Faculty of Informatics & Computer Science
[CNET101] Computer Networks - SS24
Dr. Marwa Zamzam
Eng. David Abdelmalak
Eng. Mariham Wasfy
Eng. Peter Wafik

Project

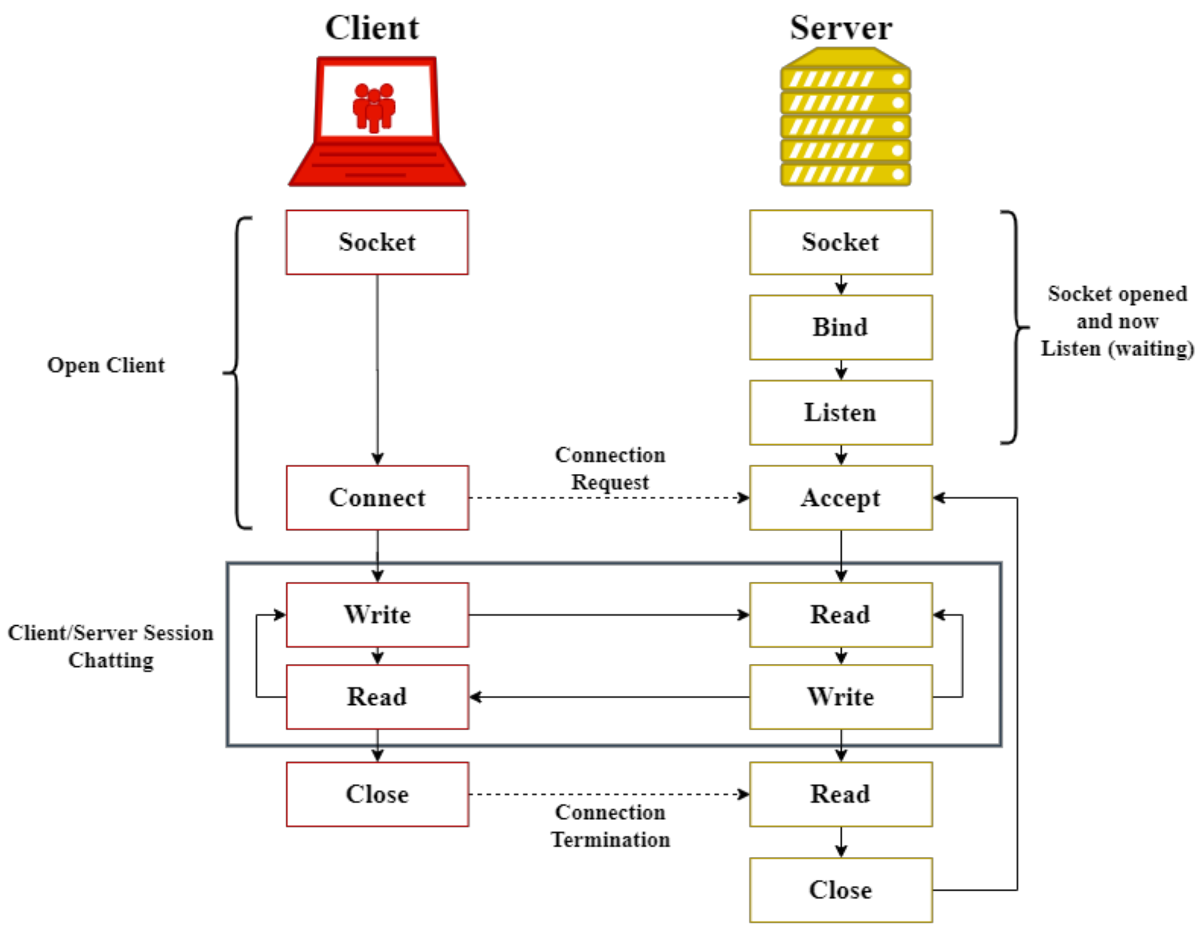
Deadline Saturday, 25th of May 2024 11:59 PM

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 ranges 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.

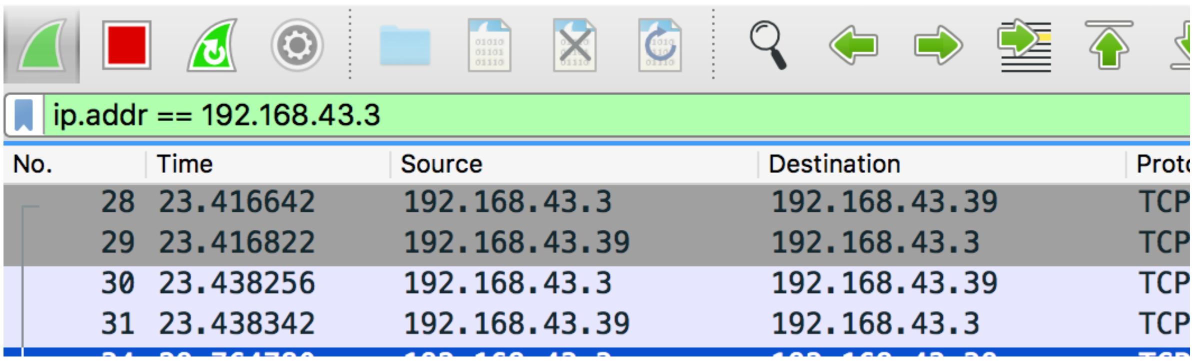
IP **Write in TCPClient.class**	Use ipconfig on the command prompt of Server's laptop to get its IPv4 address and write it in TCPClient.class.
TCP Port **Write in both TCPClient.class and TCPServer.class.**	Use any from this range 1024 - 49151 MUST be the same both code sides

