BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG
-------o0o-------

TÌM HIỂU KỸ THUẬT LẬP TRÌNH NETWORK SERVICE CHO WINDOW

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH CÔNG NGHỆ THÔNG TIN

Giáo viên hướng dẫn: **Ths. Đỗ Xuân Toàn**

Sinh viên: **Phạm Văn Ninh**

Mã số sinh viên: **121216**

Hải Phòng, 7/2012

MỤC LỤC

LỜI NÓI ĐẦU

Tin học và viễn thông là hai thành phần cốt lõi của công nghệ thông tin. Mạng máy tính không còn là thuật ngữ thuần túy khoa học mà đang trở thành một đối tượng nghiên cứu và ứng dụng cả nhiều phạm vi hoạt động khác nhau. Những năm gần đây,do sự phát triển vũ bão của công nghiệp máy tính,việc kết nối các mạng máy tính đã trở thành nhu cầu hiện thực cho người sử dụng.

Nhờ có mạng mà người dùng có thể sử dụng máy tính của mình để điều khiển các chương trình của máy tính khác trong cùng mạng.

Ngoài việc kết nối các mạng máy tính thì sự phát triển của mạng không dây còn giúp máy tính có thể kết nối với các thiết bị không dây. Vì vậy nhu cầu điều khiển các thiết bị cố định từ thiết bị di động là rất lớn.

Xuất phát từ yêu cầu trên em đã đi vào tìm hiểu và lập trình socket với đề tài: "**Tìm hiểu kỹ thuật lập trình Network Service"** để kết nối mạng máy tính và điều khiển chương trình đơn giản. Nhằm bước đầu hiểu về cách thức lập trình điều khiển từ xa.

Đề án trình bày gồm các chương:

Chương 1: Kỹ thuật lập trình mạng

Chương 2: Kỹ thuật lập trình window service

Chương 3: Xây dựng chương trình thực nghiệm

Đồ án được thực hiện trong khoảng thời gian tương đối ngắn nên không tránh khỏi còn nhiều thiếu sót. Em rất mong nhận được các ý kiến đóng góp của thầy cô và những người quan tâm.

LỜI CẢM ƠN

Trước tiên em xin chân thành cảm ơn sâu sắc đến các thầy cô trong trường Đại học Dân Lập Hải Phòng, đặc biệt các thầy cô trong khoa Khoa công nghệ thông tin đã truyền đạt kiến thức cho chúng em trong thời gian qua.

Em xin chân thành cảm ơn thầy **Đỗ Xuân Toàn** đã tận tình giúp đỡ, chỉ bảo hướng dẫn trực tiếp em để em hoàn thành tốt đồ án tốt nghiệp, trong thời gian làm được thầy hướng dẫn, em đã tiếp thu thêm nhiều kiến thức bổ ích trong quá trình học tập cũng như trong quá trình làm việc sau này.

Em xin gửi lời cảm ơn đến tất cả các bạn bè đã động viên, đóng góp ý kiến giúp đỡ trong quá trình học tập, nghiên cứu hoàn thành đồ án.

Và cuối cùng ,kính chúc thầy cô sức khỏe, tiếp tục đạt được nhiều thành tích trong giảng dạy, cũng như trong nghiên cứu khoa học và trong sự nghiệp giáo dục.

Xin chân thành cảm ơn !

Hải Phòng, ngày 2 tháng 7 năm 2012

Sinh viên thực hiện

Phạm Văn Ninh

CHƯƠNG 1: KỸ THUẬT LẬP TRÌNH MẠNG

1.1. Tổng quan về lập trình mạng

1.1.1. Mô hình tham khảo 7 tầng OSI

Mô hình kết nối hệ thống mở được Tổ chức quốc tế về tiêu chuẩn hoá ISO (International Organizaiton for Standardization) đưa ra nhằm cung cấp một mô hình chuẩn cho các nhà sản xuất và cung cấp sản phẩm viễn thông áp dụng theo để phát triển các sản phẩm viễn thông. Ý tưởng mô hình hoá được tạo ra còn nhằm hỗ trợ cho việc kết nối giữa các hệ thống và modun hoá các thành phần phục vụ mạng viến thông.

a) Chức năng của mô hình OSI

- Cung cấp kiến thức về hoạt động của kết nối liên mạng

- Đưa ra trình tự công việc để thiết lập và thực hiện một giao thức cho kết nối các thiết bị trên mạng.

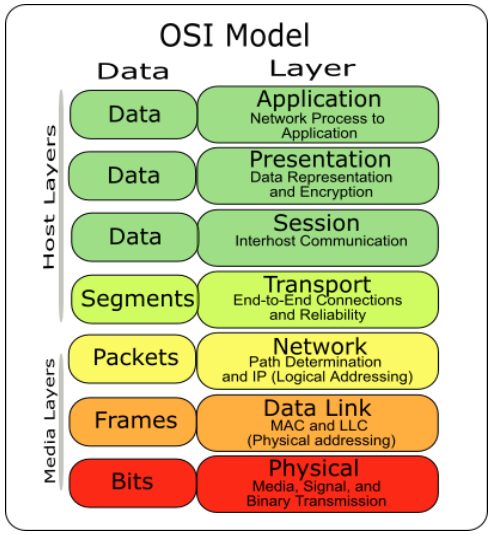
Mô hình OSI còn có một số thuận lợi sau :

+ Chia nhỏ các hoạt động phức tạp của mạng thành các phần công việc đơn giản.

+ Cho phép các nhà thiết kế có khả năng phát triển trên từng modun chức năng. Cung cấp các khả năng định nghĩa các chuẩn giao tiếp có tính tương thích cao "plug and play" và tích hợp nhiều nhà cung cấp sản phẩm.

b) Cấu trúc mô hình OSI

Mô hình OSI gồm 7 lớp (level), mỗi lớp thực hiện các chức năng riêng cho hoạt động kết nối mạng.



Mô hình 7 lớp OSI

Tầng 1: Tầng vật lí (Physical Layer)

Tầng vật lí định nghĩa tất cả các đặc tả về điện và vật lý cho các thiết bị. Trong đó bao gồm bố trí của các chân cắm (pin), các hiệu điện thế, và các đặc tả về cáp nối (cable). Các thiết bị tầng vật lí bao gồm Hub, bộ lặp (repeater), thiết bị tiếp hợp mạng (network adapter) và thiết bị tiếp hợp kênh máy chủ (Host Bus Adapter) - (HBA dùng trong mạng lưu trữ (Storage Area Network)). Chức năng và dịch vụ căn bản được thực hiện bởi tầng vật lý bao gồm:

+ Thiết lập hoặc ngắt mạch kết nối điện (electrical connection) với một môi trường truyền dẫn phương tiện truyền thông (transmission medium).

+ Tham gia vào quy trình mà trong đó các tài nguyên truyền thông được chia sẻ hiệu quả giữa nhiều người dùng. Chẳng hạn giải quyết tranh chấp tài nguyên (contention) và điều khiển lưu lượng.

+ Điều biến (modulation), hoặc biến đổi giữa biểu diễn dữ liệu số (digital data) của các thiết bị người dùng và các tín hiệu tương ứng được truyền qua kênh truyền thông (communication channel).

Tầng 2: Tầng liên kết dữ liệu (Data Link Layer)

Tầng liên kết dữ liệu cung cấp các phương tiện có tính chức năng và quy trình để truyền dữ liệu giữa các thực thể mạng, phát hiện và có thể sửa chữa các lỗi trong tầng vật lý nếu có. Cách đánh địa chỉ mang tính vật lý, nghĩa là địa chỉ (địa chỉ MAC) được mã hóa cứng vào trong các thẻ mạng (network card) khi chúng được sản xuất. Hệ thống xác định địa chỉ này không có đẳng cấp (flat scheme).

Tầng 3: Tầng mạng (Network Layer)

Tầng mạng cung cấp các chức năng và qui trình cho việc truyền các chuỗi dữ liệu có độ dài đa dạng, từ một nguồn tới một đích, thông qua một hoặc nhiều mạng, trong khi vẫn duy trì chất lượng dịch vụ (quality of service) mà tầng giao vận yêu cầu. Tầng mạng thực hiện chức năng định tuyến, .Các thiết bị định tuyến (router) hoạt động tại tầng này — gửi dữ liệu ra khắp mạng mở rộng, làm cho liên mạng trở nên khả thi (còn có thiết bị chuyển mạch (switch) tầng 3, còn gọi là chuyển mạch IP). Đây là một hệ thống định vị địa chỉ lôgic (logical addressing scheme) – các giá trị được chọn bởi kỹ sư mạng. Hệ thống này có cấu trúc phả hệ. Ví dụ điển hình của giao thức tầng 3 là giao thức IP.

