**TRƯỜNG ĐẠI HỌC LÂM NGHIỆP**

KHOA KINH TẾ VÀ QUẢN TRỊ KINH DOANH

**===o0o===**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN: LẬP TRÌNH MẠNG**

**ĐỀ TÀI: XÂY DỰNG CHƯƠNG TRÌNH CHAT QUA MẠNG LAN SỬ DỤNG SOCKET TCP**



**Giảng viên hướng dẫn: Ths. Mai Hà An**

**Sinh viên thực hiện: Vì Minh Châu**

**Mã sinh viên: 2041110025**

**Lớp: K65 - HTTT**

**Hà Nội, 2023**

MỤC LỤC

LỜI CẢM ƠN

[MỞ ĐẦU 1](#_Toc137326896)

[PHẦN I: TỔNG QUAN ĐỀ TÀI NGHIÊN CỨU 2](#_Toc137326897)

[1.1. Đặt vấn đề 2](#_Toc137326898)

[1.2. Lý do chọn đề tài 2](#_Toc137326899)

[1.3. Mục đích và ý nghĩa chọn đề tài 3](#_Toc137326900)

[1.4. Phương pháp nghiên cứu 3](#_Toc137326901)

[1.4.1. Về mặt lý thuyết 3](#_Toc137326902)

[1.4.2. Về mặt lập trình 3](#_Toc137326903)

[1.5. Phạm vi, đối tượng đề tài 3](#_Toc137326904)

[PHẦN II: CƠ SỞ LÝ THUYẾT 4](#_Toc137326905)

[2.1. Tổng quan 4](#_Toc137326906)

[2.1.1. Giới thiệu về mô hình Client/Server 4](#_Toc137326907)

[2.1.2. Nguyên lý hoạt động 5](#_Toc137326908)

[2.2. Tổng quan về giao thức TCP/IP 7](#_Toc137326909)

[2.2.1. Giao thức TCP/IP là gì? 7](#_Toc137326910)

[2.2.2. Cách thức hoạt động của TCP/IP 11](#_Toc137326911)

[2.2.3. So sánh TCP và UDP 12](#_Toc137326912)

[2.3. Tìm hiểu về Socket 13](#_Toc137326913)

[2.3.1. Socket là gì? 13](#_Toc137326914)

[2.3.2. Tại sao người dùng lại cần đến Socket 13](#_Toc137326915)

[2.3.3. Socket hoạt động như thế nào? 13](#_Toc137326916)

[2.3.4. Phân loại Socket 13](#_Toc137326917)

[2.3.5. Một số thuật ngữ liên quan đến Socket 14](#_Toc137326918)

[PHẦN III: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 15](#_Toc137326919)

[3.1. Yêu cầu đặt ra 15](#_Toc137326920)

[3.1.1. Yêu cầu đề ra với chương trình 15](#_Toc137326921)

[3.1.2. Yêu cầu đề ra với ứng dụng 15](#_Toc137326922)

[3.2. Xây dựng chương trình 15](#_Toc137326923)

[3.3. Tổng quan về mô hình Client/Server 17](#_Toc137326924)

[PHẦN IV: LẬP TRÌNH VÀ KẾT QUẢ 18](#_Toc137326925)

[4.1. Kết quả và chức năng của chương trình 18](#_Toc137326926)

[4.1.1. Đăng nhập vào hệ thống 18](#_Toc137326927)

[4.1.2. Mô tả chi tiết 19](#_Toc137326928)

[Giao diện Server 19](#_Toc137326928)

[Giao diện Login 22](#_Toc137326928)

[Giao diện Client 24](#_Toc137326928)

[PHẦN V: KẾT LUẬN 29](#_Toc137326929)

**LỜI CẢM ƠN**

Em xin được gửi lời cảm ơn trân thành đến các thầy cô trong ngành công nghệ thông tin Trường đại học lâm nghiệp và đặc biệt là thầy **Mai Hà An** là người đã trực tiếp giảng dạy và hướng dẫn, giúp đỡ về kiến thức và phương pháp nghiên cứu để em có thể hoàn thành đồ án kết thúc môn học Lập trình mạng với đề tài “***Xây dựng chương trình chat qua mạng LAN sử dụng SOCKET TCP***”.

Em xin chân thành cảm ơn!

*Hà Nội, ngày 10 tháng 06 năm 2023*

**Sinh viên thực hiện**

**Vì Minh Châu**

# MỞ ĐẦU

- Ngày nay, nhu cầu về công nghệ thông tin trong đời sống là đa dạng. Việc mở rộng các hệ thống truyền thông và ngày có nhiều máy vi tính kết nối vào mạng Internet. Với việc ứng dụng giao thức TCP/IP làm cho hệ thống mạng ngày càng rộng hơn và phát triển vượt bật. Vấn đề đề an ninh, bảo mật, …là một thế mạnh của giao thức này đem lại cho công nghệ truyền thông.

- Trao đổi thông tin hay trò chuyện với nhau bằng máy tính hay tham gia vào các phòng chat chính là những ví dụ điển hình cho hình thức này. Các room chat là những nơi mà các tài khoản đã đăng ký đăng nhập vào bằng một hình thức nào đó, ví dụ như dùng tài khoản và mật khẩu, ví dụ như các ứng dụng mà chúng ta hay sử dụng hiện nay như Zalo, Facebook hoặc Instagram. Hay đơn giản hơn là gia nhập vào một máy chủ hay còn gọi là một Server bằng địa chỉ IP của chính máy chủ đó. Sau khi gia nhập vào máy chủ ấy thì các tài khoản (còn gọi là Client) sẽ tham gia vào một phòng chat. Từ đó các tài khoản có thể thấy được những tài khoản khác cùng chung phòng chat với mình và có thể trao đổi thông tin với các tài khoản trong chính phòng chat ấy. Mô hình trên được gọi chung là “Mô hình chat Client-Server”. Mô hình này đang được ứng dụng rộng rãi ở các công ty, văn phòng, hoặc là ở các quán net. Mô hình này giúp cho các máy tính trong một văn phòng hay trong một quán net có thể trao đổi thông tin với nhau chỉ bằng cách kết nối với một máy chủ nào gần đó chứ không cần các bước rườm rà như đăng ký tài khoản mật khẩu rồi phải làm các xác nhận mất thời gian.

# PHẦN I: TỔNG QUAN ĐỀ TÀI NGHIÊN CỨU

## . Đặt vấn đề

Trên thực tế, Chat là một ứng dụng trên mạng đã được phổ biến rộng rãi. Một số ứng dụng về Chat phổ biến hiện nay như Messenger, Zalo, Skype, ... Người ta sử dụng các chương trình Chat với mục đích trao đổi thông tin như trò chuyện, làm quen bạn bè, bàn bạc công việc gián tiếp qua môi trường Internet một cách đơn giản để kết nối mọi người lại với nhau. Khi xây dựng một chương trình Chat cần phải đảm bảo các yêu cầu về kết nối, dữ liệu trong quá trình truyền đi phải đảm bảo toàn vẹn và an toàn.

Là một ứng dụng mạng nên các chương trình Chat hoạt động dựa trên mô hình Client – Server hoặc Point to Point và kết nối bằng một trong 2 giao thức TCP hoặc UDP.

## . Lý do chọn đề tài

Như chúng ta đã thấy, hiện nay nhu cầu sử dụng các thiết bị công nghệ thông tin của con người ở mức rất cao. Nhất là về vấn đề trao đổi thông tin. Dựa vào những nhu cầu đó, các phòng chat ảo đã được ra đời. Người dùng từ đó có thể sử dụng máy tính của mình thông qua một hình thức nào đó để gia nhập vào một kênh chat hay là một room chat để có thể thuận tiện trao đổi thông tin với những người dùng ở trong kênh chat ấy.

