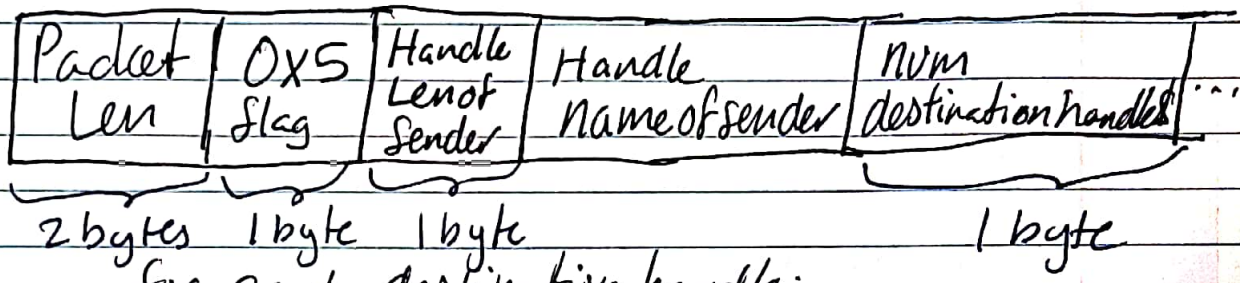
Angela Kerlin
CPE464-01

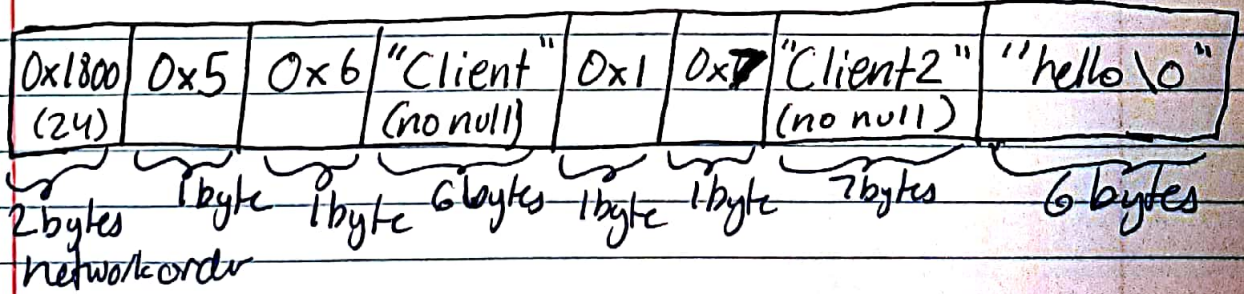
Program 2 Design Work

☆ Part 1

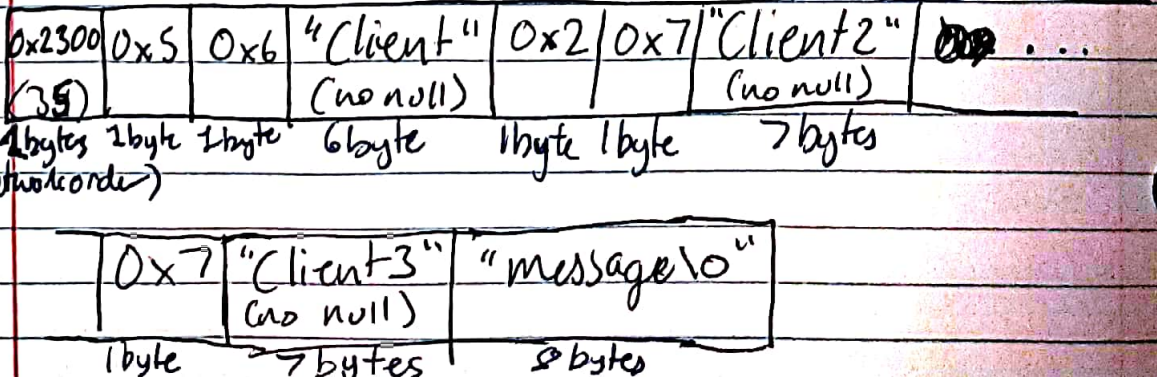
1) C to M : flag 5



- One destination handle



- Two destination handles



- To B : flag 4

Packet length	0x4 flag	Length of Sending Clients handle	Sending handle name	Text message
2 bytes (network order)	1 byte	1 byte	variable size	Variable size

- To L : flags 10

client to server

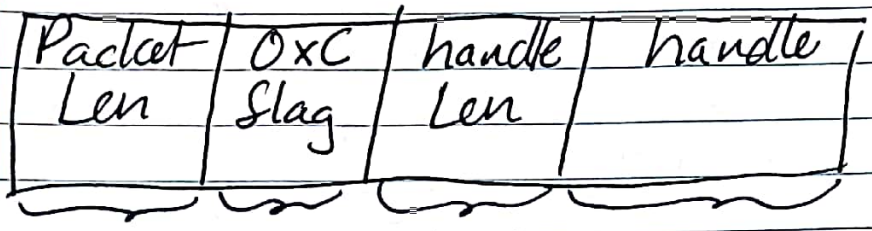
Packet Len (3)	0xA flag
2 bytes (network order)	1 byte