Tầng 4: Tầng giao vận (Transport Layer)

Tầng giao vận cung cấp dịch vụ chuyên dụng chuyển dữ liệu giữa các người dùng tại đầu cuối, nhờ đó các tầng trên không phải quan tâm đến việc cung cấp dịch vụ truyền dữ liệu đáng tin cậy và hiệu quả. Tầng giao vận kiểm soát độ tin cậy của một kết nối được cho trước. Một số giao thức có định hướng trạng thái và kết nối (state and connection orientated). Có nghĩa là tầng giao vận có thể theo dõi các gói tin và truyền lại các gói bị thất bại. Một ví dụ điển hình của giao thức tầng 4 là TCP. Tầng này là nơi các thông điệp được chuyển sang thành các gói tin TCP hoặcUDP. Ở tầng 4 địa chỉ được đánh là address ports, thông qua address ports để phân biệt được ứng dụng trao đổi.

Tầng 5: Tầng phiên (Session layer)

Tầng phiên kiểm soát các (phiên) hội thoại giữa các máy tính. Tầng này thiết lập, quản lý và kết thúc các kết nối giữa trình ứng dụng địa phương và trình ứng dụng ở xa. Tầng này còn hỗ trợ hoạt động song công (duplex) hoặc bán song công (half-duplex) hoặc đơn công (Single) và thiết lập các qui trình đánh dấu điểm hoàn thành (checkpointing) - giúp việc phục hồi truyền thông nhanh hơn khi có lỗi xảy ra, vì điểm đã hoàn thành đã được đánh dấu - trì hoãn (adjournment), kết thúc (termination) và khởi động lại (restart). Mô hình OSI uỷ nhiệm cho tầng này trách nhiệm "ngắt mạch nhẹ nhàng" (graceful close) các phiên giao dịch (một tính chất của giao thức kiểm soát giao vận TCP) và trách nhiệm kiểm tra và phục hồi phiên, đây là phần thường không được dùng đến trong bộ giao thức TCP/IP.

Tầng 6: Tầng trình diễn (Presentation layer)

Lớp trình diễn hoạt động như tầng dữ liệu trên mạng. lớp này trên máy tính truyền dữ liệu làm nhiệm vụ dịch dữ liệu được gửi từ tầng Application sang dạng Fomat chung. Và tại máy tính nhận, lớp này lại chuyển từ Fomat chung sang định dạng của tầng Application. Lớp thể hiện thực hiện các chức năng sau: - Dịch các mã kí tự từ ASCII sang EBCDIC. - Chuyển đổi dữ liệu, ví dụ từ số interger sang số dấu phảy động. - Nén dữ liệu để giảm lượng dữ liệu truyền trên mạng. - Mã hoá và giải mã dữ liệu để đảm bảo sự bảo mật trên mạng.

Tầng 7: Tầng ứng dụng (Application layer)

Tầng ứng dụng là tầng gần với người sử dụng nhất. Nó cung cấp phương tiện cho người dùng truy nhập các thông tin và dữ liệu trên mạng thông qua chương trình ứng dụng. Tầng này là giao diện chính để người dùng tương tác với chương trình ứng dụng, và qua đó với mạng. Một số ví dụ về các ứng dụng trong tầng này bao gồm Telnet, Giao thức truyền tập tin FTP và Giao thức truyền thư điện tử SMTP, HTTP, X.400 Mail remote.

1.1.2. Giao thức TCP/IP

IP là một họ giao thức để cung cấp phương tiện truyền thông liên mạng và nó được cấu trúc theo kiểu phân cấp.

Khác với mô hình OSI tầng liên mạng sử dụng giao thức kết nối mạng "không liên kết" (connectionless) IP, tạo thành hạt nhân hoạt động của Internet. Cùng với các thuật toán định tuyến RIP, OSPF, BGP, tầng liên mạng IP cho phép kết nối một cách mềm dẻo và linh hoạt các loại mạng "vật lý" khác nhau như: Ethernet, Token Ring , X.25...

Giao thức trao đổi dữ liệu "có liên kết" (connection - oriented) TCP được sử dụng ở tầng vận chuyển để đảm bảo tính chính xác và tin cậy việc trao đổi dữ liệu dựa trên kiến trúc kết nối "không liên kết" ở tầng liên mạng IP.

Các giao thức hỗ trợ ứng dụng phổ biến như truy nhập từ xa (telnet), chuyển tệp (FTP), dịch vụ World Wide Web (HTTP), thư điện tử (SMTP), dịch vụ tên miền (DNS) ngày càng được cài đặt phổ biến như những bộ phận cấu thành của các hệ điều hành thông dụng như UNIX (và các hệ điều hành chuyên dụng cùng họ của các nhà cung cấp thiết bị tính toán như AIX của IBM, SINIX của Siemens, Digital UNIX của DEC), Windows9x/NT, NovellNetware,...

1.1.3. So sánh 2 giao thức TCP và UDP

UDP (User Datagram Protocol) là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gởi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm. Các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.

TCP (Transmission Control Protocol - "Giao thức điều khiển truyền vận") là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các "kết nối" với nhau, mà qua đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.

Khác nhau (cơ bản):

Các header của TCP và UDP khác nhau ở kích thước (20 và 8 byte) nguyên nhân chủ yếu là do TCP phải hộ trợ nhiều chức năng hữu ích hơn(như khả năng khôi phục lỗi). UDP dùng ít byte hơn cho phần header và yêu cầu xử lý từ host ít hơn.

TCP :

- Dùng cho mạng WAN.
- Không cho phép mất gói tin.
- Đảm bảo việc truyền dữ liệu.
- Tốc độ truyền thấp hơn UDP

UDP:

- Dùng cho mạng LAN.
- Cho phép mất dữ liệu.
- Không đảm bảo.
- Tốc độ truyền cao, VolP truyền tốt qua UDP

1.1.4. Địa chỉ IP

Địa chỉ IP (IP là viết tắt của từ tiếng Anh: Internet Protocol - giao thức Internet) là một địa chỉ đơn nhất mà những thiết bị điện tử hiện nay đang sử dụng để nhận diện và liên lạc với nhau trên mạng máy tính bằng cách sử dụng giao thức Internet.

Một cách đơn giản hơn: IP là một địa chỉ của một máy tính khi tham gia vào mạng nhằm giúp cho các máy tính có thể chuyển thông tin cho nhau một cách chính xác, tránh thất lạc. Có thể coi địa chỉ IP trong mạng máy tính giống như địa chỉ nhà của bạn để nhân viên bưu điện có thể đưa thư đúng cho bạn chứ không phải một người nào khác.

Bất kỳ thiết bị mạng nào, bao gồm bộ định tuyến, bộ chuyển mạch mạng, máy vi tính, máy chủ hạ tầng (như NTP, DNS, DHCP, SNMP, v.v.), máy in, máy fax qua Internet, và vài loại điện thoại khi tham gia vào mạng đều có địa chỉ riêng và địa chỉ này là đơn nhất trong phạm vi của một mạng cụ thể. Vài địa chỉ IP có giá trị đơn nhất trong phạm vi Internet toàn cầu, trong khi một số khác chỉ cần phải đơn nhất trong phạm vi một công ty.

Địa chỉ IP do Tổ chức cấp phát số hiệu Internet (IANA) quản lý và tạo ra. IANA nói chung phân chia những "siêu khối" đến Cơ quan Internet khu vực, rồi từ đó lại phân chia thành những khối nhỏ hơn đến nhà cung cấp dịch vụ Internet và công ty.

1.2. Lập trình mạng trong .NET FRAMEWORK

1.2.1. Cơ sở lý thuyết về .NET

a) Nền tảng của .NET

Microsoft .Net không phải là một ngôn ngữ lập trình, đó là một không gian làm việc tổng hợp bởi bốn bộ ngôn ngữ lập trình: C#, VB.NET, Managed C++, and J# .NET. ở đó có sự chồng gối lên nhau của các ngôn ngữ, và được định nghĩa trong FCL (framework class library).

Microsoft .Net bao gồm 2 phần chính: Framework và Intergrated Development Enviroment (IDE). Framework cung cấp những gì cần thiết và căn bản, là khuôn dạng hay môi trường hỗ trợ các hạ tầng cơ sở theo một quy ước nhất định để công việc được thuận tiện. IDE cung cấp một môi trường giúp chúng ta triển khai dễ dàng và được nhanh chóng các ứng dụng dựa trên nền tảng .Net.

