ĐẠI HỌC QUỐC GIA HÀ NỘI **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ Độc lập – Tự do – Hạnh phúc**

****

**BÁO CÁO**

**BÀI TẬP LỚN**

**MÔN: LẬP TRÌNH MẠNG**

**Lớp: INT3304 1**

**Trần Thị Dung : 14020580**

**Đỗ Xuân Quý : 14020634**

**Đào Tuấn Anh :**

**Hà Nội, 2017**

**Mục lục**

1. Các yêu cầu........................................................................3.
   1. Yêu cầu
   2. Tính năng chính
   3. Môi trường chạy của chương trình
2. Giao thức MQTT.................................................................3
   1. Định nghĩa MQTT
   2. Thành phần MQTT
3. Mô tả chương trình............................................................4
   1. Thiết kế chương trình
   2. Chạy chương trình
4. Demo chương trình............................................................5
5. Ghi chú................................................................................9

**I. Các yêu cầu**

1. Yêu cầu:

Chương trình chat sử dụng giao thức kiểu publish/subscribe như MQTT bao gồm các thành phần:

- Server đóng vai trò của MQTT Broker

- Clients đóng vai trò vừa là Publisher (bên công bố thông tin) vừa là Subcriber (bên đăng ký nhận thông tin).

2. Tính năng chính:

- Cho phép người dùng nói chuyện trực tiếp với nhau

- Cho phép người dùng nói chuyện theo nhóm

- Cho phép người dùng gửi file cho nhau

- Cho phép người dùng gửi file cho một nhóm

3. Môi trường chạy của chương trình

- Hệ điều hành: Linux

- Programming language: C

**II. Giao thức MQTT**

1. Định nghĩa MQTT

***MQTT***  (Message Queue Telemetry Transport) là một chuẩn ISO (ISO/IEC PRF 20922) dưa trên giao thức gửi tin nhắn publish-subscribe, sử dụng cho các thiết bị Internet of Things với băng thông thấp, độ tin cậy cao và được sử dụng trong mạng lướng không ổnđịnh.

2. Thành phần

Trong một hệ thống sử dụng giao thức MQTT, nhiều node trạm (gọi là mqtt client - gọi tắt là client) kết nối tới một MQTT server (gọi là broker). Mỗi client sẽ đăng ký một vài kênh (topic), ví dụ như "/client1/channel1", "/client1/channel2". Quá trình đăng ký này gọi là ***"subscribe"****,* giống như chúng ta đăng ký nhận tin trên một kênh Youtube vậy. Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gởi dữ liệu và kênh đã đăng ký. Khi một client gởi dữ liệu tới kênh đó, gọi là ***"publish"****.*

**III. Mô tả chương trình**

1. Thiết kế chương trình

a. Chương trình gồm 2 phần:

- Server (Broker) chờ đợi các kết nối từ client, xử lý các yêu cầu gửi tới client.

- Các client (Publisher / Subscriber) kết nối đến broker, gửi và nhận các message cho server.

- Chương trình quản lý publisher và subscriber theo từng channel.

b. Giao diện:

- Main: Sau khi người dùng nhập tên và chưa channel thì người dùng có thể:

+ Đăng kí channel

+ Yêu cầu chat đôi

+ Chuyển file

+ Tạo nhóm chat

+ Gia nhập nhóm chat đang tồn tại

+ Xem danh sách người dùng và nhóm hiện có.

+ Thoát khỏi chương trình

- Topic: Khi người dùng ở trong nhóm chat nào đó:

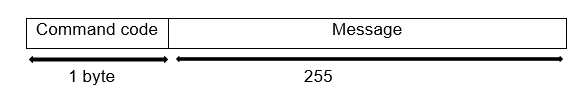
+ Mời người trong nhóm ra chat riêng

+ Xem dannh sách người dùng và nhóm hiện có.