Hình thức trao đổi thông tin là một mô hình được gọi là mô hình Client-Server (Mô hình chủ-khách). Mô hình này là một mô hình nổi tiếng trong mạng máy tính, được áp dụng rất rộng rãi và là mô hình của mọi trang web hiện có. Ý tưởng của mô hình này là máy khách sẽ gửi một yêu cầu cho máy chủ để máy chủ xử lý và trả kết quả về cho máy khách.

Mô hình này đang được ứng dụng rất rộng rãi trong các công ty cũng như các quán net, nơi có những người dùng cần chia sẻ thông tin một cách bí mật mà không thông qua Internet.

## 1.3. Mục đích và ý nghĩa chọn đề tài

Mục đích của đề tài này chính là tạo ra một ứng dụng chat đơn giản, dễ sử dụng để có thể áp dụng trong môi trường làm việc của các công ty hay văn phòng, đơn giản hơn là các quán net.

Ý nghĩa của đề tài này là người dùng có thể trao đổi thông tin với nhau một cách thoải mái mà không sợ nội dung trao đổi bị rò rỉ. Từ đó người sử dụng sẽ yên tâm và hài lòng hơn khi sử dụng ứng dụng này.

## 1.4. Phương pháp nghiên cứu

### 1.4.1. Về mặt lý thuyết

* Tìm hiểu về giao thức TCP/IP và các giao thức liên quan như UDP/IP...v.v…
* Tìm hiểu về Socket – Khái niệm quan trọng trong lập trình mạng.
* Hiểu được cách thức hoạt động của mô hình Client-Server.

### 1.4.2. Về mặt lập trình

* Sử dụng ngôn ngữ lập trình C# để xây dựng ứng dụng.

## 1.5. Phạm vi, đối tượng đề tài

Phạm vi: Chương trình Chat được xây dựng với khả năng gửi được các văn bản qua lại giữa các User thông qua sự điều khiển của một Server trong mạng Lan.

Đối tượng: Đối tượng mà đề tài hướng đến là các công ty mới thành lập hay các quán net mới mở, nơi cần những ứng dụng chat như thế này.

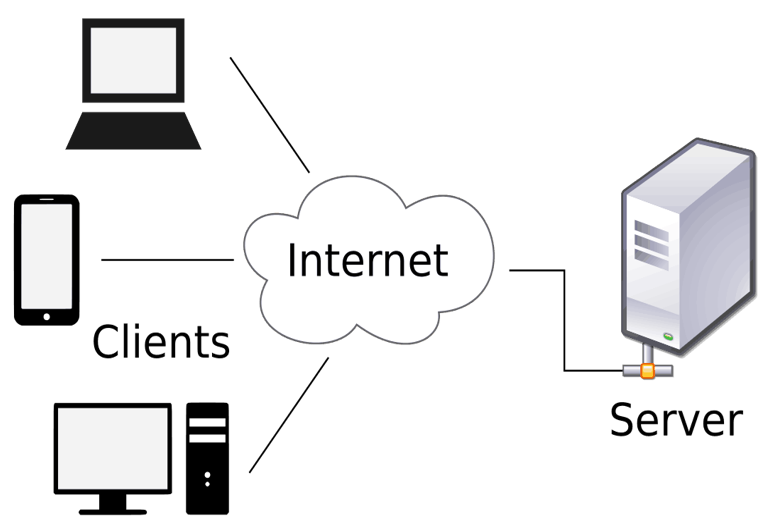
# PHẦN II: CƠ SỞ LÝ THUYẾT

## Tổng quan

### Giới thiệu về mô hình Client/Server

a. Mô hình Client/Server là gì?

Là mô hình tổ chức trao đổi thông tin trong đó mô tả cách mà các máy tính có thể giao tiếp với nhau theo một phương thức nhất định. Phương thức này là một chiến lược tổ chức phân cấp mà trong đó có một máy tính đặc biệt phục vụ các yêu cầu về lưu trữ, xử lý, tính toán tất cả các máy trên mạng. Kiểu tổ chức tổng quát của mô hình này là một mạng LAN được thiết lập từ nhiều máy tính khác nhau, trong đó một máy tính gọi là Server. Một chương trình Client chạy từ bất kỳ máy tính nào trong mạng cũng có thể gửi yêu cầu của mình đến Server, khi Server nhận được các yêu cầu này thì nó sẽ thực hiện và gửi kết quả về cho Client.



Hình 2.1 Mô hình client/server

b. Mô hình Client/Server hoạt động như thế nào?

Các ứng dụng mạng thường hoạt động theo mô hình Client/Server như thư điện tử, truyền nhận tập tin, game trên mạng, ... Mô hình này gồm có một chương trình đóng vai trò là Client và một chương trình đóng vai trò là Server. Hai chương trình này sẽ giao tiếp với nhau thông qua mạng. Chương trình Server đóng vai trò cung cấp dịch vụ. Chương trình này luôn luôn lắng nghe các yêu cầu từ phía Client, rồi tính toán và đáp trả kết quả tương ứng. Chương trình Client cần một dịch vụ và gửi yêu cầu dịch vụ tới chương trình Server và đợi đáp trả từ Server. Như vậy, quá trình trao đổi dữ liệu giữa Client/Server bao gồm:

* Truyền một yêu cầu từ chương trình Client tới chương trình Server.
* Yêu cầu Server xử lý.
* Truyền đáp ứng cho Client.

Client Trong mô hình Client/Server, người ta còn định nghĩa cụ thể cho một máy Client là một máy trạm mà chỉ được sử dụng bởi 1 người dùng để thể hiện tính độc lập cho nó. Máy Client có thể sử dụng các hệ điều hành bình thường như Win9x, DOS, OS/2… Bản thân mỗi một Client cũng đã được tích hợp nhiều chức năng trên hệ điều hành mà nó chạy, nhưng khi được nối vào một mạng LAN, WAN theo mô hình Client/Server thì nó còn có thể sử dụng thêm các chức năng do hệ điều hành mạng (NOS) cung cấp với nhiều dịch vụ khác nhau (cụ thể là các [dịch vụ máy chủ](https://vdodata.vn/thue-may-chu-chay-game-server/) trên mạng này cung cấp), ví dụ như nó có thể yêu cầu lấy dữ liệu từ một Server hay gửi dữ liệu lên Server đó…

Mô hình truyền tin này thực hiện truyền hai thông điệp qua lại giữa Client và

Server một cách đồng bộ hóa. Server nhận thông điệp từ Client thì nó phát ra yêu

cầu Client chuyển sang trạng thái chờ cho tới khi Client nhận được thông điệp đáp

ứng do Server gửi về. Mô hình Client/Server thường được cài đặt dựa trên các thao

tác cơ bản là gửi(Send) và nhận (Receive).

### Nguyên lý hoạt động

Mô hình Client-Server gồm 2 thành phần: Client và Server.

*a. Client*

Khi các máy Client kết nối vào Server thì các Client sẽ nhận được các đoạn thông báo được gửi bởi Server và phát ra âm thanh ở phía Client.

Phần phía Client tổ chức giao tiếp với người dùng, với môi trường bên ngoài tại trạm làm việc với phía Server. Sau khi tiếp nhận yêu cầu của người dùng, phần phía Client thành lập các query string gửi về phía Server, tiếp nhận kết quả và tổ chức trình diễn chúng.

*b. Server*

Server nhận các đoạn chat từ các Client và bắt đầu gửi tin nhắn (dưới dạng Stream) cho các Client khác.

Server sẽ biết được danh sách các Client đang kết nối tới mình (mỗi Client kết nối sẽ được hiển thị lên List tại Server). Server sẽ biết được thông tin chi tiết của từng Client và gửi danh sách nick name cho các Client còn lại.

Phần phía Server quản lý các giao tiếp môi trường bên ngoài tại Server và với các Client, tiếp nhận các yêu cầu dưới dạng các xâu ký tự (query string). Sau khi phân tích các query string, phần phía Server sẽ xử lý dữ liệu và gửi kết quả trả lời về phía các Client.

