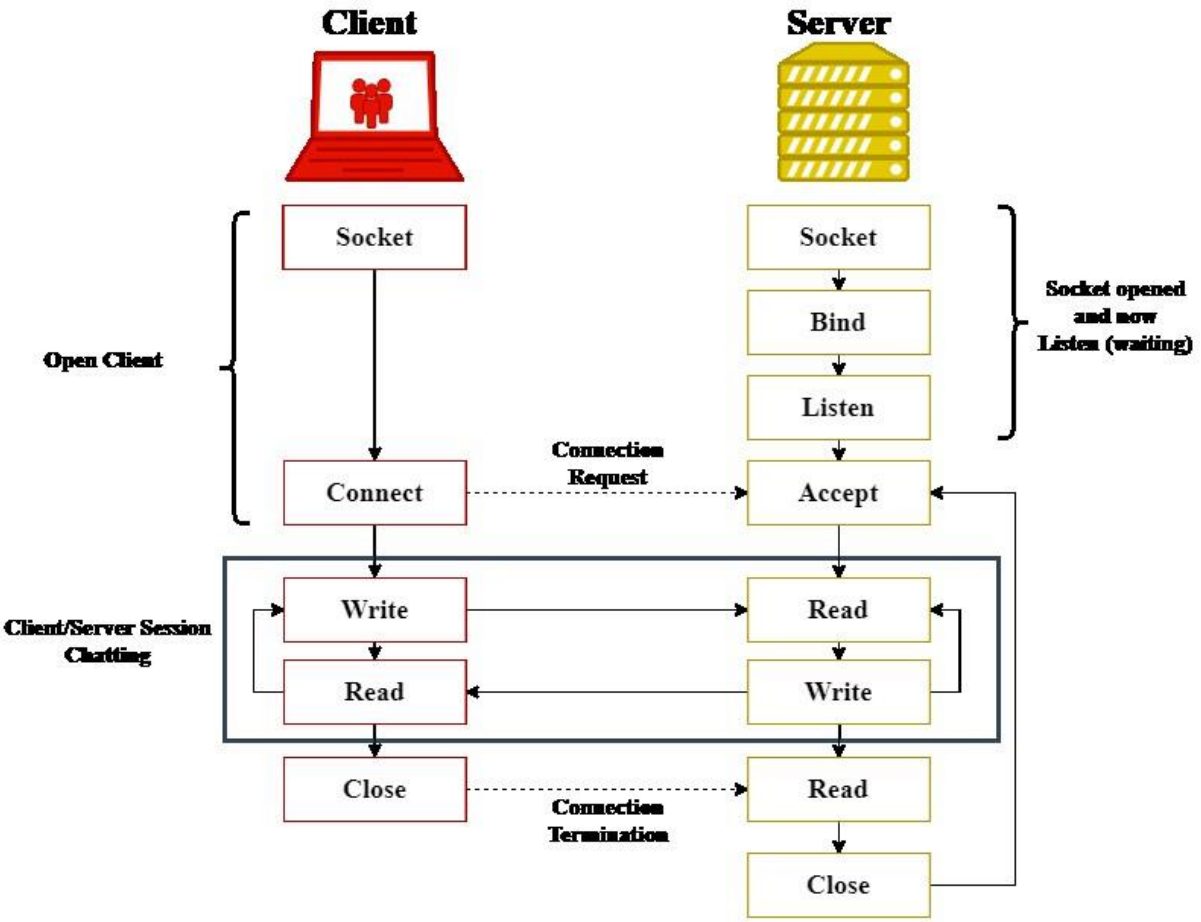
Faculty of Information Engineering
The Networks Department
Course: Communication Networks
[NETWB301]

Project Description

The World Wide Web and Internet has changed the way we live and communicate with each other and the way we conduct science, engineering, and commerce. The modern life is completely being driven around or being centered around the Internet. Businesses are continuously seeking new ways to produce and communicate with various services in a new fashion introducing innovation.

Socket programming is a way of connecting two nodes on a network to communicate with each other. Through socket (door) listens on a particular **port** at an **IP**, while other socket reaches out to the other to form a connection (TCP) or connection-less oriented (UDP). Server forms the listener socket while client reaches out to the server.



Socket parameters:

- **IP:** IP address of the server

Same Network (Milestone 1)	127.0.0.1 localhost
Different Network (Milestone 2)	Get Server's IP through command prompt and ipconfig command

- **Port number:** TCP port number must be the same in both client and server.
  It is just a number representing which application to run on a server. Use range from 1024 to 49151.

