**LAB 3**

[I. Mục đích 3](#_Toc22764)

[II. Yêu cầu 3](#_Toc22765)

[III. Hướng dẫn 3](#_Toc22766)

[III.1. Xây dựng chương trình UDP Client – Server đơn giản 3](#_Toc22767)

[III.1.1. Hướng dẫn lập trình UDP Server 3](#_Toc22768)

[III.1.2. Hướng dẫn lập trình UDP Client 4](#_Toc22769)

[III.1.3. Test chương trình: 4](#_Toc22770)

[III.1.4. Trả lời câu hỏi: 5](#_Toc22771)

[III.2. Bài tập 5](#_Toc22772)

[III.3. Cải tiến chương trình UDP client-server để có thể gởi và nhận dữ liệu liên tục 6](#_Toc22773)

[III.3.1. Hướng dẫn lập trình UDP Client 6](#_Toc22774)

[III.3.2. Hướng dẫn lập trình UDP Server 6](#_Toc22775)

[III.3.3. Test chương trình 6](#_Toc22776)

[III.3.4. Trả lời câu hỏi: 7](#_Toc22777)

[III.4. Sử dụng phương thức Connect ở client để thiết lập kết nối trước với server 8](#_Toc22778)

[III.4.1. Hướng dẫn lập trình 8](#_Toc22779)

[III.4.2. Test chương trình 8](#_Toc22780)

[III.4.3. Trả lời câu hỏi: 9](#_Toc22781)

[III.5. Kiểm tra khả năng phân biệt biên thông điệp của giao thức UDP 10](#_Toc22782)

[III.5.1. Hướng dẫn lập trình UDP Client 10](#_Toc22783)

[III.5.2. Hướng dẫn lập trình UDP Client 10](#_Toc22784)

[III.5.3. Test chương trình 10](#_Toc22785)

[III.5.4. Trả lời câu hỏi: 11](#_Toc22786)

[III.6. Ngăn cản mất dữ liệu khi lập trình mạng sử dụng giao thức UDP 12](#_Toc22787)

[III.6.1. Trả lời câu hỏi: 12](#_Toc22788)

[III.6.2. Hướng dẫn lập trình để ngăn cản mất dữ liệu 12](#_Toc22789)

[III.6.3. Test chương trình 13](#_Toc22790)

[III.6.4. Trả lời câu hỏi: 14](#_Toc22791)

[III.7. Ngăn cản mất gói tin khi lập trình mạng sử dụng giao thức UDP 14](#_Toc22792)

[III.7.1. Hướng dẫn lập trình để ngăn cản mất gói tin 14](#_Toc22793)

[III.7.2. Test chương trình 17](#_Toc22794)

[III.7.3. Trả lời câu hỏi: 18](#_Toc22795)

# I. Mục đích

Lập trình client – server sử dụng giao thức UDP

Nắm được các lỗi thường xảy ra khi lập trình theo giao thức UDP và cách hạn chế các lỗi này

# II. Yêu cầu

1) Xây dựng chương trình UDP Client – Server đơn giản (xem hướng dẫn)

# III. Hướng dẫn

### III.1. Xây dựng chương trình UDP Client – Server đơn giản

##### III.1.1. Hướng dẫn lập trình UDP Server

Để lập trình socket ta sử dụng 2 namespace:

using System.Net;

using System.Net.Sockets;

Tạo Server EndPoint, EndPoint này sẽ tham chiếu đến địa chỉ IP và Port của Server:

IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 5000);

Tạo Server Socket, Socket này dùng để trao đổi dữ liệu với client

Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,

ProtocolType.Udp);

Chương trình UDP Server khác với chương trình TCP Server ở chỗ nó không lắng nghe kết nối, trên socket ta chỉ việc Bind nó với Server EndPoint

serverSocket.Bind(serverEndPoint);

Khi client kết nối tới nó sẽ hiển thị thông tin của client đang kết nối đến:

Console.WriteLine(remote.ToString());

Để nhận dữ liệu từ client gởi lên ta dùng hàm ReceiveFrom với chú ý EndPoint chứa thông tin của client kết nối đến phải được truyền tham chiếu

serverSocket.ReceiveFrom(buff, 0, buff.Length,SocketFlags.None, ref remote);

