LAB 3

Tên :Trần Văn Nam

MSSV : 2115239

Lớp :CTK45A

Lab :3

1. [Mục đích 3](#_bookmark0)
2. [Yêu cầu 3](#_bookmark1)
3. [Hướng dẫn 3](#_bookmark2)
   1. [Xây dựng chương trình UDP Client – Server đơn giản 3](#_bookmark3)
      1. [Hướng dẫn lập trình UDP Server 3](#_bookmark4)
      2. [Hướng dẫn lập trình UDP Client 4](#_bookmark5)
      3. [Test chương trình: 4](#_bookmark6)
      4. [Trả lời câu hỏi: 5](#_bookmark7)
   2. [Bài tập 5](#_bookmark8)
   3. [Cải tiến chương trình UDP client-server để có thể gởi và nhận dữ liệu liên tục 6](#_bookmark9)
      1. [Hướng dẫn lập trình UDP Client 6](#_bookmark10)
      2. [Hướng dẫn lập trình UDP Server 6](#_bookmark11)
      3. [Test chương trình 6](#_bookmark12)
      4. [Trả lời câu hỏi: 7](#_bookmark13)
   4. [Sử dụng phương thức Connect ở client để thiết lập kết nối trước với server 8](#_bookmark14)
      1. [Hướng dẫn lập trình 8](#_bookmark15)
      2. [Test chương trình 8](#_bookmark16)
      3. [Trả lời câu hỏi: 9](#_bookmark17)
   5. [Kiểm tra khả năng phân biệt biên thông điệp của giao thức UDP 10](#_bookmark18)
      1. [Hướng dẫn lập trình UDP Client 10](#_bookmark19)
      2. [Hướng dẫn lập trình UDP Client 10](#_bookmark20)
      3. [Test chương trình 10](#_bookmark21)
      4. [Trả lời câu hỏi: 11](#_bookmark22)
   6. [Ngăn cản mất dữ liệu khi lập trình mạng sử dụng giao thức UDP 12](#_bookmark23)
      1. [Trả lời câu hỏi: 12](#_bookmark24)
      2. [Hướng dẫn lập trình để ngăn cản mất dữ liệu 12](#_bookmark25)
      3. [Test chương trình 13](#_bookmark26)
      4. [Trả lời câu hỏi: 14](#_bookmark27)
   7. [Ngăn cản mất gói tin khi lập trình mạng sử dụng giao thức UDP 14](#_bookmark28)
      1. [Hướng dẫn lập trình để ngăn cản mất gói tin 14](#_bookmark29)
      2. [Test chương trình 17](#_bookmark30)
      3. [Trả lời câu hỏi 18](#_bookmark31)

# 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

# Yêu cầu

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

# Hướng dẫn

## Xây dựng chương trình UDP Client – Server đơn giản

### 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);

### Hướng dẫn lập trình UDP Client

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

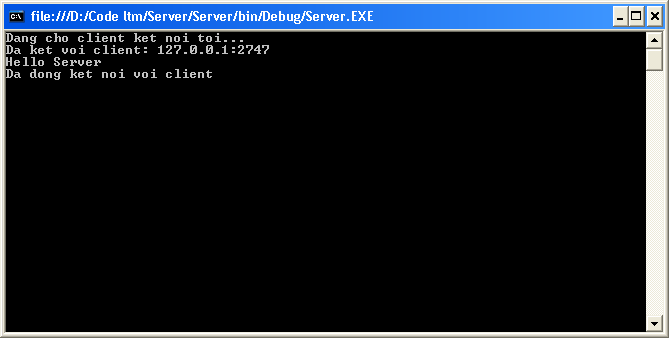
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

