1. Gói tin gửi nhận Server Client
   1. Cấu trúc gói tin

|  |  |
| --- | --- |
| HEADER REGION (256 bytes) | DATA REGION ( 7936 bytes ) |
| PACKAGE ( 8192 bytes ) | |

* **Header Region (256 bytes)**

|  |  |  |  |
| --- | --- | --- | --- |
| Tên trường | Vị trí | Kích thước | Diễn giải |
| COMMAND | 0 (0x00) | 4 bytes | Loại gói tin (EmessageCommand) |
| TOTAL\_SIZE | 4 (0x04) | 4 bytes | Kích thước gói tin |
| CUR\_PACK | 8 (0x08) | 4 bytes | Thứ tự gói tin hiện tại |
| SRC | 12 (0x0C) | 64 bytes | Hash địa chỉ nguồn (SHA256) |
| DES | 76 (0x4C) | 64 bytes | Hash địa chỉ đích  (SHA256) |
| Flag Raw | 140 (0x8C) | 1 bytes | 0x00 : Không sử dụng Raw Src, Des  0x01 : Sử dụng Raw Src, Des |
| Raw SRC | 141 (0x8D) | 11 bytes | Địa chỉ nguồn |
| Raw DES | 152 (0x98) | 11 bytes | Địa chỉ đích |
| Num Data Item | 163 (0xA3) | 4 bytes | Số lượng trường dữ liệu trong vùng Data |
| Delim | 167 (0xA7) | 1 bytes | Ký tự phân cách giữa các trường dữ liệu trong vùng Data |
| Reversed | 168 (0xA8) | 88 bytes | Dữ liệu dự trữ (dùng khi cần thiết) |

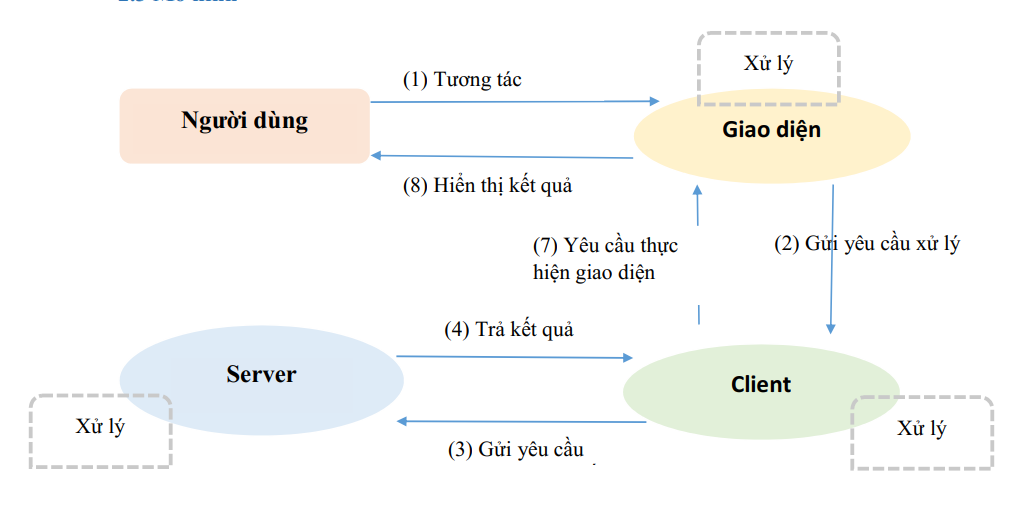
* **Data Region (8192 bytes)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DATA 1 | DELIM | DATA2 | DELIM | … |  |

* Dữ liệu được phân tách bởi Delim trong header, mặc định là \0
  1. Các cấu dựa trên các mục đích sử dụng

