✅ 1. Giới thiệu UDP (User Datagram Protocol)

UDP là giao thức tầng **Transport** giống như TCP, nhưng:

Đặc điểm	TCP	UDP
Kết nối	Có (connection-oriented)	Không (connectionless)
Đảm bảo	Có đảm bảo dữ liệu đến nơi, đúng thứ tự	Không đảm bảo
Tốc độ	Chậm hơn do kiểm tra lỗi	Nhanh hơn, ít overhead
Ứng dụng	Web, Email, File transfer	Video/Voice Call, Game Online

UDP rất phù hợp cho các ứng dụng **real-time**, cần **tốc độ nhanh** và **chịu mất mát nhẹ**.

✅ 2. Quy tắc hoạt động của UDP

💡 UDP hoạt động theo mô hình:

Client gửi dữ liệu → Server nhận (không cần thiết lập kết nối trước)

📥 Cơ chế hoạt động:

1. **Không tạo kết nối**
2. **Client tạo socket và gửi dữ liệu trực tiếp đến IP:Port**
3. **Server mở socket, lắng nghe dữ liệu gửi đến**
4. **Không có xác nhận là dữ liệu đã đến hay chưa**

✅ 3. Khai báo, tạo socket UDP

C# dùng class UdpClient

csharp
CopyEdit
UdpClient udpClient = new UdpClient(); // Tạo socket client

✅ 4. Gửi dữ liệu bằng UDP (Client)

csharp
CopyEdit
using System.Net.Sockets;
using System.Text;

```
UdpClient udpClient = new UdpClient();
string message = "Hello UDP!";
byte[] data = Encoding.UTF8.GetBytes(message);
udpClient.Send(data, data.Length, "127.0.0.1", 9000); // Gửi tới server IP + port
udpClient.Close();
```

✅ 5. Nhận dữ liệu bằng UDP (Server)

```
csharp
CopyEdit
using System.Net;
using System.Net.Sockets;
using System.Text;

UdpClient udpServer = new UdpClient(9000); // Lắng nghe tại port 9000

while (true)
{
   IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0);
   byte[] data = udpServer.Receive(ref remoteEP); // Nhận dữ liệu
   string message = Encoding.UTF8.GetString(data);
   Console.WriteLine($"Nhận từ {remoteEP}: {message}");
}
```

✅ 6. Tổng kết ưu – nhược điểm UDP

Ưu điểm	Nhược điểm
Tốc độ rất cao	Không đảm bảo dữ liệu đến nơi
Không cần thiết lập kết nối	Không có thứ tự dữ liệu
Dễ dùng cho nhiều client	Dữ liệu dễ bị trùng lặp hoặc mất mát

---

✅ 7. Ứng dụng thực tế

UDP thường dùng trong:

- Game nhiều người chơi (Multiplayer games)
- Ứng dụng chat hoặc gửi tin nhanh
- Gửi ảnh từ camera IP (stream)
- Gửi dữ liệu cảm biến trong IoT

Ý tưởng gửi file bằng UDP:

1. **Client**:
    - Gửi 1 header chứa tên file: <FILE>:TenFile.ext
    - Gửi từng phần dữ liệu file (chia nhỏ theo buffer)
    - Gửi 1 đoạn cuối để kết thúc: <EOF>
2. **Server**:
    - Nhận tên file
    - Tạo bộ nhớ đệm (MemoryStream)
    - Nhận từng phần dữ liệu
    - Khi gặp <EOF> → ghi file vào ổ cứng.

⚙ Cấu hình chung:

csharp
CopyEdit
const int BUFFER_SIZE = 8192;
const int PORT = 9000;

✅ Phần 1: Server – Nhận file UDP

csharp
CopyEdit

```
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;

class UdpFileReceiver
{
   static void Main()
   {
      UdpClient udpServer = new UdpClient(PORT);
      Console.WriteLine("Đang lắng nghe UDP...");

      IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0);
      MemoryStream ms = null;
      string fileName = "";

      while (true)
      {
         byte[] buffer = udpServer.Receive(ref remoteEP);
         string message = Encoding.UTF8.GetString(buffer);
```

```
            if (message.StartsWith("<FILE>:"))
            {
                fileName = message.Substring(7);
                ms = new MemoryStream();
                Console.WriteLine($"Đang nhận file: {fileName}");
                continue;
            }

            if (message == "<EOF>")
            {
                string savePath = Path.Combine(@"C:\ReceivedFiles", fileName);
                Directory.CreateDirectory(@"C:\ReceivedFiles");
                File.WriteAllBytes(savePath, ms.ToArray());
                ms.Close();
                Console.WriteLine($"Đã lưu file: {savePath}");
                continue;
            }

            ms?.Write(buffer, 0, buffer.Length);
        }
    }
}
```