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

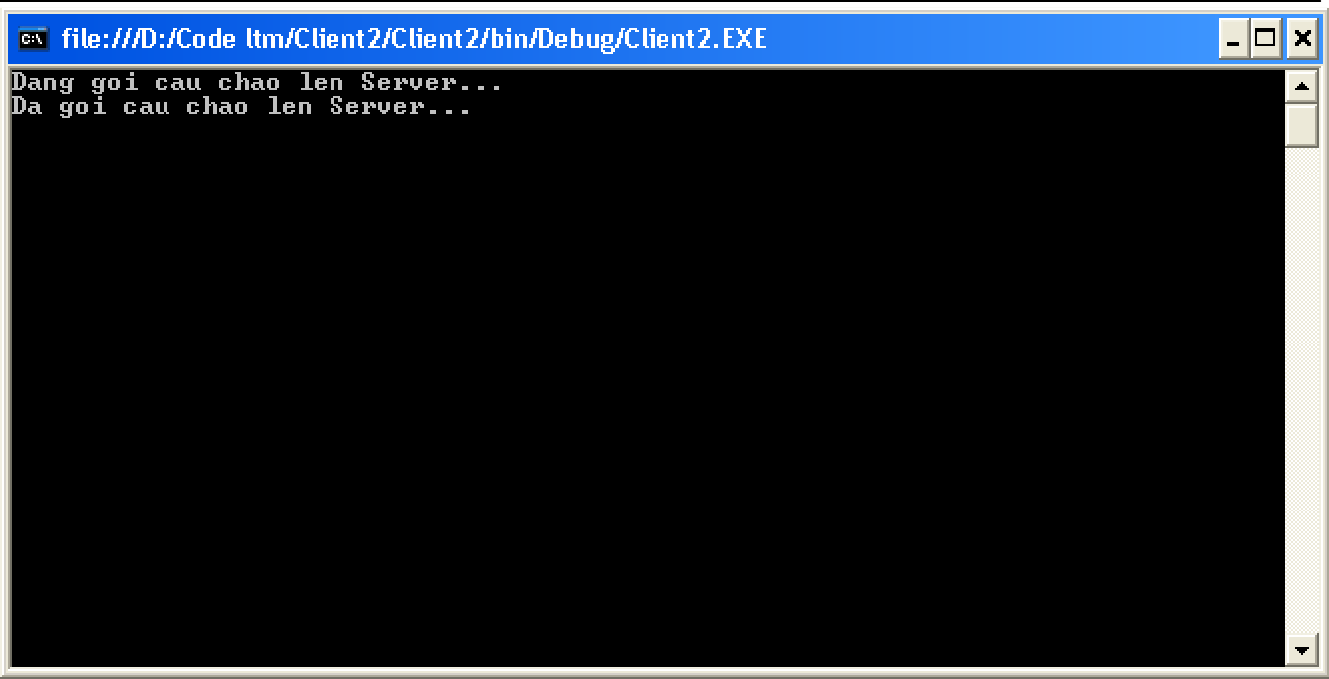
server.SendTo(buff, buff.Length, SocketFlags.None, serverEndPoint);

### Test chương trình:

Chạy chương trình Server:



Chạy chương trình ở client:



### Trả lời câu hỏi:

1. Như hình trên port 2747 ở đâu ra

* **Port 2747 được sinh ra ngẫu nhiên trong số 5000 port**

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

* **Không vì port xuất hiện ngẫu nhiên nên khi khởi động lại server có thể thông số port sẽ đổi qua một số mới.**

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

* **Vì giao thức UDP là giao thức phi kết nối, mà phi kết nối thì nó không thiết lập kết nối giữa server và client. Server không biết client đang kết nối thì lúc này client phải gửi câu chào khi nó nhận được gói tin UDP Client**