Milestone 1 "Client/Server model on the same network":

Objective:
You are required to build a **Client/Server Model.**

Tasks:

1. Copy and paste TCPClient.java and TCPServer.java to your project in Eclipse.
2. Run TCPServer.class. **(A process in the server that is idle until it is contacted by some client)**
3. Run TCPClient.class. **(A process in the client and establishes a TCP connection between the client and server processes)**
4. Make sure communication is established between client and server and that the sentences sent by sender is converted to upper case.

IP **Write in TCPClient.class**	127.0.0.1 localhost
TCP Port **Write in both TCPClient.class and TCPServer.class.**	Use any from this range 1024 - 49151 MUST be the same both code sides

Milestone 2 "Client/Server model on the different network":

Objective:
You are required to build a **Chatting Application**.

Tasks:
The chatting application should run as follows:

1. The client opens a socket connection with the server by typing the word 'CONNECT'. The socket should not be opened otherwise
2. After a connection is established. At both the client and the server sides, any text entered from the keyboard at the client/server side should appear as System.out.println at the server/client side (a bi-directional chatting application)
3. The client and the server should not be on the same laptop or machine. You can use a WLAN or a hotspot to interconnect the client and the server.

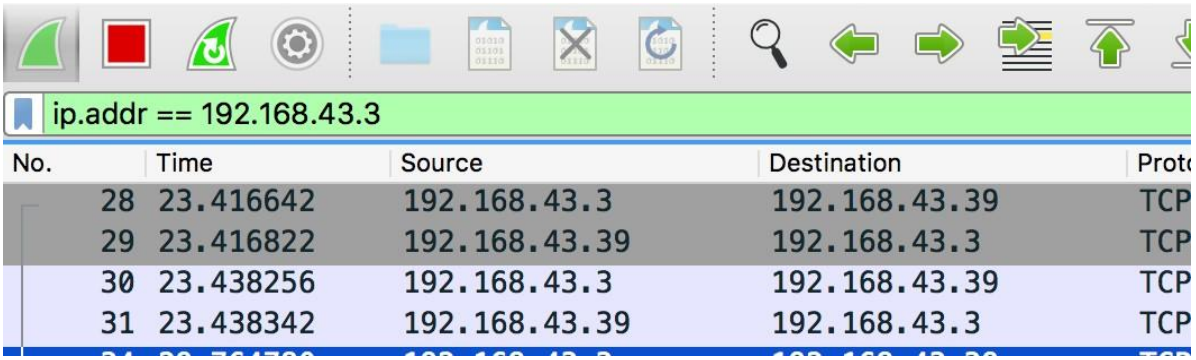
IP **Write in TCPClient.class**	Use ipconfig on the command prompt of Server's laptop to get its IPv4 address and write it in TCPClient.class.
TCP Port **Write in both TCPClient.class and TCPServer.class.**	Use any from this range 1024 - 49151 MUST be the same both code sides

Milestone 3 “Tracing TCP Packets”:

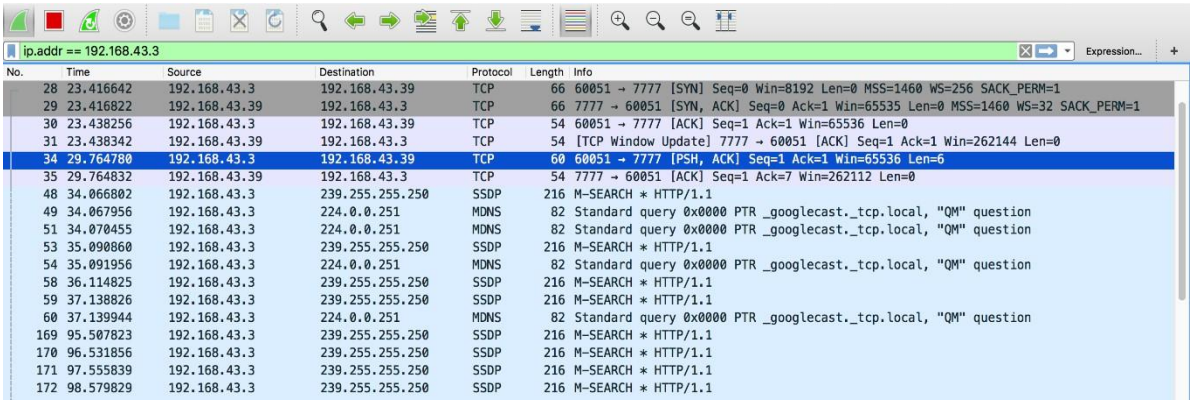
Objective:
Tracing TCP packets between client and server on **Wireshark** to identify the attributes and data values in each packet.

