95-702 Distributed Systems for ISM

Project 2 Client-Server Computing

Five Tasks

🏁 Submit to Canvas a ***single PDF file*** named Your_Last_Name_First_Name_Project2.pdf along with a single zip file

The single PDF red flag. It is important that y lso important to ***be prepared*** to on. Be sure to provide your na

The five IntelliJ f your WHOLE IntelliJ project f ntain one client and one server. in the middle. For each project, zi in IntelliJ.

When all of you o one big zip file for submission.

Learning

Our **first object** UDP). UDP is used in many interne Host Configuration Protocol (DHCP ing and Voice over IP (VoIP) use UDP. We use UDP when we need high performance and do not mind an occasional dropped packet.

TCP, on the other hand, is also widely used. It works hard to make sure that not a single bit of information is lost in transit. The Hyper Text Transfer Protocol (HTTP) uses TCP. We are not using TCP in this project.

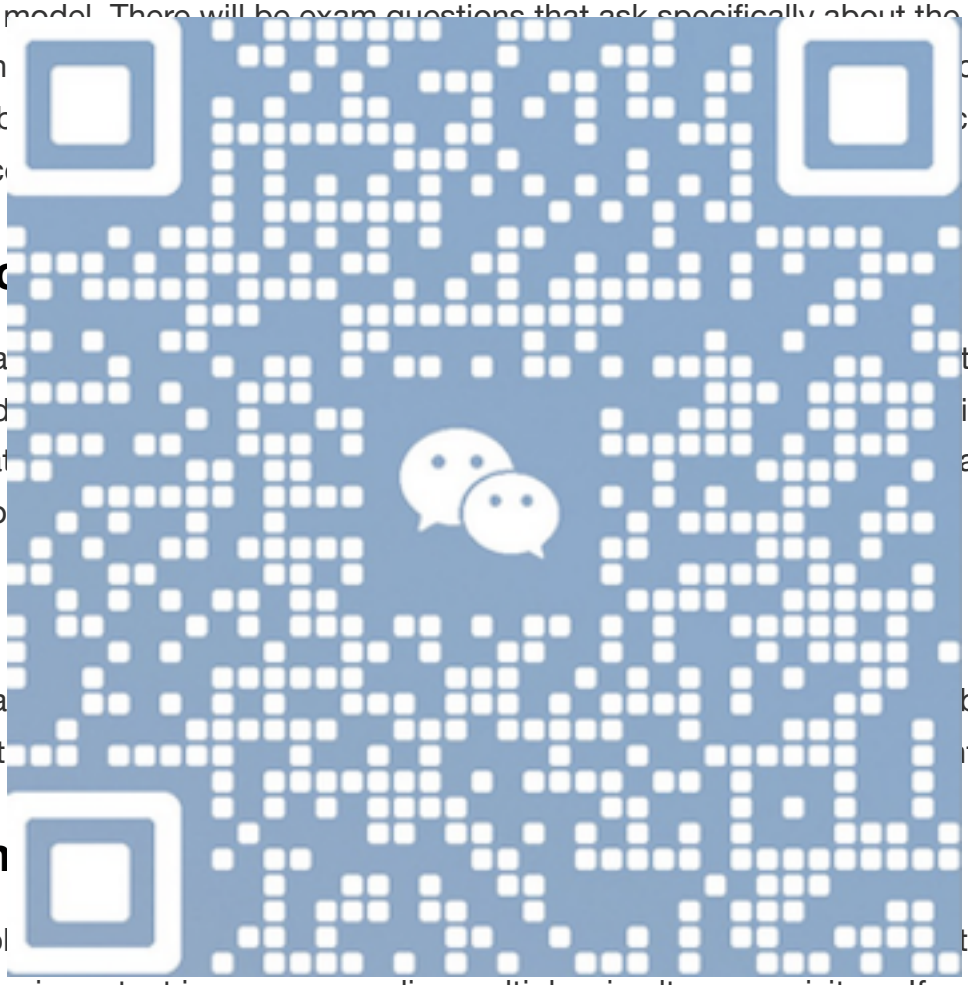
Our **second objective** is to understand the implications of a malicious player in the middle.

Our **third objective** is for you to understand the abstraction provided by Remote Procedure Calls (RPC's). We do this by asking that you use a proxy design and hide communication code and keep it

separate from your application code. RPC has been used for four decades and is at the foundation of many distributed systems.

Our **fourth objective** is the learn how to distribute a stand alone application. We use a simple neural network as our application. Our intent is not to study neural networks in a class on distributed systems. But some of you might decide to dig further into neural networks and use this application as a starting point.

Optionally, you may use a large language model (based on neural networks), such as ChatGPT or Copilot, to create some of your code. Task 0, Task 1, and Task 4 must be done without the help of a large language model. There will be exam questions that ask specifically about the code in these three tasks. Wh optional. There will be questions ab ce different students may c



Submissio

When you are a ted. Points will be deducted if cod in a comment describing what ation for an example of goo

Rubric

See the Genera bric for this assignment but t will be evaluated.

