MultiClient MultiInstance Centralized File Server using TCP protocol in Java Sockets

Arun Singh, Ajay Kr. Sharma, Ashish Kumar
Dr. B.R Ambedkar
NIT Jalandhar, India

ABSTRACT

In today's scenario File Server using reliable connection oriented protocol like TCP are well active, but majorly they are task specific like for chat server, document file server, image server, audio file server or Video Server etc. If they all are present together the server might be using other file transfer protocol and application is not transparent.

To promote easy and transparent file transfer across network using Java Sockets, we introduce a Java TCP Server. The Server is multiclient multithreaded multi format supported Java Application. The requesting client validates the user request at its end and improves server performance; on other hand Server provide data only to authenticated client and refuses connection to others. The server is profiled for number of clients for mixed type of Input files like text, doc, pdf, jpeg, mp3, flv and mp4. The client server design is well implemented on 2, 4, and 8 systems in our laboratory and tested for heap memory usage, server usage time and bandwidth utilization; results are well satisfied and become platform for this paper.

General Terms

Networking in Java, Java Socket, TCP.

Keywords

Java Server, File Server.

1. INTRODUCTION

All development in the field of network technology is used to promote fast and easy data transfer at low cost. There has been tremendous and continuous development in this direction. The network speed and capacity of workstation, made personal computers more powerful and efficient than supercomputers of only 10 years ago[3]. The development of the network technology promotes the information management and the application mode of the enterprise to develop in two directions. First, the traditional Client Server architecture to the structure mode of the web system that based on the multi-layer. Second, the network environment transfers from the concentrated processing environment to distributed environment [1].

Data file requirement is very necessary in today's environment, and if this environment is small like 5 to 50 computers, then the major challenge is to manage such file transfer at low cost and less complexity. The Internet system could not be used due to lack of security of confidential data and threat of Intruder. The Open Source file transfer protocols are difficult to implement and hard to use. The establishment and management of such protocols is again a specialized task.

So, there is requirement of personal Client Server application, with capability to support majorly used file types like txt, doc, pdf, jpeg, mp3, mp4, flv etc, with specific security in personal environment, and answer is MMCFS (MultiClient MultiInstance centralized File Server) using TCP protocol in Java Sockets.

2. RELATED WORK

Programmers find difficult to use socket programming when achieving functionality within the realm of distributed systems [1, 4, 5]. As a result, many turn to RMI (Remote Method Invocation) of Sun Microsystems or CORBA (Common Object Requests Broker Architecture) of OMG (Object Management Group), for distributed system. RMI, one of the core Java APIs since version 1.1., is a way that Java programmers can write object-oriented programs in which objects on different computers can interact with each other. A client can call a remote object in a server, and the server can also be a client of other remote objects. CORBA is well suited for Distributed system working on different platforms with varying specifications and environment. But the implementation of RMI and CORBA are very complex and costly for small applications, and if there is requirement of only single server, then Java Socket is better choice for easy implementation [1, 5, 9].

A P2P network is a network of computers or nodes where there is a concept of equality that is anyone could act like a client if required a data and the one who serves act like a server, it is different from traditional client/server architecture where only dedicated system could act like a server and requesting computers would acts like client to it. If one of the client have the data required by other client, in any case it could not serve him, this become a major drawback in traditional client/server architecture. Since all the nodes are equal in a P2P network, there is no requirement of dedicated server, as each node can act like both client/server and free to

add or leave the network any time, without disturbing others peers.
The P2P network architecture is mainly grouped into two categories: centralized systems (Napster, BitTorrent etc.) and decentralized systems (Gnutella, FreeNet etc.)[6].
It is found that P2P file sharing accounts for much more traffic than any other application on the Internet.

