



---

# NỘI DUNG

<b>I. GIỚI THIỆU</b>	3
1.1 Nội dung và yêu cầu đề án	3
1.2 Phân tích đề án	3
1.2.1 Gửi nhận dữ liệu sử dụng socket kết hợp multithread	3
1.2.2 Chức năng đăng ký tài khoản	3
1.2.3 Chức năng Chat với 1 người	3
1.2.4 Chức năng Chat nhóm	4
1.2.5 Chức năng gửi nhận gói tin	4
1.3 Thông tin nhóm	4
1.3.1 Thông tin thành viên	4
1.3.2 Phân công công việc	4
<b>II. TỔNG QUAN MÔ HÌNH</b>	6
2.1 Ý tưởng	6
2.2 Gói tin gửi nhận	6
2.2.1 Cấu trúc gói tin	6
2.2.2 Các loại gói tin	6
2.3 Mô hình	8
<b>III. CÀI ĐẶT CÁC YÊU CẦU ĐỀ ÁN</b>	9
3.1 Đăng ký tài khoản	9
3.1.1 Các lớp, đối tượng sử dụng	9
3.1.2 Các hàm xử lý	9
3.1.3 Sơ đồ gọi hàm	10
3.2 Chat với 1 người (private chat)	11
3.2.1 Các lớp, đối tượng sử dụng	11
3.2.2 Các hàm xử lý	11
3.2.3 Sơ đồ gọi hàm	14
3.3 Chat với nhóm (group chat)	14
3.3.1 Các lớp, đối tượng sử dụng	14

---

3.3.2 Các hàm xử lý.....	14
3.3.3 Sơ đồ gọi hàm .....	18
3.4 Gửi tập tin .....	18
3.4.1 Các lớp, đối tượng sử dụng .....	18
3.2.2 Các hàm xử lý.....	18
3.4.3 Sơ đồ gọi hàm .....	22
3.5 Yêu cầu nâng cao .....	23
IV. GIAO DIỆN MÀN HÌNH - DEMO .....	23
4.1 Hình ảnh.....	23
4.2 Video.....	25
V. TỔNG KẾT .....	26
VI. ĐÁNH GIÁ .....	26
6.1 Đánh giá mức độ hoàn thành đồ án.....	26
6.2 Đánh giá thành viên trong nhóm .....	26
TÀI LIỆU THAM KHẢO.....	27

---

## I. GIỚI THIỆU

### 1.1 Nội dung và yêu cầu đề án

Sử dụng kỹ thuật lập trình socket và multithread trong C++ xây dựng ứng dụng chat client – server đơn giản theo yêu cầu đề án. Thực hiện các chức năng cơ bản:

1. Đăng ký tài khoản
2. Chat (gửi dữ liệu text) với một tài khoản chỉ định (thông qua Username)
3. Gửi tập tin bất kỳ
4. Chat nhóm

Yêu cầu nâng cao:

1. Hỗ trợ giao diện.
2. Chức năng gửi tập tin cho phép hiển thị thông tin tập tin nếu file gửi là hình ảnh.
3. Hỗ trợ âm thanh khi nhận được tin nhắn mới.

### 1.2 Phân tích đề án

#### 1.2.1 Gửi nhận dữ liệu sử dụng socket kết hợp multithread

Server tạo 1 socket nằm ở 1 luồng riêng biệt để lắng nghe khi nào có client mới kết nối. Với mỗi một client kết nối đến server thì server tạo ra 1 socket mới và 1 thread mới tương ứng để luôn chờ nhận gói tin và lắng nghe khi nào client không còn kết nối sẽ loại client đó khỏi danh sách. Khi nhận được gói tin thì thực hiện theo loại của gói tin yêu cầu.

#### 1.2.2 Chức năng đăng ký tài khoản

Client gửi username và password lên server với một dấu hiệu nhận biết. Server nhận được username và password sẽ kiểm tra trong cơ sở dữ liệu hiện tại. Nếu tài khoản tồn tại -> gửi mã đăng kí thất bại về client, nếu thành công thì cập nhật lại cơ sở dữ liệu cũng như danh sách client đang online và gửi mã đăng ký thành công về cho Client. Chức năng đăng nhập tương tự.

#### 1.2.3 Chức năng Chat với 1 người

Client gửi username của đối tượng muốn chat lên server, server kiểm tra user có online hay không. Nếu có thì gửi mã lệnh thành công về client - client nhận được sẽ tạo phòng chat 2 người, ngược lại gửi mã lệnh thất bại – client báo thông tin thất bại ra cho người dùng.

Trong mỗi hành động chat thì client gửi văn bản chat lên server, server sẽ gửi lại cho user bên kia văn bản.

Một trong 2 rời khỏi phòng chat thì gửi yêu cầu lên server, server gửi thông tin kết thúc quá trình chat cho user còn lại.

### 1.2.4 Chức năng Chat nhóm

Client gửi tên nhóm chat lên server, server kiểm tra user đó đã tạo nhóm chat và nhóm chat đang tồn tại không. Nếu đã tạo rồi thì gửi mã lệnh tạo nhóm thất bại về client và client thông báo ra cho người dùng. Nếu thành công thì server tạo 1 nhóm chat và thêm user gửi yêu cầu vào nhóm chat.

Khi user muốn thêm thành viên mới vào thì gửi username lên server, server kiểm tra username nếu online và chưa nằm trong nhóm chat đó thì thêm thành viên mới vào nhóm chat đồng thời gửi mã lệnh thêm thành công về client, ngược lại thì thất bại.

Khi 1 client bắt kì chat thì gửi lên server, server gửi cho toàn bộ client trong danh sách trong nhóm.

Khi một client rời nhóm chat sẽ gửi yêu cầu lên server, server xóa client khỏi danh sách trong nhóm chat. Nếu nhóm chat không còn thành viên thì xóa nhóm chat khỏi danh sách của server.

### 1.2.5 Chức năng gửi nhận gói tin

Client gửi tên tập tin, kích thước và username người nhận lên server, server gửi cho bên còn lại. Nếu bên kia chấp nhận nhận gói tin thì quá trình gửi tập tin bắt đầu. Gói tin được chia làm nhiều phần và gửi liên tục với 1 mã lệnh gửi tập tin, bên còn lại nhận và ghép lại thành tập tin đến khi nào đủ kích thước gói tin.

## 1.3 Thông tin nhóm

### 1.3.1 Thông tin thành viên


*Bảng 1.1. Bảng thông tin thành viên nhóm*

### 1.3.2 Phân công công việc

Họ và tên	Công việc	Mô tả
	Phân công các công việc. Hướng dẫn.	Phân công công việc và hướng dẫn các thành viên.
	Viết báo cáo	Viết file báo cáo, nộp đồ án.

HV Chương	Xây dựng và hiện thực mô hình client – server để truyền nhận gói tin.	Viết 2 class ChatClient và ChatServer để thực hiện truyền nhận gói tin và hỗ trợ các công việc trong đồ án thuộc về 2 lớp này. Hỗ trợ đa luồng để có thể gửi nhận gói tin bất kì. Bao gồm các chức năng: khởi tạo socket, kết nối server, gửi nhận tin nhắn, ... Ở server còn có kiểm tra user có online, kiểm tra group có online, đọc dữ liệu username password, kiểm tra đăng nhập - đăng ký, gửi tin cho toàn group, tổ chức dữ liệu các client, group...
	Ý tưởng và thiết kế cấu trúc, nội dung phần mềm và gói tin.	Thiết kế các cơ chế hoạt động của phần mềm. Thiết kế cấu trúc qui định bên trong gói tin. Đây là gói tin yêu cầu lệnh, đây là gói tin văn bản chat, đây là gói tin gửi nhận tập tin,... Như trong phần mô tả cấu trúc gói tin.
	Thiết kế giao diện các màn hình.	Thiết kế giao diện giữa các màn hình của ứng dụng
	Viết phần phân tích gói tin khi nhận từ server và thực hiện.	Nhận gói tin từ server ở class ChatClient và phân tích gói tin, thực hiện yêu cầu, hiển thị giao diện.
PH Dương	Viết các lệnh gửi đến server và xử lý server trả về client.	Viết các lệnh mà người dùng có thể tương tác yêu cầu đến server như đăng ký tài khoản, đăng nhập, tạo group chat, tạo chat với 1 người, gửi tin nhắn, gửi tập tin, ....
	Cài đặt chức năng gửi nhận file	Cài đặt chức năng gửi nhận file để gửi tập tin từ client này qua client kia thông qua server làm trung gian.
CDD Khoa	Viết class PrivateChat (chat 1 – 1 người) và xử lý.	Viết class PrivateChat để tạo 1 cửa sổ chat box Chat 1 người với 1 người. Bao gồm các chức năng gửi nhận dữ liệu, hiển thị dữ liệu, ...
	Viết class GroupChat (chat nhóm) và xử lý.	Viết class GroupChat để tạo 1 cửa sổ chat box Chat cho 1 nhóm. Bao gồm các chức năng thêm thành viên, gửi nhận dữ liệu, hiển thị dữ liệu, ...