Some sim

In all of what fol t at a time. We are not exploring the important issues surrounding multiple, simultaneous visitors. If you write a multi-threaded server to handle several visitors at once, that is great but is not required. It gains no additional credit.

In addition, for all of what follows, we are assuming that the server is run before the client is run. If you want to handle the case where the client is run first, without a running server, that is great but will receive no additional credit.

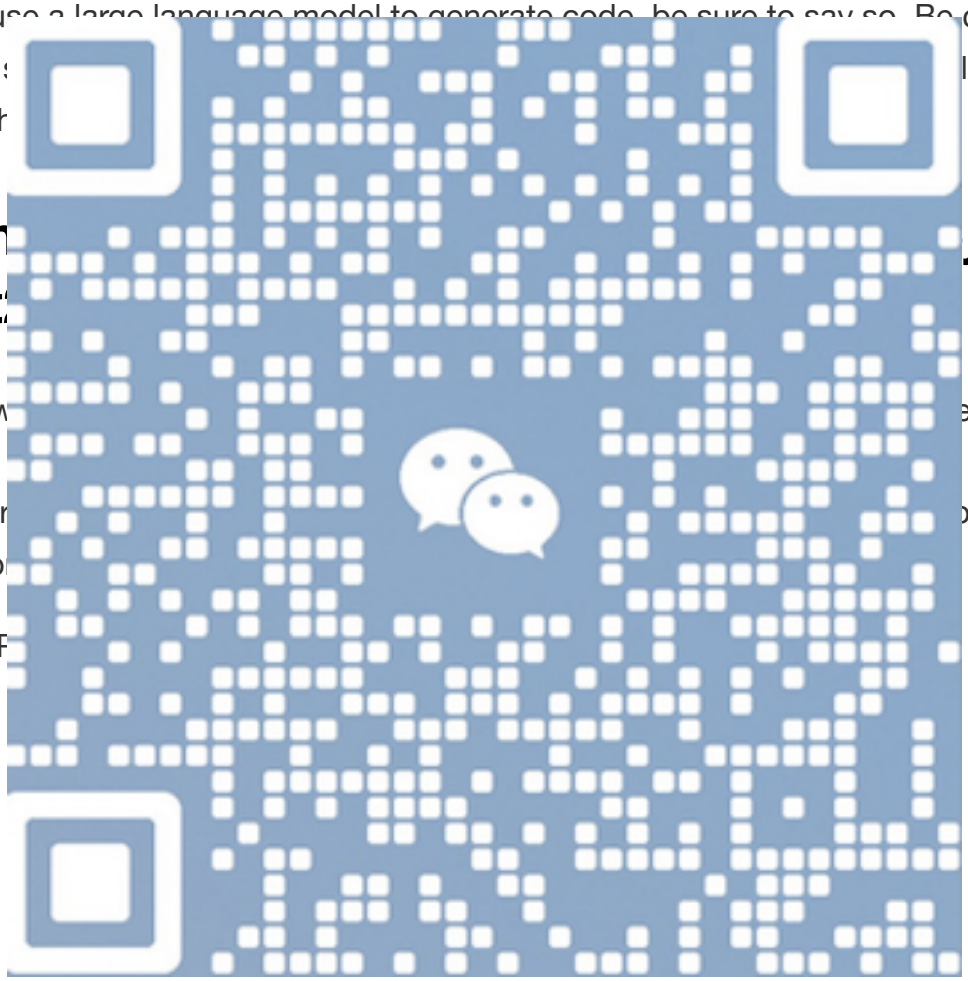
In Task 1, we are assuming that the server is run before the malicious player and the malicious player is run before the client.

In this assignment, you need not be concerned with data validation. You may assume that the data entered by users is correctly formatted.

In general, if these requirements do not explicitly ask for a certain feature, then you are not required to provide that feature. No additional points are awarded for extra features.

Cite your sources

If you use any code that is not yours (including code from a large language model), you are required to clearly cite the source - include a full URL in a comment and place it just above the code that is copied. If you use a large language model to generate code, be sure to say so. Be careful to cite your sources. If you s lude proper citations then th



Task 0 in ject "Project

In Task 0, you w entUDP.java. Note that
these two progr blication in IntelliJ. Both of these p

EchoServerUDP

```
import java.net.*;
import java.io.*;
public class EchoServerUDP{
        public static void main(String args[]){
        DatagramSocket aSocket = null;
        byte[] buffer = new byte[1000];
        try{
                aSocket = new DatagramSocket(6789);
                DatagramPacket request = new DatagramPacket(buffer, buffer.length);
                while(true){
                        aSocket.receive(request);
                                                                    est.getData(),
                                                                    Port());
                                                                    ata());


        }cat                                                        etMessage());
        }cat                                                        ge());
        }fin
        }
}
```

Note the differe r uses a DatagramPacke DatagramPacket to send data ba ased on a byte array. So, to ser sage to a byte array. To receive ay to a String message (if we

Note below how the client does the same thing. The client wants to send a String message. So, it extracts a byte array from the String (the variable m). And we then use m to build the DatagramPacket.

