



# **Lập trình Socket và UDP,TCP**

**Biên tập bởi:**

Khoa CNTT ĐHSP KT Hưng Yên

# Lập trình Socket và UDP,TCP

**Biên tập bởi:**

Khoa CNTT ĐHSP KT Hưng Yên

**Các tác giả:**

Khoa CNTT ĐHSP KT Hưng Yên

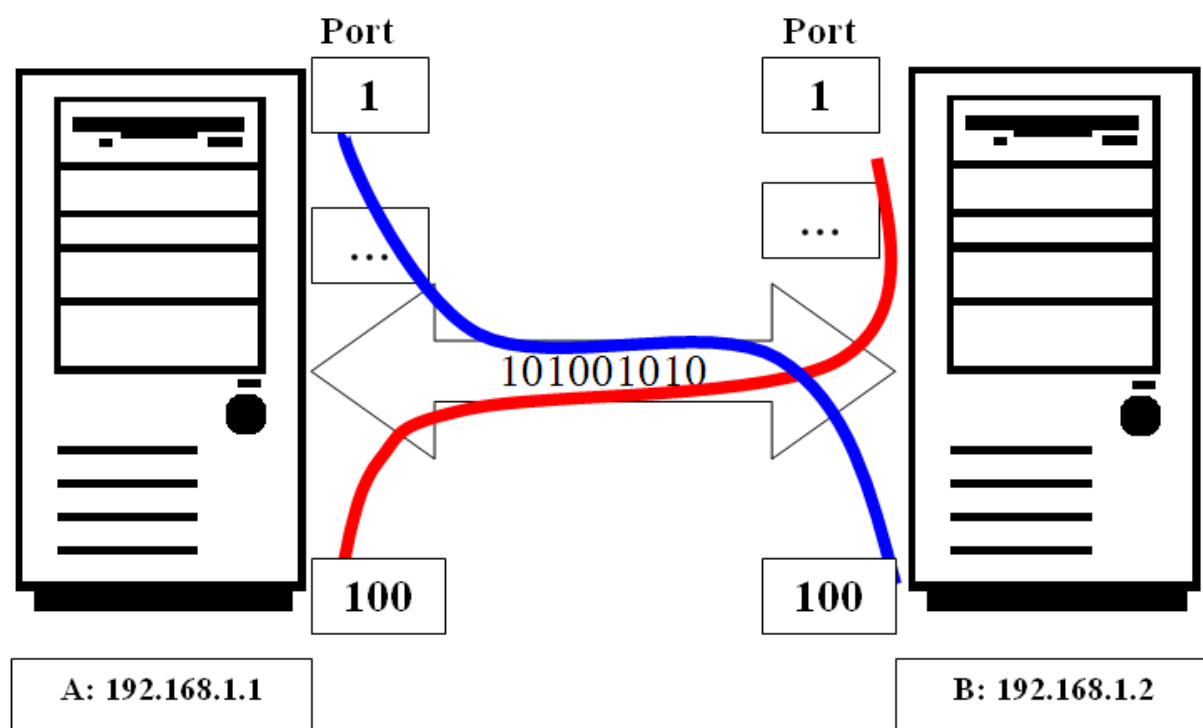
Phiên bản trực tuyến:

<http://voer.edu.vn/c/a516d57c>

# MỤC LỤC

1. Khái niệm Địa chỉ và cổng (Address & Port)
  2. Lớp IPAddress
  3. Lớp IPEndPoint
  4. Lớp IPHostEntry
  5. Lớp DNS
  6. Lớp UDP
  7. Lớp TCP
  8. Lớp TCPListener
  9. Bài tập
- Tham gia đóng góp

# Khái niệm Địa chỉ và cổng (Address & Port)



*Nguyên lý:*

+ Trong một máy có rất nhiều ứng dụng muốn trao đổi với các ứng dụng khác thông qua mạng. (ví dụ trên có 2 ứng dụng trong máy A muốn trao đổi với 2 ứng dụng trên máy B)

+ Mỗi máy tính chỉ có duy nhất một đường truyền dữ liệu (để gửi và nhận)

*Vấn đề :*

Rất có thể xảy ra "nhầm lẫn" khi dữ liệu từ máy A gửi đến máy B thì không biết là dữ liệu đó gửi cho ứng dụng nào trên máy B?

*Giải quyết:*

Mỗi ứng dụng trên máy B sẽ được gán một số hiệu (mà ta vẫn quen gọi là cổng : Port), số hiệu cổng này từ 1..65535. Khi ứng dụng trên máy A muốn gửi cho ứng dụng nào trên máy B thì chỉ việc điền thêm số hiệu cổng (vào trường RemotePort) vào gói tin cần gửi. Trên máy B, Các ứng dụng chỉ việc kiểm tra giá trị Cổng trên mỗi gói tin xem có

trùng với số hiệu Cổng của mình (đã được gán – chính là giá trị Localport) hay không ? Nếu bằng thì xử lý, còn trái lại thì không làm gì (vì không phải là của mình).

*Như vậy: Khi cần trao đổi dữ liệu cho nhau thì hai ứng dụng cần phải biết thông tin tối thiểu là Địa chỉ (Address) và số hiệu cổng (Port) của ứng dụng kia.*

+ Hai ứng dụng có thể cùng nằm trên một máy

+ Hai ứng dụng trên cùng một máy không được trùng số hiệu cổng.

+ LocalHost : (Địa chỉ máy hiện đang chạy ứng dụng):, Với B: LocalHost = 192.168.1.2, với A thì Localhost = 192.168.1.1;

+ RemoteHost (Địa chỉ của máy chạy ứng dụng đang tham gia trao đổi thông tin với ứng dụng hiện tại). RemoteHost của ứng dụng chạy trên máy A là : 192.168.1.2; RemoteHost của ứng dụng chạy trên máy B là : 192.168.1.1;

+ LocalPort: LocalPort của ứng dụng chạy trên máy A (FTP) là 100, của ứng dụng chạy trên máy B (FTP) là 5;

+ RemotePort: RemotePort của ứng dụng chạy trên máy A (FTP) là 5, của ứng dụng chạy trên máy B (FTP) là 100;

+ Hai ứng dụng đặt trên hay máy khác nhau thì LocalPort có thể giống nhau (Nhưng nếu đặt trên một máy thì không được trùng nhau)

# Lớp IPAddress

## Giới thiệu

Trên Internet mỗi một trạm (có thể là máy tính, máy in, thiết bị ...) đều có một định danh duy nhất, định danh đó thường được gọi là một địa chỉ (Address). Địa chỉ trên Internet là một tập hợp gồm 4 con số có giá trị từ 0-255 và cách nhau bởi dấu chấm.