Bảng 1.2. Bảng phân công các công việc cho các thành viên nhóm

## II. TỔNG QUAN MÔ HÌNH

### 2.1 Ý tưởng

Người dùng tương tác với phần mềm thông qua giao diện, từ giao diện ở 1 client cho biết được yêu cầu của người dùng thực hiện 1 chức năng nào đó. Với mỗi yêu cầu chức năng client sẽ gửi lên server 1 gói tin theo 1 định dạng qui định trước. Server dựa theo FLAG và nội dung gói tin để thực hiện yêu cầu và trả kết quả về cho client (nếu có).

Server ngoài việc xử lý yêu cầu còn quản lý việc client nào đã kết nối, đã ngắt kết nối; người dùng nào đã đăng ký, đăng nhập; nhóm chat nào đã được tạo,... Tất nhiên có áp dụng multi-threading vào cả client và server để thực hiện việc nhận và gửi tin đồng thời bởi các luồng riêng biệt.

### 2.2 Gói tin gửi nhận

#### 2.2.1 Cấu trúc gói tin

Gói tin được gửi là 1 chuỗi char\* và được chia làm các phần:

FLAG	Nội dung 1	NULL ('0')	Nội dung 2	NULL ('\0')
------	------------	------------	------------	-------------

Chuỗi nội dung chính là chuỗi nội dung client muốn server xử lý và thực hiện, chuỗi nội dung phụ là chuỗi nội dung kèm theo để server có thể biết client muốn thực hiện trên đâu.

Khi server nhận được gói tin sẽ biết client đang muốn gì và ngược lại client sẽ biết server yêu cầu hoặc phản hồi gì.

Ví dụ gói tin yêu cầu thêm một user có username là “viet” vào group chat tên là “MMT”:

GC_ADD_USER (1 ký tự có mã là 27)	'M'	'M'	'T'	NULL ('0')	'v'	'i'	'e'	't'	NULL ('\0')
--------------------------------------	-----	-----	-----	------------	-----	-----	-----	-----	-------------

Trong đó:

- **FLAG:** GC\_ADD\_USER còn nhận biết thêm user vào nhóm chat
- **Nội dung 1:** “MMT” là tên nhóm chat cần thêm vào
- **Nội dung 2:** “viet” là tên user muốn thêm vào

#### 2.2.2 Các loại gói tin

Tất cả các loại (FLAG) thuộc kiểu enum với tên là MessageType

FLAG	Nội dung theo sau	Ý nghĩa
PRIVATE_CHAT	receiver   NULL   sender   NULL   content   NULL	Tin nhắn 1 – 1 người

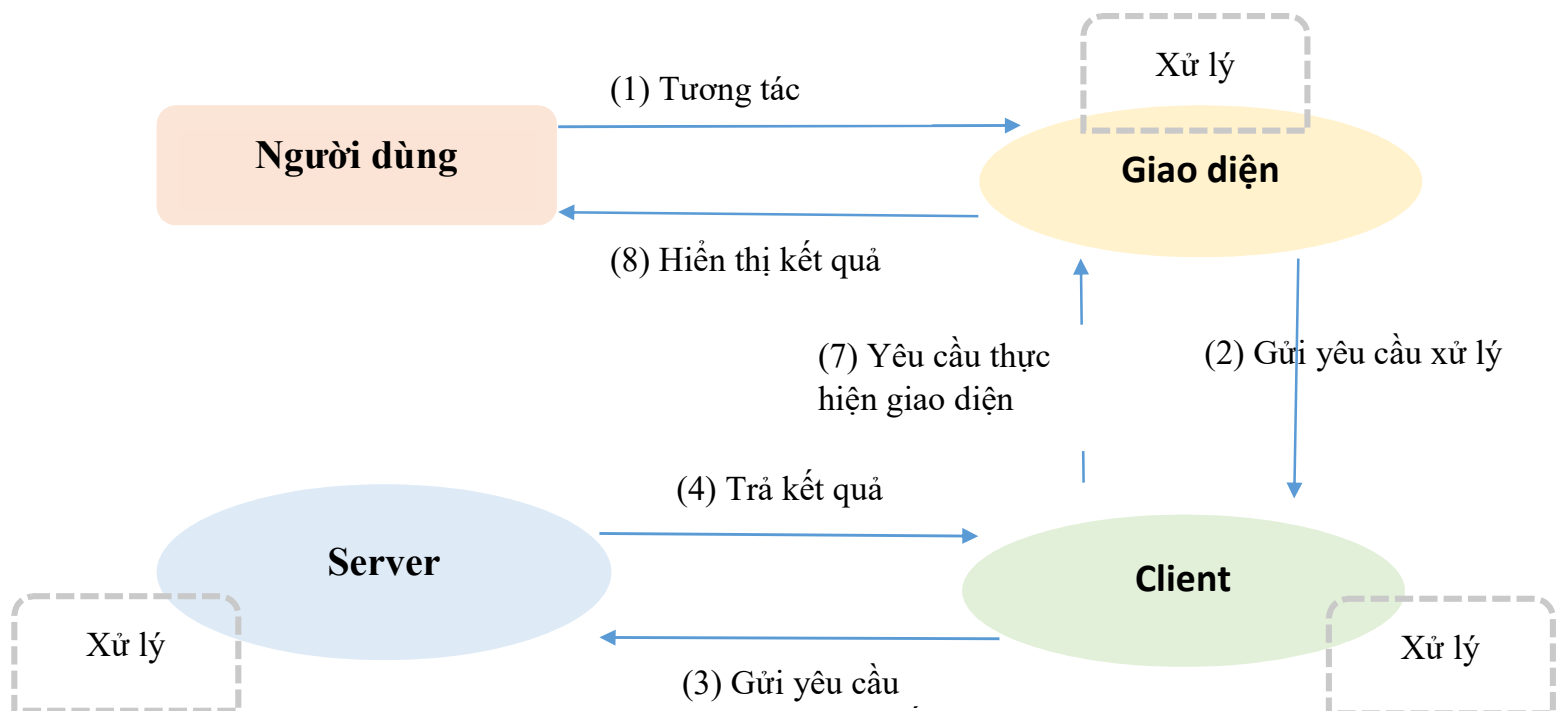
END_PRIVATE_CHAT	receiver   NULL   sender   NULL	Kết thúc cuộc trò chuyện 1 – 1 người
GROUP_CHAT	group name   NULL   sender   NULL   content   NULL	Tin nhắn của chat nhóm
END_GROUP_CHAT	group name   NULL   sender   NULL	Kết thúc cuộc trò chuyện nhóm
SIGNUP	user name   NULL   password   NULL	Đăng ký tài khoản
SU_SUCCESS		Đăng ký thành công
SU_FAILURE		Đăng ký thất bại
LOGIN	user name   NULL   password   NULL	Đăng nhập
LI_SUCCESS		Đăng nhập thành công
LI_FAILURE		Đăng nhập thất bại
CREATE_PRIVATE_CHAT	partner   NULL	Tạo cuộc trò chuyện 1 - 1
C_PC_SUCCESS	partner   NULL	Tạo cuộc trò chuyện 1 – 1 thành công
C_PC_FAILURE		Tạo cuộc trò chuyện 1 – 1 thất bại
CREATE_GROUP_CHAT	group name   NULL	Tạo cuộc chat nhóm
C_GC_SUCCESS	sender   NULL	Tạo cuộc chat nhóm thành công
C_GC_FAILURE		Tạo cuộc chat nhóm thất bại
GC_ADD_USER	group name   NULL   added user   NULL	Thêm thành viên vào chat nhóm
GC_AU_SUCCESS	group name   NULL   added user   NULL	Thêm thành viên vào chat nhóm thành công
GC_AU_FAILURE	group name   NULL   added user   NULL	Thêm thành viên vào chat nhóm thất bại
SEND_FILE	file name   NULL   file size   NULL   receiver   NULL   sender   NULL	Yêu cầu gửi file đến user khác



SF_ACCEPT	file name   NULL   file size   NULL   sender   NULL   receiver   NULL	Chấp nhận nhận file từ user được yêu cầu
SF_CANCEL	file name   NULL   file size   NULL   sender   NULL   receiver   NULL	Từ chối nhận file
FILE_DATA	file size   NULL   receiver   NULL   sender   NULL   content	Nội dung file được chia nhỏ
CONTINUE	receiver   NULL   sender   NULL	Yêu cầu user gửi file gửi tiếp tục nội dung file
STOP	receiver   NULL   sender   NULL	Nhận file thành công, yêu cầu dừng gửi

Bảng 2.1 Bảng nội dung các loại gói tin

## 2.3 Mô hình



Mô hình giao tiếp các đối tượng của phần mềm

---

## III. CÀI ĐẶT CÁC YÊU CẦU ĐỒ ÁN

### 3.1 Đăng ký tài khoản

#### 3.1.1 Các lớp, đối tượng sử dụng

##### a. Class ChatClient

Đối tượng toàn cục **gClientObj** để thực hiện việc gửi nhận gói tin đến server.