When the client receives a reply, the method reply.getData() returns a byte array - which we use to build a String object.

EchoClientUDP.java from Coulouris text

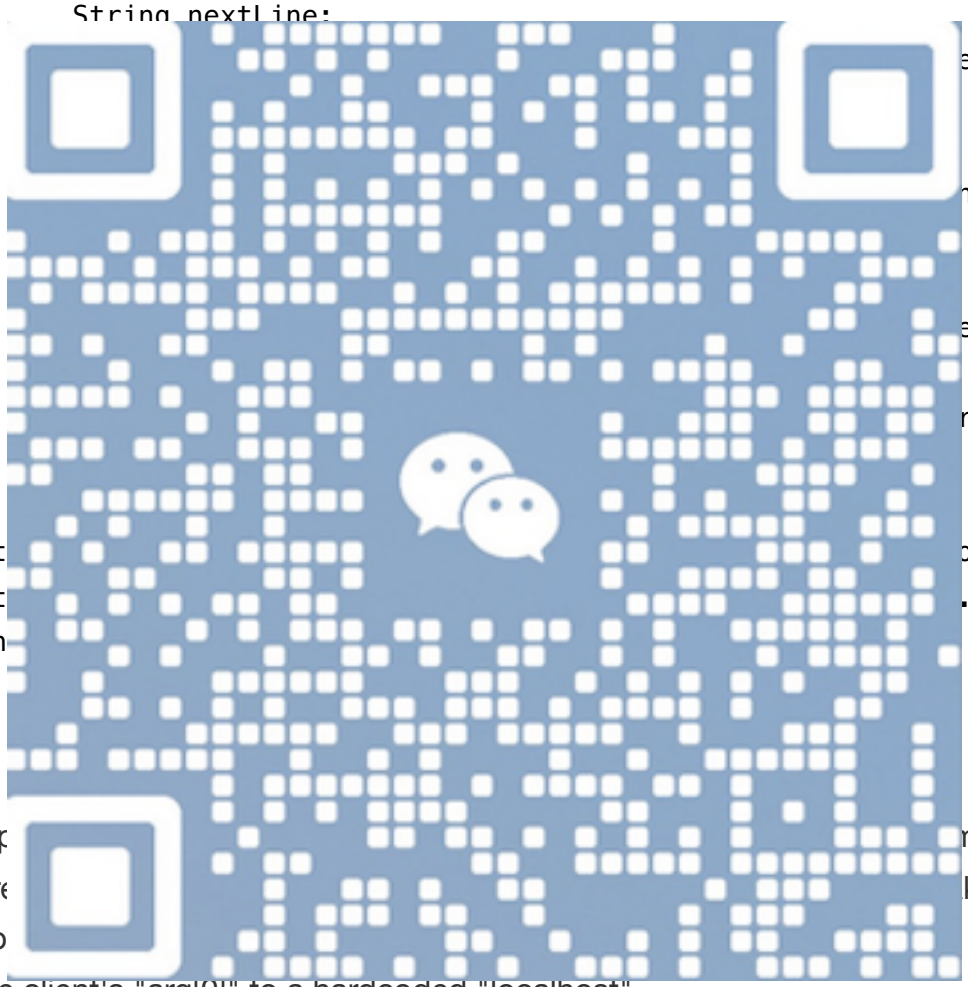
```
import java.net.*;
import java.io.*;
public class EchoClientUDP{
    public static void main(String args[]){
        // args give message contents and server hostname
        DatagramSocket aSocket = null;
        try {
                InetAddress aHost = InetAddress.getByName(args[0]);
                int serverPort = 6789;
                aSocket = new DatagramSocket();
                String nextLine;
                                                        eamReader(System.


                                                        ngth, aHost, serv


                                                        er.length);


                                                        reply.getData()))
    }

        }cat                                            on: " + e.getMess
        }cat                                            .getMessage());
        }fin
    }
}
```



0. Get these p me Intellij project and you are ke the following modificatio
1. Change the client's "arg[0]" to a hardcoded "localhost".
2. Document the client and the server. Describe what each line of code does.
3. Add a line at the top of the client so that it announces, by printing a message on the console, "The UDP client is running." at start up.
4. After the announcement that the client is running, have the client prompt the user for the server side port number. It will then use that port number to contact the server. For now, enter 6789.
5. Add a line at the top of the server so that it announces "The UDP server is running." at start up.
6. After the announcement that the server is running, have the server prompt the user for the port number that the server is supposed to listen on. Enter 6789 when prompted.