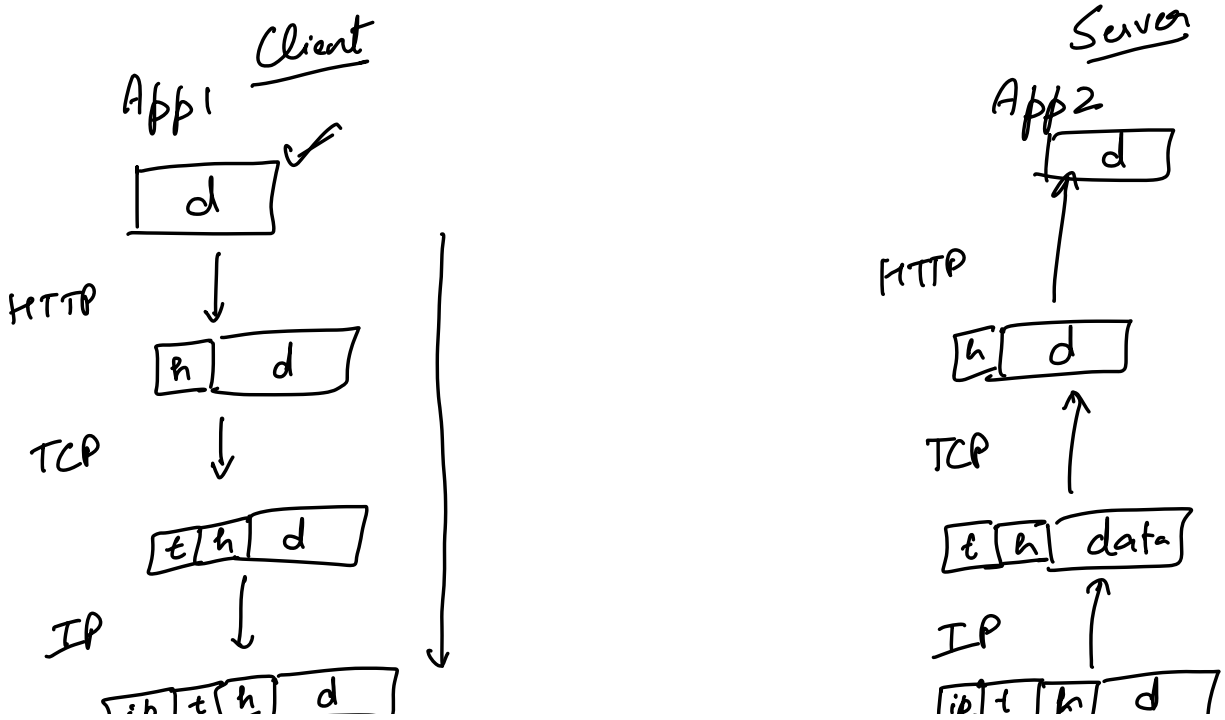
1. Good Evening
2. Lecture Begins at 9:08 pm
3. Topic : TCP & UDP + Socket Program.

Agenda

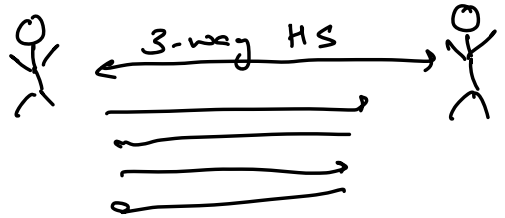
1. TCP and UDP
2. How TCP works?
3. Socket Programming

TCP & UDP



- ☆ Data increases down the layer & decreases up the layers
- ☆ Headers are information added by protocols to fulfill their responsibilities
- ☆ More responsibilities imply a bigger header size

TCP	UDP
☆ Transmission Control Protocol	☆ User Datagram Protocol
☆ Reliable protocol	☆ Best Effort
☆ Error checking & recovery	☆ Error checking
☆ Connection Oriented	☆ Connectionless



☆ Larger header size 20 – 60 bytes	☆ Smaller header size 8 bytes
☆ Slower	☆ Faster
e.g. HTTP/2, Reliable	e.g. VoIP, Games

TCP Header

- Destination Port
- Source Port
- Checksum
- Seq No
- Ack no.
- Window size.

