Lab03: Lập trình socket - 22520853 - Trần Thiện Mạnh

Bài 01: Viết ứng dụng thực hiện gửi 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.

Cách làm: Tạo 2 form Client và Server 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

Form Server:

- **Thành phần giao diện:**
  - Ô văn bản (**tb_PortListen**) để thiết lập cổng nghe.
  - Hộp văn bản phong phú (**rtb_RMessage**) để hiển thị các tin nhắn đã nhận.
  - Nút (**btn_Listen**) để bắt đầu lắng nghe các tin nhắn đến.
- **Chức năng chính:**
  - Khi nhấn **btn_Listen**, nó kiểm tra xem cổng đã được thiết lập chưa. Nếu không, nó hiển thị thông báo lỗi.
  - Máy chủ sử dụng một luồng nền (**Thread thdUDPServer**) để xử lý các gói UDP đến mà không làm đóng băng giao diện người dùng. Luồng này liên tục lắng nghe dữ liệu trên cổng được chỉ định.

Form Client:

- **Thành phần giao diện:**
  - Ô văn bản (**tb_IP_Host**) để nhập địa chỉ IP của máy chủ.
  - Ô văn bản (**tb_Port**) để nhập số cổng của máy chủ.
  - Hộp văn bản phong phú (**rtb_Message**) để viết tin nhắn cần gửi.
  - Nút (**btn_Send**) để kích hoạt hành động gửi tin nhắn.
- **Chức năng chính:**
  - Khi nhấn vào **btn_Send**, máy khách kiểm tra xem hộp tin nhắn có trống không. Sau đó, nó chuyển đổi tin nhắn văn bản thành byte sử dụng mã hóa UTF-8.
  - Máy khách sử dụng một thể hiện của **UdpClient** để gửi các byte này đến địa chỉ IP và cổng được chỉ định. Nếu hộp tin nhắn trống, nó sẽ hiển thị một thông báo cho biết tin nhắn là null.

Bài 2: Viết chương trình lắng nghe dữ liệu từ dịch vụ Telnet sử dụng kết nối TCP (sử dụng lớp Socket) với mô tả sau:

Cách làm:

Khởi Tạo và Cấu Hình Socket Lắng Nghe

```
IPEndPoint IPEP = new IPEndPoint(IPAddress.Parse("192.168.56.1"), 8080);
TcpListener tcpListener = new TcpListener(IPEP);
Socket ListenerSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
ListenerSocket.Bind(IPEP);
ListenerSocket.Listen(-1);
```

- **Khởi tạo IPEndPoint**: Đặt địa chỉ IP và cổng cho socket lắng nghe. Đây là địa chỉ mà máy chủ sẽ nhận các kết nối đến.
- **Tạo và cấu hình Socket**: **ListenerSocket** được tạo với các tham số cho phép giao tiếp TCP/IP trong mạng IPv4. **Bind** gắn socket này với địa chỉ IP và cổng được chỉ định.
- **Bắt đầu lắng nghe**: **Listen(-1)** cho phép socket bắt đầu lắng nghe các kết nối đến. Tham số **-1** thường không được khuyến khích vì có thể dẫn đến hành vi không mong muốn; nên thay thế bằng một số dương để giới hạn số lượng kết nối đang chờ xử lý.

Lắng nghe socket

```
//Đồng ý kết nối
ClientSocket = ListenerSocket.Accept();
```

- **Chấp nhận kết nối**: **Accept()** là phương thức chặn chờ cho đến khi có kết nối đến. Khi có máy khách kết nối, một socket mới (**ClientSocket**) được tạo để xử lý giao tiếp với máy khách đó.

Nhận và Hiển Thị Dữ Liệu

```
byte[] receive = new byte[1];

while (ClientSocket.Connected)
{
    string text = "";
    ClientSocket.Receive(receive);
    text += Encoding.ASCII.GetString(receive);
    rtb_ShowMessage.AppendText(text);
}
```

- **Khởi tạo bộ đệm**: Một mảng byte **receive** được sử dụng để lưu dữ liệu nhận được.
- **Vòng lặp nhận dữ liệu**: Trong khi **ClientSocket** còn kết nối, dữ liệu được đọc từ client. Dữ liệu nhận được chuyển đổi từ mảng byte sang chuỗi văn bản ASCII và nối vào nội dung của rich text box trên giao diện người dùng.

Bài 3: 1 Server – 1 Client 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 và thông điệp từ Client

Ý tưởng Chung

Ứng dụng cho phép máy khách kết nối với máy chủ qua TCP, gửi và nhận thông điệp. Máy chủ lắng nghe các kết nối đến và xử lý các tin nhắn từ máy khách. Cả hai bên cập nhật trạng thái và thông tin giao tiếp trên giao diện người dùng.

Cách làm: Tạo hai form client và server sử dụng phương thức TCP để gửi và nhận thông điệp.