## 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.

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

### 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);

}

### 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);

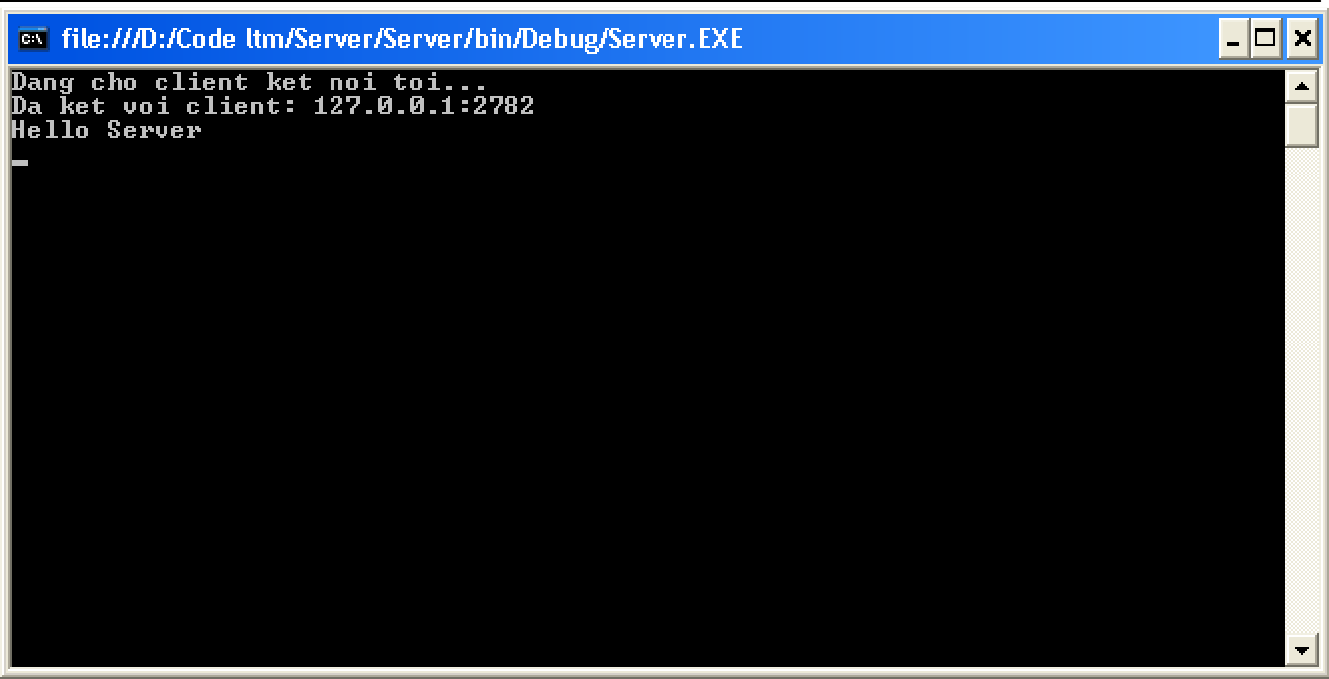
str = Encoding.ASCII.GetString(buff, 0, byteReceive); Console.WriteLine(str);

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

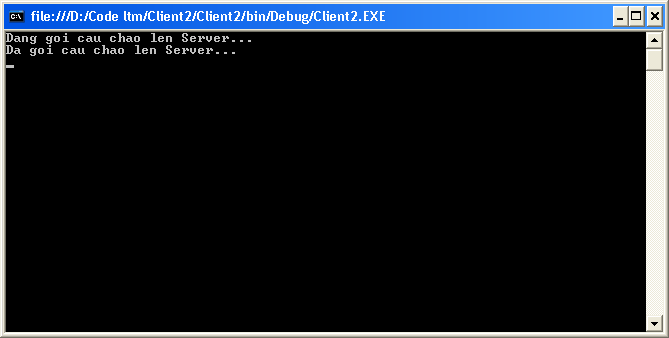
}

### 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

### 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 này là do khi client gửi gói tin đầu tiên yêu cầu nhập dữ liệu, nếu không nhập thì gói tin UDP đầu tiên này thì trên server không phản hồi lại client.**

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

* **Server chưa bật thì chương trình trên không bị lỗi là vì giao thức UDP là giao thức phi kết nối mà phi kết nối thì nó sẽ không quan tâm đến việc có thiệt lập kết nối tới client hay không nên không không xảy ra lỗi.**

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

**Giống như phía Server khi chạy chương trình mà tắt client thì chương trình không bị lỗi bên phía server sẽ không nhận được tín hiêu phía client và không gửi gói tin UDP phản hồi.**