UDP Header

- Dest port 2
- Source port 2
- Checksum 2
- Length 2

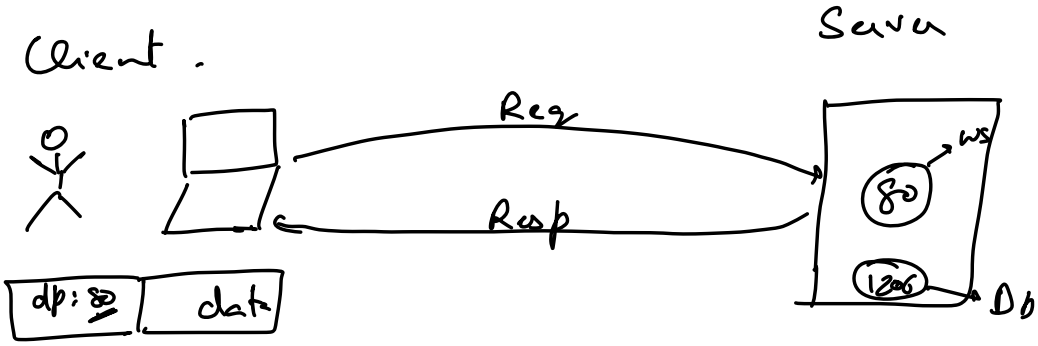
OSI Model

1. App → JO
2. Pre → Comp/Encr
3. Session → User

TCP/IP Model

Application (Http)

4. Transport → End to End → Transport (TCP/~~UDP~~)
5. Network → Routing → Network (IP)
6. Data Link → Hop to Hop → Data Link
7. Physical →

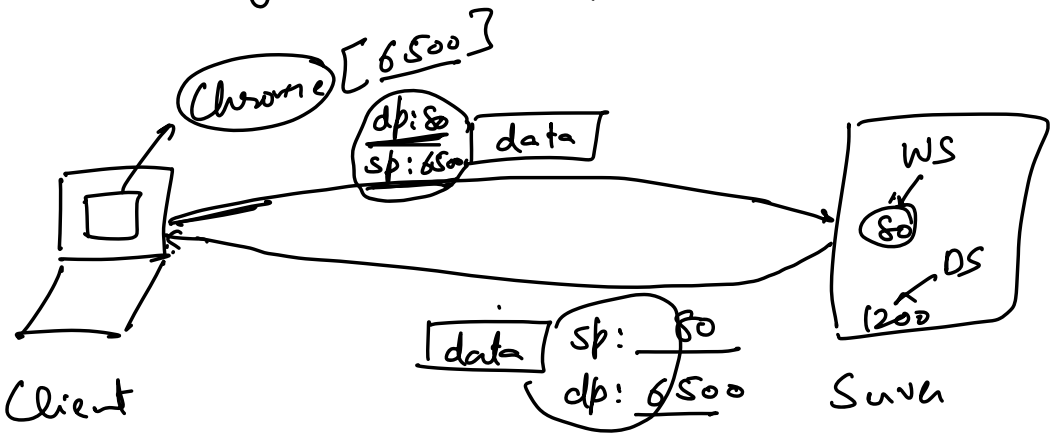


TCP & UDP

↳ Destination Port ?

↳ Because data needs to be sent to an app on the server

→ Why is source port reqd. ?



