ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Báo Cáo
Đồ Án Chat Sử Dụng GRPC

Môn Học: Chuyên Đề Hệ Thông Phân Tán

Giáo Viên Hướng Dẫn:
Dũng Trần Trung
Huỳnh Thụy Bảo Trân

Sinh Viên Thực Hiện
Trần Mạnh Khương - 20127540
Lớp: 20MMT

TP Hồ Chí Minh, 06 tháng 02 năm 2024

Contents

I. Tổng Quan

1. Thông tin cá nhân

MSSV	Họ và Tên
20127540	Trần Mạnh Khương

2. Thông tin đồ án

Viết chương trình chat trong đó nhiều người dùng có thể chat nhóm với nhau. Có tối thiểu 5 người dùng:

- Có 1 chương trình server:
  + Phục vụ chương trình người dùng (client), cho phép client đăng kí, và tìm kiếm user khác trong nhóm chat.
  + Cho phép user tạo 1 nhóm chat, hỗ trợ broadcast msg trong 1 nhóm, user1 muốn gửi msg tới tất cả users còn lại trong nhóm, thì gửi msg tới server, rồi server fwd đến các users còn lại.
  + Cho phép user chat riêng với 1 user khác, user 1 gửi msg tới server, rồi server forward msg này tới user chat riêng.
- Mỗi người dùng chạy một client riêng, đại diện 1 user chat.
- Người dùng có thể gửi msg trực tiếp cho 1 người khác, hay gửi trong 1 nhóm mình tạo, không bị ràng buộc điều kiện.

Thiết kế chương trình:

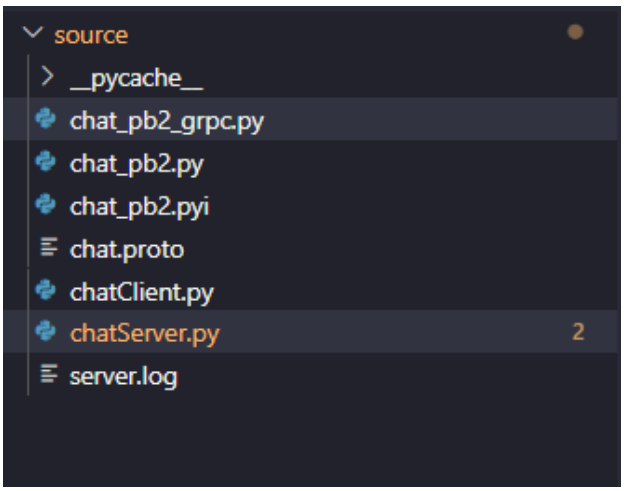
- Giao tiếp giữa các tiến trình phải dùng grpc. Grpc là thư viện do google phát triển, có hỗ trợ nhiều ngôn ngữ lập trình khác nhau.

3. Đánh giá mức độ hoàn thành đồ án

Công việc	*Tỷ lệ hoàn thành (%)*
1. Đăng kí user	100%
2. Tìm kiếm user trong nhóm chat	100%
3. User tạo nhóm chat	100%
4. User tham gia vào nhóm chat	100%
5. User gửi tin nhắn broadcast đến nhóm chat	100%
6. User gửi tin nhắn riêng đến một user khác	100%

II. Nội Dung Báo Cáo

1. Cấu trúc file chương trình



Chương trình được viết bằng ngôn ngữ python. Gồm các file sau:

- chat.proto: định nghĩa data cho request và response của server và client.
- chatClient.py: chứa mã nguồn xử lý các yêu cầu của đồ án để giao tiếp với server.
- chatServer.py: chứa mã nguồn xứ lý các yêu cầu của đồ án để giao tiếp với client.
- server.log: dùng để ghi lại lịch sử quá trình client và server giao tiếp với nhau.
- chat_pb2_grpc.py, chat_pb2.py và chat_pb2.pyi: được generate từ file chat.proto.

2. *Thông tin mã nguồn từng file*

❖ *File chat.proto:*

```
message GroupInfo {
    string group_id = 1;
    repeated string members = 2;
}
message GroupInfoList {
    repeated GroupInfo groups = 1;
}
message JoinGroupRequest {
  string username = 1;
  string group_id = 2;
}

message JoinGroupResponse {
  string message = 1;
}

message SearchUserRequest {
    string groupId_to_search = 1;
    string username_to_search = 2;
}

message SearchUserResponse {
    string message = 1;
}

message BroadcastMessageRequest {
    string group_id = 1;
    string sender = 2;
    string content = 3;
}

message BroadcastMessageResponse {
    bool success = 1;
    string message = 2;
}

message GetGroupChatRequest {
  string group_id = 1;
}

message Message {
    int32 index = 1;
    string sender = 2;
    string content = 3;
}
```

```
message GetGroupChatResponse {
    bool success = 1;
    string message = 2;
    repeated Message messages = 3;
}

message PrivateMessageRequest {
    string sender = 1;
    string receiver = 2;
    string content = 3;
}


service ChatService {
    rpc RegisterUser(RegisterRequest) returns (RegisterResponse);

    rpc CreateGroup(GroupRequest) returns (GroupResponse);

    rpc GetGroupInfoList(google.protobuf.Empty) returns (GroupInfoList);

    rpc JoinGroup(JoinGroupRequest) returns (JoinGroupResponse);

    rpc SearchUserInGroup(SearchUserRequest) returns (SearchUserResponse);

    rpc BroadcastMessageStream(BroadcastMessageRequest) returns (stream BroadcastMessageResponse);

    rpc GetGroupChat(GetGroupChatRequest) returns (GetGroupChatResponse);

    rpc SendPrivateMessage(PrivateMessageRequest) returns (Empty);

}
```