Thành phần Framework là quan trọng nhất .NET là cốt lõi và tinh hoa của môi trường, còn IDE chỉ là công cụ để phát triển dựa trên nền tảng đó thôi. Trong .NET toàn bộ các ngôn ngữ C#, Visual C++ hay Visual Basic.NET đều dùng cùng một IDE.

Microsoft .NET là nền tảng cho việc xây dựng và thực thi các ứng dụng phân tán thế hệ kế tiếp. Bao gồm các ứng dụng từ client đến server và các dịch vụ khác. Một số tính năng của Microsoft .NET cho phép những nhà phát triển sử dụng như sau:

- Một mô hình lập trình cho phép nhà phát triển xây dựng các ứng dụng dịch vụ web và ứng dụng client với Extensible Markup Language (XML).
- Tập hợp dịch vụ XML Web, như Microsoft .NET My Services cho phép nhà phát triển đơn giản và tích hợp người dùng kinh nghiệm.
- Cung cấp các server phục vụ bao gồm: Windows 2000, SQL Server, và BizTalk Server, tất cả điều tích hợp, hoạt động, và quản lý các dịch vụ XML Web và các ứng dụng.
- Các phần mềm client như Windows XP và Windows CE giúp người phát triển phân phối sâu và thuyết phục người dùng kinh nghiệm thông qua các dòng thiết bị.
- Nhiều công cụ hỗ trợ như Visual Studio .NET, để phát triển các dịch vụ Web XML,ứng dụng trên nền Windows hay nền web một cách dễ dàng và hiệu quả.

b) Ngôn ngữ C#.

C# là một ngôn ngữ rất đơn giản, với khoảng 80 từ khoá và hơn mười kiểu dữ liệu dựng sẵn, nhưng C# có tính diễn đạt cao. C# hỗ trợ lập trình có cấu trúc, hướng đối tượng, hướng thành phần (component oriented).

Trọng tâm của ngôn ngữ hướng đối tượng là lớp. Lớp định nghĩa kiểu dữ liệu mới, cho phép mở rộng ngôn ngữ theo hướng cần giải quyết. C# có những từ khoá dành cho việc khai báo lớp, phương thức, thuộc tính (property) mới. C# hỗ trợ đầy đủ khái niệm trụ cột trong lập trình hướng đối tượng: đóng gói, thừa kế, đa hình.

Định nghĩa lớp trong C# không đòi hỏi tách rời tập tin tiêu đề với tập tin cài đặt như C++. Hơn thế, C# hỗ trợ kiểu sưu liệu mới, cho phép sưu liệu trực tiếp trong tập tin mã nguồn. Đến khi biên dịch sẽ tạo tập tin sưu liệu theo định dạng XML.

C# hỗ trợ khái niệm giao diện, interfaces (tương tự Java). Một lớp chỉ có thể kế thừa duy nhất một lớp cha nhưng có thế cài đặt nhiều giao diện.

C# có kiểu cấu trúc, struct (không giống C++). Cấu trúc là kiểu hạng nhẹ và bị giới hạn.Cấu trúc không thể thừa kế lớp hay được kế thừa nhưng có thể cài đặt giao diện.

C# cung cấp những đặc trưng lập trình hướng thành phần như property, sự kiện và dẫn hướng khai báo (được gọi là attribute). Lập trình hướng component được hỗ trợ bởi CLR thông qua siêu dữ liệu (metadata). Siêu dữ liệu mô tả các lớp bao gồm các phương thức và thuộc tính, các thông tin bảo mật.

Assembly là một tập hợp các tập tin mà theo cách nhìn của lập trình viên là các thư viện liên kết động (DLL) hay tập tin thực thi (EXE). Trong .NET một assembly là một đơn vị của việc tái sử dụng, xác định phiên bản, bảo mật, và phân phối. CLR cung cấp một số các lớp để thao tác với assembly.

C# cũng cho truy cập trực tiếp bộ nhớ dùng con trỏ kiểu C++, nhưng vùng mã đó được xem như không an toàn. CLR sẽ không thực thi việc thu dọn rác tự động các đối tượng được tham chiếu bởi con trỏ cho đến khi lập trình viên tự giải phóng.

1.2.2. Lập trình Socket

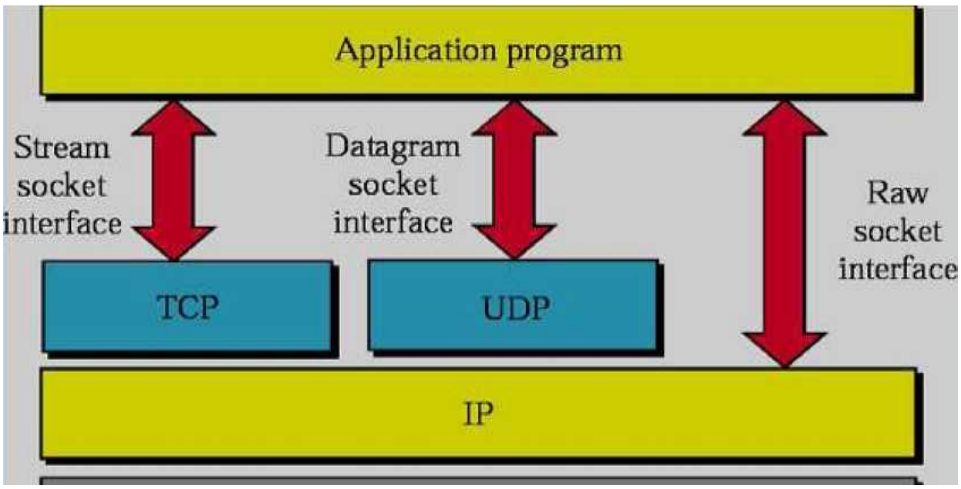
Socket là một giao diện lập trình ứng dụng (API) mạng

Thông qua giao diện này chúng ta có thể lập trình điều khiển việc truyền thông giữa hai máy sử dụng các giao thức mức thấp là TCP, UDP...

Socket là sự trừu tượng hoá ở mức cao, có thể tưởng tượng nó như là thiết bị truyền thông hai chiều gửi - nhận dữ liệu giữa hai máy tính với nhau.

* Các loại Socket

- Socket hướng kết nối (TCP Socket)
- Socket không hướng kết nối (UDP Socket)
- Raw Socket



Mô hình các loại Socket

* Số hiệu cổng của Socket

- Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng.

- Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống. Khi quá trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gởi đến cổng này từ các quá trình khác.

- Quá trình còn lại cũng yêu cầu tạo ra một socket.

a) Socket hướng kết nối (TCP Socket)

* Đặc điểm của Socket hướng kết nối

- Có 1 đường kết nối ảo giữa 2 tiến trình

+ Một trong 2 tiến trình phải đợi tiến trình kia yêu cầu kết nối.

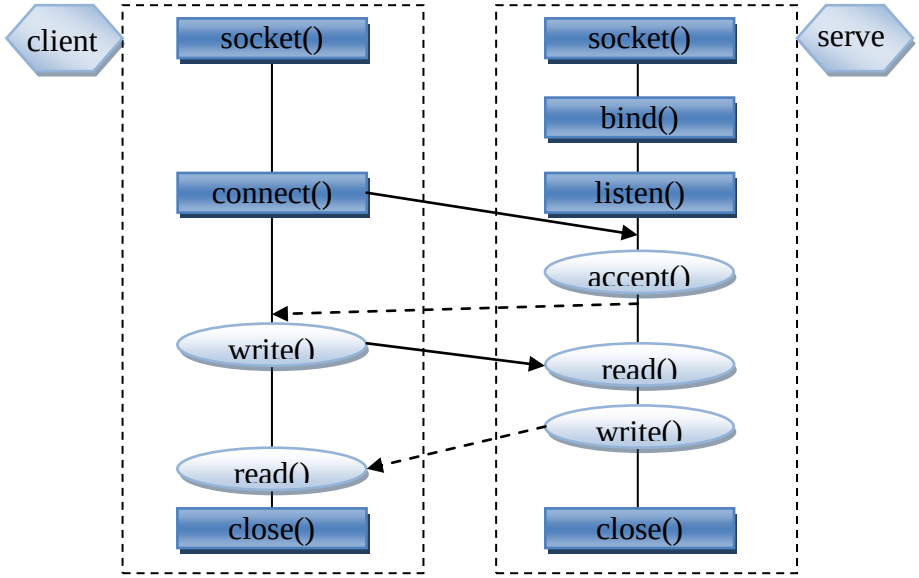
- Có thể sử dụng để liên lạc theo mô hình Client/Server.

- Trong mô hình Client/Server thì Server lắng nghe và chấp nhận một yêu cầu kết nối.

- Mỗi thông điệp gửi đều có xác nhận trở về.

