**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

🙤🙧 🕮 🙥🙦

**Logo, icon

Description automatically generated**

**BÁO CÁO THỰC HÀNH**

**Lab 3: Lập trình Sockets trong C#**

**Môn học: Lập trình mạng căn bản**

**Lớp: NT106.M21.1**

**GVHD: Lê Minh Khánh Hội**

**Nhóm 5**

|  |  |
| --- | --- |
| **Võ Thanh Phong** | **19522017** |
| **Huỳnh Quang Vũ** | **19522532** |
| **Trương Huỳnh Quý** | **19522110** |
| **Đỗ Trần Phương Nam** | **19520749** |
| **Lưu Tinh Anh** | **20520386** |

***TP. Hồ Chí Minh, tháng 4, năm 2022***

# Phân chia công việc

|  |  |  |  |
| --- | --- | --- | --- |
| **Họ và tên** | **MSSV** | **Nhiệm vụ** | **Mức độ hoàn thành** |
| Võ Thanh Phong | 19522017 | Câu 2 + Form điều hướng | Hoàn thành |
| Huỳnh Quang Vũ | 19522532 | Câu 4 | Hoàn thành |
| Trương Huỳnh Quý | 19522110 | Câu 3 | Hoàn thành |
| Đỗ Trần Phương Nam | 19520749 | Câu 1 | Hoàn thành |
| Lưu Tinh Anh | 20520386 | Câu 4 | Hoàn thành |

# 1. Viết ứng dụng thực hiện gởi và nhận dữ liệu giữa hai bên sử dụng giao thức UDP (UDP Client và UDP Server). Người dùng ở Client sẽ chỉ định IP, port cần kết nối và thông điệp gửi đến Server. Tại Server sẽ nhận được thông điệp gửi từ Client.

- Khi khởi động chương trình bằng 1 form điều hướng :

A screenshot of a computer

Description automatically generated with medium confidence

- Chuyển từ trang điều hướng đến client và server :

A screenshot of a computer

Description automatically generated with medium confidence

- Thực hiện kết nối và gửi gói tin:

A screenshot of a computer

Description automatically generated with medium confidence

# 2. Viết chương trình lắng nghe dữ liệu từ dịch vụ Telnet sử dụng kết nối TCP (sử dụng lớp Socket).

Graphical user interface, application

Description automatically generated

Hình 2.1. Giao diện chính của TCP Server lúc khởi động.

Graphical user interface, application

Description automatically generated

Hình 2.2. Giao diện chính của TCP Server lúc nhấn nút Listen.

Graphical user interface, application

Description automatically generated

Hình 2.3. Giao diện chính của TCP Server khi đã kết nối với Client.

Graphical user interface

Description automatically generated

Hình 2.4. Giao diện chính của TCP Server khi đang lắng nghe kết nối bên Client.

- Sau khi mở TCP Server, ta sẽ nhấn vào nút Listen để lắng nghe các kết nối từ Client đến IP Address 127.0.0.1 port 8080. Sau khi nhấn vào nút Listen thì màn hình sẽ hiển thị ra thông báo “Listening on 127.0.0.1:8080” và nút Listen sẽ bị khóa lại (Tránh trường hợp nhấn nhiều lần). Tiếp theo ta sẽ mở cmd lên đóng vai trò là Client, gõ lệnh “telnet 127.0.0.1 8080” để telnet đến Server. Sau khi đã kết nối thành công thì màn hình sẽ hiển thị thông báo “Telnet running on 127.0.0.1:8080” và trong ô nội dung sẽ hiển thị dòng chữ “New client connected”. Giờ đây khi ta nhập chữ vào cmd thì sẽ hiển thị bên Server nhưng không hiển thị bên cmd. Cuối cùng là nhấn vào nút Close để đóng kết nối.

# 3. Viết ứng dụng thực hiện gởi và nhận dữ liệu sử dụng giao thức TCP (TCP Client và TCP Listener). Server lắng nghe kết nối thông điệp từ Client.

Graphical user interface

Description automatically generated

Hình 3.1. Form giao diện điều hướng open TCP server và Client.