*c. Tính năng của Client/Server*

Client-server cho phép mạng tập trung các chức năng và các ứng dụng tại một hay nhiều máy dịch vụ file chuyên dụng. Các máy dịch vụ file trở thành trung tâm của hệ thống, cung cấp truy cập tới các tài nguyên và cung cấp sự bảo mật.

Hệ điều hành Client-server cung cấp cơ chế tích hợp tất cả các bộ phận của mạng và cho phép nhiều người dùng đồng thời chia sẻ cùng một tài nguyên, bất kể vị trí địa lý.

*d. Ưu điểm của Client/Server*

Với Client/Server, bạn có thể điều khiển cả tập trung lẫn không tập trung các tài nguyên và bảo mật dữ liệu có thể được điều khiển qua một số máy chuyên dụng. Phân quyền truy nhập tài nguyên, Directory Server, Domain Controller.

Client/Server chống quá tải mạng. Bạn có thể được đảm bảo toàn vẹn dữ liệu trong trường hợp có sự cố xảy ra do tập trung quản lý ở Server.

Tiết kiệm chi phí phát triển các hệ thống ứng dụng phần mềm triển khai trên mạng. Có thể mở rộng (thay đổi) phạm vi (Scale) mạng dễ dàng.

Cung cấp một nền tảng lý tưởng cho phép tích hợp các kỹ thuật hiện đại như mô hình thiết kế hướng đối tượng, hệ chuyên gia, hệ thông tin địa lý (GIS)…

Người dùng có thể truy cập dữ liệu từ xa để thao tác như gửi và nhận file, tìm kiếm thông tin, …

*e. Hạn chế của Client/Server*

Đòi hỏi quá trình bảo trì bảo dưỡng Server.

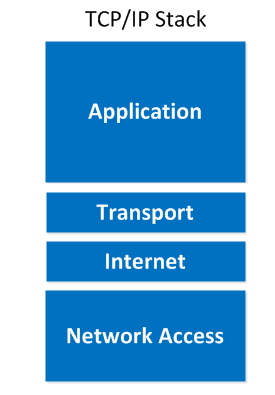
Do nguyên lí làm việc phải trao đổi dữ liệu giữa hai máy ở hai khu vực địa lý khác nhau, Client/Server dễ dàng xảy ra hiện tượng thông tin truyền trên mạng bị lộ. Tính an toàn và bảo mật thông tin trên mạng là hạn chế mà Client/server cần khắc phục.

## Tổng quan về giao thức TCP/IP

### 2.2.1. Giao thức TCP/IP là gì?

**TCP/IP** là tên viết tắt của cụm từ **Transmission Control Protocol/Internet Protocol**, là một tập hợp các giao thức (protocol) trao đổi thông tin được sử dụng để truyền tải và kết nối các thiết bị trong mạng Internet. TCP/IP là tên chung cho một tập hợp hơn 100 giao thức được sử dụng để kết nối các máy tính vào mạng, trong đó hai giao thức chính là TCP và IP.

Trong phạm vi Internet, thông tin được truyền tải như một dòng riêng biệt từ máy tính này tới máy tính khác. Thay vào đó, dữ liệu được chia thành những gói nhỏ gọi là packet. Các packet này được gửi trên mạng máy tính. Công việc của IP là chuyển chúng đến các máy tính ở xa. Tại trạm cuối, TCP nhận các packet và kiểm tra lỗi. Nếu một lỗi xuất hiện, TCP yêu cầu gói riêng biệt đó phải được gửi lại. Chỉ khi tất cả các packet đã nhận được là đúng, TCP sẽ sử dụng số thứ tự để tạo lại thông tin ban đầu.

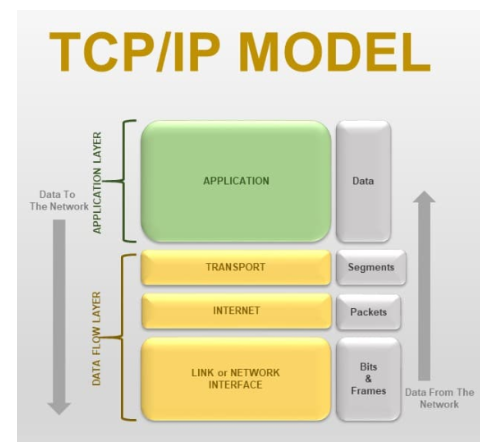
Hình 2.2 Giao thức TCP/IP

Mô hình TCP/IP tiêu chuẩn bao gồm 4 tầng được chồng lên nhau là:

* Tầng 1: Tầng vật lý (Network Access)
* Tầng 2: Tầng mạng (Internet)
* Tầng 3: Tầng giao vận (Transport)
* Tầng 4: Tầng ứng dụng (Application).

Tầng 4 - Tầng Ứng dụng (Application)

Đây là lớp giao tiếp trên cùng của mô hình. Đúng với tên gọi, tầng Ứng dụng đảm nhận vai trò giao tiếp dữ liệu giữa 2 máy khác nhau thông qua các dịch vụ mạng khác nhau (duyệt web, chat, gửi email, một số giao thức trao đổi dữ liệu: SMTP, SSH, [FTP](https://www.totolink.vn/article/154-giao-thuc-ftp-la-gi-nhung-dieu-ban-chua-biet-ve-giao-thuc-ftp.html),...). Dữ liệu khi đến đây sẽ được định dạng theo kiểu Byte nối Byte, cùng với đó là các thông tin định tuyến giúp xác định đường đi đúng của một gói tin.

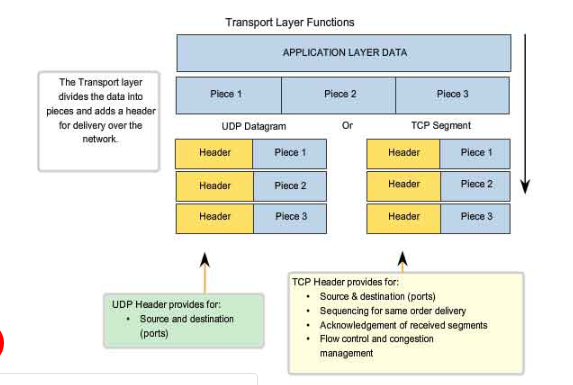


Hình 2.3 Tầng ứng dụng

Tầng 3 - Tầng Giao vận (Transport)

Chức năng chính của tầng 3 là xử lý vấn đề giao tiếp giữa các máy chủ trong cùng một mạng hoặc khác mạng được kết nối với nhau thông qua bộ định tuyến. Tại đây dữ liệu sẽ được phân đoạn, mỗi đoạn sẽ không bằng nhau nhưng kích thước phải nhỏ hơn 64KB. Cấu trúc đầy đủ của một Segment lúc này là Header chứa thông tin điều khiển và sau đó là dữ liệu.

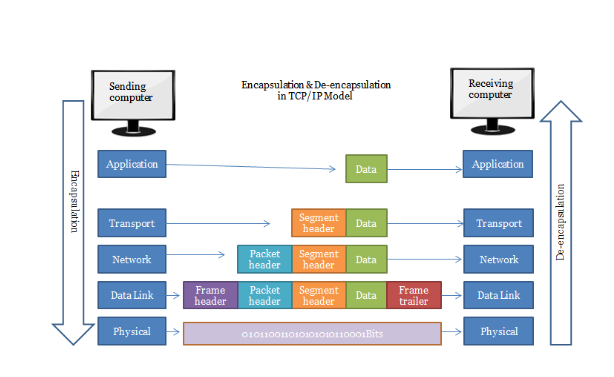
Trong tầng này còn bao gồm 2 giao thức cốt lõi là TCP và UDP. Trong đó, TCP đảm bảo chất lượng gói tin nhưng tiêu tốn thời gian khá lâu để kiểm tra đầy đủ thông tin từ thứ tự dữ liệu cho đến việc kiểm soát vấn đề tắc nghẽn lưu lượng dữ liệu. Trái với điều đó, UDP cho thấy tốc độ truyền tải nhanh hơn nhưng lại không đảm bảo được chất lượng dữ liệu được gửi đi.



