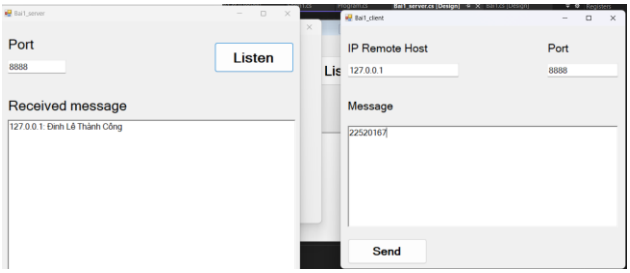
BÁO CÁO THỰC HÀNH LAB 3

Đinh Lê Thành Công – 22520167

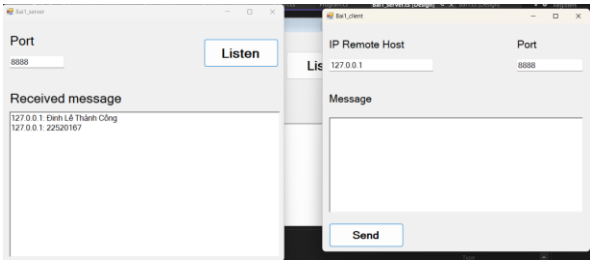
Mục lục

Bài 1: UDP Client – Server

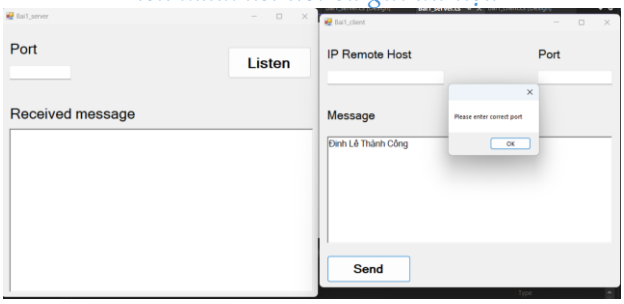
1.1 Chạy chương trình



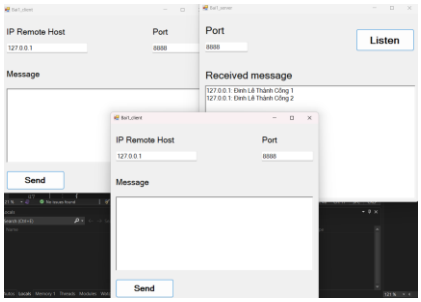
Tiến hành kết nối và gửi dữ liệu



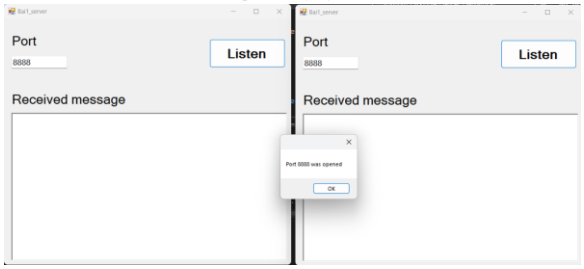
Gửi dữ liệu thành công



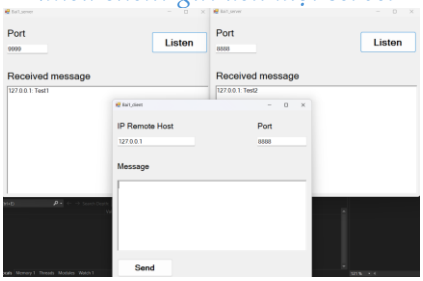
Lỗi không nhập port và IP



Nhiều client gửi đến một server



Lỗi khi hai server cùng mở chung port



Một client gửi đến nhiều server

1.2 Xử lý chương trình

- Ở trong cả client và server phần nhập port và IP em đã thêm chức năng ngăn chặn người dùng nhập vào chữ, quá 3 dấu chấm, …