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

### Hướng dẫn lập trình

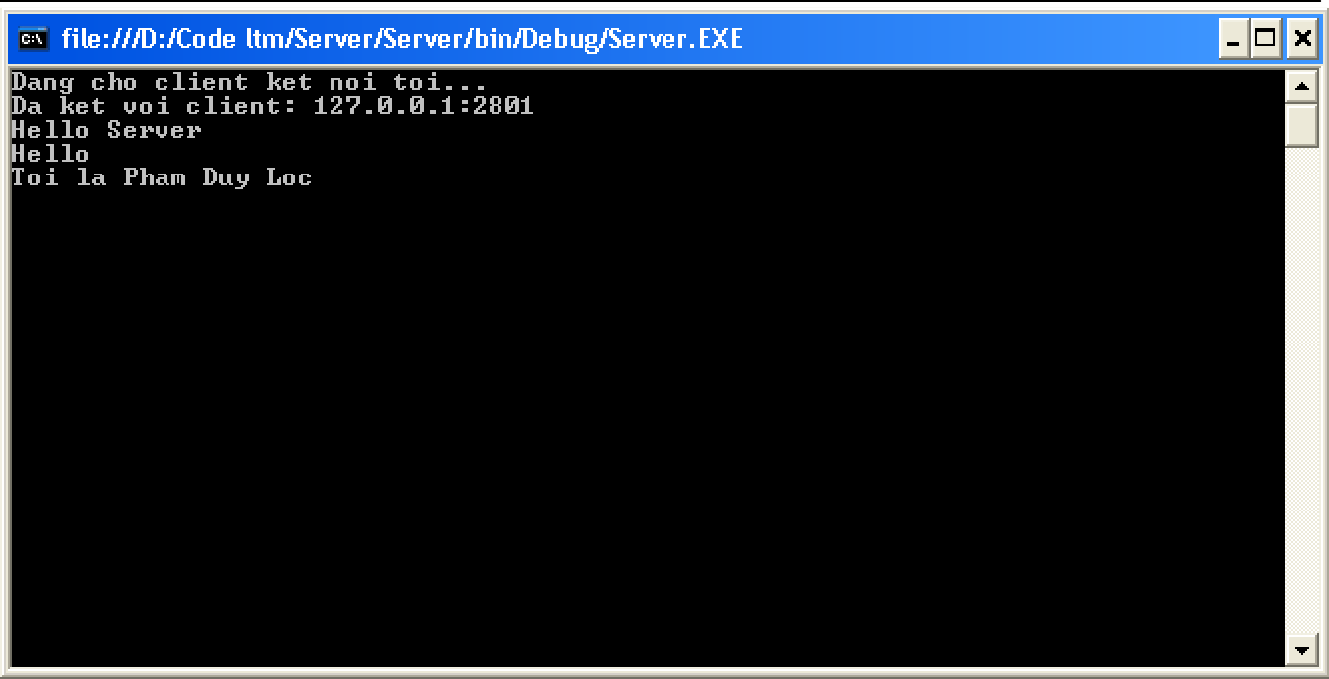
Việc này được thực hiện bằng cách sử dụng phương thức Connect() ở lớp socket

serverSocket.Connect(remote);

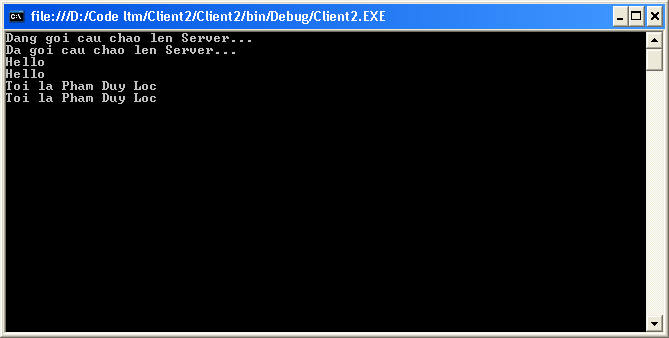
Sau khi gọi phương thức Connect() xong, ta có thể lập trình giống như lập trình với giao thức UDP với client

### Test chương trình

Chạy chương trình ở server:



Chạy chương trình ở client:



### 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?

* Client sẽ không nhận đươc phản hổi gói tin đầu tiên mà UDP Client đã gửi.

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

* Không xảy ra lỗi vì server của giao thức UDP phi kết nối, không thiết lập với Client nên không có xẩy ra xung đột qua lại.