- flag 11

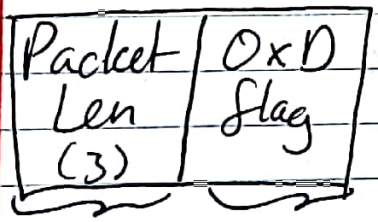
server to client

Packet Len	0xB flag	Number of handles known by the server
2 bytes (network order)	1 byte	4 bytes (network order)

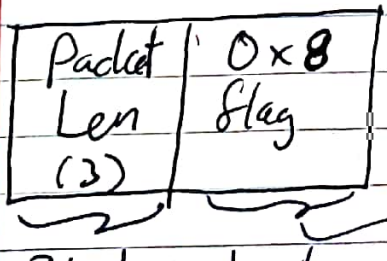
- flag 12 Server to client



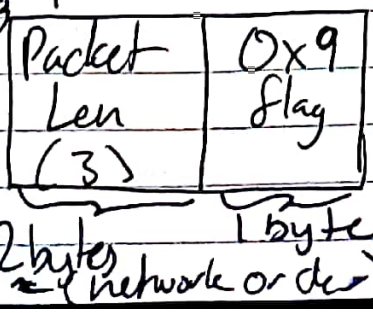
- flag 13 Server to client



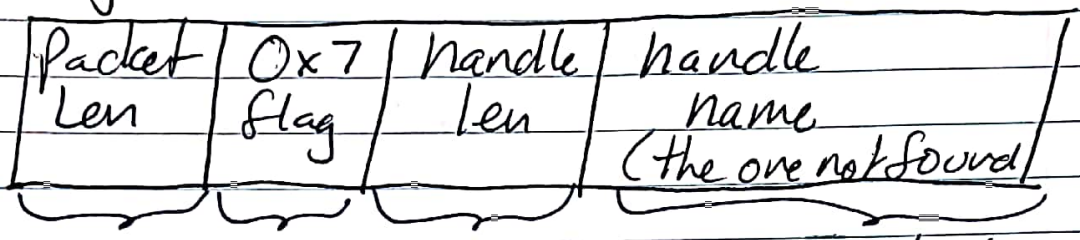
- 0xE : flag 8



- flag 9

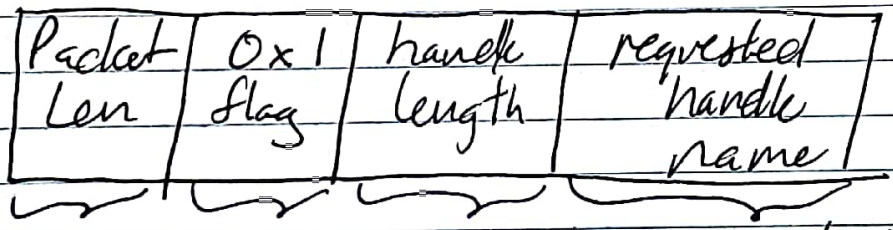


- Flag 7 error packet, handle does not exist



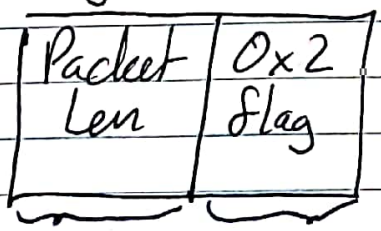
2 bytes (network order) 1 byte 1 byte variable length

- Flag 1 initial packet to server



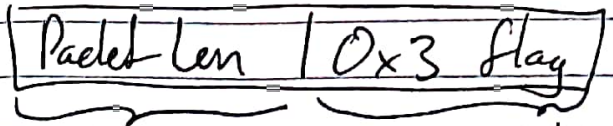
2 bytes (network order) 1 byte 1 byte variable length