3. Design and Implementation of MMCFS

A simple client/server architecture, with single server connected to multiple clients by default 50. Figure 1 shows the architecture view of MMCFS. The server composed of functionality of creating multiple instances for each client, as

```
public class ts implements Runnable
{
Socket S;
ts(Socket S)
{
this.S=S;
}
public static void main(String aa[])
{
ServerSocket SS = new ServerSocket(8000);
While(true){
Socket S1 = SS.accept();
new Thread(new ts(S1)).start();
..
}}}
```

After accepting the client connection, the server accesses the client Ipaddress and checks it with the list of authenticated clients with it. If the client Ipaddress present in its list then it allows authenticated client to access the data file otherwise close the connection at the same time without any transfer. The required code is as

```
DataInputStream dis0=new DataInputStream (new FileInputStream ("auth.txt"));
while((str0=dis0.readLine())!=null)
{
if(str0.equals((S1.getInetAddress()).toString()))
{
new Thread(new ts(S1)).start();
flag0=true;
break;
}}
if(!flag0)
{
S1.close();
}
```

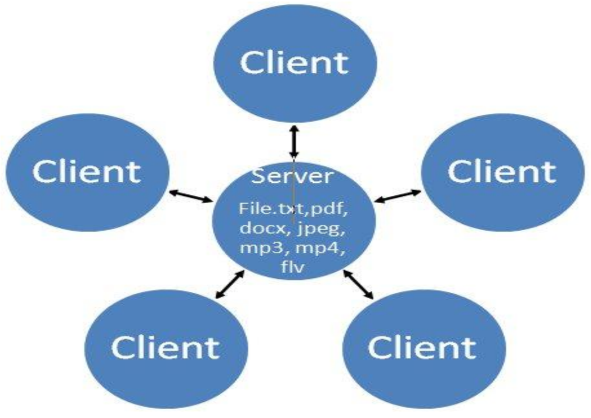


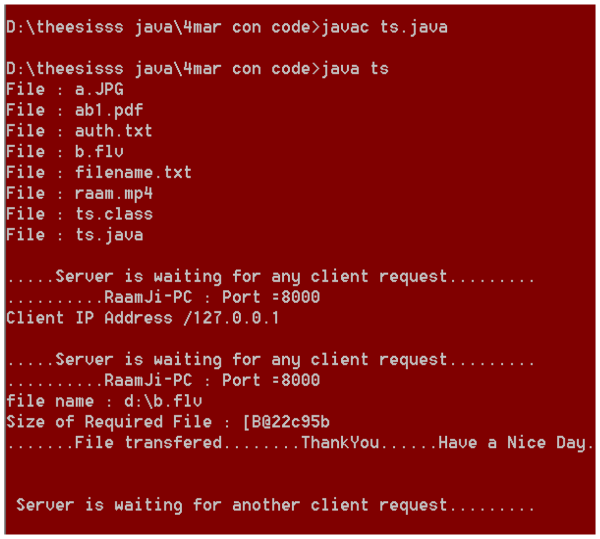
Figure 1. The Architecture view of MMCFS

The server Administrator is provided the functionality of looking the file present at server repository and client request for the file as

```
DataOutputStream dos0=new DataOutputStream(new FileOutputStream("d:\\filename.txt"));
File folder=new File("d:\\shared");
File[] lof=folder.listFiles();
for(int i=0;i<lof.length;i++)
{
if(lof[i].isFile())
{
System.out.println("File : "+lof[i].getName());
dos0.writeBytes((lof[i].getName())+"\r\n");
}
}
dos0.close();
```

The waiting server looks like as in Figure 2. The server automatically map the client request for the respective file present in its repository otherwise return no match, hence breaking down the connection with the client. The server entertains client request for file by following code.

```
BufferedInputStream bis = new BufferedInputStream(fis);
try {
bis.read(buf, 0, buf.length);
out = S.getOutputStream();
out.write(buf,0,buf.length);
out.flush();
}
catch (IOException e1
{
e1.printStackTrace();
}
```



```
D:\theesisss java\4mar con code>javac ts.java

D:\theesisss java\4mar con code>java ts
File : a.JPG
File : ab1.pdf
File : auth.txt
File : b.flv
File : filename.txt
File : raam.mp4
File : ts.class
File : ts.java

.....Server is waiting for any client request.........
..........RaamJi-PC : Port =8000
Client IP Address /127.0.0.1

.....Server is waiting for any client request.........
..........RaamJi-PC : Port =8000
file name : d:\b.flv
Size of Required File : [B@22c95b
.......File transfered........ThankYou......Have a Nice Day.


 Server is waiting for another client request.........
```