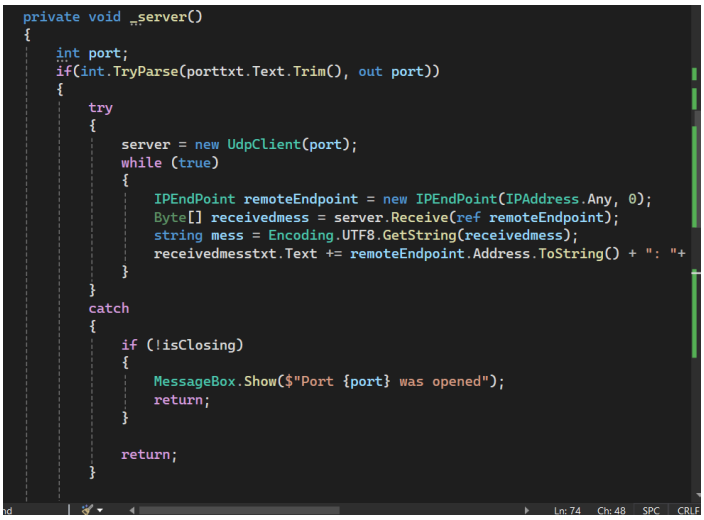
```
private void txtPortPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsNumber(e.KeyChar) && !char.IsControl(e.KeyChar))
    {
        e.Handled = true;
    }
}
```

Chỉ cho nhập số ở port

```
private void txtIpPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&
        (e.KeyChar != '.'))
    {
        e.Handled = true;
    }

    if (e.KeyChar == '.')
    {
        int count = 0;
        foreach (char c in (sender as TextBox).Text)
        {
            if (c == '.')
            {
                count++;
            }
        }
        if (count >= 3)
            e.Handled = true;
    }
}
```

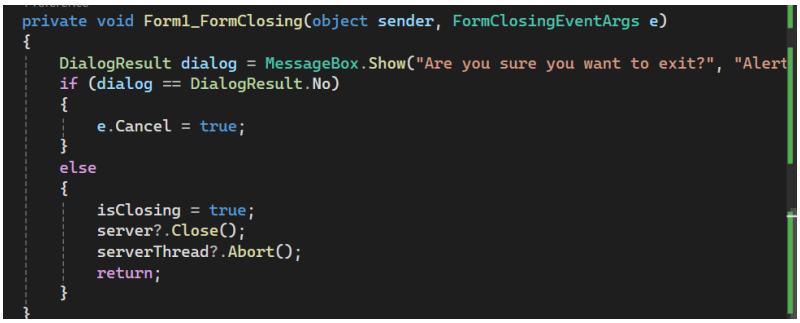
Chỉ cho nhập số và 3 dấu chấm ở IP

- Ở phần nhiều server chạy cùng một lúc, nếu như mở cùng một port thì sẽ sinh ra lỗi và cảnh báo.



```
private void _server()
{
    int port;
    if(int.TryParse(porttxt.Text.Trim(), out port))
    {
        try
        {
            server = new UdpClient(port);
            while (true)
            {
                IPEndPoint remoteEndpoint = new IPEndPoint(IPAddress.Any, 0);
                Byte[] receivedmess = server.Receive(ref remoteEndpoint);
                string mess = Encoding.UTF8.GetString(receivedmess);
                receivedmesstxt.Text += remoteEndpoint.Address.ToString() + ": "+
            }
        }
        catch
        {
            if (!isClosing)
            {
                MessageBox.Show($"Port {port} was opened");
                return;
            }

            return;
        }
```

- Ở mục đóng form server, em thêm một hàm để tiến hành đóng luôn port mà form server đang mở.

