# LOAD SHARING

## PROJECT REPORT P-1205

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE AWARD OF THE DEGREE

**BACHELOR OF COMPUTER SCIENCE AND ENGINEERING**
**OF**
**BHARATHIAR UNIVERSITY, COIMBATORE.**


Submitted by,

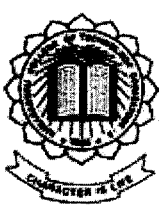
**A.LAVANYA (0027KO177)**

**S.SUGANYA (0027KO204)**

Guided by,

**Ms. S. RAJINI B.E.,**

Senior Lecturer, Department of Computer Science and Engineering.



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**KUMARAGURU COLLEGE OF TECHNOLOGY**
**(Affiliated to Bharathiar University, Coimbatore)**
**COIMBATORE – 641006.**


**MARCH 2004.**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**KUMARAGURU COLLEGE OF TECHNOLOGY**
**(Affiliated to Bharathiar University, Coimbatore)**

**CERTIFICATE**

This is to certify that the project entitled

# LOAD SHARING

is done by

**A.LAVANYA (0027KO177)**

**S.SUGANYA (0027KO204)**

And submitted in partial fulfillment of the requirement
For the award of the degree

**BACHELOR OF COMPUTER SCIENCE AND ENGINEERING**

**Of**

**BHARATHIAR UNIVERSITY, COIMBATORE.**

S. Thangasamy

**Professor and Head of the Department**
**(Dr. S. THANGASAMY Ph.D.,)**

Rajini

**Guide**
**(Ms. S. RAJINI B.E.,)**

Certified that the candidate with University Register No. 0027KO177, 0027KO204 was examined by us in the project work Vivavoce held on 24-3-04

24/3/'04

**Internal Examiner**

24/3/04

**External Examiner**

# DECLARATION

We,

**A.LAVANYA (0027KO177)**

**S.SUGANYA (0027KO204)**

declare that we do the project entitled "LOAD SHARING" and to the best of our knowledge, a similar work has not been submitted to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This report is submitted in partial fulfillment of the requirement for the award of the degree of Bachelor of Computer Science and Engineering of Bharathiar University.

**A.LAVANYA**

**S.SUGANYA**

Countersigned:

GUIDE: Ms. S. RAJINI B.E.,
Senior Lecturer, Department of Computer Science and Engineering,
Kumaraguru College of Technology,
Coimbatore – 641006.

Place: coimbatore - 6

Date: 17-3-04

# *Acknowledgement*

# ACKNOWLEDGEMENT

We would like to begin with a special note of gratitude to our Principal **Dr.K.K.Padmanaban, B.Sc. (Engg), M.Tech, Ph.D.,** and our Head of Department **Prof.Dr.S.Thangaswamy, Ph.D.** for their encouragement and making available to us the much needed facilities for successfully completing this project.

We also take immense pleasure in thanking our project coordinator **Mrs.D.Chandrakala, M.E, Senior Lecturer,** without whose encouragement our endeavor would not have been successful.

We are eternally indebted to our project guide **Ms.S.Rajini B.E, Senior Lecturer,** who has been a constant source of inspiration and guidance. Her gentle yet firm goading helped us finish our project on time

Our sincere thanks to our parents, all the staff members and our dear friends who all played small but important parts in helping us complete this project.

Above all, we owe our gratitude to the Almighty, for showering abundant blessings on us.

# *Synopsis*

# SYNOPSIS

The phenomena of Load sharing aims at providing cost-effective and more efficient substitute to the powerful and more expensive stand-alone computers, and are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of a normal stand-alone system. Our project Load Sharing on Linux aims at splitting a time consuming problems Into sub problems and distribute them over the idle systems.

We have, in our endeavor, distributed the load over the system, which has sufficient processing power, thereby utilizing the unused processing power and reducing computational time. The application we have taken is the processing of voluminous file. The time difference when the job is done on a stand-alone machine and that when done on the network is notable.

# *Contents*

# CONTENTS

# *Introduction*

## 1.1 Existing system and Limitations:

- Time consumption:

Whenever any time consuming operation is done on a stand-alone machine, there is a risk of a large delay as the whole operation is carried out by the single machine's processor, which can get overloaded.

- Inadequate performance:

Lack of resources such as memory, required to run the bulky job in the stand-alone machine leads to inadequate performance. So the stand-alone computer has to be a powerful one making it expensive.

## 1.2 Proposed system and Advantages:

Here the processing potentials of a group, say two to three machines on the LAN are made use of thereby

- Reducing total time consumption:

Since the job is distributed across a number of machines, the total time taken to complete the process is greatly reduced.