Ephemeral Port

Chrome opened 50 tabs

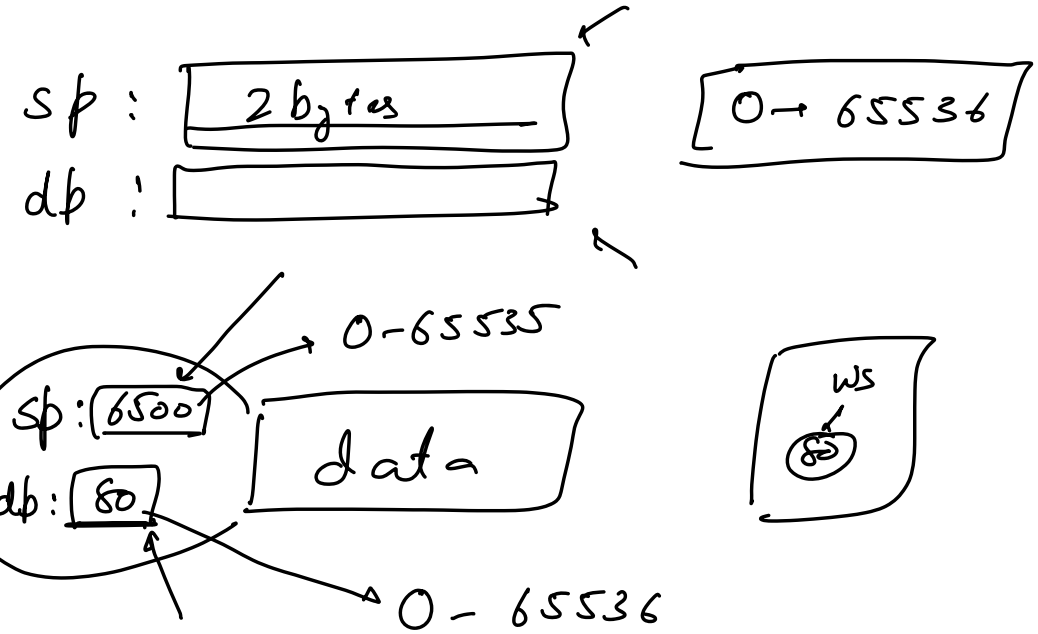
↙

Request OS for a port to do communication

OS assigns them a port temporarily [Ephemeral Port]

