TRƯỜNG ĐẠI HỌC MỞ TP.HCM
Khoa Công Nghệ Thông Tin

LẬP TRÌNH MẠNG
(Network programming)

MÔN HỌC: LẬP TRÌNH MẠNG

- Chương 1: Những vấn đề cơ bản về lập trình mạng
- Chương 2: Lập trình SOCKET hướng kết nối
- Chương 3: Lập trình SOCKET phi kết nối
- Chương 4: Sử dụng các lớp hỗ trợ SOCKET
- Chương 5: Lập trình đa luồng,xây dựng, phát triển ứng dụng mạng
- Chương 6: Giao tiếp với Web Server
- Chương 7: Giao tiếp với email server
- Chương 8: Web services and Remoting

CHƯƠNG 1: NHỮNG VẤN ĐỀ CƠ BẢN CỦA LẬP TRÌNH MẠNG

- Cơ bản về mạng máy tính
- Mô hình truyền thông
- Các mô hình tham chiếu
- Tổng quan về lập trình mạng

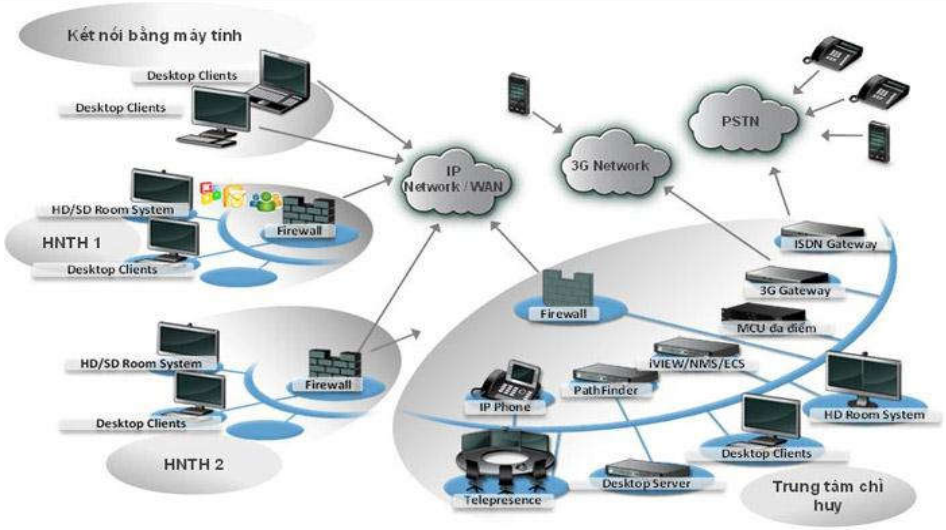
MỤC ĐÍCH – YÊU CẦU

- Mục đích: Giới thiệu các khái niệm cơ bản về
  - Mạng máy tính, truyền thông mạng máy tính
  - Mô hình OSI, TCP/IP
  - Windows Socket
  - Lập trình Socket trong C#
  - Các lớp DNS trong C#
- Yêu cầu: Sinh viên nắm vững các khái niệm cơ bản về
  - Lập trình mạng
  - Sử dụng socket trong lập trình mạng

NHỮNG VẤN ĐỀ CƠ BẢN CỦA LẬP TRÌNH MẠNG

- Cơ bản về mạng máy tính
- Mô hình truyền thông
- Các mô hình tham chiếu
- Tổng quan về lập trình mạng

Cơ bản về mạng máy tính



Cơ bản về mạng máy tính

- **Giao thức mạng**

Protocol: Quy tắc để truyền thông

Gửi: Một thông điệp với yêu cầu hoặc thông tin

Nhận: Nhận một thông điệp với thông tin, sự kiện hoặc hành động

Định nghĩa khuôn dạng và thứ tự truyền, nhận thông điệp giữa các thực thể trên mạng hoặc các hành động tương ứng khi nhận thông điệp

Ví dụ về giao thức mạng: TCP, UDP, IP, HTTP, Telnet, SSH, Ethernet, …

NHỮNG VẤN ĐỀ CƠ BẢN CỦA LẬP TRÌNH MẠNG

- Cơ bản về mạng máy tính
- Mô hình truyền thông
- Các mô hình tham chiếu
- Tổng quan về lập trình mạng