- Utilizing idle process time:

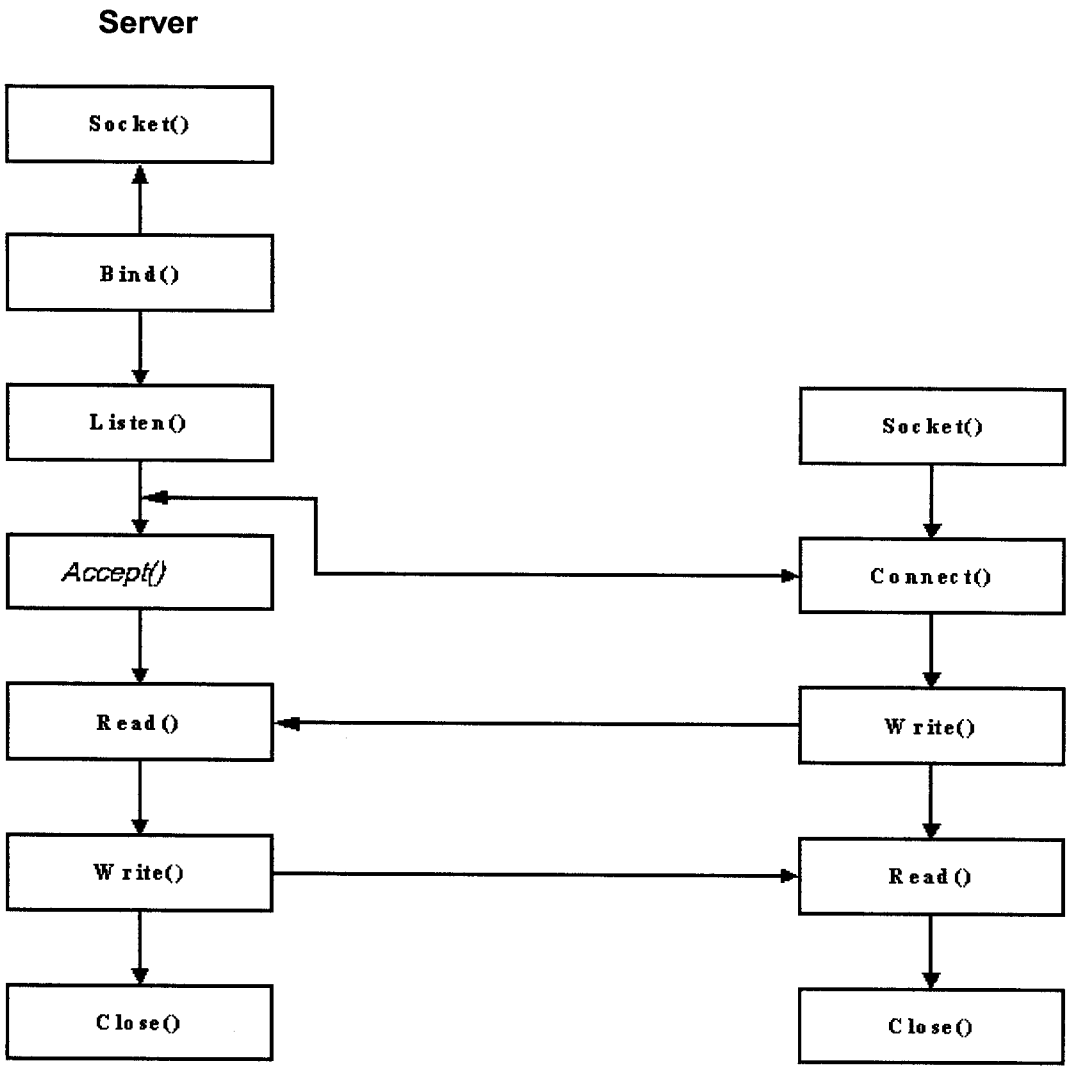
Instead of overloading a single machine and wasting the processor potential of neighboring computers, the idle time of their processors is made use of thereby distributing and balancing the load.

## SOCKETS

The socket is a logical connection between two computers that helps to transfer data from one computer to another. In order to transfer the data we require an imaginary entry and exit point called the port. This port can be used for any other transaction after the present transaction is complete.

The main types of sockets are TCP sockets and UDP sockets. Among them we have used the connection oriented TCP sockets. This is because connection oriented TCP sockets are more reliable than UDP sockets, which do not guarantee the safe transfer of data. UDP is faster than TCP, but without error and flow control.