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

* Giống như Server, Client của giao thức UDP cũng là phi kết nối nên nó chỉ gửi gói tin đầu tiên và chờ phản hồi lại vì vậy không có lỗi.

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

### 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);

}

* + 1. **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++)

{

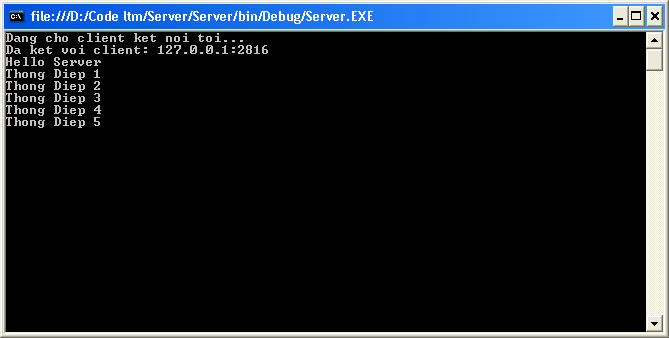
byteReceive = serverSocket.ReceiveFrom(buff, 0, buff.Length, SocketFlags.None, ref remote);

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

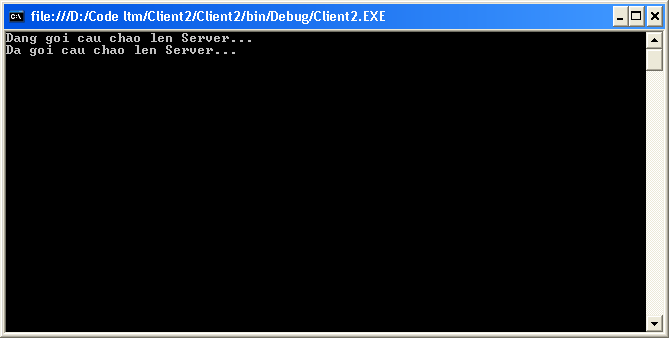
}

### Test chương trình

Chạy chương trình server:



Chạy chương trình client:



### Trả lời câu hỏi:

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

**Lý do tại sao các gói tin được phân biệt với nhau trong giao thức UDP:**

* **Không có đánh số thứ tự tại mức giao thức:** UDP không duy trì số thứ tự của các gói tin và không cố gắng sắp xếp chúng. Vì vậy, các gói tin có thể đến server hoặc client theo thứ tự khác nhau so với thứ tự ban đầu khi chúng được gửi.
* **Không có xác nhận nhận được:** UDP không yêu cầu xác nhận từ người nhận để xác định rằng gói tin đã đến đích. Điều này có nghĩa là nếu gói tin bị mất trong quá trình truyền tải, không có cách tự động để phát hiện hoặc sửa chữa việc này.
* **Không có cơ chế kiểm tra tính toàn vẹn dữ liệu:** UDP không cung cấp kiểm tra tính toàn vẹn dữ liệu tự động. Nếu dữ liệu trong gói tin bị thay đổi trong quá trình truyền tải, nó sẽ không được phát hiện và sửa chữa tự động.
  1. **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);

}

* + 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 dữ liệu gửi từ client có kích thước lớn hơn 10 byte, các byte thừa sẽ bị mất đi vì chúng không thể fit vào bộ đệm 10 byte. Điều này có thể dẫn đến việc mất dữ liệu hoặc đọc dữ liệu không hợp lệ từ bộ đệm. Cụ thể, lỗi này có thể làm mất tính toàn vẹn của dữ liệu và gây ra sự cố trong ứng dụng
  + 1. **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)

{

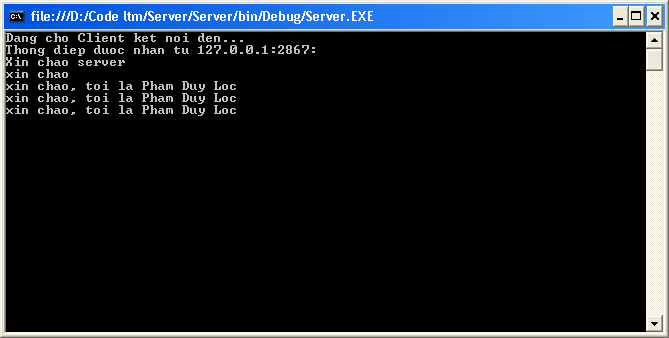
Console.WriteLine("Canh bao: du lieu bi mat, hay thu lai"); i += 10;