Để thể hiện địa chỉ này, người ta có thể viết dưới các dạng sau:

- Tên : ví dụ May01, Server, ....
- Địa chỉ IP nhưng đặt trong một chuỗi: ", "127.0.0.1"
- Đặt trong một mảng 4 byte, mỗi byte chứa một số từ 0-255. Ví dụ để biểu diễn địa chỉ 192.168.1.1 ta có thể viết:

Dim DiaChi(3) as Byte"192.168.1.1

DiaChi(0) = 192

DiaChi(1) = 168

DiaChi(2) = 1

DiaChi(3) = 1

- Hoặc cũng có thể là một số (long), có độ dài 4 byte. Ví dụ, với địa chỉ 192.168.1.1 ở trên thì giá trị đó sẽ là: 16885952 (đây là số ở hệ thập phân khi xếp liền 4 byte ở trên lại với nhau **00000001 00000001 10101000 11000000**)













1 (Byte 0) 1 168 192 (Byte 3)

? Như vậy, để đổi một địa chỉ chuẩn ra dạng số ta chỉ việc tính toán cho từng thành phần. Ví dụ: Đổi địa chỉ 192.168.1.2 ra số, ta tính như sau :

$$2 * 256 ^ 3 + 1 * 256 ^ 2 + 168 * 256 ^ 1 + 192 * 256 ^ 0$$

Trong MS.NET, IPAddress là một lớp dùng để mô tả địa chỉ này. Đây là lớp rất cơ bản được sử dụng khi chúng ta thao tác (truyền) vào các lớp như IPEndpoint, UDP, TCP, Socket ...

## Các thành viên của lớp

Phạm vi	Name	Description
	<a href="#">Any</a>	Cung cấp một địa chỉ IP (thường là 0.0.0.0) để chỉ ra rằng Server phải lắng nghe các hoạt động của Client trên tất cả các Card mạng (sử dụng khi xây dựng Server). Thuộc tính này chỉ đọc.
	<a href="#">Broadcast</a>	Cung cấp một địa chỉ IP quảng bá (Broadcast, thường là 255.255.255.255), ở dạng số Long. Muốn lấy ở dạng xâu, viết: <a href="#">Broadcast.ToString()</a> . This field is read-only.
	<a href="#">Loopback</a>	Trả về một địa chỉ IP lặp (IP Loopback, ví dụ 127.0.0.1). This field is read-only.
	<a href="#">Address</a>	Một địa chỉ IP (An Internet Protocol (IP) address) ở dạng số Long. (Muốn chuyển sang dạng dấu chấm, viết: <a href="#">Address.ToString()</a> ). (Không còn sử dụng trong phiên bản mới !!!!!)
	<a href="#">AddressFamily</a>	Trả về họ địa chỉ của địa chỉ IP hiện hành. Nếu địa chỉ ở dạng IPv4 thì kết quả là Internetwork, và InternetworkV6 nếu là địa chỉ IPv6.
P/Vi	Method Name	Description
k/tạo	Constructor	<ul style="list-style-type: none"> <li>- <a href="#">IPAddress(Số_Long) As IPAddress</a> → Tạo địa chỉ IP từ một số long</li> <li>- <a href="#">IPAddress(Mảng_Byte)</a> → Tạo địa chỉ IP từ một mảng byte (4 byte).</li> </ul>
	<a href="#">GetAddressBytes</a> as bytes()	Chuyển địa chỉ thành mảng byte (4 byte).
	<a href="#">HostToNetworkOrder</a>	Đảo thứ tự byte của một số cho đúng với thứ tự byte trong địa chỉ IPAddress.
	<a href="#">IsLoopback</a>	Cho biết địa chỉ có phải là địa chỉ lặp hay không?
	<a href="#">NetworkToHostOrder</a>	Đảo thứ tự byte của một địa chỉ cho đúng với thứ tự byte thông thường.
	<a href="#">Parse</a>	Chuyển một địa chỉ IP ở dạng <u>xâu</u> thành một địa chỉ <u>IP chuẩn</u> (Một đối tượng IPAddress)
	<a href="#">ToString</a> as String	Trả về địa chỉ IP (một xâu) nhưng ở dạng ký pháp có dấu chấm. (Ví dụ "192.168.1.1").
	<a href="#">TryParse</a> (Địa_ChỉIP: String)	Kiểm tra xem một địa chỉ IP (ở dạng xâu) có phải đúng là địa chỉ IP hợp lệ hay không ? True = đúng

## Ví dụ

1. Tạo một địa chỉ IP (Tạo một đối tượng IPAddress) có giá trị là 16885952

00000001 00000001 10101000 11000000

1. Tạo một địa chỉ IP từ một mảng byte tương ứng với địa chỉ 192.168.10.10
2. Tạo một địa chỉ IP từ một chuỗi.
3. Tạo một địa chỉ 192.168.1.2

```
Imports System.Net
Public Class Form1
    Private Sub TaoDiaChi()
        Dim b(3) As Byte
        b(0) = 192
        b(1) = 168
        b(2) = 10
        b(3) = 10

        '///Tạo địa chỉ từ các hàm khởi tạo
        Dim Ip1 As New IPAddress(b) '//Tạo địa chỉ từ mảng byte ở trên

        Dim Ip2 As New IPAddress(16885952)
        Dim Ip3 As IPAddress=IPAddress.Parse("172.16.1.1")

        MsgBox(Ip1.ToString)
        MsgBox(Ip2.ToString)
        MsgBox(Ip3.ToString)

        '///Tạo địa chỉ thông qua việc tính toán.
        Dim So As Long = 192* 256^0+168* 256^1+1* 256^2 + 2*256^3
        Dim Ip4 As New IPAddress(So)
        MsgBox Ip4.ToString()
    End Sub
End Class
```

1. Kiểm tra xem 192.168.1.300 có phải là địa chỉ IP hợp lệ không ?

```
Private Sub KiemTra()
    Dim Ip4 As string = "127.0.0.1"
    Dim Ip5 As String = "999.0.0.1"

    MsgBox(IPAddress.TryParse(Ip4, New IPAddress(1)))
    MsgBox(IPAddress.TryParse(Ip5, New IPAddress(1)))
End Sub
```

\*\*\* Lưu ý: Tham số thứ hai là một đối tượng bất kỳ thuộc kiểu IPAddress, do vậy bạn có thể viết New IPAddress(0), IPAddress(1),...

1. Chuyển địa chỉ hiện hành ra mảng byte và hiển thị từng thành phần trong mảng đó



```
Sub ChuyenDoi()  
    Dim Ip3 As New IPAddress(16885952)  
    Dim b() As Byte  
  
    b = Ip3.GetAddressBytes()  
    MsgBox("Address: " & b(0) & "." & b(1) & "." & b(2) & "." & b(3))  
End Sub
```






# Lớp IPEndpoint

## Giới thiệu

Trong mạng, để hai trạm có thể trao đổi thông tin được với nhau thì chúng cần phải biết được địa chỉ (IP) của nhau và số hiệu cổng mà hai bên dùng để trao đổi thông tin. Lớp IPAddress mới chỉ cung cấp cho ta một về là địa chỉ IP (IPAddress), còn thiếu về thứ hai là số hiệu cổng (Port number). Như vậy, lớp IPEndpoint chính là lớp chứa đựng cả IPAddress và Port number.

Đối tượng IPEndpoint sẽ được dùng sau này để truyền trực tiếp cho các đối tượng UDP, TCP...