Mô hình truyền thông

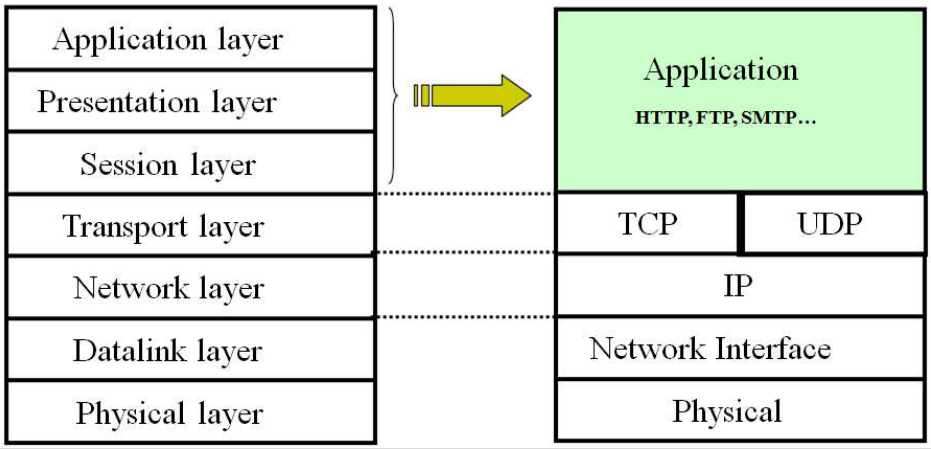
- **Truyền thông hướng liên kết và không hướng liên kết**
  - Truyền thông hướng kết nối
    - Dữ liệu được truyền qua một liên kết đã được thiết lập
    - Thông qua 3 giai đoạn: Thiết lập liên kết, truyền dữ liệu hủy bỏ liên kết.
    - Đáng tin cậy
  - Truyền thông không hướng kết nối
    - Không thiết lập liên kết, chỉ có giai đoạn truyền dữ liệu
    - Không tin cậy - "Best effort"

NHỮNG VẤN ĐỀ CƠ BẢN CỦA LẬP TRÌNH MẠNG

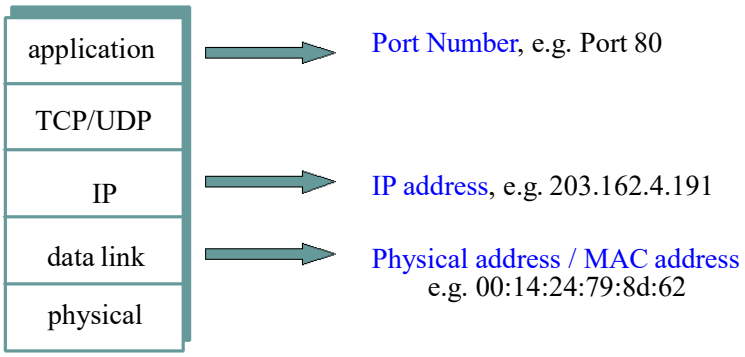
- Cơ bản về mạng máy tính
- Mô hình truyền thông
- Các mô hình tham chiếu
- Tổng quan về lập trình mạng

Các mô hình tham chiếu

- **Mô hình OSI và TCP/IP**
  - Trong mô hình TCP/IP (Internet), chức năng 3 tầng trên được phân định vào một tầng duy nhất



Định danh trên Internet và quan hệ với các tầng

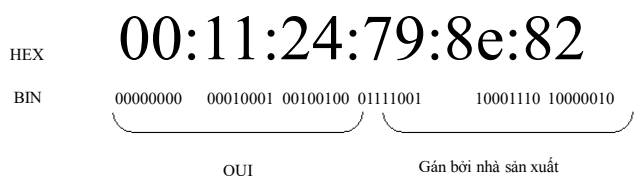


Địa chỉ dùng trong tầng liên kết dữ liệu

Địa chỉ vật lý /địa chỉ MAC

Sử dụng trong tầng liên kết dữ liệu
Cố định trên card mạng NIC (Network Interface Card)