- Flag 2 Confirm handle name



2 bytes (network order) 1 byte

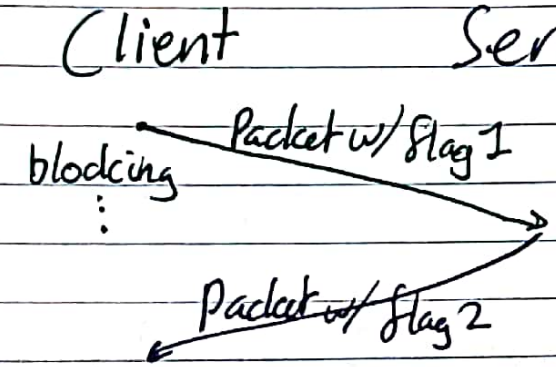
- Flag 3 error bad handle requested



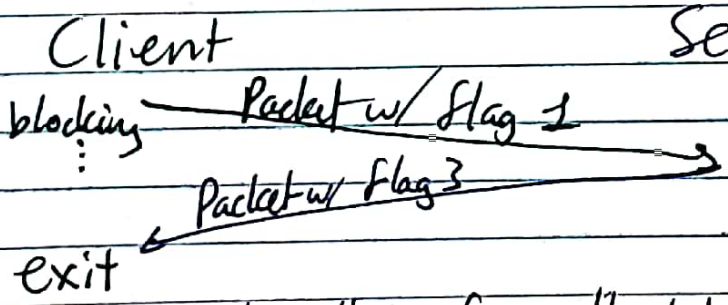
2 bytes Network order 1 byte

2) (a) Connection setup

- Case 1: handle is valid, no errors



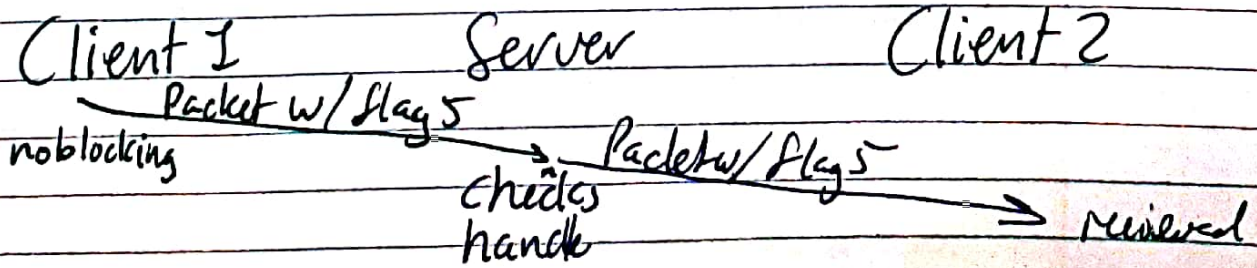
- Case 2: handle already exists, server throws error



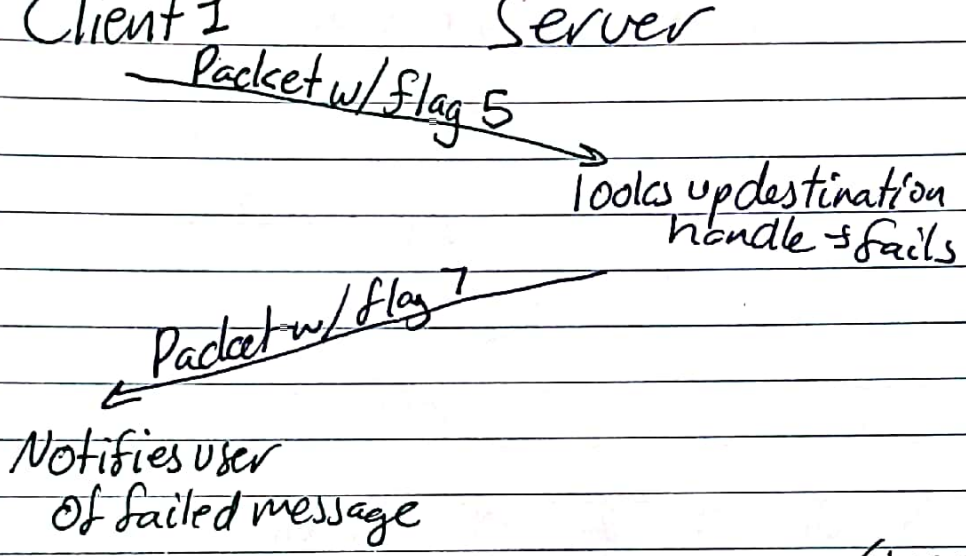
- Case 3: handle is formatted badly (too long, includes whitespace, etc). Client processes handle and does not send it to server, client exits