##### III.1.2. Hướng dẫn lập trình UDP Client

Tạo Server Socket, Socket này sẽ được dùng để gởi dữ liệu tới Server

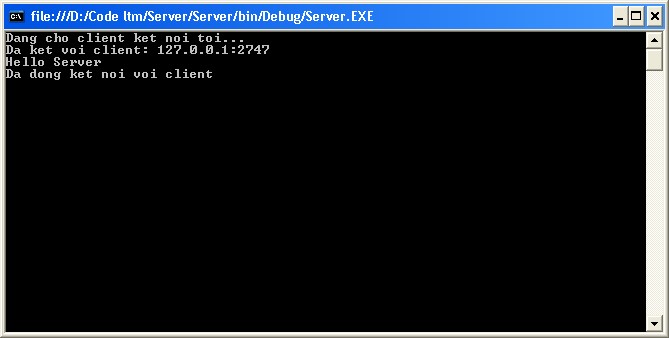
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

Gởi câu chào lên server, câu chào này sẽ được đặt trong mảng buff

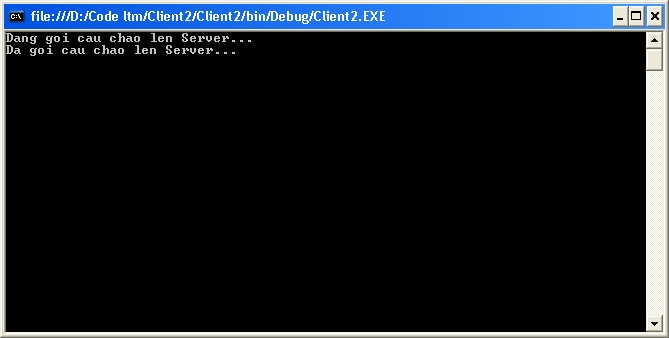
server.SendTo(buff, buff.Length, SocketFlags.None, serverEndPoint);

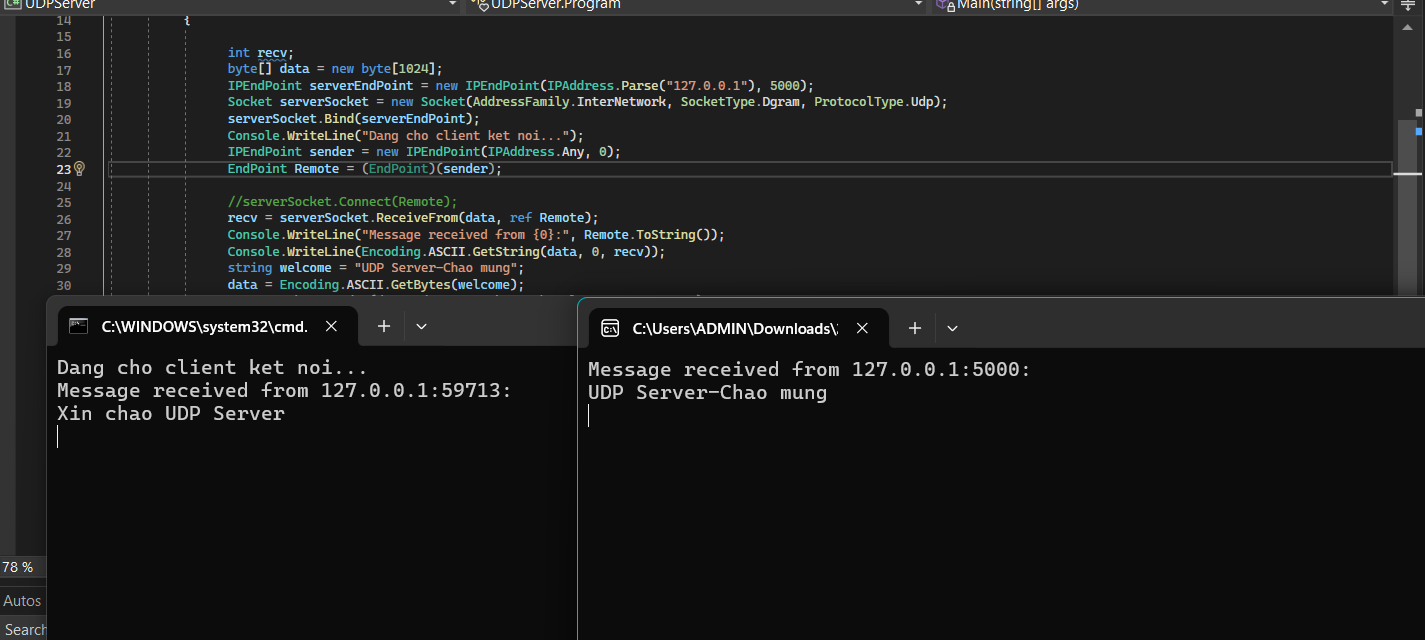
##### III.1.3. Test chương trình:

Chạy chương trình Server:



Chạy chương trình ở client:





##### III.1.4. Trả lời câu hỏi:

1. Như hình trên port 2747 ở đâu ra

Vì UDP là một giao thức không kết nối (connectionless), các cổng được sử dụng để phân biệt các dịch vụ hoặc ứng dụng đang giao tiếp trên cùng một thiết bị. Cổng 2747 trong trường hợp này có thể là một cổng ngẫu nhiên do hệ điều hành chỉ định nếu không có yêu cầu đặc biệt từ ứng dụng.

1. Có phải lúc nào client cũng mở port 2747 để kết nối với Server không ?

Thông thường: Client sử dụng cổng ngẫu nhiên (ephemeral) để kết nối đến server.

Trường hợp đặc biệt: Nếu ứng dụng được lập trình để sử dụng cổng 2747 cố định, thì client sẽ luôn mở cổng này khi kết nối đến server.Tuy nhiên mình nghĩ cách này thì dễ gây xung đột, vì có trường hợp sẽ bị trùng port

1. Tại sao khi lập trình mạng dùng giao thức UDP thì client phải gởi câu chào lên server trước?

Client gửi câu chào trước để thông báo cho server về sự hiện diện của nó và cung cấp địa chỉ IP và cổng nguồn mà server có thể dùng để trả lời. Đây là yêu cầu cần thiết trong mô hình UDP, nơi không có kết nối "cố định" trước khi bắt đầu truyền dữ liệu.

### III.2. Bài tập

1. Cải tiến chương trình để ở client gõ “exit” thì đóng client, khi client gõ “exit all” thì đóng cả client và server.

|  |  |
| --- | --- |
| UDP Client  using System;  using System.Collections.Generic;  using System.Linq;  using System.Net.Sockets;  using System.Net;  using System.Text;  using System.Threading.Tasks;  namespace UDPClient  {  internal class Program  {  public static void Main()  {  try  {  byte[] data = new byte[1024];  string input, str;  IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 5000);  Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);  server.EnableBroadcast = true;  string welcome = "Xin chao UDP Server";  data = Encoding.ASCII.GetBytes(welcome);  server.SendTo(data, data.Length, SocketFlags.None, ipep);  IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);  EndPoint Remote = (EndPoint)sender;  data = new byte[1024];  int recv = server.ReceiveFrom(data, ref Remote);  Console.WriteLine("Message received from {0}:", Remote.ToString());  Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));  while (true)  {  input = Console.ReadLine();  if (input == "exit all")  {  server.SendTo(Encoding.ASCII.GetBytes(input), Remote);  server.Close();  break;  }  if (input == "exit")  {  string exit = "Client stopped";  data = Encoding.ASCII.GetBytes(exit);  server.SendTo(data, data.Length, SocketFlags.None, ipep);  break;  }  server.SendTo(Encoding.ASCII.GetBytes(input), Remote);  data = new byte[1024];  recv = server.ReceiveFrom(data, ref Remote);  str = Encoding.ASCII.GetString(data, 0, recv);  Console.WriteLine(str);  }  }  catch (SocketException ex)  {  Console.WriteLine("Server chua khoi dong");  }  Console.ReadKey();  }  }  } | UDP Server  using System;  using System.Collections.Generic;  using System.Linq;  using System.Net.Sockets;  using System.Net;  using System.Text;  using System.Threading.Tasks;  namespace UDPServer  {  internal class Program  {  static void Main(string[] args)  {  int recv;  byte[] data = new byte[1024];  IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 5000);  Socket serverSocket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);  serverSocket.Bind(serverEndPoint);  Console.WriteLine("Dang cho client ket noi...");  IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);  EndPoint Remote = (EndPoint)(sender);  //serverSocket.Connect(Remote);  recv = serverSocket.ReceiveFrom(data, ref Remote);  Console.WriteLine("Message received from {0}:", Remote.ToString());  Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));  string welcome = "UDP Server-Chao mung";  data = Encoding.ASCII.GetBytes(welcome);  serverSocket.SendTo(data, data.Length, SocketFlags.None, Remote);    while (true)  {  data = new byte[1024];  recv = serverSocket.ReceiveFrom(data, ref Remote);  if (Encoding.ASCII.GetString(data, 0, recv) == "exit all")  {  serverSocket.Close();  break;  }  Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));  serverSocket.SendTo(data, recv, SocketFlags.None, Remote);  }  }  }  } |

