be able to configure the IP address and the port address .This enables the client to be connected with the server mentioned in the configuring list. As many as 1000 clients can be connected to the server.

3.4 CLIENT INTERFACE MODULE

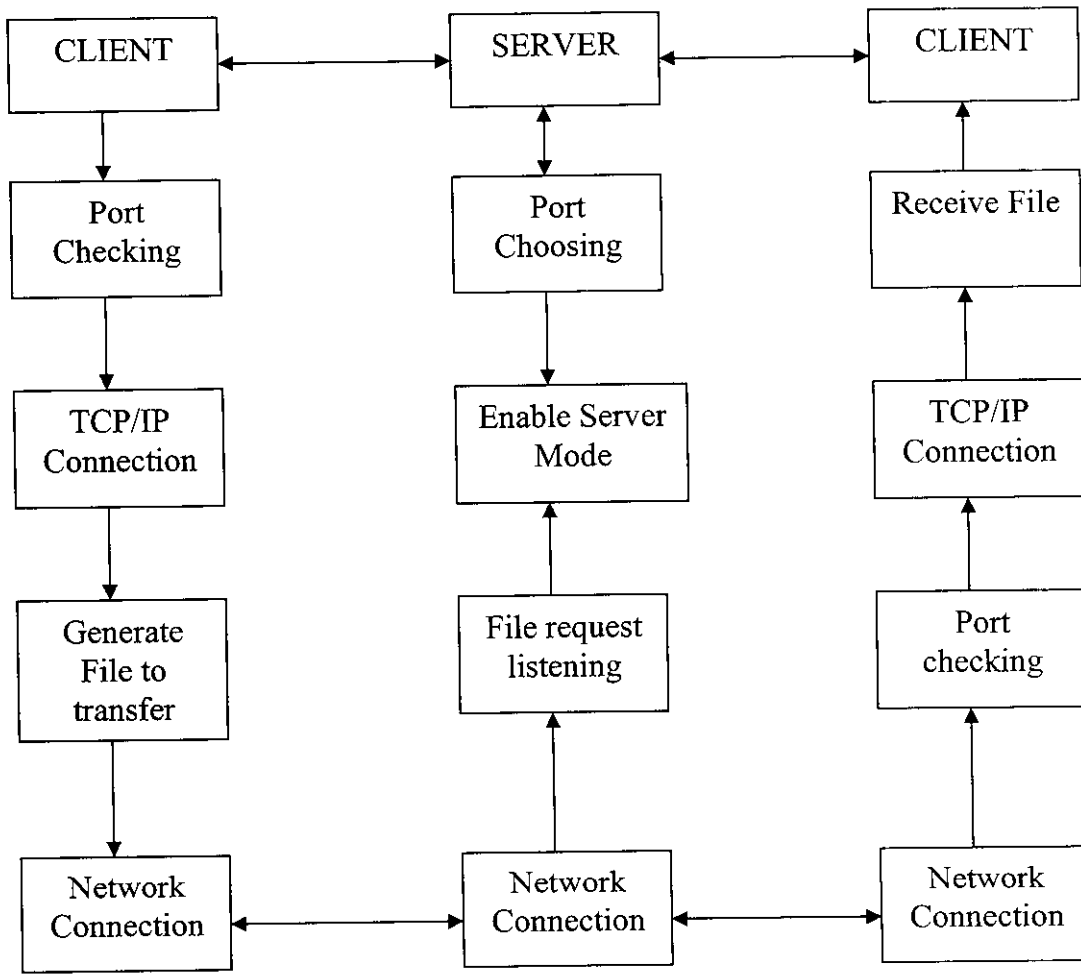
Client interacts with the server through the client interface module. It consists of the functions necessary to design the GUI for the purpose of interaction with the user. It also consists of the coding for the events that should be carried out for every button clicks. Through the client interface it is also possible

to know the other clients that are connected to the server.

3.5 CPU MODULE

CPU module is used to extract the processor performance information from the operating system for the purpose of providing the user with the performance of the CPU both during the normal transmission and the scheduled transmission.

3.6 FILE TRANSFER



SYSTEM TESTING

4. SYSTEM TESTING

4.1 TESTING OBJECTIVE

The testing objectives are summarized in the following three steps. Testing is the process of executing a program with the intent of finding an error. A good test is one that has high probability of finding an error. A successful test is one that uncovers as-yet-undiscovered errors.

4.2 TESTING PRINCIPLES

All tests should be traceable to customer requirements. Tests should be planned long before testing begins, that is, the test planning can begin as soon as the requirements model is complete. Testing should begin "in the small" and progress towards resting "in large". The focus of testing will shift progressively from programs to individual modules and finally to the entire project. Exhaustive testing is not possible. To be more effective, testing should be one, which has highest probability of finding errors.

The following are the attributes of good tests:

1 A good test has a high probability of finding an error.
2 A good test is not redundant.
3 A good test should be "best of breed".
4 A good test should be neither too simple nor too complex.

