Assignment1—Report

Code of client:

```
IServer server =  null;
IServerFactory stub =
(IServerFactory)Naming.lookup("rmi://"+serverMachine+":"+serverPort+"/"+serve
rName);
server = stub.createServer();
```

Create a stub which is a proxy for RMI server on client side and handles remote invocation on behave of RMI client. And try to lookup on the Registry Service.

Code in put() method of client:

```
if(mode == Mode.PASSIVE)
{
	svr.put(inputs[1]);
	File f = new File(inputs[1]);
	FileInputStream in = new FileInputStream(f);
	Socket xfer = new Socket(serverAddress, serverSocket.getPort());
	DataOutputStream dos = new DataOutputStream(new BufferedOut-
putStream(xfer.getOutputStream()));

	dos.writeUTF(f.getName());
	dos.flush();
	dos.writeLong(f.length());
	dos.flush();

	byte[]  fbuf =  new byte[BUFFER_SIZE];
	int length = 0;

	while((length =in.read(fbuf,0,fbuf.length))>0)
	{
		dos.write(fbuf,0,length);
		dos.flush();
	}
	System.out.println("---Transfer file success----");

	in.close();
	dos.close();
	xfer.close();
}
else if(mode == Mode.ACTIVE)
{
	FileInputStream f = new FileInputStream(inputs[1]);
	new Thread(new PutThread(dataChan,f)).start();
	msgln("start a new thread to get the file!");
	svr.put(inputs[1]);
}
```

In this snippet, if the mode is passive which means the server should start a thread to waiting for the client send socket connection, in here, we remotely invoke the method svr.put() to start the thread in putThread() for the server side. The client build the socket and get connected to the server, create the dataoutputstream object to handle the file bytes and send it to the server .After finished the while loop print out the sentence “transfer file success” and close those object; if the mode is active which indicate that the client will run a putThread() to waiting for the server to connect. At the same, invoking the method of Iserver to make such connection.

Code in the get() method of client:

```
if (mode == Mode.PASSIVE) {
      svr.get(inputs[1]);
      FileOutputStream f = new FileOutputStream(inputs[1]);
      Socket xfer = new Socket(serverAddress, serverSocket.getPort());
      /*
       * TODO: connect to server socket to transfer file.
       */
      DataInputStream dis = new DataInputStream(new
BufferedInputStream(xfer.getInputStream()));
      byte[]  fbuf =  new byte[BUFFER_SIZE];
      int bytesRead = dis.read(fbuf,0,BUFFER_SIZE);
      while(bytesRead!=-1)
      {
            f.write(fbuf,0,bytesRead);
            f.flush();
            bytesRead = dis.read(fbuf,0,BUFFER_SIZE);
      }
      msgln("-----Success Receiving File-----");
      dis.close();
      f.close();
      xfer.close();
} else if (mode == Mode.ACTIVE) {

      FileOutputStream f = new FileOutputStream(inputs[1]);
      new Thread(new GetThread(dataChan, f)).start();
      msgln("start a new thread to get the file!");
      svr.get(inputs[1]);

} else {
      msgln("GET: No mode set--use port or pasv command.");
```

In this get() method, client using this to get file from server side, when the mode is PASSIVE,

First call the get method of the server which will start a thread to listen. The client start a socket object xfer to connect which is quite same with the code above ,just write and read from the buffer and input,output stream. If the mode is active, the client just start the getThread and waiting for the server to make the connection. Invoking the get()method to let the server know what is going on.

Code in getThread() method of client:

```
Socket xfer = dataChan.accept();
DataInputStream dis =new
DataInputStream(newBufferedInputStream(xfer.getInputStream()));
byte[] sendBytes = new byte[BUFFER_SIZE];
String fileName = dis.readUTF();
long fileLength = dis.readLong();


msgln("start to receive "+fileName+" and size is "+fileLength);

while(true){
       int read = 0;
      read=dis.read(sendBytes);
      if(read==-1)
            break;
      file.write(sendBytes,0,read);
      file.flush();
      }
msgln("-----Success Receiving File!!-----");

dis.close();
file.close();
xfer.close();
```

This method will be invoked when the mode is ACTIVE and the command is get the file from the server. First ,getting the file name and file size for printing ,and start read the bytes from the DataInputStream object.Using the FileOutputStream to write the bytes into file.

Code in putThread() method of client：

```
Socket xfer = dataChan.accept();
DataOutputStream dos = new DataOutputStream(new
BufferedOutputStream(xfer.getOutputStream()));

byte[]  fbuf =  new byte[BUFFER_SIZE];
int length = 0;

while((length =file.read(fbuf,0,fbuf.length))>0)
{
      dos.write(fbuf,0,length);
      dos.flush();
}
System.out.println("---Transfer file success----");
file.close();
dos.close();
xfer.close();
```

In this piece of code, it is quite similar to the GetThread() method ,just start a new thread to waiting for the server to connect. And write the data into the DataOutputStream object.

