# Fundamentals of Client-Server Communication using TCP protocol.

Prof. Anand D Dave

Department of Information Technology,

Dharmsinh Desai University, Nadiad.

# Distributed Computing: Client- Server Communication Using TCP protocol

- Since the TCP protocol is connection oriented protocol. Connection between Client – Server must be established before the communication takes place.
- This connection is established with "Three way Handshake" sequence.
- In case of Client –Server communication before client sends request to connect to server, the server must be in the listening mode.

# Distributed Computing: Client- Server Three-way Handshake

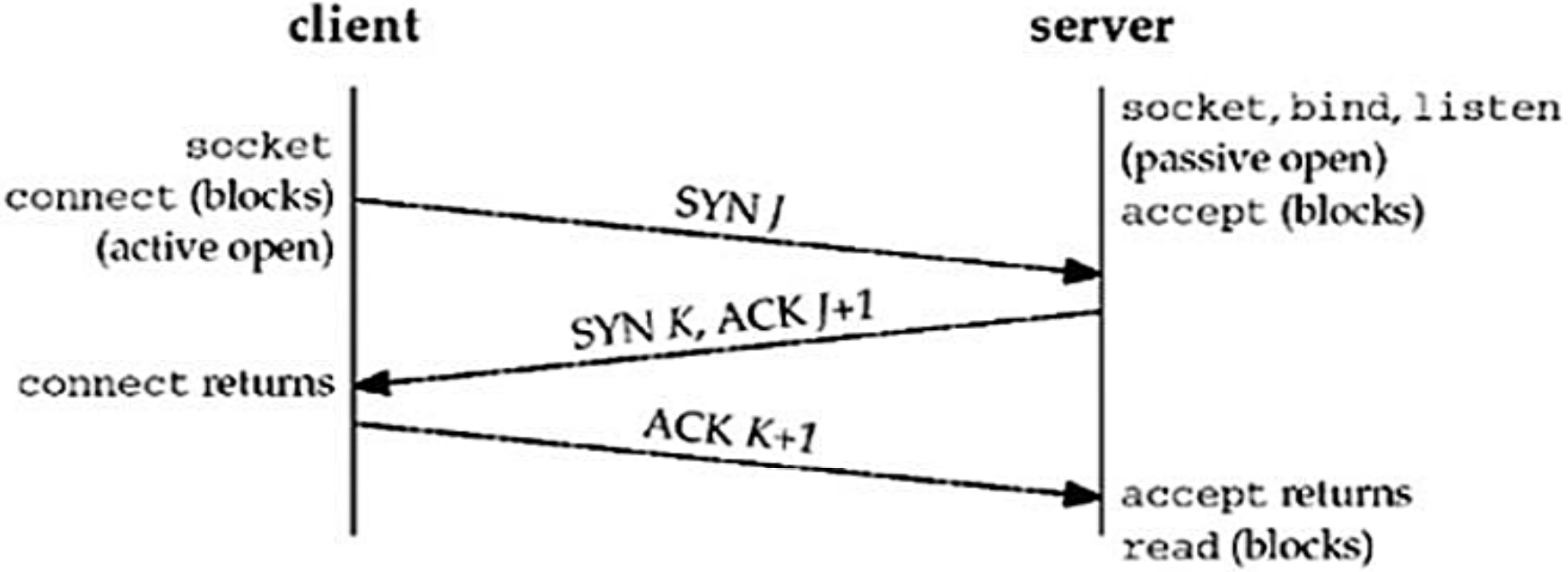
- The Server must be ready to accept the connection.
- This is normally done by the calling socket, bind, and listen functions, and it is called a passive open.

- The client issues an active open by calling connect call.
- This will cause the client process to initiate **Three way Handshake** sequence.
- In this Client's TCP will send a "synchronize"(SYN) segment which tells server the client's initial sequence number, there is no data sent with the SYN: It just contains an IP header, TCP header and possible TCP options.