✅ Phần 2: Client – Gửi file UDP

```
csharp
CopyEdit
using System;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;

class UdpFileSender
{
    static void Main()
    {
        UdpClient udpClient = new UdpClient();
        string filePath = @"C:\Temp\test.pdf";
        string fileName = Path.GetFileName(filePath);

        // Gửi header chứa tên file
        byte[] header = Encoding.UTF8.GetBytes("<FILE>:" + fileName);
        udpClient.Send(header, header.Length, "127.0.0.1", PORT);
```

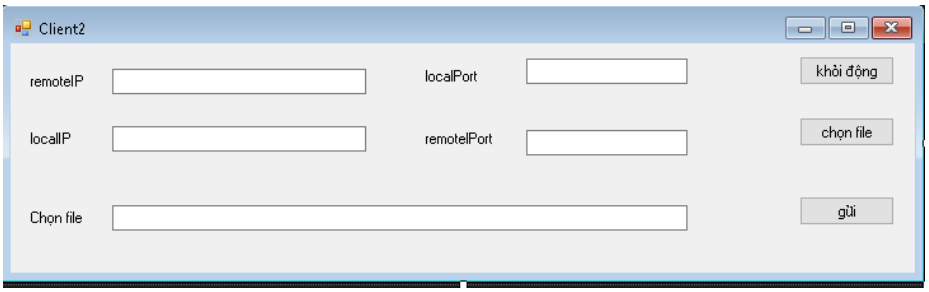
```
        // Gửi nội dung file
        using (FileStream fs = new FileStream(filePath, FileMode.Open, FileAccess.Read))
        {
            byte[] buffer = new byte[BUFFER_SIZE];
            int bytesRead;

            while ((bytesRead = fs.Read(buffer, 0, buffer.Length)) > 0)
            {
                udpClient.Send(buffer, bytesRead, "127.0.0.1", PORT);
                Thread.Sleep(1); // tránh nghẽn buffer
            }
        }

        // Gửi kết thúc file
        byte[] eof = Encoding.UTF8.GetBytes("<EOF>");
        udpClient.Send(eof, eof.Length, "127.0.0.1", PORT);

        Console.WriteLine("Đã gửi xong file.");
    }
}
```

📌 **Lưu ý:**

Mục	Chi tiết
Giới hạn UDP	Gói tối đa ~65KB, thường dùng ~8KB an toàn
Giảm mất gói	Dùng Thread.Sleep(1) hoặc chờ ACK (nâng cao)
Gửi file lớn	Có thể bị mất gói – cần dùng TCP để ổn định
Thư mục lưu file	C:\ReceivedFiles – tạo tự động nếu chưa có

Tiêu chí	client_server3	client2
Tên namespace	client_server3	client2
Chunk Size	8192 bytes (8 KB)	512 bytes
UDP Client	Dùng chung một udpClient cho cả gửi và nhận	Gửi thì tạo UdpClient mới (senderClient), nhận thì dùng udpClient riêng
Cách gửi dữ liệu	Đọc file bằng FileStream, gửi từng phần	Đọc toàn bộ file bằng File.ReadAllBytes, sau đó chia nhỏ
Luồng gửi	Tạo luồng riêng sendThread để gửi	Không dùng luồng riêng

Tiêu chí	client_server3	client2
Xử lý tên file nhận	Dùng cờ receivedFileNameFlag để đánh dấu	Gán tên file ngay khi nhận gói đầu tiên
Lưu file	Ghi file sau khi nhận toàn bộ vào MemoryStream	Tương tự, nhưng lưu sau khi nhận <EOF>
Thread-safe UI	Gọi Invoke khi cập nhật UI (MessageBox)	Cũng dùng Invoke, nhưng có thêm Console.WriteLine