##### b. Class ChatServer

Đối tượng toàn cục **gServerObj** để thực hiện việc gửi nhận gói tin đến client. Đăng ký tài khoản cho client mới cũng như quản lý cơ sở dữ liệu các người dùng đã đăng ký.

#### 3.1.2 Các hàm xử lý

##### a. Ở Client

- **LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

*Tham số:* Tham số cơ bản của hàm Callback của Win32

*Ý nghĩa:*

1. Bắt message khi người dùng nhấn vào button “Sign Up” thông qua **WM\_COMMAND** với ID của button là **IDC\_SIGNUP**. Lấy username và password ở 2 ô edittext “Username:” – “Password:” kiểm tra hợp lệ và tạo 1 gói tin yêu cầu đăng ký gửi lên server với FLAG là **SIGN\_UP** bằng hàm **sendMessagePort**.
2. Nhận kết quả trả về thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. wParam là con trỏ trỏ tới tin nhắn nhận được. Kết quả trả về có thể là: **SU\_SUCCESS** (đăng ký thành công) hoặc **SU\_FAILURE** (đăng ký thất bại) -> hiện thông báo người dùng

- **int sendMessagePort(WCHAR\* message, int len);**

*Tham số:*

1. **message:** Gói tin cần gửi
2. **len:** độ dài gói tin

*Ý nghĩa:* Gửi gói tin đến server. Được yêu cầu từ WndProc thông qua gClientObj.

- **int recMessagePort();**

*Ý nghĩa:* Nhận gói tin trả lời từ server và gửi message đến WndProc để xử lý

##### b. Ở Server

- **int recClient(SOCKET recSocket);**

**Tham số: recSocket:** là socket nhận được tin nhắn.

**Ý nghĩa:** Nhận tin nhắn từ Client và dựa là FLAG để xử lý. Ở đây là đăng ký (FLAG là SIGN\_UP) thì sử dụng phương thức **signUp** để kiểm tra và đăng ký. Sau đó dùng phương thức **sendMessageClient** để gửi trả lời về client.

▪ **bool signUp(User\* user);**

**Tham số: user:** user cần đăng ký

**Ý nghĩa:** Tìm trong danh sách user của server đã tồn tại chưa. Nếu chưa thì thêm user mới vào trả về true, ngược lại trả về false.

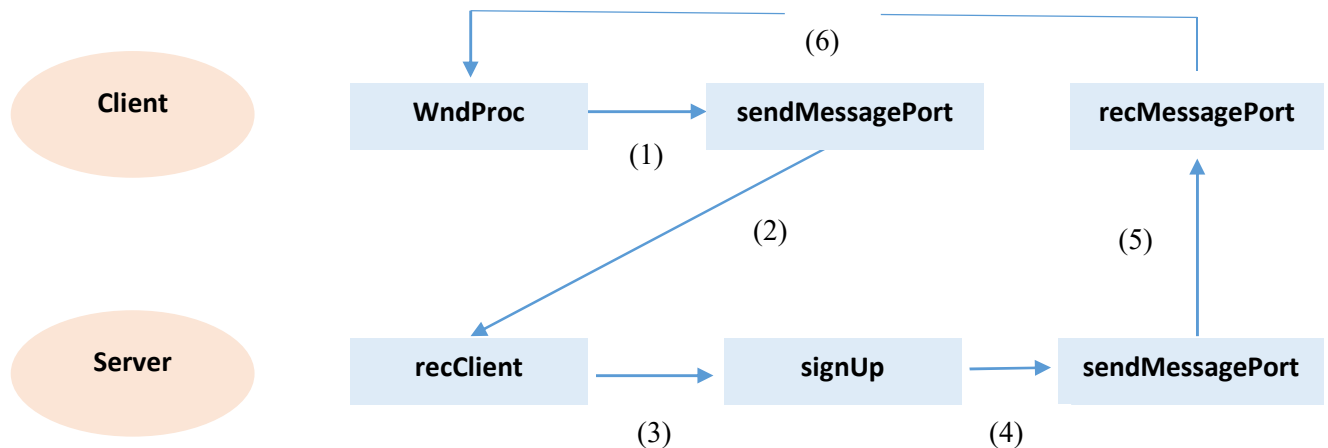
▪ **int sendMessageClient(ClientPacket\* client, WCHAR\* message, int len);**

**Tham số:**

1. **client:** client cần gửi gói tin trả lời đến.
2. **message:** gói tin phản hồi đăng ký thành công hay thất bại.
3. **len:** độ dài gói tin.

**Ý nghĩa:** Gửi gói tin phản hồi đăng ký thành công hay thất bại đến client.

### 3.1.3 Sơ đồ gọi hàm



---

## 3.2 Chat với 1 người (private chat)

### 3.2.1 Các lớp, đối tượng sử dụng

#### a. Class ChatClient

Đối tượng toàn cục **gClientObj** để thực hiện việc gửi nhận gói tin yêu cầu chat, tin nhắn chat đến server.

#### b. Class PrivateChat

Quản lý cửa sổ chat đối với chức năng chat với 1 người. Biến cục bộ sử dụng được chứa trong 1 list các hộp thoại private chat: **gPrivateChat**

#### c. Class ChatServer

Đối tượng toàn cục **gServerObj** để thực hiện việc gửi nhận gói tin đến client. Nhận lệnh tạo cuộc hội thoại, kiểm tra user có online không và là trung gian nhận gửi tin nhắn giữa 2 user.

### 3.2.2 Các hàm xử lý

#### a. Ở Client

- **LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

*Tham số:* Tham số cơ bản của hàm Callback của Win32

*Ý nghĩa:*

1. Bắt message khi người dùng nhấn vào button “Create” thông qua **WM\_COMMAND** với ID của button là **IDC\_CREATE**. Lấy ô edittext “Username:” kiểm tra hợp lệ và tạo 1 gói tin yêu cầu chat 1-1 gửi lên server với FLAG là **CREATE\_PRIVATE\_CHAT** bằng hàm **sendMessagePort**.
2. Nhận kết quả tạo chat 1-1 trả về thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ tới tin nhắn nhận được. Kết quả trả về có thể là: **C\_PC\_SUCCESS** (tạo chat 1-1 thành công, người dùng có online) hoặc **C\_PC\_FAILURE** (tạo chat 1-1 thất bại, người dùng không tồn tại hoặc offline) -> hiện thông báo người dùng. Đối với tạo chat 1-1 thành công nếu chưa có cửa sổ chat thì tạo cửa sổ chat mới, còn có rồi thì hiện thị tin nhắn vào cửa sổ đó.
3. Nhận tin nhắn chat thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ tới tin nhắn nhận được. Kết quả trả về là gói tin có FLAG: **PRIVATE\_CHAT**. Tìm đến ChatBox nhận tin nhắn và hiển thị tin nhắn.

4. Nhận tin có user bên kia đã ngừng chat thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ trỏ tới tin nhắn nhận được. Kết quả trả về là gói tin có FLAG: **END\_PRIVATE\_CHAT** và hiển thị user bên kia đã ngừng chat đồng thời kết thúc cuộc trò chuyện.

▪ **LRESULT CALLBACK ChatBoxProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

**Tham số:** Tham số cơ bản của hàm Callback của Win32

**Ý nghĩa:**

1. Bắt message khi người dùng nhấn vào button “Send” thông qua **WM\_COMMAND** với ID của button là **IDC\_CREATE**. Lấy nội dung ô edittext – nội dung tin nhắn, kiểm tra hợp lệ và tạo 1 gói tin yêu cầu gửi tin nhắn đến user bên kia và gửi lên server với FLAG là **PRIVATE\_CHAT** bằng hàm **sendMessagePort**.
2. Bắt sự kiện **WM\_DESTROY** khi người dùng đóng chatbox. Gửi gói tin ngừng chat cho user bên kia thông qua **sendMessagePort** với gói tin có FLAG **END\_PRIVATE\_CHAT**.