4.3 LEVELS OF TESTING

The details of the software functionality tests are given below. The testing procedure that has been used is as follows:

1 Unit testing
2 Integration testing
3 Validation testing
4 Output testing

4.3.1 UNIT TESTING

Unit testing is carried out to verify and uncover errors within the boundary of the smallest unit or a module. In our project,each module was found to be working satisfactory as per the expected output of the module.In the package development,each module is tested separately after it has been completed and checked with valid data.Unit testing exercise specific paths in the modules control structure to ensure complete coverage and maximum error detection.

The project is divided into four modules.These four modules are developed separately and verified whether they function properly.After the completion of each module sample text files are used and tested.

The modules are

1 Server Module
  a)Server Interface Module
2 Client Module
  b)Client Interface Module
3 CPU Module

4.3.2 INTEGRATION TESTING

Integration Testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of higher-order tests are conducted. The main objective in this testing process is to take unit-tested modules and build a program structure that has been dictated by design.

The following are the types of integrated testing

TOP DOWN INTEGRATION

This method is an incremental approach to the construction of the program structure. Modules are integrated by moving downward the control hierarchy, beginning with the main program module are incorporated to the structure in either a depth-first manner.

BOTTOM UP INTEGRATION

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for the modules subordinate to given level is always available and the need for stubs is eliminated.

The bottom up integration strategy may be implemented with the following steps:

i. The low level modules are combined into clusters that perform a specific software sub-function.
ii. A driver i.e. the control program for testing is return to coordinate test case input and output.
iii. The cluster is tested and drives are removed and clusters are combined moving upward in the program structure.

The four modules developed during unit testing were combined together and

they were executed and verified for linking of the files and the corresponding modules successfully.

4.3.3 VALIDATION TESTING

At the end of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and correction testing begins.

VALIDATION TEST CRITERIA

Software testing and validation is achieved through series of black box test that demonstrate conformity with the requirements is achieved, documentation is covered and other requirement is met. Here the three modules which are checked in the integration testing are assembled and is tested for any correction.

5. FUTURE ENHANCEMENT

This project can be enhanced in the future by implementing the very thing in the internet environment which is a very large network than the LAN. Although our project has eliminated the loss due to congestion, we still face the bottleneck of trading off between the delay and loss of data.

This can also restrict the protocol dependent use of applications. So this is protocol independent.

6. CONCLUSION

This system will be able to cater to the requirements of WAN, MAN, CAN and Internet. The system's performance has been very impressive and further enhancements when considered, shows promise of an effective Congestion Management that could be of immense use to the network users.

The techniques applied in the design of the programs provide a scope for expansion and implementations of changes, which may be required in future, all programs have been tested and have found to execute correctly. All programs have been documented and can be easily understood.