Trong đó:

- message RegisterRequest là định nghĩa gói tin chứa trường username để lưu tên người dùng khi đăng kí một client mới.
- message RegisterResponse là định nghĩa gói tin chứa các trường message và success. Trong đó trường message, success dùng để phản hồi thông điệp và trạng thái đăng kí thành công chưa cho cho client biết.
- message GroupRequest là định nghĩa gói tin chứa các trường group_id và username. Trong đó trường group_id và username là id nhóm muốn tạo và tên người tạo nhóm để gửi request từ client đến server.
- message GroupResponse là định nghĩa gói tín trả về từ server cho client chứa trường success để phản hồi là tạo group chat đã thành công hay chứa.
- message GroupInfo là định nghĩa gói tin chứa trường group_id và member. Trong đó group_id là id của nhóm chat và member là danh sách các thành viên trong nhóm chat.
- message GroupInfoList là định nghĩa gói tin chứa một danh sách lặp của các message GroupInfo (repeated GroupInfo). Nó đại diện cho danh sách các nhóm, mỗi nhóm được mô tả bằng một message GroupInfo.
- message JoinGroupRequest là định nghĩa gói tin gửi yêu cầu request từ client đến server chứa các trường username và group_id. Trong đó trường username là tên của

user muốn tham gia vào group chat còn group_id là id của nhóm chat mà user muốn tham gia.

- message JoinGroupResponse là định nghĩa gói tin phản hồi response từ server về client chứa trường message để thông báo cho client biết là đã tham gia nhóm thành công hay chưa.
- message SearchUserRequest là định nghĩa gói tin gửi yêu cầu request từ client đến server chứa các trường groupId_to_search và username_to_search_id. Trong đó trường username_to_search là tên của user muốn tìm kiếm trong nhóm chat vào groupId_to_search là id của nhóm chat để user đó.
- message SearchUserResponse là định nghĩa gói tin phản hồi response từ server về client chứa trường message để thông báo cho client biết là tìm kiếm thành công hay không.
- message BroadcastMessageRequest là định nghĩa gói tin gửi yêu cầu request từ client đến server chứa các trường group_id, sender và content. Trong đó trường group_id là id của nhóm chat cần gửi tin nhắn, sender là user gửi tin, content là nội dung của tin nhắn.
- message BroadcastMessageResponse là định nghĩa gói tin phản hồi response từ server về client chứa trường success và message để thông báo cho client biết là gửi tin nhắn lên nhóm chat thành công hay chưa.
- message GetGroupChatRequest định nghĩa gói tin chứa trường group_id là id của nhóm chat.
- message Message định nghĩa gói tin chứa các trường index, sender và content. Trong đó các trường index, sende và content lần lượt là vị trí gửi tin nhắn, sender là người gửi, content là nội dung gửi.
- message GetGroupChatResponse là định nghĩa gói tin chứa trường success, message và danh sách lặp của message Message.
- message PrivateMessageRequest là định nghĩa gói tin gửi yêu cầu request từ client đến server chứa các trường sender, receiver và content. Trong đó trường sender là người gửi tin nhắ, receiver là gửi nhận tin nhắn, content là nội dung của tin nhắn.

Để generate ra các gile chat_pb2_grpc.py, chat_pb2.py và chat_pb2.pyi thì chạy lệnh bên dưới trong folder chứa file chat.proto: python -m grpc_tools.protoc -I. --python_out=. --pyi_out=. --grpc_python_out=. chat.proto

❖ *File chatServer.py*

Gồm các hàm sau:

- ***def __init__(self)***: dùng để khởi tạo một số biến cần thiết trong đó

```
def __init__(self):
    self.users = {}  # Dùng để lưu thông tin của người dùng (username và địa chỉ)
    self.groups=[] # Dùng để lưu các group chat
    self.box_chat_groups={} # Dùng để lưu các hộp thoại group chat
    self.connected_clients = {} # Dùng để lưu các client đang connected để server
```

  + users: dùng để lưu trữ thông tin của các người dùng bao gồm username và địa chỉ ipv6
  + groups: dùng để lưu trữ các group chat đã được tạo
  + box_chat_groups: dùng để lưu trữ hộp thoại tin nhắn của các group và của người dùng với người dùng
  + connected_clients: dùng để lưu trữ các client đang connected đến server
- def RegisterUser(self, request, context): hàm này sẽ nhận request từ client đó là username để thực hiện việc đăng user và lưu thông tin vào list users. Và username được đăng kí là duy nhất không được trùng nhau.

```
def RegisterUser(self, request, context):
    username = request.username
    if username in self.users:
        return chat_pb2.RegisterResponse(success=False)
    else:
        self.users[username] = {"username": username, "ip_address": context.peer()}
        response = chat_pb2.RegisterResponse(success=True)
        server_logger.info(f"User {username} registered from address {context.peer()}")
        return response
```

