Department of ICT
Faculty of Technology
University of Ruhuna

ICT3253 – Distributed and Cloud Computing **Level 3 - Semester –**

2 Lab Sheet 07

Objectives:
Understanding of basic concepts of socket programming and practicing socket programming with TCP and UDP implementations.
Java should be used as the programming language for the implementations of distributed computing.

Part A

Answers the following questions and save your work as pdf file by using your index number (Ex: Lab07_PartA_TGXXX.pdf) and submit it to the given submission link in the LMS.

01. What is the concept of Java socket programming?

Java socket programming involves the use of sockets, which are endpoints for sending and receiving data across a network. It enables communication between two or more devices over a network, such as the internet. Sockets provide a low-level, bidirectional communication channel that allows data to be transferred between programs running on different machines.

02. What are the two ways of socket programming?

- TCP (Transmission Control Protocol): TCP is a connection-oriented protocol, which means that a connection must be established between the client and server before any data can be exchanged. Once a connection is established, data can be exchanged reliably and in order.
- UDP (User Datagram Protocol): UDP is a connectionless protocol, which means that data can be exchanged without first establishing a connection. UDP is faster and more lightweight than TCP, but it is also less reliable and does not guarantee that data will arrive in order.

03. What types of sockets are available to the users?

- Stream Sockets (TCP Sockets): These sockets provide a reliable two-way, persistent, point-to-point communication between the hosts on the Internet. The Java I/O system uses sockets to connect to other programs on the local system or other systems on the Internet.
- Raw Sockets (UDP Sockets): These sockets support fast, connectionless, unreliable packet transfers. They are used when the overhead of a reliable connection is not needed.

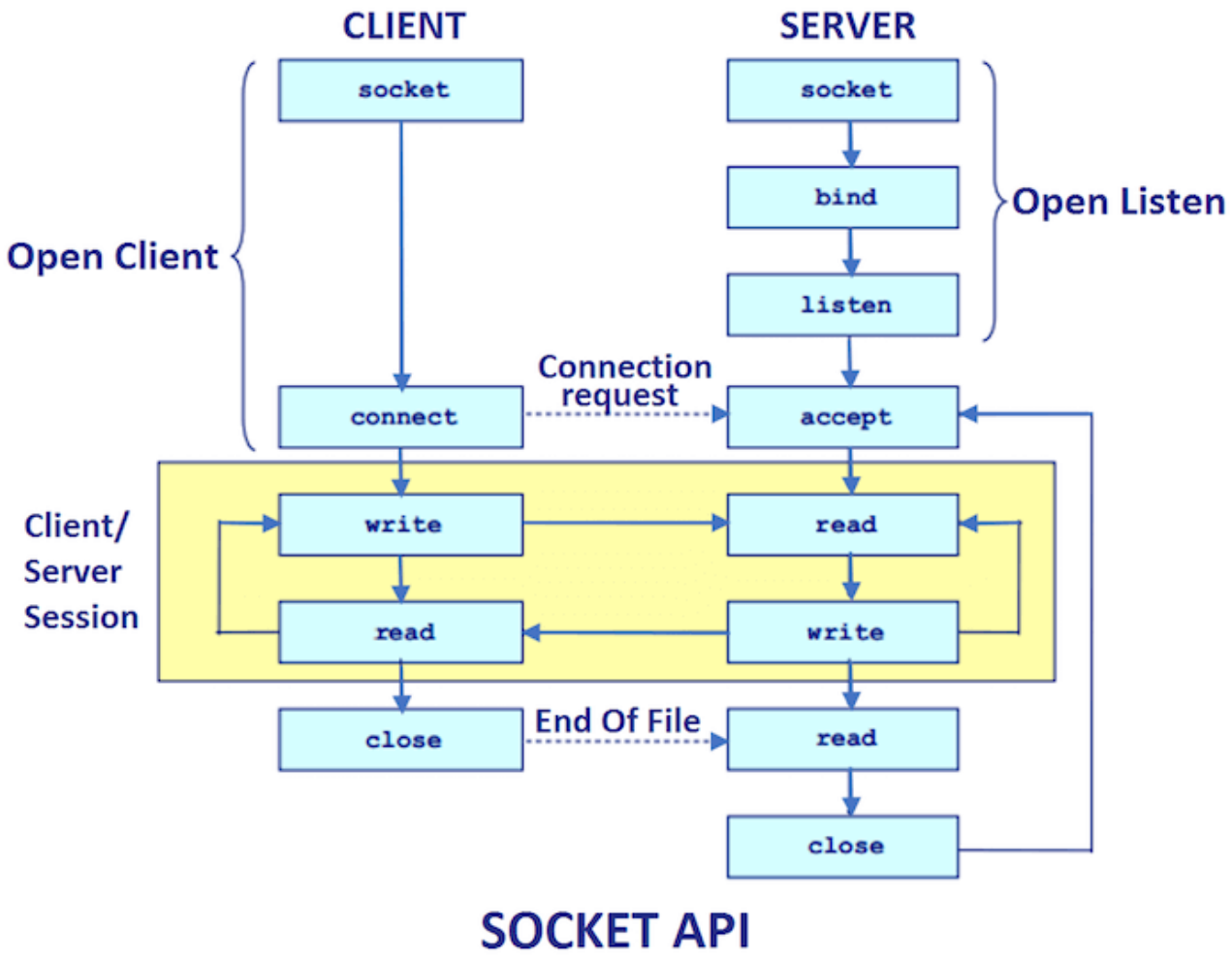
04. How the server and the client establishes the connection. Explain step by step. (draw the diagram as well)

Server Side:

- Create a ServerSocket object and bind it to a specific port number.
- Listen for incoming connection requests using the accept() method of the ServerSocket.
- When a connection request is received, the accept() method returns a Socket object representing the client's connection.
- You can then read and write data using the InputStream and OutputStream of the Socket to communicate with the client.

Client Side:

- Create a Socket object and specify the IP address and port number of the server to which you want to connect.
- Use the connect() method of the Socket to establish a connection to the server.
- Once the connection is established, you can send and receive data using the InputStream and OutputStream of the Socket.



05. Briefly explain the functions (system calls) of both client and server side in socket connection.

Server Side:

- ServerSocket(): Constructor to create a new server socket.
- bind(): Binds the server socket to a specific address and port.
- listen(): Listens for incoming connection requests.
- accept(): Accepts an incoming connection request and returns a socket for communication.
- Reading and writing data using InputStream and OutputStream of the socket.

Client Side:

- Socket(): Constructor to create a new client socket.
- connect(): Establishes a connection to a remote server.
- Reading and writing data using InputStream and OutputStream of the socket.