7. APPENDIX

7.1 Sample source code

SERVER

```
import java.io.*;
import java.net.*;
import java.util.*;
import javax.swing.*;
import messages.*;

public class Server
{
	int serverPort;
	int serverLimit;
	ServerSocket server;

	int onlineNodes;
	Vector handlers; //list of all client handling threads in the server
	ServerInterface window;

	//constructor
	Server()
	{
		getConfiguration();
		onlineNodes=0;

		try
		{
			//bind server on socket,show interface and listen for clients
			server=new ServerSocket(serverPort);
			handlers=new Vector();
			window=new ServerInterface(this);
			listenForClients(server);
		}
		catch(IOException io_e)
		{
			JOptionPane.showMessageDialog(null,"Cannot Start
	Server","ServerSocket Error",JOptionPane.ERROR_MESSAGE);
			System.exit(0);
```

```
        }
        finally
        {
                try
                {
                        if(server!=null)
                        {
                                server.close();
                        }
                }
                catch(IOException e)
                {}
        }
}

public class Handler implements Runnable
{
        protected ClientInfo clientInfo=new ClientInfo();

        protected Socket socket;
        protected ObjectInputStream in;
        protected ObjectOutputStream out;

        //thread on which to run this handler
        protected Thread listener;

        public Handler(int clientID,Socket socket)
        {
                clientInfo.clientID=clientID;
                this.socket=socket;
                //onlineNodes=0;
        }

        //set up parameters and required data with the client
        protected boolean handShake()
        {
                try
                {
                        clientInfo.clientName=(String)in.readObject();


        if(window.clientList.size()>0&&window.clientList.indexOf(clientIn
fo.clientName)!=-1)
```

```
                {
                        try
                        {
                                ConnectionNotice status=new
ConnectionNotice();
                                status.status=false;
                                out.writeObject(status);
                                out.flush();
                                socket.close();
                                return(false);
                        }
                        catch(IOException w)
                        {}
                }

                try
                {
                        ConnectionNotice status=new ConnectionNotice();
                        status.status=true;
                        out.writeObject(status);
                }
                catch(IOException w)
                {}

        //add client to list and send it the complete client     list
        window.clientList.addElement(clientInfo.clientName);
        out.writeObject(window.clientList);
        //tell all other clients about the new client
        UpdateList newClient=new UpdateList();
        newClient.requestType=true;
        newClient.newClient=clientInfo.clientName;
        broadcast(newClient);
        onlineNodes++;
        }
                catch(Exception e)
                {}

                return(true);
        }

        public synchronized void start()
        {
                if(listener==null)
```

```
        {
                try
                {
                //the order in whick these streams are created
                //is important; reverse the order in client
               out=newObjectOutputStream(socket.getOutputStream());
                in=new ObjectInputStream(socket.getInputStream());
                ConnectionNotice status=new ConnectionNotice();
                if(onlineNodes>=serverLimit)
                {
                        try
                        {
                                status.status=false;
                                out.writeObject(status);
                                out.flush();
                                socket.close();
                                return;
                        }
                        catch(IOException w)
                        {}
                }
                try
                {
                        status.status=true;
                        out.writeObject(status);
                }
                catch(IOException w)
                {}
                if(!handShake())
                {
                        return;
                }
        //create a new thread and start listening to the client
                listener=new Thread(this);
                listener.start();
                setContentsText();
        }
        catch(IOException ignored)
        {}
        }
}

public synchronized void stop()
```

```
{
        if(listener!=null)
        {
                try
                {
                        listener.interrupt();
                        listener=null;
                //remove handler from list; remove client from active list;
                //notify all other nodes that this client has left
                handlers.removeElement(this);
               window.clientList.removeElement(clientInfo.clientName);
                UpdateList newClient=new UpdateList();
                newClient.requestType=false;
                newClient.newClient=clientInfo.clientName;
                broadcast(newClient);
                //close the sockets and show message in server window
                out.close();
                socket.close();
                setContentsText();
                        }
                        catch(IOException io_ex)
                        {
                        JOptionPane.showMessageDialog(window,"Error
closing connection to client","Termination
Error",JOptionPane.ERROR_MESSAGE);
                        }
                }
        }

        public void run()
        {
                try
                {
                        handlers.addElement(this);
                //listen to input stream for messages from this client
                        while(true)
                        {
                                try
                                {
                                        Object clientMsg=in.readObject();
                                        if(clientMsg instanceof Message)
                                        {
```

```
                                        if(((Message)clientMsg).audience==true)
                                        {
                                                broadcast(clientMsg);
                                        }
                                        else //private message
                                        {
                                                //to Locate the intended reciever:
                                                //1)get the id of recipient(recieverID)
                                                //2)get the reference of handler
                                                //3)send message on its output stream