(b) Sending a message to a single client

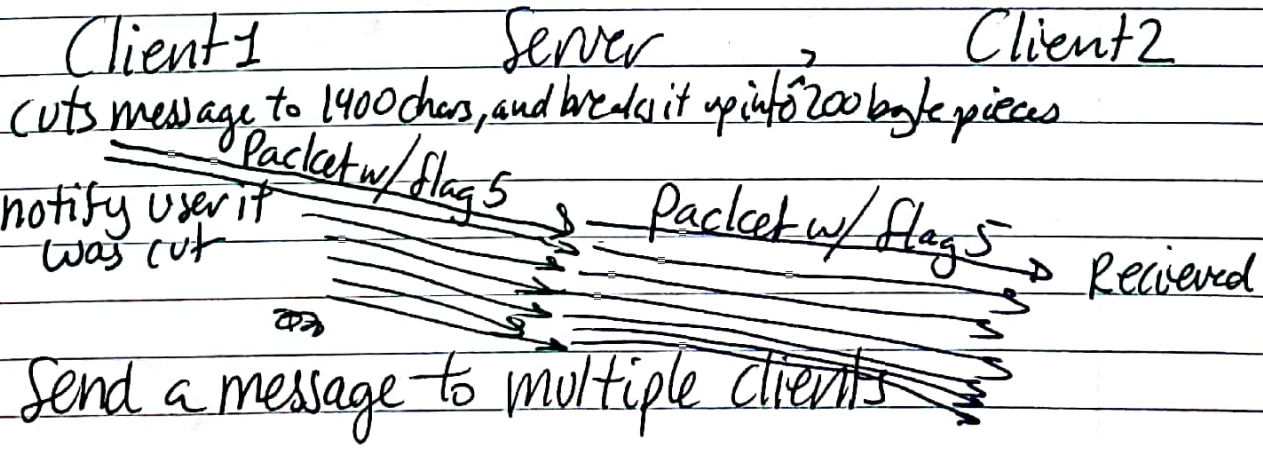
- Case 1: no errors, destination client exists