+ Chuyển file

+ Thoát khỏi chương trình

c. Định dạng gói tin



\* Buffer của file: 1024 byte

2. Chạy chương trình

Cú pháp chạy lệnh

Sever: ./server

Client: $ ./client <ip\_server>

Command table:

|  |  |
| --- | --- |
| **Command** | **Nội dung** |
| %channel\_name | Đăng kí channel |
| $user\_name | Mời chat |
| !y | Đồng ý chat |
| !n | Không đồng ý chat |
| !list | Xem danh sách user |
| #file\_name | Gửi file |
| @ | Kết thúc chương trình client |

**3 . Mô tả source code**

Chương trình có 2 struct để mô tả Channel ( kênh), và User ( người dùng )



Chương trình quản lý theo kênh :

+ một kênh sẽ chứa tối đa một lượng MAX\_USR tùy theo khai báo

+ type : thông tin về loại kênh, type = 0 ( public ) hoặc 1 (private) , kênh public là kênh mà bất kì người dùng nào cũng có thể tham gia, kênh private dành cho cặp user muốn chat riêng với nhau

+ useFlag : dùng để kiểm tra kênh này có đang được sử dụng không, useFlag = 1 nếu kênh đang được sử dụng ,0 nếu kênh đang không được sử dụng.

+ Chỉ số cur : chỉ số user đã đạt đến, dùng để xác định chỉ số tối đa mà các hàm gửi tin nhắn hoặc gửi file cần quét qua, mà không cần quét cả list user trong một channel, chỉ số cur ban đầu được đặt bằng 0, sau khi có một user được thêm vào channel, server tìm slot trống từ 0 đến cur -1, nếu không có thì đẩy user vào vị trí cur và tăng cur lên 1.

Struct User gồm có thông tin về user, sockfd, và cờ sử dụng.

Biến toàn cục CH\_TEMP : chỉ số đạt đến của channel, giúp hạn chế chỉ số quét cho các hàm quét kênh như tìm user hoặc tìm chỉ số kênh

**Các hàm sử dụng trong chương trình :**

*Server:*

void listUser ( int index ): list tất cả các user trong tất cả các channel, dùng để debug

void publicMessagetoChannel(int sock ,char message[] , int index) : public một tin nhắn đến tất cả user ( trừ user có sockfd = sock vì user đó là người gửi) của một kênh có chỉ số index, nội dung message

void publicFile(int sock, char \*filename) : gửi file đến user có sockfd sock , phục vụ

cho hàm publicFiletoChannel()

void publicFiletoChannel(int sock , char \*filename, int cur\_index) : gửi file đến tất cả user của một kênh ( trừ user có sockfd = sock)

void subcribeFile(int sock, char \*filename) : nhận file từ user có sockfd bằng sock

void publicList( int sock ) : gửi list user và kênh đến user có sockfd bằng sock

int findIndex () : tìm chỉ số cho kênh chuẩn bị tạo

User getUserbyName(char \*name) : trả về strcut User của user có tên bằng name

void createChannel(int index , char \*channel): tạo một channel mới với chỉ số index và tên channel

void addUserToChannel( User user , int chan\_index ): thêm một user vào kênh có chỉ số chan\_index

void sendInvite(int recv\_sock, char \*send\_name): gửi lời mời từ user send\_name đến user có sockfd bằng recv\_sock

void makeMess(char send[], char \*name, char message[]) : tạo chuỗi string có dạng “tên\_người\_dùng: nội dung tin nhắn”

int getChannelIndexbyName(char \*channel\_name) : tìm kiếm và trả về chỉ số của kênh có tên “channel\_name” , trả về -1 nếu không tìm được

int getUserChannelIndex(char \*username ) : tìm kiếm và trả về chỉ số kênh của user có tên “username” , trả về -1 nếu không tìm được

void clearUser(int channel\_index, char \*username) : xóa user có tên username ra khỏi channel có chỉ số channel\_index

void clearEmptyChannel (): loại bỏ những kênh không có người dùng nào ở trong

Client:  
void \*send\_thread\_func(void \*sockfd) :gửi data tới server, nó là hàm được tạo khi luồng bắt đầu, “sockfd” là con trỏ trỏ tới địa chỉ của socket kết nối với server.

void \*recv\_thread\_func(void \*sockfd) : nhận data từ server, nó là hàm được tạo bởi luồng bắt đầu, “sockfd” là con trỏ trỏ tới địa chỉ của socket kết nối với server.

void sendFile(int sock, char \*filename): gửi file tới server với tên là “filename” và với sock là socket của server.

void recvFile(int sock, char \*filename): nhận file tới server với tên là “filename” và với sock là socket của server.

void rFCaTrim( char str[]) : xóa đi kí tự đầu tiên của chuỗi.