        ((Handler)(handlers.elementAt(((Message)clientMsg).recieverID))).
out.writeObject(clientMsg);
                                        }
                                }
                                else if(clientMsg instanceof Request)
                                {
                                        //tell the intended recipient that
this
                                        //client wants to start a private
message transfer
                                        //to Locate the intended reciever.
                                        //1)get the id of
recipient(recieverID)
                                        //2)get the reference of handler
                                        //3)send request on its output
stream

        ((Handler)(handlers.elementAt(((Request)clientMsg).recieverID))).o
ut.writeObject(clientMsg);
                                }
                                else if(clientMsg instanceof
LogOutNotice)
                                {
                                        onlineNodes--;
                                        break;
                                }
                                else
                                {
                                        //if an unknown object is recieved
                                        System.out.println(clientMsg);
                                }
```

```
                    }
                    catch(ClassNotFoundException cnfe)
                    {

        JOptionPane.showMessageDialog(window,"Class of a serialized
object cannot be found.","Termination
Error",JOptionPane.ERROR_MESSAGE);
                            break;
                    }
                    catch(InvalidClassException ice)
                    {

        JOptionPane.showMessageDialog(window,"Something is wrong
with a class used by serialization.","Termination
Error",JOptionPane.ERROR_MESSAGE);
                            break;
                    }
                    catch(StreamCorruptedException sce)
                    {

        JOptionPane.showMessageDialog(window,"Control information in
the stream is inconsistent.","Termination
Error",JOptionPane.ERROR_MESSAGE);
                            break;
                    }
                    catch(OptionalDataException ode)
                    {

        JOptionPane.showMessageDialog(window,"Primitive data was
found in the stream instead of objects.","Termination
Error",JOptionPane.ERROR_MESSAGE);
                            break;
                    }
                    catch(IOException ioe)
                    {

        //JOptionPane.showMessageDialog(null,"Any of the usual
Input/Output related exceptions.","Termination
Error",JOptionPane.ERROR_MESSAGE);
                            break;
                    }
                }
            }
```

```
                catch(Exception ignored)
                {}
                finally
                {
                        stop();
                }
        }

        //broadcasts a message to all clients
        protected void broadcast(Object clientMsg)
        {
                //object streams have to be protected
                synchronized(handlers)
                {
                        Enumeration enum=handlers.elements();

                        //loop through all the clients
                        while(enum.hasMoreElements())
                        {
                                Handler
handler=(Handler)enum.nextElement();
                                try
                                {
                                        handler.out.writeObject(clientMsg);
                                        out.flush();
                                }
                                catch(IOException ex)
                                {
                                        handler.stop();
                                }
                        }
                }
        }
}

//listen to connection requests from clients
protected void listenForClients(ServerSocket server)
{
        setContentsText();

        int clientID=0;

        while(true)
```

```
        {
                try
                {
                        //listen to socket. When request recieved start
                        //a handler and start listening again
                        Socket client=server.accept();
                        Handler handler=new Handler(++clientID,client);
                        handler.start();
                }
                catch(IOException io_ex)
                {
                        JOptionPane.showMessageDialog(window,"Cannot
Setup Connection","Client Connection
Error",JOptionPane.ERROR_MESSAGE);
                }
        }
}

protected void shutDown()
{
        Enumeration enum=handlers.elements();
        ServerShutDown shutdown=new ServerShutDown();

        synchronized(handlers)
        {
                //close connection to all clients.Stop all handlers
                while(enum.hasMoreElements())
                {
                        try
                        {
        ((Handler)(handlers.firstElement())).out.writeObject(shutdown;
                        }
                        catch(Exception e)
                        {}

                        ((Handler)(handlers.firstElement())).stop();
                }
        }
        System.exit(0);
}

//save new configuration options to file
protected  synchronized void setConfiguration(int newPort ,int newLimit)
```

```
{
	try
	{
		FileWriter configFile=new FileWriter("serverinfo.txt");
		configFile.write(newPort+";"+newLimit+";");
		configFile.close();
		serverLimit=newLimit;
	}
	catch(IOException io_e)
	{
		JOptionPane.showMessageDialog(window,"Cannot Save
Configuration File","File Error",JOptionPane.ERROR_MESSAGE);
	}
}

protected synchronized void getConfiguration()
{
	try
	{
		char[] buffer=new char[15];
		FileReader configFile=new FileReader("serverinfo.txt");
		configFile.read(buffer);
		String value=String.copyValueOf(buffer);
		String[] temp=value.split(";");
		serverPort=Integer.parseInt(temp[0]);
		serverLimit=Integer.parseInt(temp[1]);
		configFile.close();
	}
	catch(FileNotFoundException fnf_e)
	{
		JOptionPane.showMessageDialog(null,"Configuration File
Not Found,Using Defaults","Configuration File
Missing",JOptionPane.ERROR_MESSAGE);

		serverPort=7000;
		serverLimit=20;
	}
	catch(IOException io_e)
	{
		JOptionPane.showMessageDialog(null,"Error Reading
Configuration File,Using Defaults","Configuration
Error",JOptionPane.ERROR_MESSAGE);
```

```
                serverPort=7000;
                serverLimit=20;
            }
        }
        //methods
        void setContentsText()
        {
            String tStr="ClientCount:
        "+window.lstClients.getModel().getSize()+".\n"+"Server Listening...\n";
            window.txtContents.setText(tStr);
        }

        public static void main(String[] args)throws IOException
        {
            new Server();
        }
}
```