## CONNECTION ORIENTED TCP SOCKETS




Flow of Socket Programming

# *Software Requirements Analysis*

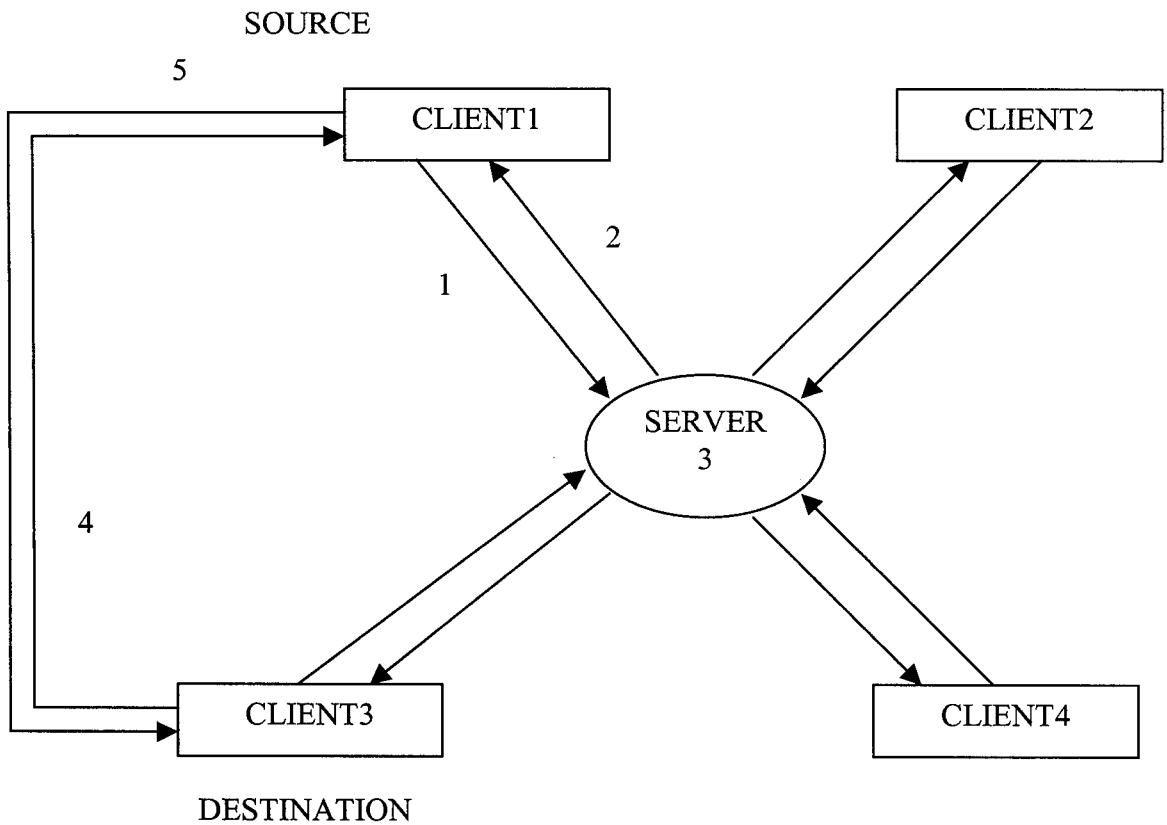
# SOFTWARE REQUIREMENTS ANALYSIS

## Project Definition

The load sharing algorithm is used to distribute the load equally among the systems connected in the network, thereby reducing the total time taken for completing the jobs and also utilizing the unused processing power of the systems on the network.

Though the concept of load balancing and distribution are effective on their own, they have been mostly done dependent on each other. This tool we have developed aims to combine the advantages of both, by distribution based on the load of the systems on the network.

Dataflow diagram:




1. Request to server
2. Intimation to client
3. Check the CPU usage
4. File transfer
5. Merging of output

## Module Description

1. Designing and Implementation of Centralized Management
2. Designing and Implementation of Distributed Management
3. Designing and Implementation of Distributed Management using cache

## Designing and Implementation of Centralized Management

A centralized management is achieved by designing a Job Management System and has three parts: design of mirror server that handles the request from the clients, design of Job Management System that handles and monitors the data transfer between the client and server and the client design to do the requested job by the user.

## Designing and Implementation of Distributed Management

Two designing are done, one is the server design and the other is the client design. The server is designed so that the client can request a free server and that does the job of splitting the load and responds to the client. The client does the job requested by the user.

## Designing and Implementation of Distributed Management using cache

This is advancement to the distributed management by having cache and lookup table to monitor the happening between its clients.