Hình 2.4 Tầng giao vận

Tầng 2 - Tầng mạng (Internet)

Gần giống như tầng mạng của [mô hình OSI](http://www.totolink.vn/article/136-mo-hinh-osi-la-gi-chuc-nang-cua-cac-tang-giao-thuc-trong-mo-hinh-osi.html). Tại đây, nó cũng được định nghĩa là một giao thức chịu trách nhiệm truyền tải dữ liệu một cách logic trong mạng. Các phân đoạn dữ liệu sẽ được đóng gói (Packets) với kích thước mỗi gói phù hợp với mạng chuyển mạch mà nó dùng để truyền dữ liệu. Lúc này, các gói tin được chèn thêm phần Header chứa thông tin của tầng mạng và tiếp tục được chuyển đến tầng tiếp theo. Các giao thức chính trong tầng là IP, [ICMP](https://www.totolink.vn/article/73-icmp-la-gi-9-loai-icmp-thuong-thay.html) và ARP.

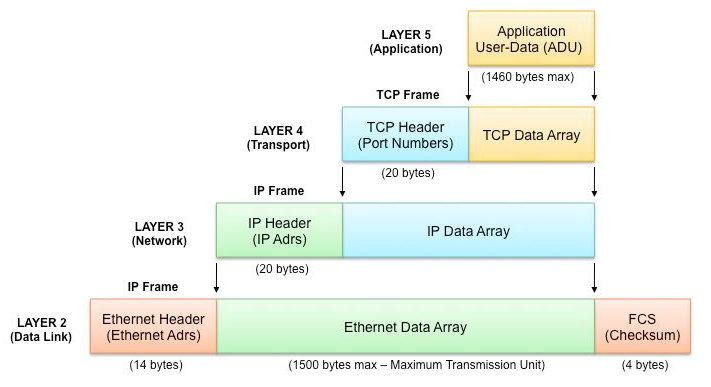


Hình 2.5 Tầng mạng

Tầng 1 - Tầng Vật lý (Physical)

Là sự kết hợp giữa tầng Vật lý và tầng liên kết dữ liệu của mô hình OSI. Chịu trách nhiệm truyền dữ liệu giữa hai thiết bị trong cùng một mạng. Tại đây, các gói dữ liệu được đóng vào khung (gọi là Frame) và được định tuyến đi đến đích đã được chỉ định ban đầu.

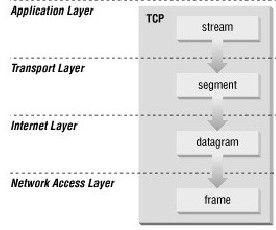
### Cách thức hoạt động của TCP/IP



Hình 2.6 Cách thức hoạt động TCP/IP

Khi truyền dữ liệu , quá trình tiến hành từ tầng trên xuống tầng dưới, qua mỗi tầng dữ liệu được thêm vào thông tin điều khiển gọi là Header. Khi nhận dữ liệu thì quá trình xảy ra ngược lại, dữ liệu được truyền từ tầng dưới lên và qua mỗi tầng thì phần header tương ứng sẽ được lấy đi và khi đến tầng trên cùng thì dữ liệu không còn phần header nữa.

* Ở đây, **IP** có vai trò quan trọng, nó cho phép các gói tin được gửi đến đích đã định sẵn, bằng cách thêm các thông tin dẫn đường (chính là Header) vào các gói tin để các gói tin được đến đúng đích đã định sẵn ban đầu.
* Giao thức **TCP** đóng vai trò kiểm tra và đảm bảo sự an toàn cho mỗi gói tin khi đi qua mỗi trạm. Trong quá trình này, nếu giao thức TCP nhận thấy gói tin bị lỗi, một tín hiệu sẽ được truyền đi và yêu cầu hệ thống gửi lại một gói tin khác.



Hình 2.7 Cấu trúc dữ liệu

Hình trên là cấu trúc dữ liệu qua các tầng. Trong hình mọi người sẽ thấy ở mỗi tầng khác nhau dữ liệu được truyền vào là khác nhau.

* Tầng ứng dụng: dữ liệu là các luồng được gọi là stream.
* Tầng giao vận: đơn vị dữ liệu mà TCP gửi xuống gọi là TCP segment.
* Tầng mạng: dữ liệu mà IP gửi xuống tầng dưới gọi là IP Datagram.
* Tầng liên kết: dữ liệu được truyền đi gọi là frame.

### So sánh TCP và UDP

Giống nhau: đều là các giao thức mạng TCP/IP, đều có chức năng kết nối các máy lại với nhau, và có thể gửi dữ liệu cho nhau.

Khác nhau (cơ bản): Các header của TCP và UDP khác nhau ở kích thước (20 và 8 byte) nguyên nhân chủ yếu là do TCP phải hộ trợ nhiều chức năng hữu ích hơn (như khả năng khôi phục lỗi). UDP dùng ít byte hơn cho phần header và yêu cầu xử lý từ host ít hơn.

*Bảng: Điểm khác nhau giữa TCP và UDP*

|  |  |
| --- | --- |
| **TCP** | **UDP** |
| * Dùng cho mạng WAN, không cho phép mất gói tin, đảm bảo việc truyền dữ liệu, tốc độ truyền thấp. * TCP hoạt động theo hướng kết nối, trước khi truyền dữ liệu giữa 2 máy, nó thiết lập một kết nối giữa 2 máy theo phương thức “bắt tay 3 bước   (three-way-hand-shake)”  => Truyền dữ liệu chậm hơn UDP nhưng đáng tin cậy hơn UDP | * Dùng cho mạng LAN, cho phép mất dữ liệu, không đảm bảo. Tốc độ truyền cao, VolP truyền tốt qua UDP. * UDP hoạt động theo hướng ko kết nối, không yêu cầu thiết lập kết nối giữa 2 máy gửi và nhận, ko có sự đảm bảo gói tin khi truyền đi cũng như ko thông báo về việc mất gói tin, ko kiểm tra lỗi của gói tin.   => Truyền dữ liệu nhanh hơn TCP tuy nhiên lại ko đáng tin cậy bằng TCP |

## Tìm hiểu về Socket

### 2.3.1. Socket là gì?

Socket là một điểm cuối (end-point) của liên kết truyền thông hai chiều (two-way communication) giữa hai chương trình chạy trên mạng. Các lớp Socket được sử dụng để biểu diễn kết nối giữa client và server, được ràng buộc với một cổng port (thể hiện là một con số cụ thể) để các tầng TCP (TCP Layer) có thể định danh ứng dụng mà dữ liệu sẽ được gửi tới.

Lập trình Socket là lập trình cho phép người dùng kết nối các máy tính truyền tải và nhận dữ liệu từ máy tính thông qua mạng. Hiểu đơn giản, Socket là thiết bị truyền thông hai chiều gửi và nhận dữ liệu từ máy khác.

### Tại sao người dùng lại cần đến Socket

Sockets cung cấp một interface để lập trình mạng tại tầng Transport. Một socket là một end-point của một liên kết giữa hai ứng dụng. Ngày nay, Socket được hỗ trợ trong hầu hết các hệ điều hành như MS Windows (WinSock), Linux và được sử dụng trong nhiều ngôn ngữ lập trình khác nhau: như C, C++, Java, Visual Basic, C#,…

### Socket hoạt động như thế nào?

Là một giao diện lập trình ứng dụng mạng, Socket giúp bạn lập trình kết nối các ứng dụng để truyền và nhận giữ liệu trong môi trường có kết nối Internet bằng cách sử dụng phương thức TCP IP và UDP.