Figure 2. The Server Instance in Waiting State

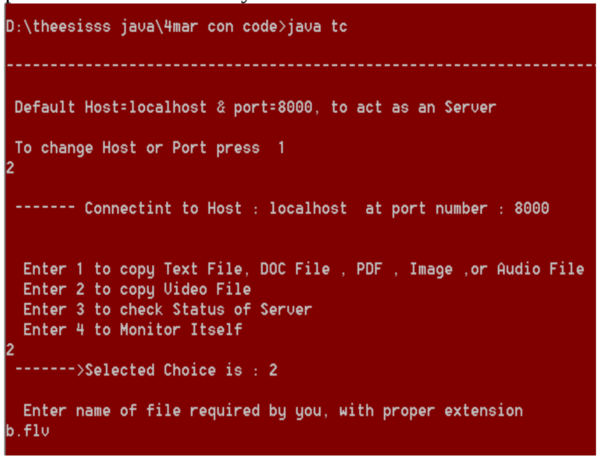
The client is provided with the functionality of authenticating client request at its end, by checking the connecting server Ipaddress at specified port. The requesting client is shown in figure 3. The client is dynamically connected to default server, if required Ipaddress and port are not correct or does not exist. There is an additional functionality of requesting the file efficiently, by categorizing the request. This categorization is much more efficient for video files and support multiple video formats, as shown in figure4.
Java code at client side for requesting files of multiple formats like txt, doc, pdf, jpeg, and mp3 etc. is as

```
BufferedOutputStream bos = new
BufferedOutputStream(fos);
try {
num_bytes_read = in.read(buf,0,buf.length);
current=num_bytes_read;
do{
num_bytes_read = in.read(buf, current,(buf.length - current));
if(num_bytes_read>=0)current += num_bytes_read;
}while(num_bytes_read>-1);
bos.write(buf, 0,current);
bos.flush();
bos.close();
} catch (IOException e)
{
e.printStackTrace();
}
```

Java code specially for requesting mp4, flv etc. is as

```
BufferedOutputStream bos = new
BufferedOutputStream(fos);
try {
num_bytes_read=64;
do{
num_bytes_read=in.read(buf,current,num_bytes_read);
bos.write(buf, current, num_bytes_read);
current=current + num_bytes_read;
}while(num_bytes_read!=0);
bos.flush();
bos.close();
}
catch (IOException e) {
e.printStackTrace();
}
```

Java code written specially for mp4 and flv file increases the performance and accuracy for file transfer.



```
D:\theesisss java\4mar con code>java tc

--------------------------------------------------------------

 Default Host=localhost & port=8000, to act as an Server

 To change Host or Port press  1
2

 ------- Connectint to Host : localhost  at port number : 8000


  Enter 1 to copy Text File, DOC File , PDF , Image ,or Audio File
  Enter 2 to copy Video File
  Enter 3 to check Status of Server
  Enter 4 to Monitor Itself
2
 ------->Selected Choice is : 2

  Enter name of file required by you, with proper extension
b.flv
```

Figure 3. The Client Instance in Requesting State

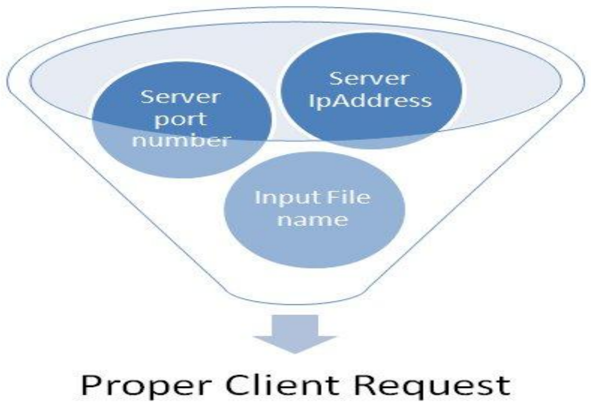


Figure 4. The Client Filtering process

4. Performance Evaluation

The implemented server is tested on Systems with Intel(R) Pentium Processor(R) D CPU 2.80 GHz 2.79 GHz, 1.25 GB Ram, Windows 7 (32 bit) Operating System.
The server contains a dedicated folder to respond to all client requests, the file contained in a folder composed of various extensions and sizes, as stated in table 1.

The clients have access to only dedicated folder, hence provide a security; also clients have permission only to copy a file.