Sử dụng để địa chỉ hóa máy tính trong các mạng quảng bá



OUI (Organizationally Unique Identifier): Mã nhà sản xuất
Mỗi nhà sản xuất có các giá trị OUI riêng
Một nhà sản xuất có thể có nhiều OUI

Định danh dùng trên Internet

Địa chỉ IP
Dùng trong giao thức IP - Internet Protocol (tầng mạng)
Giá trị phụ thuộc từng mạng, mỗi card mạng được gán một địa chỉ IP

Sử dụng để định danh máy tính trong một mạng IP
ví dụ :
133.113.215.10 (ipv4)
2001:200:0:8803::53 (ipv6)
2001:260:0:10:0:0:0:1
2001:260:0:10::1

Địa chỉ sử dụng trong tầng giao vận

- Số hiệu cổng: Một địa chỉ phụ, dùng kèm theo địa chỉ IP
- Các ứng dụng được định danh bởi một địa chỉ IP và một số hiệu cổng
- E.g. HTTP cổng 80, FTP cổng 20, 21 …

Các RFC quan trọng

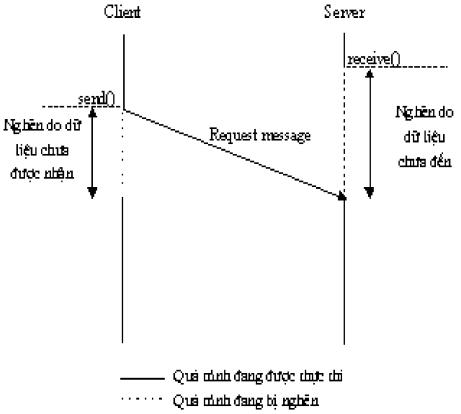
RFC Document	Giao thức
RFC 821	SMTP (email, outgoing)
RFC 954	WHOIS
RFC 959	FTP (uploading and downloading)
RFC 1939	POP3 (email, incoming)
RFC 2616	HTTP (Web browsing)
RFC 793	TCP (runs under all above protocols)
RFC 792	ICMP (ping)
RFC 791	IP (runs under TCP and ICMP)

NHỮNG VẤN ĐỀ CƠ BẢN CỦA LẬP TRÌNH MẠNG

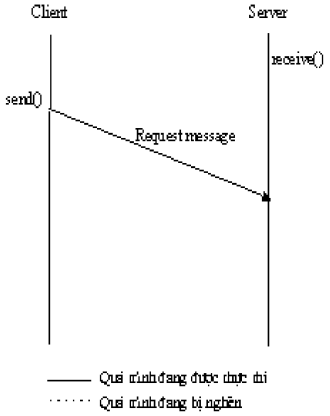
- Cơ bản về mạng máy tính
- Mô hình truyền thông
- Các mô hình tham chiếu
- Tổng quan về lập trình mạng

Mô hình Client / Server

- **Các chế độ giao tiếp:**



Chế độ giao tiếp nghẽn



Chế độ giao tiếp không nghẽn

Giới thiệu về Socket

- Sockets cung cấp một interface để lập trình mạng tại tầng Transport.
- Một socket là một end-point của một liên kết giữa hai ứng dụng mạng.
- Nhiều NNLT: C, C++, Java, VB, C#, . . .
- Windows Socket Application Programming Interface (Winsock API)
- Winsock hỗ trợ xây dựng các ứng dụng mạng trên nền TCP/IP.
- .NET hỗ trợ lập trình mạng tốt nhất so với các sản phẩm khác của Microsoft

=> **Mục tiêu:** nghiên cứu cách xây dựng ứng dụng truyền thông client/server dùng Sockets