Khi cần trao đổi dữ liệu cho nhau thì 2 ứng dụng cần phải biết thông tin tối thiểu là IP và số hiệu cổng của ứng dụng kia, 2 ứng dụng có thể nằm cùng trên một máy không được cùng số hiệu cổng.

### 2.3.4. Phân loại Socket

a. Stream Socket

Dựa trên giao thức TCP (Tranmission Control Protocol), việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình đã thiết lập kết nối. Do đó, hình thức này được gọi là Socket hướng kết nối.

Ưu điểm: Có thể dùng để liên lạc theo mô hình Client và Server. Nếu là mô hình Client-Server thì Server lắng nghe và chấp nhận từ Client. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn. Đồng thời, mỗi thông điệp gửi phải có xác nhận trả về và các gói tin chuyển đi tuần tự.

Hạn chế: Có một đường kết nối (địa chỉ IP) giữa 2 tiến trình nên 1 trong 2 tiến trình kia phải đợi tiến trình kia yêu cầu kết nối.

b. Datagram Socket

Dựa trên giao thức UDP (User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 quá trình. Do đó, hình thức này được gọi là Socket không hướng kết nối.

Ưu điểm: Do không yêu cầu thiết lập kết nối, không phải có những cơ chế phức tạp nên tốc độ giao thức khá nhanh, thuận tiện cho các ứng dụng truyền dữ liệu nhanh như chat, game…..

Hạn chế: Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thế không đúng trình tự và lặp lại.

### 2.3.5. Một số thuật ngữ liên quan đến Socket

a. Web Socket

WebSocket là công nghệ hỗ trợ giao tiếp hai chiều thông qua việc sử dụng TCP Socket, để tạo ra một kết nối hiệu quả ít tốn kém giữa Client và Server.

WebSocket không chỉ được sử dụng cho các ứng dụng web, mà người dùng có thể sử dụng chúng trong bất kì loại ứng dụng nào.

b. Unix Socket

Unix socket dùng để trao đổi dữ liệu giữa các ứng dụng trên cùng máy tính. Unix socket có khả năng tránh được các bước kiểm tra hoặc routing, do đó trong đầu kết nối rất nhanh và nhẹ nhàng hơn so với giao thức TCP.

Tuy nhiên Unix Socket có nhược điểm đó là không thể kết nối hai ứng dụng được với nhau. Việc phân quyền có thể gây đến nhiều rắc rối cho bạn, do bản chất của Unix Socket chỉ là một tập tin trên máy chủ.

# PHẦN III: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 3.1. Yêu cầu đặt ra

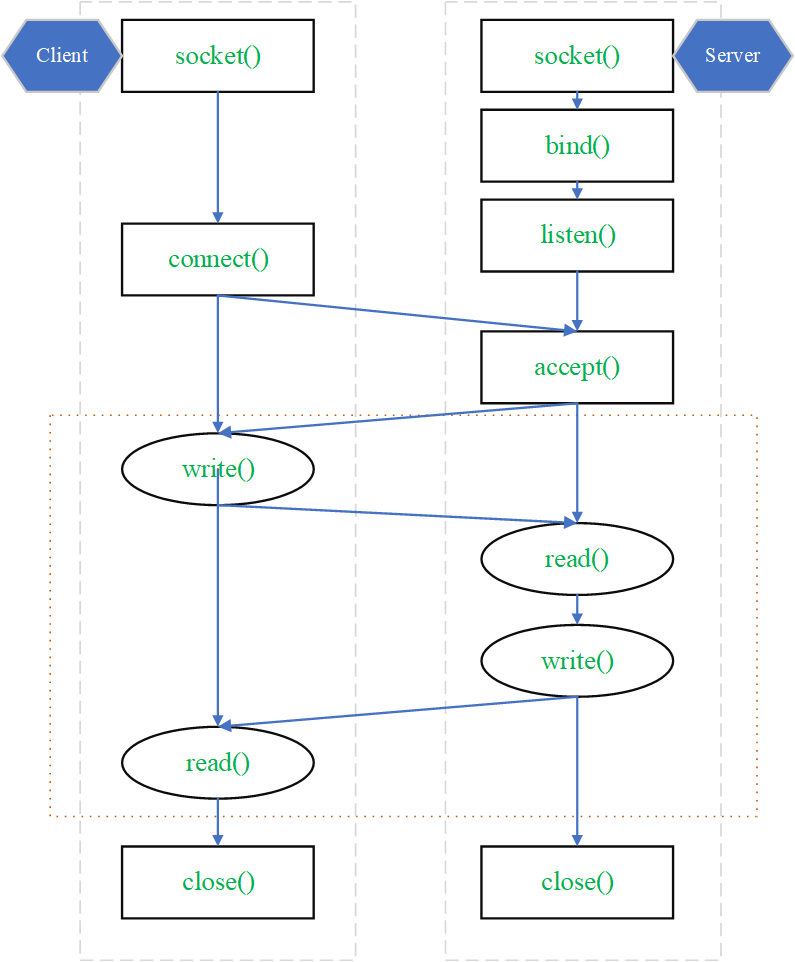
### 3.1.1. Yêu cầu đề ra với chương trình

* Dùng ngôn ngữ C# để xây dựng 1 trương trình Chat qua mạng theo mô hình Client/Server.
* Tạo ra được 1 chương trình mà các Client kết nối được với nhau thông qua Server.
* Các Client có thể gửi tin nhắn, hoặc trò chuyện trực tiếp với các Client khác.

### 3.1.2. Yêu cầu đề ra với ứng dụng

* Server có khả năng phục vụ đồng thời nhiều người chat cùng lúc.
* Thiết kế đơn giản, giao diện thân thiện.
* Có khả năng Chat cơ bản bằng chữ.
* Các Client có thể chat với nhau qua IP của Server.
* Có khả năng chat phòng (Room Chat).

## 3.2. Xây dựng chương trình



Hình 3.1 Chat Client/Server

Các giai đoạn của mô hình Chat Client-Server bằng giao tức TCP có thể phân thành 4 giai đoạn như sau:

Giai đoạn 1: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu kết nối. Server sẵn sàng phục vụ Client.Socket(): Server yêu cầu tạo một Socket để có thể sử dụng các dịch vụ của tầng vận chuyển.

* Bind (): Server yêu cầu gán số hiệu cổng (port) cho Socket.
* Listen (): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.

Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.

Socket (): Client yêu cầu tạo một Socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho Socket của Client.

* Connect (): Client gởi yêu cầu nối kết đến Server có địa chỉ IP và Port xác định.
* Accept (): Server chấp nhận nối kết của Client, khi đó một kênh giao tiếp ảo được hình thành, Client và Server có thể trao đổi thông tin với nhau thông qua kênh ảo này.

Giai đoạn 3: Trao đổi thông tin giữa Client và Server.

Sau khi chấp nhận yêu cầu kết nối, thông thường Server thực hiện lệnh read () và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ Client gửi đến.

* Server phân tích và thực thi yêu cầu. Kết quả sẽ được gửi về Client bằng lệnh write ().
* Sau khi gửi yêu cầu bằng lệnh write (), Client chờ nhận thông điệp kết quả (ReplyMessage) từ Server bằng lệnh read ().

Giai đoạn 4: Kết thúc phiên làm việc. Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).

* Kênh ảo sẽ bị xóa khi Server hoặc Client đóng Socket bằng lệnh close().

