**• Xác định message type cho login và text message**

| **Tên Message** | **Type ID (Hex)** | **Hướng** | **Mô tả** |
| --- | --- | --- | --- |
| LOGIN\_REQUEST | 0x01 | Client -> Server | Client gửi tên đăng nhập. |
| LOGIN\_ACK | 0x81 | Server -> Client | Server phản hồi đăng nhập. |
| MESSAGE\_DATA | 0x02 | Client -> Server | Client gửi một tin nhắn văn bản. |
| ERROR\_NOT\_LOGGED\_IN | 0xF1 | Server -> Client | (Tùy chọn) Server báo lỗi. |

**• Xác định các trường trong mỗi message và độ dài cũng như kiểu của từng trường**

**a) LOGIN\_REQUEST (Client -> Server)** Gói tin này dùng để client đăng ký login\_name của mình với server.

#define MAX\_NAME\_LEN 30

struct LoginRequest {

uint8\_t type; // Luôn là 0x01

char login\_name[31]; // Tên đăng nhập (tối đa 30 ký tự + 1 ký tự null '\0')

};

// Tổng kích thước: 32 bytes

**b) LOGIN\_ACK (Server -> Client)** Server cần phản hồi để client biết đã đăng nhập thành công hay chưa.

struct LoginAck {

uint8\_t type; // Luôn là 0x81

uint8\_t status; // 0 = OK, 1 = Tên đã tồn tại, 2 = Tên không hợp lệ

};

// Tổng kích thước: 2 bytes

**c) MESSAGE\_DATA (Client -> Server)** Gói tin chứa văn bản cần ghi log.

#define MAX\_MSG\_LEN 510

struct MessageData {

uint8\_t type; // Luôn là 0x02

char message[511]; // Nội dung tin nhắn (tối đa 510 ký tự + 1 ký tự null '\0')

};

// Tổng kích thước: 512 bytes

**• Xác định xem client/server xử lý các message như thế nào**

**Server**

Server cần duy trì một "bảng" (ví dụ: hash map, std::map trong C++) để ánh xạ địa chỉ của client (struct sockaddr\_in) với login\_name của họ.

Map<ClientAddress, string> client\_sessions;

1. Khởi tạo socket (UDP), bind() vào một port (ví dụ: 9000).
2. Vào vòng lặp vô tận:

a. Gọi recvfrom() để chờ nhận một gói tin. Hàm này sẽ trả về packet\_data (dữ liệu) và client\_address (địa chỉ của người gửi).

b. Kiểm tra type của gói tin (ví dụ: packet\_data[0]).

c. **Nếu type == LOGIN\_REQUEST (0x01):**

i. Đọc login\_name từ gói tin.

ii. Kiểm tra xem login\_name này đã được sử dụng bởi một client\_address *khác* chưa.

iii. **Nếu hợp lệ:**

1. Lưu trữ ánh xạ: client\_sessions[client\_address] = login\_name.

2. Tạo gói tin LOGIN\_ACK (status 0 = OK).

3. Gửi LOGIN\_ACK về client\_address bằng sendto().

iv. **Nếu không hợp lệ:** Gửi LOGIN\_ACK (status 1 hoặc 2) về client\_address.

d. **Nếu type == MESSAGE\_DATA (0x02):**

i. Tra cứu client\_address trong bảng client\_sessions.

ii. **Nếu tìm thấy:**

1. Lấy ra login\_name tương ứng.

2. Mở (hoặc tạo mới) file log, ví dụ login\_name + ".log".

3. Ghi nội dung message từ gói tin vào cuối file.

iii. **Nếu không tìm thấy:** (Client gửi tin nhắn mà chưa login)

1. (Tùy chọn) Gửi gói tin ERROR\_NOT\_LOGGED\_IN (0xF1) về client\_address.

2. Hoặc, đơn giản là bỏ qua gói tin này.

e. **Nếu type không xác định:** Bỏ qua gói tin.

**Client**

1. Khởi tạo socket (UDP).
2. Lấy địa chỉ server (IP, port).
3. **Giai đoạn Login:**

a. Lấy login\_name từ người dùng.

b. Tạo gói tin LOGIN\_REQUEST.

c. sendto() gói tin này đến server.

d. Chờ phản hồi LOGIN\_ACK từ server (nên dùng select() hoặc setsockopt với SO\_RCVTIMEO để xử lý timeout).

e. Nếu nhận được LOGIN\_ACK với status = 0 (OK), chuyển sang giai đoạn gửi tin.

f. Nếu status báo lỗi hoặc bị timeout, thông báo cho người dùng và thử lại hoặc thoát.

1. **Giai đoạn Gửi tin (đã login thành công):** a. Vào vòng lặp: i. Lấy nội dung tin nhắn văn bản từ người dùng. ii. Tạo gói tin MESSAGE\_DATA. iii. sendto() gói tin này đến server. b. (Theo yêu cầu, client không cần chờ phản hồi cho mỗi tin nhắn).

• Vẽ protocol state machine