- def CreateGroup(self, request, context): dùng để tạo group chat. Nhận request từ client với 2 tham số đó là username và group_id trong đó username là tên người tạo group còn group_id là id của group. Nếu group chat chưa tồn tại thì tạo group chat

```
def CreateGroup(self, request, context):
    group_id = request.group_id
    username = request.username

    # Kiểm tra xem nhóm đã tồn tại hay chưa
    existing_group = next((group for group in self.groups if group["group_id"] == group_id), None)

    if existing_group is None:
        # Nếu nhóm chưa tồn tại, tạo mới một từ điển nhóm và thêm vào danh sách
        new_group = {"group_id": group_id, "members": [{"username": username, "ip_address": context.peer()}]}
        self.groups.append(new_group)
        server_logger.info(f"User {username} successfully created group chat {group_id}")
        return chat_pb2.GroupResponse(success=True)
    else:
        server_logger.info(f"User {username} created a group chat {group_id} that failed")
        return chat_pb2.GroupResponse(success=False)
```

mới và thêm group chat mới được tạo vào list groups để lưu lại. Id group chat là duy nhất không được trùng nhau.

- Hàm def GetGroupInfoList(self, request, context): dùng in thông tin trong một group chat như group id, tên user tham gia vào group chat, địa chỉ ipv6 của user đó. Và chỉ nhận request từ client và không cần trả response về client.

```
def GetGroupInfoList(self, request, context):
        group_info_list = []
        for group in self.groups:
            members_list = [f"Username: {member['username']} ({member['ip_address']})" for member in group["members"]]
            group_info = chat_pb2.GroupInfo(group_id=group["group_id"], members=members_list)
            group_info_list.append(group_info)
            server_logger.info(f"Group {group['group_id']} members: {members_list}")

        server_logger.info(f"Group info list: {group_info_list}")
        # Không cần trả về thông tin cho client, chỉ cần in ra console
        return Empty()
```

- def JoinGroup(self, request, context): dùng để user tham gia vào một group chat nhận 2 thông đầu vào từ client đó là username và group_id. Trước tiên kiểm tra xem group id cần tham gia vào có hợp lệ không nếu không hợp lệ thì phản hồi cho client biết là group chat không tồn tại còn hợp lệ thì tiếp tục kiểm tra xem user đó đã tham gia chưa nếu chưa nếu rồi thì phản hồi về client đã tham gia vào nhóm từ trước rồi còn nếu user chưa tham gia vào nhóm thì thêm vào nhóm chat.

```
def JoinGroup(self, request, context):
    username = request.username
    group_id = request.group_id

    # Kiểm tra xem nhóm có tồn tại không
    existing_group = next((group for group in self.groups if group["group_id"] == group_id), None)

    if existing_group:
        # Kiểm tra xem usern                    a
        if username not in [member["username"] for member in existing_group["members"]]:
            existing_group["members"].append({"username": username, "ip_address": context.peer()})
            response = chat_pb2.JoinGroupResponse( message=f"User {username} joined the group {group_id}")
            server_logger.info(f"User {username} joined the group {group_id}")

        else:
            response = chat_pb2.JoinGroupResponse(message=f"User {username} is already in the group {group_id}")
            server_logger.info(f"User {username} is already in the group {group_id}")
    else:
        response = chat_pb2.JoinGroupResponse(message=f"Group {group_id} does not exist")
        server_logger.info(f"User {username} failed to join group  {group_id} because group  {group_id} does not exist")

    return response
```

(variable) member: Any

- def SearchUserInGroup (self, request, context): dùng để tìm kiếm user trong một group chat cho trước, hàm này nhận hai tham số từ client đó là tên cần tìm và group id để tìm user trong group đó. Trước tiên kiểm tra group tồn tại không nếu không thì phản hồi cho client nếu tồn tại tìm xem user nào ở trong cái group id hợp có tên trùng với cần tìm và thông báo về cho client.

```python
def SearchUserInGroup (self, request, context):
    group_id_to_search = request.groupId_to_search
    username_to_search = request.username_to_search

    existing_group = next((group for group in self.groups if group["group_id"] == group_id_to_search), None)
    if existing_group:
        user_exists = any(member["username"] == username_to_search for member in existing_group["members"])
        if user_exists:
            response = chat_pb2.SearchUserResponse( message=f"User {username_to_search} is in the group" )
            server_logger.info(f"User {username_to_search} is in the group.")

        else:
            response = chat_pb2.SearchUserResponse(message=f"User {username_to_search} is not in the group.")
            server_logger.info(f"User {username_to_search} is not in the group.")

    else:
        response = chat_pb2.SearchUserResponse(message=f"Group {group_id_to_search} does not exist.")
        server_logger.info(f"Group {group_id_to_search} does not exist.")

    return response
```