→ **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:

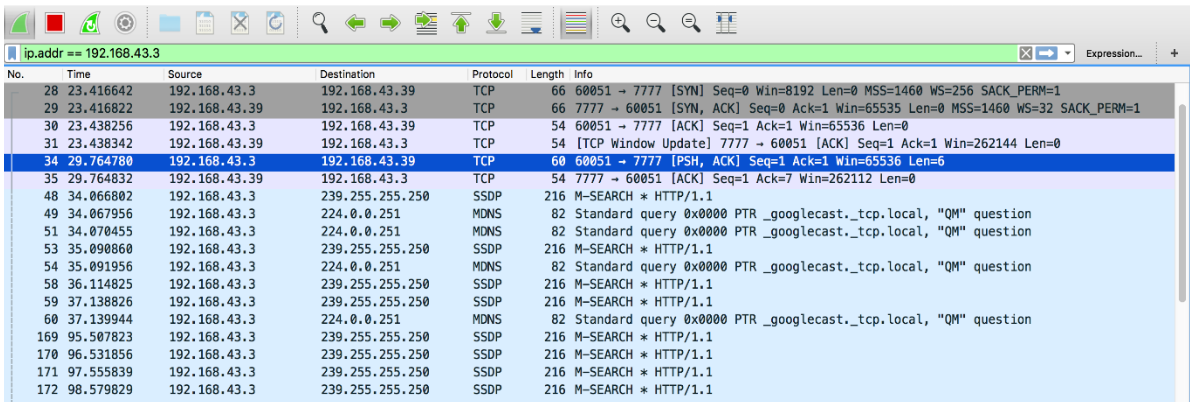
1. Download and install Wireshark on your computer from this link: https://www.wireshark.org
2. Run your project code on both devices.
3. Choose WiFi/Hotspot interface in Wireshark and press start to start the sniffing process.
4. Filter packets shown to your chat packets only by adding **ip.addr ==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



ip.addr == 192.168.43.3

No.	Time	Source	Destination	Prot
28	23.416642	192.168.43.3	192.168.43.39	TCP
29	23.416822	192.168.43.39	192.168.43.3	TCP
30	23.438256	192.168.43.3	192.168.43.39	TCP
31	23.438342	192.168.43.39	192.168.43.3	TCP

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).



ip.addr == 192.168.43.3

No.	Time	Source	Destination	Protocol	Length	Info
28	23.416642	192.168.43.3	192.168.43.39	TCP	66	60051 → 7777 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
29	23.416822	192.168.43.39	192.168.43.3	TCP	66	7777 → 60051 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=32 SACK_PERM=1
30	23.438256	192.168.43.3	192.168.43.39	TCP	54	60051 → 7777 [ACK] Seq=1 Ack=1 Win=65536 Len=0
31	23.438342	192.168.43.39	192.168.43.3	TCP	54	[TCP Window Update] 7777 → 60051 [ACK] Seq=1 Ack=1 Win=262144 Len=0
34	29.764780	192.168.43.3	192.168.43.39	TCP	60	60051 → 7777 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=6
35	29.764832	192.168.43.39	192.168.43.3	TCP	54	7777 → 60051 [ACK] Seq=1 Ack=7 Win=262112 Len=0
48	34.066802	192.168.43.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
49	34.067956	192.168.43.3	224.0.0.251	MDNS	82	Standard query 0x0000 PTR _googlecast._tcp.local, "QM" question
51	34.070455	192.168.43.3	224.0.0.251	MDNS	82	Standard query 0x0000 PTR _googlecast._tcp.local, "QM" question
53	35.090860	192.168.43.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
54	35.091956	192.168.43.3	224.0.0.251	MDNS	82	Standard query 0x0000 PTR _googlecast._tcp.local, "QM" question
58	36.114825	192.168.43.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
59	37.138826	192.168.43.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
60	37.139944	192.168.43.3	224.0.0.251	MDNS	82	Standard query 0x0000 PTR _googlecast._tcp.local, "QM" question
169	95.507823	192.168.43.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
170	96.531856	192.168.43.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
171	97.555839	192.168.43.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
172	98.579829	192.168.43.3	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1