## Inputs to the System:

The inputs are passed as command line arguments. They are

- The name of the file to be processed
- The IP address of the server
- The port number

## Outputs of the System:

The outputs obtained from the system are as follows:-

- The size of the file sent to the server
- The total size of the file
- The time required for processing by each server

## Project plan:

In the analysis phase, requirements for load sharing namely Client-Server mode of interactions, interprocess communication and concepts of Linux were studied.

Next the system design was established where the complete system was depicted in the form of event flow diagram.

In the Implementation phase, all the necessary coding were written to split the overloaded application, process each file and merge the processed contents.

In the testing phase, all the necessary tests were performed to test the validity of the code developed.

# *Software Requirement Specification*

# SOFTWARE REQUIREMENTS SPECIFICATION

## Introduction

### Purpose:

The intended goal of this SRS is to illustrate the requirements of the Load sharing tool on Linux. The Load sharing tool distributes a given job over a number of connected nodes so as to reduce processing time as well as to utilize the unused processing power of the computers on the network, so that all nodes are equally used.

### Scope:

The SRS concentrates primarily on distributing the load given to a particular system.

### Overview:

Exploring further the Software Requirements Specification gives generic information about the various services employed by the tool. It also gives a detailed flow of how we intend to go about designing and implementing Load sharing in Linux based network.

## Overall Description:

## Elements of the system:

The system consists of two main entities-Client and Server.

Client-The client is the one to which the application is given and this application is split and distributed among the three systems.

Server-This processes the load given to it

## Supportability:

Throughout the coding,'C' coding conventions are followed.

## Communication interfaces:

The nodes are connected and they are made to communicate to each other by means of Socket programming.

## Introduction to C:

The language selected for this project is "C" efficient for network oriented. This language is efficient, powerful and compact.

C is a general purpose programming language which was originally designed for and implemented on the UNIX operating system on the DEC PDP – 11,by Dennis Ritchie. C is not tied to any particular hardware or system, however, and it is easy to write programs that will run without change on any machine that supports C.

C has emerged as the language of choice for the most applicable due to the speed, portability and compactness code. C is highly portable; programs running on the computer can be used with different operating system with slight or no modification.

C has got powerful operators and it has got several standard functions for developing programs. Another important feature of C is its ability to extend itself.

A C program is basically a collection library. With the availability of large number of functions, the programming task becomes simple.

## Introduction to Linux:

There are many reasons behind choosing this and all are equally important. First one is the availability of source code, as the operating system comes with source code. It is written in high level language C, which is very powerful, so we could easily understand the codes and even write new codes to suit our needs. Developers develop it, so plenty of documentation is available on the subject.

Linux operating system is free. No price has to be paid to get a copy of detain LINUX distribution. This is used developers.

Another important aspect is the power. Even with old 386 or 486 with 8mb of RAM, we can run a LINUX, which is highly impossible with other kind of operating system. So we could select the necessary file copy into a floppy and boot the system.

The system will be up and running.

It is and operating system for the network, which are multi-user, multitasking and failsafe. There is a very less chance that the system may crash. If at all mistake, happen with a particular program, we can kill that particular program and the system keeps on running.

LINUX is portable, which will support almost all the existing architecture whether it is INTEL,MACHINTOSH or SPARC,RISC OR THE ALPHA, Linux runs fine.

# Design Documents

# DESIGN DOCUMENTS

## Introduction:

The software architecture document lists the various issues concerning the architectural design phase of the software life cycle process. It provides a detailed explanation of the purpose, scope, definitions and abbreviations, concepts and references encountered in the design phase. The beneficiaries of this document are intended to be:

- The design team, which uses this document as a guide for the design process.
- The implementation team which carries out all implementation tasks and development of the user interface based on the guidelines provided in this document.
- The testing team which tests the developed software system and checks if it conforms to the original specifications and design motives.

## Purpose:

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of this system. It is intended to capture and convey the significant architectural decisions, which have been made on the system and

which are to be implemented. The reader of this document is equipped with a number of object and sequence diagrams which depict the control flow in the designed system and the various state transitions, which the system passes through.

## Scope:

The contents of this document cover the entire spectrum of design related issues and they influence the implementation process, the data structures used in the source code and the methods and functions present in the source code. This manuscript serves as a formal guideline for the design process and methodology.

## Overview:

The remaining portion of this document contains details about the architectural representation and also detailed explanations about the various views of the designed system. References to the quality measures adopted are also made available.