### III.3. Cải tiến chương trình UDP client-server để có thể gởi và nhận dữ liệu liên tục

##### III.3.1. Hướng dẫn lập trình UDP Client

Dùng vòng lặp vô hạn để khi người dùng nhập dữ liệu xong thì sẽ gởi lên server và chờ server gởi kết quả trả vể, lấy kết quả đó hiển thị lên màn hình

while (true)

{

str = Console.ReadLine(); buff = Encoding.ASCII.GetBytes(str);

serverSocket.SendTo(buff, 0, buff.Length, SocketFlags.None, remote); byteReceive = serverSocket.ReceiveFrom(buff, 0, buff.Length, SocketFlags.None, ref remote);

str = Encoding.ASCII.GetString(buff, 0, byteReceive);

Console.WriteLine(str);

}

##### III.3.2. Hướng dẫn lập trình UDP Server

Dùng vòng lặp vô hạn lấy kết quả client gởi lên, chuyển nó thành chuỗi và hiển thị lên màn hình đồng thời gởi lại dữ liệu nhận được về lại cho client

while (true)

{

buff = new byte[1024];

byteReceive = serverSocket.ReceiveFrom(buff, 0, buff.Length, SocketFlags.None, ref remote);

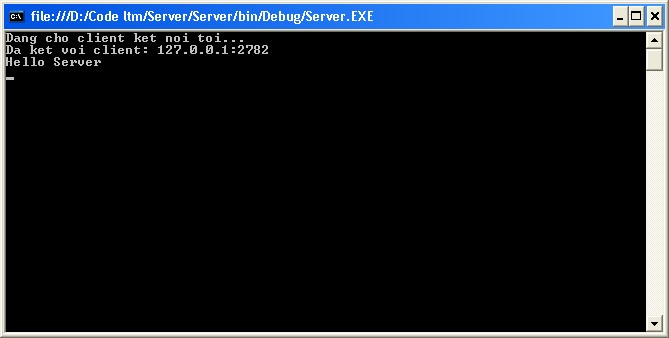
str = Encoding.ASCII.GetString(buff, 0, byteReceive); Console.WriteLine(str);

serverSocket.SendTo(buff, 0, buff.Lengh, SocketFlags.None, remote);

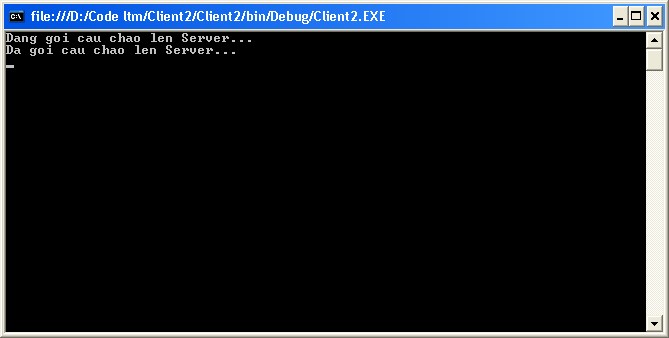
}

##### III.3.3. Test chương trình

Chạy chương trình ở server:



Chạy chương trình ở client:



Nhập thông điệp từ client và gởi lên server, lỗi sẽ xảy ra, hãy sửa lại cho hết lỗi

##### III.3.4. Trả lời câu hỏi:

1. Khi chạy chương trình với đoạn code gởi nhận dữ liệu như trên, lúc chưa nhập dữ liệu cho client để gởi lên thì không xảy ra lỗi nhưng khi nhập dữ liệu để gởi lên server sẽ xảy ra lỗi, vì sao lại xảy ra lỗi này ?