|  |  |  |
| --- | --- | --- |
| FLAG | Các trường header sử dụng | Ý nghĩa |
| CLIENT\_SIGN\_IN | Src : client  Des: server | Username: tên tài khoản  Password: mật khẩu |
| SERVER\_REPONSE\_SIGN\_IN\_SUCESS | Src: server  Des: client | Tên user |
| SERVER\_RESPONSE\_SIGN\_IN\_ERROR | Src: server  Des: client | Tên user |
| CLIENT\_SIGN\_UP | Src : client  Des: server | Username: tên tài khoản  Password: mật khẩu |
| CLIENT\_REQUEST\_PRIVATE\_CHAT | Src: client  Des: server | Username: Tên tài khoản cần tạo hội thoại riêng |
| Src: server  Des: client | Des\_user: Tài khoản người chat  Src\_user:  Tên tài khoản người tạo hội thoại reieng  Hash\_conversation: Hash id cuộc trò chuyện đã được tạo từ server |
| CLIENT\_SEND\_PRIVATE\_CHAT | Src: tên tài khoản gửi  Des: Server  HashSrc: SHA256 của tài khoản gửi  HashDes: ID hash của cuộc hội thoại | Message: Nội dung cuộc trò chuyện dưới dạng chuỗi không chứa NULL “\0” |
| CLIENT\_REQUEST\_LIST\_ONLINE | Src: Client  Des: Server | Yêu cầu server gửi danh sách tài khoản online |
| Src: Server  Des: Client | U1,U2,…Un là tên các tài khoản đang online |
| CLIENT\_REQUEST\_GROUP\_CHAT | Src: client  Des: server | ChatName: tên nhóm hội thoại  U1,U2,…Un Danh sách thêm vào nhóm |
| SERVER\_RESPONSE\_HASH\_KEY | Src: server  Des: client | HashID: SHA256 Hash id của cuộc trò chuyện  ChatName: Tên cuộc trò chuyện  U1,U2,U3,…Un Danh sách thành viên cuộc trò chuyện |
| CLIENT\_SEND\_GROUP\_CHAT | Src: tên tài khoản gửi  Des: Server  HashSrc: SHA256 của tài khoản gửi  HashDes: ID hash của cuộc hội thoại | Message: Nội dung cuộc trò chuyện dưới dạng chuỗi không chứa NULL “\0” |
| CLIENT\_REQUEST\_TRANSFER\_FILE | Src: Tên tài khoản gửi  Des: “Server”  SrcHash: SHA256 tên tài khoản gửi  DesHash: ID của cuộc trò chuyện  NumPackage: ChunkSize (**sử dụng đặc biệt**)  TotalSize: Kích thước file truyền | FileName: Tên file truyền |
| CLIENT\_BEGIN\_TRANSFER\_FILE | Src: Tên tài khoản gửi  Des: “Server”  SrcHash: SHA256 tên tài khoản gửi  DesHash: ID của cuộc trò chuyện  NumPackage: Số thứ tự của chunk  TotalSize: Kích thước của File | Dữ liệu file (có thể chứa NULL) kích thước bằng chunksize yêu cầu |
| CLIENT\_END\_TRANSFER\_FILE | Src: Tên tài khoản gửi  Des: “Server”  SrcHash: SHA256 tên tài khoản gửi  DesHash: ID của cuộc trò chuyện | Không sử dụng |

1.3. Mô hình đối tượng



* Thành phần trong hệ thống
* Server: Cấu trúc xử lý trong lớp ServerSocket, tiếp nhận và xử lý trực tiếp đối với các gói tin nhận từ Client
* Client:

+ ClientBackgroundService: Tiếp nhận trung gian điều khiển quá trình xử lý. Được tạo ra một lần duy nhất và được truyền giữa các Form khác nhau để sử dụng

+ ClientSocket: Tiếp nhận và gửi gói tin giao tiếp với Server. Các sự kiện nhận được sẽ gửi qua lớp Giao diện để xử lý thông qua

**SendMessageW**(

    \_In\_ HWND hWnd,

    \_In\_ UINT Msg,

    \_Pre\_maybenull\_ \_Post\_valid\_ WPARAM wParam,

    \_Pre\_maybenull\_ \_Post\_valid\_ LPARAM lParam);

1. Kịch bản và Luồng xử lý dữ liệu
   1. Đăng ký tài khoản
2. Sau khi người dùng click vào đăng ký tài khoản, Client sẽ gọi hàm

**void Register(string username, string password)** để gửi gói tin với cấu trúc **CLIENT\_SIGN\_UP** lên server

* **Tham số:**
  + **username: tên tài khoản**
  + **password: mật khẩu**

1. ServerSocket tiếp nhận gói tin trên và kiểm tra bằng hàm trong **AccountManagement**

* Nếu tài khoản tồn tại, phản hồi gói tin **SERVER\_RESPONSE\_SIGN\_UP\_ERROR**
* Nếu tài khoản chưa tồn tại, thêm vào tệp tin dữ liệu tài khoản với dữ liệu tài khoản mới **username,password** và gửi phản hồi **SERVER\_RESPONSE\_SIGN\_UP\_SUCCESS**

1. ClientSocket tiếp nhận phản hồi và gửi gói tin tới lớp giao diện để xử lý

* Thành công:

**SendMessageW**(hwnd, IDC\_FORM\_LOGIN\_MSG\_HANDLER, IDC\_FORM\_LOGIN\_MSG\_HANDLER\_REGISTER\_SUCCESS, 0);

* Thất bại:

**SendMessageW**(hwnd, IDC\_FORM\_LOGIN\_MSG\_HANDLER, IDC\_FORM\_LOGIN\_MSG\_HANDLER\_REGISTER\_ERROR, 0);

1. Lớp giao diện tiếp nhận message gửi từ SendMesssageW

* Thành công: Thông báo thành công và yêu cầu người đăng nhập login để thực hiện các chức năng của client
* Thất bại: Thông báo thất bại, vui lòng chọn tài khoản khác.
  1. Đăng nhập tài khoản

1. Sau khi người dùng click vào nhập tài khoản, Client sẽ gọi hàm

**void Login(string username, string password)** để gửi gói tin với cấu trúc **CLIENT\_SIGN\_UP** lên server

* **Tham số:**
  + **username: tên tài khoản**
  + **password: mật khẩu**

1. **ServerSocket** tiếp nhận gói tin trên và kiểm tra bằng hàm trong **AccountManagement**

* Nếu tài khoản sai mật khẩu hoặc không tồn tại, phản hồi gói tin **SERVER\_RESPONSE\_SIGN\_IN\_ERROR**
* Nếu kiểm tra tài khoarn đúng server sẽ phản hồi gói tin thành công **SERVER\_RESPONSE\_SIGN\_IN\_SUCCESS**

1. **ClientSocket** tiếp nhận phản hồi và gửi gói tin tới lớp giao diện để xử lý

* Thành công:

**SendMessageW**(hwnd, IDC\_FORM\_LOGIN\_MSG\_HANDLER, IDC\_FORM\_LOGIN\_MSG\_HANDLER\_LOGIN\_SUCESS, 0);

* Thất bại:

**SendMessageW**(hwnd, IDC\_FORM\_LOGIN\_MSG\_HANDLER, IDC\_FORM\_LOGIN\_MSG\_HANDLER\_LOGIN\_ERROR, 0);

1. Lớp giao diện tiếp nhận message gửi từ **SendMesssageW**

* Thành công: Khởi tạo lớp **messenger** để mở form cho phép người dùng sử dụng các chức năng với tài khoản đã đăng nhập
* Thất bại: Thông báo thất bại, vui lòng nhập lại mật khẩu
  1. Danh sách tài khoản online
* Kịch bản gửi danh sách tài khoản online được gửi từ Server xuống Client trong các tình huống sau:

+ Có client bị mất kết nối hoặc đăng xuất

+ Có client đăng nhập mới

+ 1 Client yêu cầu lấy danh sách online mới

* Kịch bản xử lý

1. Có yêu cầu hoặc gặp tình huống cần lấy danh sách online
2. Server tiếp nhận yêu cầu và lấy danh sách qua hàm **ServerSocket::GetListOnline();**
3. **ServerSocket** gửi gói tin với các Client hoặc 1 client chỉ định với FLAG **CLIENT\_REQUEST\_LIST\_ONLINE**
4. **ClientSocket** nhận được gói tin và lưu danh sách online vào trong **ClientSocket::\_list\_online** và gửi 1 message tới lớp giao diện

**SendMessageW**(hwnd, IDC\_FORM\_CHAT\_MSG\_HANDLER, IDC\_FORM\_CHAT\_MSG\_HANDLER\_LIST\_ONLINE, 0);

1. Lớp giao diện nhận được message và cập nhật lại **ListControl** trên giao diện và hiển thị thông báo tới các cuộc hội thoại có user logout
   1. CHAT riêng tư

* Tạo chat riêng tư

1. Người sử dụng chọn 1 tài khoản đang online bằng cách double click trên **ListControl** friends.
2. Client sẽ gửi 1 gói tin với cấu trúc **CLIENT\_REQUEST\_PRIVATE\_CHAT** với dữ liệu
   * Tài khoản nhận
   * Tên cuộc trò chuyện (mặc định là tên tài khoản người nhận)
3. Server tiếp nhận gói tin **CLIENT\_REQUEST\_PRIVATE\_CHAT** xử lý tạo **PrivateConversation** và lưu vào danh sách \_lcs quản lý các cuộc hội thoại đã tạo với 1 khoá SHA256 xây dựng dựa trên tên tài khoản khởi tạo cuộc trò chuyện + random 1 chuỗi ngẫu nhiên
4. Server gửi gói tin **SERVER\_RESPONSE\_HASH\_KEY** chứa mã **HASH\_ID** của cuộc trò chuyện và danh sách thành viên trong cuộc trò chuyện
5. Các client có tham gia cuộc trò chuyện nhận được gói tin **SERVER\_RESPONSE\_HASH** xử lý và thêm cuộc trò chuyện vào ListControl bên phải. Nếu cuộc trò chuyện đang được chọn thì hiển thị trực tiếp vào khu vực chat