# *Product testing*

# PRODUCT TESTING

## Unit testing:

Unit testing focuses verification effort on the smallest unit of software product developed-software component or module. The different modules that were developed in the distribution tool were centralized management system, distributed management system, distributed management system using cache were developed to achieve the efficiency of load sharing.

In unit testing, the following were examined to ensure the reliability and correctness of each module

- Client server connection
- Input and output files

Some of the common errors were detected and corrected namely

- File permissions
- Handling multiple clients

## Integration Testing:

Integration testing is a systematic approach for constructing the program structure while at the same time conducting tests to uncover errors associated with communication interfacing.

In the Distribution algorithm, testing was done by top-down integration where each unit is integrated with its previously created unit and tested for its validity.

# *Future enhancements*

# FUTURE ENHANCEMENTS

The implementation has been done only for the static distributions. It can be enhanced to dynamic distribution. Also this tool can be used for the distribution of either one general application or several different applications.

# *Conclusion*

# CONCLUSION

The project entitled "load sharing in Linux" is developed using Linux. C language is used for coding in the system. The software is developed so as to fulfill the problem of heavily work-loaded machine.

The list of beneficiaries of this modern technology is almost endless. Thus computer plays a vital role in every human's life and become part and parcel of everyone's life. Computer has driven with blazing speed, radical upheaval everywhere. These machines have literally appended traditional practices. The efficiency of this system understanding conditions point to its attractiveness. The communication speed is the only thing that connects for higher efficiency. As the technology is fast changing we need to perform the operation in a fast and efficient manner. Further this project is developed for the Linux environment that can be extended to many platforms.

# References

# REFERENCES

## Bibliography:


- Yung-Terng Wang and Robert J. T. Morris," *Load sharing in distributed systems",* IEEE Transactions on Computers, C-34 (3): 204--217, March 1985.
- Thomas schenk et al, "Red Hat Linux System Administration", BPB Publications, First Edition, 2000.
- Herbert Schildt, "C++ - The Complete Reference", Tata McGraw Hill Publishing Company Limited, Third Edition , 1999.
- Beej, "Beej's Guide to Network Programming", Brian "Beej" Hall, 2001


## Websites:


- http://linux.com.hk/man/showman.cgi?manpath=/man/man2/send.2.inc
- http://linux.com.hk/man/showman.cgi?manpath=/man/man2/recv.2.inc

# *Appendix*

# APPENDIX

Sample source code:

```
// client program

#include <stdio.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <sys/times.h>
#include <string.h>

typedef struct
{
char s[3][20];
char p[3][20];
}ip;


main (argc, argv)
int argc;
char *argv[];
{
int             sock[3],numbytes[3],counter[3],addr_len,temp,val,cnt,          i,
j,dn_time[3],con_time[3],t1[3],f_no,new_sock;
float tr_time;
char name[3][9],f_name[25],st_cat[10],d;
FILE *file1,*file2,*file3;
struct sockaddr_in myname;
struct sockaddr_in *nptr;
struct tms before, after;
ip serverip;
char buf[1000];
struct hostent *hp, *gethostbyaddr();

if ( argc <3 )
{
```

```
printf("network client failure: required parameters");
printf(" missing from the command line\n");
printf("network client: usage");
printf("[executable-name] [host name] [port number]\n");
exit(1);
}

if (( new_sock = socket(AF_INET, SOCK_STREAM, 0)) < 0 )
{
printf("network client socket failure %d\n", errno);
perror("network client");
exit(2);
}

myname.sin_port = htons( atoi(argv[2]) );
myname.sin_family = AF_INET;
printf("\n\nnetwork client connects to JMS %s ", argv[1]);
printf("using port number %d\n",ntohs(myname.sin_port));

hp = gethostbyname ( argv[1] );
if ( hp == NULL ) {
printf("network client gethostbyname failure %d\n",errno);
perror("network client");
close (new_sock);
exit(3);
}

else
 {
printf("JMS information obtained:\n");
printf("\tThe Host IPAddress is %s\n\n", hp -> h_name);
}

bcopy ( hp -> h_addr_list[0], &myname.sin_addr.s_addr,hp -> h_length );
memcpy ( &myname.sin_addr.s_addr,hp ->h_addr_list[0],hp -> h_length );

if ((connect(new_sock,(struct sockaddr *)&myname, sizeof(myname)))< 0 )
{
printf("network client %s connect failure %d\n",argv[0], errno);
perror("network client");
close (sock);
exit(4);
}

strcpy(buf,"send");
write (new_sock, buf, sizeof(buf) );
```

```
addr_len = sizeof(struct sockaddr);
if ( ( numbytes[0] = read (new_sock,buf,sizeof(buf)) ) < 0 )
{
printf("network client socket read failure %d\n", errno);
perror("network client");
close(new_sock);
exit(5);
}
else
{
file1=fopen("tmpcli","w");
fwrite(&buf,sizeof(buf),1,file1);
fclose(file1);
file1=fopen("tmpcli","r");
fread(&serverip,sizeof(serverip),1,file1);
fclose(file1);
}
bzero  ( buf,sizeof( buf) );
memset ( buf, 0, sizeof( buf) );

close(new_sock);
system("rm tmpcli");

for(i=0;i<3;i++)
{
if (( sock[i] = socket(AF_INET, SOCK_STREAM, 0)) < 0 ) {
printf("network client socket failure %d\n", errno);
perror("network client");
exit(2);
}
}
for(i=0;i<3;i++)
{
myname.sin_port = htons( atoi(serverip.p[i]));
myname.sin_family = AF_INET;
printf("Client connects server%d of address %s ",i+1,serverip.s[i]);
printf("of port number %d\n",ntohs(myname.sin_port));

hp = gethostbyname (serverip.s[i]);
if ( hp == NULL ) {
printf("network client gethostbyname failure %d\n",errno);
perror("network client");
close ( sock[i]);
exit(3);
}
else {
```