- def BroadcastMessageStream(self, request, context): hàm này dùng để gửi tin nhắn từ 1 user đến tất cả user còn lại trong nhóm, hàm này yêu cầu 3 tham số từ client đó là sender người gửi, group_id là id group chat cần gửi, content là nội dung gửi. Trước tiên kiểm tra nhóm tồn tại hay không, nếu tồn tại ta sẽ lưu tất cả thông tin cần thiết vào box_chat_groups để hiển thị ra cho client xem. Sau đó lấy từng thành viên trong nhóm và gửi tin nhắn đi cho tất cả thành viên đó.

```python
def BroadcastMessageStream(self, request, context):
    group_id = request.group_id
    sender = request.sender
    content = request.content

    # Kiểm tra xem group_id có tồn tại không
    if group_id not in [group['group_id'] for group in self.groups]:
        server_logger.info(f"User {sender} sends a broadcast messagefailed to go to group chat {group_id} because group chat {group_id} does not exist")
        yield chat_pb2.BroadcastMessageResponse(success=False, message=f"Group {group_id} does not exist")
        return

    # Lưu thông điệp vào box_chat_groups
    if group_id not in self.box_chat_groups:
        self.box_chat_groups[group_id] = {"messages": [], "members": []}

    # Lấy vị trí index gửi (số thứ tự trong danh sách tin nhắn của nhóm)
    index = len(self.box_chat_groups[group_id]["messages"]) + 1

    # Lưu thông điệp vào danh sách tin nhắn của nhóm
    message = {"index": index, "sender": sender, "content": content}
    self.box_chat_groups[group_id]["messages"].append(message)

    # Lấy danh sách thành viên trong nhóm
    group_members = self.connected_clients.get(group_id, {}).items()

    # Lưu danh sách thành viên vào box_chat_groups
    self.box_chat_groups[group_id]["members"] = [member_username for member_username, _ in group_members]

    for member_username, member_channel in group_members:
        if member_username != sender:
            response = chat_pb2.BroadcastMessageResponse(success=True, message=f"{sender}: {content}")

            # Kiểm tra điều kiện trước khi gửi dữ liệu đến từng thành viên trong nhóm
            if member_channel is not None:
                member_channel.SendMessage(response)
            else:
                # Xử lý lỗi khi thành viên trong nhóm không có kênh
                server_logger.info(f"Error: User {member_username} does not have a valid channel")

    server_logger.info(f"User {sender} send the broadcast message successfully to group chat {group_id}")
    # Trả về phản hồi cho client gửi tin nhắn

    yield chat_pb2.BroadcastMessageResponse(success=True, message="Message broadcasted successfully")
```

– def GetGroupChat(self, request, context): dùng để hiện thị nội dung tin nhắn sẽ các thành viên trong group và giữa một user với một user khác.

```
def GetGroupChat(self, request, context):
    group_id = request.group_id

    # Kiểm tra xem nhóm có tồn tại không
    if group_id in self.box_chat_groups:
        # Trả về danh sách tin nhắn của nhóm
        messages = self.box_chat_groups[group_id]["messages"]
        response = chat_pb2.GetGroupChatResponse(success=True, messages=messages)
    else:
        response = chat_pb2.GetGroupChatResponse(success=False, message=f"Group {group_id} does not exist")

    return response
```

– def SendPrivateMessage(self, request : chat_pb2.PrivateMessageRequest, context): dùng để gửi nhắn giữa một user với một user khác. Và lưu nội dung tin nhắn vào box_chat_groups, hàm này nhận yêu cầu request từ clien với các tham số như sender, receiver, content. Trước tiên ta cần hash id của sender và receiver thành id mới thông qua hàm def myHash(text:str), tiếp theo kiểm tra id đó có nằm trong box_chat_groups và có nội dung chưa nếu chưa có nội dùng conten thì ta sẽ tạo một thông tin mới và lưu vào box_chat_groups còn nội dung content có rồi thì ta sẽ lưu nội dung conten đó tiếp tục vào.

```
def SendPrivateMessage(self, request : chat_pb2.PrivateMessageRequest, context):
    sender = request.sender
    receiver = request.receiver
    content =  request.content

    id = str(myHash(str(int(myHash(sender)) + int(myHash(receiver)))))
    list_id =  list(self.box_chat_groups.keys())
    if id not in list_id and content=="":
        self.box_chat_groups[id] = {"messages": [], "members": []}
    if content != "":
        index = len(self.box_chat_groups[id]["messages"]) + 1
        message = {"index": index, "sender": sender, "content": content}
        self.box_chat_groups[id]["messages"].append(message)
    server_logger.info(f"User {sender} has successfully sent a message to user {receiver}.")
    return chat_pb2.Empty()
```

```
def myHash(text:str):
    hash=0
    for ch in text:
        hash = ( hash*281 ^ ord(ch)*997) & 0xFFFFFFFF
    return str(hash)
```