Dest Port, Sour Port] 2 bytes = 2^{16} different port addresses

0 → 65535 65536

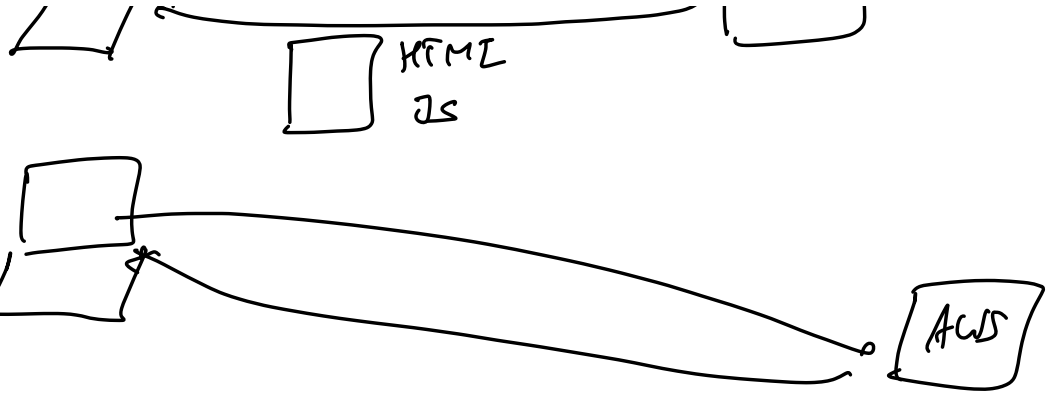


Command to check

MSYS

Cygwin

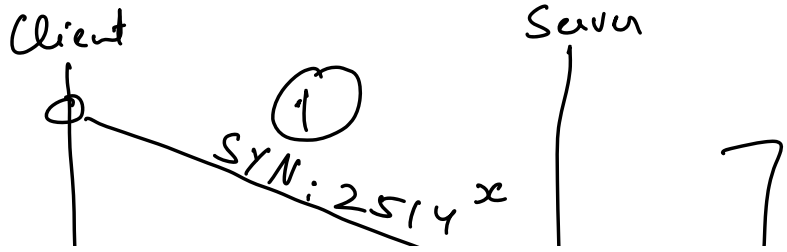
↳ lsof -i -a -P

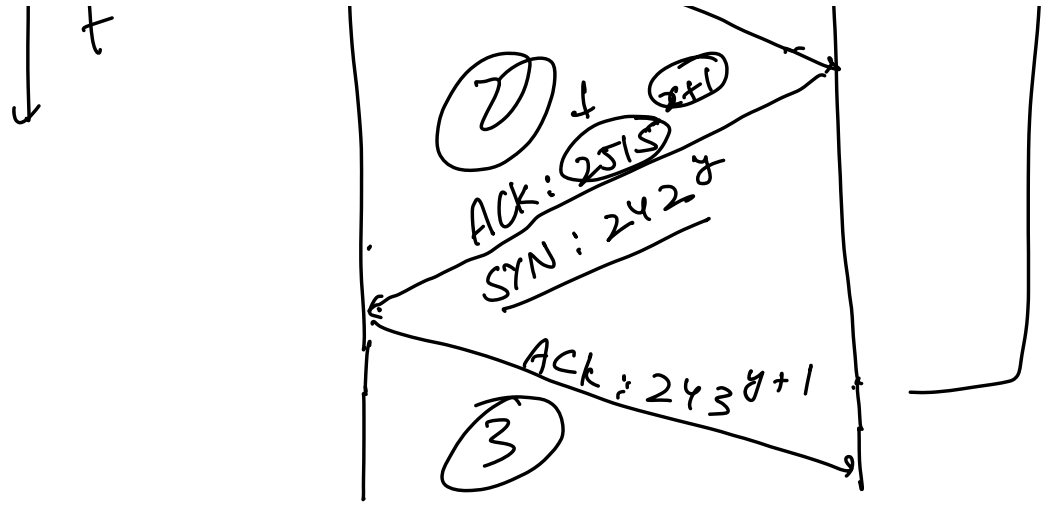


How TCP Works? [Seq, Ack, WS]

- Connection Establishment
  - 3 way Handshake
- Data Transfer
  - SYN + ACK
  - Errors
    - Retry
    - Hashmap
  - window size

TCP Connection Establishment [3 way HS]