## Các thành viên của lớp

Hàm khởi tạo		
<a href="#">IPEndPoint(Int64, Int32)</a>		Tạo một đối tượng mới của lớp <b>IPEndPoint</b> , tham số truyền vào là địa <b>chỉ IP</b> (ở dạng số) và <b>cổng</b> sẽ dùng để giao tiếp.
<a href="#">IPEndPoint(IPAddress, Int32)</a>		Tạo một đối tượng mới của lớp <b>IPEndPoint</b> , Tham số truyền vào là một địa chỉ <b>IP</b> và số hiệu <b>cổng</b> dùng để giao tiếp. (Tham khảo cách tạo IPAddress ở phần trên)
P/Vi	Thuộc tính	Description
	<a href="#">Address</a>	Trả về hoặc thiết lập địa chỉ IP cho endpoint. (Trả về một đối tượng IPAddress)
	<a href="#">AddressFamily</a>	Lấy về loại giao thức mà Endpoint này đang sử dụng.
	<a href="#">Port</a>	Gets or sets số hiệu cổng của endpoint.
P/Vi	Phương thức	Description
	<a href="#">Create</a>	Tạo một endpoint từ một địa chỉ socket (socket address).
	<a href="#">ToString</a>	Trả về địa chỉ IP và số hiệu cổng theo khuôn dạng ĐịaChỉ: Cổng, ví dụ: "192.168.1.1:8080"

## Ví dụ

Tạo một đối tượng *IPEndpoint* có địa chỉ là "127.0.0.1", cổng là 1000

Để tạo một *IPEndpoint*, ta có thể dùng 2 hàm thiết lập, trong đó có một hàm thiết lập đòi hỏi phải truyền một đối tượng *IPAddress* vào. Khi đó chúng ta cần phải tạo đối tượng *IPAddress* trước theo các cách như đã đề cập trong phần 1.

```
Private Sub TaoEndpoint()  
    '/// Tạo một địa chỉ IP  
    Dim IPAdd As IPAddress = IPAddress.Parse("127.0.0.1")  
  
    '/// Truyền vào cho hàm khởi tạo để tạo IPEndpoint  
    Dim IPep As New IPEndPoint(IPAdd, 1000)  
  
    MsgBox(IPep.ToString)  
End Sub
```

Tạo một *EndPoint* từ tên máy: Ta cũng có thể tạo đối tượng *IPAddress* từ tên của máy thông qua **phương thức tĩnh** *DNS.GetHostAddresses* của lớp *DNS*. Sau đó truyền đối tượng *IP* này vào cho phương thức khởi tạo của *IPEndPoint* để tạo đối tượng *IPEndpoint* mới.

```
Private Sub TaoEndPointBoiTenMay()  
  
    Dim IPAdd As IPAddress  
    IPAdd = Dns.GetHostAddresses("localhost")(0)  
  
    Dim IPep As New IPEndPoint(IPAdd, 1000)  
    MsgBox(IPep.ToString)  
End Sub
```

\*\*\* Lưu ý : Vì một máy tính có thể có nhiều Card mạng (Interface) do vậy có thể có nhiều hơn 1 địa chỉ IP. Hàm *GetHostAddresses* sẽ trả về cho ta một **mảng chứa tất cả các địa chỉ** đó. Ta truyền giá trị 0 để lấy địa chỉ của Card mạng đầu tiên.

# Lớp IPHostEntry

## Giới thiệu




IPHostEntry là lớp chứa (Container) về thông tin địa chỉ của các máy trạm trên Internet.

Lưu ý: Nó chỉ là nơi để "chứa" , do vậy trước khi sử dụng cần phải "Nạp" thông tin vào cho nó.

Lớp này rất hay được dùng với lớp DNS

## Các thành viên của lớp

### Public Properties

	Name	Description
	<a href="#">AddressList</a>	Gets or sets a list of IP addresses that are associated with a host.
	<a href="#">Aliases</a>	Gets or sets a list of aliases that are associated with a host.
	<a href="#">HostName</a>	Gets or sets the DNS name of the host.

□

# Lớp DNS

## Giới thiệu

DNS (Domain Name Service) là một lớp giúp chúng ta trong việc phân giải tên miền (Domain Resolution) đơn giản. (Phân giải tên miền tức là : Đầu vào là Tên của máy trạm, ví dụ ServerCNTT thì đầu ra sẽ cho ta địa chỉ IP tương ứng của máy đó, ví dụ 192.168.3.8)

Ngoài ra lớp Dns còn có rất nhiều phương thức cho ta thêm thông tin về máy cục bộ như tên, địa chỉ v.v...

## Các thành viên của lớp

	Name	Description
	<del><code>GetHostByAddress (IP As String)</code> <code>GetHostByAddress (IP As IPAddress) As IPHostEntry</code></del>	Trả về thông tin (IPHostEntry) của trạm có địa chỉ IP được truyền vào.  → Thay bằng <b>GetHostEntry()</b>
	<del><code>GetHostByName (Tên trạm: String) As IPHostEntry</code></del>	Trả về thông tin (IPHostEntry) DNS của một trạm. → Đã bị loại bỏ. Thay bằng <b>GetHostEntry()</b>
	Thuộc tính <u>HostName</u>	Cho ta biết tên của máy vừa được phân giải. Nếu không phân giải được thì có giá trị là địa chỉ IP.
	<u><code>GetHostAddresses</code></u> (IP_Or_HostName: String) as IPAddress()	Trả về tất cả các địa chỉ IP của một trạm.
	<u><code>GetHostEntry</code></u> (IP_Or_HostName As String) as IPHostEntry  <u><code>GetHostEntry</code></u> (IP As IPAddress)	Giải đáp tên hoặc địa chỉ IP truyền vào và trả về một đối tượng <b>IPHostEntry</b> tương ứng.
	<u><code>GetHostName</code></u> As String	Lấy về tên của máy tính cục bộ.
	<u><code>Resolve</code></u> (Hostname: String)	Chuyển tên của máy hoặc địa chỉ IP thành <b>IPHostEntry</b> tương ứng. → Đã bị bỏ !, Thay bằng <b>GetHostEntry()</b>

\*\*\* Lưu ý: Đây là các **phương thức tĩnh**, do vậy khi gọi thì gọi trực tiếp từ tên lớp mà không cần phải khai báo một đối tượng mới của lớp này. Ví dụ ta gọi: DNS.Resolve, Dns.GetHostname, Dns.GetHostEntry v.v...

## Ví dụ

1. Hiện thị tên của máy tính hiện tại

```
MsgBox(Dns.GetHostName())
```

1. Hiện thị tất cả địa chỉ IP của một máy nào đó.

```

Private Sub ShowIPs()
    Dim ip As IPAddress
    Dim add() As IPAddress
    Dim i As Integer

    '/// Lấy tất cả địa chỉ IP của máy Notebook. (Một máy có thể có nhiều IP)
    add = Dns.GetHostAddresses("notebook")

    '/// Duyệt sử dụng For Each (tập hợp)
    For Each ip In add
        MsgBox(ip.ToString)
    Next

    '/// Or Duyệt theo kiểu mảng
    For i = 0 To add.Length - 1
        MsgBox(add(i).ToString)
    Next
End Sub

```

1. Tạo một IPHostEntry từ máy có tên là "Notebook"
2. Tạo một IPHostEntry từ địa chỉ "127.0.0.1"
3. Tạo một IPHostEntry từ một đối tượng IPAddress, có địa chỉ IP là 127.0.0.1

```

Private Sub CreatIPHostEntry()
    Dim iphe1, iphe2, iphe3 As IPHostEntry
    Dim ipadd As IPAddress = IPAddress.Parse("127.0.0.1")

    iphe1 = Dns.GetHostEntry("Notebook ")
    iphe2 = Dns.GetHostEntry("127.0.0.1")
    iphe3 = Dns.GetHostEntry(ipadd)

    MsgBox(iphe1.HostName)           '/// Notebook (tùy vào máy)
    MsgBox(iphe2.HostName)           '/// Notebook
    MsgBox(iphe3.HostName)           '/// Notebook
End Sub

```

\*\*\* Lưu ý: Đối tượng IPHostEntry chúng ta tạo ở trên sẽ được dùng rất nhiều trong các phần sau của bài giảng này.