CLIENT

```
import java.io.*;
import java.net.*;
import javax.swing.*;
import messages.*;

public class Client
{
        String Name;
        int clientID;
        private Socket socket;
        int serverSocketNumber;
        String serverAddress;
        boolean connected;
        ObjectOutputStream out;
        ObjectInputStream in;

        DefaultListModel clientList;
        InputListener listener;
        ClientInterface window;

        //constructor
        Client()
        {
```

```
        connected=false;
        listener=new InputListener();
        window=new ClientInterface(this);
}

class InputListener extends Thread
{
        //provides a way to pause the listener
        boolean running=false;

        public void run()
        {
                //loops forever but if listener is paused then skips code
                while(true)
                {
                        if(running)
                        {
                                //detect the type of message and take
 appropriate action
                                try
                                {
                                        Object serverMsg=in.readObject();

                                        if(serverMsg instanceof Message)
                                        {

        window.showMessage((Message)serverMsg);
                                        }
                                        else if(serverMsg instanceof Request)
                                        {

        window.openNewTab(((Request)serverMsg).senderID);
                                        }
                                        else if(serverMsg instanceof UpdateList)
                                        {
                                                //if information is about new node
 logging in

        if(((UpdateList)serverMsg).requestType==true)
                                                {

        clientList.addElement(((UpdateList)serverMsg).newClient);
```

```
                                        //mainsystem: new client
added
                                        //
                                    }
                                    //if information is about node
logging out
                                    else
                                    {

        window.notifyNodeLeft(((UpdateList)serverMsg).newClient);

        clientList.removeElement(((UpdateList)serverMsg).newClient);
                                        }
                                }
                                else if(serverMsg instanceof
 ServerShutDown)
                                {
                                        disconnectFromServer(false);
                                        window.notifyDisconnect();

        JOptionPane.showMessageDialog(window,"Server Has Been Shut
 Down","Connection Error",JOptionPane.ERROR_MESSAGE);
                                }
                                else if(serverMsg instanceof
 KickedOutNotice)
                                {
                                        disconnectFromServer(false);
                                        window.notifyDisconnect();

        JOptionPane.showMessageDialog(window,"Server Kicked You
 Out","Connection Error",JOptionPane.ERROR_MESSAGE);
                                }
                        }
                        catch(ClassNotFoundException cnfe)
                        {

        JOptionPane.showMessageDialog(window, "Class of a serialized
 object cannot be found.", "Termination Error",
 JOptionPane.ERROR_MESSAGE);
                                shutDown();
                        }
                        catch(InvalidClassException ice)
                        {
```

```
        JOptionPane.showMessageDialog(window, "Something is wrong
with a class used by serialization.", "Termination Error",
JOptionPane.ERROR_MESSAGE);
                                shutDown();
                        }
                        catch(StreamCorruptedException sce)
                        {

        JOptionPane.showMessageDialog(window, "Control information in
the stream is inconsistent.", "Termination Error",
JOptionPane.ERROR_MESSAGE);
                                shutDown();
                        }
                        catch(OptionalDataException ode)
                        {

        JOptionPane.showMessageDialog(window, "Primitive data was
found in the stream instead of objects.", "Termination Error",
JOptionPane.ERROR_MESSAGE);
                                shutDown();
                        }
                        catch(IOException ioe)
                        {
                                //JOptionPane.showMessageDialog(null,
"Any of the usual Input/Output related exceptions.", "Termination Error",
JOptionPane.ERROR_MESSAGE);
                                //shutDown();
                        }
                }
        }
    }
}

//transmit the message from node to server for the main room
void sendPublicMessage(String nodeMsg) throws IOException
{
        Message msg=new Message();
        msg.audience=true;
        msg.roomNumber=0;
        msg.senderID=clientID;
        msg.message=nodeMsg;
```

```
        out.writeObject(msg);
        out.flush();
}

//transmit the message from node to server for one intended client
void sendPrivateMessage(int recipient,String nodeMsg) throws
IOException
{
        Message msg=new Message();
        msg.audience=false;
        msg.recieverID=recipient;
        msg.senderID=clientID;
        msg.message=nodeMsg;

        out.writeObject(msg);
        out.flush();
}

//transmit a request that this node wants to communicate privately
//with someone on the list
void sendRequest(int recieverID)
{
        Request request=new Request();
        request.recieverID=recieverID;
        request.senderID=clientID;

        try
        {
                out.writeObject(request);
        }
        catch(IOException io_e)
        {}
}

//save new configuration values to file
void setConfiguration(String newServer,int newPort)
{
        try
        {
                FileWriter configFile=new FileWriter("clientinfo.txt");
                configFile.write(newServer.trim()+";"+newPort+";");
                configFile.close();
        }
```

```
        catch(IOException io_e)
        {
                JOptionPane.showMessageDialog(window,"Cannot Save
Configuration File","File Error",JOptionPane.ERROR_MESSAGE);
        }
}

//get configuration options from file and store in variables
void getConfiguration()
{
        try
        {
                char[] buffer=new char[255];

                FileReader configFile=new FileReader("clientinfo.txt");

                configFile.read(buffer);
                serverAddress=String.copyValueOf(buffer);
                String[] temp=serverAddress.split(";");

                serverAddress=temp[0];
                serverSocketNumber=Integer.parseInt(temp[1]);


        }
        catch(FileNotFoundException fnf_e)
        {
                JOptionPane.showMessageDialog(window,"Configuration
 File Not Found, Using Defaults","Configuration File
 Missing",JOptionPane.ERROR_MESSAGE);

                serverSocketNumber=1665;
                serverAddress="localhost";
        }
        catch(IOException io_e)
        {
                JOptionPane.showMessageDialog(window,"Error Reading
 Configuration File, Using Defaults","Configuration
 Error",JOptionPane.ERROR_MESSAGE);

                serverSocketNumber=1665;
                serverAddress="localhost";
        }
}
```

```
//take steps to disconnect from server
//reason=true means node choice; reason=false means by server
synchronized void disconnectFromServer(boolean reason)
{
        try
        {
                if(connected)
                {
                        //pause the listener thread
                        listener.running=false;

                        if(reason=true)
                        {
                                out.writeObject(new LogOutNotice());
                                out.flush();
                        }

                        out.close();
                        socket.close();

                        clientList.clear();
                        connected=false;
                }
        }
        catch(IOException ex)
        {}
}

//take steps to conect to the server
synchronized boolean connectToServer()
{
        getConfiguration();

        try
        {
                InetAddress addr=InetAddress.getByName(serverAddress);
                socket=new Socket(addr,serverSocketNumber);
        }
        catch(UnknownHostException e)
        {
                JOptionPane.showMessageDialog(window,"Host Not Found,
 Reconfigure.","Host Lookup Error",JOptionPane.ERROR_MESSAGE);
```

```
                return(false);
        }
        catch(IOException e)
        {
                JOptionPane.showMessageDialog(window,"Server Not
Found, Check If Server Exists.","Socket
Error",JOptionPane.ERROR_MESSAGE);
                return(false);
        }

        try
        {
                in=new ObjectInputStream(socket.getInputStream());
                out=new ObjectOutputStream(socket.getOutputStream());
        }
        catch(IOException e)
        {
                JOptionPane.showMessageDialog(window,"Cannot Create
Data Stream, Closing Client.","Data Stream
Error",JOptionPane.ERROR_MESSAGE);
                try
                {
                        socket.close();
                }
                catch(IOException io_e)
                {}

                return(false);
        }

        if(!handShake())
        {
                return(false);
        }

        listener.running=true;

        //if connecting for first time start the listener object
        if(listener.isAlive()==false)
        {
                listener.start();
        }
```

```
        connected=true;
        return(true);
}

//set up parameters with the server
private boolean handShake()
{
        try
        {
                if(((ConnectionNotice)in.readObject()).status)
                {
                        out.writeObject(Name);
                        if(((ConnectionNotice)in.readObject()).status==false)
                        {

        JOptionPane.showMessageDialog(window,"Name Already In Use.
 Change Client Name","Nick Error",JOptionPane.ERROR_MESSAGE);
                                return(false);
                        }

                        clientList=(DefaultListModel)in.readObject();
                        clientID=clientList.getSize()-1;
                        return(true);
                }
                else
                {
                        JOptionPane.showMessageDialog(window,"Maximum
 Node Limit Reached. Server Rejected Connection","Connection
 Rejected",JOptionPane.ERROR_MESSAGE);
                }
        }
        catch(Exception e)
        {}

        return(false);
 }

 void shutDown()
 {
        disconnectFromServer(true);
        listener=null;
        System.exit(0);
 }
```

```
        public static void main(String args[])  throws IOException
        {
                new Client();
        }
}
```