Lỗi xảy ra khi client gửi dữ liệu lên server thường do server không chạy, địa chỉ hoặc cổng sai, hoặc vấn đề với dữ liệu hoặc mạng.

1. Khi server chưa bật thì chương trình trên có bị lỗi không? Tạo sao

Client **không nhận biết** ngay lập tức rằng server chưa bật. Dữ liệu vẫn được gửi đi, nhưng nếu server không lắng nghe, gói tin sẽ bị "mất", không đến được server.

Để tránh lỗi, cần thêm cơ chế xử lý lỗi, timeout và thông báo người dùng trong trường hợp server không hoạt động.

1. Khi đang chạy chương trình tắt client thì chương trình trên có bị lỗi không? Tại sao

**Không gặp lỗi ngay**: Vì UDP không dựa vào kết nối, server không quan tâm đến việc client có tắt hay không, miễn là server không cần nhận thêm dữ liệu từ client đó. Server sẽ tiếp tục chạy và lắng nghe các gói tin từ các client khác (nếu có).

### III.4. Sử dụng phương thức Connect ở client để thiết lập kết nối trước với server

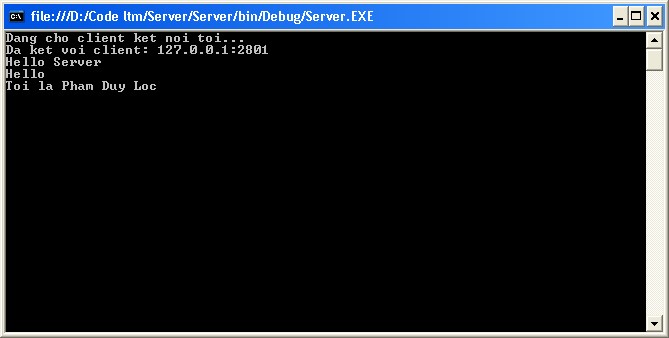
##### III.4.1. Hướng dẫn lập trình

Việc này được thực hiện bằng cách sử dụng phương thức Connect() ở lớp socket serverSocket.Connect(remote);

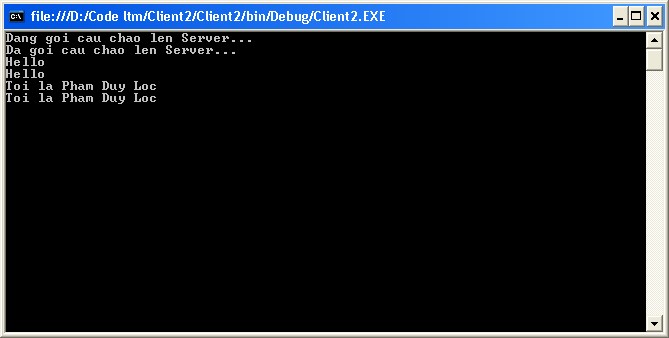
Sau khi gọi phương thức Connect() xong, ta có thể lập trình giống như lập trình với giao thức UDP với client

##### III.4.2. Test chương trình

Chạy chương trình ở server:



Chạy chương trình ở client:



##### III.4.3. Trả lời câu hỏi:

1. Khi chạy chương trình mà server chưa được bật thì có hiện tượng gì xảy ra? Tại sao lại có hiện tượng này?

Khi server chưa bật, client sẽ vẫn gửi gói tin UDP, nhưng không có server để nhận, dẫn đến việc gói tin bị mất.

Không có lỗi ngay lập tức, nhưng nếu client mong đợi phản hồi từ server, có thể xảy ra lỗi timeout hoặc chương trình không phản hồi.

Để tránh lỗi, cần thêm cơ chế timeout, retry, và xử lý ngoại lệ trong chương trình client.

1. Khi đang chạy tắt server thì chương trình trên có bị lỗi không? Tạo sao

Khi server bị tắt trong lúc client vẫn đang chạy, client sẽ không nhận biết điều này ngay lập tức. Client vẫn sẽ tiếp tục gửi các gói tin như bình thường, nhưng gói tin đó sẽ bị **mất** vì server không còn lắng nghe trên cổng UDP nữa.