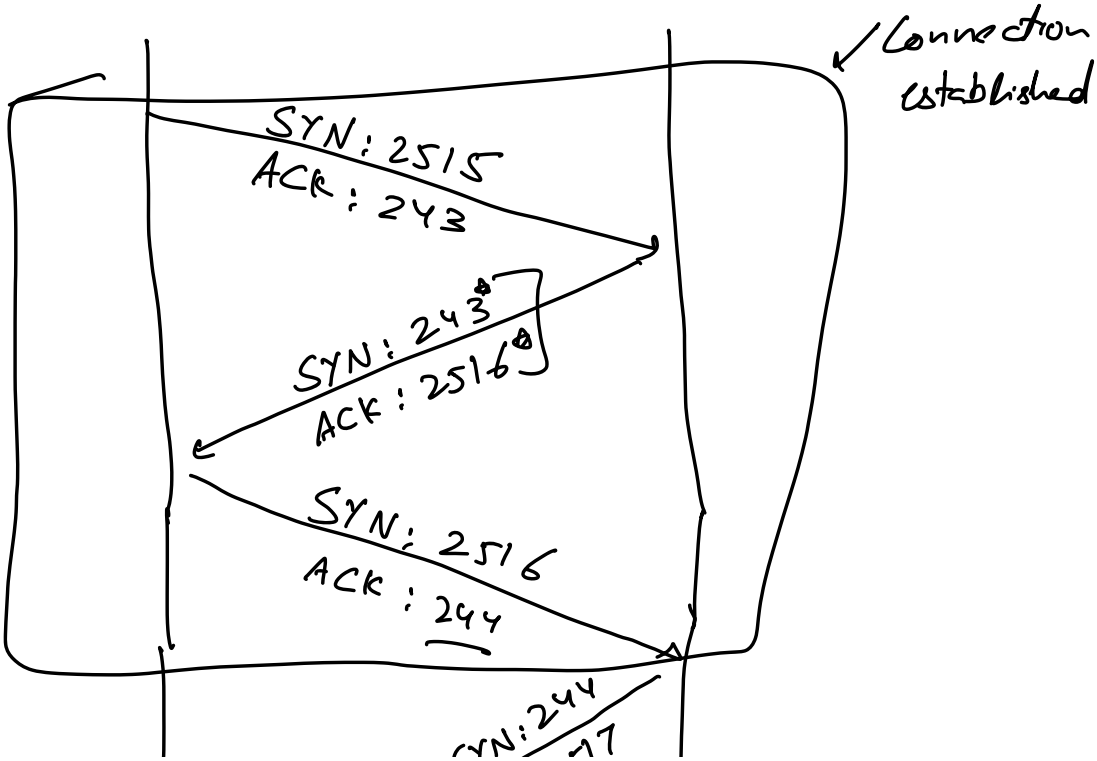


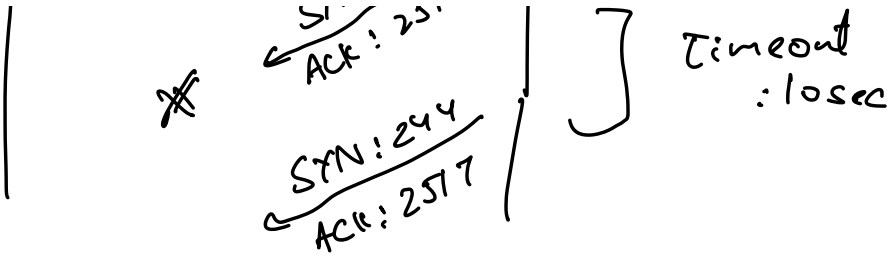


1. SYN
2. SYN + ACK
3. ACK

3 way HS to establish connection

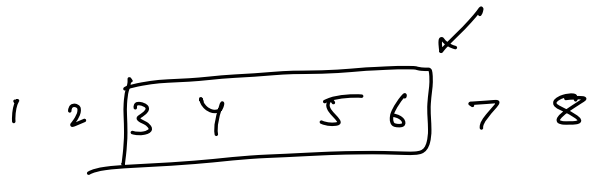
Data Transfer

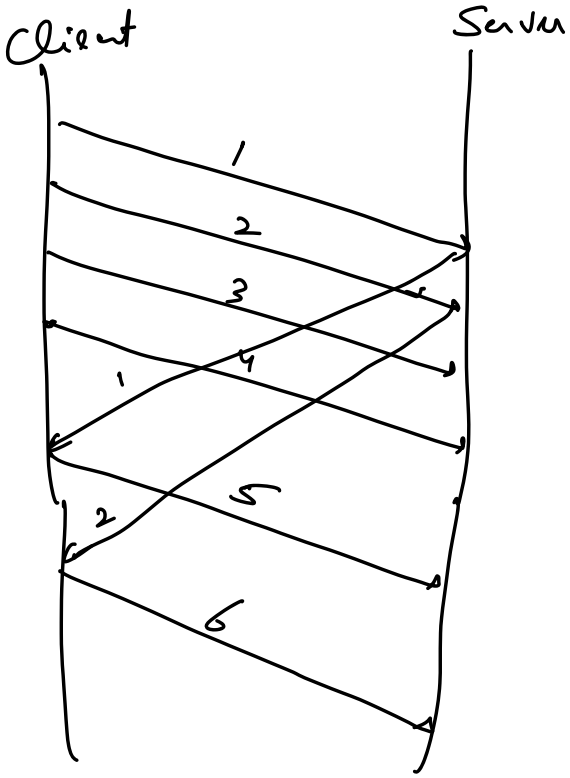




Window Size : 4