Giới thiệu về Socket

- Giới thiệu về Windows Socket
  - Windows Sockets Application Programming Interface (WinSock API) là một giao diện lập trình mạng dưới nền tảng của Windows.
  - Đầu tiên Windows Socket được phát triển cho hệ điều hành Linux, nhưng nay nó đã được cài đặt trền Windows và hỗ trợ cơ chế điều khiển thông điệp tự nhiên của Windows.
  - Windows Socket đã phát triển qua nhiều phiên bản
  - Vd phiên bản chạy

    trên Windows XP và 2000 là 2.2

Platform	*Winsock Version*
Windows 95	1.1 (2.2)
Windows 98	2.2
Windows NT 4.0	2.2
Windows 2000	2.2
Windows CE	1.1

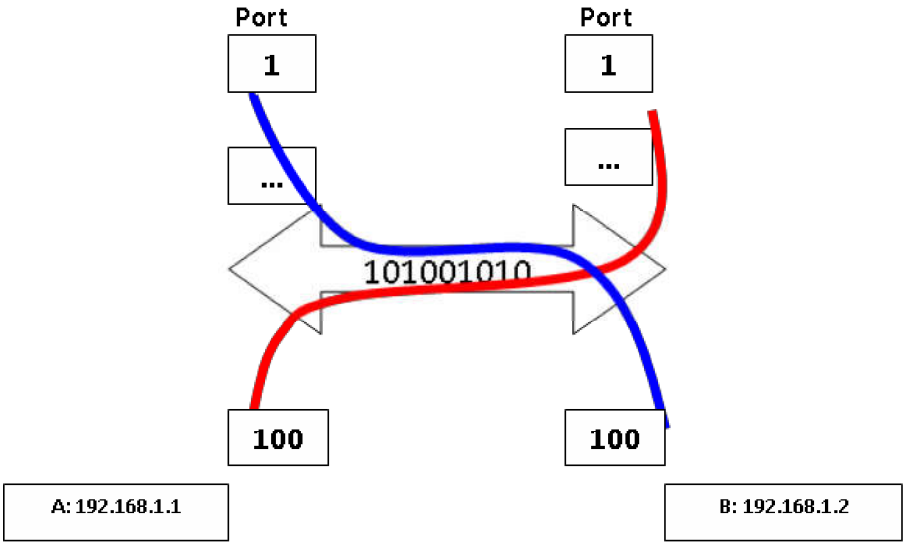
Giới thiệu về Socket

- Giới thiệu về Windows Socket
  - Windows Sockets cho phép những nhà sản xuất cung cấp một giao diện đồng nhất để người lập trình có thể viết các chương trình ứng dụng theo những đặc tả của Windows Sockets và có thể chạy được trên bất kỳ hệ thống nào có hỗ trợ Windows Sockets.
  - Các phiên bản sau của Windows socket đều tương thích với các phiên bản trước của nó. Điều đó có nghĩa là một ứng dụng được viết cho phiền bản trước cũng hoàn toàn có thể chạy được trên phiên bản sau của Windows Sockets.
  - Windows Sockets hỗ trợ nhiều giao thức khác: IPX/SPX, Apple's Appletalk, ATM,….

Giới thiệu về Socket

- Một số khái niệm
  - Socket: Một điểm giao tiếp đầu cuối của một chương trình trên mạng.
  - Một TCP/IP Sockets được xác định bằng sự kết nối của một địa chỉ và một số hiệu cổng
    - Địa chỉ sẽ xác định máy mà Sockets đó làm việc
    - Cổng sẽ xác định chính xác tiến trình đang thực hiện Sockets trên máy có địa chỉ đó.

Address & Port



Address & Port

- **Nguyên lý**
  - Trong máy có rất nhiều ứng dụng muốn trao đổi với các ứng dụng khác thông qua mạng.

Ví dụ: có 2 ứng dụng của máy A muốn trao đổi với 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**
  - Có thể xảy ra "nhầm lẫn" khi dữ liệu từ máy A gửi đến máy B thì trên máy B không biết là dữ liệu đó gửi cho ứng dụng nào?
  - Mỗi ứng dụng trên máy B sẽ được gán một số hiệu (cổng: Port), từ 0..65535.