▪ **int sendMessagePort(WCHAR\* message, int len);**

**Tham số:**

1. **message:** Gói tin cần gửi
2. **len:** gói tin tin nhắn

**Ý nghĩa:** Gửi gói tin đến server. Được yêu cầu từ WndProc thông qua gClientObj.

▪ **int recMessagePort();**

**Ý nghĩa:** Nhận gói tin từ server và gửi message đến WndProc để xử lý.

▪ **Nhóm hàm tạo hộp thoại (chatbox) mới**

1. **static PrivateChatBox\* create(HWND hParent, HINSTANCE hInst, Point pos, Size size, wstring partner);**

**Tham số:** **hParent:** handle cửa sổ cha

**hInst:** Instance

**pos:** vị trí

**size:** kích thước;

**partner:** username của người đang chat.

**Ý nghĩa:** Tạo 1 cửa sổ chat mới và thêm vào danh sách các chatbox

---

2. **void setFont(HFONT hFont);**

**Tham số: hFont:** handle của font cần đặt

**Ý nghĩa:** Đổi font cho các đối tượng trong chatbox.

3. **void setUsername(const wstring& username);**

**Tham số: username:** username của người partner đang chat.

**Ý nghĩa:** Thiết lập tên username của partner đang chat.

**b. Ở Server**

▪ **int recClient(SOCKET recSocket);**

**Tham số: recSocket:** là socket nhận được tin nhắn.

**Ý nghĩa:**

1. Nhận gói tin yêu cầu tạo chat 1-1. Kiểm tra user cần chat có online hay tồn tại hay không. Nếu có thì gửi tin nhắn có FLAG **C\_PC\_SUCCESS** về client ngược lại gửi tin có FLAG **C\_PC\_FAILURE** thông qua hàm **sendMessageClient**.
2. Nhận gói tin chat 1-1 giữa 2 client. Dùng hàm **sendMessagePort** để gửi gói tin tin nhắn đến user bên kia.
3. Nhận gói tin kết thúc chat của 1 user (**END\_PRIVATE\_CHAT**) và gửi yêu cầu đến user còn lại.

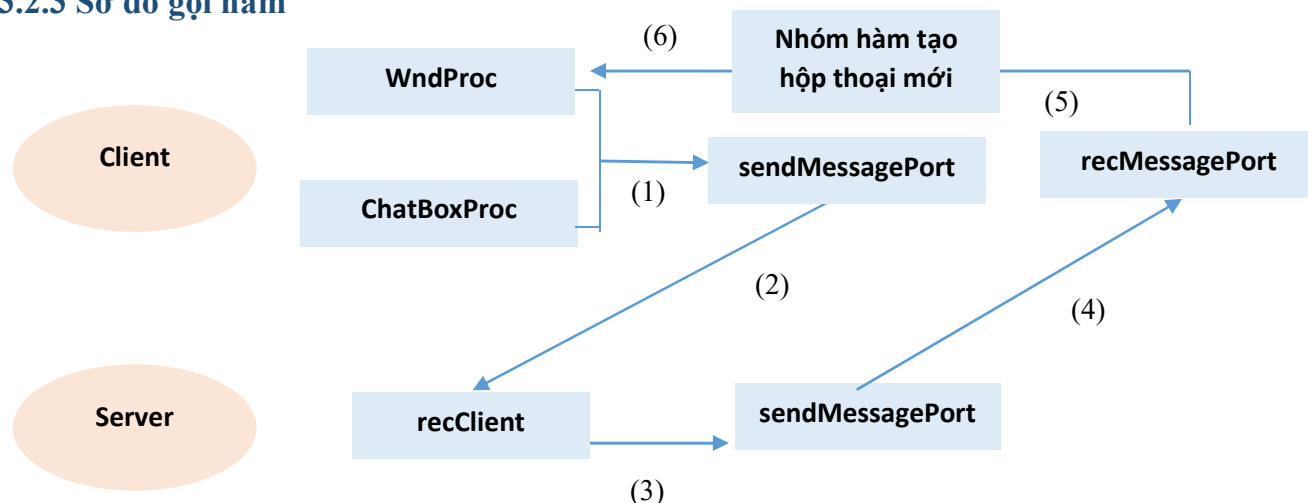
▪ **int sendMessageClient(ClientPacket\* client, WCHAR\* message, int len);**

**Tham số:**

1. **client:** client cần gửi gói tin trả lời đến.
2. **message:** gói tin phản hồi tạo chat 1-1 thành công hoặc gói tin chứa tin nhắn.
3. **len:** độ dài gói tin.

**Ý nghĩa:** Gửi gói tin phản hồi chat 1-1 thành công hoặc thất bại về client hoặc gửi gói tin chứa tin nhắn đến client được nhận.

### 3.2.3 Sơ đồ gọi hàm



## 3.3 Chat với nhóm (group chat)

### 3.3.1 Các lớp, đối tượng sử dụng

#### a. Class ChatClient

Đối tượng toàn cục **gClientObj** để thực hiện việc gửi nhận gói tin yêu cầu chat, tin nhắn chat đến server.

#### b. Class GroupChat

Quản lý cửa sổ chat đối với chức năng chat nhóm. Biến cục bộ sử dụng được chứa trong 1 list các hộp thoại group chat: **gGroupChat**

#### c. Class ChatServer

Đối tượng toàn cục **gServerObj** để thực hiện việc gửi nhận gói tin đến client. Nhận lệnh tạo cuộc hội thoại, kiểm tra user có online không và là trung gian nhận gửi tin nhắn giữa 2 user.

### 3.3.2 Các hàm xử lý

#### a. Ở Client

- **LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

**Tham số:** Tham số cơ bản của hàm Callback của Win32

**Ý nghĩa:**

1. Bắt message khi người dùng nhấn vào button “Create” thông qua **WM\_COMMAND** với ID của button là **IDC\_CREATE**. Lấy ô edittext “Group name:” kiểm tra hợp lệ và tạo 1 gói tin yêu cầu chat nhóm gửi lên server với FLAG là **CREATE\_GROUP\_CHAT** bằng hàm **sendMessagePort**.
  2. Nhận kết quả tạo chat nhóm trả về thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ tới tin nhắn nhận được. Kết quả trả về có thể là: **C\_GC\_SUCCESS** (tạo chat nhóm thành công, người dùng có online) hoặc **C\_PGC\_FAILURE** (tạo chat nhóm thất bại, người dùng không tồn tại hoặc offline) -> hiện thông báo người dùng. Đối với tạo chat nhóm thành công nếu chưa có cửa sổ chat thì tạo cửa sổ chat mới, còn có rồi thì hiện thị tin nhắn vào cửa sổ đó.
  3. Nhận tin nhắn chat thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ tới tin nhắn nhận được. Kết quả trả về là gói tin có FLAG: **GROUP\_CHAT**. Tìm đến ChatBox nhận tin nhắn và hiển thị tin nhắn.
  4. Nhận tin nhắn thêm user thành công thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ tới tin nhắn nhận được. Kết quả trả về là gói tin có FLAG: **GC\_AU\_SUCCESS** (thêm user thành công) hoặc **GC\_AU\_FAILURE** (thêm user thất bại) -> hiển thị kết quả.
  5. Nhận tin nhắn thêm user thành công thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ tới tin nhắn nhận được. Gói tin có FLAG **GC\_ADD\_USER** cho biết user nào mới được thêm vào -> hiển thị lên giao diện.
  6. Nhận tin nhắn có user rời khỏi nhóm chat thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ tới tin nhắn nhận được. Gói tin có FLAG **END\_GROUP\_CHAT** và hiển thị user rời khỏi nhóm lên chatbox.
- **LRESULT CALLBACK ChatBoxProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

**Tham số:** Tham số cơ bản của hàm Callback của Win32

**Ý nghĩa:**

1. Bắt message khi người dùng nhấn vào button “Send” thông qua **WM\_COMMAND** với ID của button là **IDC\_CREATE**. Lấy nội dung ô edittext – nội dung tin nhắn, kiểm tra hợp lệ và tạo 1 gói tin yêu cầu gửi tin



---

nhấn đến user bên kia và gửi lên server với FLAG là **GROUP\_CHAT** bằng hàm **sendMessagePort**.

2. Bắt message khi người dùng nhấn vào button “Add user” thông qua **WM\_COMMAND** với ID của button là **IDC\_CREATE**. Lấy nội dung ô edittext – ô tên user, kiểm tra hợp lệ và tạo 1 gói tin yêu cầu thêm user gửi đến lên server với FLAG là **GC\_ADD\_USER** bằng hàm **sendMessagePort**.
3. Bắt sự kiện **WM\_DESTROY** khi người dùng tắt hộp thoại -> kết thúc chat nhóm. Gửi gói tin có FLAG **END\_GROUP\_CHAT** đến tất cả user trong nhóm để thông báo user nào rời khỏi nhóm.

▪ **int sendMessagePort(WCHAR\* message, int len);**

*Tham số:*

1. **message:** Gói tin cần gửi
2. **len:** gói tin tin nhắn

*Ý nghĩa:* Gửi gói tin đến server. Được yêu cầu từ WndProc thông qua gClientObj.