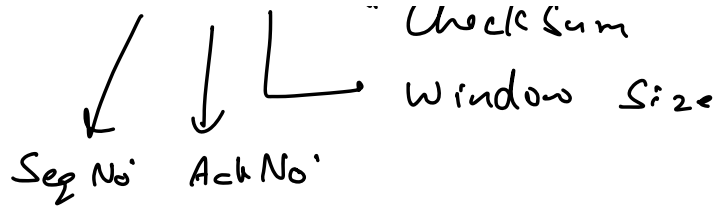
8 packets

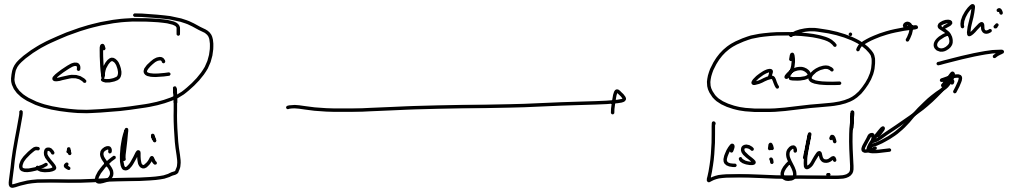


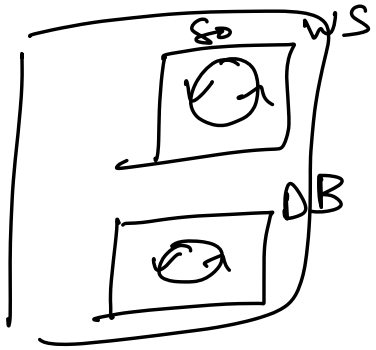


TCP

- UDP
- Header
  - Dest Port
  - Source Port







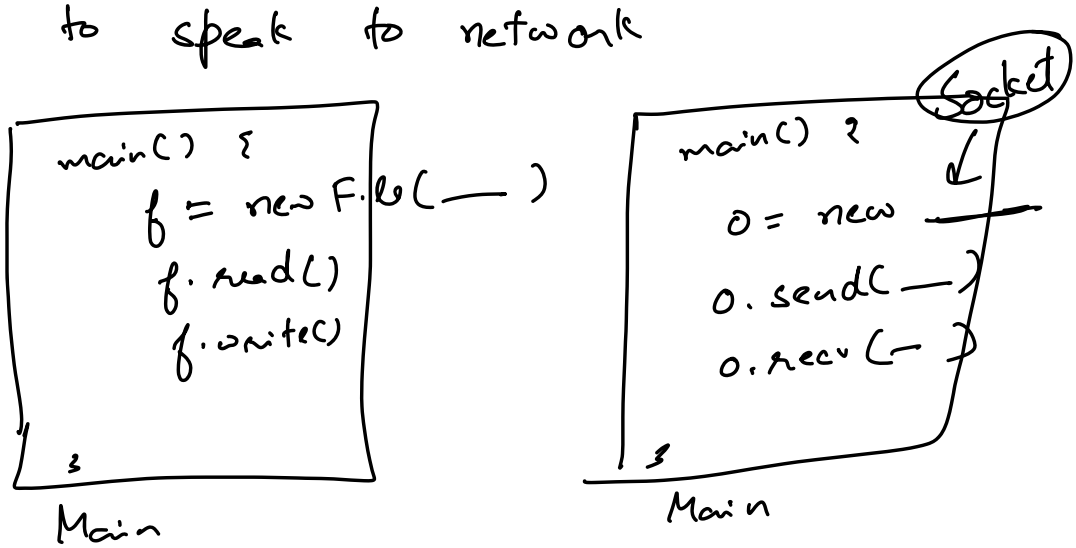
Break : (10:20 - 10:28)