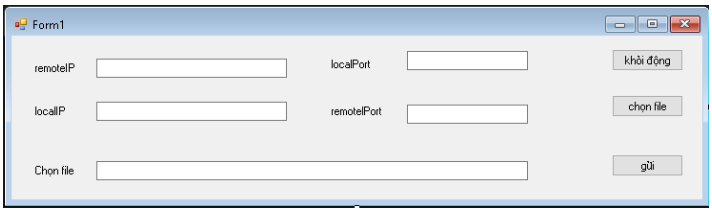



```
public partial class Client2 : Form
{
    public Client2()
    {
        InitializeComponent();
    }

    string fileName;
    string filePath;
    int chunkSize = 512;
    private UdpClient udpClient;
    private IPEndPoint ipRemote;
    private Thread receiveThread;
    private void Form1_Load(object sender, EventArgs e) { }
    private void btnChonFile_Click(object sender, EventArgs e)
    {
        OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*";
        openFileDialog.Title = "Chọn tập tin để gửi";

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            filePath = openFileDialog.FileName;
            txtfileName.Text = filePath;
            fileName = Path.GetFileName(filePath);
```

```
        }
    }
    private void btnStart_Click(object sender, EventArgs e)
    {
        try
        {
            udpClient = new UdpClient(int.Parse(txtlocalP.Text));
            ipRemote = new IPEndPoint(IPAddress.Parse(txtRIP.Text.Trim()),
int.Parse(txtRemoteP.Text));
            receiveThread = new Thread(ReceiveData);
            receiveThread.IsBackground = true;
            receiveThread.Start();
            MessageBox.Show("Đã khởi động chế độ lắng nghe.");
        }
        catch (Exception ex)
        {
            MessageBox.Show("Lỗi khởi động UDP: " + ex.Message);
        }
    }
    private void ReceiveData()
    {
        try
        {
            IPEndPoint remote = new IPEndPoint(IPAddress.Any, 0);
            MemoryStream ms = new MemoryStream();

            // Nhận tên file đầu tiên
            byte[] fileNameBytes = udpClient.Receive(ref remote);
            string receivedFileName = Encoding.UTF8.GetString(fileNameBytes);

            Console.WriteLine("Đang nhận file: " + receivedFileName);

            while (true)
            {
                byte[] data = udpClient.Receive(ref remote);
                string msg = Encoding.UTF8.GetString(data);

                if (msg == "<EOF>")
                {
                    Console.WriteLine("Đã nhận xong file.");
                    break;
                }

                ms.Write(data, 0, data.Length);
            }
            // Lưu với đúng tên file nhận được
```

```
        string folderPath = @"C:\ReceivedFiles";

        // Tạo thư mục nếu chưa có
        if (!Directory.Exists(folderPath))
        {
            Directory.CreateDirectory(folderPath);
        }

        string savePath = Path.Combine(folderPath, receivedFileName);

        // Ghi file
        File.WriteAllBytes(savePath, ms.ToArray());
        MessageBox.Show("Đã lưu file tại: " + savePath);

        Invoke((MethodInvoker)(() =>
        {
            MessageBox.Show("Đã lưu file: " + savePath);
        }));
    }
    catch (Exception ex)
    {
        Invoke((MethodInvoker)(() =>
        {
            MessageBox.Show("Lỗi khi nhận: " + ex.Message);
        }));
    }
}

private void btnSend_Click(object sender, EventArgs e)
{
    try
    {
        if (string.IsNullOrEmpty(filePath))
        {
            MessageBox.Show("Vui lòng chọn file trước khi gửi.");
            return;
        }

        UdpClient senderClient = new UdpClient();

        // Gửi tên file trước
        byte[] fileNameBytes = Encoding.UTF8.GetBytes(fileName);
        senderClient.Send(fileNameBytes, fileNameBytes.Length, ipRemote);

        // Gửi nội dung file theo chunk
        byte[] fileBytes = File.ReadAllBytes(filePath);
```

```
            for (int i = 0; i < fileBytes.Length; i += chunkSize)
            {
                int size = Math.Min(chunkSize, fileBytes.Length - i);
                byte[] chunk = new byte[size];
                Array.Copy(fileBytes, i, chunk, 0, size);
                senderClient.Send(chunk, chunk.Length, ipRemote);
                Thread.Sleep(5); // tránh nghẽn gói
            }

            // Gửi <EOF>
            byte[] eof = Encoding.UTF8.GetBytes("<EOF>");
            senderClient.Send(eof, eof.Length, ipRemote);

            MessageBox.Show("Gửi file thành công.");
        }
        catch (Exception ex)
        {
            MessageBox.Show("Lỗi khi gửi: " + ex.Message);
        }
    }
}
```