**Thiết kế các tính năng chương trình :**

Mỗi khi user kết nối đến server , user đó sẽ được đẩy vào trong channel PUBLIC là channel đặc biệt, user trong channel này không thể chat được , kênh PUBLIC là kênh có chỉ số bằng 0 trong danh sách kênh

User sử dụng lệnh !list để yêu cầu server trả về các user hiện tại và kênh user đó đang ở trong, server chạy hàm **publicList()** tìm kiếm tất cả user trong tất channel trả về cho user, hàm **publicList()** chỉ quét các kênh có chỉ số < **CH\_TEMP** và trong mỗi kênh chỉ quét list user đến chỉ số **channel.cur** nên giảm được thời gian quét

Khi một user đăng kí một kênh sử dụng lệnh %channel\_name , server tìm kiếm kênh xem kênh đó đã tồn tại chưa bằng hàm **getChannelIndexbyName()** , nếu đã tồn tại thì kiểm tra channel đó là public hay private , private thì không cho phép user đó vào

Nếu channel chưa tồn tại, tạo channel bằng hàm **createChannel()** và thêm user đó vào bằng hàm **addUserToChannel()**

Mỗi khi thêm user vào channel mới, server kiểm tra user đó đã có channel chưa, nếu có remove thông tin user đó trong channel cũ bằng hàm **clearUser()** , sau đó chạy hàm **clearEmptyChannel()** để loại bỏ những channel không còn user nào, việc này để trả lại chỗ cho các kênh hoặc user mới vào tiếp theo

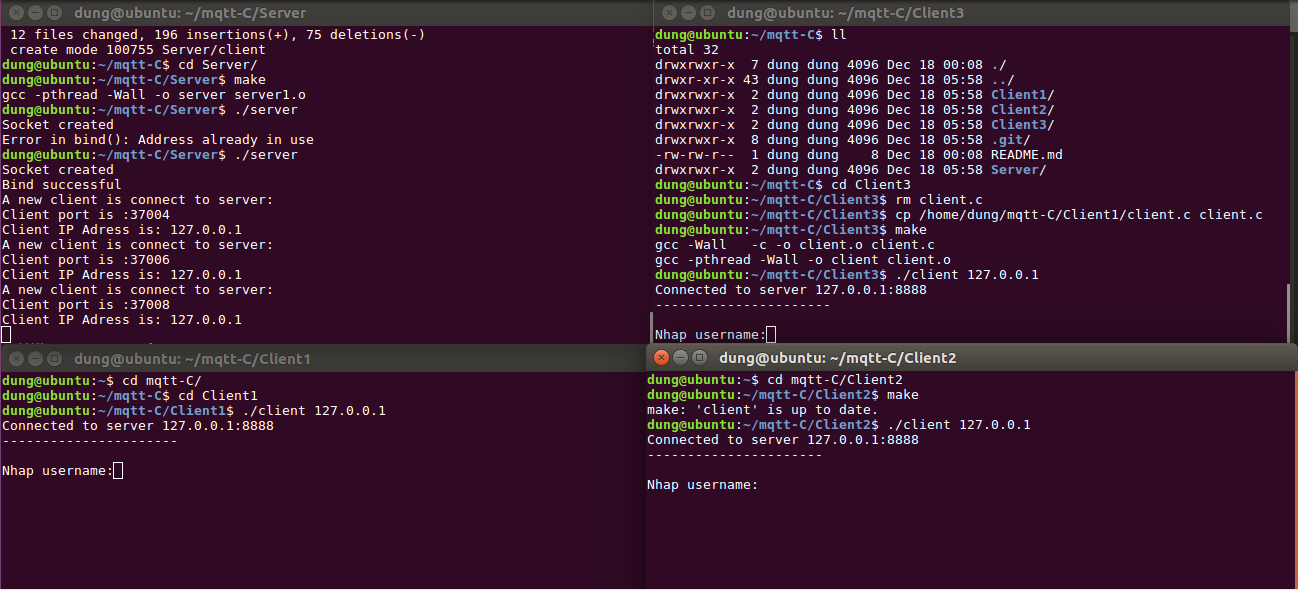
Hàm **clearUser()** và **clearEmptyChannel()** cũng được thực hiện khi user thoát ra