}

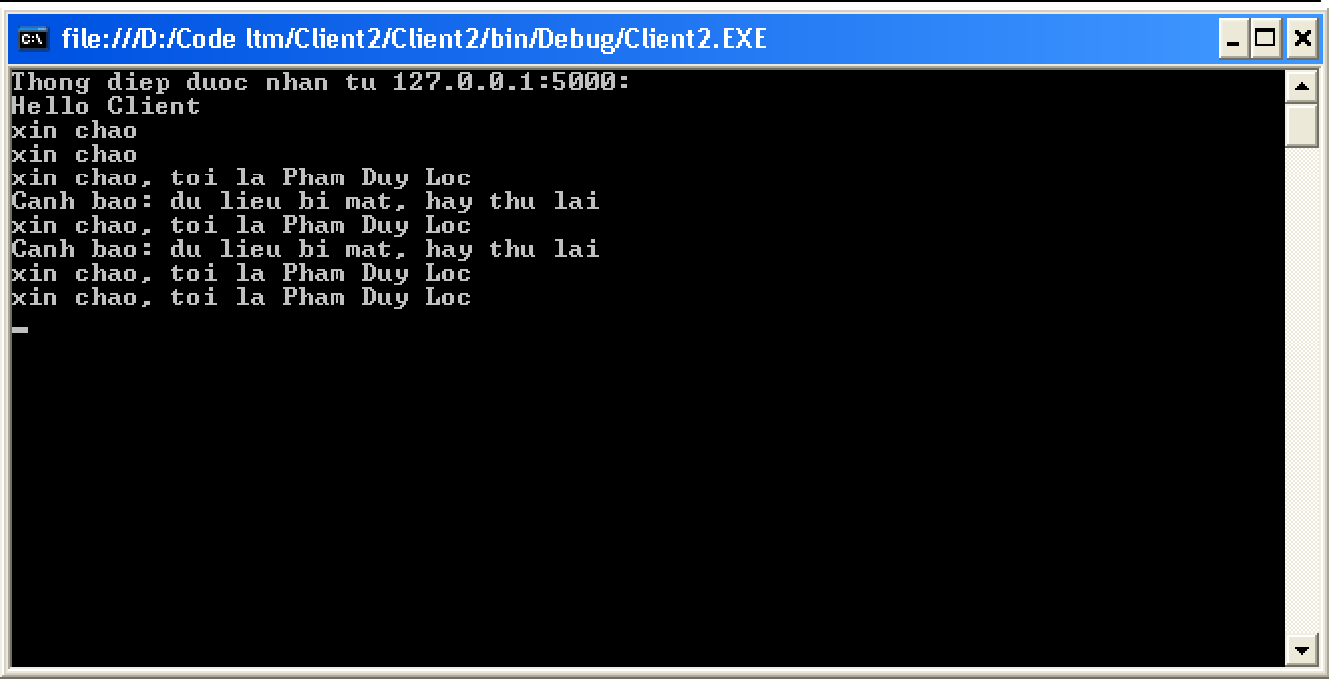
}

* + 1. **Test chương trình**

Chạy chương trình server:



Chạy chương trình client:



* + 1. **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?

.....................................................................................................................................

.....................................................................................................................................

* 1. **Ngăn cản mất gói tin khi lập trình mạng sử dụng giao thức UDP**
     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)

{

}

else

{

return recv;

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)

{

}

else

{

}

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

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)

{

}

else

}

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

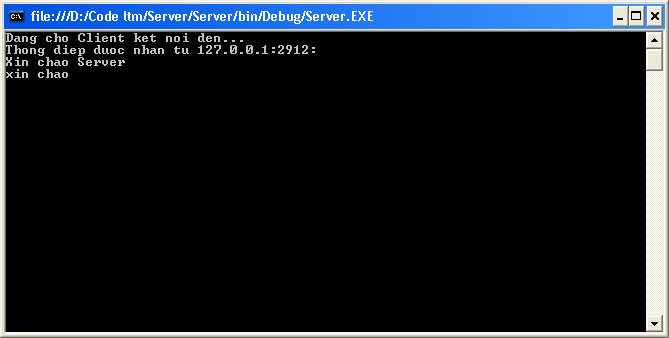
Console.WriteLine("Khong nhan duoc cau tra loi");

