Aditya dixit -22030

Pranjal Bharti-21090
Kanav meena-23266

Design Documentation: Inter-Client Communication via Server

1. Overview

This assignment extends a multi-client server system to allow **inter-client messaging** via the server.
Clients can now send messages to:

1. The server (broadcast from server to all clients).
2. One or more clients (inter-client communication).

2. Implementation Details

2.1 Server (`server.c` + `server_helper.c`)

- The server maintains:
  - `client_socks[MAX_CLIENTS]`: stores socket descriptors for each client.
  - `valid_ids[MAX_CLIENTS]`: marks valid/active clients.
  - `num_clients`: number of connected clients.

- Incoming client messages are handled in `recv_message()`:

  - **LIST**

    - Server collects all active client IDs.
    - Sends the list **back to the requesting client** (or optionally broadcasts to all clients).
    - Server logs the request and response.

```
// Example server log:
```

```
client 4 requested LIST -> sent list: Active clients: 0 1 2 3 4
```

- **DATA <N> id1 id2 ... idN: message**
  - Parses the recipient count and client IDs.
  - Validates each recipient against `valid_ids`.
  - Sends the message to each valid client using `send_message()`.
  - Acknowledges the sender with the number of recipients and any invalid IDs.

```
// Example server log:
client 4 sent DATA to 3 clients: hello world
```

- **Unknown commands**
  - Logs unrecognized commands.
  - Optionally, can send an error message to the client.

- **Server `send_message()`**
  - Prepends either `server:` or `client-<id>:` to messages.
  - Sends via the client socket.
  - Handles socket errors and marks disconnected clients as invalid.

- **Select-based I/O**
  - The server uses `select()` to monitor:
    - `STDIN` (server input for broadcast)
    - Client sockets
    - New connections

2.2 Client (`client.c`)

- Connects to server using TCP on port `5432`.
- Uses `select()` to monitor:
  - `STDIN` for user input
  - Server socket for incoming messages

- Messages from server (broadcasts or replies) are printed immediately.
- Sends user messages to the server:

- `LIST` → requests active client list
- `DATA` → sends messages to specific clients
- Any other text → server can broadcast if implemented

3. Message Formats

LIST

Server replies with a list of active clients:

Client 4 input:

```
LIST
```

Server terminal output:

```
client 4 requested LIST -> sent list: Active clients: 0 1 2 3 4
```

Client 4 terminal output:

```
server: Active clients: 0 1 2 3 4
```

DATA

Server terminal output:

```
client 4 sent DATA to 3 clients: hello world
```

Client terminals:

- **Client 1 receives:**
- `client-4: hello world`
- `client-4: hello world`
- `client-4: hello world`

  Client 0 and Client 4: No output (client 0 was not targeted, client 4 is the sender).

DATA 1 0: hello world

Server terminal output:

```
client 4 sent DATA to 1 clients: hello world
```

- **Client 0 receives:**

```
client-4: hello world
```

Clients 1–4:

```
No output (only client 0 was targeted).
```

Command	Server Output	Client 0	Client 1	Client 2	Client 3	Client 4
LIST	client 4 requested LIST -> sent list: Active clients: 0 1 2 3 4	–	–	–	–	server: Active clients: 0 1 2 3 4
DATA 3 3 1 2: hello world	client 4 sent DATA to 3 clients: hello world	–	client-4: hello world	client-4: hello world	client-4: hello world	–
DATA 1 0: hello world	client 4 sent DATA to 1 clients: hello world	client-4: hello world	–	–	–	–