Khi user muốn chat riêng, user gửi lệnh có dạng $username với username là tên user muốn mời đến server, server chuyển tiếp lời mời này đến user được mời bằng hàm **sendInvite()** , tạo một kênh private và add user mời vào kênh đó ,user được mời trả lời bằng các gửi tin nhắn dạng !y ( đồng ý) hoặc !n ( không đồng ý đến server ) server nhận tin nhắn trả lời , nếu là !y thì add user được mời vào kênh của user mời, nếu không thì bỏ qua, kết quả mời được trả về cho user mời.

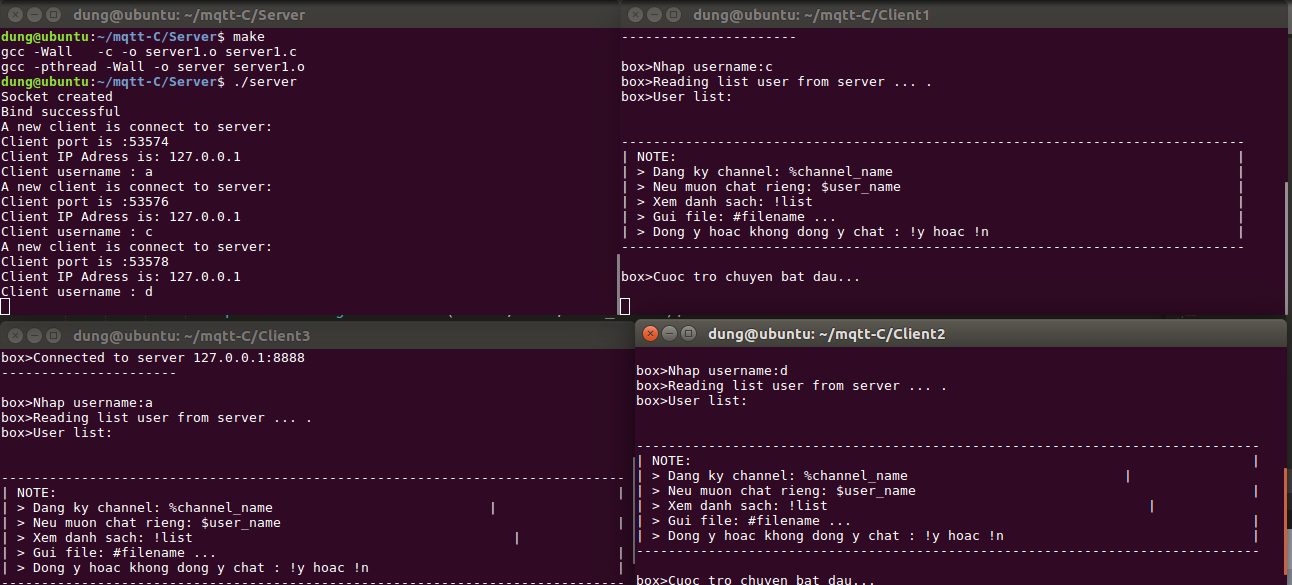
Chức năng gửi và nhận file được thực hiện bằng 2 hàm : **publicFile()** và **publicFileToChannel()** , server nhận file từ client gửi, sau đó chuyển cho user hoặc kênh được nhận