- Case 2: destination handle doesn't exist

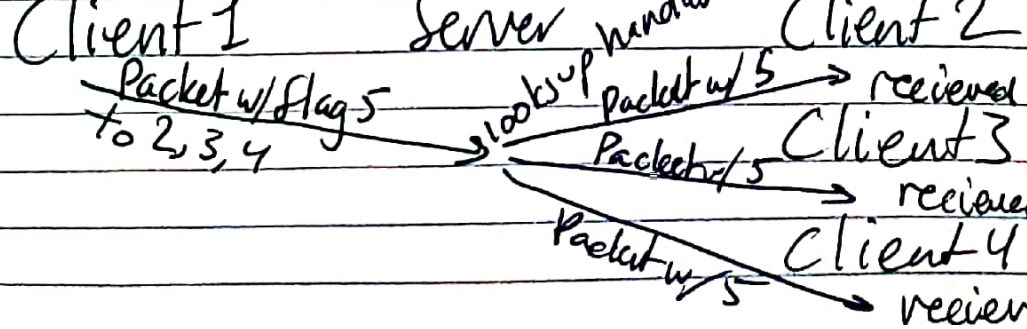


- Case 3: message exceeds max length (1400 characters), client cuts it off

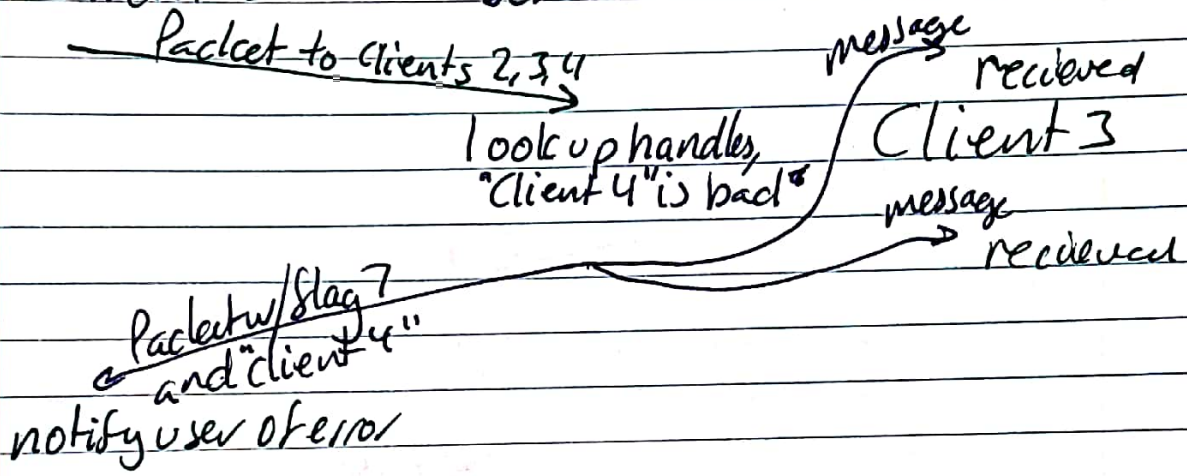


(C) Send a message to multiple clients

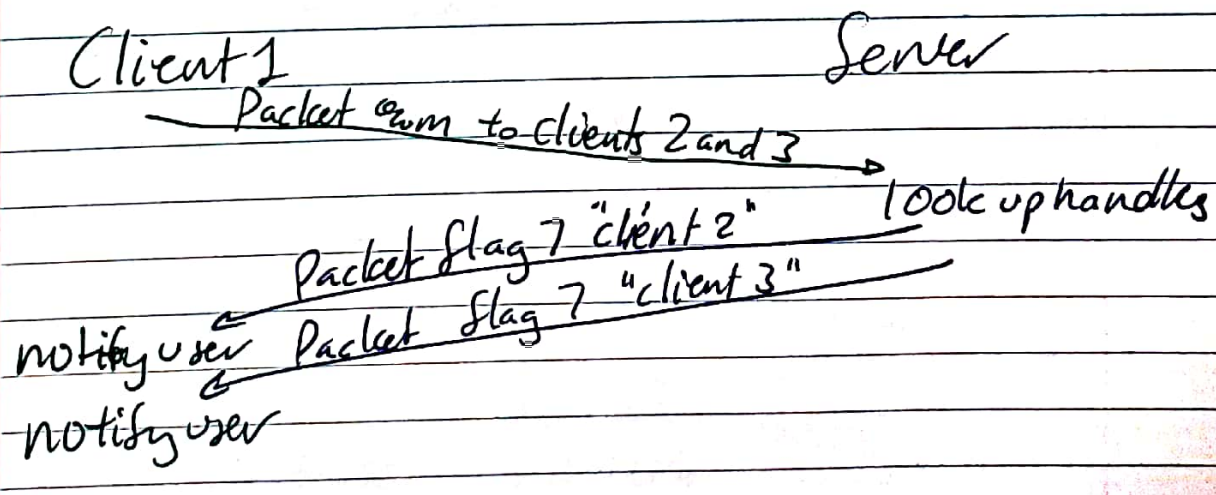
- Case 1: no errors, handles are all valid



- Case 2: One bad handle, server sends to valid, and errors to client 1

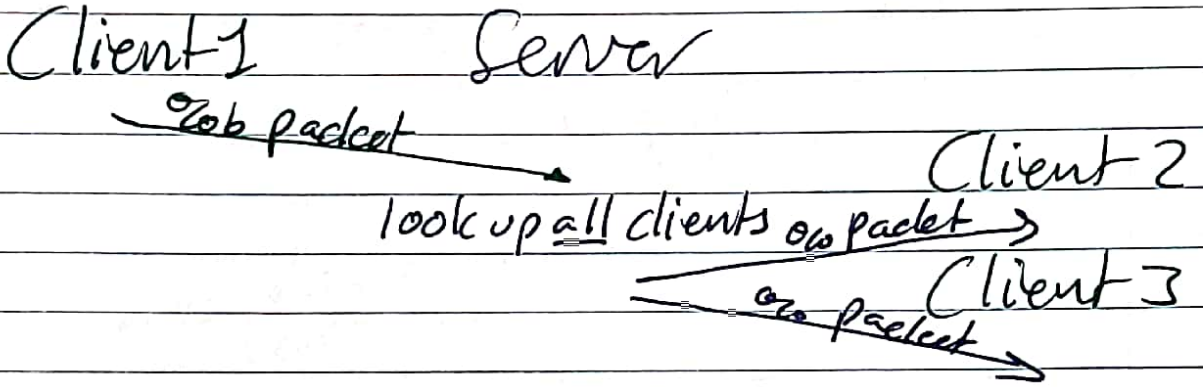


- Case 2: two bad handles, no good handles, server sends two error packets (flag 7)