– Def server(): dùng để chạy server và ghi log file

```
def serve():
    for file in log_file_list:
        os.remove(file)
    runlog("server.log")
    port=54321
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=30))
    chat_pb2_grpc.add_ChatServiceServicer_to_server(ChatServer(), server)
    server.add_insecure_port('[::]:'+str(port))
    server.start()
    server_logger.info("Server is running...")
    server.wait_for_termination()
```

❖ *File chatClient.py*

Gồm các hàm chính sau:

– def __init__(self, username, server_address='localhost', server_port=54321):

```
def __init__(self, username, server_address='localhost', server_port=54321):
    self.username = username
    self.channel = grpc.insecure_channel(f'{server_address}:{server_port}')
    self.stub = chat_pb2_grpc.ChatServiceStub(self.channel)
```

+ username: là tên của user điền vào
+ channel và stub: : khởi tạo một channel kết nối với server với prot là 54321

– def register_user(self): hàm này dùng để gửi yều đăng kí client đến server và nhận phản hồi từ server

```
def register_user(self):
    request = chat_pb2.RegisterRequest(username=self.username)
    response = self.stub.RegisterUser(request)
    return response.success
```

– def create_group_chat (self, group_id): dùng để gửi yêu cầu tạo một group chat với tham số là group_id, và nhận về phản hồi về từ server.

```
def create_group_chat (self, group_id):
    request = chat_pb2.GroupRequest(group_id=group_id, username=self.username)
    response = self.stub.CreateGroup(request)
    return response.success
```

– def get_group_info_list(self): dùng để gửi yêu cầu lấy thông tin các group đã được tạo với id group và user đang ở trong group đó.

```
def get_group_info_list(self):
    response = self.stub.GetGroupInfoList(google_dot_protobuf_dot_empty__pb2.Empty())
    for group_info in response.groups:
        members_list = [f"Username: {member}" for member in group_info.members]
        print(f"Group {group_info.group_id} members: {members_list}")
```

– def join_group(self, group_id): dùng để gửi yêu cầu đến server để tham gia vào group_id cho trước và nhận phản hồi từ server.

```
def join_group(self, group_id):
    request = chat_pb2.JoinGroupRequest(username=self.username, group_id=group_id)
    response = self.stub.JoinGroup(request)
    return response.message
```

– def search_user_in_group(self,group_id_to_search, username_to_search): gửi yêu cầu tìm kiếm một user cho trước từ một group cũng cho trước đến server và nhận về phản hồi về server.

```
def search_user_in_group(self,group_id_to_search, username_to_search):
    request = chat_pb2.SearchUserRequest(groupId_to_search=group_id_to_search,username_to_search=username_to_search)
    response = self.stub.SearchUserInGroup(request)
    return response.message
```

– def broadcast_message(self, group_id, content): dùng để gửi yêu cầu gửi tin nhắn broadcast đến các thành viên trong nhóm với các tham số là group_id và content trong đó group_id là id nhóm chat cần gửi nội dung, conten là nội dung cần gửi. Và nhận phản hồi từ server.

```
def broadcast_message(self, group_id, content):
    request = chat_pb2.BroadcastMessageRequest(group_id=group_id, sender=self.username, content=content)
    response_stream = self.stub.BroadcastMessageStream(request)

    # Kiểm tra xem response_stream có tồn tại và không rỗng không
    if response_stream:
        for response in response_stream:
            print(f"Received broadcast: {response.message}")
    else:
        print("No response received from the server.")
```

– def get_group_chat(self, group_id): dùng để hiện thị box chat của từng group chat và box chat của một user với một user khác.

```
def get_group_chat(self, group_id):
    request = chat_pb2.GetGroupChatRequest(group_id=group_id)
    try:
        response = self.stub.GetGroupChat(request)
        if response.success:
            messages = response.messages
            for message in messages:
                print(f"{message.index}              {message.sender}                {message.content}")
        else:
            print(f"Failed to get group chat for group {group_id}: {response.message}")
    except grpc.RpcError as e:
        print(f"Error in RPC call: {e}")
```

- def send_private_message(self, receiver, content): dùng để gửi cầu gửi tin nhắn riêng với một user khác với các tham số receiver là người nhận còn content là nội dung tin nhắn cần gửi.

- def client(): dùng để xử lý logic và thực các yêu cầu của đồ án.

```
def send_private_message(self, receiver, content):
    request = chat_pb2.PrivateMessageRequest(sender=self.username, receiver=receiver, content= content)
    try:
        self.stub.SendPrivateMessage(request)
    except grpc.RpcError as e:
        print(f"Error in RPC call: {e}")
```

III. Hình Ảnh Thực Thi Chương Trình

1. Link Video demo

https://drive.google.com/file/d/1wcCl2AqsBCPOHcOtlvF65WzfV2QS7cA1/view?usp=sharing