# Lớp UDP

## Giới thiệu

Giao thức UDP (User Datagram Protocol hay User Define Protocol) là một giao thức phi kết nối (Connectionless) có nghĩa là một bên có thể gửi dữ liệu cho bên kia mà không cần biết là bên đó đã sẵn sàng hay chưa ? (Nói cách khác là không cần thiết lập kết nối giữa hai bên khi tiến hành trao đổi thông tin). Giao thức này không tin cậy bằng giao thức TCP nhưng tốc độ lại nhanh và dễ cài đặt. Ngoài ra, với giao thức UDP ta còn có thể gửi các gói tin quảng bá (Broadcast) cho đồng thời nhiều máy.

Trong .NET, lớp **UDPClient** (nằm trong System.Net.Sockets) đóng gói các chức năng của giao thức UDP.



## Các thành viên của lớp UDPClient

Constructor methods		Description
<a href="#">UdpClient ()</a>		Tạo một đối tượng (thể hiện) mới của lớp UDPClient.
<a href="#">UdpClient (AddressFamily)</a>		Tạo một đối tượng (thể hiện) mới của lớp UDPClient. Thuộc một dòng địa chỉ (AddressFamily) được chỉ định.
<a href="#">UdpClient (LocalPort: Int32)</a>		Tạo một <b>UdpClient</b> và gắn (bind) một cổng cho nó.
<a href="#">UdpClient (IPEndPoint)</a>		Tạo một <b>UdpClient</b> và gắn (bind) một IPEndPoint (gán địa chỉ IP và cổng) cho nó.
<a href="#">UdpClient (Int32, AddressFamily)</a>		Tạo một <b>UdpClient</b> và gán số hiệu cổng, AddressFamily
<a href="#">UdpClient (RemoteHost: String, Int32)</a>		Tạo một <b>UdpClient</b> và thiết lập với một trạm từ xa mặc định.
PUBLIC Method		
	Name	Description
💜	<a href="#">BeginReceive</a>	Nhận dữ liệu Không đồng bộ từ máy ở xa
💜	<a href="#">BeginSend</a>	Gửi không đồng bộ dữ liệu tới máy ở xa
💜📱	<a href="#">Close</a>	Đóng kết nối.
💜📱	<a href="#">Connect</a>	Thiết lập một Default remote host.
💜	<a href="#">EndReceive</a>	Kết thúc nhận dữ liệu không đồng bộ ở trên
💜	<a href="#">EndSend</a>	Kết thúc việc gửi dữ liệu không đồng bộ ở trên
💜📱	<a href="#">Receive (EndPoint của máy ở xa) As Byte()</a>	Nhận dữ liệu (đồng bộ) do máy ở xa gửi. (Đồng bộ có nghĩa là các lệnh ngay sau lệnh Receive chỉ được thực thi nếu Receive đã nhận được dữ liệu về. Còn nếu nó chưa nhận được — dù chỉ một chút — thì nó vẫn cứ chờ (blocking))
💜📱	<a href="#">Send</a>	Gửi dữ liệu (đồng bộ) cho máy ở xa.

- đồng bộ : Synchronous
- Không đồng bộ : Asynchronous

## Ví dụ

Chuyển đổi một chuỗi ký tự sang mảng byte:

```
Dim Msg() As Byte
Msg = System.Text.Encoding.UTF8.GetBytes("Xin chào !")
```

Chuyển đổi mảng byte sang chuỗi ký tự:

```
S = System.Text.Encoding.UTF8.GetString(Msg)
```

Ví dụ tổng hợp 2 hàm chuyển đổi trên:

```
Private Sub ConvertingDemo()
    Dim Msg() As Byte           ''' Hỏi thêm: Tại sao không có New !?'''
    Msg = System.Text.Encoding.UTF8.GetBytes("Xin Chào !")

    Dim S As String
    S = System.Text.Encoding.UTF8.GetString(Msg)
    MsgBox("Giá trị của mảng byte Msg là : " & S)
End Sub
```

1. Tạo một UDPClient gắn vào cổng 10 và Gửi một gói tin "Hello" tới một ứng dụng UDP khác đang chạy trên máy có địa chỉ là "127.0.0.1" và cổng 1000.

Ung dụng A:

```

Imports System.Net
Imports System.Net.Sockets

Public Class Form1
    Const LOCAL_PORT = 10
    Const REMOTE_PORT = 1000

    '/// Tạo một UDP và gắn (Bind) vào cổng 10
    Dim UngDung1 As New UdpClient(LOCAL_PORT)

    Private Sub Gửi_Dữ_Liệu()
        Dim Msg() As Byte

        '/// Chuyển chuỗi "Hello there !" thành mảng byte để gửi đi
        Msg = System.Text.Encoding.UTF8.GetBytes("Hello there !")

        '/// Gửi vào cổng 1000 của máy 127.0.0.1
        UngDung1.Send(Msg, Msg.Length, "127.0.0.1", REMOTE_PORT)
    End Sub
End Class

```

1. Tạo một UdpClient gắn vào cổng 1000 và nhận dữ liệu từ ứng dụng khác gửi đến.

```

Imports System.Net
Imports System.Net.Sockets

Public Class Form1
    Const LOCAL_PORT = 1000
    Const REMOTE_PORT = 10

    Dim UngDung2 As New UdpClient(LOCAL_PORT)

    Private Sub Nhận_Dữ_Liệu()
        Dim Msg() As Byte

        ''' Vì phương thức Receive yêu cầu phải cho biết là nhận từ máy nào (mà đại
        ''' diện cho một máy là một IPEndPoint) nên trước tiên ta cần phải tạo một
        ''' IPEndPoint. ở đây ta muốn lấy về từ máy 127.0.0.1 và RemotePort là 100
        Dim ep As New IPEndPoint(IPAddress.Parse("127.0.0.1"), 100)

        Msg = UngDung2.Receive(ep)

        Dim S As String
        S = System.Text.Encoding.UTF8.GetString(Msg)    '''Chuyển byte -> String

        MsgBox(S)
    End Sub
End Class

```

# 1. Viết chương trình tổng hợp CHAT giữa hai máy dùng giao thức UDP

Mô tả giao diện:

**Simple CHAT - UTEHY 2006 (Ứng dụng 1 - Máy A)**

Gõ tin cần gửi đi và nhấn Enter:

Xin chào B

Cổng để truyền thông của A: Ứng dụng 1

Thông số của máy B: Ứng dụng 2

Địa chỉ

Cổng (Port)

Thông tin nhận về

Cảm ơn. Tôi là B đây ! (Người gửi : 127.0.0.1)

Giao diện của ứng dụng A (Ứng dụng 1)

**Simple CHAT - UTEHY 2006 (Ứng dụng 2 - Máy B)**

Gõ tin cần gửi đi và nhấn Enter:

Cảm ơn. Tôi là B đây !