- The server TCP must acknowledge(ACK) the client's (TCP) SYN and the server TCP must send its own SYN containing the initial sequence number for the data that the server (TCP) will send on the connection. The server (TCP) sends its SYN and the ACK of client's SYN in a single segment.
- The client (TCP) must acknowledge the server's SYN.

# Distributed Computing: Client- Server Three-way Handshake

- TCP Connection establishment sequence :



# Distributed Computing: Client- Server Three-way Handshake

- TCP Options:

- Each SYN can contain TCP options
- Commonly used are
- MSS(Maximum Segment Size) option:
- Using this option, the TCP that is sending the SYN announces its MSS(the maximum amount of data that it is willing to accept in each TCP segment on this connection.)

# Distributed Computing: Client- Server Three-way Handshake

- TCP Options:
- Window Scale option:
- The Maximum window that either TCP can advertize to the other TCP is 65535.

- The newer option specifies that the advertized window in the TCP header must be scaled by 0-14 bits, i.e. 2^14.
- The both end systems must support this option for its usage.

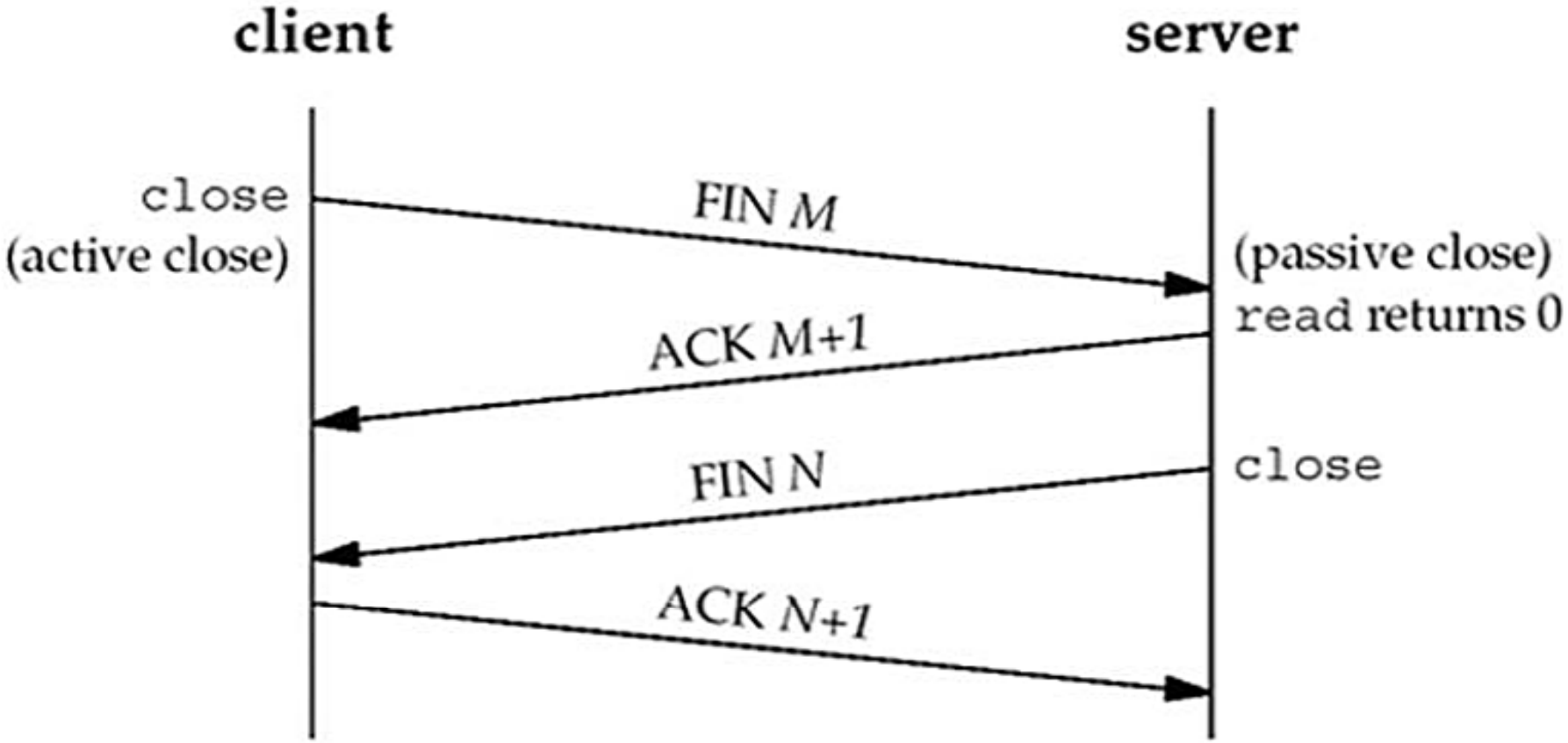
# Distributed Computing: Client- Server Three-way Handshake