1. Khi đang chạy chương trình tắt client thì chương trình trên có bị lỗi không? Tại sao

Khi bạn tắt client, server không có cách nào biết ngay lập tức rằng client đã bị tắt. Server chỉ đơn giản là không nhận được dữ liệu từ client nữa.

### III.5. Kiểm tra khả năng phân biệt biên thông điệp của giao thức UDP

##### III.5.1. Hướng dẫn lập trình UDP Client

Thay đoạn code gởi và nhận thông điệp vô hạn bằng đoạn code gởi 5 thông điệp phân biệt lên server

for (int i = 1; i <= 5; i++)

{

buff = Encoding.ASCII.GetBytes("Thong Diep " + i.ToString()); serverSocket.SendTo(buff, 0, buff.Length, SocketFlags.None, remote); }

##### III.5.2. Hướng dẫn lập trình UDP Server

Thay đoạn code gởi và nhận thông điệp vô hạn bằng đoạn code nhận 5 thông điệp trên server

for (int i = 1; i <= 5; i++)

{

byteReceive = serverSocket.ReceiveFrom(buff, 0, buff.Length, SocketFlags.None, ref remote);

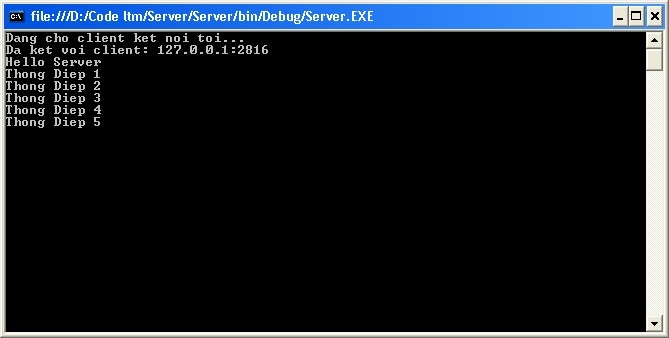
str = Encoding.ASCII.GetString(buff, 0, byteReceive);

Console.WriteLine(str);

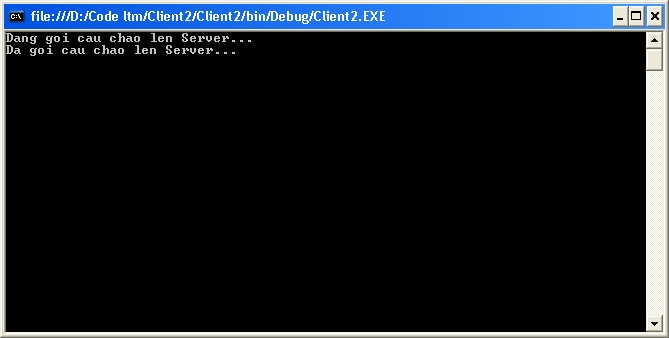
}

##### III.5.3. Test chương trình

Chạy chương trình server:



Chạy chương trình client:



#### III.5.4. Trả lời câu hỏi:

1. Tại sao khi lập trình bằng giao thức UDP các thông điệp được phân biệt với nhau ?

Trong UDP, việc phân biệt các thông điệp được thực hiện chủ yếu nhờ vào việc sử dụng cổng nguồn, cổng đích, và địa chỉ IP. Trong khi UDP không cung cấp cơ chế tích hợp để phân biệt hay sắp xếp gói tin, các ứng dụng có thể sử dụng các kỹ thuật bổ sung như số hiệu gói tin hoặc định dạng dữ liệu để phân biệt các thông điệp và xử lý chúng theo cách mong muốn.

## 

## III.6. Ngăn cản mất dữ liệu khi lập trình mạng sử dụng giao thức UDP

Trong đoạn code gởi và nhận thông điệp của chương trình UDP Client đơn giản ở trên ta thay đổi kích thước bộ đệm như sau:

while (true)

{

str = Console.ReadLine(); buff = Encoding.ASCII.GetBytes(str);

serverSocket.SendTo(buff, 0, buff.Length, SocketFlags.None, remote); **buff = new byte[10];**

byteReceive = serverSocket.ReceiveFrom(buff, 0, buff.Length, SocketFlags.None, ref remote);

str = Encoding.ASCII.GetString(buff, 0, byteReceive);

Console.WriteLine(str);

}

#### III.6.1. Trả lời câu hỏi:

1. Khi gởi dữ liệu với kích thước lớn hơn 10 byte thì có lỗi xảy ra không? Tại sao?

Khi bạn nhận dữ liệu với ReceiveFrom, nếu dữ liệu gửi lớn hơn kích thước bộ đệm, chỉ một phần của dữ liệu (tối đa là kích thước bộ đệm) sẽ được đọc vào bộ đệm. Phần còn lại của dữ liệu sẽ bị mất trong quá trình nhận.