▪ **int recMessagePort();**

*Ý nghĩa:* Nhận gói tin từ server và gửi message đến WndProc để xử lý.

▪ **Nhóm hàm tạo hộp thoại (chatbox) mới**

1. **static GroupChatBox\* create(HWND hParent, HINSTANCE hInst, Point pos, Size size, wstring groupname);**

*Tham số:* **hParent:** handle cửa sổ cha

**hInst:** Instance

**pos:** vị trí

**size:** kích thước;

**groupname:** tên của nhóm chat.

*Ý nghĩa:* Tạo 1 cửa sổ chat mới và thêm vào danh sách các chatbox:

**gGroupChatBoxList**

2. **void setFont(HFONT hFont);**

*Tham số:* **hFont:** handle của font cần đặt

*Ý nghĩa:* Đổi font cho các đối tượng trong chatbox.

3. **void setUsername(const wstring& username);**

*Tham số:* **username:** username người chủ đang mở chatbox.

*Ý nghĩa:* Thiết lập username của chủ đang mở chatbox.

---

## b. Ở Server

### ▪ **int recClient(SOCKET recSocket);**

**Tham số: recSocket:** là socket nhận được tin nhắn.

**Ý nghĩa:**

1. Nhận gói tin yêu cầu tạo chat nhóm (**CREATE\_GROUP\_CHAT**). Kiểm tra nhóm chat online hay tồn tại hay không. Nếu có thì gửi tin nhắn có **FLAG C\_GC\_SUCCESS** về client ngược lại gửi tin có **FLAG C\_GC\_FAILURE** thông qua hàm **sendMessageClient**.
2. Nhận gói tin chat nhóm (**GROUP\_CHAT**). Dùng hàm **sendMessageGroup** để gửi gói tin tin nhắn đến các user trong nhóm.
3. Nhận gói tin yêu cầu thêm user vào nhóm (**GC\_ADD\_USER**). Kiểm tra user có online hay tồn tại hay có trong nhóm chưa, nếu thỏa mãn để thêm vào thì thêm user vào danh sách và gửi gói tin đã thêm user đến các user khác trong nhóm (**GC\_ADD\_USER**). Gửi **FLAG GC\_AU\_SUCCESS** (thêm user thành công) hoặc **GC\_AU\_FAILURE** (thêm user thất bại) về client yêu cầu.
4. Nhận gói tin yêu cầu rời khỏi nhóm chat (**END\_GROUP\_CHAT**). Xóa user khỏi danh sách trong nhóm và gửi gói tin user rời nhóm đến tất cả các user khác trong nhóm.

### ▪ **int sendMessageClient(ClientPacket\* client, WCHAR\* message, int len);**

**Tham số:**

1. **client:** client cần gửi gói tin trả lời đến.
2. **message:** gói tin phản hồi tạo chat 1-1 thành công hoặc gói tin chứa tin nhắn.
3. **len:** độ dài gói tin.

**Ý nghĩa:** Gửi gói tin phản hồi chat nhóm thành công hoặc thất bại về client hoặc gửi gói tin chứa tin nhắn đến client được nhận.

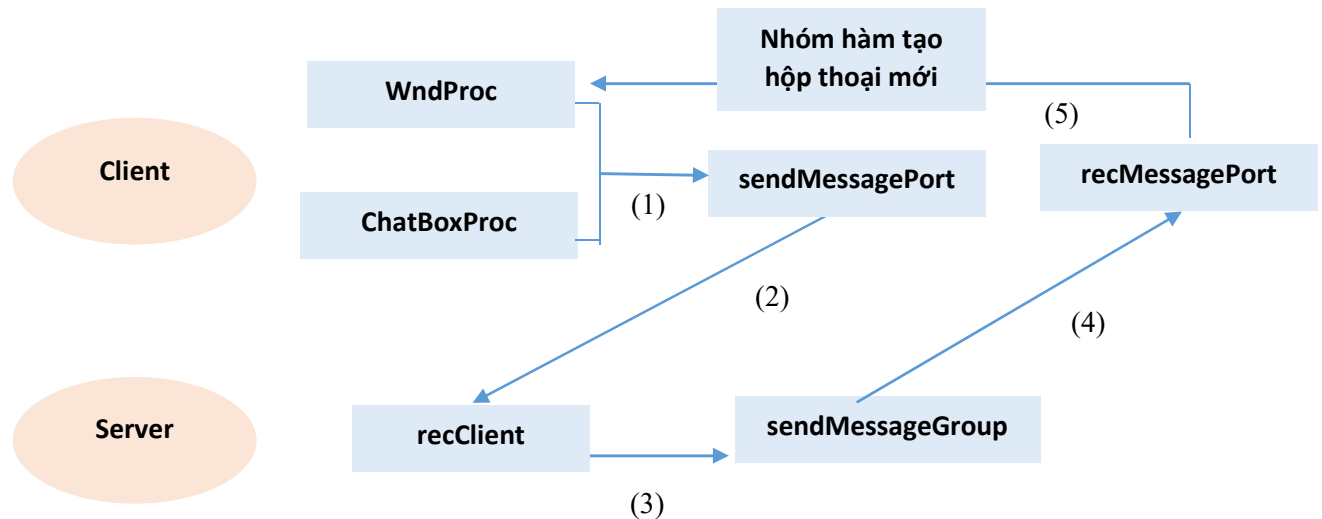
### ▪ **void sendMessageGroup(wstring groupname, wstring sender, WCHAR\* message, int len, bool isSendToSender = false);**

**Tham số:**

1. **groupname:** tên nhóm chat cần gửi tin nhắn
2. **sender:** username người gửi
3. **message:** nội dung gói tin
4. **isSendToSender:** Có gửi gói tin đến người gửi hay không (mặc định là không)

**Ý nghĩa:** Gửi tin nhắn đến toàn nhóm.

### 3.3.3 Sơ đồ gọi hàm



## 3.4 Gửi tập tin

### 3.4.1 Các lớp, đối tượng sử dụng

#### a. Class ChatClient

Đối tượng toàn cục **gClientObj** để thực hiện việc gửi nhận gói tin yêu cầu chat, tin nhắn chat đến server.

#### b. Class PrivateChat

Quản lý cửa sổ chat đối với chức năng chat với 1 người. Biến cục bộ sử dụng được chứa trong 1 list các hộp thoại private chat: **gPrivateChat**. Ở chức năng gửi file chỉ áp dụng với chat 1-1 nên sử dụng class **PrivateChat**.

#### c. Class ChatServer

Đối tượng toàn cục **gServerObj** để thực hiện việc gửi nhận gói tin đến client. Nhận lệnh tạo cuộc hội thoại, kiểm tra user có online không và là trung gian nhận gửi tin nhắn giữa 2 user.

### 3.2.2 Các hàm xử lý

Chức năng gửi tập tin chỉ có thể gửi 1 tập tin đến 1 user tại 1 thời điểm (không thể gửi nhiều tập tin đến 1 user cùng lúc).

#### a. Ở Client

---

- **LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

**Tham số:** Tham số cơ bản của hàm Callback của Win32

**Ý nghĩa:**

1. Nhận gói tin yêu cầu gửi file thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ trỏ tới tin nhắn nhận được. Kết quả trả về là gói tin có FLAG: **SEND\_FILE**. Tìm đến ChatBox nhận tin nhắn và hiển thị yêu cầu chấp nhận gửi nhận file hay không. Nếu người dùng chấp nhận thì gọi hàm **myCreateSaveFile** để chọn nơi lưu file, gọi hàm **preReceiveFile** để tạo file và gửi lại gói tin **SF\_ACCEPT** đến user bên kia để bắt đầu quá trình gửi tập tin, ngược lại gửi gói **SF\_CANCEL** để dừng.
2. Nhận gói tin **SF\_ACCEPT** thì bắt đầu gửi phần đầu tiên của gói tin đến user bên kia.
3. Nhận gói tin **SF\_CANCEL** thì gọi hàm **onRefuseReceiveFile** đóng file và dừng việc gửi gói tin.
4. Nhận gói tin là dữ liệu của 1 file thông qua **recMessagePort** và gửi message đến giao diện với **WM\_COMMAND** và ID là **IDC\_RECEIVE**. **wParam** là con trỏ trỏ tới tin nhắn nhận được. Gói tin có FLAG **FILE\_DATA**. Đọc gói tin và ghi dữ liệu nhận được vào file. Nếu ghi đủ thì đóng file và gửi gói tin có FLAG **STOP** đến user bên kia để dừng gửi, ngược lại gửi gói **CONTINUE** để yêu cầu tiếp tục gửi.
5. Nhận gói tin **CONTINUE** thì gửi phần tiếp theo của gói tin đến user bên kia. Dùng hàm **onSendFile** để tiếp tục đọc tập tin.
6. Nhận gói tin **STOP** thì dừng việc gửi gói tin và gọi **onStop** để đóng file.