- Các gói tin chuyển đi tuần tự.

* Mô hình hoạt động



Mô hình hoạt động của TCP Socket

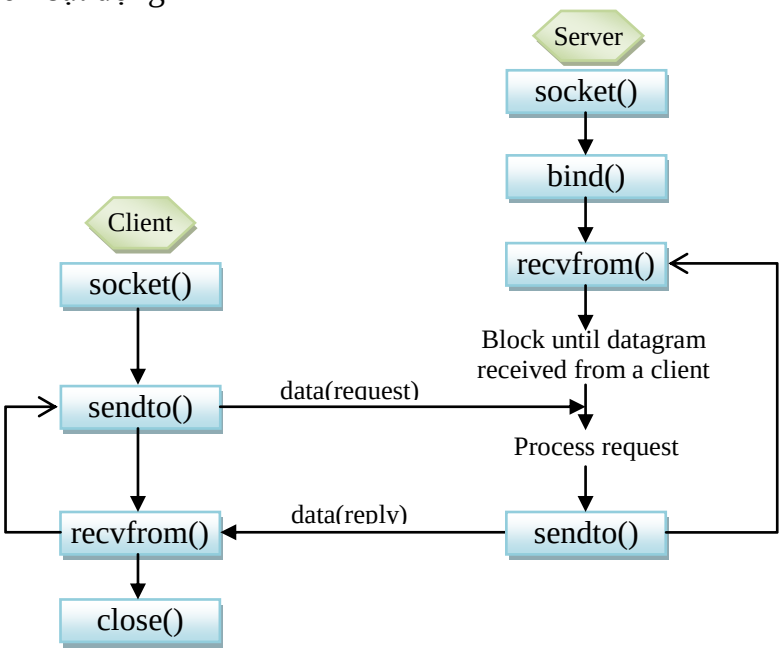
b) Socket không hướng kết nối (UDP Socket)

* Đặc điểm của Socket không hướng kết nối

- Hai tiến trình liên lạc với nhau không kết nối trực tiếp
- Thông điệp gửi đi phải kèm theo địa chỉ của người nhận
- Thông điệp có thể gửi nhiều lần
- Người gửi không chắc chắn thông điệp tới tay người nhận

+ Thông điệp gửi sau có thể đến đích trước thông điệp gửi trước đó.

* Sơ đồ hoạt động



Mô hình hoạt động của TCP Socket

1.2.3. Sử dụng các lớp hỗ trợ được xây dựng từ lớp Socket

a) Lớp TCPClient

Dùng giao thức này thì hai bên không cần phải thiết lập kết nối trước khi gửi do vậy mức dộ tin cậy không cao. Để đảm bảo độ tin cậy trong các ứng dụng mạng người ta còn sử dụng một giao thức khác gọi là giao thức có kết nối: TCP (transport control protocol). Để lập trình theo giao thức TCP, MS.NET cung cấp hai lớp có tên là TCPClient và TCPListener.

Các thành phần của lớp TcpClient

- Phương thức khởi tạo:

Tên	Mô tả
TcpClient ()	Tạo một đối tượng TcpClient. Chưa đặt thông số gì.
TcpClient (IPEndPoint)	Tạo một TcpClient và gắn 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)
TcpClient (RemoteHost: String Int32)	Tạo một đối tượng TcpClient 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.

- Một số thuộc tính:

Tên	Mô tả
Available	Cho biết số byte đã nhận về từ mạng và có sẵn để đọc
Client	Trả về socket ứng với TCPClient hiện hành
Connected	Trạng thái cho biết đã kết nối được đến server hay chưa ?

- Một số phương thức:

Tên	Mô tả
Close	Giải phóng đối tượng TcpClient nhưng không đóng kết nối
Connect (RemoteHost, Port)	Kết nối đến một máy TCP khác có tên và số hiệu cổng
GetStream	Trả về NetworkStream để từ đó giúp ta gửi hay nhận dữ liệu. (thường làm tham số khi tạo StreamReader và StreamWriter). Khi đã gắn vào StreamReader vá StreamWriter rồi thì ta có thể gửi và nhận dữ liệu thông qua các phương thức Readln, writeline tương ứng của các lớp này.

b) Lớp TCPListener

TCPListener là một lớp cho phép người lập trình có thể xây dựng các ứng dụng server.

Các thành phần của lớp TcpListent:

- Phương thức khởi tạo:

Tên	Mô tả
TcpListener (Port: Int32)	Tạo một TcpListener và lắng nghe tại cổng chỉ định
TcpListener (IPAddress, Int32)	Tạo một TcpListener và lắng nghe các kết nối đến tại địa chỉ IP và cổng chỉ định
TcpListener (IPEndPoint)	Tạo một TcpListener với giá trị EndPoint truyền vào.

- Các phương thức khác:

Tên	Mô tả
AcceptTcpClient	Chấp nhận một yêu cầu kết nối đang chờ. (ứng dụng sẽ dừng tại câu lệnh này cho đến khi nào có một kết nối đến)
AcceptSocket	Chấp nhận một yêu cầu kết nối đang chờ.
Pending	Cho biết liệu có kết nối nào đang chờ đợi không? (True = có).
Start	Bắt đầu lắng nghe các yêu cầu kết nối
Stop	Dừng việc nghe.

c) Lớp UDPClient

Giao thức UDP (user datagram protocol hay user define protocol) là một giao thức phi kết nối có nghĩa là một bên có thể gửi dữ liệu cho bên kia mà không cần biết là bên đó có sẵn sàng hay chưa? Giao thức này không tin cậy bằng giao thức TCP nhưng tốc độ của nó nhanh và dễ cài đặt. ngoài ra, với giao thức UDP ta còn có thể gửi được gói tin quảng bá đến nhiều máy.

Trong .NET, lớp UDPClient đóng gói các chức năng của giao thức UDP.

- Phương thức khởi tạo

Tên	Mô tả
UdpClient ()	Tạo một đối tượng (thể hiện) mới của lớp UDPClient.
UdpClient (AddressFamily)	Tạo một đối tượng mới của lớp UDPClient. Thuộc một dòng địa chỉ được chỉ định.
UdpClient (Int32)	Tạo một UdpClient và gắn một cổng cho nó
UdpClient (IPEndPoint)	Tạo một UdpClient và gắn một IPEndPoint cho nó
UdpClient (Int32, AddressFamily)	Tạo một UdpClient và gắn số hiệu cổng, AddressFamily
UdpClient (String, Int32)	Tạo một UdpClient và thiết lập với một máy trạm từ xa mặc định.

- Phương thức khác

Tên	Mô tả
BeginReceive	Nhận dữ liệu không đồng bộ từ máy tính từ xa.
BeginSend	Gửi không đồng bộ dữ liệu tới máy ở xa
Close	Đóng kết nối
Connect	Thiết lập một default remote host.
EndReceive	Kết thúc nhận dữ liệu không đồng bộ ở trên
EndSend	Kết thúc việc gửi dữ liệu không đồng bộ ở trên
Receive	Nhận dữ liệu (đồng bộ) do máy tính ở xa gửi
Send	Gửi dữ liệu (đồng bộ) cho máy ở xa.

1.2.4. Sử dụng Thread trong các ứng dụng mạng

a) Một số khái niệm

Đa nhiệm (multitasking): là khả năng hệ điều hành làm nhiều công việc tại một thời điểm.

Tiến trình (Process): khi chạy một ứng dụng hệ điều hành sẽ cấp phát riêng cho ứng dụng đó bộ nhớ và các tài nguyên khác. Bộ nhớ và tài nguyên vật lý riêng biệt này được gọi là một tiến trình. Các tài nguyên và bộ nhớ của một tiến trình thì chỉ tiến trình đó được phép truy cập.

Tuyến (Thread): trong hệ thống một tiến trình có thể có một hoặc nhiều chuỗi thực hiện tách biệt khác nhau và có thể chạy đồng thời. mỗi chuỗi thực hiện này được gọi là 1 tuyến (Thread). Trong 1 ứng dụng Thread khởi tạo đầu tiên gọi là Thread sơ cấp hay Thread chính.

b) Ưu nhược điểm

Ưu điểm của Thread:

Thời gian chuyển đổi giữa các luồng ít hơn hẳn so với giữa các quá trình vì không cần phải chuyển đổi không gian địa chỉ. Ngoài ra, vì chúng chia sẻ không gian địa chỉ nên các luồng trong một quá trình có thể giao tiếp với nhau dễ dàng hơn nhiều.