- TCP Options:
- Timestamp option:
- This option is needed for high-speed connection
- as reusing of sequence numbers would be fast in high speed connection, if lost packet reappear, it can cause data corruption.
- If two packets appear with same sequence number, they can be differentiate based on the timestamp value.

# Distributed Computing: Client- Server Connection Termination

- TCP Connection Termination :
- One application calls close first.
- This end performs the active close.
- This end's TCP sends a FIN segment, which means it is finish sending data.(However, it can receive data from other end).
- The other end that receives the FIN performs the passive close.
- The received FIN is acknowledged by TCP.
- The receipt of the FIN is also passed to the application as an end-of-file( after any data that may have already been queued for the application to receive), since the receipt of the FIN means application will not receive any additional data on the connection.
- Sometime later, the application that received the end-of-file will close its socket. This cause TCP of other end to send a FIN.
- The TCP on the system that receive this final FIN segment (the end that did the active close) acknowledges the FIN segment.

# Distributed Computing: Client- Server Connection Termination



# Distributed Computing: Client- Server TCP Connection Termination

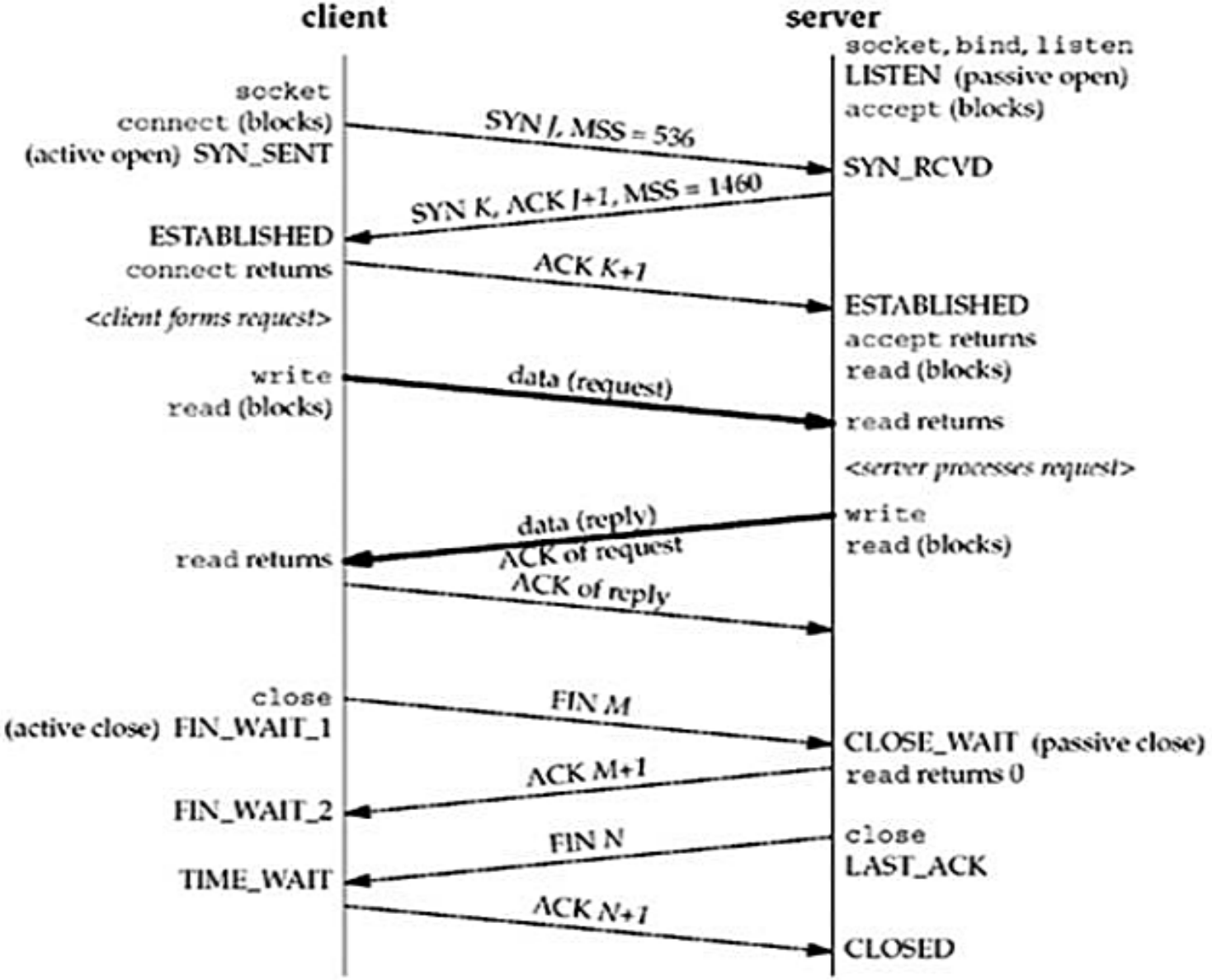
- TCP Connection termination sequence:

- Between step 2 and 3, there can be data flow. End performing active close can receive data.(Called Half-close).
- When process terminates, all the open descriptor are closed. As a result, this process gets initiated.
- often client performs the active close, but in protocols (e.g. HTTP), the server performs the active close.

# Distributed Computing: Client- Server TCP Communication

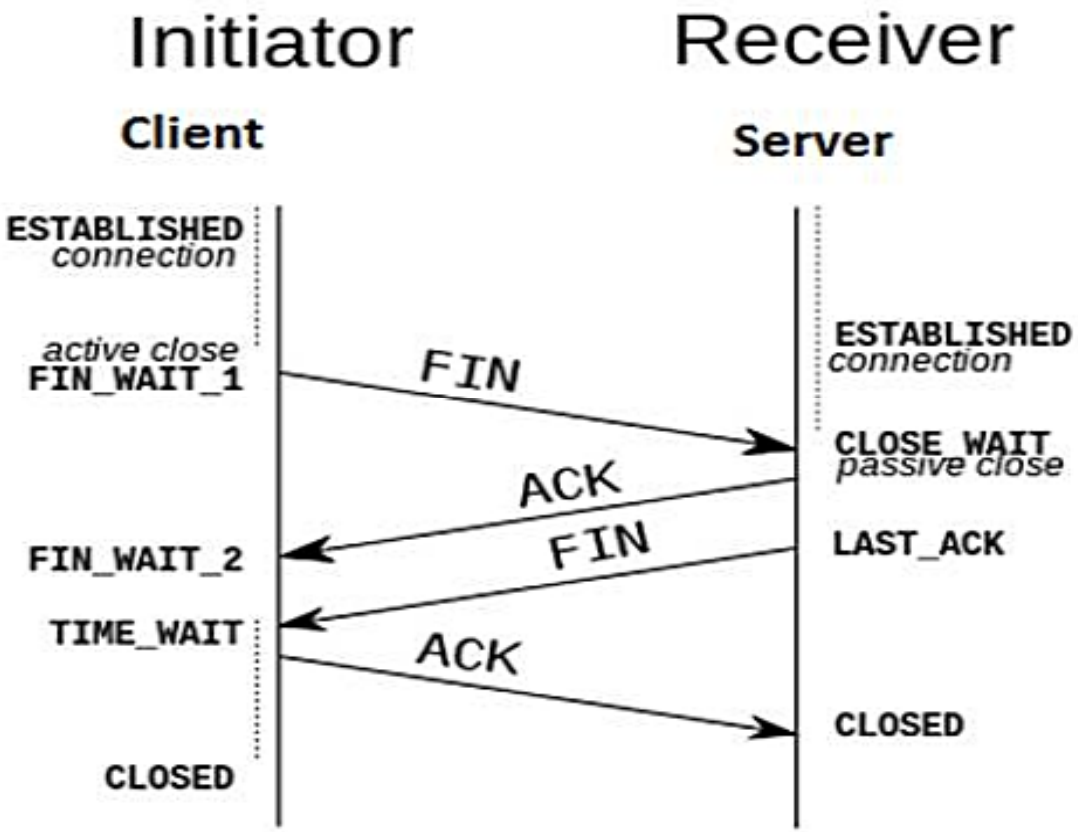
- If the acknowledgement of the client's request is sent with the server's reply(in a single segment), it is called piggybacking.
- It normally happens when the server takes less than 200ms to generate a response to the client's request.