- **LRESULT CALLBACK ChatBoxProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

**Tham số:** Tham số cơ bản của hàm Callback của Win32

**Ý nghĩa:**

1. Bắt message khi người dùng nhấn vào button “Attach” thông qua **WM\_COMMAND** với ID của button là **IDC\_CREATE**. Gọi hàm **myCreateOpenFile** để yêu cầu người dùng chọn gói tin cần gửi, gọi hàm **getSizeAndOpenFile** để mở tập tin và đọc kích thước để tạo gói tin **SEND\_FILE** và gửi đến user bên kia bằng hàm **sendMessagePort**.

- 
- **int sendMessagePort(WCHAR\* message, int len);**

*Tham số:*

1. **message:** Gói tin cần gửi
2. **len:** gói tin tin nhắn

*Ý nghĩa:* Gửi gói tin đến server. Được yêu cầu từ WndProc thông qua gClientObj.

- **int recMessagePort();**

*Ý nghĩa:* Nhận gói tin từ server và gửi message đến WndProc để xử lý.

- **DWORD getSizeAndOpenFile(WCHAR\* filename);**

*Tham số: filename:* tên tập tin cần mở và lấy kích thước.

*Ý nghĩa:* mở trước tập tin và lấy kích thước.

- **void preReceiveFile(WCHAR\* filename, DWORD filesize);**

*Tham số: filename:* tên tập tin cần mở để ghi.

**Filesize:** kích thước gói tin sẽ nhận.

*Ý nghĩa:* mở trước tập tin và lưu kích thước sẽ nhận.

- **void onAcceptReceiveFile();**

*Ý nghĩa:* Đóng tập tin khi user bên kia không chấp nhận nhận gói tin.

- **void onStop();**

*Ý nghĩa:* Đóng tập tin khi kết thúc quá trình gửi gói tin.

- **DWORD onSendFile(WCHAR\* message);**

*Tham số: message:* buffer nhận được sau khi tạo gói tin từ việc tiếp tục đọc từ file

*Ý nghĩa:* Tạo 1 gói tin từ việc đọc tiếp tục file đang gửi.

- **BOOL myCreateOpenFile(HWND hwnd, WCHAR\* filename);**

*Tham số: hwnd:* handle cửa sổ yêu cầu

**filename:** tên tập tin trả về.

*Ý nghĩa:* Mở hộp thoại mở file để người dùng chọn file cần gửi.

- **BOOL myCreateSaveFile(HWND hwnd, WCHAR\* filename);**

*Tham số: hwnd:* handle cửa sổ yêu cầu

---

**filename:** tên tập tin mặc định hiển thị và cũng là tên tập tin trả về.

**Ý nghĩa:** Mở hộp thoại lưu file để người dùng chọn nơi lưu và tên file lưu.

▪ **Nhóm hàm tạo hộp thoại (chatbox) mới**

1. **static PrivateChatBox\* create(HWND hParent, HINSTANCE hInst, Point pos, Size size, wstring partner);**

**Tham số:** **hParent:** handle cửa sổ cha

**hInst:** Instance

**pos:** vị trí

**size:** kích thước;

**partner:** username của người đang chat.

**Ý nghĩa:** Tạo 1 cửa sổ chat mới và thêm vào danh sách các chatbox

2. **void setFont(HFONT hFont);**

**Tham số:** **hFont:** handle của font cần đặt

**Ý nghĩa:** Đổi font cho các đối tượng trong chatbox.

3. **void setUsername(const wstring& username);**

**Tham số:** **username:** username của người partner đang chat.

**Ý nghĩa:** Thiết lập tên username của partner đang chat.

**b. Ở Server**

▪ **int recClient(SOCKET recSocket);**

**Tham số:** **recSocket:** là socket nhận được tin nhắn.

**Ý nghĩa:**

1. Nhận gói tin yêu cầu gửi file **SEND\_FILE**. Gửi lại gói tin cho user cần nhận.
2. Nhận gói tin **CONTINUE, CANCEL, STOP, FILE\_DATA**. Gửi lại gói tin cho user cần nhận.

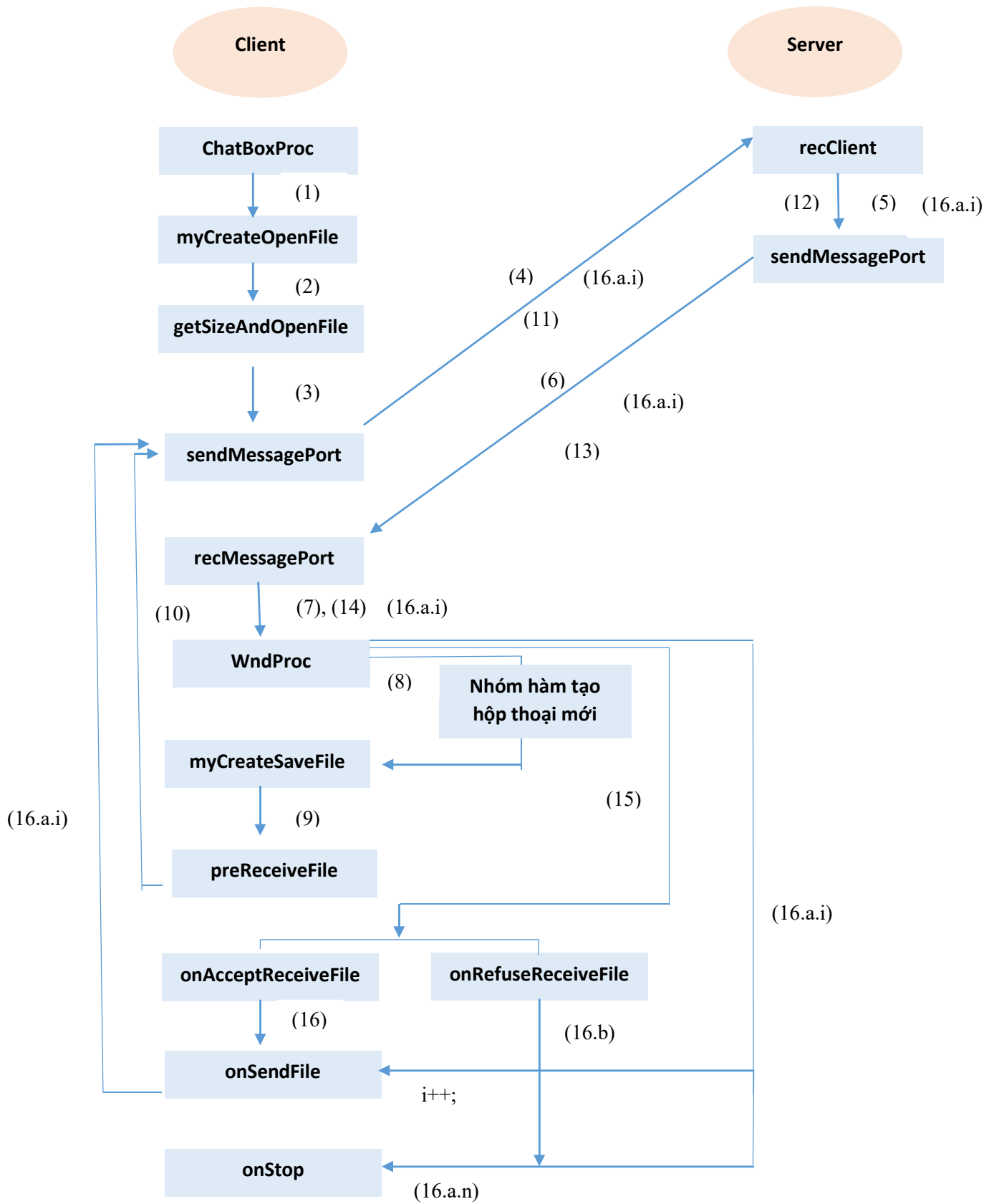
▪ **int sendMessageClient(ClientPacket\* client, WCHAR\* message, int len);**

**Tham số:**

1. **client:** client cần gửi gói tin trả lời đến.
2. **message:** gói tin phản hồi tạo chat 1-1 thành công hoặc gói tin chứa tin nhắn.
3. **len:** độ dài gói tin.