Now it is time of the server side

The code in put() for the server:

```
} else if (mode == Mode.ACTIVE) {
      Socket xfer = new Socket (clientSocket.getAddress(), clientSocket.get-
Port());
      /*
       * TODO: connect to client socket to transfer file.
       */
      DataOutputStream dos = new DataOutputStream(new BufferedOut-
putStream(xfer.getOutputStream()));
      File f = new File(path()+file);
      FileInputStream fis = new FileInputStream(f);

      dos.writeUTF(f.getName());
      dos.flush();
      dos.writeLong(f.length());
      dos.flush();

      byte[] fbuf = new byte[BUFFER_SIZE];
      int length = 0;

      while((length = fis.read(fbuf,0,fbuf.length))>0)
      {
            dos.write(fbuf,0,length);
            dos.flush();
      }
      System.out.println("Transfer file !!");
      fis.close();
      dos.close();
      xfer.close();

      /*
       * End TODO.
       */
} else if (mode == Mode.PASSIVE) {
    FileInputStream f = new FileInputStream(path()+file);
    new Thread (new GetThread(dataChan, f)).start();
    System.out.println("Wait for the client to connect...");
}
```

Server will run the code when the client remotely invoke, if the mode is ACTIVE, the server will start the connection to the client with sending the file name and file size. If the mode is passive, just call the GetThead Methods to start a new thread to wait for the connection.

The put method of server side:

```
if (!valid(file)) {
    throw new IOException("Bad file name: " + file);
}
else if (mode == Mode.ACTIVE)
{
      System.out.println("Start connect to the server...");
      Socket xfer = new Socket (clientSocket.getAddress(),
clientSocket.getPort());
      FileOutputStream fos = new FileOutputStream(path()+file);
      DataInputStream dis = new DataInputStream(new
BufferedInputStream(xfer.getInputStream()));
      byte[] fbuf = new byte[BUFFER_SIZE];
      int bytesRead = dis.read(fbuf,0,BUFFER_SIZE);
      while(bytesRead!=-1)
      {
            fos.write(fbuf,0,bytesRead);
            bytesRead = dis.read(fbuf);
      }
      fos.close();
      dis.close();
      xfer.close();
}
else if (mode == Mode.PASSIVE)
{
      FileOutputStream f = new FileOutputStream(path()+file);
      new Thread(new PutThread(dataChan,f)).start();
      System.out.println("----Start thread to wait for connction(passive)----
-");
}
```

If the put method is called and mode is equal to ACTIVE, the server start connecting to client and begin to receive file from the client and output the file in the current path. When the mode is PASSIVE, the server just start a new thread to listen the connection from the clients.

The getThread of server:

```
Socket csocket = dataChan.accept();
DataOutputStream dos = new DataOutputStream(new
BufferedOutputStream(csocket.getOutputStream()));
byte[]  fbuf =  new byte[BUFFER_SIZE];
int bytesRead = file.read(fbuf,0,BUFFER_SIZE);
while(bytesRead!=-1)
{
      dos.write(fbuf,0,bytesRead);
      bytesRead = file.read(fbuf);
}
file.close();
dos.close();
csocket.close();
```

The PutThread of server:

```
Socket csocket = dataChan.accept();
DataInputStream dis = new DataInputStream(new
BufferedInputStream(csocket.getInputStream()));
byte[] sendBytes = new byte[BUFFER_SIZE];
String fileName = dis.readUTF();
long fileLength = dis.readLong();

System.out.println("start to receive "+fileName+" and size is "+fileLength);

while(true){
      int read = 0;
      read=dis.read(sendBytes);
      if(read==-1)
            break;
      file.write(sendBytes,0,read);
      file.flush();
}
System.out.println("-----Success Receiving File!!-----");

file.close();
dis.close();
csocket.close();
```

Testing code: Start the server in local and start client, when the client found the proxy server in the registry, the command line start to show.

Test in the passive mode:

The get method:

```
ftp> pasv
PASV: Server in passive mode.
ftp> dir
test.txt
client.policy
ftp> get test.txt
-----Success Receiving File-----
ftp>
```

The put method:

```
ftp> dir
test.txt
client.policy
ftp> ldir
client.policy
ftp.jar
ftp.sh
ftpd.jar
put_test.txt
test.txt
ftp> put put_test.txt
---Transfer file success----
ftp> ldir
client.policy
ftp.jar
ftp.sh
ftpd.jar
put_test.txt
test.txt
ftp>
```

Testing in the passive mode, to get or put file with the server and correct the code, make sure the file that been transferred is complete.

Changing it to active mode and do the same thing like the get and put methods.

Using dir to show the current root that the get_test.txt is already in the server side:

Test the get method：

```
ftp> pwd
CWD: /
ftp> dir
get_test.txt
ftp> port
listening on port: 33769
PORT: Server in active mode.
ftp> ldir
ftp.jar
client.porl
ftp.sh
test.tt
test.txt
put_test.txt
ftp
client.policy
ftp> get get_test.txt
start a new thread to get the file!
start to receive get_test.txt and size is 5
ftp> -----Success Receiving File!!-----
```

Test in active mode:

Using the put methods to put the put_test.txt:

```
ftp> put put_test.txt
start a new thread to get the file!
---Transfer file success----
```

The video will make sure the code is right and show the demon.