Cổng để truyền thông của A: Ứng dụng 1

Thông số của máy B: Ứng dụng 2

Địa chỉ

Cổng (Port)

Thông tin nhận về

Xin chào B (Người gửi : 127.0.0.1)

Giao diện của ứng dụng b (Ứng dụng 2)

Code cho mỗi ứng dụng là hoàn toàn giống nhau.

**Lưu ý:** Nếu 2 ứng dụng đặt trên 2 máy khác nhau thì chúng ta có thể đặt Remote Port và Local Port của hai ứng dụng giống nhau (Vỡ khựng bị xung đột).

Listing 1: Chương trình CHAT giữa hai ứng dụng

```
Imports System.Net
Imports System.Net.Sockets
Imports System.Threading

Public Class frmCHAT
    Const LOCAL_PORT = 10
    Const REMOTE_PORT = 1000

    '/// Tạo một UDP và gắn vào cổng 10
    Dim UngDung1 As UdpClient

    Dim Th As Thread
    Dim Thoat As Boolean = False

    Public Event Dữ_Liệu_Về (ByVal Data As String, ByVal RemoteHost As String)

    Private Sub Tham_Dò()
        '/// Tạo một địa chỉ IP (ở đây dùng GetHostEntry để ta nhập địa chỉ or Hostname)
        Dim Ip As IPAddress
        Ip = Dns.GetHostEntry(txtRemoteHost.Text).AddressList(0)

        Dim Msg() As Byte, S As String
        Dim Ep As New IPEndPoint(Ip, Integer.Parse(txtRemotePort.Text))

        Do While Thoat = False
            Application.DoEvents()
            If UngDung1.Available > 0 Then
                Msg = UngDung1.Receive(Ep)
                S = System.Text.Encoding.UTF8.GetString(Msg)
                RaiseEvent Dữ_Liệu_Về (S, Ep.Address.ToString)
            End If
        Loop
    End Sub
```

```

Private Sub Gửi_Dữ_Liệu()
    Dim Msg() As Byte

    '/// Chuyển xâu sáng mảng byte để gửi đi
    Msg = System.Text.Encoding.UTF8.GetBytes(txtMsg.Text)

    '/// Gửi vào cổng có số hiệu đặt trong txtRemotePort.Text
    UngDung1.Send(Msg, Msg.Length, txtRemoteHost.Text, Integer.Parse(txtRemotePort.Text))
End Sub

Private Sub cmdSend_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSend.Click
    Gửi_Dữ_Liệu ()
    lstSent.Items.Insert(0, txtMsg.Text)
    txtMsg.Text = ""
End Sub

Private Sub frmCHAT_Dữ_Liệu_Về (ByVal Data As String, ByVal RemoteHost As String) Handles Me.Dữ_Liệu_Về
    Dim Msg As String
    Msg = Data & " (Người gửi : " & RemoteHost & ")"
    lstReceived.Items.Insert(0, Msg)
'□ xử lý
'.... Xử lý
'.....

End Sub

Private Sub frmCHAT_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
    Thoat = True '/// Thoát khỏi vòng lặp.
End Sub

Private Sub frmCHAT_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    UngDung1 = New UdpClient(Integer.Parse(txtLocalPort.Text))
    Th = New Thread(AddressOf Thăm_Dò)
    Th.Start()
End Sub
End Class

```

**\*\* Câu hỏi:** Trong sự kiện **Dữ\_Liệu\_Về** ta có thể bỏ bớt tham số **RemoteHost** đi được không ? (Hay có thể tăng thêm được không ?)

## Tổng kết:

Khi muốn gửi dữ liệu qua mạng bằng lớp **UdpClient**, ta theo cách đơn giản nhất như sau:

1	Tạo một UDPClient và gán (Bind — gán) cho nó một số hiệu cổng.	Dim udp as New UDPClient(1000)
2	Tạo một địa chỉ IP ứng với địa chỉ của máy mà ta muốn giao tiếp bằng IPEndPoint hoặc IPAddress hoặc IPHostEntry. (Lưu ý: Nếu dùng DNS.GetHostEntry thì ta có thể truyền vào là tên của máy. Sau đó muốn lấy địa chỉ thì chỉ việc viết, ví dụ: DNS.GetHostEntry("Tên_Máy").Address(0))	Dim IpAdd as IPAddress  IpAdd = Dns.GetHostEntry("RemoteHost").Address(0)
3	Gửi dữ liệu đi: - b1: Chuyển xâu thành mảng byte - b2: Gọi phương thức Send, trong đó truyền địa chỉ IP của máy ở xa mà ta vừa tạo ở 2 và thêm vào số hiệu cổng mà máy ở xa đang dùng để nhận dữ liệu.	UDPObj.Send(Msg, IpAdd, RemotePort)

### Khi nhận:

Dùng phương thức Receive để nhận dữ liệu về. Phương thức đòi hỏi ta phải chỉ ra là lấy về từ máy nào ? (mà đại diện là một IPEndPoint). Khi đó ta cần tạo một đối tượng IPEndPoint với địa chỉ và số hiệu cổng của máy chạy ứng dụng mà ta muốn nhận dữ liệu.

Phương thức này trả về cho ta dữ liệu ở dạng mảng byte, do vậy để chuyển sang dạng xâu ký tự thì cần dùng lớp Encoding để chuyển đổi.

Phương thức Receive làm việc ở chế độ đồng bộ (Tức là sẽ luôn “Blocking” khi chưa có dữ liệu nhận) do vậy thường ta sử dụng cơ chế đa tuyến để giải quyết trường hợp này. (Phần Receive sẽ được đặt trong một tuyến riêng biệt)

### Bài tập:

**Bài 1:** Viết chương trình UDP đặt ở hai máy thực hiện công việc sau:

- Khi một ứng dụng gửi xâu "OPEN#<Đường dẫn >" thì ứng dụng trên máy kia sẽ mở file nằm trong phần <đường dẫn>
- Khi một ứng dụng gửi xâu "SHUTDOWN" thì ứng dụng kia sẽ tắt máy tính.
- Khi một ứng dụng gửi xâu "RESTART" thì ứng dụng kia sẽ tắt khởi động lại máy tính.

**Bài 2:** Viết chương trình UDP (ứng dụng A) đặt trên một máy. thực hiện các công việc sau:



- Khi một ứng dụng (B) gửi một xâu chữ Tiếng Anh thì ứng A sẽ gửi trả lại nghĩa tiếng Việt tương ứng. Nếu từ Tiếng Anh không có trong từ điển (từ điển ở đây chỉ có 3 từ Computer, RAM, HDD) thì ứng dụng A gửi trả lại xâu "Not found".

.... → Viết các ứng dụng khác !

# Lớp TCP




## Giới thiệu

Mục đích của lớp UDPClient ở trên là dùng cho lập trình với giao thức UDP, với giao thức này thì hai bên không cần phải thiết lập kết nối trước khi gửi do vậy mức độ tin cậy không cao. Để đảm bảo độ tin cậy trong các ứng dụng mạng, người ta còn dùng một giao thức khác, gọi là giao thức có kết nối : TCP (Transport Control Protocol). Trên Internet chủ yếu là dùng loại giao thức này, ví dụ như Telnet, HTTP, SMTP, POP3... Để lập trình theo giao thức TCP, MS.NET cung cấp hai lớp có tên là TCPClient và TCPListener.