**Ý nghĩa:** Gửi gói tin đến user theo yêu cầu.

### 3.4.3 Sơ đồ gọi hàm

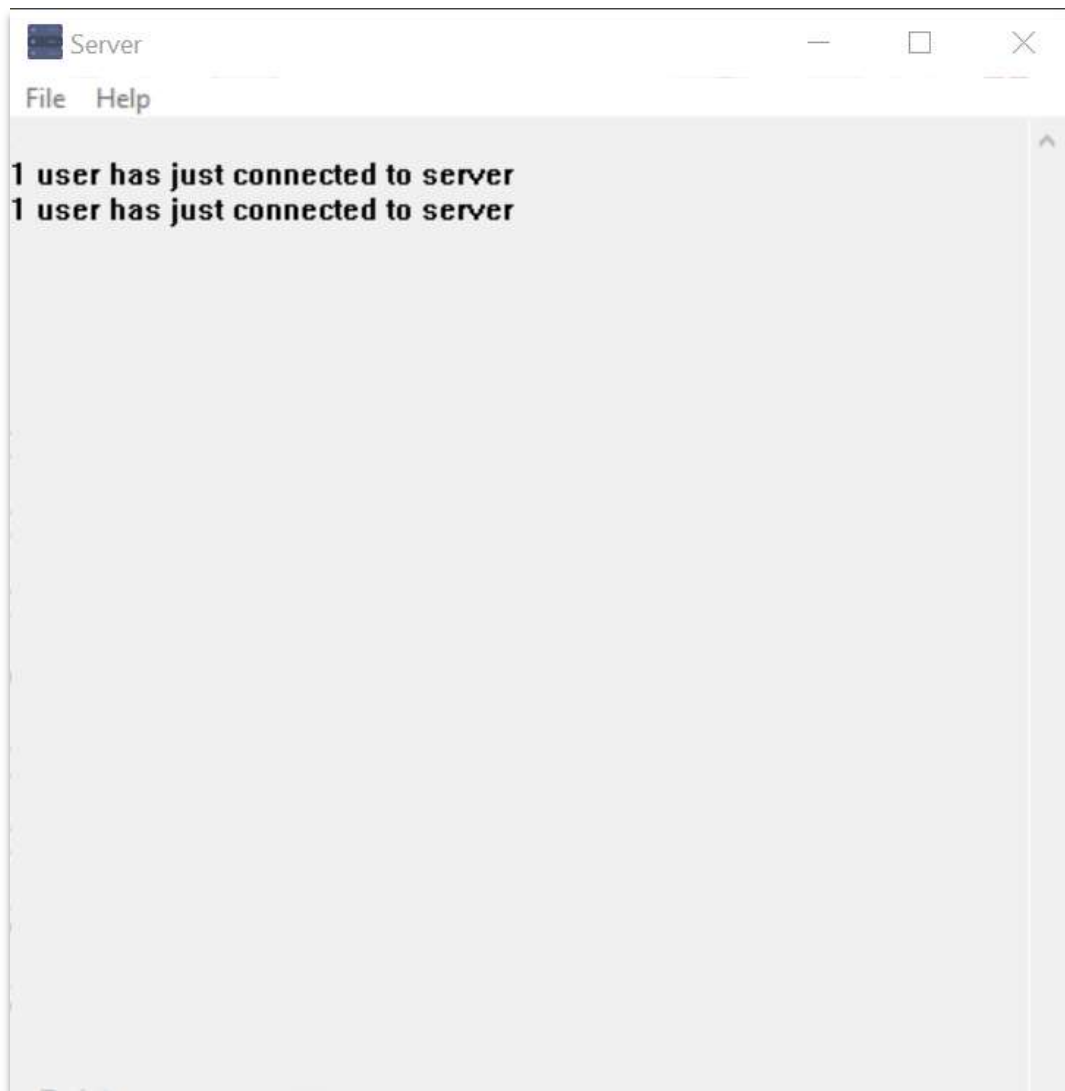


### 3.5 Yêu cầu nâng cao

- Hỗ trợ giao diện Win32 .
- Khi nhận được tập tin có gửi hộp thoại hỏi người dùng có muốn mở không. Nếu người dùng chọn có thì mở tập tin bằng 1 bên thứ 3 (không quan tâm vấn đề tập tin hình ảnh).
- Khi nhận được tin nhắn có phát âm thanh thông báo.

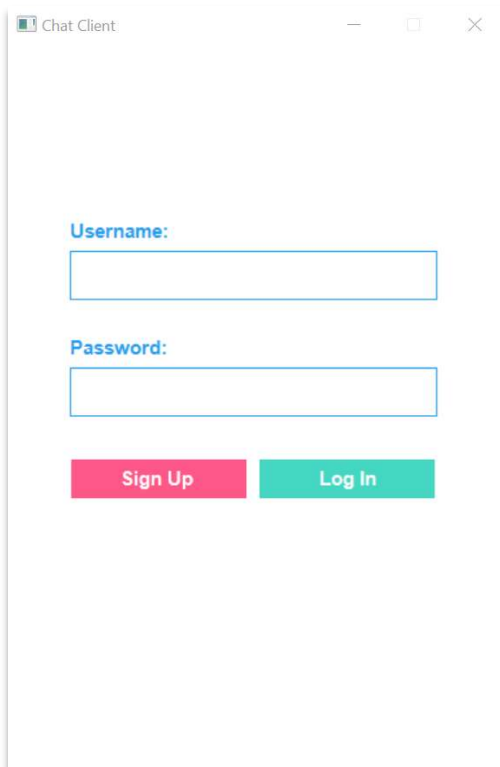
## IV. GIAO DIỆN MÀN HÌNH - DEMO

### 4.1 Hình ảnh



*Hình 4.1 Màn hình của Chat server khi có 2 user vừa kết nối*





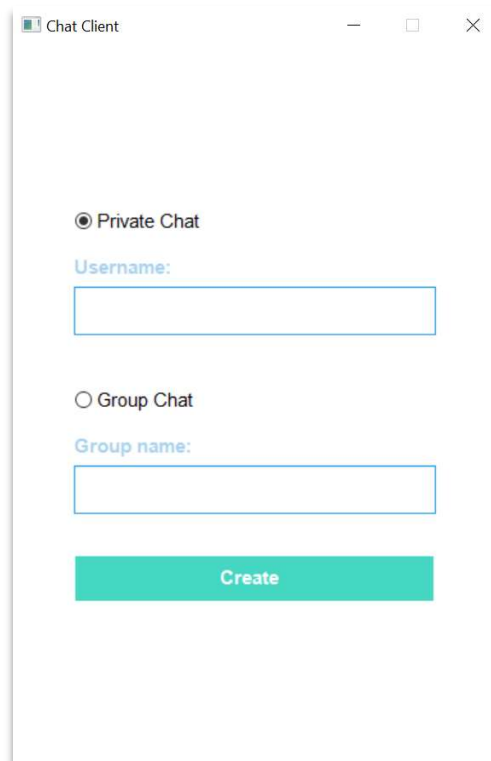
Chat Client

Username:

Password:

Sign Up Log In

Hình 4.2 Màn hình đăng nhập, đăng ký



Chat Client

☒ Private Chat

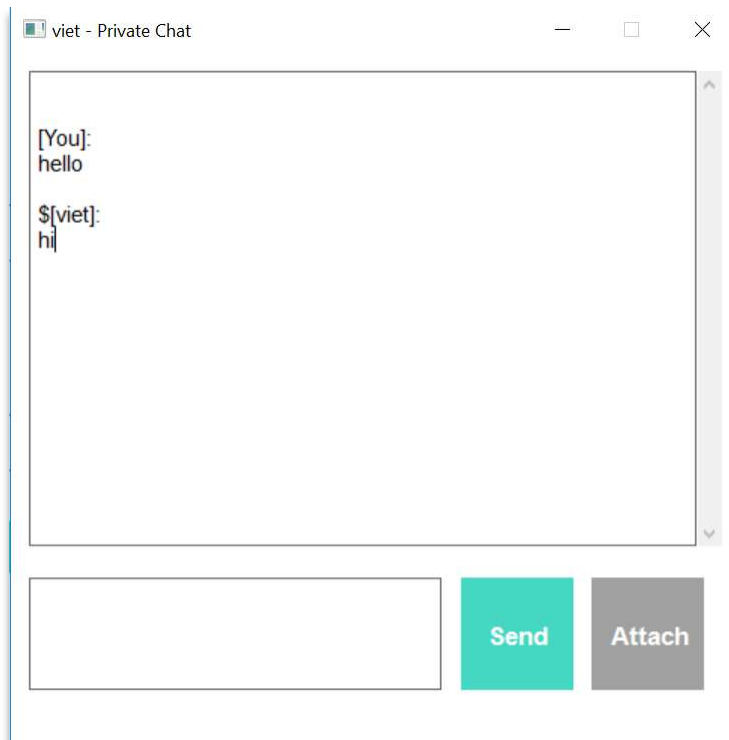
Username:

☐ Group Chat

Group name:

Create

Hình 4.3 Màn hình sau khi đăng nhập thành công.