Address & Port

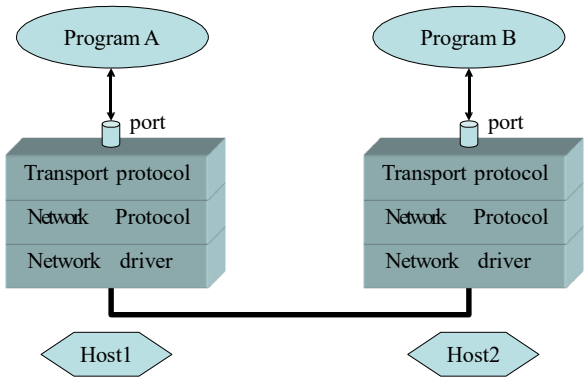
- **Cách giải quyết**
  - Khi ứng dụng trên máy A muốn gửi cho ứng dụng nào trên máy B thì chỉ việc điền thêm số hiệu cổng (vào trường RemotePort) vào gói tin cần gửi.
  - Trên máy B, các ứng dụng chỉ việc kiểm tra giá trị cổng trên mỗi gói tin xem có trùng với số hiệu cổng của mình (đã được gán – chính là giá trị LocalPort) hay không? Nếu bằng thì xử lý, trái lại thì không làm gì (vì không phải là của mình).

Port	Protocol
20	FTP data
21	FTP control
25	SMTP (email, outgoing)
53	DNS (Domain Name Service)
80	HTTP (Web)
110	POP3 (email, incoming)
143	IMAP (email, incoming)

Address & Port

- **Một số quy định**
  - Không bao giờ có 2 ứng dụng lại cùng dùng 1 port
  - Các port từ 0 – 1023 (Well-known): dùng cho các ứng dụng quan trọng trên hệ điều hành
  - Các port từ 1024 – 49151 (Registered): dành cho người lập trình (khuyến cáo nên tuân thủ)
  - Các port từ 49152 – 65535 (Dynamic): dự trữ

- Giao tiếp sử dụng socket có thể ở chế độ hướng kết nối (giao thức TCP) hoặc không hướng kết nối (giao thức UDP) bằng cách xác định giao thức của tầng vận chuyển.



Giao tiếp giữa 2 quá trình sử dụng Socket

- Chế độ có kết nối (connection):
  - Tồn tại kênh giao tiếp ảo giữa client và server.
  - Dữ liệu được gởi đi theo chế độ bảo đảm:
  - Kiểm tra lỗi.
  - Truyền lại gói tin lỗi, mất
  - Bảo đảm thứ tự các gói tin
  - . . .
- Dữ liệu chính xác, Tốc độ truyền chậm.

- Chế độ không kết nối (connectionless):
  - Không tồn tại kênh giao tiếp ảo giữa server.
  - Dữ liệu được gởi đi theo chế độ không bảo đảm:
    - Không kiểm tra lỗi.
    - Không phát hiện,không truyền lại gói tin lỗi,mất.
    - Không bảo đảm thứ tự cái gói tin
    - . . .
  - Dữ liệu không chính xác, tốc độ truyền nhanh.
  - Thích hợp cho các ứng dụng cần tốc độ, không cần độ chính xác cao như hình ảnh, video . . .

Lập trình Socket với C#

- Giao tiếp sử dụng socket có thể ở chế độ hướng kết nối (giao thức TCP) hoặc không hướng kết nối (giao thức UDP) bằng cách xác định giao thức của tầng vận chuyển.

Lập trình Socket với C#

- Các lớp trong .NET Framework được tạo ra để cung cấp một giao diện dễ dàng cho các lập trình viên.
- Nội dung phần này sẽ gồm:
  - IP Addresses in C#
  - Using C# Sockets

Lớp IPAddress

- 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 như May01, Server, …

Địa chỉ IP nhưng đặt trong một chuỗi: "192.168.1.1", "127.0.0.1"

Đặt trong một mảng 4 byte, mỗi byte chứa một số từ 0-255.