Client - Máy Khách

Logic của Máy Khách

- **Kết Nối đến Máy Chủ**: Khi nhấn nút "Connect", máy khách thử kết nối đến máy chủ sử dụng địa chỉ IP và cổng được cấu hình trước. Nếu kết nối thành công, nút gửi tin nhắn sẽ được kích hoạt, và nút kết nối sẽ bị vô hiệu hóa để tránh kết nối lại.
- **Gửi Tin Nhắn**: Sau khi nhấn nút "Send", tin nhắn trong rich text box được gửi đến máy chủ qua TCP. Sau khi gửi, nội dung tin nhắn được xóa khỏi rich text box để nhập tin nhắn mới.
- **Ngắt Kết Nối**: Khi nhấn nút "Disconnect", kết nối TCP đến máy chủ được đóng, và trạng thái các nút được thiết lập lại để cho phép kết nối lại.

Ý Nghĩa Của Các Phương Thức

- **Connect()**: Thiết lập kết nối TCP đến máy chủ.
- **Send(string message)**: Gửi dữ liệu qua kết nối TCP.
- **Disconnect()**: Đóng kết nối TCP.

Server - Máy Chủ

Logic của Máy Chủ

- **Lắng Nghe Kết Nối**: Khi nhấn nút "Listen", máy chủ bắt đầu lắng nghe trên cổng được chỉ định. Thông tin về trạng thái máy chủ được hiển thị trên giao diện người dùng.
- **Chấp Nhận và Xử Lý Kết Nối**: Máy chủ chấp nhận kết nối TCP từ máy khách và bắt đầu nhận dữ liệu. Khi có dữ liệu từ máy khách, dữ liệu được hiển thị trên giao diện người dùng. Khi kết nối bị ngắt, thông báo tương ứng được hiển thị.

Ý Nghĩa Của Các Phương Thức

- **Listen(TcpListener listener)**: Bắt đầu lắng nghe các kết nối đến và xử lý chúng trên một luồng riêng.
- **Receive(TcpClient client)**: Nhận dữ liệu từ máy khách qua TCP và cập nhật giao diện người dùng.

Bài 6: 1 Server – Multi Client 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. Thêm vào đó là tính năng nhắn tin riêng, người dùng có thể chọn một người dùng khác để nhắn tin riêng.

Ý tưởng Chung

Ứng dụng chat client-server mà bạn cung cấp qua mã nguồn sử dụng giao thức TCP/IP để thực hiện giao tiếp hai chiều giữa máy chủ và các máy khách. Ý tưởng chung là tạo một môi trường giao tiếp trực tiếp, nơi máy khách có thể gửi tin nhắn đến máy chủ, và máy chủ sau đó phát sóng những tin nhắn này tới tất cả các máy khách khác đang kết nối. Cả hai bên (máy khách và máy chủ) đều có giao diện người dùng đồ họa (GUI) để hiển thị thông tin và cho phép nhập liệu từ người dùng.

Client - Máy Khách

Các Thành Phần Chính

- **TcpClient**: Đối tượng dùng để thiết lập kết nối TCP.
- **NetworkStream**: Luồng dùng để gửi và nhận dữ liệu qua kết nối TCP.
- **Thread**: Sử dụng luồng riêng để nhận dữ liệu mà không ảnh hưởng đến luồng UI chính.
- **Form Controls**: Bao gồm các nút (buttons), hộp nhập liệu (textbox) và một ListView để hiển thị tin nhắn.

Logic của Máy Khách

- **Kết Nối đến Máy Chủ**: Trong sự kiện nhấn nút "Connect", phương thức **ConnectToServer()** được gọi. Máy khách sử dụng IP là "127.0.0.1" và cổng 8080 để thử kết nối đến máy chủ. Nếu thành công, luồng nhận tin **ReceiveFromServer** được khởi chạy và nút kết nối được vô hiệu hóa.
- **Nhận Tin Nhắn**: Trong phương thức **ReceiveFromServer()**, máy khách liên tục nhận dữ liệu từ server qua luồng đã mở. Khi nhận được dữ liệu, nó sử dụng phương thức **Invoke** để cập nhật tin nhắn vào ListView mà không gây xung đột với luồng UI.
- **Gửi Tin Nhắn**: Trong sự kiện nhấn nút "Send", phương thức **SendMess(string text)** được gọi để gửi dữ liệu đã nhập qua mạng. Máy khách gửi dữ liệu dưới dạng chuỗi byte sau khi định dạng tin nhắn với tên của người gửi.

Server - Máy Chủ

Các Thành Phần Chính

- **Socket**: Sử dụng để lắng nghe và chấp nhận kết nối TCP từ client.
- **IPEndPoint**: Định nghĩa địa chỉ IP và cổng cho socket lắng nghe.
- **List<Socket>**: Danh sách các socket của client đã kết nối.
- **Thread**: Sử dụng luồng riêng để lắng nghe và xử lý các kết nối đến.

Logic của Máy Chủ

- **Lắng Nghe Kết Nối**: Phương thức **StartListening()** khởi động luồng **ListenThread()** để bắt đầu lắng nghe trên địa chỉ và cổng đã chỉ định. Khi một client kết nối, máy chủ chấp nhận kết nối và bắt đầu luồng mới để nhận dữ liệu từ client đó.
- **Nhận và Xử Lý Dữ liệu**: Trong luồng **ReceiveDataThread()**, máy chủ nhận dữ liệu từ client và hiển thị lên giao diện người dùng. Dữ liệu nhận được cũng được phát sóng lại cho tất cả các client khác.
- **Xử Lý Ngắt Kết Nối Client**: Khi một client ngắt kết nối, máy chủ đóng socket của client đó và loại bỏ khỏi danh sách.