* Gửi tin nhắn chat riêng tư:

1. Client nhập đoạn hội thoại và ấn button có hình máy bay để gửi.

- Gói tin gửi có cấu trúc CLIENT\_SEND\_PRIVATE\_CHAT

2. Server nhận được sẽ lấy hash\_id của cuộc trò chuyện và gửi tới tất cả client với nội dung gói tin giữ nguyên

3. Các Client nhận gói tin sẽ tìm kiếm cuộc trò chuyện và thêm tin nhắn vào **list\_mess**

**-** Nếu khung chat hiện tại đang là cuộc trò chuyện hiện thời thì hiển thị nội dung tin nhắn lên khung chat

* 1. Tạo CHAT nhóm
* Tạo chat nhóm

1. Người sử dụng 2 tài khoản đang online trở lên bằng cách click chuột trái trên **ListControl** friends chọn các tài khoản đang online và click chuột phải điền các thông tin tên cuộc trò chuyện.
2. Client sẽ gửi 1 gói tin với cấu trúc **CLIENT\_REQUEST\_GROUP\_CHAT** với dữ liệu
   * Tên cuộc trò chuyện
   * Danh sách thành viên trong cuộc trò chuyện
3. Server tiếp nhận gói tin **CLIENT\_REQUEST\_GROUP\_CHAT** xử lý tạo **GroupConversation** và lưu vào danh sách \_lcs quản lý các cuộc hội thoại đã tạo với 1 khoá SHA256 xây dựng dựa trên tên tài khoản khởi tạo cuộc trò chuyện + random 1 chuỗi ngẫu nhiên
4. Server gửi gói tin **SERVER\_RESPONSE\_HASH\_KEY** chứa mã **HASH\_ID** của cuộc trò chuyện và danh sách thành viên trong cuộc trò chuyện
5. Các client có tham gia cuộc trò chuyện nhận được gói tin **SERVER\_RESPONSE\_HASH** xử lý và thêm cuộc trò chuyện vào **ListControl** bên phải. Nếu cuộc trò chuyện đang được chọn thì hiển thị trực tiếp vào khu vực chat

* Gửi tin nhắn nhóm

1. Client nhập đoạn hội thoại và ấn button có hình máy bay để gửi.

- Gói tin gửi có cấu trúc **CLIENT\_SEND\_GROUP\_CHAT**

2. Server nhận được sẽ lấy hash\_id của cuộc trò chuyện và gửi tới tất cả client với nội dung gói tin giữ nguyên

3. Các Client nhận gói tin sẽ tìm kiếm cuộc trò chuyện và thêm tin nhắn vào **list\_mess**

**-** Nếu khung chat hiện tại đang là cuộc trò chuyện hiện thời thì hiển thị nội dung tin nhắn lên khung chat

* 1. Gửi file

1. Người sử dụng chọn button đính kèm file và chọn file cần đính kèm

2. Client sẽ gửi gói tin với cấu trúc **CLIENT\_REQUEST\_TRANSFER\_FILE** với một số dữ liệu đặc biệt:

- NumPackage = Chunksize (kích thước 1 chunk truyền file)

- TotalSize = Kích thước tệp tin

- FileName : Tên file

3. Server nhận được sẽ tìm kiếm dựa trên Hash\_id của cuộc trò chuyện và gửi yêu cầu truyền file tới tất cả các Client thành viên trừ người gửi

4. Các Client thành viên nhận được và gọi hàm void **ClientConversation::InitFileTransferManagement**(string file\_name, int total\_size, int chunk\_size) và gán **transfering = true** để khoá không cho truyền file khác

5. Client gửi file sẽ gửi tiếp các gói tin **CLIENT\_BEGIN\_TRANSFER\_FILE** với

- NumPackage = vị trí gói tin

- TotalSize = kích thước tệp tin

- Dữ liệu: ChunkSize (bytes) của gói tin đang truyền

6. Sau khi gửi hết dữ liệu thì Client gửi file sẽ gửi gói tin **CLIENT\_END\_TRANSFER\_FILE** để kết thúc truyền file

7. Các client sẽ đóng buffer, thông báo MessageBox truyền file thành công và chuyển **transfering = false.**

Lưu ý:

- File có thể mở trực tiếp bằng click vào danh sách file truyền hoặc tin nhắn trên khung chat

- Trong 1 cuộc hội thoại chỉ 1 client truyền và truyền trong 1 thời điểm. Các client khác ấn truyền sẽ bị thông báo lỗi