Socket Programming

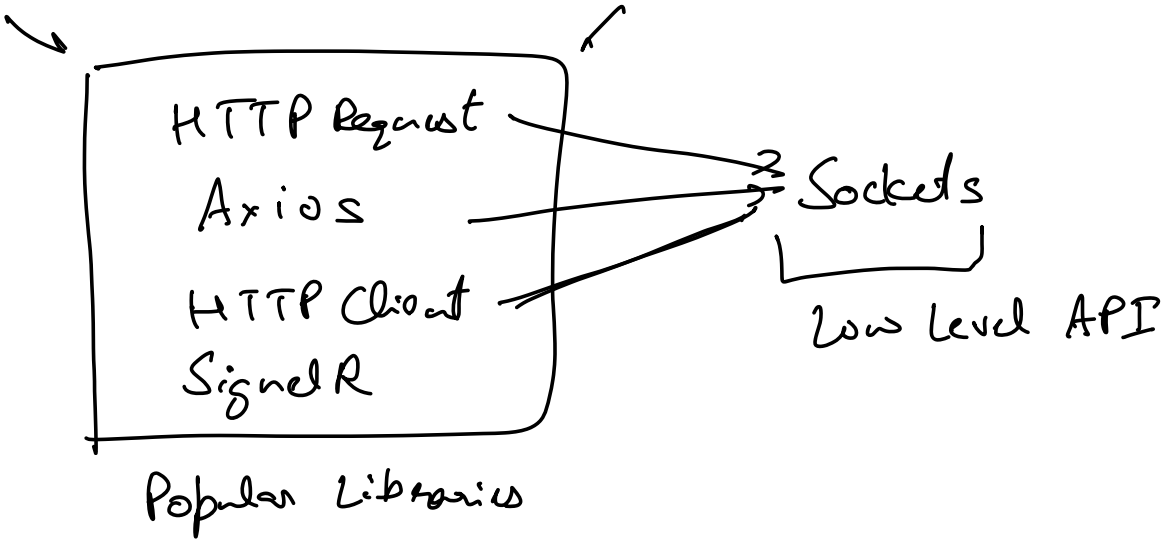
1. Theoretical Concepts
2. Basic Client - Server App
3. Multi Threaded Server

5.

→ A Socket is a programming construct reqd. to speak to network



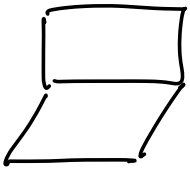
Socket ≠ Web Sockets



Socket is object of a class

15 minutes

Client - Server App



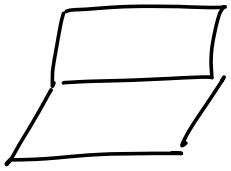
Client

1. Create Socket
2. Connect to server

Request OS for ephemeral port

3. Send data ★
4. Receive data

5. Close.



Server

1. Create a socket
2. Bind the socket to a port
3. Start Listening to the port
4. Accept connections from clients

5. Receive ★
6. Send

using child socket

★ A child socket per client

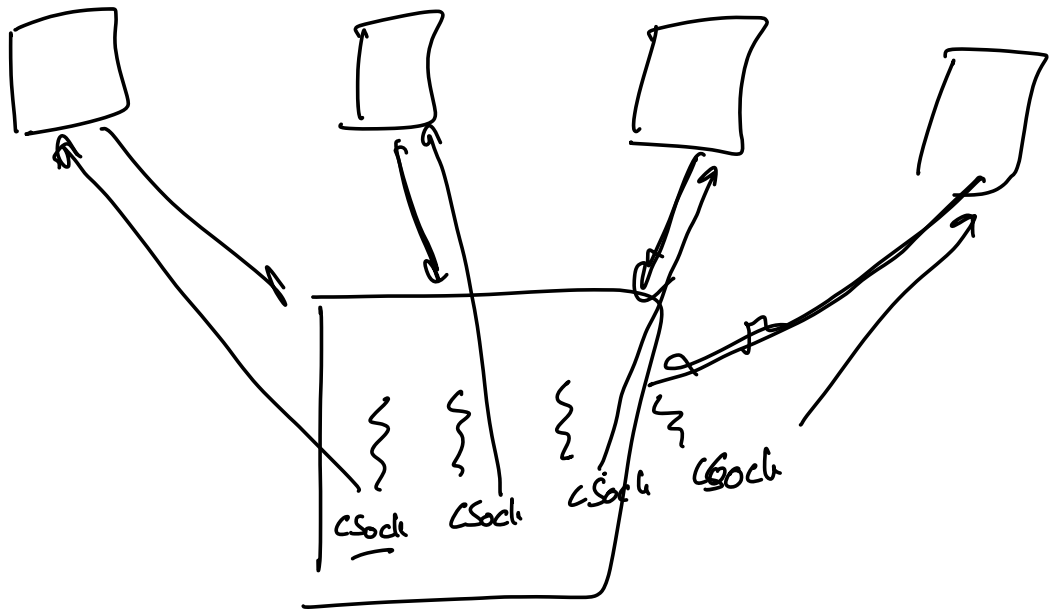
7. Close

Client - Server App

Multi-threaded Server

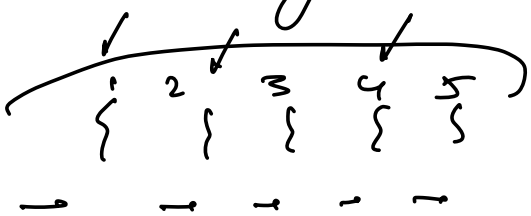
└ [Able to listen to many clients]