Table 1. The Server Repository

File Extension	File size
Txt	822 Bytes
Java	1.99 KB
Docx	13.0 KB
PDF	52.0 KB
HTM	19.4 KB
MP3	6.80 MB
MP4	10.1 MB
FLV	39.6 MB

Server is tested for 2, 4, and 8 numbers of clients, with multiple requests in mixed format. The Heap memory utilized by CPU running the server for responding to different number of clients for different period of time is shown in Figure 5, 6, and 7.

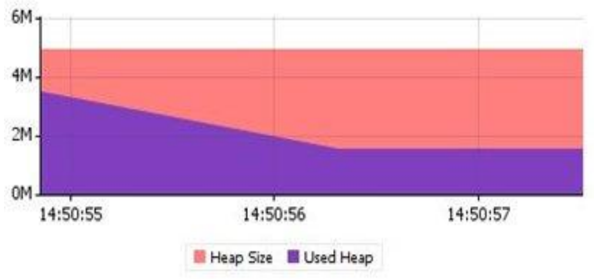


Figure 5. Server Connected to 2 Clients

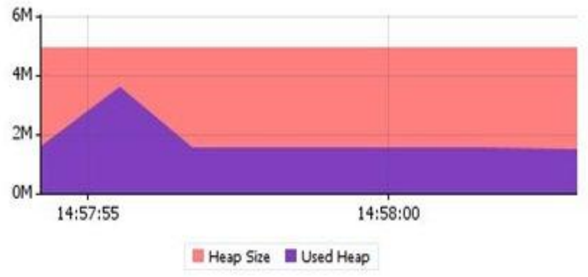


Figure 6. Server Connected to 4 Clients

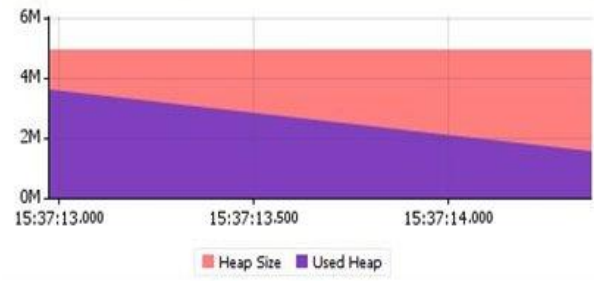


Figure 7. Server Connected to 8 Clients

The Figure 5, 6 and 7 shows that for mixed format file transfer, the heap memory utilization does not exceed 4M for given input files and after the file transfer the memory consumption decreases slowly.

The Figure 8, 9 and 10 specially shows the interaction time of clients with the server for file transfer that is serving time versus time utilized for garbage collection. In any case the percent of time utilized for garbage collection is negligible and shown by blue line traced along x axis.

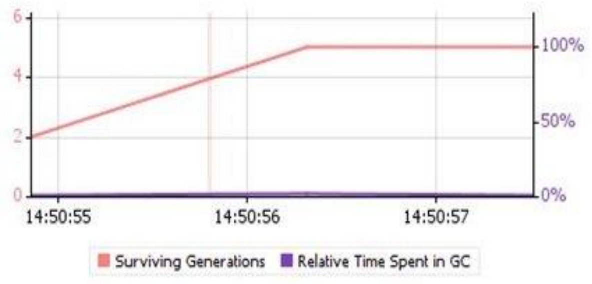


Figure 8. Server Connected to 2 Clients

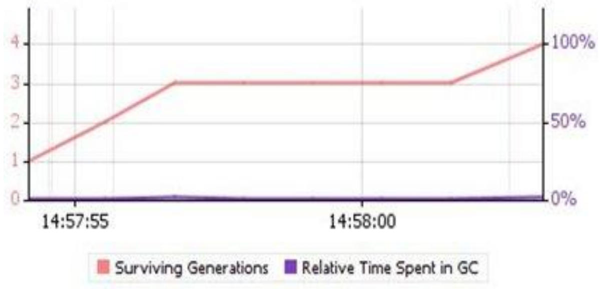


Figure 9. Server Connected to 4 Clients

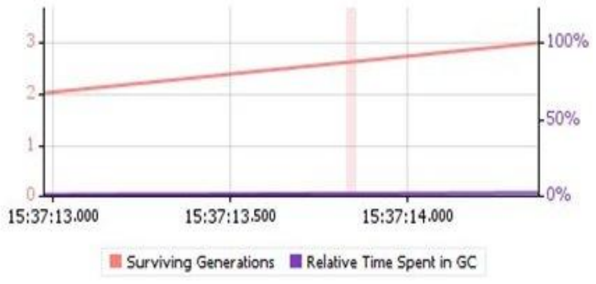


Figure 10. Server Connected to 8 Clients

The figure 11, 12 and 13 shows network bandwidth utilization for file transfer present at server which is mentioned in Table 1.