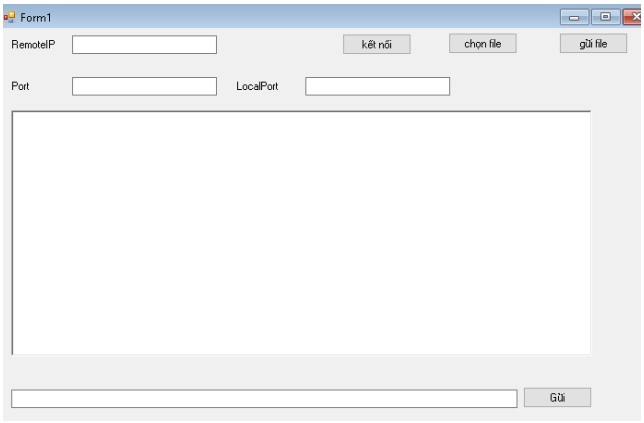



```
public partial class client_server3 : Form
{
    private UdpClient udpClient;
    private IPEndPoint ipRemote;
    private Thread receiveThread;

    string filePath;
    string fileName;
    int chunkSize = 8192; // 8KB cho mỗi gói UDP
    public client_server3()
    {
        InitializeComponent();
    }

    private void btnStart_Click(object sender, EventArgs e)
    {
        int localPort = int.Parse(txtlocalP.Text.Trim());
        int remotePort = int.Parse(txtRemoteP.Text.Trim());
```

```
    string remoteIP = txtRIP.Text.Trim();

    udpClient = new UdpClient(localPort);
    ipRemote = new IPEndPoint(IPAddress.Parse(remoteIP), remotePort);

    receiveThread = new Thread(ReceiveData);
    receiveThread.IsBackground = true;
    receiveThread.Start();

    MessageBox.Show("Bắt đầu lắng nghe...");
}

private void btnChonFile_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "All files (*.*)|*.*";
    openFileDialog.Title = "Chọn tập tin để gửi";

    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        filePath = openFileDialog.FileName;
        fileName = Path.GetFileName(filePath);
        txtfileName.Text = filePath;
    }
}

private void btnSend_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(filePath) || !File.Exists(filePath))
    {
        MessageBox.Show("Vui lòng chọn file hợp lệ.");
        return;
    }

    Thread sendThread = new Thread(SendFile);
    sendThread.IsBackground = true;
    sendThread.Start();
}

private void client_server3_Load(object sender, EventArgs e)
{

}
private void SendFile()
{
    try
```

```
    {
        // Gửi tên file trước
        byte[] nameBytes = Encoding.UTF8.GetBytes(fileName);
        udpClient.Send(nameBytes, nameBytes.Length, ipRemote);
        Thread.Sleep(100); // tránh trùng gói

        // Gửi nội dung file theo từng chunk
        byte[] buffer = new byte[chunkSize];
        using (FileStream fs = new FileStream(filePath, FileMode.Open, FileAccess.Read))
        {
            int bytesRead;
            while ((bytesRead = fs.Read(buffer, 0, buffer.Length)) > 0)
            {
                udpClient.Send(buffer, bytesRead, ipRemote);
                Thread.Sleep(5); // delay nhỏ tránh nghẽn mạng
            }
        }

        // Gửi ký hiệu kết thúc
        byte[] endSignal = Encoding.UTF8.GetBytes("<EOF>");
        udpClient.Send(endSignal, endSignal.Length, ipRemote);

        MessageBox.Show("Gửi file thành công!");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Lỗi gửi: " + ex.Message);
    }
}
private void ReceiveData()
{
    try
    {
        string receivedFileName = "";
        bool receivedFileNameFlag = false;
        MemoryStream ms = new MemoryStream();

        while (true)
        {
            IPEndPoint remote = new IPEndPoint(IPAddress.Any, 0);
            byte[] data = udpClient.Receive(ref remote);

            string msg = Encoding.UTF8.GetString(data);

            if (!receivedFileNameFlag)
            {
```

```
                receivedFileName = msg;
                receivedFileNameFlag = true;
                continue;
            }

            if (msg == "<EOF>")
            {
                string saveDir = @"C:\ReceivedFiles";
                if (!Directory.Exists(saveDir))
                    Directory.CreateDirectory(saveDir);

                string savePath = Path.Combine(saveDir, receivedFileName);
                File.WriteAllBytes(savePath, ms.ToArray());
                ms.Dispose();

                this.Invoke((MethodInvoker)(() =>
                {
                    MessageBox.Show("Đã nhận xong file: " + receivedFileName);
                }));

                // Reset
                receivedFileNameFlag = false;
                ms = new MemoryStream();
            }
            else
            {
                ms.Write(data, 0, data.Length);
            }
        }
    }
    catch (Exception ex)
    {
        this.Invoke((MethodInvoker)(() =>
        {
            MessageBox.Show("Lỗi nhận: " + ex.Message);
        }));
    }
  }
}
```