#### III.6.2. Hướng dẫn lập trình để ngăn cản mất dữ liệu

Thay đoạn code gởi và nhận thông điệp vô hạn bằng đoạn code gởi 5 thông điệp phân biệt lên server

while (true)

{

input = Console.ReadLine(); if (input == "exit") break;

server.SendTo(Encoding.ASCII.GetBytes(input), tmpRemote); data = new byte[i]; try {

recv = server.ReceiveFrom(data, ref tmpRemote); stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

}

catch (SocketException)

{

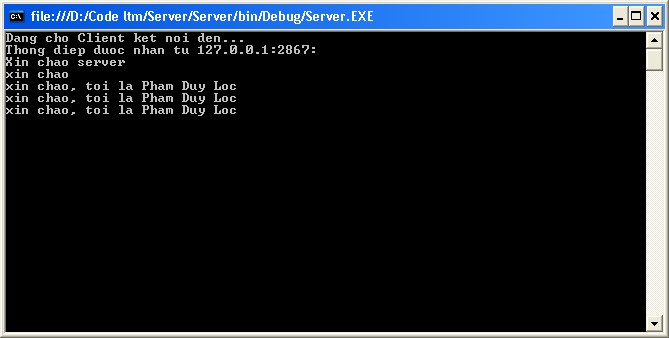
Console.WriteLine("Canh bao: du lieu bi mat, hay thu lai"); i += 10;

}

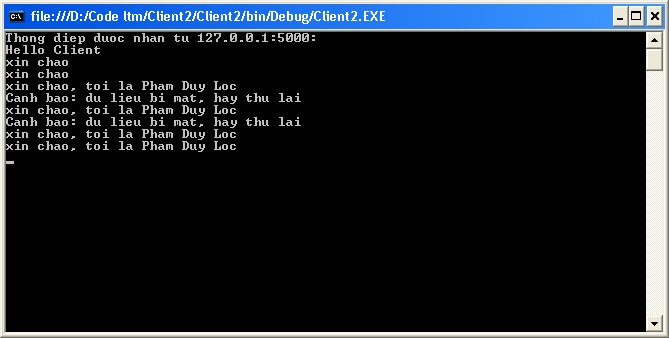
}

#### III.6.3. Test chương trình

Chạy chương trình server:



Chạy chương trình client:



#### III.6.4. Trả lời câu hỏi:

1. Khi dùng phương pháp này các dữ liệu bị mất do kích thước bộ đệm ban đầu nhỏ có lấy lại được không?

UDP không cung cấp cơ chế để khôi phục dữ liệu bị mất hoặc để yêu cầu gửi lại dữ liệu. Dữ liệu bị mất do bộ đệm nhỏ không thể lấy lại từ UDP.

Dữ liệu bị bỏ qua: Phần dữ liệu không được lưu vào bộ đệm sẽ bị bỏ qua và không thể được phục hồi từ gói tin UDP đó.

## III.7. Ngăn cản mất gói tin khi lập trình mạng sử dụng giao thức UDP

#### III.7.1. Hướng dẫn lập trình để ngăn cản mất gói tin

Khi gởi dữ liệu bằng giao thức UDP với các chương trình UDP đơn giản ở trên, nếu dữ liệu không đến được tới đích vì một lý do nào đó thì không thể nào biết được gói tin gởi đi đã bị mất.