## Các thành viên của lớp TCPClient




Constructor Method	
Name	Description
<a href="#">TcpClient ()</a>	Tạo một đối tượng <b>TcpClient</b> . Chưa đặt thông số gì.
<a href="#">TcpClient (EndPoint)</a>	Tạo một <b>TcpClient</b> và gán cho nó một EndPoint cục bộ. (Gán địa chỉ máy cục bộ và số hiệu cổng để sử dụng trao đổi thông tin về sau)
<a href="#">TcpClient (RemoteHost: String, RemotePort: Int32)</a> <a href="#">[link]</a>	Tạo một đối tượng <b>TcpClient</b> và kết nối đến một máy có địa chỉ và số hiệu cổng được truyền vào.. RemoteHost có thể là địa chỉ IP chuẩn hoặc tên máy.

## Public Properties (see also [Protected Properties](#) )

	Name	Description
	<a href="#">Available</a>	Cho biết số byte đã nhận về từ mạng và có sẵn để đọc.
	<a href="#">Client</a>	Trả về Socket ứng với TCPClient hiện hành.
	<a href="#">Connected</a>	Trạng thái cho biết đã kết nối được đến Server hay chưa ?

## Public Methods (see also [Protected Methods](#) )

	Name	Description
--	------	-------------

	<a href="#">Close</a>	Giải phóng đối tượng <b>TcpClient</b> nhưng không đóng kết nối.
	<a href="#">Connect</a> (RemoteHost, RemotePort)	Kết nối đến một máy TCP khác có Tên và số hiệu cổng.
	<a href="#">GetStream</a>	Trả về <a href="#">NetworkStream</a> để từ đó giúp ta gửi hay nhận dữ liệu. (Thường làm tham số khi tạo StreamReader và StreamWriter để gửi và nhận dữ liệu dưới dạng xâu ký tự) .re6Khi đã gắn vào StreamReader và StreamWriter rồi thì ta có thể gửi và nhận dữ liệu thông qua các phương thức Readline, writeline tương ứng của các lớp này.

Từ các thành viên của lớp TCPClient ở trên ta thấy rằng, việc kết nối và thực hiện gửi nhận rất đơn giản. Theo các trình tự sau:

B1: Tạo một đối tượng TCPClient

B2: Kết nối đến máy chủ (Server) dùng phương thức **Connect**

B3: Tạo 2 đối tượng StreamReader (Receive)và StreamWriter (Send) và "**nối**" với GetStream của TCPClient

B4: - Dùng đối tượng StreamWriter.Writeline/write vừa tạo ở trên để gửi dữ liệu đi.

- Dùng đối tượng StreamReader.Readline/Read vừa tạo ở trên để đọc dữ liệu về.

B5: Đóng kết nối.

\*\*\* Nếu muốn gửi/nhận dữ liệu ở mức byte (nhị phân) thì dùng NetworkStream. (truyền GetStream cho NetworkStream)

## Ví dụ

Tạo một TCP Client và kết nối đến server (FTP Server-listen on 21 port), sau đó gửi 1 xâu.

```
Imports System.Net.Sockets
```

```
Imports System.Net
```

```
Imports System.IO
```

Public Class Form1

'/// Tạo địa chỉ ứng với 127.0.0.1 (Có thể sử dụng nhiều cách đã được đề cập)

Dim DiaChi As Long = 1 \* 256 ^ 3 + 127 \* 256 ^ 0 '//= 127.0.0.1

'// Tạo một IPEndPoint từ địa chỉ IP và cổng (Vì TCPClient cần một IPEndPoint)

Dim LocalEP As New IPEndPoint(DiaChi, 100) '// cho cục bộ (client)

'/// Tạo một đối tượng TCP ứng với địa chỉ và cổng ở trên

Dim tcp As New TcpClient(LocalEP)

'/// Hai luồng nhập và xuất dùng để đọc/ghi vào kết nối TCP

Dim Ghi As StreamWriter

Dim Doc As StreamReader

Private Sub Form1\_Load(...)

tcp.Connect("localhost", 21) '//Kết nối đến máy chủ FTP

'MsgBox(tcp.Connected)

'/// Nối

Doc = New StreamReader(tcp.GetStream())

Ghi = New StreamWriter(tcp.GetStream())

'/// Gửi thử một xâu (tên đăng nhập) cho server (FTP Server)

Ghi.WriteLine("User quynm")

Ghi.Flush()

'/// Đọc dữ liệu do Server gửi về

Dim S As String

S = Doc.ReadLine()

```
MsgBox("Dữ liệu gửi từ server : " & S)
```

```
End Sub
```

```
Private Sub Gui_Du_Lieu(ByVal Data As String)
```

```
Ghi.WriteLine(Data)
```

```
Ghi.Flush()
```

```
End Sub
```

```
End Class
```

Ở ví dụ trên ta thấy rằng việc gửi thì có thể thực hiện nhiều lần với việc gọi nhiều lần phương thức `Gửi_Dữ_Liệu`. Tuy nhiên, đối với việc nhận dữ liệu thì ta chỉ thực hiện một lần. Trong trường hợp nếu ta muốn nhận dữ liệu bất cứ khi nào có dữ liệu về thì cần áp dụng kỹ thuật "Thăm dò" và "kích hoạt sự kiện" như trong phần UDPCClient.

Ý tưởng thực hiện như sau:

B1 : Tạo một TCPClient

B2 : Kết nối

B3 : Tạo một luồng mới, luồng này "chuyên theo dõi" xem có dữ liệu mới về hay không (chỉ việc kiểm tra bộ đệm (đối tượng `StreamReader.EndOfStream = True/False`). Nếu bộ đệm không rỗng (có dữ liệu mới) thì giá trị `EndOfStream` sẽ bằng `False`. Khi có dữ liệu trong bộ đệm thì ta kích hoạt (Raise) sự kiện `Có_Dữ_Liệu` lên. Trong sự kiện này ta sẽ viết các lệnh xử lý.