Trên máy tính có nhiều bộ xử lý, chương trình dạng một quá trình đơn chỉ chạy trên một CPU, còn chương trình dạng luồng có thể chia các luồng cho tất cả các bộ xử lý. Vì thế, nếu bạn chuyển chương trình dạng luồng sang máy chủ nhiều bộ xử lý thì nó sẽ chạy nhanh hơn.

Nhược điểm của Thread:

Chương trình dạng luồng khó viết và kiểm lỗi hơn. Không phải mọi thư viện lập trình đều được thiết kế để dùng với luồng và không phải mọi ứng dụng cũ đều có thể làm việc tốt với ứng dụng dạng luồng. Một vài công cụ lập trình cũng làm cho việc thiết kế và thử nghiệm mã luồng khó khăn hơn.

Lỗi liên quan đến luồng cũng khó phát hiện hơn. Các luồng trong một quá trình có thể bị chồng chéo dữ liệu với nhau. Hệ điều hành có thể hạn chế số luồng thực thi chẳng hạn đọc và ghi dữ liệu cùng lúc. Việc định thời cho các luồng khác nhau để tránh xung đột là rất khó khăn.

c) Sử dụng Thread trong chương trình .Net

Để sử dụng Thread trong .NET ta sử dụng namespace System.Threading

- Một số phương thức thường dùng:

Tên	Mô tả
Abort ()	Kết thúc Thread
Join ()	Buộc chương trình phải chờ cho thread kết thúc (Block) thì mới thực hiện tiếp (các câu lệnh đứng sau Join).
Resume ()	Tiếp tục chạy thread đã tạm ngừng – suspended
Sleep ()	Static method: tạm dừng thread trong một khoảng thời gian.
Start ()	Bắt đầu chạy (khởi động) một thread. Sau khi gọi phương thức này, trạng thái của thread chuyển từ trạng thái hiện hành sang Running.
Suspend ()	Tạm ngưng thread. (phương thức này được loại khỏi phiên bản VS.NET 2005)

- Một số thuộc tính thường dùng:

Tên	Mô tả
CurrentThread	This static property: trả về thread hiện hành đang chạy.
IsAlive	Trả về giá trị cho biết trạng thái thực thi của thread hiện hành.
IsBackground	Sets or gets giá trị cho biết là thread là background hay foreground thread.
IsThreadPoolThread	Gets a value indicating whether a thread is part of a thread pool.
Priority	Sets or gets giá trị để chỉ định độ ưu tiên (dành nhiều hay ít CPU cho thread). Cao nhất là 4, thấp nhất là 0.
ThreadState	Lấy về trạng thái của thread (đang dừng, hay đang chạy…)

CHƯƠNG 2 : KỸ THUẬT LẬP TRÌNH WINDOW SERVICES

2.1 - Tổng quan về windows service

2.1.1. Khác niệm window service

Windows service là một ứng dụng windows, không có giao diện, chạy thường trú.

Các dịch vụ Windows có thể được cấu hình tự động chạy khi hệ điều hành được khởi động hoặc có thể được cấu hình bằng tay người dùng khi cần thiết.

Windows services thường được sử dụng đối với các ứng dụng phía server, các ứng dụng đó luôn ở trạng thái sẵn sàng phục vụ các yêu cầu từ client.

Windows service là một ứng dụng chạy trên máy server hoặc workstation và cung cấp những chức năng mà sự diễn tiến của nó không cần sự tương tác trực tiếp của người dùng. Windows services thường được dùng để giám sát hoạt động hệ thống.

Một Windows service sẽ chạy trong tiến trình của riêng nó, không phụ thuộc người dùng hay các chương trình khác đang chạy trên cùng máy tính. Windows services thường được cấu hình để tự động bắt đầu khi nào máy tính khởi động. Windows services có thể bắt đầu chạy ngay cả khi không có người dùng (user) đăng nhập máy tính.

Một Windows service không bao gồm bất kỳ các yếu tố giao tiếp người dùng như *message boxes* hay *dialog boxes*. Một Windows service không có tính chất cung cấp một giao diện trực quan (visual interface) cho người dùng.

Thông thường một Windows service trao đổi thông tin với bên ngoài, báo cáo kết quả qua cơ chế thông điệp hay ghi chú sự kiện (event log). Tuy nhiên cũng có thể kết hợp nó với Web Service để chuyển thông tin từ máy mà nó đang được cài đặt đến một máy ở xa thông qua đường truyền internet.

* Phân loại

Có 2 kiểu dịch vụ window:

- Dịch vụ ứng dụng
- Điều khiển dịch vụ

2.1.2. Bộ điều khiển dịch vụ

- Bộ điều khiển dịch vụ được bắt đầu khi hệ thống khởi động. Nó là một thủ tục gọi máy chủ từ xa, để cấu hình dịch vụ và chương trình kiểm soát dịch vụ có thể thao tác dịch vụ trên máy từ xa. Bộ điều khiển dịch vụ cung cấp một giao diện cho các tác vụ sau được thực hiện:

- Duy trì cơ sở dữ liệu của các dịch vụ đã cài đặt.

- Bắt đầu dịch vụ và trình điều khiển khi hệ thống khởi động hoặc khi có nhu cầu.

- Liệt kê các dịch vụ và trình điều khiển dịch vụ đã cài

- Duy trình trạng thái thông tin cho dịch vụ và trình điều khiển dịch vụ đang chạy.

- Truyền điều khiển yêu cầu để chạy dịch vụ.

- Khóa và mở khóa dịch vụ cơ sở dữ liệu.

2.1.3. Cơ sở dữ liệu của dịch vụ đã cài đặt

Một Window Serivce được cài trong thanh nghi như là 1 đối tượng thực thi Service Control Manager (SCM) quản lý tất cả các dịch vụ . SCM điều khiển việc gọi thủ tục máy chủ , hỗ trợ máy trạm hoặc điều hành việc quản lý dịch vụ. Window service có thể được tạo bằng công cụ Visual Studio .NET . The .NET Framework cung cấp các lớp mà cho phép tạo, cài đặt và điều hành 1 cách dễ dàng.

Windows Service có 3 thành phần:

+ Service ứng dụng: Một ứng dụng bao gồm 1 hoặc nhiều dịch vụ mà cung cấp các chức năng.

+ Service controller application: Một ứng dụng cho phép có khả năng điều hành dịch vụ.

+ Service Control Manager: Một công cụ cho phép điều hành dịch vụ đã cài đặt trong máy tính

Một Window service sau khi tạo và cài đặt được đăng ký tại:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services

Dữ liệu của mỗi dịch vụ và trình điều khiển dịch vụ gồm các thông tin sau:

- Kiểu dịch vụ: Dịch vụ thực hiện trong quá trình riêng của mình hoặc chia sẻ một quá trình với các dịch vụ khác.

- Kiểu khởi động:
    + Dịch vụ được khởi động tự động khi hệ thống khởi động
    + Dịch vụ khởi động khi có yêu cầu
    + Không khởi động dịch vụ
- Mức độ kiểm soát lỗi:
- Đường dẫn đầy đủ của tập tin thực thi
- Tên đối tượng

2.1.4. Tài khoản dịch vụ

Mỗi dịch vụ thực hiện trong mức độ bảo mật của một tài khoản người dùng. Tên và mật khẩu của tài khoản được chỉ định bởi hàm CreateService(). Tên người dùng và mật khẩu có thể được thay đổi bằng cách sử dụng hàm ChangeServiceConfig(). Sử dụng hàm QueryServiceConfig() để lấy tên người dung gắn với một dịch vụ.

- LocalService.
- NetworkService.
- LocalSystem.

* Tài khoản LocalService

Các tài khoản LocalService là một tài khoản địa phương được xác định trước. Nó có đặc quyền tối thiểu trên máy tính địa phương và chứng chỉ chưa xác định trên mạng. Tên của các tài khoản trong tất cả các miền địa phương là NT AUTHORITY\LocalService. Tài khoản này không có mật khẩu

* Tài khoản NetworkService

Các tài khoản NetworkService là một tài khoản địa phương được xác định trước. Nó có đặc quyền tối thiểu trên máy tính địa phương và hoạt động như máy tính trên mạng. Tên của các tài khoản trong tất cả các miền địa phương là NT AUTHORITY\NetworkService. Tài khoản này không có mật khẩu

* LocalSystem

Các tài khoản LocalSystem là một tài khoản địa phương được xác định trước. Nó có nhiều quyền trên máy tính địa phương, và hoạt động như máy tính trên mạng. Tên của các tài khoản trong tất cả các miền địa phương là. \LocalSystem. Tên, "LocalSystem" hoặc "ComputerName/LocalSystem" cũng có thể được sử dụng. Tài khoản này không có mật khẩu

2.2 - Cấu trúc của windows service trong .NET

2.2.1. Cấu trúc tổng quát

ServiceBase : Tạo một lớp dịch vụ mới, nó được thừa kế từ SerciveBase Class. Các method của lớp có thể được ghi đè để thay đổi chức năng của họ nếu cần.