2. Hình ảnh để demo các yêu cầu

- Cài đặt số thư viên cần thiết sử dụng commandline:

  python -m pip install --upgrade pip

  python -m pip install grpcio

  python -m pip install grpcio-tools

- Để generate ra các gile chat_pb2_grpc.py, chat_pb2.py và chat_pb2.pyi thì chạy lệnh bên dưới trong folder chứa file chat.proto: python -m grpc_tools.protoc -I. --python_out=. --pyi_out=. --grpc_python_out=. chat.proto
- Sử dụng terminal run các python: python server.py và python client.py (để tạo nhiều client thì ta cần chạy terminal python client.py nhiều lần)

- Chạy server

```
PS D:\Mon Hoc\Nam 4\HK2\Chuyen De He Thong Phan Tan\TH\DA1\Project_1\source> python chatServer.py
__SERVER__LOG__ - INFO:
Server is running...
--------------------
```

a) Đăng kí user

```
PS D:\Mon Hoc\Nam 4\HK2\Chuyen De He Thong Phan Tan\TH\DA1\Project_1\source> python chatClient.py
Enter your username to register: client1
Successfully registered username: client1

Options:
1. Search User In A Group
2. Create Group Chat
3. Join Group
4. Send Group Chat Message
5. Send Private Chat Message
6. Exit
Enter your choice:
```

```
__SERVER__LOG__ - INFO:
User client1 registered from address ipv6:%5B::1%5D:64505
--------------------

__SERVER__LOG__ - INFO:
User client2 registered from address ipv6:%5B::1%5D:64506
--------------------

__SERVER__LOG__ - INFO:
User client3 registered from address ipv6:%5B::1%5D:64507
--------------------

__SERVER__LOG__ - INFO:
User client4 registered from address ipv6:%5B::1%5D:64508
--------------------

__SERVER__LOG__ - INFO:
User client5 registered from address ipv6:%5B::1%5D:64509
--------------------
```

b) User tạo group chat

– Bước 1: đăng kí user

```
PS D:\Mon Hoc\Nam 4\HK2\Chuyen De He Thong Phan Tan\TH\DA1\Project_1\source> python chatClient.py
Enter your username to register: client1
Successfully registered username: client1

Options:
1. Search User In A Group
2. Create Group Chat
3. Join Group
4. Send Group Chat Message
5. Send Private Chat Message
6. Exit
Enter your choice:
```

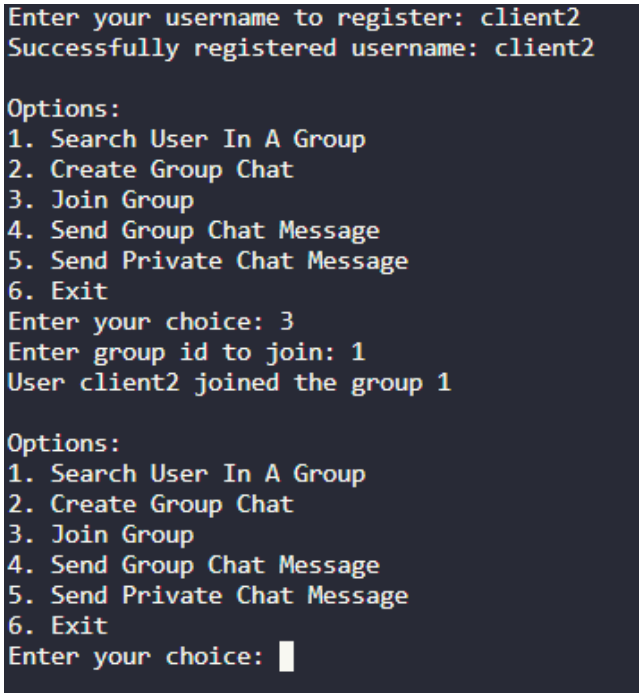
– Bước 2: chọn phím trên để tạo group chat

```
Options:
1. Search User In A Group
2. Create Group Chat
3. Join Group
4. Send Group Chat Message
5. Send Private Chat Message
6. Exit
Enter your choice: 2
Enter group id to create group chat: 1
Successfully created a group chat

Options:
1. Search User In A Group
2. Create Group Chat
3. Join Group
4. Send Group Chat Message
5. Send Private Chat Message
6. Exit
Enter your choice:
```

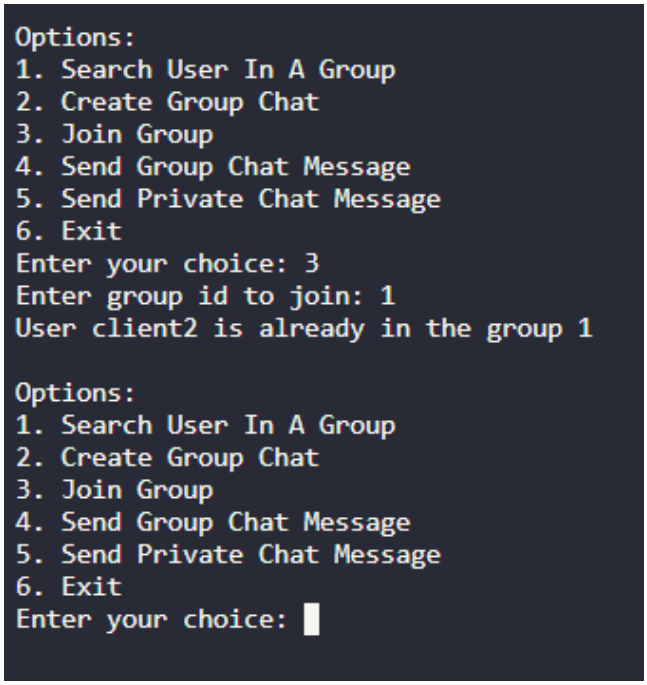
```
__SERVER__LOG__ - INFO:
User client1 successfully created group chat 1
--------------------

__SERVER__LOG__ - INFO:
Group 1 members: ['Username: client1 (ipv6:%5B::1%5D:64505)']
--------------------

__SERVER__LOG__ - INFO:
Group info list: [group_id: "1"
members: "Username: client1 (ipv6:%5B::1%5D:64505)"
]
--------------------
```