- Ban đầu sẽ có giao diện điều hướng như hình 3.1.

- Tiếp theo, ta nhấp vào Open TCP Server để cho Server lắng nghe và để kết nối với client, giao diện như hình 3.2.

Graphical user interface, application

Description automatically generated

Hình 3.2: Giao diện TcpServer.

- Ta nhấp vào Listen để server kết nối với Client. Sau đó ta sử dụng giao diện hình 1 để mở client nếu thành công thì ListView trên Server sẽ hiện thông báo “New client connected ” ( hình 3.3), còn không thành công thì sẽ hiển thị thông báo lỗi như hình 3.4.

Graphical user interface, text, application

Description automatically generated

Hình 3.3. Giao diện khi kết nối thành công.

Graphical user interface, application

Description automatically generated

Hình 3.4. Giao diện khi kết nối không thành công.

- Khi nhấp send message thì client sẽ gởi thông điệp đến server cụ thể là “ Hello server” ( hình 3.5).

Graphical user interface, application

Description automatically generated

Hình 3.5. Gửi dữ liệu từ client đến server.

- Trong giao diện TcpClient, nhấp disconnect thì sẽ hủy kết nối với server và đóng giao diện lại như hình 3.6.

Graphical user interface

Description automatically generated

Hình 3.6. Giao diện khi client hủy kết nối.

# 4. Viết chương trình Chat Room/ Gửi và nhận dữ liệu sử dụng TCP Client và TCP Listener. Mỗi người dùng sẽ có một tài khoản, khi một người dùng gửi tin nhắn thì tất cả mọi người còn lại đều sẽ nhận được tin nhắn đó.

### Phía Client:

- Tạo form chat\_client với cấu trúc như sau và bắt đầu cấu hình cho Button **Send**:

Graphical user interface, application, Word

Description automatically generated

- Ngoài hàm sự kiện Click của Button **Send**, ta tạo 2 biến:

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Vấn đề đầu tiên cần giải quyết là phải kết nối. Ta tạo hàm **void Connect** để cấu hình quá trình kết nối:

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ **IP** gán giá trị **IP** đích với địa chỉ mặc định( nói chuyện trên máy local) và 1 **port** 8080

+ Gán biến **client**.

+Ta bắt đầu sử dụng event **Connect(IP)** để kết nối. Để gọn gang và dễ dàng hơn trong quá trình kết nối ta dùng **try catch** để bắt lỗi nếu **client** và **server** không kết nối với nhau được. Khi đó sẽ có 1 **MessageBox** báo lỗi về.

+Tạo 1 luồng lắng nghe, cho chạy nền để có thể trao đổi **Message** giữa **server** vs các **client** với nhau.

- Tiếp theo, ta giải quyết vấn đề gửi và nhận tin. Để gửi và nhận tin, ta tạo 2 hàm **Serialize** và **Deserialize** để **PHÂN MẢNH** và **TÁI CẤU TRÚC MẢNH** lại. Khi truyền qua mạng, ta biến các chuỗi tin thành dạng byte rồi gửi đi, đến bên đích nhận được sẽ dùng **Deserialize** gom các tín hiệu dạng **byte** lại rồi tái cấu trúc lại và ra được tin nhắn:

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Ta tạo 2 hàm **Send** và **Receive** để gửi và nhận tin dựa trên cách thức trên:

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Ở hàm **Send**, chỉ khi ô **textbox** không rỗng thì mới có thể gửi đi dữ liệu

+ Ở hàm **Receive**, ta tạo mảng **byte[] data** với **1024\*5000** là dung lương tối đa của mảng. Tạo **message** cho chạy qua hàm **Deserialize** để tái dựng dữ liệu rồi sau đó dùng hàm **AddMessage** để gắn tin nhắn dô khung chat. Đồng thời ta sử dụng vòng lặp **While** để nhận tin liên tục và dùng **try catch** lồng vào để ghi nhận lỗi nếu trong quá trình lắng nghe xảy ra.