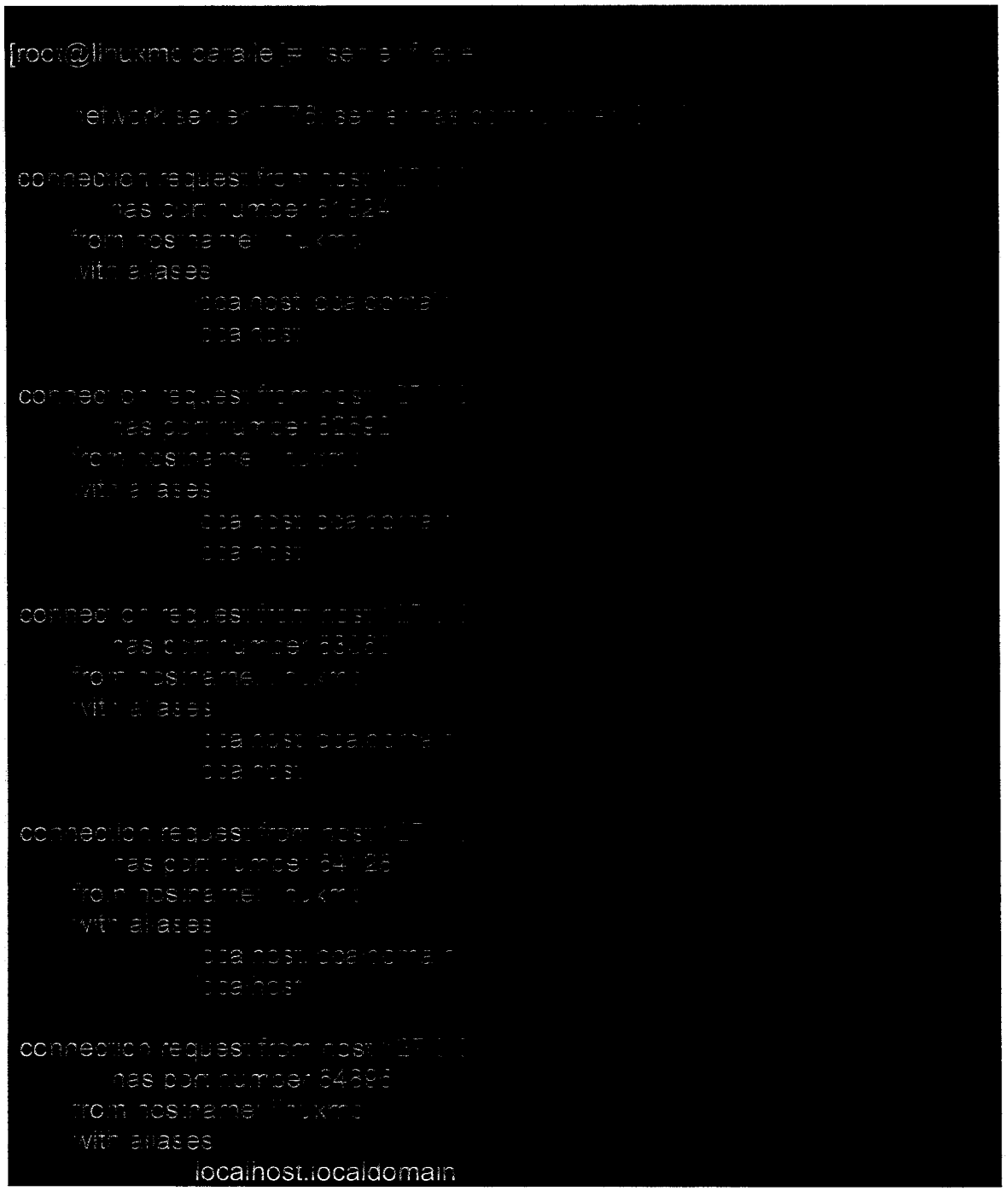
```
printf("Server%d information obtained:\n",i+1);
printf("\tThe sever%d IPAddress is %s\n\n",i+1,hp -> h_name);
}


bcopy ( hp -> h_addr_list[0], &myname.sin_addr.s_addr,
hp -> h_length );
memcpy ( &myname.sin_addr.s_addr, hp -> h_addr_list[0],
hp -> h_length );

if ( ( connect (sock[i],(struct sockaddr *)&myname, sizeof(myname)) ) < 0 )
{
printf("network client %s connect failure %d\n",
argv[0], errno);
perror("network client");
close (sock);
exit(4);
}

}
bzero  ( buf,    sizeof( buf) );
memset ( buf, 0, sizeof( buf) );
for(i=0;i<3;i++)
{
numbytes[i]=900;
counter[i]=0;
dn_time[i]=0;
con_time[i]=0;
t1[i]=0;
tr_time=0;
}
printf("\nEnter the output filename you need:\t");
scanf("%s",f_name);
f_no=strlen(f_name);
f_name[f_no]=49;
f_name[f_no+1]='\0';
file1=fopen(f_name,"w");
f_name[f_no]=50;
file2=fopen(f_name,"w");
f_name[f_no]=51;
file3=fopen(f_name,"w");
fseek(file1,0,SEEK_SET);
fseek(file2,0,SEEK_SET);
fseek(file3,0,SEEK_SET);
j=0;
for(;j==0;)
```