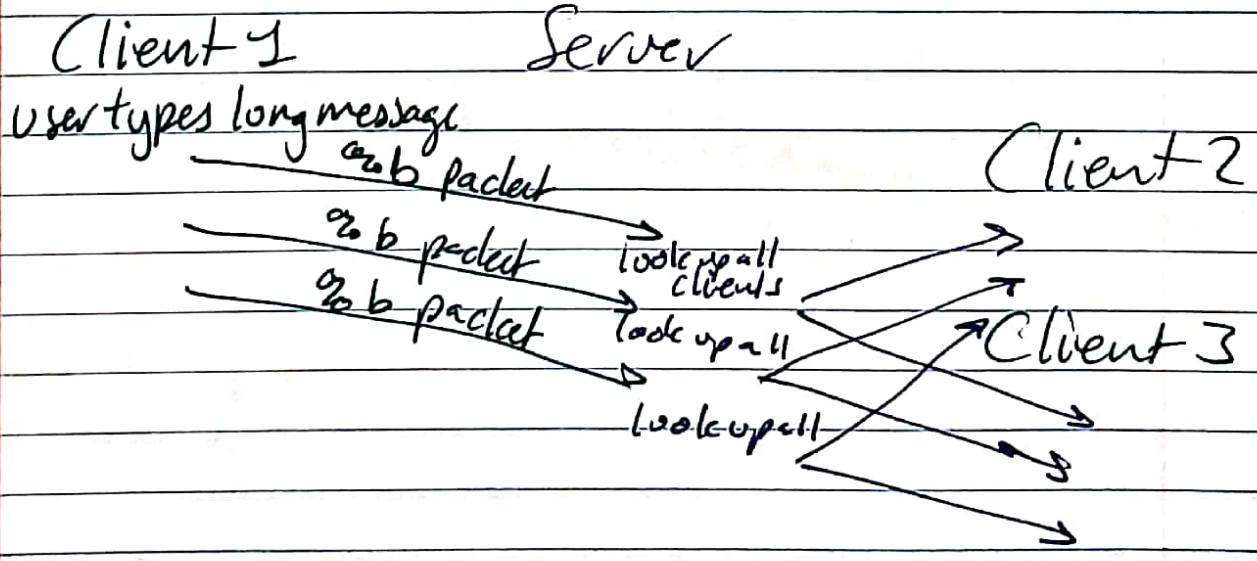


① Broadcast a message

-Case 1: broadcast message 200 bytes or smaller

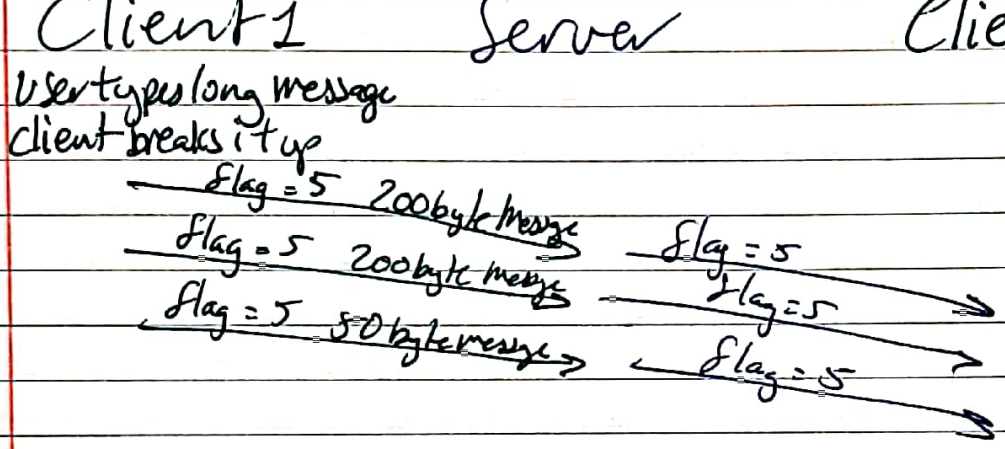


-Case 2: broadcast message is greater than 200 bytes, Client breaks message into 200 byte pieces and sends separately