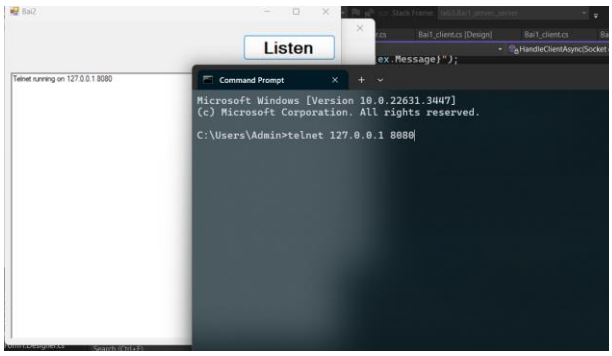


```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult dialog = MessageBox.Show("Are you sure you want to exit?", "Alert
    if (dialog == DialogResult.No)
    {
        e.Cancel = true;
    }
    else
    {
        isClosing = true;
        server?.Close();
        serverThread?.Abort();
        return;
    }
}
```

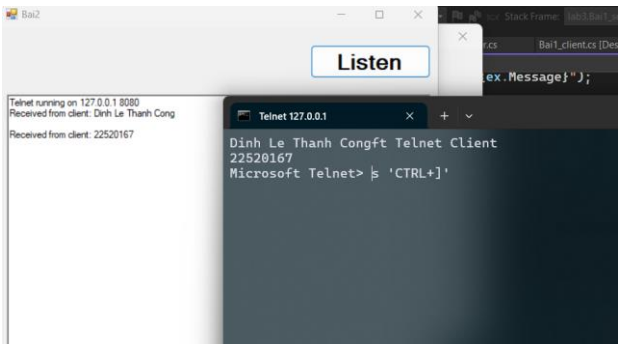
- Với client em nhận thấy được rằng kết nối Udp không kiểm tra được port đó có đang mở hay không. Vì thế khi tiến hành gửi mà không có port nào đang lắng nghe, thì dữ liệu sẽ tự động bị loại bỏ và không có bất kì một cảnh báo nào.