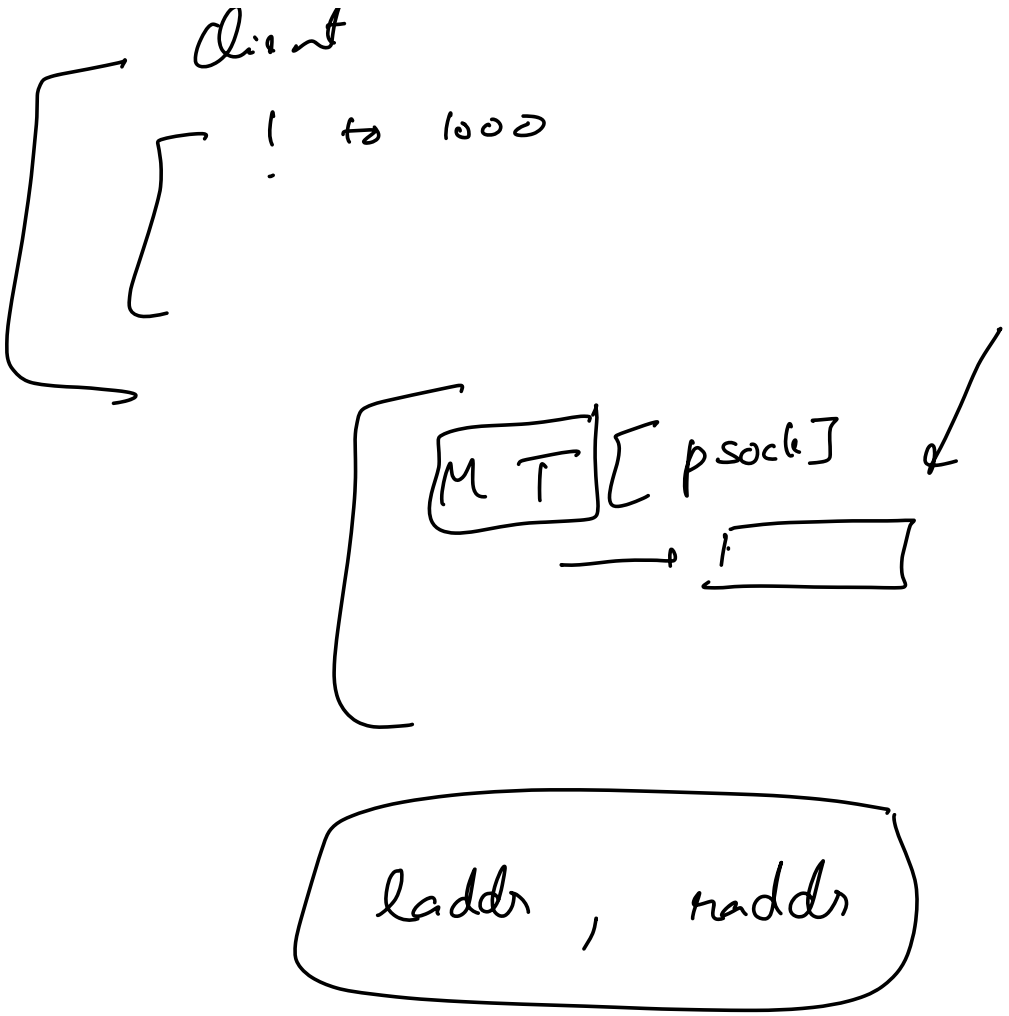
☆ Multi-threading



Server

HTTP ↗ Application Layer




client
1 to 1000
MT
psock
laddr , raddr