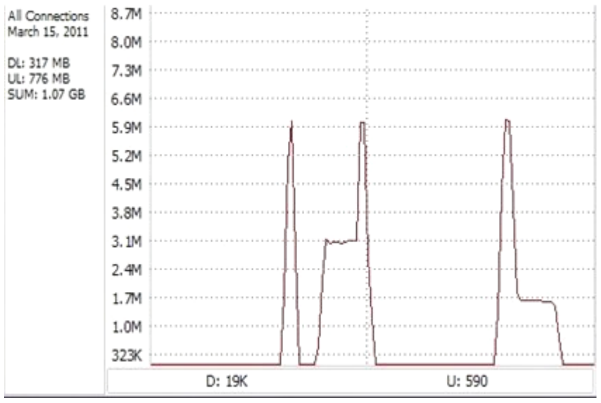


Figure 11. Bandwidth Utilization for 2 Clients

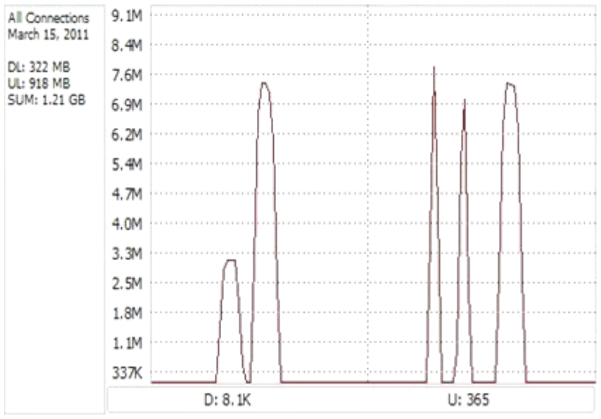


Figure 12. Bandwidth Utilization for 4 Clients

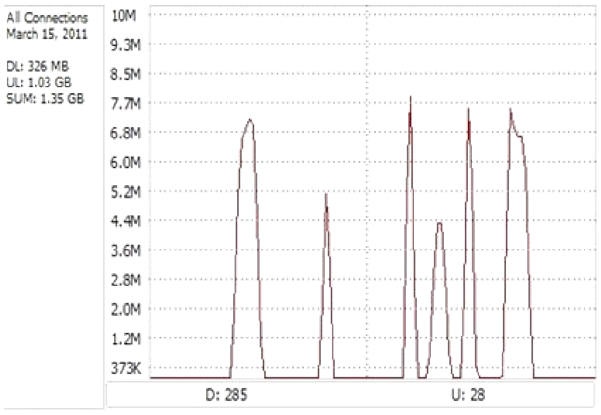


Figure 13. Bandwidth Utilization for 8 Clients

Figure 11, 12 and 13 shows that for as the number of clients increases their connecting request and file request will increase correspondingly hence increasing bandwidth utilization.

5. Conclusion and Future Work

This article puts forward a kind of MultiClient MultiInstance Centralized Java Server, design and implemented for limited number of users. The client/server system proposed is effectively deployed in our laboratory, with a dedicated server and 0 to 50 clients at a time connecting to it. The system supports a large variety of files with varying sizes and tested for heap memory utilization which does not increase 4M in any case and maximum bandwidth utilization of 7.7M for defined input files at server. The transfer is implemented for authenticated users in personal network, hence secured.

The work is going on to make server lighter by distributing server job with clients connecting with it and also with peer servers.

6. REFERENCES

[1] Delin Hou and Huosong Xia,"Design of Distributed Architecture based on Java Remote Method Invocation Technology", pp. 618, 2009 International Conference on Environmental Science and Information Application Technology (IEEE Computer Society 2009).