Để ngăn cản mất gói tin ta dùng phương thức SetSocketOption() để thiết lập giá trị TimeOut để sau một thời gian không nhận được hồi báo thì gởi lại dữ liệu server.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout, 3000); Và viết lại hàm gởi và nhận dữ liệu:

private int SndRcvData(Socket s, byte[] message, EndPoint rmtdevice)

{ int recv; int retry = 0; while (true)

{

Console.WriteLine("Truyen lai lan thu: #{0}", retry); try

{

s.SendTo(message, message.Length, SocketFlags.None, rmtdevice); data = new byte[1024];

recv = s.ReceiveFrom(data, ref Remote);

}

catch (SocketException)

{ recv = 0;

} if (recv > 0)

{

return recv;

} else { retry++; if (retry > 4)

{ return 0;

}

}

}

}

Viết lại class điều khiển việc gởi và nhận dữ liệu ngăn cản mất gói tin

public RetryUdpClient()

{

string input, stringData; int recv;

IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 5000);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

int sockopt = (int)server.GetSocketOption(SocketOptionLevel.Socket,

SocketOptionName.ReceiveTimeout);

Console.WriteLine("Gia tri timeout mac dinh: {0}", sockopt); server.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout, 3000);

sockopt = (int)server.GetSocketOption(SocketOptionLevel.Socket,

SocketOptionName.ReceiveTimeout);

Console.WriteLine("Gia tri timeout moi: {0}", sockopt); string welcome = "Xin chao Server"; data = Encoding.ASCII.GetBytes(welcome); recv = SndRcvData(server, data, ipep); if (recv > 0)

{

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

} else

{

Console.WriteLine("Khong the lien lac voi thiet bi o xa"); return; } while (true)

{

input = Console.ReadLine(); if (input == "exit") break;

recv = SndRcvData(server, Encoding.ASCII.GetBytes(input), ipep); if (recv > 0)

{

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

} else

Console.WriteLine("Khong nhan duoc cau tra loi");

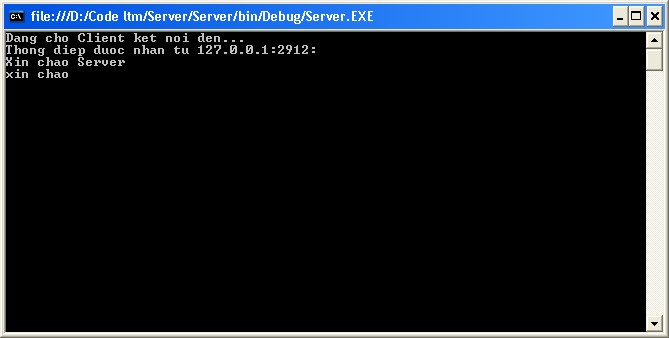
}

Console.WriteLine("Dang dong client"); server.Close();

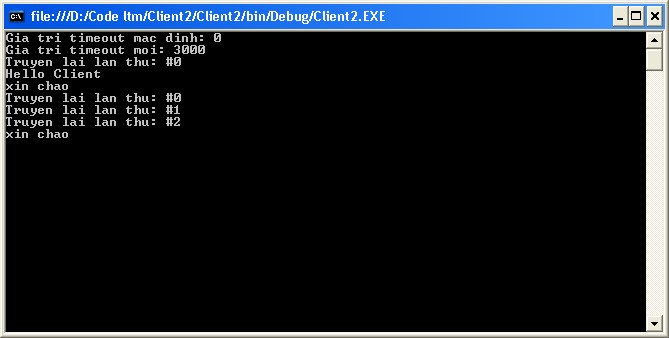
}

#### III.7.2. Test chương trình

Để test chương trình ta dùng hàm Sleep() trong vòng lặp While ở server để giả lập lỗi Chạy chương trình server:



Chạy chương trình client:



#### III.7.3. Trả lời câu hỏi:

3) Sử dụng phương pháp này có ngăn cản triệt để mất dữ liệu không? Nếu không hãy đề xuất phương pháp khác

**Chỉ giảm thiểu mất gói tin:** Việc thiết lập ReceiveTimeout và gửi lại dữ liệu giúp cải thiện khả năng phục hồi khi mất gói tin, nhưng không thể đảm bảo rằng tất cả dữ liệu sẽ được truyền thành công.

**Phương pháp thay thế và cải thiện**

* **Sử dụng Giao thức TCP**:
  + **TCP** là giao thức kết nối và cung cấp các cơ chế đảm bảo rằng dữ liệu được truyền đầy đủ và đúng thứ tự. TCP tự động xử lý việc gửi lại các gói tin mất và đảm bảo rằng dữ liệu không bị mất.
  + **Ưu điểm**: Đảm bảo tính toàn vẹn của dữ liệu, tự động gửi lại gói tin nếu cần thiết.