- Ta code hàm **AddMessage**:

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Ta thêm hàm đóng kết nối:

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Bây giờ ta đã đủ các hàm cần thiết để tạo kết nối, nhắn và nhận tin. Ta sẽ code cho **Button Send**:

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Đồng thời ta tạo kết nối ngay khi vừa mở form:

Ảnh có chứa văn bản

Mô tả được tạo tự động

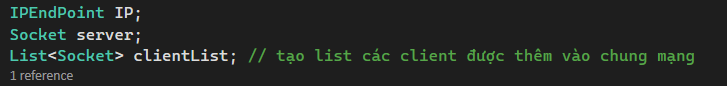
### Phía Server:

- Tạo form server với giao diện như sau và bắt đầu cấu hình cho **Button Listen**:

Graphical user interface, application, Word

Description automatically generated

- Ta cũng khởi tạo 2 biến **IP** đích và **Socket server**. Ta tạo thêm 1 list các **Client** sẽ kết nối với **server**:



- Ở **Server**, ta cũng tạo hàm kết nối để kết nối tất các các **client** với nhau và với **server**:

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Bởi vì **Server** được coi như là gốc để tạo và nhận kết nối. Ta dùng **Bind(IP)** để khẳng định **IP** là củng bản thân và đợi **IP** của **client** kết nối tới.

+ Ta tạo **Thread** để đợi xem **Client** nào sẽ nói chuyện với **Server**. Bắt đầu lắng nghe.

+ Lồng trong vòng lặp **While** để lắng nghe liên tục.

+ Khi một **client** kết nối thành công sẽ add 1 tin nhắn **“Đã kết nối thành công”** vào **Chat box** của **Server**

+ **Server.Listen(100)** là cho phép tối đa 100 **client** trong hang chờ.

+ Lắng nghe được 1 **client** thì ta **Accept()** và thêm vào **list Client**.

+ Sau đó ta tạo Luồng(Thread) để Receive.

+ Lồng trong **try catch**. Ở chỗ catch ta gán lại **IP** và **Socket server** để phòng trường hợp khi xảy ra việc lỗi 1 **client** gây mất kết nối, nên ta gán lại để khởi tạo **IP** và **server** lại.

- Ta cũng có các hàm tương tự để **phân mảnh** **và tái cấu trục tin** lại để nhận tin nhắn:

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Tạo hàm **Receive**:

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Vẫn dùng vòng lặp **While** để lắng nghe liên tục, và **try catch** để bắt lỗi trong quá trình lắng nghe.

+ Tham số truyền vào sẽ là 1 **object**, gán các **client** như 1 đối tượng **Socket**.

+ Ta tạo thêm 1 vòng lặp **Foreach** để duyệt hết **Client** đã kết nối rồi **send** thông tin cho chúng. Mục đích là để 1 client có thể **send message** cho tất cả **các client khác và server**. Kiểm tra điều kiện **item** gửi đi phải khác rỗng và khác với chính **item** của **client** gốc gửi thông tin đi để tránh **server** gửi lại vào **client** gửi gây lặp và tốn tài nguyên.

+ Khi bắt được lỗi thì ra xóa đi **client** lỗi và đóng kết nối lại.

- Ta cũng tạo thêm **hàm Close** để đóng kết nối và **hàm AddMessage**:

Graphical user interface, application

Description automatically generated

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Đã đủ các hàm, bây giờ ta code cho **Button Listen**:

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Đồng thời, ta thêm 1 dòng CheckForIllegalCrossThreadCalls = false trong hàm Public Chat\_Server:

Ảnh có chứa văn bản

Mô tả được tạo tự động

+ Mục đích là tránh gây đụng độ.

- Theo nhu cầu ta có thể tạo thêm nhiều **client** tương tự **client** đã tạo và tạo 1 form điều hướng cho dễ sử dụng:

Graphical user interface, application

Description automatically generated

- Đầu tiên hay vào **server** để nhấn nút **Lắng nghe**, sau đó ta có thể mở các **client** ra để tạo **Name** và **chat** với nhau bình thường:

Ảnh có chứa văn bản, ảnh chụp màn hình, trong nhà

Mô tả được tạo tự động