[2] SeungJun Bang and JinHo Ahn,"Implementation and Performance Evaluation of Socket and RMI based Java Message Passing Systems", pp.153, Fifth International Conference on Software Engineering Research, Management and Application (IEEE Computer Society 2007).

[3] J.T. Rough and A.M. Goscinski,"The development of an efficient checkpointing facility exploiting operating systems services of the GENESIS cluster operating system", Future Generation Computer Systems, Vol. 20, No. 4, pp 523-538, 2004.

[4] V.Getov, G. von Laszewski, M. Philippsen, I. Foster. Multi-paradigm Communications in Java for Grid Computing. Communications of the ACM, Vol. 44, No. 10, pp. 118-125, 2001.

[5] R. Metkowski and P.Bala, "Parallel Computing in Java: Looking for the Most Effective RMI Implementation for Clusters", Lecture Notes in Computer Science, Springer-Verlag Berlin, Vol. 3911, pp.272-277,2006.

[6] Amol Vasudeva, Sandeepan, Nitin Kumar," PASE: P2P Network Based Academic Search and File Sharing Application", 2009 First International Conference on Computational Intelligence, Communication Systems and Networks.

[7] What is peer-to-peer? Peer-to-Peer working group http://www.p2pwg.org/

[8] Martin Alt, Sergei Gorlatch, "Adapting Java RMI for grid computing", Future Generation Computer Systems, Vol 21, pp699-707, 2005.

[9] J2SE 1.5.0 API Specification, http://java.sun.com.

[10] Java Socket Tutorial, http://www.cs.swan.ac.uk/~csneal/InternetComputing/JavaSockets.html.

[11] C. Nester, M. Phillippsen, and B. Haumacher, "A more efficient RMI for java", In Proc. Of the ACM Java grande Conference, pp. 152-159, June 1999.

Arun Singh is an M.Tech. Student in computer science & engineering department of Dr. B R Ambedkar National Institute of Technology. He has completed his B.Tech. Degree in 2009 from A.K.G.E.C Ghaziabad affiliated to Uttar Pradesh Technical University (Lucknow). He is SCJP certified (2009). His research area is Distributed System and Computer Networks with a special interest in Java Technologies.

Ajay k Sharma received his BE in Electronics and Electrical Communication Engineering from Punjab University Chandigarh, India in 1986, MS in Electronics and Control from Birla Institute of Technology (BITS), Pilani in the year 1994 and PhD in Electronics Communication and Computer Engineering in the year 1999. His PhD thesis was on "Studies on Broadband Optical Communication Systems and Networks". From 1986 to 1995 he worked with TTTI, DTE Chandigarh, Indian Railways New Delhi, SLIET Longowal and National Institute of technology (Erstwhile Regional Engineering College), Hamirpur HP at various academic and administrative positions. He has joined National Institute of Technology (Erstwhile Regional Engineering College) Jalandhar as Assistant Professor in the Department of Electronics and Communication Engineering in the year 1996. From November 2001, he has worked as Professor in the ECE department and presently he is working as Professor in Computer Science & Engineering in the same institute. His major areas of interest are broadband optical wireless communication systems and networks, dispersion compensation, fiber nonlinearities, optical soliton transmission, WDM systems and networks, Radio-over-Fiber (RoF) and wireless sensor networks and computer communication. He has published 237 research papers in the International/National Journals/Conferences and 12 books. He has supervised 12 Ph.D. and 36 M.Tech theses. He has completed two R&D projects funded by Government of India and one project is ongoing. Presently he was associated to implement the World Bank project of 209 Million for Technical Education Quality Improvement programme of the institute. He is technical reviewer of reputed international journals like: Optical Engineering, Optics letters, Optics Communication, Digital Signal Processing. He has been appointed as member of technical Committee on Telecom under International Association of Science and Technology Development (IASTD) Canada for the term 2004-2007 and he is Life member of Indian Society for Technical Education (I.S.T.E.), New Delhi..

Ashish Kumar is an M.Tech. Student in computer science & engineering department of Dr. B R Ambedkar National Institute of Technology.He has completed his B.Tech. Degree in 2007 from Vivekananda Institute Of Technology Bangalore affiliated to Visvesvaraya Technological University (Belgaum). He is CCNA cerified. His research area is Computer Networks, Distributed System and Operating System.