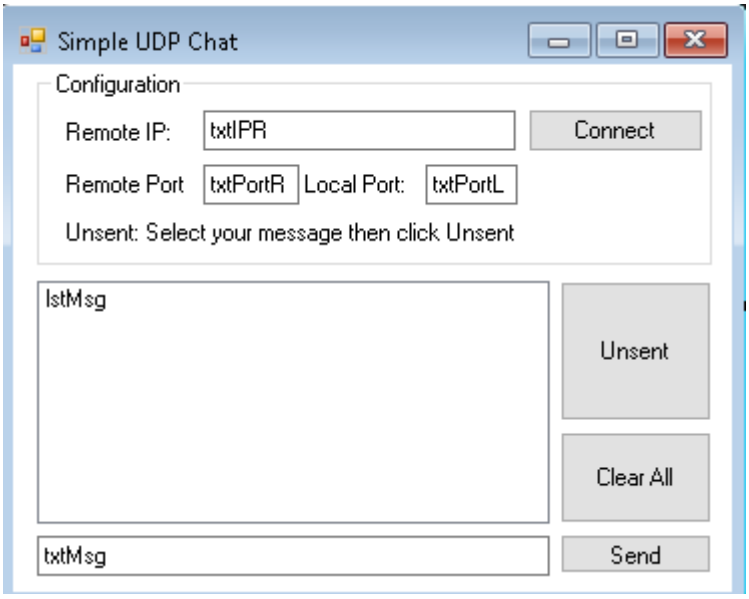



```
public partial class Form1: Form
{
    public Form1()
    {
        InitializeComponent();
    }
    private Socket udpLocal = null;
    private IPEndPoint ipRemote, ipLocal;
    Thread getThread = null;

    string filePath = "", fileName = "";
    const int BUFFER_SIZE = 8192;

    private void Form1_Load(object sender, EventArgs e)
    {
        CheckForIllegalCrossThreadCalls = false;
        this.FormClosed += new FormClosedEventHandler(closeForm);
    }
    private void closeForm(object sender, EventArgs e)
    {
        try
        {
            getThread?.Abort();
            udpLocal?.Close();
        }
        catch { }
    }

    private void openinput(bool state)
    {
        btnGui.Enabled = !state;
        //btnSendFile.Enabled = !state;
        //btnBrowseFile.Enabled = !state;
        txtIPR.ReadOnly = !state;
```

```
        txtPortL.ReadOnly = !state;
        txtPortR.ReadOnly = !state;
        btntKN.Enabled = state;
    }

    private void btntKN_Click(object sender, EventArgs e)
    {
        try
        {
            int portL = int.Parse(txtPortL.Text.Trim());
            int portR = int.Parse(txtPortR.Text.Trim());

            udpLocal = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
ProtocolType.Udp);
            ipLocal = new IPEndPoint(IPAddress.Parse("0.0.0.0"), portL);
            udpLocal.Bind(ipLocal);

            ipRemote = new IPEndPoint(IPAddress.Parse(txtIPR.Text.Trim()), portR);

            openinput(false);
            getThread = new Thread(new ThreadStart(ReceiveData));
            getThread.Start();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Lỗi kết nối: " + ex.Message);
        }
    }

    private void btnGui_Click(object sender, EventArgs e)
    {
        try
        {
            byte[] data = Encoding.UTF8.GetBytes(txtMsg.Text);
            udpLocal.SendTo(data, ipRemote);
            rtxMsg.AppendText("Send: " + txtMsg.Text + "\r\n");
        }
        catch (Exception ex)
        {
            MessageBox.Show("Lỗi gửi tin nhắn: " + ex.Message);
        }
    }

    private void ReceiveData()
    {
        try
```

```
    {
        EndPoint remote = new IPEndPoint(IPAddress.Any, 0);
        MemoryStream ms = null;
        string receivedFileName = "";
        while (true)
        {
            byte[] buffer = new byte[BUFFER_SIZE];
            int rec = udpLocal.ReceiveFrom(buffer, ref remote);
            string header = Encoding.UTF8.GetString(buffer, 0, rec);

            if (header.StartsWith("<FILE>:"))
            {
                receivedFileName = header.Substring(7);
                ms = new MemoryStream();
                continue;
            }

            if (header == "<EOF>")
            {
                string saveDir = @"C:\ReceivedFiles";
                Directory.CreateDirectory(saveDir);
                string savePath = Path.Combine(saveDir, receivedFileName);
                File.WriteAllBytes(savePath, ms.ToArray());
                rtxMsg.AppendText($"[Đã nhận file: {receivedFileName}]\r\n");
                ms.Close();
                ms = null;
                continue;
            }

            if (ms != null)
            {
                ms.Write(buffer, 0, rec);
            }
            else
            {
                string text = Encoding.UTF8.GetString(buffer, 0, rec);
                rtxMsg.AppendText("Receive: " + text + "\r\n");
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Lỗi nhận dữ liệu: " + ex.Message);
    }
}
```

```
        private void btnBrowseFile_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                filePath = ofd.FileName;
                fileName = Path.GetFileName(filePath);
                txtMsg.Text = fileName;
            }
        }

        private void btnSendFile_Click(object sender, EventArgs e)
        {
            try
            {
                if (string.IsNullOrEmpty(filePath)) return;

                // Gửi tên file
                string header = "<FILE>:" + fileName;