## 3.3. Tổng quan về mô hình Client/Server

* Sử dụng mô hình mạng Client/Server, Giao thức TCP/IP.
* Sử dụng Socket và Thread trong C#, và một số tính năng khác.
* Sử dụng ngôn ngữ: lập trình hướng đối tượng trong C#.
* Server có khả năng phục vụ đồng thời nhiều người chat cùng lúc.
* Thiết kế đơn giản, giao diện thân thiện.
* Có khả năng Chat cơ bản.
* Có khả năng chat phòng (Room Chat).

a.Chương trình thực hiện ở Server:

-  Làm trung gian để nối kết giữa các Client với nhau.

b**.** Chương trình thực hiện ở Client:

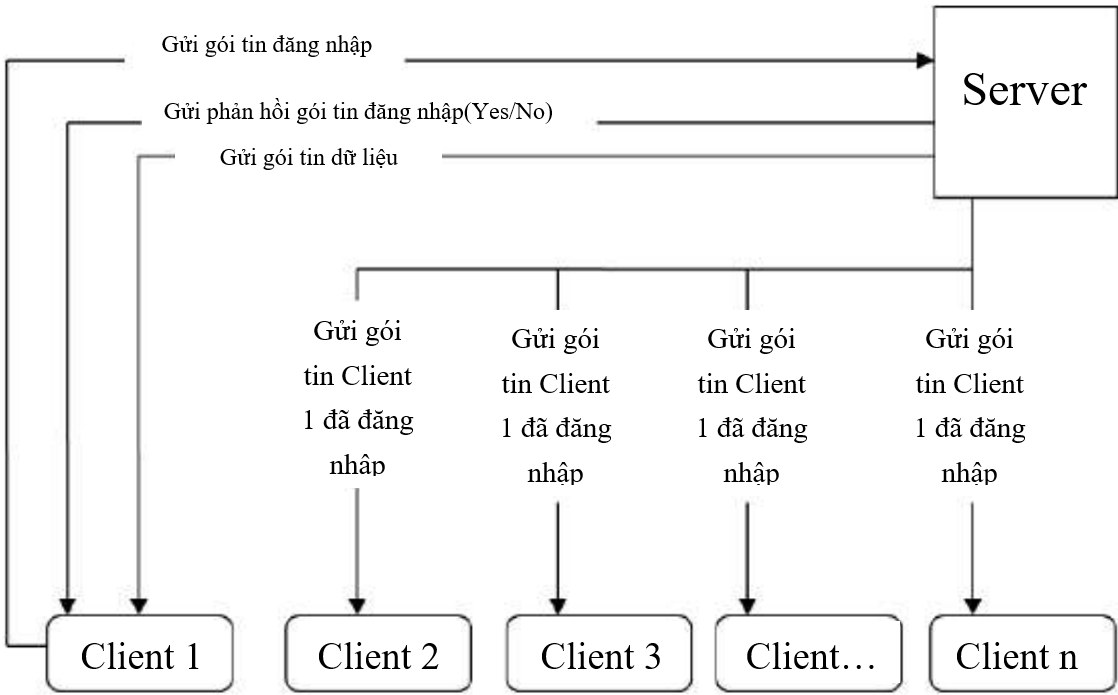
 - Thực hiện các công việc của ứng dụng chat cho người dùng cuối như: Chat phòng.

# PHẦN IV: LẬP TRÌNH VÀ KẾT QUẢ

## 4.1. Kết quả và chức năng của chương trình

### 4.1.1. Đăng nhập vào hệ thống

* Yêu cầu: Hệ thống yêu cầu người dùng cung cấp thông tin đăng nhập là tên đăng nhập. Nhập xong bấm nút đăng nhập. Hệ thống kiểm tra và thông báo đăng nhập thành công/ thất bại. Nếu thành công, hiển thị giao diện phần mềm. Nếu thất bại, yêu cầu đăng nhập lại.
* Quá trình đăng nhập:



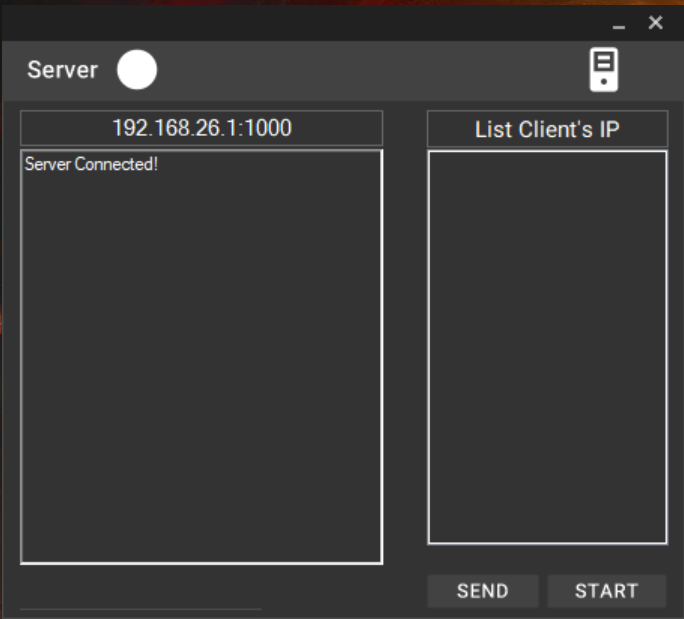
* Mô tả: Khi một User yêu cầu đăng nhập hệ thống, Client sẽ gửi Username cho Server. Sau đó, Server sẽ kiểm tra Username này đã đăng nhập chưa.
* Kết nối giữa máy chủ và máy khách:

Khi Client A muốn gửi tin nhắn đến client B, Client A sẽ phát sự kiện gửi tin nhắn cùng với dữ liệu lên Server, Server lắng nghe sự kiện từ Client A sau đó phát sự kiện gửi tin nhắn cùng với dữ liệu từ Server xuống Client B, Client B lắng nghe sự kiện từ Server sau đó nhận được dữ liệu từ Server.

### 4.1.2. Mô tả chi tiết

### Giao diện Server

- Nhập port cho Server và khởi động (Start) máy chủ, phía Server hiện thông báo máy chủ đã kết nối (Server Connected!):



using MaterialSkin.Controls;

using System;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using TCP\_IP\_Server.ViewController;

using SimpleTcp;

namespace TCP\_IP\_Server.View

{

public partial class FrmMain : MaterialForm

{

SimpleTcpServer server;

string \_server;

public FrmMain()

{

InitializeComponent();

}

private void FrmMain\_Load(object sender, EventArgs e)

{

\_server = ServerConfig.ConfigurationServer();

ControlForms();

ControlButtons(false);

}

private void BtnStart\_Click(object sender, EventArgs e)

{

StartServer();

}

private void Events\_ClientDisconnected(object sender, ClientDisconnectedEventArgs e)

{

UserDisconnect(e);

}

private void Events\_ClientConnected(object sender, ClientConnectedEventArgs e)

{

NewUserConnected(e);

}

private void Events\_DataReceived(object sender, DataReceivedFromClientEventArgs e)

{

Invoke((MethodInvoker)delegate

{

ReceiveMessages(e);

});

}

private void CloseServerToolStripMenuItem\_Click(object sender, EventArgs e)

{

CloseServer();

}

private void TxtText\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Enter)

{

SendMessenger();

}

}

private void BtnSend\_Click\_1(object sender, EventArgs e)

{

SendMessenger();

}

private void StartServer()

{

try

{

ConfigurationServer();

server.Start();

cmsCloseServer.Enabled = true;

rtbMessages.Text += $"Server Connected!{Environment.NewLine}";

ControlButtons(true);

txtText.Focus();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

rtbMessages.Text += $"Server startup failed :({Environment.NewLine}";

}

}

private void ReceiveMessages(DataReceivedFromClientEventArgs e)

{

this.Invoke((MethodInvoker)delegate

{

string ReceiveMessages = $"{Encoding.UTF8.GetString(e.Data)}";

rtbMessages.Text += ReceiveMessages;

UserUpdate(ReceiveMessages, e.IpPort);

SendMessageAllUsers(ReceiveMessages, e.IpPort);

});

}

private void UserDisconnect(ClientDisconnectedEventArgs e)