## **Listing 2 : Viết chương trình Telnet**

```
Imports System.Net.Sockets
```

```
Imports System.Net
```

```
Imports System.IO
```

```
Imports System.Threading
```

```
Public Class frmTelnet
```

```
'/// Tạo một đối tượng TCPClient
```

Dim tcp As New TcpClient()

'/// Hai luồng nhập và xuất dùng để ghi vào kết nối TCP

Dim Ghi As StreamWriter

Dim Doc As StreamReader

'/// Tạo một thread chuyên thăm dò dữ liệu

Dim Th As Thread

'/// Cờ báo hiệu khi thoát. Để tránh việc lặp vô hạn

Dim Thoat As Boolean = False

Public Event Dữ\_Liệu\_Về(ByVal Data As String)

Sub Thăm\_Dò()

Dim S As String

Do While Thoat = False

Application.DoEvents()

If Doc.EndOfStream = False Then

S = Doc.ReadLine

RaiseEvent Dữ\_Liệu\_Về(S)

End If

Loop

End Sub

Private Sub frmTelnet\_Dữ\_Liệu\_Về(ByVal Data As String) Handles Me.Dữ\_Liệu\_Về

lstreceived.Items.Insert(0, Data)

End Sub

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
```

```
Dim RPort As Long = Integer.Parse(txtRemotePort.Text)
```

```
Dim IpEnd As New IPEndPoint(IPAddress.Parse(txtRemoteHost.Text), RPort)
```

```
'/// Kết nối tới máy chủ
```

```
tcp.Connect(IpEnd)
```

```
Doc = New StreamReader(tcp.GetStream())
```

```
Ghi = New StreamWriter(tcp.GetStream())
```

```
Th = New Thread(AddressOf Thăm_Dò)
```

```
Th.Start()
```

```
End Sub
```

```
Private Sub cmdSend_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSend.Click
```

```
Gui_Du_Lieu(txtMsg.Text)
```

```
End Sub
```

```
Private Sub frmTelnet_FormClosing(ByVal sender As Object, ByVal e As System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
```

```
Thoat = True
```

```
End Sub
```

```
Private Sub Gui_Du_Lieu(ByVal Data As String)
```

```
Ghi.WriteLine(Data)
```

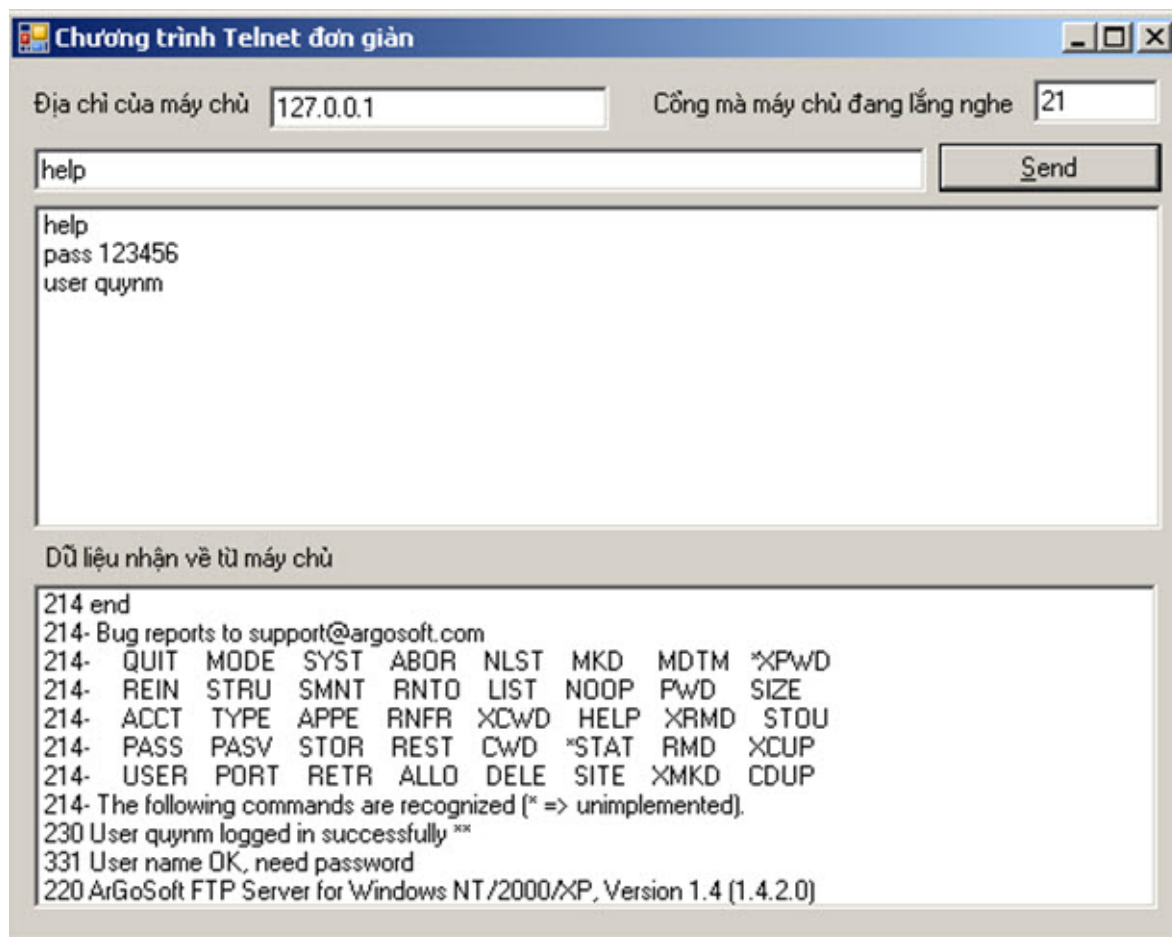
```
Ghi.Flush()
```

```
lstSent.Items.Insert(0, txtMsg.Text)
```

```
End Sub
```

End Class

Giao diện:



**Ghi chú:** Nếu muốn đọc hay ghi dữ liệu ở dạng chuỗi byte thì khai báo **Doc, Ghi AsNetworkStream**.

**Bài tập:** Viết ứng dụng chơi cờ Caro / Cờ tướng (Hay bất kỳ cờ gì khác !!!) qua mạng. (Sử dụng giao thức UDP). Gợi ý: mỗi khi người dùng đi thì sẽ gửi vị trí của ô vừa đi cho ứng dụng kia (đối phương).



# Lớp TCPListener






## Giới thiệu

TCPListener là một lớp cho phép người lập trình có thể xây dựng các ứng dụng Server (Ví dụ như SMTP Server, FTP Server, DNS Server, POP3 Server hay server tự định nghĩa ....). Ứng dụng server khác với ứng dụng Client ở chỗ nó luôn luôn thực hiện lắng nghe và chấp nhận các kết nối đến từ Client.

## Các thành viên của lớp

Constructor method	
Name	Description
<a href="#">TcpListener (Port: Int32)</a>	Tạo một <b>TcpListener</b> và lắng nghe tại cổng chỉ định.
<a href="#">TcpListener (EndPoint)</a>	Tạo một <b>TcpListener</b> với giá trị Endpoint truyền vào.
<a href="#">TcpListener (IPAddress, Port: Int32)</a>	Tạo một <b>TcpListener</b> và lắng nghe các kết nối đến tại địa chỉ IP và cổng chỉ định.

## Public Methods (see also [Protected Methods](#) )

	Name	Description
	<a href="#">AcceptSocket</a>	Chấp nhận một yêu cầu kết nối đang chờ.
	<a href="#">AcceptTcpClient</a>	Chấp nhận một yêu cầu kết nối đang chờ. (Ứng dụng sẽ dừng tại lệnh này cho đến khi nào có một kết nối đến – “Blocking”)
	<a href="#">Pending</a>	Cho biết liệu có kết nối nào đang chờ đợi không ? (True = có).
	<a href="#">Start</a>	Bắt đầu lắng nghe các yêu cầu kết nối.
	<a href="#">Stop</a>	Dừng việc nghe.

Khi AcceptTcpClient, thông tin riêng một Client sẽ giao tiếp với client vừa Accept

## Ví dụ

Tạo một server trong đó, khi có một client kết nối đến thì server chuyển xâu đó thành chữ HOA và gửi trả lại cho Client.

Listing 3: Xây dựng một ứng dụng Server đơn giản