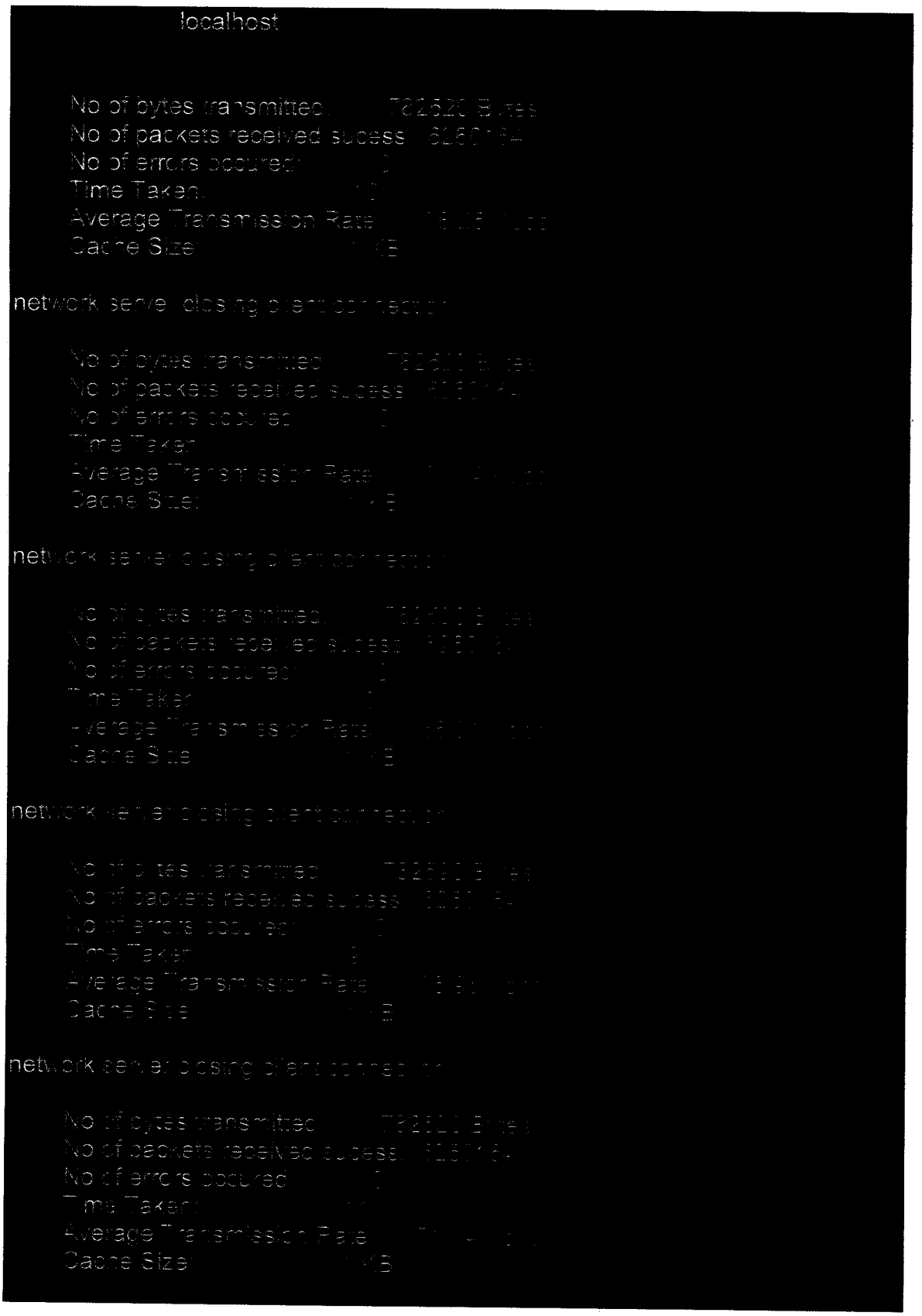
```
{
if((t1[0]==0)||(t1[1]==0)||(t1[2]==0))
{
for(i=0;i<=2;i+=1)
{
if(numbytes[i]==900)
{
times(&before);
strcpy(buf,"send");
write ( sock[i], buf, sizeof(buf) );
addr_len = sizeof(struct sockaddr);
if ( ( numbytes[i] = read (sock[i], buf,sizeof(buf)) ) < 0 ) {
printf("network client socket read failure &d\n", errno);
perror("network client");
close(sock);
exit(5);
}
else
{
if(i==0)
{
fwrite(&buf,numbytes[0],1,file1);
}
if(i==1)
{
fwrite(&buf,numbytes[1],1,file2);
}
if(i==2)
{
fwrite(&buf,numbytes[2],1,file3);
}
times(&after);
counter[i]+=numbytes[i];
dn_time[i]+=(after.tms_stime - before.tms_stime);
}
}
else
{ t1[i]=1;}
}
}
else
{
j=1;
}

}
```

```
fclose(file1);
fclose(file2);
fclose(file3);
file3=fopen("tmp.pg","w");
f_name[f_no]=49;
f_name[f_no+1]='\0';
d=10;
fprintf(file3,"./mer ""%s""%c",f_name,d);
fprintf(file3,"rm ""%s""%c",f_name,d);
f_name[f_no]=50;
fprintf(file3,"rm ""%s""%c",f_name,d);
f_name[f_no]=51;
fprintf(file3,"rm ""%s""%c",f_name,d);
fprintf(file3,"rm tmp.pg""%c",d);
fclose(file3);
system("sh tmp.pg");
f_name[f_no]='\0';
printf("\n\n\tThe file downloaded is named as %s\n\n",f_name);
for(i=0;i<3;i++)
close (sock[i]);
printf("Source\t  trans rate\tdoc. size\tbl. size\tdn.time\n");
printf("-------------------------------------------------------------");
for(i=0;i<3;i++)
{
tr_time=0.0;
tr_time=counter[i]/8000;
tr_time=tr_time/dn_time[i];
printf("\n%s%.2fKBps\t%d\t\t%d\t\t%d",
serverip.s[i],tr_time,counter[0]+counter[1]+counter[2],counter[i],dn_time[i]);
}
printf("\n");
exit(0);

}
```

## Sample output:

Output of the client screen:




localhost
No of bytes transmitted
No of packets received success
No of errors occured
Time Taken
Average Transmission Rate
Cache Size
network server closing client connection
No of bytes transmitted
No of packets received success
No of errors occured
Time Taken
Average Transmission Rate
Cache Size
network server closing client connection
No of bytes transmitted
No of packets received success
No of errors occured
Time Taken
Average Transmission Rate
Cache Size
network server closing client connection
No of bytes transmitted
No of packets received success
No of errors occured
Time Taken
Average Transmission Rate
Cache Size
network server closing client connection
No of bytes transmitted
No of packets received success
No of errors occured
Time Taken
Average Transmission Rate
Cache Size