# Distributed Computing: Client- Server TCP: State Transition Diagram



# Distributed Computing: Client- Server TCP Connection Termination

- TCP State Transition on closing the connection
- The end that performs active close enters in the Time_Wait state.



# Distributed Computing: Client- Server TCP Connection Termination

**TIME_WAIT State**

- The end that performs active close enters in the TIME_WAIT state.
- MSL(Maximum Segment Lifetime): It is the maximum amount of time that any given IP datagram can live in an internet.
- The duration of the TIME_WAIT is twice the MSL, sometimes called 2MSL.
- The recommended value of MSL is 2 minutes. (as it every implementation of TCP have to choose the value of MLS).
- The duration of TIME_WAIT state is between 1 to 4 minutes.
- TTL(Time To Live): There is a 8-bit hop limit for IP packet. It is assumed that a packet with the maximum hop limit of 255 cannot exist in an internet from more than MSL seconds.

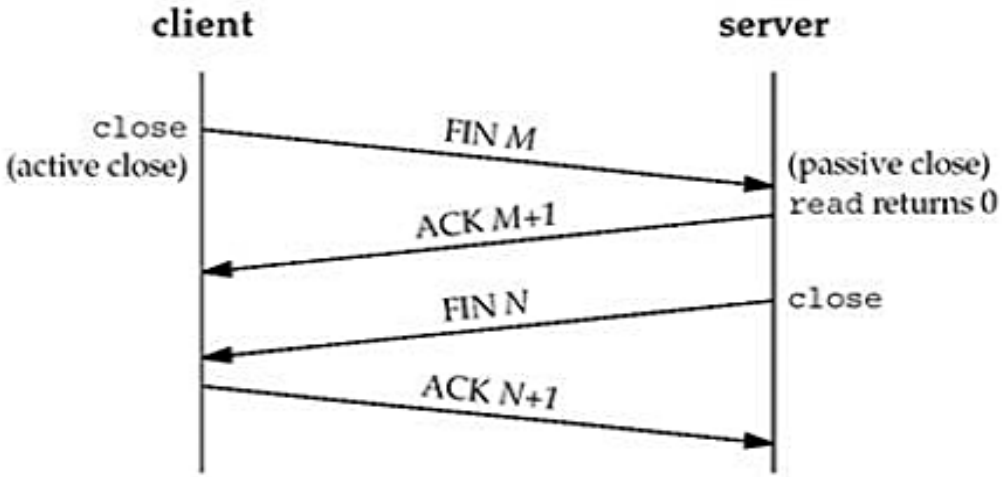
# Distributed Computing: Client- Server
# TCP Connection Termination

- There are Two reasons for the TIME_WAIT state:

1. To implement TCP's full-duplex connection termination reliably.
2. To allow old duplicate segment to expire in the network.