                byte[] headerBytes = Encoding.UTF8.GetBytes(header);
                udpLocal.SendTo(headerBytes, ipRemote);

                // Gửi nội dung file
                using (FileStream fs = new FileStream(filePath, FileMode.Open, FileAccess.Read))
                {
                    byte[] buffer = new byte[BUFFER_SIZE];
                    int bytesRead;
                    while ((bytesRead = fs.Read(buffer, 0, buffer.Length)) > 0)
                    {
                        udpLocal.SendTo(buffer.Take(bytesRead).ToArray(), ipRemote);
                        Thread.Sleep(1); // tránh nghẽn buffer
                    }
                }

                // Gửi EOF
                byte[] eof = Encoding.UTF8.GetBytes("<EOF>");
                udpLocal.SendTo(eof, ipRemote);
                rtxMsg.AppendText($"[Đã gửi file: {fileName}]\r\n");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Lỗi gửi file: " + ex.Message);
            }
        }
    }
\
```




```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
        CheckForIllegalCrossThreadCalls = false;
    }
    private UdpClient udpLocal = null;
    private IPEndPoint ipremote;
    Thread getThread = null;

    private void openinput(bool state)
    {
        this.btnGui.Enabled = !state;
        this.txtIPR.ReadOnly = !state;
        this.txtPortL.ReadOnly = !state;
        this.txtPortR.ReadOnly = !state;
        this.btnKN.Enabled = state;
```

```
}
private bool IsContainFilter(string msg,string[] key)
{
    string[] msgToken = msg.Split(' ');
    for (int i = 0; i < msgToken.Length; i++)
    {
        //MessageBox.Show(msgToken[i]);
        for (int j = 0; j < key.Length; j++)
            if (msgToken[i].Equals(key[j]))
                return true;
    }
    return false;
}
private void NhanDL()
{
    try
    {
        while (true)
        {
            byte[] data = new byte[1024];
            IPEndPoint ipe = new IPEndPoint(IPAddress.Any, int.Parse(txtPortL.Text.Trim()));
            data = udpLocal.Receive(ref ipe);
            string s = Encoding.UTF8.GetString(data, 0, data.Length);
            if(s.StartsWith("UNS:"))
            {
                string[] tmp=s.Split(' ');
                lstMsg.Items[Int32.Parse(tmp[1])] = "UNSENT MESSAGE [FRIEND]";
```

```
            }
            else if(s.StartsWith("CLE:"))
            {
               lstMsg.Items.Clear();
            }
            else
               lstMsg.Items.Add("REC: " + s);
         }
      }
      catch (Exception ex)
      {
         MessageBox.Show(ex.ToString());
      }
   }
   private void btnKN_Click(object sender, EventArgs e)
   {
      try
      {
         udpLocal = new
UdpClient(int.Parse(txtPortL.Text.Trim()),AddressFamily.InterNetwork);
         ipremote = new IPEndPoint(IPAddress.Parse(txtIPR.Text.Trim()),
int.Parse(txtPortR.Text.Trim()));
         openinput(false);
         getThread = new Thread(new ThreadStart(NhanDL));
         getThread.IsBackground = true;
         getThread.Start();
      }
      catch (Exception ex)
      {
```

```
        openinput(true);
        MessageBox.Show(ex.ToString());
    }
}
private void btnGui_Click(object sender, EventArgs e)
{
    try
    {
    byte[] data = new byte[1024];
    data = Encoding.UTF8.GetBytes(txtMsg.Text);
    udpLocal.Send(data,data.Length,ipremote);
    lstMsg.Items.Add("SEN: " + txtMsg.Text);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
private void btnUnsent_Click(object sender, EventArgs e)
{
    int index = lstMsg.SelectedIndex;
    string line = lstMsg.Items[index].ToString();
    if (line.Substring(0, 4) == "SEN:")
    {
        byte[] data = new byte[1024];
        data = Encoding.UTF8.GetBytes("UNS: "+index.ToString());
        udpLocal.Send(data, data.Length, ipremote);
      lstMsg.Items[index] = "UNSENT MESSAGE [YOU]";
```

```
        }
    }
    private void btnClearAll_Click(object sender, EventArgs e)
    {
        byte[] data = new byte[1024];
        data = Encoding.UTF8.GetBytes("CLE:");
        udpLocal.Send(data, data.Length, ipremote);
        lstMsg.Items.Clear();
    }
}
```