ServiceProcessInstaller: Được sử dụng để cài đặt tiến trình quản lý dịch vụ

ServiceInstaller : Được sử dụng để cài đặt dịch vụ vào hệ điều hành.

namespace của các lớp là System.ServiceProcess và System.ServiceProcess. Trong thư viện system.serviceprocess.dll.

2.2.2. Các phương thức, thuộc tính của lớp

*** Lớp ServiceBase:**

- Phương thức

	Tên	Mô tả
	CreateObjRef	Tạo ra một đối tượng có chứa tất cả thông tin có liên quan, cần thiết để tạo ra một proxy được sử dụng để giao tiếp với một đối tượng từ xa. (Kế thừa từ MarshalByRefObject.)
	Dispose()	Giải phóng tất cả tài nguyên được sử dụng bởi Component. (Kế thừa từ Component.)
	Dispose(Boolean)	Giải phóng các nguồn tài nguyên (khác với bộ nhớ) được sử dụng bởi ServiceBase. (Overrides Component.Dispose(Boolean).)
	Equals(Object)	Xác định xem các Object được chỉ định là tương đương với Object hiện tại. (Kế thừa từ Object.)
	GetHashCode	Server như một hàm băm cho một loại hình cụ thể. (Kế thừa từ Object.)
	GetLifetimeService	Trả về thời gian tồn tại của dịch vụ. (Kế thừa từ MarshalByRefObject.)
	GetService	Trả về một đối tượng mà đại diện cho một dịch vụ được cung cấp bởi các thành phần, hoặc bằng Container của nó. (Kế thừa từ Component.)
	OnContinue	Được gọi khi window service khởi động sau khi pause
	OnPause	Được gọi khi window service tạm dừng
	OnShutdown	Được gọi khi hệ thống tắt máy

	OnStart	Được gọi khi Window service khởi động
	OnStop	Được gọi khi Window service kết thúc
	RequestAdditionalTime	Yêu cầu thêm thời gian cho một hoạt động đang chờ giải quyết
	Run(ServiceBase)	Đăng ký thực thi cho một dịch vụ với Service Control Manager (SCM) .
	Run(ServiceBase[])	Đăng ký thực thi cho nhiều dịch vụ với Service Control Manager (SCM) .
	Stop	Dừng các dịnh vụ thực hiện

- Thuộc tính

	Tên	**Mô tả**
	AutoLog	Tự động báo cáo dữ liệu lênh Start, Stop, Pause, và Tiếp tục vào các bản ghi sự kiện
	CanPauseAndContinue	Có thể tạm dựng và tiếp tục dịch vụ
	CanShutdown	Lấy hoặc gán một giá trị khi dịch vụ đang tắt
	CanStop	Có thể tắt dịch vụ
	Container	Lấy thành phần trong IContainer (Kế thừ từ Component)

*** ServiceProcessInstaller**

- Phương thức

	Tên	**Mô tả**
	CreateObjRef	Tạo ra một đối tượng có chứa tất cả thông tin có liên quan, cần thiết để tạo ra một proxy được sử dụng để giao tiếp với một đối tượng từ xa. (Kế thừa từ MarshalByRefObject.)
	Dispose()	Giải phóng tất cả tài nguyên được sử dụng bởi Component. (Kế thừa từ Component.)
	Dispose(Boolean)	Giải phóng các nguồn tài nguyên (khác với bộ nhớ) được sử dụng bởi ServiceBase. (Overrides Component.Dispose(Boolean).)
	Equals(Object)	Xác định xem các Object được chỉ định là tương đương với Object hiện tại. (Kế thừa từ Object.)
	GetHashCode	Server như một hàm băm cho một loại hình cụ thể.

		(Kế thừa từ Object.)
	GetLifetimeService	Trả về thời gian tồn tại của dịch vụ. (Kế thừa từ MarshalByRefObject.)
	GetService	Trả về một đối tượng mà đại diện cho một dịch vụ được cung cấp bởi các thành phần, hoặc bằng Container của nó. (Kế thừa từ Component.)
	Install	Phương thức này được sử dụng bởi công cụ cài đặt (Ghi đè Installer.Install(IDictionary))
	IsEquivalentInstaller	Xác định nếu trình cài đặt đã chỉ rõ cài đặt các đối tượng tương như bộ cài đặt. (Kế thừa từ ComponentInstaller.)
	MemberwiseClone()	Tạo ra một bản sao của đối tượng hiện tại. (Kế thừa từ Object.)
	MemberwiseClone (Boolean)	Tạo ra một bản sao của đối tượng MarshalByRef Object hiện tại. (Kế thừa từ MarshalByRefObject.)
	OnAfterInstall	Raises the AfterInstall event (Kế thừ từ Installer.)
	OnAfterRollback	Raises the AfterRollback event (Kế thừ từ Installer.)
	OnAfterUninstall	Raises the AfterUninstall event (Kế thừ từ Installer.)
	OnBeforeInstall	Raises the BeforeInstall event (Kế thừ từ Installer.)
	OnBeforeRollback	Raises the eforeRollback event (Kế thừ từ Installer.)
	OnBeforeUninstall	Raises the BeforeUninstall event (Kế thừ từ Installer.)
	OnCommitted	Raises the Committed event (Kế thừ từ Installer.)
	OnCommitting	Raises the Committing event (Kế thừ từ Installer.)

*** ServiceInstaller**

- Phương thức

	Tên	Mô tả
	CreateObjRef	Tạo ra một đối tượng có chứa tất cả thông tin có liên quan, cần thiết để tạo ra một proxy được sử dụng để giao tiếp với một đối tượng từ xa. (Kế thừa từ MarshalByRefObject.)
	Dispose()	Giải phóng tất cả tài nguyên được sử dụng bởi Component. (Kế thừa từ Component.)
	Dispose(Boolean)	Giải phóng các nguồn tài nguyên (khác với bộ nhớ) được sử dụng bởi ServiceBase. (Overrides Component.Dispose(Boolean).)
	Equals(Object)	Xác định xem các Object được chỉ định là tương đương với Object hiện tại. (Kế thừa từ Object.)
	GetHashCode	Server như một hàm băm cho một loại hình cụ thể. (Kế thừa từ Object.)
	GetLifetimeService	Trả về thời gian tồn tại của dịch vụ. (Kế thừa từ MarshalByRefObject.)
	GetService	Trả về một đối tượng mà đại diện cho một dịch vụ được cung cấp bởi các thành phần, hoặc bằng Container của nó. (Kế thừa từ Component.)
	Install	Phương thức này được sử dụng bởi công cụ cài đặt (Ghi đè Installer.Install(IDictionary))
	IsEquivalentInstaller	Xác định nếu trình cài đặt đã chỉ rõ cài đặt các đối tượng tương như bộ cài đặt. (Kế thừa từ ComponentInstaller.)
	MemberwiseClone()	Tạo ra một bản sao của đối tượng hiện tại. (Kế thừa từ Object.)

	MemberwiseClone (Boolean)	Tạo ra một bản sao của đối tượng MarshalByRefObject hiện tại. (Kế thừa từ MarshalByRefObject.)
	OnAfterInstall	Raises the AfterInstall event
	OnAfterRollback	Raises the AfterRollback event
	OnAfterUninstall	Raises the AfterUninstall event
	OnBeforeInstall	Raises the BeforeInstall event
	OnBeforeRollback	Raises the BeforeRollback event
	OnBeforeUninstall	Raises the BeforeUninstall event
	OnCommitted	Raises the Committed event
	OnCommitting	Raises the Committing event

CHƯƠNG 3 - XÂY DỰNG CHƯƠNG TRÌNH THỰC NGHIỆM

3.1 - Mô tả chương trình thực nghiệm

- Phần Server, sẽ chạy dưới dạng Network Service với các tính năng:

  + Chạy ngầm dưới dạng service với quyền hệ thống.

  + Lắng nghe yêu cầu kết nối từ Client trên 1 cổng TCP.

  + Nhận lệnh từ Client khi kết nối được chấp thuận và thực thi lệnh dưới quyền Administrator.

  + Tạo 1 FTP server dưới dạng service để giúp truyền file cho các Client khi có yêu cầu.

  + Server có thể không cần giao diện người dùng.

- Phần Client bao gồm các chức năng:

  + Thực hiện kết nối tới Server.

  + Gửi lệnh tới Server và nhận về trạng thái lệnh từ server.