CPU-USAGE

```
//CPU.java

package cpuusage;
import java.lang.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public final class CPU extends JFrame implements ActionListener
{
        //cpu (jni)
        public static final CPU INSTANCE=new CPU();
        static { System.loadLibrary("CPUUSAGE"); }
        public native int getCpuUsage();

        //ui
        JFrame frmMain=new JFrame("CPU-Usage");
        JButton btStart=new JButton("Start");
        int top=74,left=18,height=100,width=280,x=0,prevy=0;

        //constructor
        public CPU()
        {
                frmMain.setBounds(100,100,320,192);
                frmMain.setResizable(false);
                frmMain.getContentPane().setLayout(null);

                btStart.setBounds(200,10,100,20);
                btStart.addActionListener(this);
                frmMain.getContentPane().add(btStart);

                frmMain.setVisible(true);
```

```
}

public void actionPerformed(ActionEvent evt)
{
        if(evt.getSource()==btStart)
        {
                monitor_CPU_Usage();
        }
}

public void monitor_CPU_Usage()
{
        try
        {
                while(true)
                {
                        Graphics g=frmMain.getGraphics();
                        g.setColor(Color.BLACK);
                        Graphics2D g2d=(Graphics2D)g;
                        g2d.setStroke(new BasicStroke(1));

                        Thread.sleep(1000);
                        int tUsage=getUsage();
                        frmMain.setTitle("CPU-Usage:
 "+Integer.toString(tUsage));

                        //draw graph
                        int graphunits=100;
                        int
 unitheight=(int)((double)height/(double)graphunits);
                        int y=tUsage*unitheight;
                        g2d.drawRect(left-2,top-2,width+4,height+4); //border
                        g2d.drawLine(left+x,height-
 prevy+top,left+x+2,height-y+top);
                        x+=3;
                        prevy=y;

                        //keep graph within the border
                        if(x>width)
                        {
                                x=0;
                                g2d.clearRect(left,top,left+width,top+height);
                        }
```