c) User tham gia vào group chat

- Bước 1: để user tham gia vào group chat thì ta tạo thêm một user mới và đăng kí tương tự như phần a.
- Bước 2: chọn phím 3 để tham vào group chat



- Bước 3: Nhập id cho group_id để tham gia vào group chat. nếu group_id không có thì ta quay lại bước 2



```
__SERVER__LOG__ - INFO:
Group info list: [group_id: "1"
members: "Username: client1 (ipv6:%5B::1%5D:64505)"
members: "Username: client2 (ipv6:%5B::1%5D:64506)"
members: "Username: client3 (ipv6:%5B::1%5D:64507)"
members: "Username: client4 (ipv6:%5B::1%5D:64508)"
]
--------------------
```

d) Tìm kiếm user trong group chat

- Để tìm kiếm user trong nhóm chat ta cần nhập id nhóm chat và tên user cần tìm. Hình ảnh các bước bên dưới cho từng trường hợp:

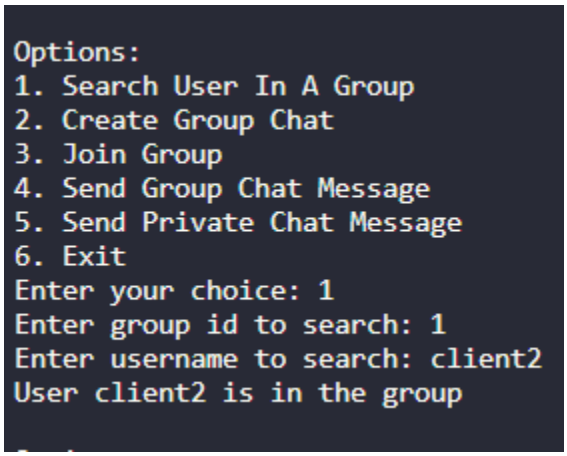
trường hợp group chưa được tạo

```
Options:
1. Search User In A Group
2. Create Group Chat
3. Join Group
4. Send Group Chat Message
5. Send Private Chat Message
6. Exit
Enter your choice: 1
Enter group id to search: 2
Enter username to search: 3
Group 2 does not exist.
```

trường hợp có group nhưng tìm không thấy user name cần tìm

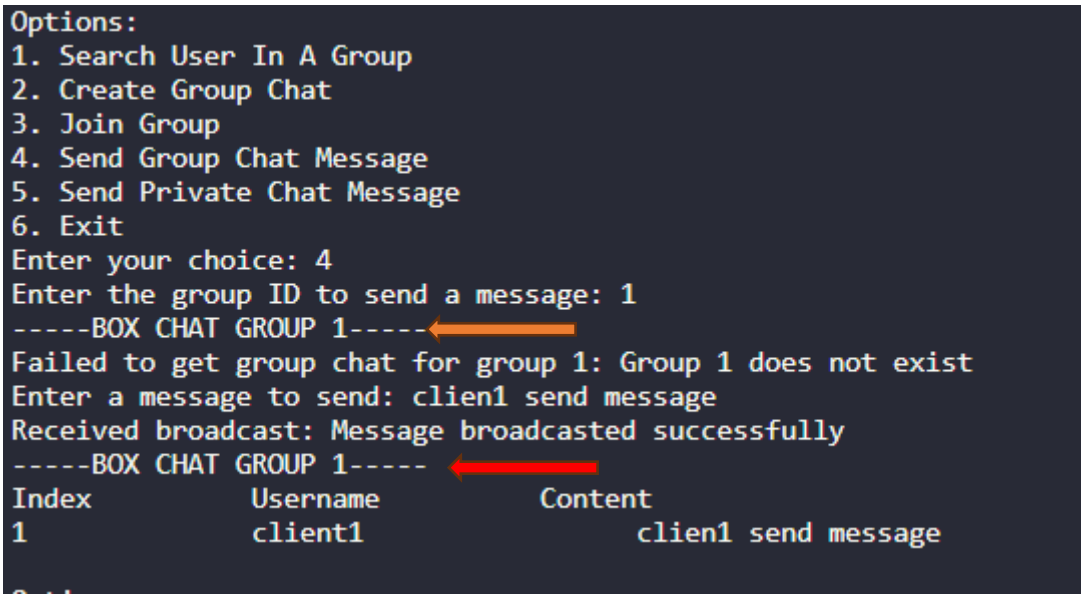
```
Options:
1. Search User In A Group
2. Create Group Chat
3. Join Group
4. Send Group Chat Message
5. Send Private Chat Message
6. Exit
Enter your choice: 1
Enter group id to search: 1
Enter username to search: 2
User 2 is not in the group.
```