{

this.Invoke((MethodInvoker)delegate

{

int index = User.FindString(e.IpPort);

rtbMessages.Text += $"{User.Items[index]} disconnected.{Environment.NewLine}";

User.Items.RemoveAt(index);

});

}

private void NewUserConnected(ClientConnectedEventArgs e)

{

this.Invoke((MethodInvoker)delegate

{

rtbMessages.Text += $"{e.IpPort} connected.{Environment.NewLine}";

User.Items.Add(e.IpPort);

});

}

private void ControlForms()

{

txtServer.Text = \_server;

rtbMessages.ReadOnly = true;

rtbMessages.SelectionProtected = true;

rtbMessages.BackColor = BackColor;

rtbMessages.ForeColor = Color.Black;

User.BackColor = BackColor;

User.ForeColor = Color.Black;

btnSend.Enabled = false;

}

private void ControlButtons(bool status)

{

if (status == true) { btnStart.Enabled = false; }

else { btnStart.Enabled = true; }

btnSend.Enabled = status;

}

public void SendMessenger()

{

if (VerifyServer())

{

string messenger = $"Server: {txtText.Text}{Environment.NewLine}";

string[] \_user = User.SelectedItem.ToString().Split('(');

string destiny = \_user[0].Trim();

server.Send(destiny, messenger);

rtbMessages.Text += messenger;

txtText.Text = "";

}

}

public void SendMessageAllUsers(string messenger, string ipPort)

{

if (server.IsListening)

{

foreach (string user in User.Items)

{

string[] \_user = user.Split('(');

if (!ipPort.Equals(\_user[0].Trim()))

{

server.Send(\_user[0].Trim(), messenger);

}

}

}

}

public bool VerifyServer()

{

if (server.IsListening)

{

if (!string.IsNullOrEmpty(txtText.Text) && User.SelectedItem != null)

{

return true;

}

}

return false;

}

public void ConfigurationServer()

{

server = new SimpleTcpServer(txtServer.Text);

server.Events.ClientConnected += Events\_ClientConnected;

server.Events.ClientDisconnected += Events\_ClientDisconnected;

server.Events.DataReceived += Events\_DataReceived;

}

private void UserUpdate(string messageReceived, string ipPort)

{

string[] user = messageReceived.Split(':');

int index = User.FindString(ipPort);

if (!User.Items[index].ToString().Contains("("))

{

User.Items[index] = User.Items[index].ToString() + " (" + user[0].Replace(":", ",") + ")";

}

}

private void CloseServer()

{

DisconnectClient();

server.Stop();

User.Items.Clear();

rtbMessages.Text += $"Server has stopped!{ Environment.NewLine}";

btnStart.Enabled = true;

}

private void DisconnectClient()

{

string[] users = server.GetClients().ToArray();

foreach(string user in users)

{

server.DisconnectClient(user);

}

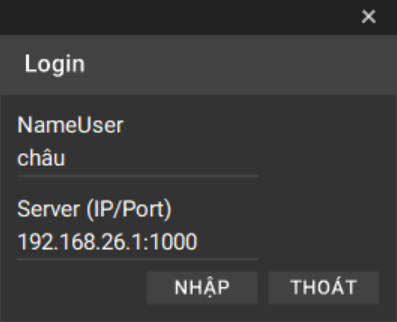
}

}

}

### Giao diện Login

- Nhập tên người dùng (NameUser), địa chỉ IP, và port của Server muốn đăng nhập:



using MaterialSkin.Controls;

using System;

using System.Windows.Forms;

using TCP\_IP\_Chat.Controller;

namespace TCP\_IP\_Chat

{

public partial class FrmLogin : MaterialForm

{

public string nickname;

public string server;

public FrmLogin(string nickname)

{

InitializeComponent();

this.nickname = nickname;

}

private void FrmLogin\_Load(object sender, EventArgs e)

{

txtServer.Text = ClientConfig.IPHost();

txtNickName.Text = nickname;

}

private void BtnEnter\_Click(object sender, EventArgs e)

{

bool choose = VerifyInformation();

if (choose == true)

{

nickname = txtNickName.Text;

server = txtServer.Text;

DialogResult = DialogResult.OK;

Close();

};

}

private void BtnExit\_Click(object sender, EventArgs e)

{

Close();

}

private bool VerifyInformation()

{

foreach(Control control in Controls)

{

if(control is MaterialSingleLineTextField)

{

if (string.IsNullOrEmpty(control.Text))

{

MessageBox.Show("Không được để trống!", "Attention",MessageBoxButtons.OK,MessageBoxIcon.Error);

return false;

}

}

}

return true;

}

private void TxtNickName\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Enter)

{

BtnEnter\_Click(null,null);

}

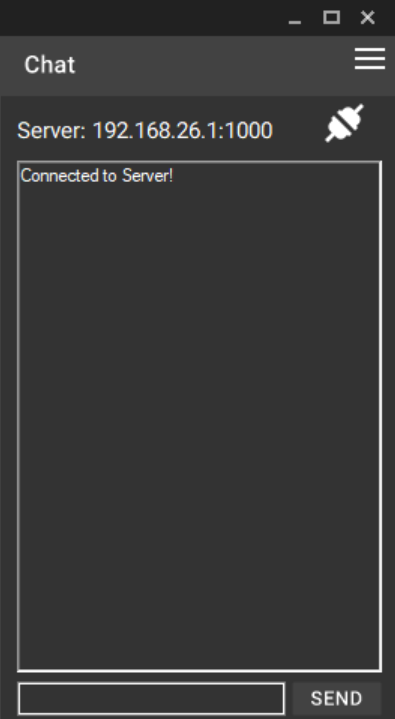
}

}

}

### Giao diện Client

- Sau khi đăng nhập thành công sẽ vào giao diện của phòng chat:



using MaterialSkin.Controls;

using SimpleTcp;

using System;

using System.ComponentModel;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TCP\_IP\_Chat

{

public partial class FrmMain : MaterialForm

{

SimpleTcpClient client;

string nickname = "";

public FrmMain()

{

InitializeComponent();

OpenLogin();

}

private void FrmMain\_Load(object sender, EventArgs e)

{

ControlForms();

}

private void Events\_DataReceived(object sender, DataReceivedFromServerEventArgs e)

{

Invoke((MethodInvoker)delegate

{

rtbMessages.Text += $"{Encoding.UTF8.GetString(e.Data)}";

});

}

private void Events\_Disconnected(object sender, EventArgs e)

{

Invoke((MethodInvoker)delegate

{

rtbMessages.Text += $"Disconnected from Server!{ Environment.NewLine}";

});

}

private void Events\_Connected(object sender, EventArgs e)

{

Invoke((MethodInvoker)delegate

{

rtbMessages.Text += $"Connected to Server!{ Environment.NewLine}";

});

}

private void BtnSend\_Click(object sender, EventArgs e)

{

SendMessenger();

}

private void BtnMenu\_Click(object sender, EventArgs e)

{

cmsOptions.Show(Cursor.Position);

}

private void ExitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Close();

}

private void DisconnectToolStripMenuItem1\_Click(object sender, EventArgs e)

{

client.Disconnect();

}

private void SwitchConnectionToolStripMenuItem\_Click(object sender, EventArgs e)

{

OpenLogin();

}

private void ClearConversationToolStripMenuItem\_Click(object sender, EventArgs e)

{

rtbMessages.Text = "";

}

private void CmsOptions\_Opening(object sender, CancelEventArgs e)

{

if (client != null)

{

if (client.IsConnected)

{

ControleDisconnect(true);

}

else

{

ControleDisconnect(false);

}

}

else

{

ControleDisconnect(false);

}

}

private void TxtText\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar == (char)Keys.Enter)

{

SendMessenger();

}

}

private void ControlForms()