Hoặc cũng có thể là một số (long), có độ dài 4 byte. Ví dụ, với địa chỉ 192.168.1.2 ở trên thì giá trị đó sẽ là 33663168 (số ở hệ thập phân khi xếp liền 4 byte ở trên lại với nhau) **00000010**00000001**10101000**11000000

2 (byte 0) 1 (byte 1) 168 (byte 2) 192 (byte 3)

31 65

Lớp IPAddress

- Như vậy, để đổi một địa chỉ chuẩn ra dạng số chúng 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 =

  33663168
- **IPAddress** được sử dụng để biểu hiện một địa chỉ IP duy nhất
- Hàm tạo là : public IPAddress(long *address*)
- Tuy nhiên hàm tạo này không được sử dụng thường xuyên.
- Lớp IPAddress cung cấp một số phương thức khác để làm việc với địa chỉ IP

32

Lớp IPAddress: các thành viên

Tên phương thức	Mô tả
AddressFamily	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.
Constructor	- IPAddress(Số_Long) → Tạo địa chỉ IP từ một số kiểu long - IPAddress(Mảng_Byte) → Tạo địa chỉ IP từ một mảng byte (4 byte).
GetAddressBytes	Chuyển địa chỉ thành mảng byte (4 byte).
HostToNetworkOrder	Đảo thứ tự byte của một số cho đúng với thứ tự byte trong địa chỉ IPAddress.

Lớp IPAddress: các thành viên

Tên phương thức	Mô tả
IsLoopback	Cho biết địa chỉ có phải là địa chỉ lặp hay không?
NetworkToHostOrder	Đảo thứ tự byte của một địa chỉ cho đúng với thứ tự byte thông thường.
Parse	Chuyển một địa chỉ IP ở dạng chuỗi thành một địa chỉ IP chuẩn (Một đối tượng IPAddress)
ToString	Trả về địa chỉ IP (một chuỗi) nhưng ở dạng ký pháp có dấu chấm. (Ví dụ "192.168.1.1").
TryParse (S: String)	Kiểm tra xem một địa chỉ IP (ở dạng chuỗi) có phải đúng là địa chỉ IP hợp lệ hay không? True = đúng

IPAddress: Ví dụ tạo địa chỉ

- Cách 1: *Dùng hàm khởi tạo*

```
Byte[] b = new Byte[4];  b[0] = 192;
b[1] = 168;
b[2] = 1;
b[3] = 2;
IPAddress Ip1 = new IPAddress(b);
```

- Cách 2: *Dùng hàm khởi tạo*

```
IPAddress Ip2 = new IPAddress(33663168);
```

- Cách 3: *Dùng hàm khởi tạo*

```
IPAddress Ip3 = IPAddress.Parse("192.168.1.2")
```

- Cách 4: *Thông qua tính toán*

```
Long So = 192* 256^0+168* 256^1+1* 256^2 + 2*256^3;
IPAddress Ip4 = new IPAddress(So);
```


Lớp IPAddress: các thành viên

Tên thuộc tính	Mô tả
Any	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.
Broadcast	Cung cấp một địa chỉ IP quảng bá (Broadcast, thường là 255.255.255.255), ở dạng số Long. Muốn lấy ở dạng chuỗi, viết: Broadcast.ToString(). Thuộc tính này chỉ đọc.
Loopback	Trả về một địa chỉ IP lặp (IP Loopback, ví dụ 127.0.0.1). Thuộc tính này chỉ đọc.
Address	Một địa chỉ IP (An Internet Protocol (IP) address) ở dạng số Long. (Muốn chuyển sang dạng dấu chấm, viết : Address.ToString())

IPAddress: Ví dụ kiểm tra địa chỉ

```
private void KiemTra()
{
    IPAddress ip;
    String Ip4 = "127.0.0.1";
    String Ip5 = "10.0.0.1";
    MessageBox.Show(IPAddress.TryParse(Ip4, out ip).ToString());
    MessageBox.Show(IPAddress.TryParse(Ip5, out ip).ToString());
}
```


IPAddress: Ví dụ chuyển địa chỉ hiện hành ra mảng

```
void ChuyenDoi()
{
        IPAddress Ip3 = new IPAddress(16885952);
        Byte[] b= new Byte[4];
        b = Ip3.GetAddressBytes();
        MessageBox.Show("Address: " + b[0] + "." + b[1] + "." + b[2]
    + "." + b[3]);
}
```