```
Imports System.Net.Sockets
```

```
Imports System.Net
```

```
Imports System.IO
```

```
Imports System.Threading
```

```
Public Class frmServer
```

```
Dim TCPServer As New System.Net.Sockets.TcpListener(21)
```

```
Dim Thoat As Boolean = False
```

```
Dim Clients(100) As TcpClient
```

```
Dim CurrClient As Integer = 0
```

```
-----  
-----
```

```
Sub Xử_Lý_Kết_Nối()
```

```
Dim LastClient As Integer = CurrClient - 1
```

```
Dim Con As TcpClient = Clients(LastClient)
```

```
Dim Doc As New StreamReader(Con.GetStream)
```

```
Dim Ghi As New StreamWriter(Con.GetStream)
```

```
Dim S As String
```

```
While Thoat = False
```

```
Application.DoEvents()
```

```
If Doc.EndOfStream = False Then
```

```
S = Doc.ReadLine
```

```
'MsgBox("Client đã gửi xâu: " & S)
```

```
'// Xử lý tại đây: (Ví dụ chuyển thành chữ HOA)
```

```
S = S.ToUpper
```

```
Ghi.WriteLine(S) '//Gửi lại cho Client...
```

```
Ghi.Flush()
```

```
End If
```

```
End While
```

```
End Sub
```

```
-----  
-----
```

```
Sub Nghe_Kết_Nối()
```

```
Do While Thoat = False
```

```
Clients(CurrClient) = TCPServer.AcceptTcpClient()
```

```
CurrClient += 1
```

```
'MsgBox("Đã có " & (CurrClient + 1) & " kết nối !")
```

```
Dim Th As New Thread(AddressOf Xử_Lý_Kết_Nối)
```

```
Th.Start()
```

```
Loop
```

```
End Sub
```

```
-----  
-----
```

```
Private Sub frmClose(ByVal s As Object, ByVal e As FormClosingEventArgs) Handles Me.FormClosing
```

```
    Thoat = True
```

```
End Sub
```

```
-----  
-----
```

```
Private Sub Form1_Load(ByVal s As Object, ByVal e As EventArgs) Handles Me.Load
```

```
    TCPServer.Start()
```

```
    Nghe_Kết_Nối()
```

```
End Sub
```

```
End Class
```

Lưu ý: Vì phương thức **AcceptTCPClient()** luôn bị khóa (blocking) cho đến khi nào có một kết nối đến. Do vậy, để ứng dụng có thể thoát bình thường, ta nên thêm lệnh sau vào trước dòng `Clients(CurrClient) = TCPServer.AcceptTcpClient()`:

**If TCPServer.Pending = False Then Continue Do**

Tức là ta chỉ `AcceptTcpClient()` khi có kết nối đến !

# Bài tập

- Viết chương trình kiểm tra xem máy 192.168.1.1 có dịch vụ FTP đang chạy hay không ?
- Viết chương trình kiểm tra xem máy "Servercntt" có dịch vụ FTP đang chạy hay không ?
- Viết chương trình Telnet ở trỏn cho hoàn thiện.(có thể thay đổi tên máy, cổng...)
- Viết chương trình Server giải đáp tên miền. Nếu máy khách gửi tên máy thờ server sẽ gửi về địa chỉ IP. (danh sách này tự tạo ra – khoảng 3 cặp để minh họa).
- Viết chương trình Client/Server. Khi Client gửi đường dẫn của tệp nằm trên máy server thờ server gửi trả cho Client nội dung của tệp đó.
- Viết chương trình Client/Server để thực hiện CHAT. Trong đó các client gửi cho nhau thông qua "trạm trung chuyển" là Server.
- Viết chương trình SMTP server (giao thức đó được gửi cho lớp).
- Viết chương trình client/server trong đó, khi client di chuyển chuột thờ server cũng di chuyển chuột theo. (dùng cóc hàm API về SetCursorPos...)
- Viết chương trình Client/Server: Khi client gửi số xõu "shutdown", "restart" thờ Server sẽ tắt máy và khởi động tương ứng. (dùng hàm API ExitWindow...)

## Tham gia đóng góp

Tài liệu: Lập trình Socket và UDP, TCP

Biên tập bởi: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://voer.edu.vn/c/a516d57c>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Khái niệm Địa chỉ và cổng (Address & Port)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/f0b92bca>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lớp IPAddress

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/de7fc0f9>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lớp IPEndpoint

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/9ed2c3fa>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lớp IPHostEntry

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/3c3b4078>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lớp DNS

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/e30bc4e0>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lớp UDP

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/b06d7be5>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lớp TCP

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/5d97d7d9>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lớp TCPListener

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/cde766da>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Bài tập

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/99bc6925>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

## **Chương trình Thư viện Học liệu Mở Việt Nam**

Chương trình Thư viện Học liệu Mở Việt Nam (Vietnam Open Educational Resources – VOER) được hỗ trợ bởi Quỹ Việt Nam. Mục tiêu của chương trình là xây dựng kho Tài nguyên giáo dục Mở miễn phí của người Việt và cho người Việt, có nội dung phong phú. Các nội dung đều tuân thủ Giấy phép Creative Commons Attribution (CC-by) 4.0 do đó các nội dung đều có thể được sử dụng, tái sử dụng và truy nhập miễn phí trước hết trong môi trường giảng dạy, học tập và nghiên cứu sau đó cho toàn xã hội.

Với sự hỗ trợ của Quỹ Việt Nam, Thư viện Học liệu Mở Việt Nam (VOER) đã trở thành một cổng thông tin chính cho các sinh viên và giảng viên trong và ngoài Việt Nam. Mỗi ngày có hàng chục nghìn lượt truy cập VOER ([www.voer.edu.vn](http://www.voer.edu.vn)) để nghiên cứu, học tập và tải tài liệu giảng dạy về. Với hàng chục nghìn module kiến thức từ hàng nghìn tác giả khác nhau đóng góp, Thư Viện Học liệu Mở Việt Nam là một kho tàng tài liệu khổng lồ, nội dung phong phú phục vụ cho tất cả các nhu cầu học tập, nghiên cứu của độc giả.

Nguồn tài liệu mở phong phú có trên VOER có được là do sự chia sẻ tự nguyện của các tác giả trong và ngoài nước. Quá trình chia sẻ tài liệu trên VOER trở lên dễ dàng như đếm 1, 2, 3 nhờ vào sức mạnh của nền tảng Hanoi Spring.

Hanoi Spring là một nền tảng công nghệ tiên tiến được thiết kế cho phép công chúng dễ dàng chia sẻ tài liệu giảng dạy, học tập cũng như chủ động phát triển chương trình giảng dạy dựa trên khái niệm về học liệu mở (OCW) và tài nguyên giáo dục mở (OER). Khái niệm chia sẻ tri thức có tính cách mạng đã được khởi xướng và phát triển tiên phong bởi Đại học MIT và Đại học Rice Hoa Kỳ trong vòng một thập kỷ qua. Kể từ đó, phong trào Tài nguyên Giáo dục Mở đã phát triển nhanh chóng, được UNESCO hỗ trợ và được chấp nhận như một chương trình chính thức ở nhiều nước trên thế giới.