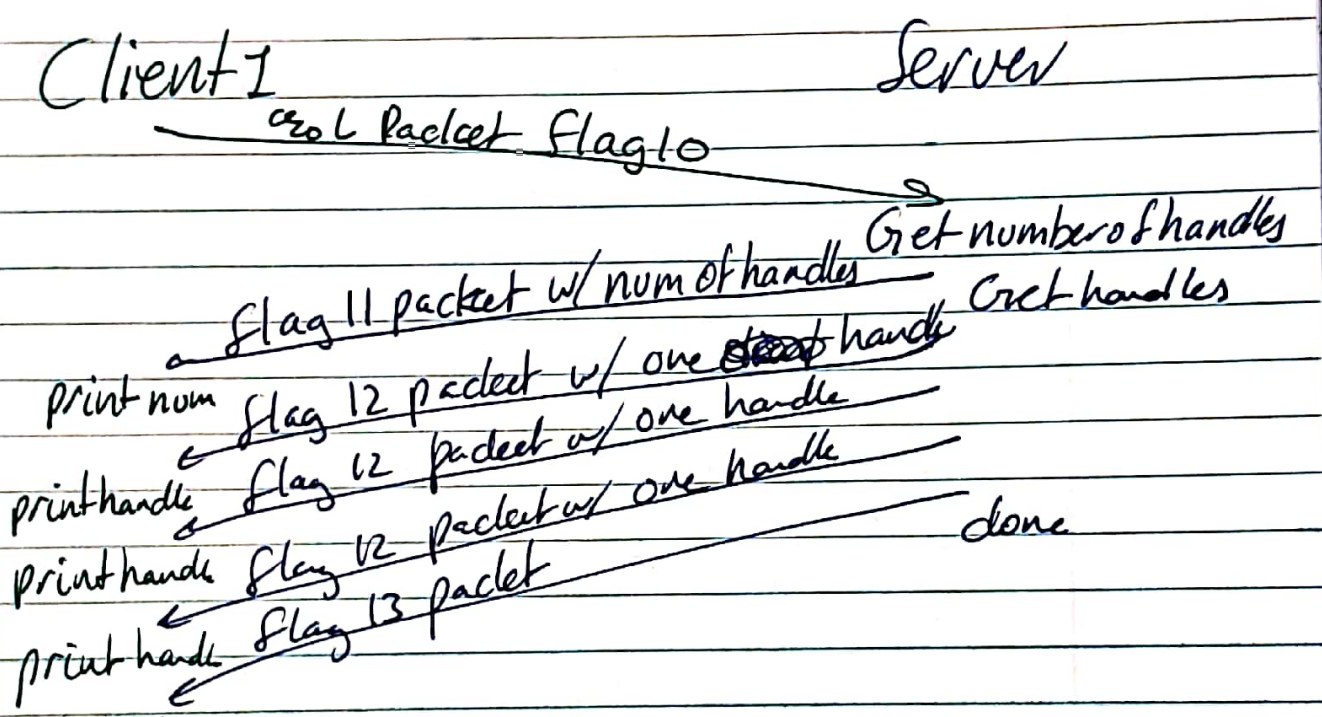
(e) sending a message of 450 characters

Client will break it into 3 seperate messages and the server will just see it as 3 messages



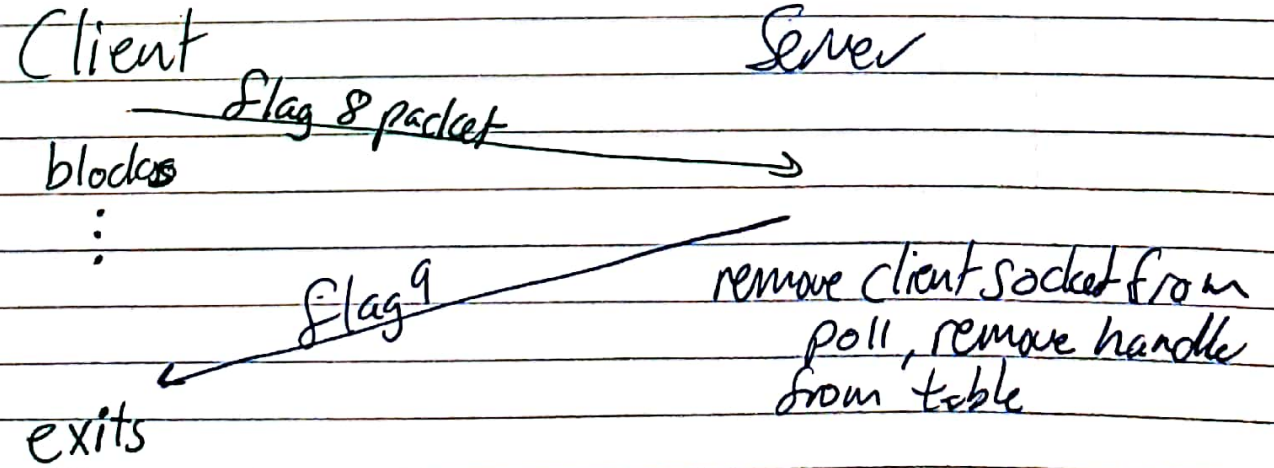
(f) Listing the handles

-Case 1: no errors, assume there are 3 clients



(g) ending a connection

-Case 1: no errors, client sends flag 8 message, server ACKs then removes handle from system



3) Handle Table data structure on the server.

a.

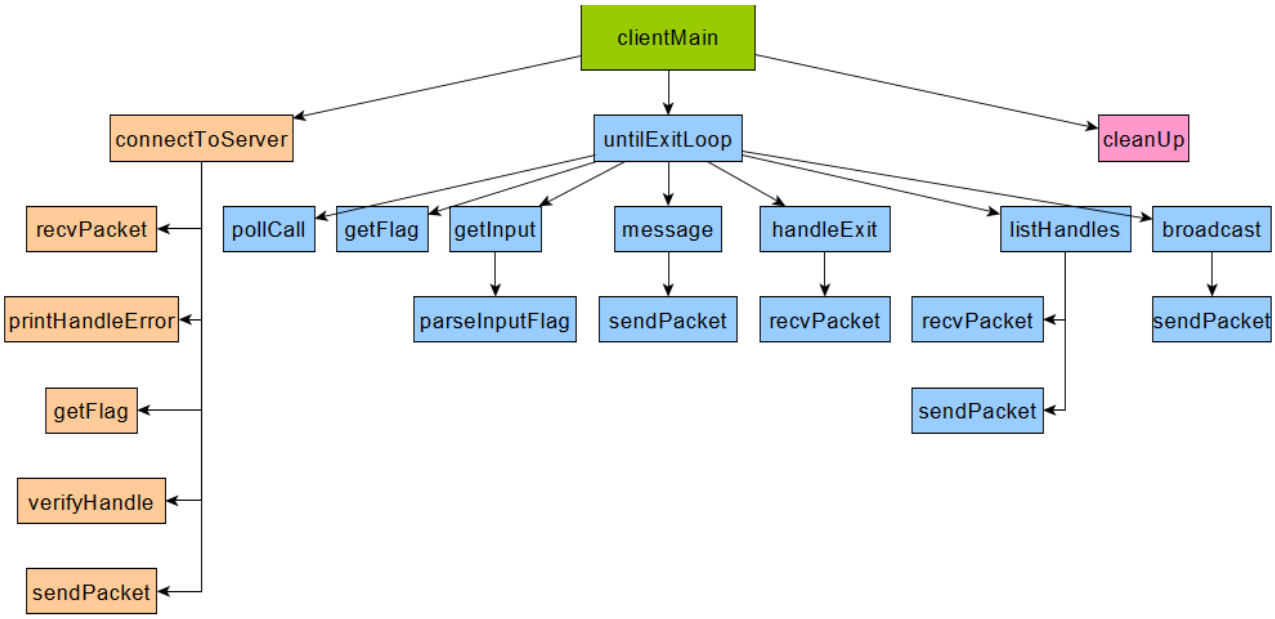
Socket/Handle State on the <u>Server</u>	**How can this happen (what can cause this on the server)**	**If you looked at your socket/handle table, how do you know the socket/handle is currently in this state (what in your data structure would tell you this)**
Client socket is opened but handle not valid	A client attempted to connect with a bad handle, but the server did not close the socket.	I would indicate this with a null value for the handle in the table.
Client Socket open and client handle is valid	A client successfully connected to the server with a valid handle	Entry in the socket/handle table for the open client and valid handle.
Socket was opened and had a handle, but then the socket was closed	Client successfully connected then exited.	No entry for the handle or socket