# Distributed Computing: Client- Server TCP Connection Termination

1. To implement TCP's full-duplex connection termination reliably.




- If final ACK is lost, the server will resend its final FIN. Thus, the client must maintain state information, so that it can resend the final ACK.
- And this is the reason why the end performs active close is the end that remain in the TIME_WAIT State.

# Distributed Computing: Client- Server TCP Connection Termination

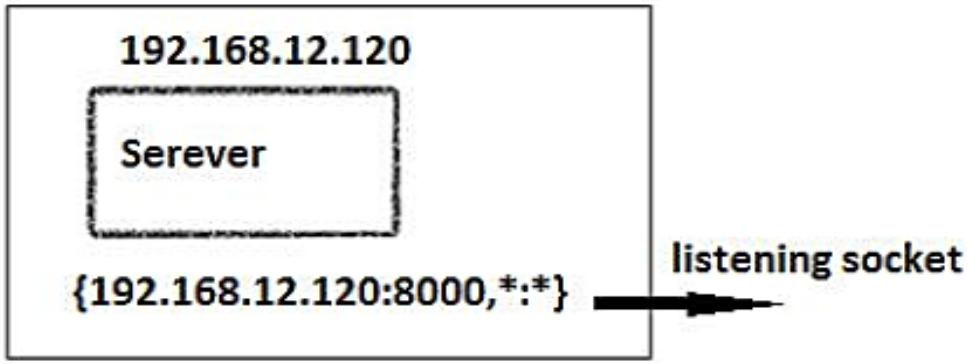
- 2. To allow old duplicate segments to expire in the network

- Assume we have a TCP connection between
- IP: 206.62.226.33 and IP: 192.69.10.2
- Port : 1500 and Port: 21

- If this connection is closed, and sometime later, the client establish a new connection with same port number(i.e. 1500), what about the old duplicate packets from earlier connection reappearing to a new connection?
- TCP will not initiate a new incarnation of a connection that is currently in the TIME_WAIT State.
- 2MSL duration of TIME_WAIT allows MSL second for a packet in one direction to be lost, and another MSL for reply to be lost.