**IV. Demo chương trình**

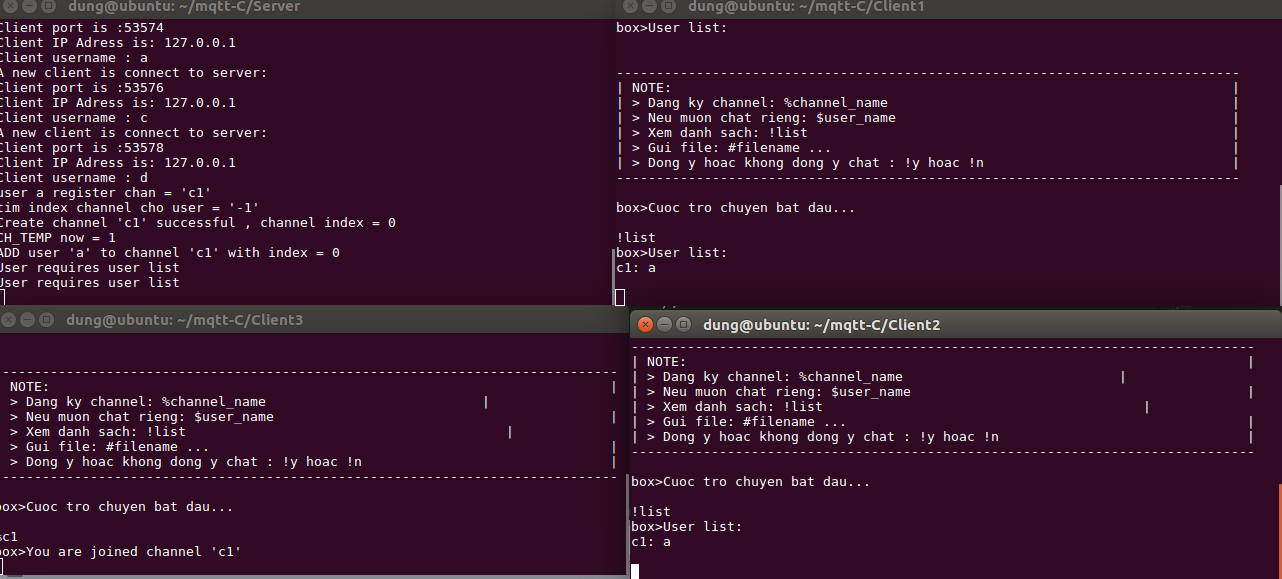
Khởi tạo chương trình:

****

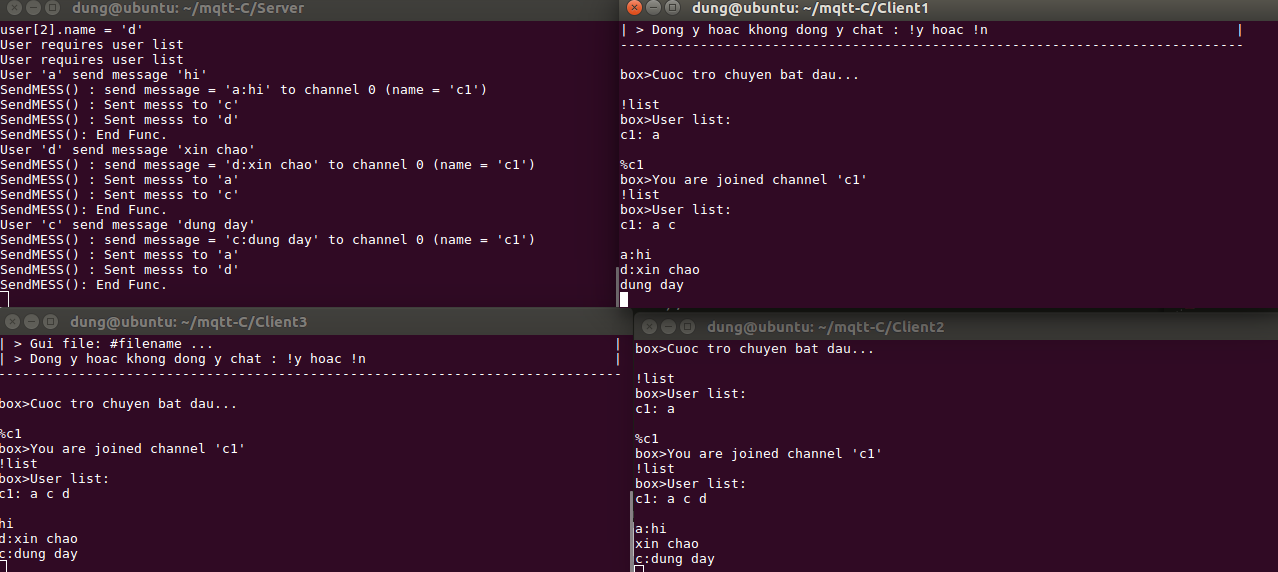
Đăng nhập:



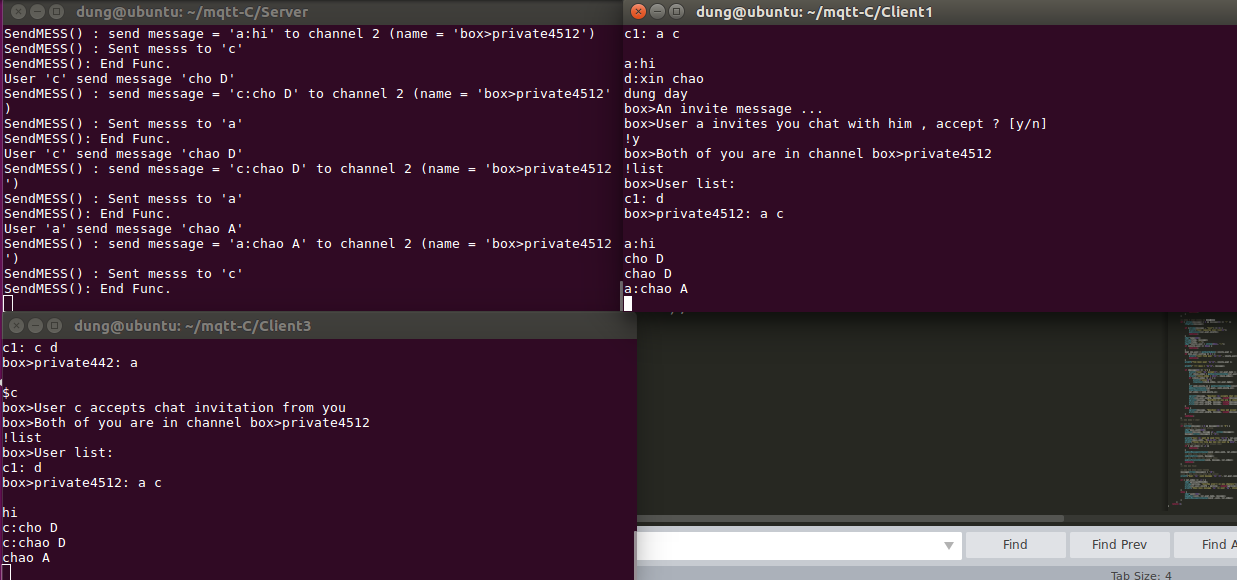
Đăng kí kênh:

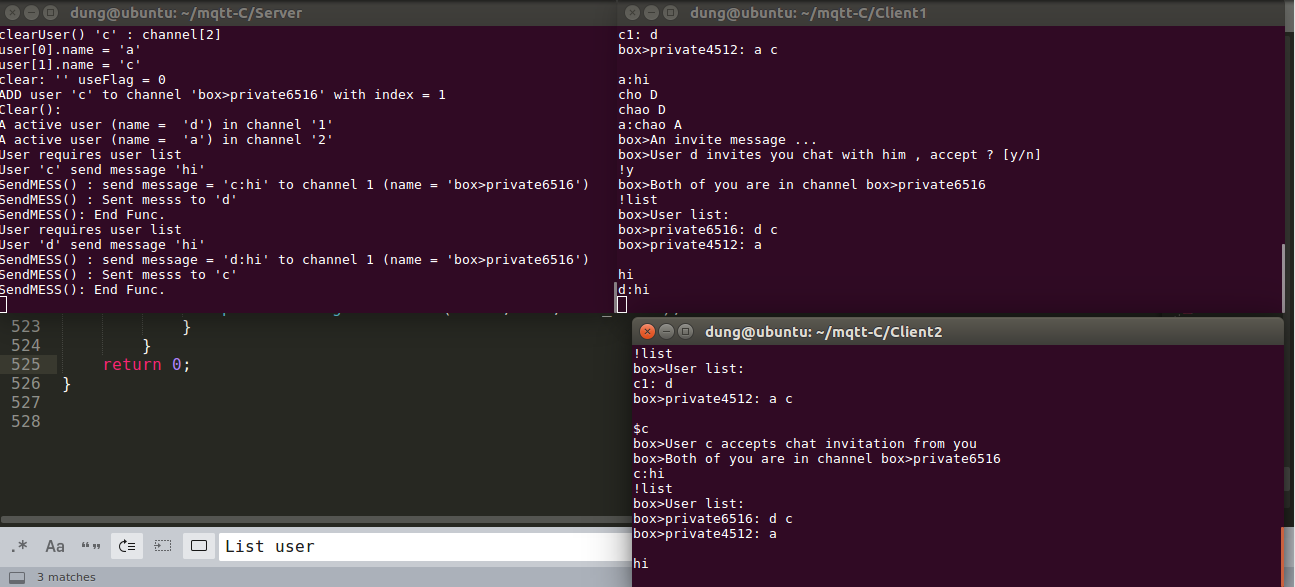


Để chat nhóm thì cả 3 user phải vào cùng một channel:

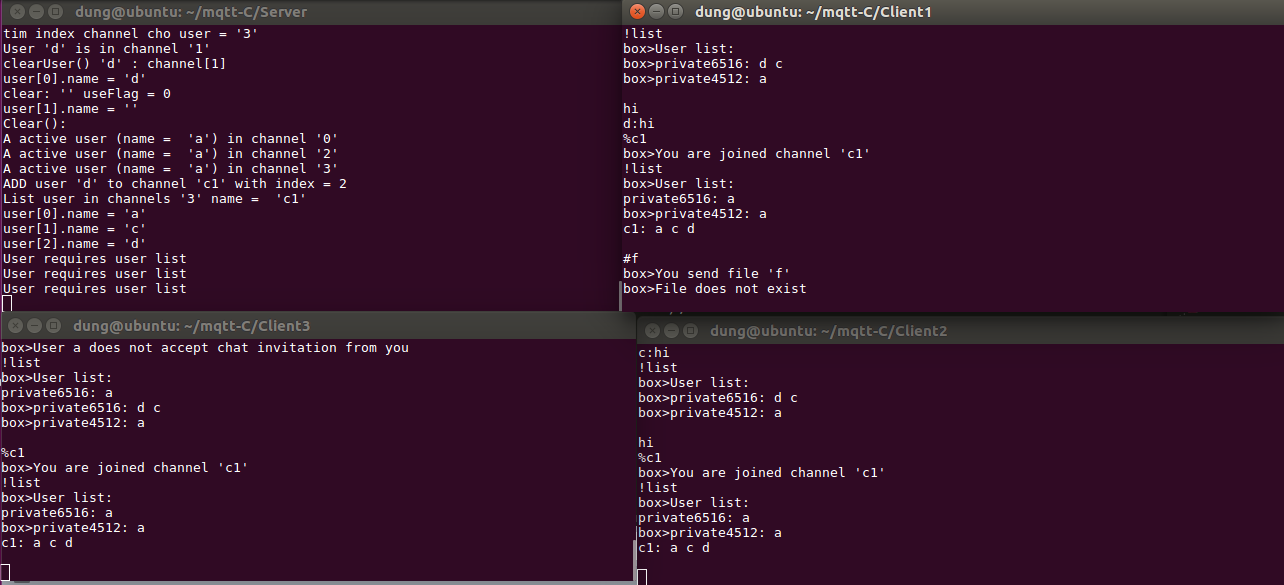


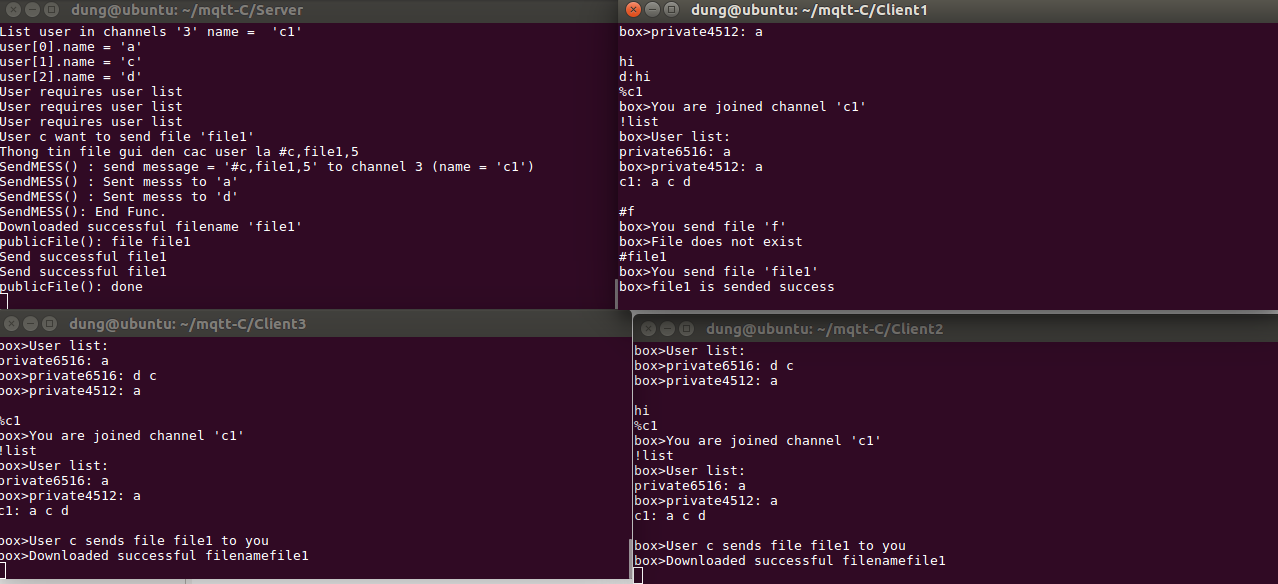
Chat riêng:



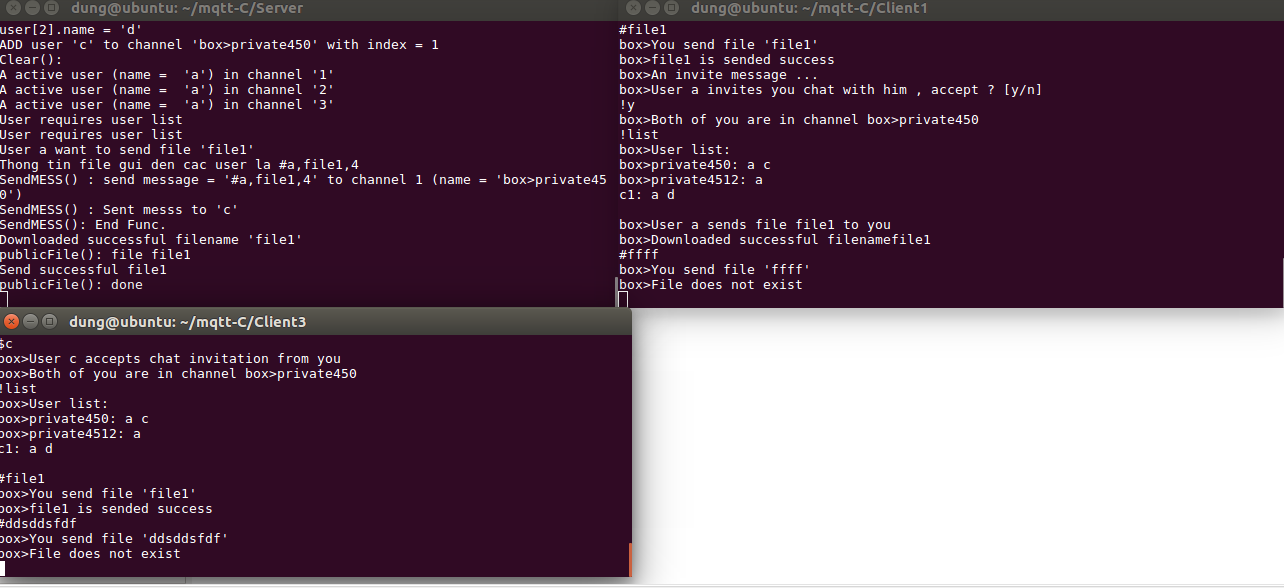


Gửi file:





Gửi file chat riêng:



**V. Ghi chú**

|  |  |
| --- | --- |
| **Tên thành viên** | **Phân chia công việc** |
| Trần Thị Dung | Viết báo cáo, chương trình server |
| Đỗ Xuân Quý | Nêu ý tưởng,viết khung chương trình, chương trình server |
| Đào Tuấn Anh | Chương trình client |

**Link github**: https://github.com/dungtran211096/mqtt-C