Lớp IPEndpoint

- 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 địa chỉ IP (IPAddress), như vậy vẫn còn thiếu số hiệu cổng (Port number).
- .NET Framework sử dụng IPEndPoint để biểu diễn một sự kết hợp giữa cổng và địa chỉ IP
- Hai hàm tạo của lớp này là:
  - IPEndPoint(long *address*, int *port*)
  - IPEndPoint(IPAddress *address*, int *port*)
- Hàm tạo thứ 2 được sử dụng phổ biến hơn

39

Lớp IPEndpoint: các thành viên

Tên thuộc tính	Mô tả
Address	Trả về hoặc thiết lập địa chỉ IP cho endpoint. (Trả về một đối tượng IPAddress)
AddressFamily	Lấy về loại giao thức mà Endpoint này đang sử dụng.
Port	Lấy về hoặc thiết lập số hiệu cổng của endpoint.

40

Lớp IPEndpoint: các thành viên

Tên phương thức	Mô tả
IPEndPoint (Int64, Int32)	Tạo một đối tượng mới của lớp **IPEndPoint,** tham số truyền vào là địa chỉ IP (ở dạng số) và **cổng** sẽ dùng để giao tiếp.
IPEndPoint (IPAddress, Int32)	Tạo một đối tượng mới của lớp **IPEndPoint,** Tham số truyền vào là một địa chỉ **IPAddress** và số hiệu cổng dùng để giao tiếp. (Tham khảo cách tạo IPAddress ở phần trên)
Create	Tạo một endpoint từ một địa chỉ socket (socket address).
ToString	Trả về địa chỉ IP và số hiệu cổng theo khuôn dạng ĐịaChỉ: Cổng, ví dụ: 192.168.1.1:8080

Lớp IPEndpoint: ví dụ khởi tạo

```
private void TaoEndpoint()
{
    // Tạo một địa chỉ IP
    IPAddress IPAdd = IPAddress.Parse("127.0.0.1");
    // Truyền vào cho hàm khởi tạo để tạo IPEndpoint
    IPEndPoint IPep = new IPEndPoint(IPAdd, 10000);
}
```

Lớp IPEndpoint: ví dụ khởi tạo

```
private void CreateEndPointByName()
{
    IPAddress IPAdd;
    //tạo đối tượng IP từ tên của máy thông qua phương thức tĩnh
    Dns.GetHostAddresses của lớp DNS
    IPAdd = Dns.GetHostAddresses("Localhost")[0];
    IPEndPoint IPep = new IPEndPoint(IPAdd, 10000);
}
```


Lớp IPEndpoint: ví dụ khởi tạo

- 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 chúng ta một mảng chứa tất cả các địa chỉ đó.
- Chúng ta lấy chỉ số là 0 để chọn địa chỉ của card mạng đầu tiên.
- 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

Lớp IPHostEntry: các thành viên

Tên thuộc tính	Mô tả
AddressList	Lấy về hoặc thiết lập danh sách các địa chỉ IP liên kết với một host.
Aliases	Lấy về hoặc thiết lập danh sách các bí danh (alias) liên kết với một host.
HostName	Lấy về hoặc thiết lập DNS name của host.

45

Sử dụng C# Sockets

- System.Net.Sockets namespace có chứa các lớp cung cấp giao diện lập trình Net cho các hàm mức thấp API winsock
- Tạo socket
- Sử dụng hàm tạo của lớp Socket

Socket(AddressFamily *af*, SocketType *st*, ProtocolType *pt*)

- Trong đó:
  - An *AddressFamily* to define the network type
  - A *SocketType* to define the type of data connection
  - A *ProtocolType* to define a specific network protocol
- Giá trị của *AddressFamily* luôn là *InterNetwork*
- Còn giá trị của *SocketType và ProtocolType phải được kết hợp đúng*

46

Sử dụng C# Sockets

- Bảng kết hợp

SocketType	Protocoltype	Description
Dgram	Udp	Connectionless communication
Stream	Tcp	Connection-oriented communication
Raw	Icmp	Internet Control Message Protocol
Raw	Raw	Plain IP packet communication