Broadcast

```
class UdpBroadcastReceiver
{
    static void Main()
    {
        // Mở cổng 8888 để lắng nghe broadcast
        UdpClient receiver = new UdpClient(8888);
        IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0);

        Console.WriteLine("Đang lắng nghe Broadcast trên cổng 8888...");

        while (true)
        {
            byte[] data = receiver.Receive(ref remoteEP);
```

```
            string message = Encoding.UTF8.GetString(data);

            Console.WriteLine($"Nhận từ {remoteEP.Address}: {message}");
        }
    }
}



using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

class UdpBroadcastSender
{//client
    static void Main()
    {
        UdpClient sender = new UdpClient();
        sender.EnableBroadcast = true; // Bắt buộc để gửi broadcast

        IPEndPoint broadcastEP = new IPEndPoint(IPAddress.Broadcast, 8888);

        Console.Write("Nhập nội dung muốn gửi: ");
        string message = Console.ReadLine();

        byte[] data = Encoding.UTF8.GetBytes(message);

        sender.Send(data, data.Length, broadcastEP);
```

```
        Console.WriteLine("Đã gửi broadcast!");


        sender.Close();
    }
}
```

Multicast Receiver (máy nhận tham gia nhóm multicast)

```
csharp
CopyEdit
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

class MulticastReceiver
{
    static void Main()
    {
        const int port = 5000;
        const string multicastIP = "224.0.0.1";

        UdpClient client = new UdpClient(port);
        client.JoinMulticastGroup(IPAddress.Parse(multicastIP)); // Tham gia
nhóm

        Console.WriteLine($"[Receiver] Đang lắng nghe multicast trên
{multicastIP}:{port}");

        IPEndPoint remoteEP = new IPEndPoint(IPAddress.Any, 0);

        while (true)
        {
            byte[] data = client.Receive(ref remoteEP);
            string message = Encoding.UTF8.GetString(data);
            Console.WriteLine($"Nhận từ {remoteEP.Address}: {message}");
        }
    }
}
```

Multicast Sender (máy gửi tới nhóm)

```
csharp
CopyEdit
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

class MulticastSender
{
    static void Main()
    {
```

```
        const int port = 5000;
        const string multicastIP = "224.0.0.1";

        UdpClient client = new UdpClient();
        IPEndPoint multicastEP = new IPEndPoint(IPAddress.Parse(multicastIP),
port);

        Console.Write("Nhập nội dung cần gửi multicast: ");
        string message = Console.ReadLine();

        byte[] data = Encoding.UTF8.GetBytes(message);
        client.Send(data, data.Length, multicastEP);

        Console.WriteLine("Đã gửi multicast!");
        client.Close();
    }
}
```