Inter-client communication via the server

In this homework, you are provided with a multiclient-server implementation. The clients can send messages to the server, and the server can broadcast messages to the clients. You need to extend this implementation to allow one client to talk to one or more clients.

Get and build the assignment using:

git clone https://github.com/Systems-IIITD/CN
cd hw_cs
make

It'll create two files: server and client.
./server starts the server
./client starts the client

When you enter something on a client's terminal, it should be displayed on the server's terminal.
When you enter something on the server's terminal, it must be printed on all client terminals.

When a server receives a message from the client, the `recv_message` routine (as described below) in the `server_helper.c` is called.

```
void recv_message(char *msg, int len, int client_id, int *valid_ids, int num_clients)
```

The first argument is the message.
The second argument is the length of the message.
The third argument is an integer ID corresponding to the client.
The fourth argument is an array of valid client IDs. If `valid_ids[cid]` is -1, it means the client ID `cid` is invalid.
The fifth argument is the size of the `valid_ids` array. It means the maximum value of a valid client ID can be `num_clients - 1`.

You can use the `send_message` API for sending messages to other clients. `void`

```
send_message(char *msg, int len, int dst_id, int src_id);
```

The first argument is a character array, `msg`, that needs to be sent.
The second argument is the length of the array `msg`.
The third argument is the destination client ID.

The fourth argument is the source client ID. If the source is the server, you should use `SERVER_ID` as `src_id`.
To facilitate inter-client communication, we require that client messages follow a specific format.

The client can obtain the list of all the valid client IDs using the following message.
LIST

If a client wants to send "Hello World" to client IDs 5, 7, and 8, it can use the following format.
DATA 3 5 7 8: Hello World

The message starts with DATA followed by a space and the number of recipients, followed by a space and the IDs of each of the recipients separated by spaces, followed by a colon(:), a space, and the message.

It means that in the `recv_message`, you can receive data that either starts with "DATA" or "LIST". You can use `sscanf` or any other library calls to parse the string, and use the `send_message` API to send messages as required. The `valid_ids` and `num_clients` arguments can be used to create a list of valid clients.

Submission:

Implement everything in the `server_helper.c`. Create a design documentation with the following information.

Start the server.
Start five clients from different terminals.

- What is the output on all clients' and the server's terminals when the fifth client enters **LIST**
- What is the output on all clients' and the server's terminals when the fifth client enters **DATA 3 3 1 2: hello world**
- What is the output on all clients' and the server's terminals when the fifth client enters **DATA 1 0: hello world**

Submit `server_helper.c` and your design documentation. Use the submission guidelines for the assignments and the homework.