trường hợp tìm được usernam cần tìm trong group



e) Gửi tin broadcast đến group chat

- Hình ảnh box chat ở client1. Box chat ở mũi tên màu cam dùng để hiển thị tin nhắn đã được gửi lúc trước của các thành viên trong nhóm chat gửi lên. Còn box chat ở mũi tên màu đỏ là box chat đã được cập nhật khi mình vừa gửi tin nhắn lên



- Hình ảnh box chat bên client2. Box chat tương tự như client1

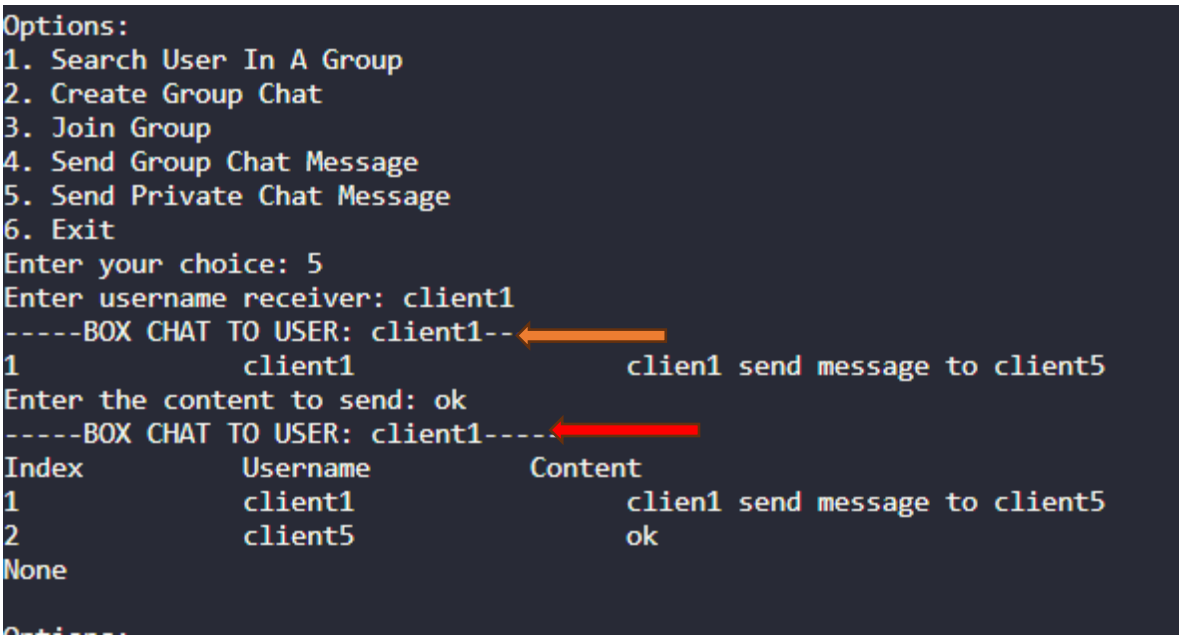
```
Options:
1. Search User In A Group
2. Create Group Chat
3. Join Group
4. Send Group Chat Message
5. Send Private Chat Message
6. Exit
Enter your choice: 4
Enter the group ID to send a message: 1
-----BOX CHAT GROUP 1---
1               client1                 clien1 send message
Enter a message to send: client2 send message
Received broadcast: Message broadcasted successfully
-----BOX CHAT GROUP 1-----
Index           Username            Content
1               client1                 clien1 send message
2               client2                 client2 send message

Options:
```

f) Gửi tin nhắn giữa một user với user khác

- Ta sẽ thực hiện gửi tin nhắn riêng từ client 1 đến clien 5
- Hình ảnh ở client1. Box chat mũi tên tương tự như của gửi tin nhắn broadcast gửi trong nhóm

```
Options:
1. Search User In A Group
2. Create Group Chat
3. Join Group
4. Send Group Chat Message
5. Send Private Chat Message
6. Exit
Enter your choice: 5
Enter username receiver: client5
-----BOX CHAT TO USER: client5--
Enter the content to send: clien1 send message to client5
-----BOX CHAT TO USER: client5---
Index           Username            Content
1               client1                 clien1 send message to client5
None

Options:
```

- Hình ảnh ở client5. Box chat tương tự như trên



IV. Tài Liệu Tham Khảo

1. https://grpc.github.io/grpc/python/
2. https://grpc.io/docs/languages/python/
3. https://melledijkstra.github.io/science/chatting-with-grpc-in-python?fbclid=IwAR2qIiu21YDTKBbysjDNcKDwmu-ZlLTpfalBk9m2eobfOeqwrYrSIKWsmbA