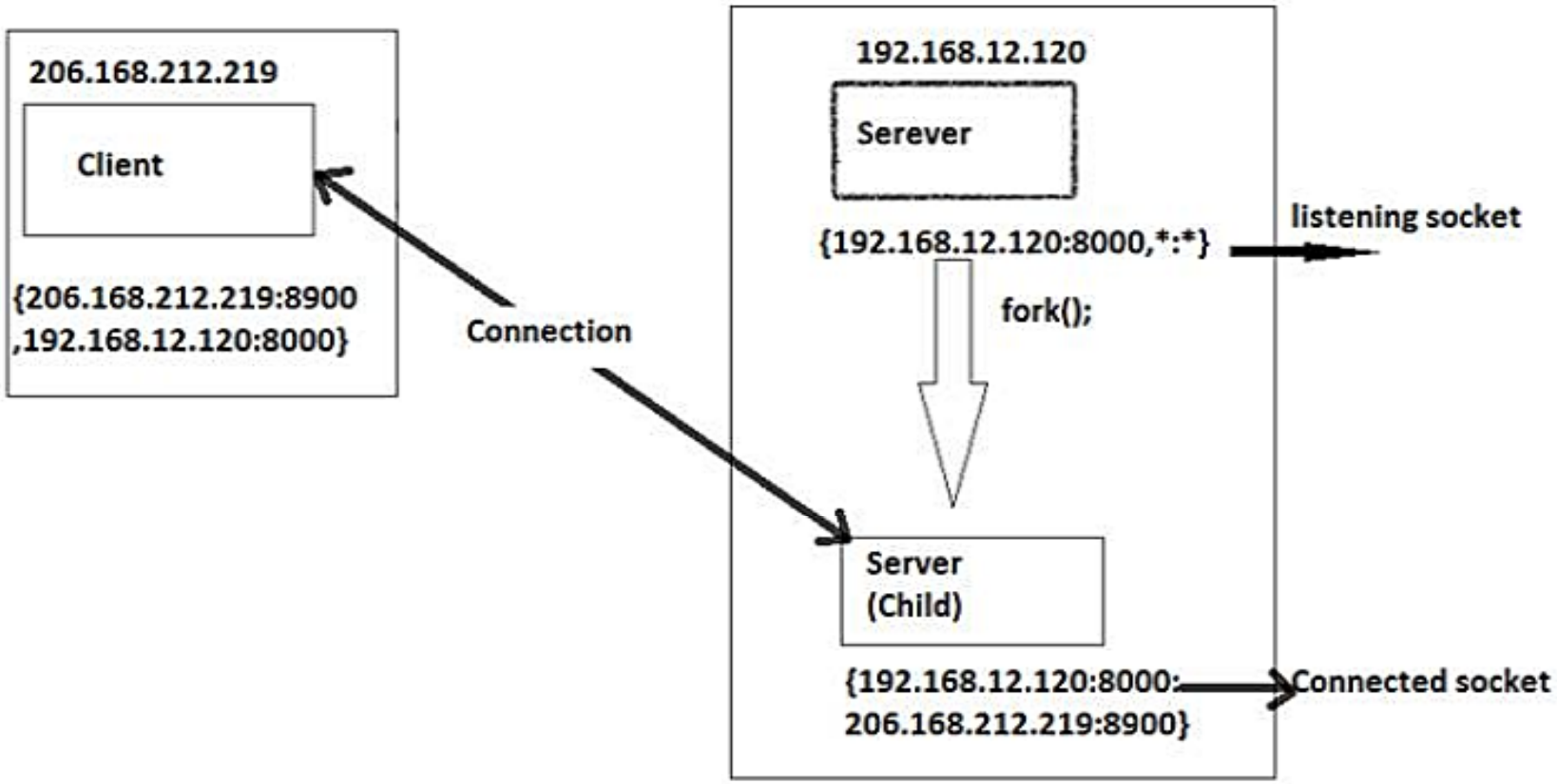
# Distributed Computing: Client- Server
# TCP Concurrent Server

- The TCP servers are concurrent server by nature.
- It means the main server loop creates a child to handle each new connection.
- Lets assume that server has IP Address 192.168.12.120.
- The server does passive open using a well-known port 8000. It is now waiting for client to request.
- Socket pair can be assumed as: {192.168.12.120:8000,*:*}
- One the client sends request and it is accepted wild card character *:* is replaced with the IP Address and Port Number of Client.