Bài 2: Chương trình TCP Server đơn giản

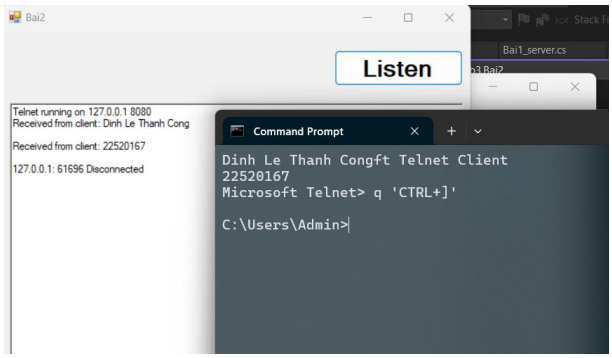
2.1 Chạy chương trình



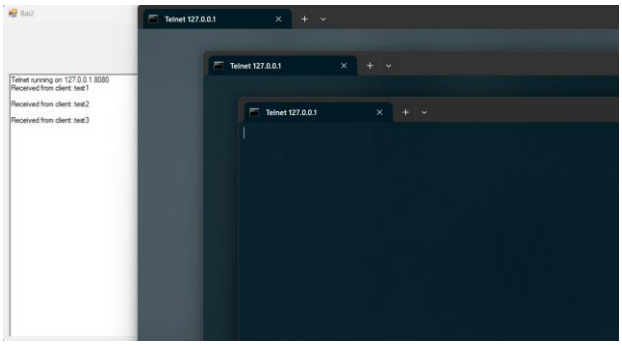
Tiến hành kết nối đến server



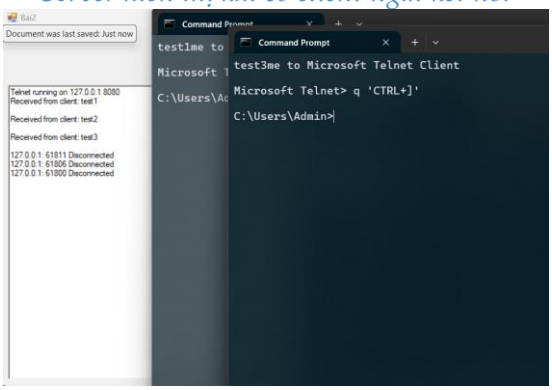
Kết nối thành công và gửi dữ liệu



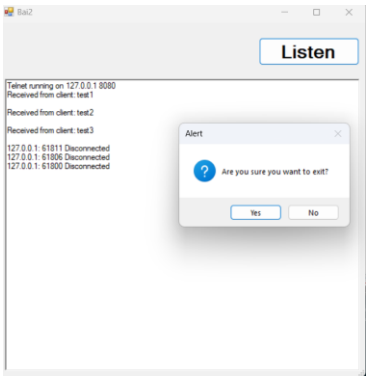
Server hiển thị khi có client ngắt kết nối



Nhiều client gửi lên một server



Client ngắt kết nối



Thoát server

2.2 Xử lý chương trình

- Ở chương trình này em đã quy định sẵn port và IP cho server là 127.0.0.1: 8080

```
listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
IPEndPoint ipepserver = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 8080);
listener.Bind(ipepserver);
listener.Listen(-1);
servertxt.Text += "Telnet running on 127.0.0.1 8080 \n";

while (true)
{
    Socket clientsocket = listener.Accept();
    Thread clientThread = new Thread(() => HandleClient(clientsocket));
    clientThread.Start();
}
```

- Tương tự như bài 1, em cũng tạo thêm một sự kiện nhận biết người dùng nếu đóng form thì sẽ đóng luôn port đang mở (8080).

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult dialog = MessageBox.Show("Are you sure you want to exit?", "Alert", Messag
    if (dialog == DialogResult.No)
    {
        e.Cancel = true;
    }
    else
    {
        isClosing = true;
        listener ?.Close();
        servertr?.Abort();
        return;
    }
}
```

- Để server nhận kết nối được từ nhiều client thì em đã tạo ra một thread riêng biệt cho từng client và tiến hành xử lí từng thread đó.

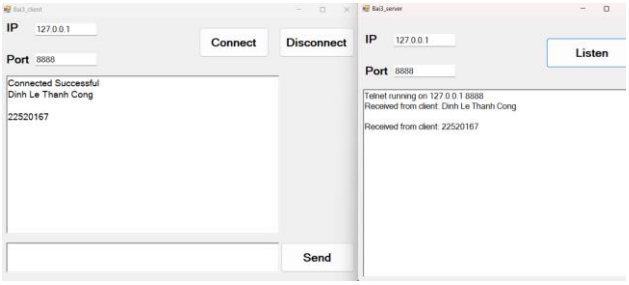
```
while (true)
{
    Socket clientsocket = listener.Accept();
    Thread clientThread = new Thread(() => HandleClient(clientsocket));
    clientThread.Start();
}
```

- Nếu người dùng không gửi dữ liệu nữa thì server sẽ nhận biết bằng cách kiểm tra bytereceive. Nếu bằng 0 thì sẽ hiển thị lên IP và Port client đã ngắt kết nối. Và đi kèm với return để thoát hẳn thread đã tạo.

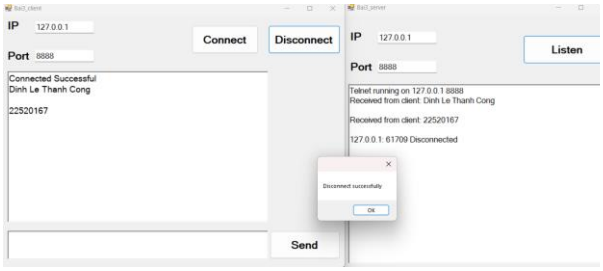
```
IPEndPoint clientEndPoint = clientsocket.RemoteEndPoint as IPEndPoint;