Tasks:

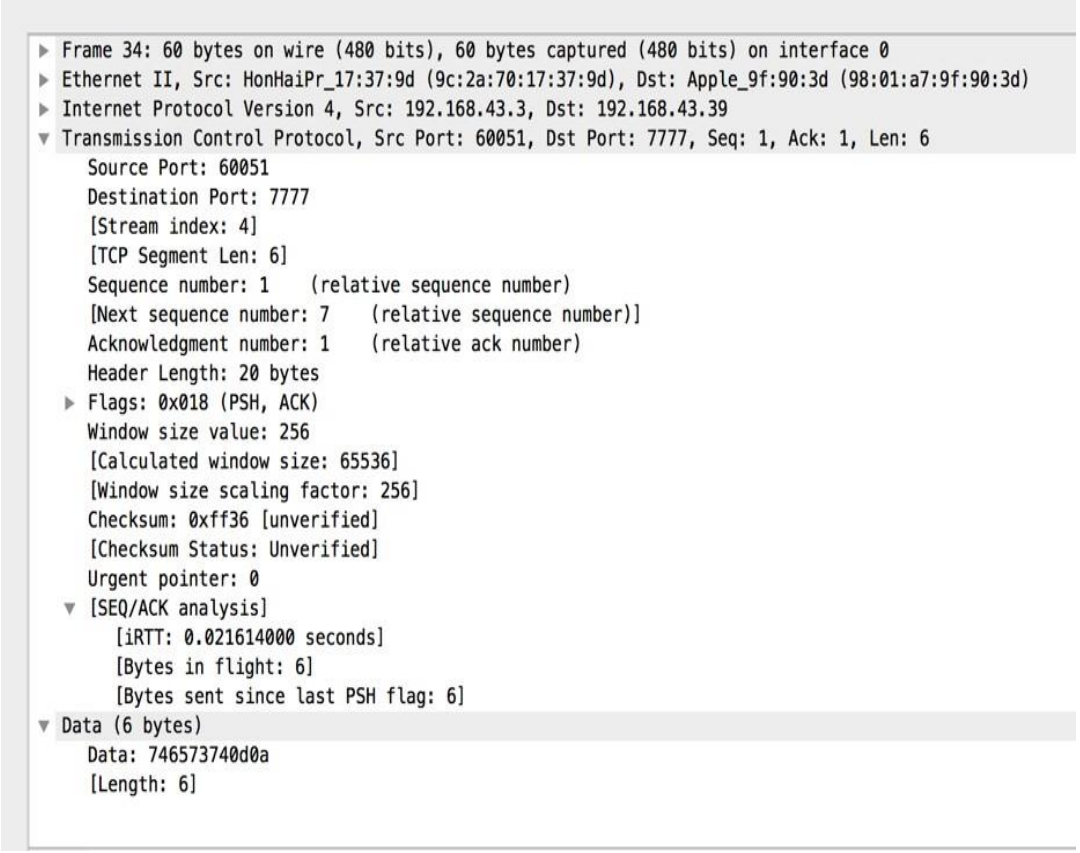
1. Download and install Wireshark on your computer from this link: https://www.wireshark.org
2. Run your project code on both devices.
3. Choose WiFi/Hotspot interface in Wireshark and press start to start the sniffing process.
4. Filter packets shown to your chat packets only by adding **ip.addr == <ip address of your server>** (you got from command prompt) in the filter text field and press apply.



5. Return to your code on Eclipse and start chatting between the two devices.
6. Observe the packets sent by looking at the columns (Protocol, Source IP address, Destination IP address).



7. Start tracing the packets you sent on Wireshark. **(Hint: You can use online Hex to ASCII converter to convert the data bits in the message into a readable form)**



Useful Links: https://www.javatpoint.com/socket-programming
https://www.tutorialspoint.com/java/java_networking.htm

Deadline:
Maximum by Sunday 24/05/2020 11:59 PM.

Deliverables:
Submit a clear video for your up and running code as a demo. Prepare slides explaining the steps you did and showing that your code is running, add voice along with your live image explaining all steps and demo. Video duration should be 5-10 minutes.
Make sure to include the following:

- Screenshot for the WHOLE window for Eclipse and console part.
- Screenshots for the WHOLE window from 2 devices client and server console part.
- Screenshots for Wireshark captured packets (step 5 in Milestone 3) and at least one traced packet (step 7 in Milestone 3).

Submission:
Submit a zipped file of your video along with your code files through the online submission system from the intranet.

Best of luck.