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

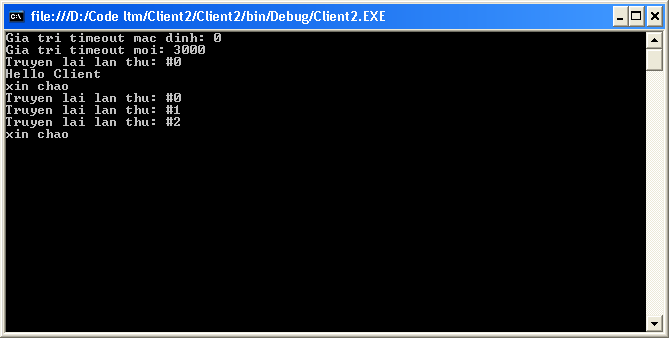
}

* + 1. **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:



* + 1. **Trả lời câu hỏi:**
       1. Sử dụng phương pháp này có ngăn cản triệt để mất dữ liệu không? Nếu không hãy đề xuất phương pháp khác
* Phương pháp này không ngăn cản triệt hạng mất dữ liệu hoàn toàn, nhưng nó giúp cải thiện khả năng đảm bảo tính toàn vẹn và đáng tin cậy của dữ liệu trong môi trường giao tiếp qua UDP (User Datagram Protocol).
* Trong UDP, không có đảm bảo về việc dữ liệu sẽ đến được đích hoặc không bị mất. Điều này có nghĩa là UDP không đảm bảo tính toàn vẹn dữ liệu và không đảm bảo dữ liệu sẽ không bị mất trong quá trình truyền tải. Tùy thuộc vào môi trường mạng và tình hình, dữ liệu UDP có thể bị mất, trùng lặp hoặc đến ngoài thứ tự.
* Phương pháp đã sử dụng, tức là việc gửi lại dữ liệu sau một số lần thất bại (retry) và sử dụng Timeout, có thể cải thiện đáng kể tính toàn vẹn của dữ liệu. Nhưng nó vẫn không đảm bảo mất dữ liệu tuyệt đối và có thể gây trễ trong trường hợp dữ liệu bị mất và cần gửi lại.
* UDP là một giao thức không đáng tin cậy mặc dù nhanh chóng, nhưng ngoài phương pháp timeouts có thể thực hiện các biện pháp sau để cải thiện độ tin cậy của nó:
* **Checksums**: Sử dụng kiểm tra tổng (checksum) để kiểm tra tính toàn vẹn của dữ liệu. Bạn có thể tính toán checksum cho mỗi gói tin trước khi gửi và kiểm tra checksum sau khi nhận. Nếu checksum không khớp, bạn có thể yêu cầu gửi lại gói tin.
* **ACK (Acknowledgment)**: Bạn có thể thực hiện một cơ chế đơn giản để xác nhận nhận được dữ liệu. Sau khi gửi một gói tin, đợi cho đến khi bạn nhận được một xác nhận từ bên nhận trước khi gửi gói tiếp theo. Nếu không nhận được xác nhận sau một khoảng thời gian, bạn có thể gửi lại gói tin đó.
* **Retry Mechanism**: Như bạn đã thực hiện trong mã của bạn, bạn có thể xây dựng một cơ chế retry. Khi bạn gửi một gói tin, đợi một khoảng thời gian để nhận phản hồi. Nếu không nhận được phản hồi, hãy gửi lại gói tin. Lặp lại quá trình này một số lần trước khi xem xét gói tin bị mất.
* **Sequence Number**: Gắn số thứ tự cho từng gói tin bạn gửi đi. Khi bên nhận nhận được các gói tin, họ kiểm tra số thứ tự để đảm bảo dữ liệu đến theo đúng thứ tự. Nếu một gói tin bị mất, họ có thể yêu cầu bạn gửi lại gói đó.