- Vị dụ về tạo Socket

```
Socket newsock =
Socket(AddressFamily.InterNetwork,
SocketType.Stream, ProtocolType.Tcp);
```


Sử dụng C# Sockets

- Một số thuộc tính của lớp Socket

Property	Description
AddressFamily	Gets the address family of the Socket
Available	Gets the amount of data that is ready to be read
Blocking	Gets or sets whether the Socket is in blocking mode
Connected	Gets a value that indicates if the Socket is connected to a remote device
Handle	Gets the operating system handle for the Socket
LocalEndPoint	Gets the local EndPoint object for the Socket
ProtocolType	Gets the protocol type of the Socket
RemoteEndPoint	Gets the remote EndPoint information for the Socket
SocketType	Gets the type of the Socket

DNS (Domain Name Service)

- 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 thì đầu ra sẽ cho ta địa chỉ IP tương ứng của máy đó, ví dụ: Server1 →192.168.10.10
- Ngoài ra lớp Dns còn có rất nhiều phương thức cho chúng ta thêm thông tin về máy cục bộ như tên, địa chỉ, v.v.
- System.Net namespace có chứa lớp Dns trong đó có chứa tất cả các hàm cần thiết khi ta làm việc với hệ thống tên miền.
- Trong các phương thức này có
  - Các phương thức đồng bộ
  - Các phương thức không đồng bộ

Lớp DNS: các thành viên

Tên phương thức	Mô tả
GetHostByAddress (String IP) GetHostByAddress (IPAddress IP)	Trả về thông tin (IPHostEntry) của trạm có địa chỉ IP được truyền vào. → Có thể sử dụng **GetHostEntry**()
GetHostByName (String hostname)	Trả về thông tin (IPHostEntry) DNS của một trạm → Có thể sử dụng **GetHostEntry()**
HostName	Cho ta biết tên của máy vừa được phân giải. Nếu không phân giải được thì có giá trị là địa chỉ IP.

Lớp DNS: các thành viên

Tên phương thức	Mô tả
GetHostAddresses (String IP_Or_HostName)	Trả về tất cả các địa chỉ IP của một trạm. Kiểu IPAddress
GetHostEntry (String IP_Or_HostName) GetHostEntry (IPAddress IP)	Giải đáp tên hoặc địa chỉ IP truyền vào và trả về một đối tượng **IPHostEntry** tương ứng.
GetHostName	Lấy về tên của máy tính cục bộ (String).
Resolve (String Hostname)	Chuyển tên của máy hoặc địa chỉ IP thành **IPHostEntry** tương ứng. → Đã bị bỏ, thay bằng **GetHostEntry()**

- 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: Dns.Resolve, Dns.GetHostname, Dns.GetHostEntry, v.v...

51

Lớp DNS: ví dụ 1

```
private void ShowIPs()
{
        // Lấy tất cả địa chỉ IP của máy
         IPAddress[] add = Dns.GetHostAddresses("PC");
         foreach (IPAddress ip in add) {
            MessageBox.Show(ip.ToString());
         }
         //Cách 2
         //for (int i = 0; i < add.Length; i++)
         //{
         //    MessageBox.Show(add[i].ToString());
         //}
}
```

52

Lớp DNS: ví dụ 2

```
private void CreatIPHostEntry() {
        IPHostEntry iphe1, iphe2, iphe3;
        IPAddress ipadd = IPAddress.Parse("127.0.0.1");
        iphe1 = Dns.GetHostEntry("Notebook");
        iphe2 = Dns.GetHostEntry("127.0.0.1");
        iphe3 = Dns.GetHostEntry(ipadd);
        MessageBox.Show(iphe1.HostName);
        MessageBox.Show(iphe2.HostName) ;
        MessageBox.Show(iphe3.HostName) ;
}
```


TÓM LƯỢC BÀI HỌC

- Cơ bản về mạng máy tính
- Mô hình truyền thông
- Các mô hình tham chiếu
- Tổng quan về lập trình mạng

Hướng Dẫn Ví Dụ Trên Chương Trình Thực Tế

Chương 2

Lập trình TCP Socket