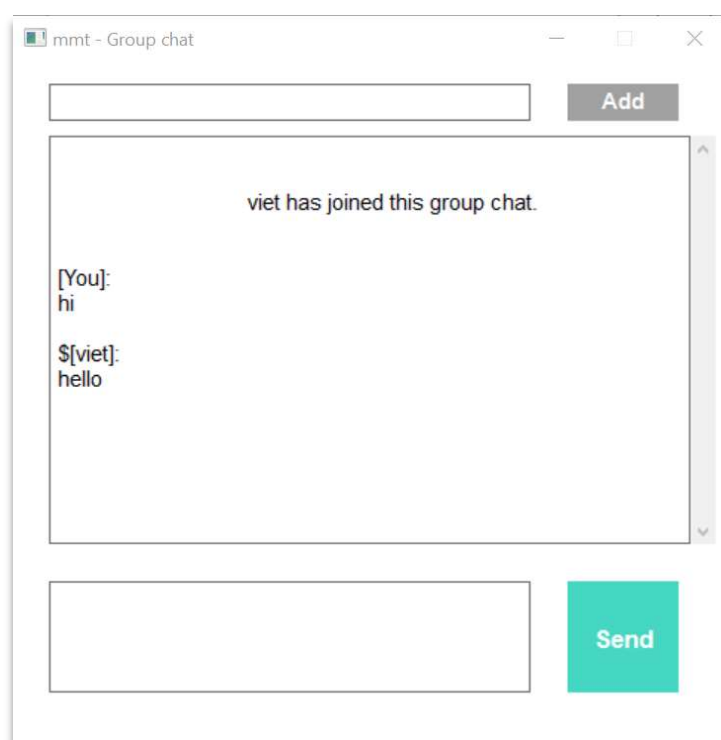
viet - Private Chat

[You]:  
hello

\$(viet):  
hi

Send Attach

Hình 4.4 Cửa sổ chat với 1 người



mmt - Group chat

Add

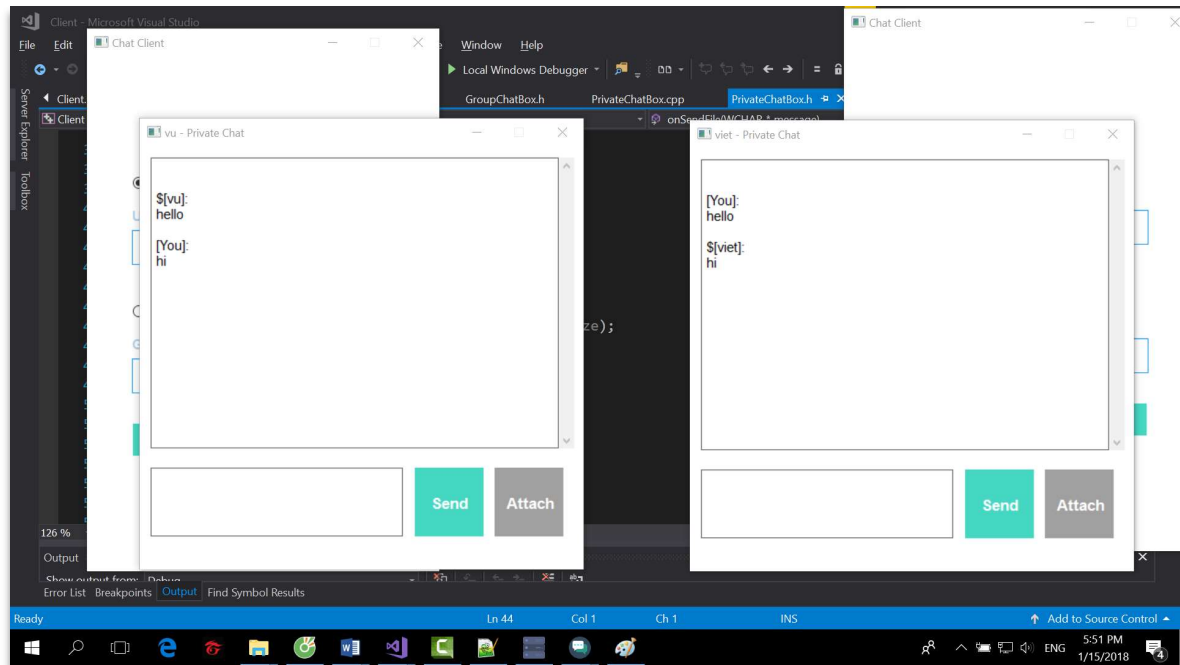
viet has joined this group chat.

[You]:  
hi

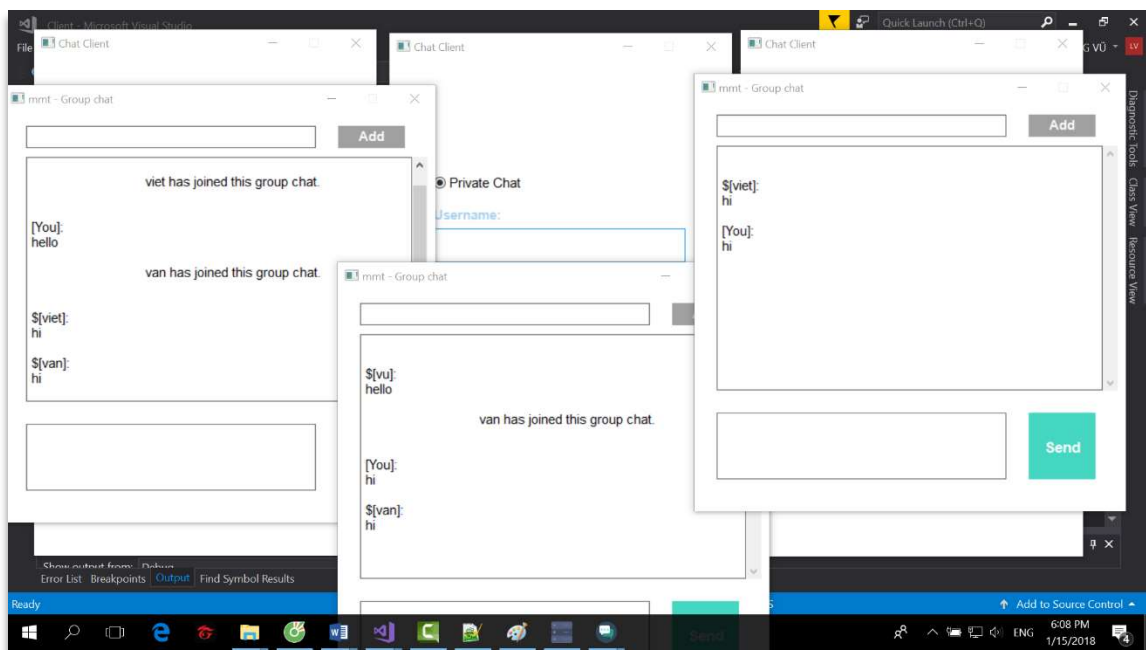
\$(viet):  
hello

Send

Hình 4.5 Cửa sổ chat nhóm



Hình 4.6 Hình chat với 1 người



Hình 4.7 Hình chat nhóm (3 người, tên nhóm mmt)

**\*Demo gửi tập tin không thể thể hiện bằng hình ảnh được nên sẽ demo trong video**

## 4.2 Video

**Link youtube:** <https://www.youtube.com/watch?v=qeX0n49aeII>

---

## V. TỔNG KẾT

Thông qua đồ án nhóm đã biết và viết được 1 ứng dụng hỗ trợ giao diện nhận gửi gói tin thông qua socket của C/C++ kết hợp multi-thread. Tạo nên 1 mô hình chat client-server đơn giản phục vụ các chức năng, yêu cầu của đồ án.

## VI. ĐÁNH GIÁ

### 6.1 Đánh giá mức độ hoàn thành đồ án

Yêu cầu	Mức độ hoàn thành
Đăng ký tài khoản	100%
Chat với 1 người	100%
Chat nhóm	100%
Gửi tập tin	100%
Nâng cao	100%

## TÀI LIỆU THAM KHẢO

- [1] Demo **Socket\_Chat.rar**, Th.S Lê Viết Long, ĐH Khoa học Tự nhiên, ĐH Quốc Gia Tp. HCM.
- [2] Demo **Demo\_Nhieu\_Client.rar**, Th.S Lê Viết Long, ĐH Khoa học Tự nhiên, ĐH Quốc Gia Tp. HCM.
- [3] Các bài viết - thread trên trang **stackoverflow.com**
- [4] Các bài viết, mã nguồn trên trang **codeproject.com**
- [5] Các bài viết trên trang **daynhauhoc.com**
- [6] Bài viết **<http://diendan.congdongcviet.com/threads/t267714::c-gui-nhan-file-qua-socket.cpp>**

*Thời gian cập nhật lần cuối: 7h ngày 15/01/2018*

**-HẾT-**