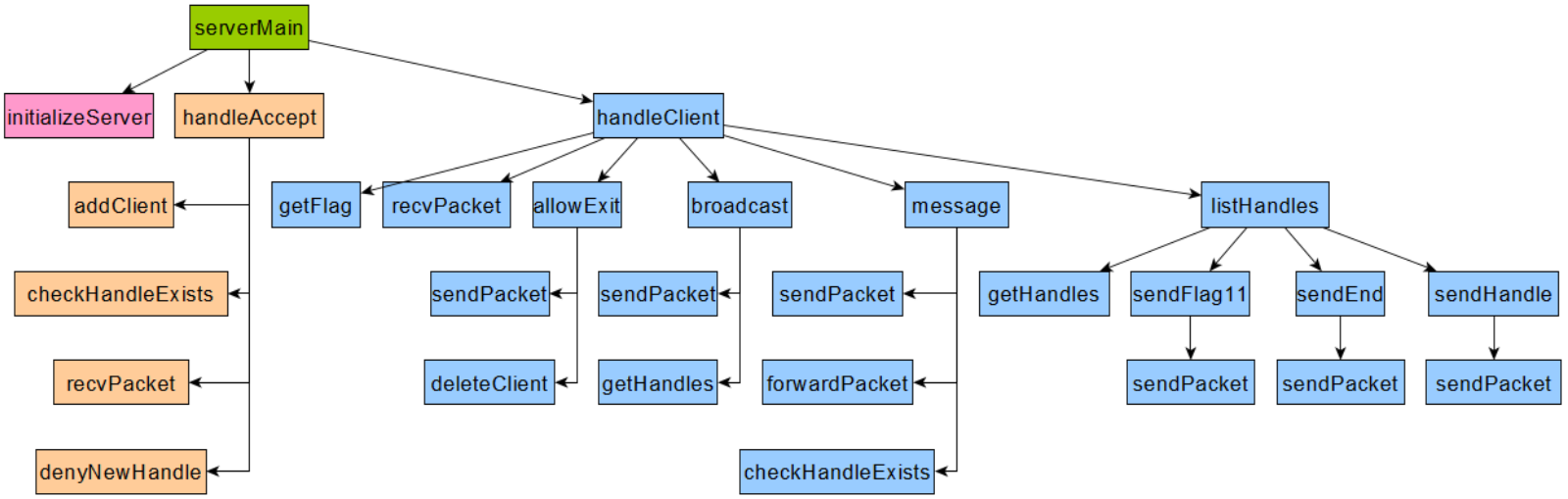
b.

- getSocket(handle) -> returns socket number associated with that handle or null if there is no such handle
- getHandle(socket) -> returns the handle associated with that socket if it exists, else null
- addSocketHandle(socketNumber, handle) -> returns 0 if the handle is invalid, 1 on success: checks for valid handle then creates a new entry in the table for socketNumber/handle and adds the socketNumber to the poll set
- closeSocketHandle(socketNumber) -> returns 0 if no such socket exists, 1 on success: closes socket, removes socket from poll set, and deletes entry in table
- getAllHandles(char **list) -> returns number of handles and fills given list with the handles

Part II – Hierarchical drawing and listing common code

a. **Hierarchical drawing:**





b. **Common Code**:

- Flag macros
- Uint16_t getPacketLen(char * buf)
- Int getFlag(char * buf)
- sendPacket()
- recvPacket()
- pollLibrary