while (true)
{
    string text = "";
    do
    {
        bytereceive = clientsocket.Receive(recv);
        if (bytereceive == 0)
        {
            clientsocket.Close();
            servertxt.Text += $"{clientEndPoint.Address}: {clientEndPoint.Port} Disconnected\n";
            return;
        }
        text += Encoding.UTF8.GetString(recv);
    } while (text[text.Length - 1] != '\n');
```

Bài 3: Chương trình gửi nhận dữ liệu TCP (1C-1S)

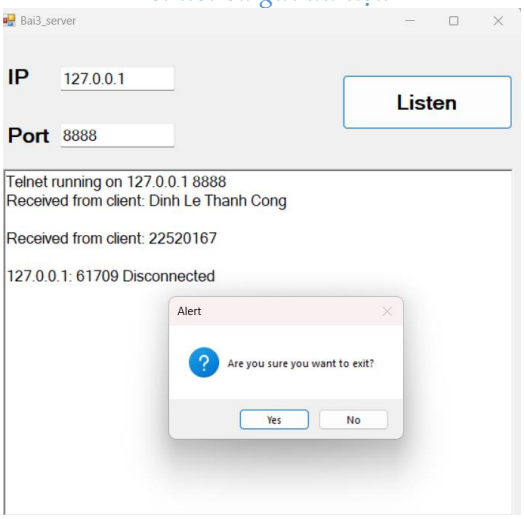
3.1 Chạy chương trình



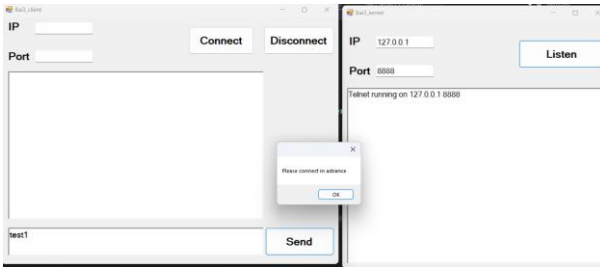
Kết nối và gửi dữ liệu



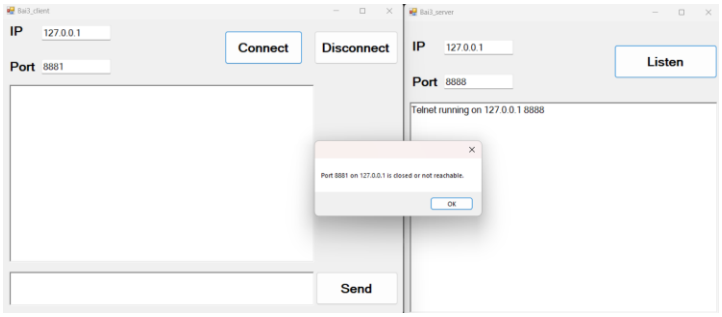
Ngắt kết nối ở client



Thoát server



Lỗi khi chưa kết nối nhưng gửi nội dung



Lỗi khi kết nối đến Port không được mở

3.2 Xử lý chương trình

- Tương tự như các bài trên, em đã để các hàm để tránh người dùng nhập port và IP những ký tự không hợp lệ.
- Vì yêu cầu đề bài 1S – 1C nên em đã giới hạn lại số form có thể hiển thị ra từ người dùng, đảm bảo rằng người dùng không cố tình tạo ra nhiều Server và nhiều Client.

```
private void btnServer_Click(object sender, EventArgs e)
{
    if(cntServer == 0)
    {
        cntServer++;
        Task.Run(() =>
        {
            Bai3_server serverForm = new Bai3_server();
            serverForm.FormClosing += (s, args) => ResetCountsServer();
            Application.Run(serverForm);
        });
    }
    else
    {
        return;
    }
}
```

```
1 reference
private void btnClient_Click(object sender, EventArgs e)
{
    if(cntClient == 0)
    {
        cntClient++;
        Task.Run(() =>
        {
            Bai3_client clientForm = new Bai3_client();
            clientForm.FormClosing += (s, args) => ResetCountsClient();
            Application.Run(clientForm);
        });
    }
    else { return; }
}
```

- Ở server bài này thì em hoàn toàn lấy từ bài số 2 chỉ thêm đoạn gửi trả về cho client nội dung mà client đã gửi lên.