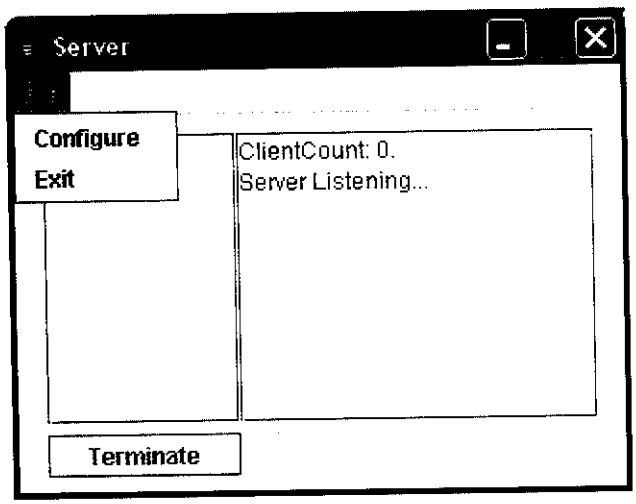
```
                }
            }
            catch(Exception err){}
        }

        //methods
        static int getUsage()
        {
            return(CPU.INSTANCE.getCpuUsage());
        }

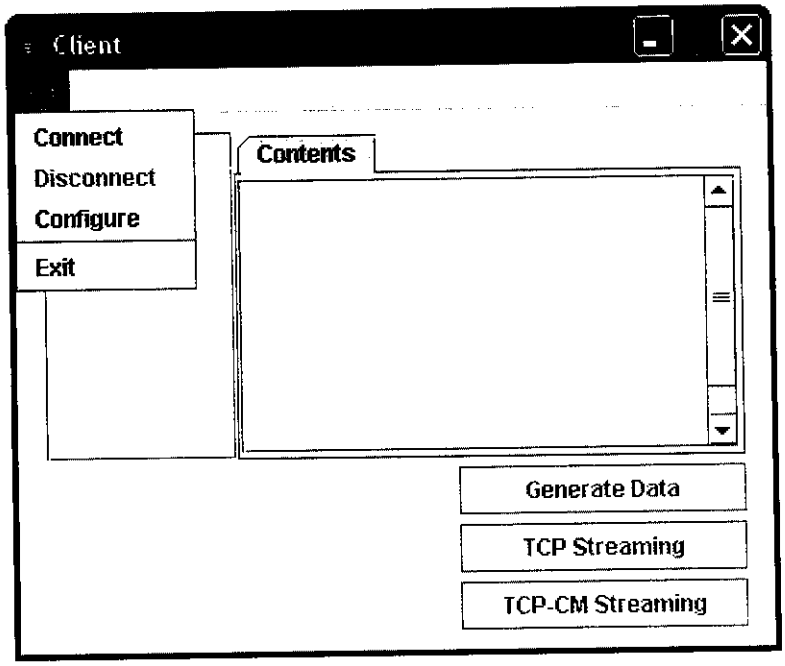
        public static void main(String args[])
        {
            //
        }
}
```