  + Thực hiện kết nối tới FTP server và thực hiện truyền/nhận file.

  + Có 2 giao diện chính: Giao diện gửi lệnh và giao diện FTP client.

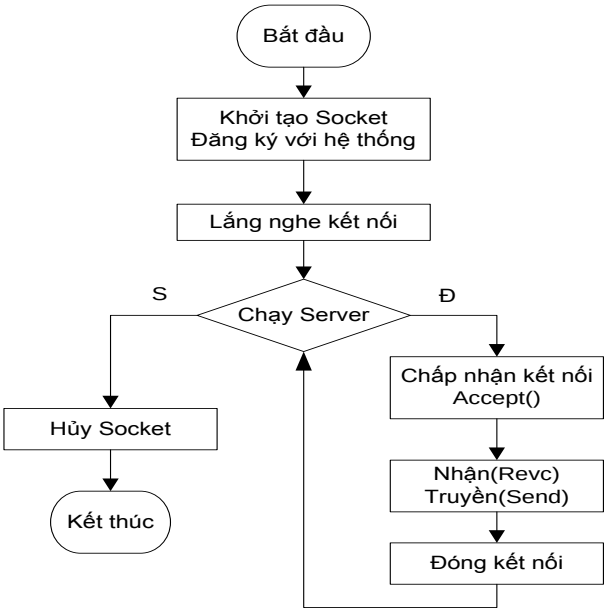
3.2 - Thiết kế chương trình

3.2.1. Server

a) Các chức năng

- *Lắng nghe kết nối*

** Lưu đồ*



**Giải thuật*

- Tạo socket , đăng kí nó với hệ thống (hàm bind).
- Đặt socket ở chế độ chờ , lắng nghe kết nối.
- Khi có request từ client , chấp nhận kết nối , tạo một process con để xử lý .
- Quay lại trạng thái chờ lắng nghe kết nối mới .
- Công việc của process mới gồm :
  + Nhận thông tin kết nối của client
  + Giao tiếp với client theo giao thức lớp ứng dụng đã thiết kế
  + Đóng kết nối và kết thúc process con

➢ *Chạy một chương trình khi Client gửi yêu cầu*

**Giải thuật*

- Input: Tên tiến trình
- Output: Thực thi tiến trình

Bước 1: Tạo đối tượng *ProcessStartInfo*

Bước 2: Gán tên tiến trình

Bước 3: Tạo mới đối tượng *Process*

Bước 4: Gán đối tượng ProcessStartInfo cho thuộc tính StartInfo của đối tượng Process

Bước 5: *Start* đối tượng *Process*

➢ *Truyền file*

- Input: Địa chỉ đầy dủ của file cần truyền
- Output: File được truyền

Bước 1: Truyền thông tin về file cần gửi (Đường dẫn đầy đủ)

Bước 2: Mở file cần truyền

Bước 3: Lặp liên tục đến khi truyền xong hoặc có yêu cần ngừng gửi:

+ Đọc ra từ file nguồn 1 mảng byte vào buffer.

+ Ghi mảng byte ấy vào socket stream.

Bước 4: Kết thúc quá trình (Đóng file)

➢ *Nhận file*

- Input: Đường dẫn lưu file

- Output: File được nhận

Bước 1: Nhận thông tin về file cần nhận từ bên truyền.

Bước 2: Tạo mở/tạo 1 file để lưu file chuẩn bị nhận

Bước 3: Dùng buffer để đọc ghi dữ liệu

Bước 4: Lặp liên tục đến khi nhận xong hoặc có yêu cần ngừng nhận:

+ Đọc từ socket stream ra 1 mảng byte vào buffer

+ Ghi mảng byte đã đọc đc vô file lưu.

Bước 5: Kết thúc quá trình (Đóng file)

b) Các lớp đối tượng sử dụng

➢ Lớp *Socket*: được sử dụng để tạo ra một sự kết nối giữa một chương trình Client and một chương trình Server .

* *Các phương thức sử dụng*

Tên	Mô tả
Accept()	Tạo Socket mới khi client kết nối đến
Bind()	Gán tên cho Socket
Listen()	Xác định số lượng kết nối chờ có thể được xếp hàng cho 1 server socket
Receive(Byte[])	Chứa số byte thực sự nhận được
Send(Byte[], Int32, SocketFlags)	Gửi dữ liệu

Lớp *ProcessStartInfo:* Sử dụng để chỉ định chi tiết cho ứng dụng cần chạy.

* *Các thuộc tính sử dụng*

Tên	Mô tả
FileName	Tên ứng dụng cần chạy.
UseShellExecute	Gọi Shell của hệ điều hành
Verb	Hành động khi chạy ứng dụng

➢ Lớp *Process:* Sử dụng để để mô tả tiến trình mới, gán đối tượng ProcessStartInfo cho thuộc tính StartInfo của đối tượng Process, và rồi khởi chạy ứng dụng bằng cách gọi Process.Start().

➢ Lớp *NetworkStream:*

Cung cấp các phương thức gửi và nhận dữ liệu dạng mảng byte.

** Các thuộc tính sử dụng*

Tên	Mô tả
Read	Đọc dữ liệu từ NetworkStream
Write	Ghi dữ liệu tới NetworkStream
Flush	Cho xoá sạch tất cả các buffer đối với writer hiện hành

➢ Lớp *FileStream*:

Được sử dụng đọc và viết dữ liệu vào hoặc từ một file

** Các thuộc tính sử dụng*

Tên	Mô tả
Read	Đọc một khối byte từ stream và ghi dữ liệu vào một buffer
Write	Ghi một khối byte tới stream bằng cách sử dụng một buffer
Flush	Cho xoá sạch tất cả các buffer đối với writer hiện hành

➢ Lớp *ServiceController:* Dùng để truy xuất danh sách các dịch vụ chạy trên máy tính.

** Các phương thức sử dụng*

Tên	Mô tả
Refresh	Làm mới lại trạng thái của dịch vụ
Start()	Khởi động dịch vụ
Stop	Dừng dịch vụ
Startus	Lấy thông tin trạng thái của dịch vụ

3.2.2. Client

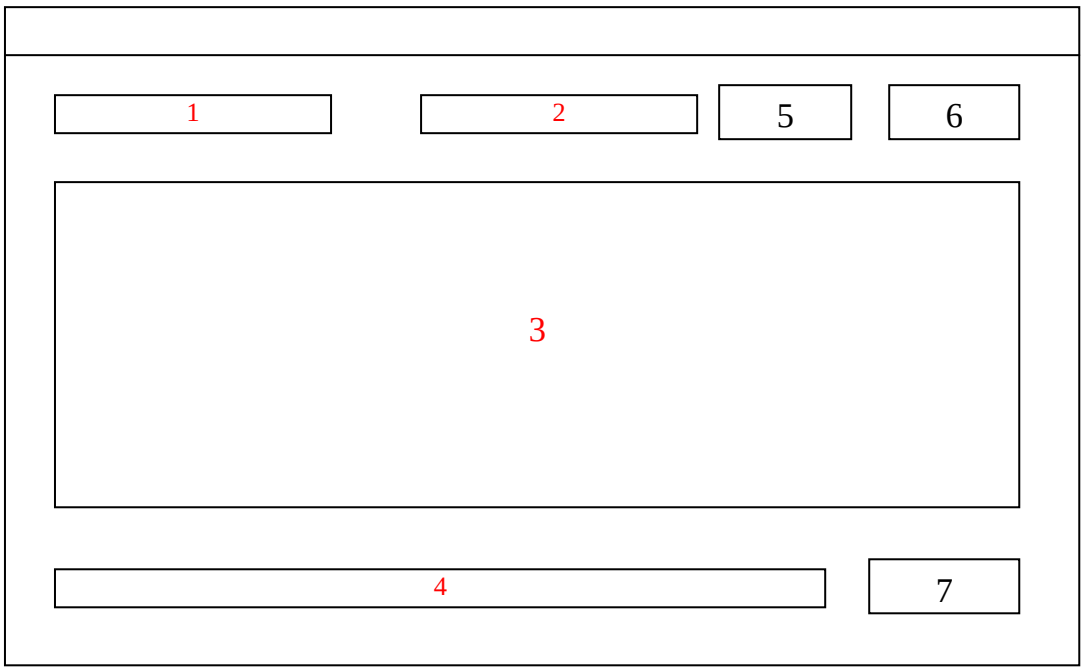
a) Các chức năng

➢ *Chức năng kết nối và gửi lệnh tới Server*

** Giải thuật*

- Xác định địa chỉ server
- Tạo socket
- Kết nối đến server
- Gửi / nhận dữ liệu theo giao thức lớp ứng dụng đã thiết kế
- Đóng kết nối

** Giao diện chính của chương trình*



(1): Textbox nhập địa chỉ IP của server

(2): Textbox nhập số cổng

(3): Textbox hiển thị thông tin thực hiện lệnh

(4): Textbox nhập lệnh

(5): Button kết nối

(6): Button ngắt kết nối

(7): Button gửi lệnh

** Quá trình hoạt động*

- Khi người dùng nhấn nút kết nối (5) để kết nối đến Server

+ Nếu địa chỉ IP, số cổng không đúng thì chương trình sẽ hiển thị thông báo lỗi.