# Distributed Computing: Client- Server TCP Concurrent Server

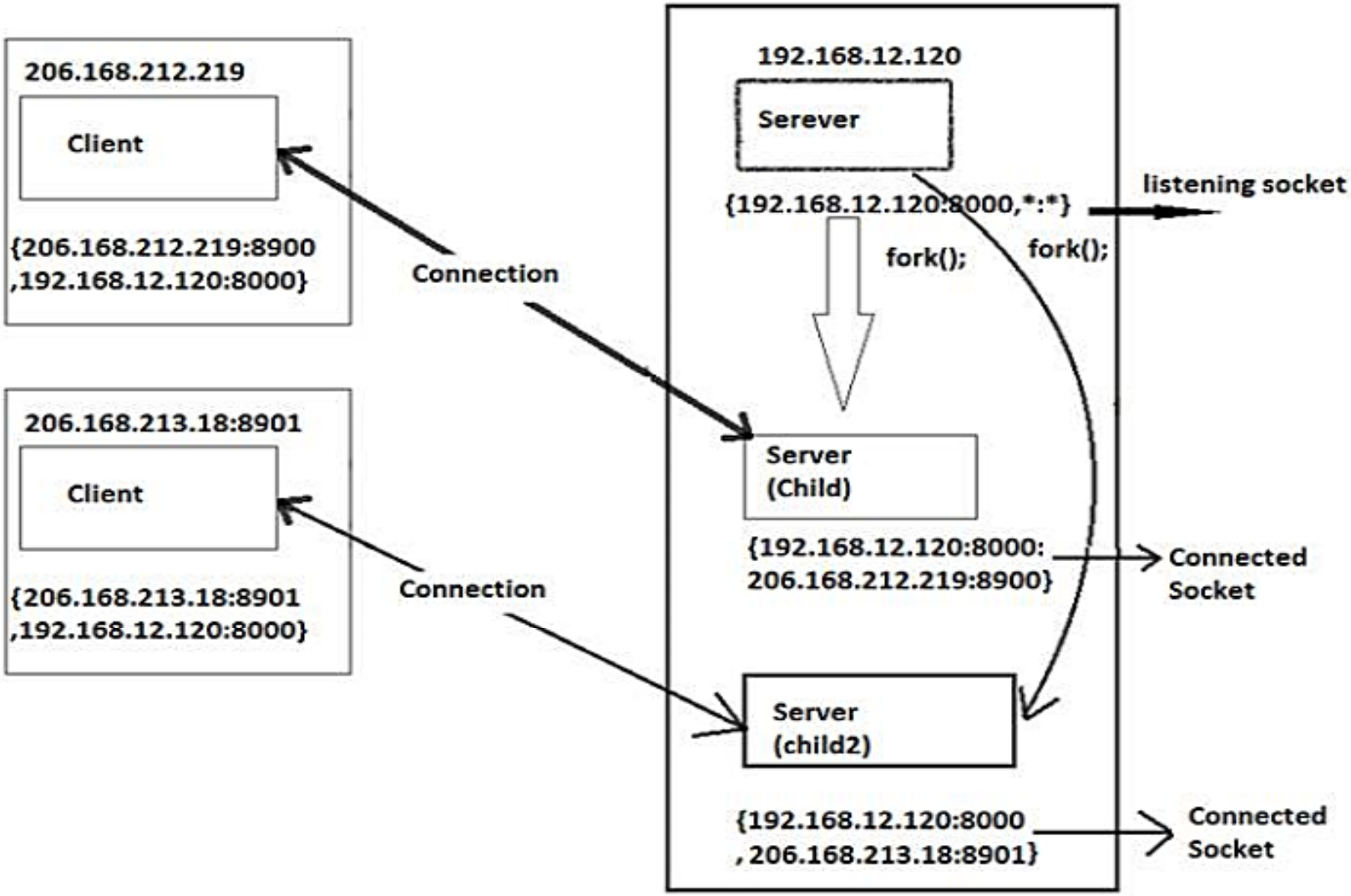
- Connected Socket with Client.



# Distributed Computing: Client- Server TCP Concurrent Server

- Lets assume that client start on the host with IP address 206.168.212.219 with port number 8900.
- When server receives and accepts the client's connection request, it forks a copy of itself, letting the child handle the client.
- Connected socket uses the same local port 8000.
- If another client send the request the server will again create its copy and let second child to handle the second client.

# Distributed Computing: Client- Server TCP Concurrent Server



# Distributed Computing: Client- Server
# TCP Concurrent Server

- How TCP does de-multiplexing.
- TCP can not de-multiplex incoming segment by looking at just the destination port number.
- It must look at all four element in the socket pair.
- We have three socket with same local post 8000.
- Depending upon the values of socket pair packets will be delivered to appropriate destination.

# References


- Sources :
- https://www.javatpoint.com/osi-model
- https://www.tutorialspoint.com/software_architecture_design/distributed_architecture.htm
- https://sites.google.com/site/prajapatiharshadb/class-notes-for-students
- UNIX Network Programming by W. Richard Stevens, Prentice Hall Publication
- Distributed Computing: Concepts &Applications: by M. L. Liu Addison Wiselly