```
            txtContent.Text += "Received from client: " + text + '\n';
            byte[] data = Encoding.UTF8.GetBytes(text);
            clientsocket.Send(data);
        }
```

- Ở client em đã đặt một biến **cnt** để kiểm tra việc server đã có mở port hay chưa và kiểm tra đã kết nối đến server từ client hay chưa. Và biến này sẽ được đặt lại khi ngắt kết nối từ client.

```
private void btnConnect_Click(object sender, EventArgs e)
{
    if(cnt < 1)
    {
        CheckConnect();
    }
    else return;

}

1 reference
private void btnSend_Click(object sender, EventArgs e)
{
    if(cnt == 0)
    {
        MessageBox.Show("Please connect in advance");
        return;
    }
    Connected();
    txtContentClientPre.Text = "";
}
```

```
private void btnDisconnect_Click(object sender, EventArgs e)
{
    try
    {
        tcpClient.GetStream().Close();
        tcpClient.Close();
        MessageBox.Show("Disconnect successfully");
        cnt--;
        txtContentClientSent.Text = "";
    }
    catch
    {
        return;
    }
}
```

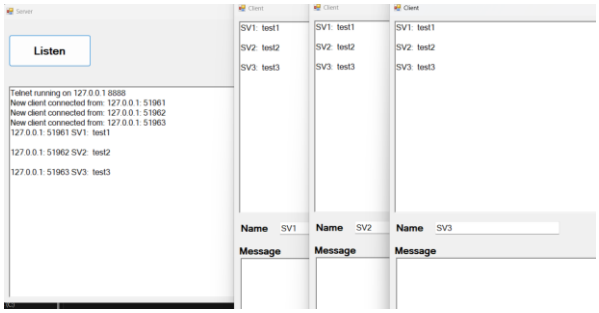
- Đối với client em sử dụng bất đồng bộ để tiến hành gửi dữ liệu và nhận lại dữ liệu.

```
private void Connected()
{
    NetworkStream stream = tcpClient.GetStream();

    string textData = txtContentClientPre.Text.Trim() + '\n';
    Byte[] data = Encoding.UTF8.GetBytes(textData);
    stream.Write(data, 0, data.Length);
    _ = ReceiveDataAsync();
}
```

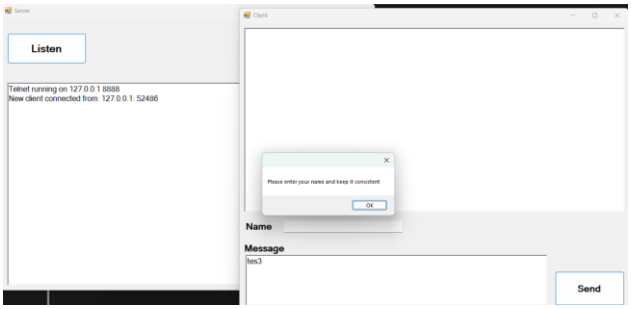
```
private async Task ReceiveDataAsync()
{
    NetworkStream stream = tcpClient.GetStream();
    byte[] buffer = new byte[1024];
    int bytesRead;