7.2 Sample output

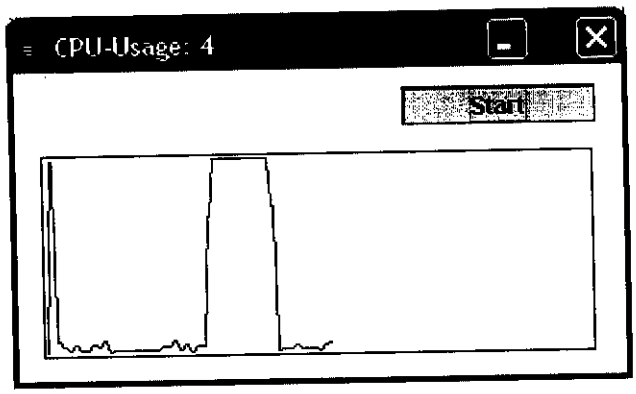
Server Listening



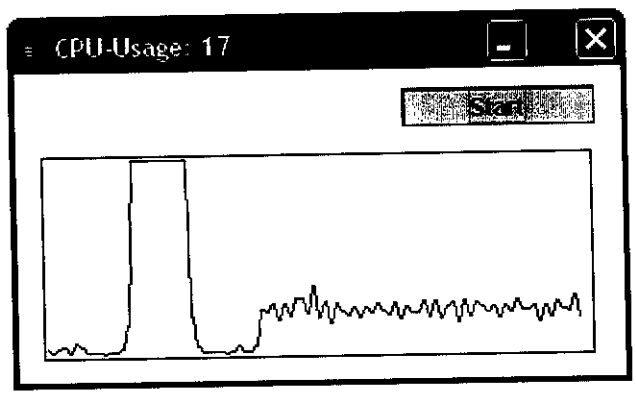
Client Connecting



CPU Usage- TCP

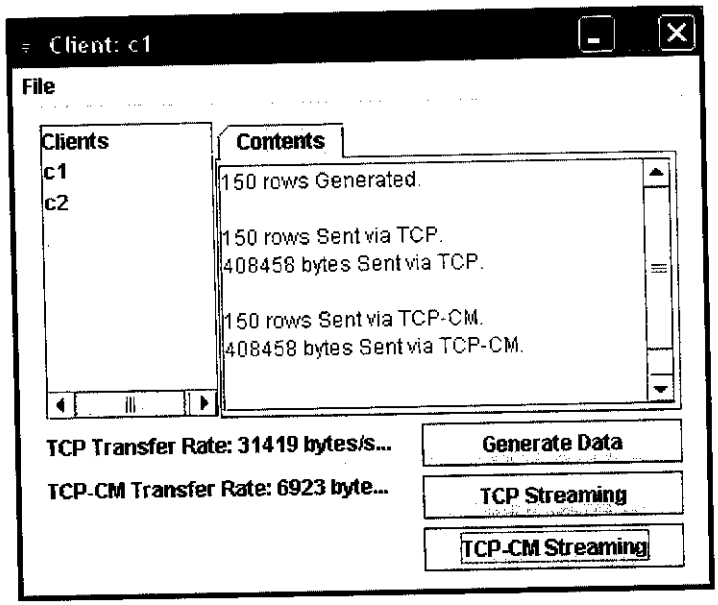


CPU Usage- TCP-CM

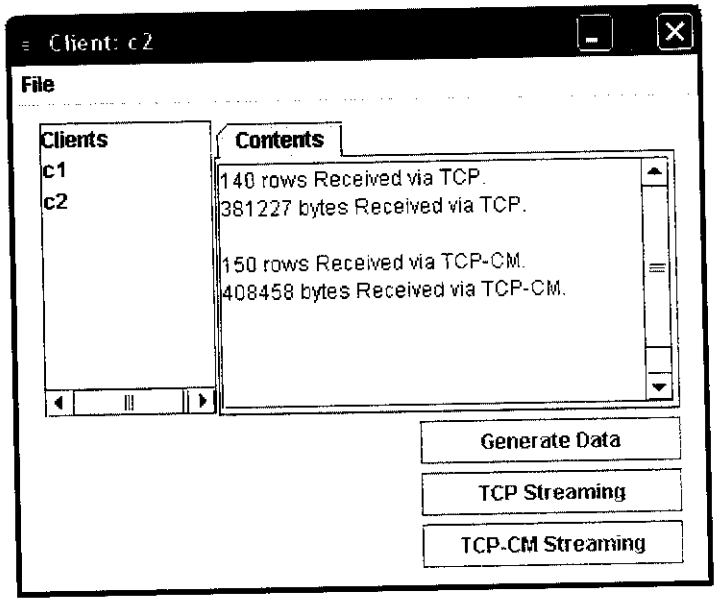


LIBRARY

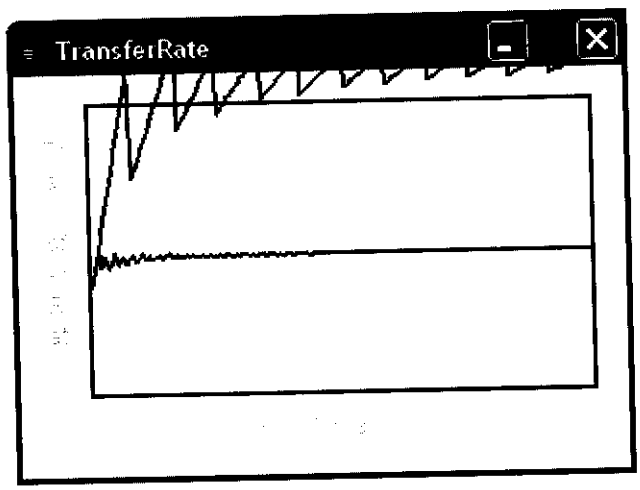
File Sent



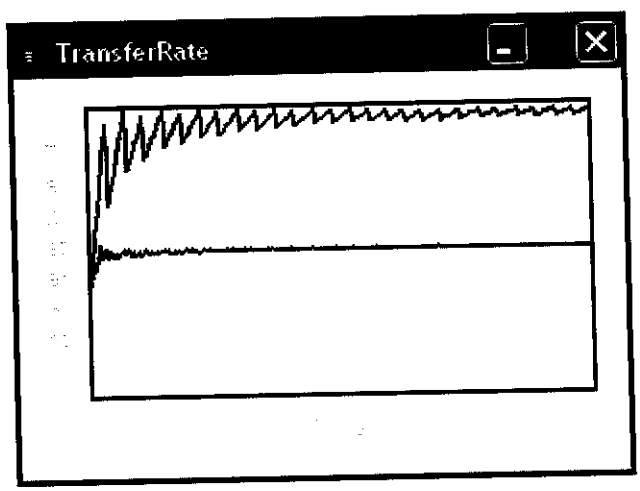
File Received



Performance Graph



For 150 rows



For 200 rows

8. REFERENCES

[1]. JAIN, R. "The Art of Computer Systems Performance Analysis", John Wiley and Sons, 1991.

[2]. PATRICK NAUGHTON, HERBERT SCHILDT ,"The Complete Reference" - Java2 3rd edition.

[3]. ROGER.S.PRESSMAN(2001), "Software Engineering a Practitioner's Approach", McGraw Hill.

[4]. ANDREW.S.TANENBAUM, "Computer networks", Prentice Hall, Fourth Edition.

[5]. Tutorial on Java Collections Framework
Website:http://java.sun.com/developer/onlineTraining/collections/Collection.html