{

rtbMessages.ReadOnly = true;

rtbMessages.SelectionProtected = true;

rtbMessages.BackColor = BackColor;

txtText.BackColor = BackColor;

txtText.Focus();

}

private void ConfigurationServer(string server)

{

client = new SimpleTcpClient(server);

client.Events.Connected += Events\_Connected;

client.Events.Disconnected += Events\_Disconnected;

client.Events.DataReceived += Events\_DataReceived;

}

private void ControlButtons(bool status)

{

btnSend.Enabled = status;

}

private void OpenLogin()

{

FrmLogin frmLogin = new FrmLogin(nickname);

frmLogin.ShowDialog();

if (frmLogin.DialogResult == DialogResult.OK)

{

if (client != null)

{

client.Disconnect();

Task.Delay(2000);

};

nickname = frmLogin.nickname;

Connect(nickname, frmLogin.server);

UpdateLabels(nickname, frmLogin.server);

}

}

private void UpdateLabels(string nickname, string server)

{

Invoke((MethodInvoker)delegate

{

lblServer.Text = "Server: " + server;

});

}

private void Connect(string nickname, string server)

{

try

{

ConfigurationServer(server);

client.Connect();

ControlButtons(true);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Attention", MessageBoxButtons.OK);

}

}

private void SendMessenger()

{

if (VerifyConnection())

{

string messages = nickname + ": " + txtText.Text + "\n";

client.Send(messages);

rtbMessages.Text += messages;

txtText.Text = "";

}

}

public bool VerifyConnection()

{

try

{

if (client != null)

{

if (client.IsConnected)

{

if (!string.IsNullOrEmpty(txtText.Text))

{

return true;

}

}

}

return false;

}

catch (NullReferenceException)

{

MessageBox.Show("You are not connected to any server!", "Attention!", MessageBoxButtons.OK);

return false;

}

}

private void ControleDisconnect(bool status)

{

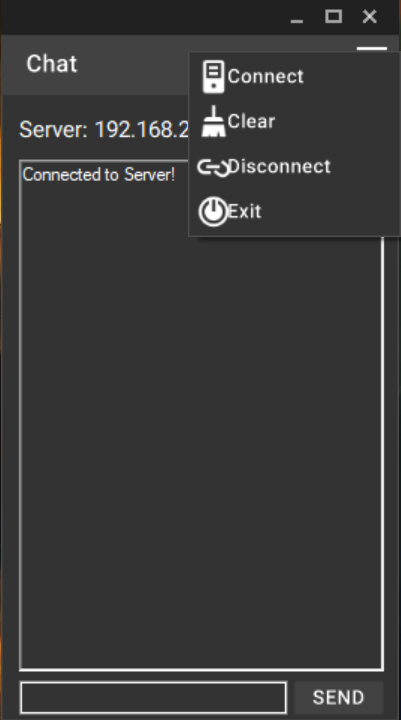
cmsDisconnect.Enabled = status;

}

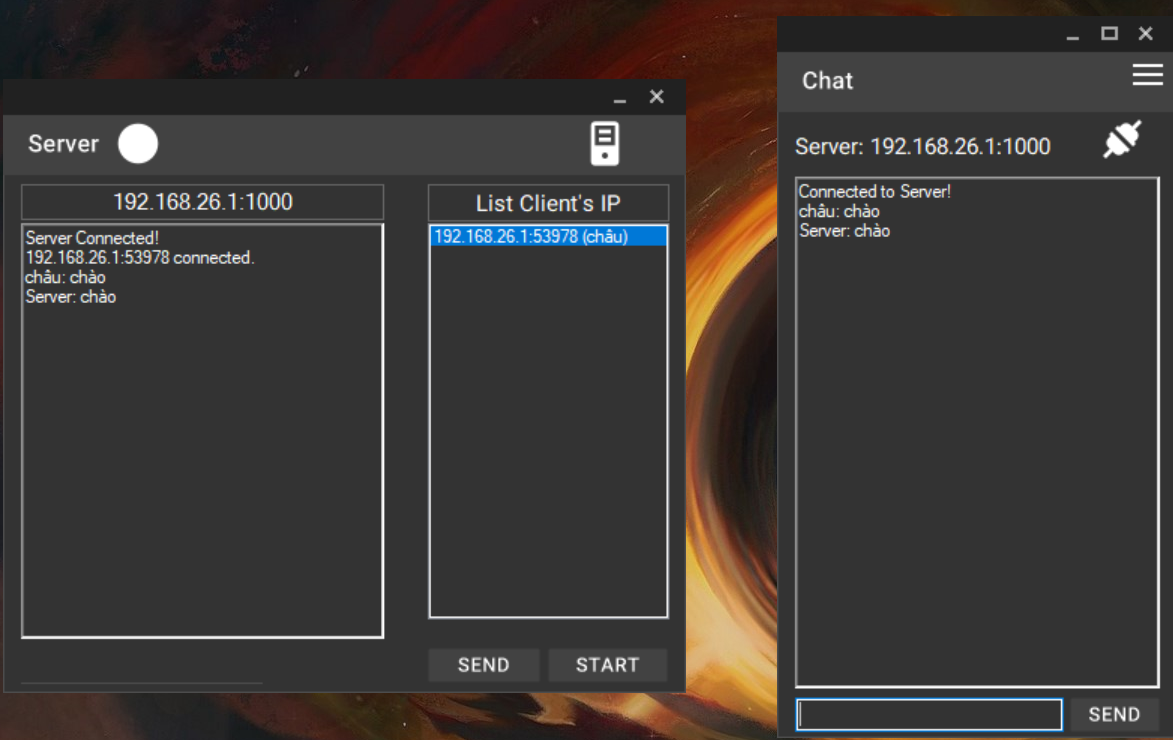
}

}

- Các Client có thể xoá lịch sử chat, ngắt kết nối và thoát với Server bằng cách click vào biểu tượng 3 gạch ngang phía trên bên phải:



- Giao diện Server và Client chat với nhau:



# PHẦN V: KẾT LUẬN

* Xây dựng kiến trúc đơn giản dựa trên mô hình Client- Server với 2 phần: Server, Client.
* Kiến trúc đã đáp ứng được yêu cầu cơ bản của đề bài về thiết kế hệ thống chat.
* Với việc sử dụng mô hình Client – Server nên sẽ có một vài ưu điểm:
* Client Server có khả năng chống quá tải mạng.
* Mô hình Client Server hỗ trợ, giúp chúng ta có thể làm việc trên bất kì một máy tính nào có hỗ trợ giao thức truyền thông.
* Client Server đảm bảo được sự toàn vẹn dữ liệu khi có sự cố xảy ra.
* Dễ dàng mở rộng, xây dựng hệ thống mạng.
* Chỉ cần chung định dạng giao tiếp mà không cần chung nền tảng là có thể hoạt động được.
* Có thể có nhiều Server cùng làm một dịch vụ, chúng có thể nằm trên nhiều máy tính hoặc một máy tính.
* Bên cạnh đó cũng sẽ có những nhược điểm:
* Một trong những vấn đề nảy sinh trong mô hình này đó là tính an toàn và bảo mật thông tin trên mạng. Do phải trao đổi dữ liệu giữa hai máy ở hai khu vực khác nhau cho nên dễ dàng xảy ra hiện tượng thông tin truyền trên mạng bị lộ.
* Cần bảo trì, bảo dưỡng Server thường xuyên.
  + Chưa gửi kèm các hình ảnh khi Chat.
  + Chưa có chức năng gửi File giữa các Users.
  + Chỉ hổ trợ Chat Text đơn thuần.
* Hướng phát triển:
* Thiết kế và xây dựng thành hệ thống hoàn chỉnh.
* Hạn chế những nhược điểm của hệ thống.
* Hỗ trợ gửi hình ảnh khi Chat.
* Hỗ trợ chức năng gửi file.
* Hỗ trợ chức năng Voice Chat và Webcam.