    while ((bytesRead = await stream.ReadAsync(buffer, 0, buffer.Length)) > 0)
    {
        string receivedData = Encoding.UTF8.GetString(buffer, 0, bytesRead);
        txtContentClientSent.Text += receivedData + '\n';
    }
}
```

Bài 4: Chương trình chatroom (1S-nC)

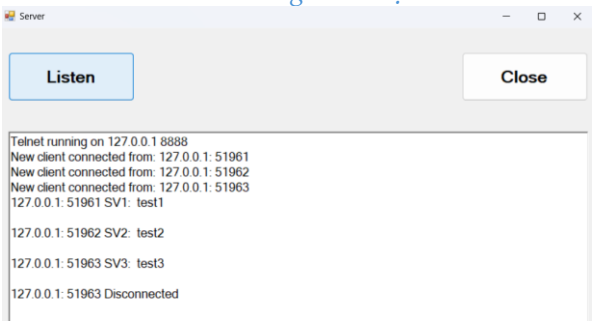
4.1 Chạy chương trình



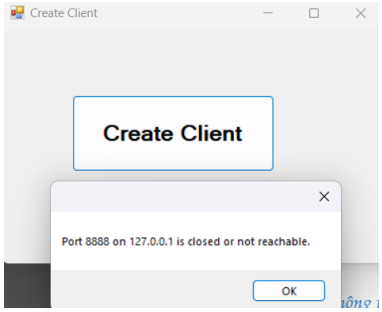
Kết nối và gửi dữ liệu



Gửi dữ liệu nhưng không kèm tên



Server khi có client ngắt kết nối



Lỗi khi chưa có server

4.2 Xử lý chương trình

- Với server em đã quy định sẵn port và IP không cần phải thao tác nhập.

```
listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
IPEndPoint ipepserver = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 8888);
listener.Bind(ipepserver);
listener.Listen(-1);
txtContentServer.Text += "Telnet running on 127.0.0.1 8888\n";
```

- Phần còn lại của server tương tự bài 2, 3 chỉ khác một điểm duy nhất đó chính là chức năng broadcast thì ở đây em sẽ dùng vòng lặp để gửi cho các client đã kết nối lên server. *(clientSockets là một list chứa thông tin client đã kết nối)*

```
foreach (var client in clientSockets)
{
    client.Send(data);
}
```

- Khi có một client ngắt kết nối đến server thì lúc này server sẽ tiến hành ngắt kết nối và xóa khỏi danh sách những client đã kết nối.

```
if (bytereceive == 0)
{
    clientsocket.Close();
    txtContentServer.Text += $"{clientEp.Address}: {clientEp.Port} Disconnected\n";
    clientSockets.Remove(clientsocket);
    return;
}
```

- Đối với client thì trong quá trình tạo ra form mới em sẽ cho tự động kết nối tới server. Nếu kết nối thành công thì mới hiển thị form và dùng bất đồng bộ để mở luồng nhận dữ liệu từ server.

```
1 reference
public FormClient()
{
    InitializeComponent();
    CheckConnect();
    this.FormClosing += new FormClosingEventHandler(Form1_FormClosing);
}
```

```
private void CheckConnect()
{
    try
    {
        tcpClient.Connect("127.0.0.1", 8888);
        MessageBox.Show("Connect Successful");
        _ = ReceiveDataAsync();
    }
    catch (Exception)
    {
        MessageBox.Show($"Port 8888 on 127.0.0.1 is closed or not reachable.");
        this.Close();
    }
}
```

- Đối với một client khi tham gia chat bắt buộc phải điền một tên và tên sẽ chỉ được nhập 1 lần duy nhất.

```
private void btnSend_Click(object sender, EventArgs e)
{
    if (userName == null && txtName.Text.Trim() != "")
    {
        userName = txtName.Text.Trim();
    }

    if (txtName.Text.Trim() == "" || txtName.Text.Trim() != userName)
    {
        MessageBox.Show("Please enter your name and keep it consistent");
        return;
    }
    else
    {
        Connected();
    }
}
```

- Tương tự các bài trên khi thoát sẽ hiển thị lên một thông báo và nếu người dùng đồng ý thoát thì sẽ ngắt kết nối và đóng form.

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult dialog = MessageBox.Show("Are you sure you want to exit?", "Alert", Messa
    if (dialog == DialogResult.No)
    {
        e.Cancel = true;
    }
    else
    {
        try
        {
            tcpClient.GetStream().Close();
            tcpClient.Close();
            return;
        }
        catch
        {
            return;
        }

    }
}
```