+ Nếu kết nối thành công thì chức năng gửi lệnh sẽ được hiển thị

- Khi người dùng nhập lệnh tại ô nhập lệnh (4)

+ Khi người dùng gửi lệnh "ftp" thì chương trình sẽ gửi yêu cầu khởi động dịch vụ FTP bên server và chờ kết quả trả về của server.

- Nếu Server thông báo lại là dịch vụ được khởi động thì Client sẽ gọi chương trình FTP Client.

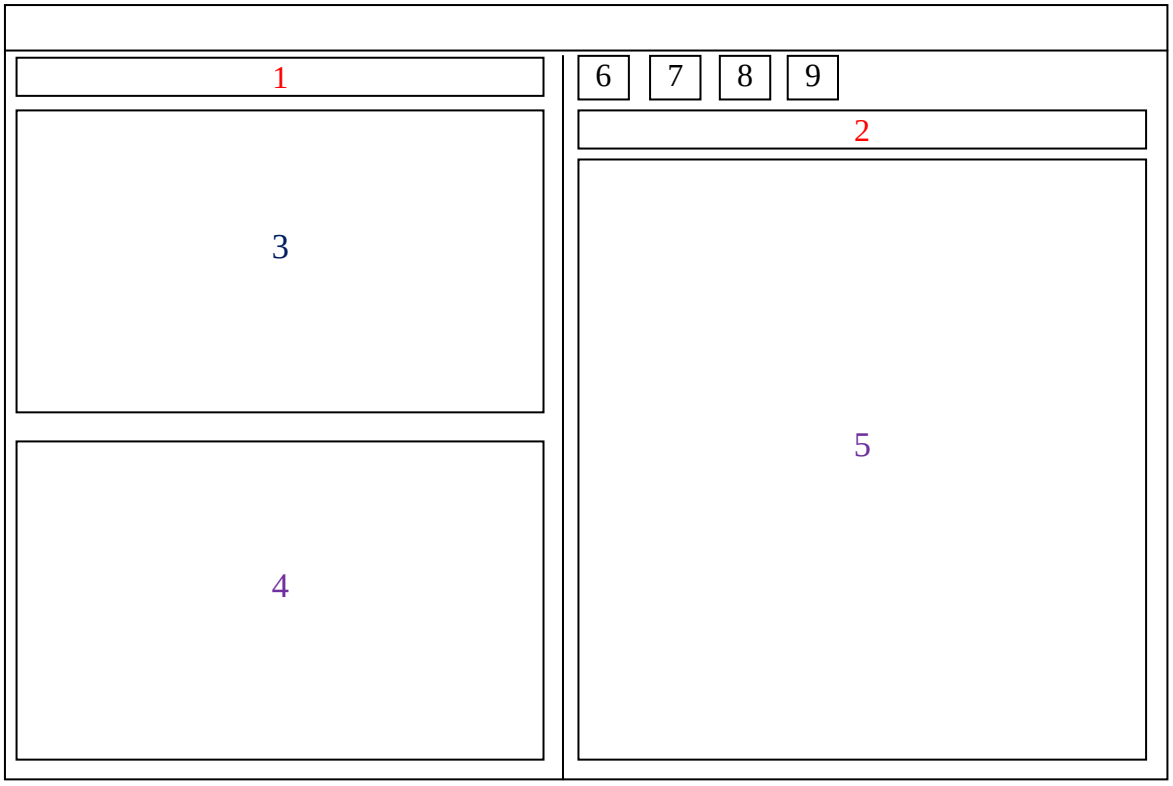
- Nếu Server thông báo lại là dịch vụ không khởi động được thì FTP Client sẽ không được hiển thi.

+ Khi người dùng gửi khác với lệnh "ftp" thì chương trình sẽ yêu cầu chạy một chương trình bên Server.

- Khi người dùng nhấn nút ngắt kết nối (6) thì chương trình sẽ gửi lệnh "quit" và yêu cầu ngắt kết nối đến Server.

➢ *Chức năng Download , Upload*

** Giao diện chương trình*



(1): Textbox hiển thị đường dẫn tệp tin

(2): Textbox hiển thị đường dẫn tệp tin bên FTP

(3): Treeview hiển thị cây thư mục trên máy Cliert

(4): Listview hiển thị danh sách file bên Client

(5): Listview hiển thị danh sách file bên FTP Server

(6): Button quay lại thư mục trước

(7): Button download

(8): Button upload

(9): Button làm mới danh sách

** Quá trình hoạt động*

- Chương trình được gọi khi có lệnh "ftp" thành công.

- Khi chương trình bắt đầu khởi động thì danh sách file bên Server gửi về sẽ hiển thị tại ô số (5).

- Khi người dùng nhấn nút Download thì chương trình kiểm tra sự tồn tại file trong ô (5) và đường dẫn ô (1).

+ Nếu đủ điều kiện thì yêu cầu tải xuống sẽ được gửi đến server. Sau khi Server nhận yêu cầu và thực thì thì file sẽ được tải về và lưu theo đường dẫn ở ô (1).

+ Nếu đường dẫn không đúng hoặc chưa chọn file cần tải về thì chương trình sẽ hiển thị thông báo cho người dùng yêu cầu kiểm tra lại.

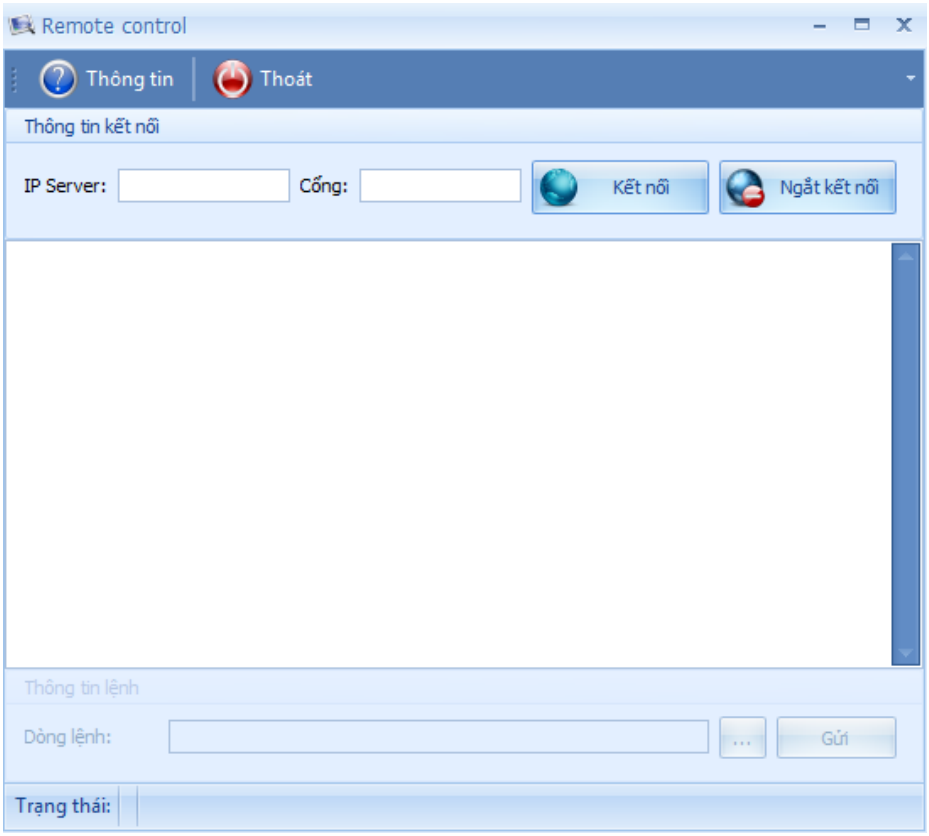
- Khi người dùng nhấn nút Upload thì chương trình sẽ kiểm tra sự tồn tại file trong ô (4) và đường dẫn ô (2).

+ Nếu đủ điều kiện thì yêu cầu tải lên sẽ được gửi đến server. Sau khi Server nhận yêu cầu và thực thì thì file sẽ được tải lên server theo đường dẫn ở ô (2).

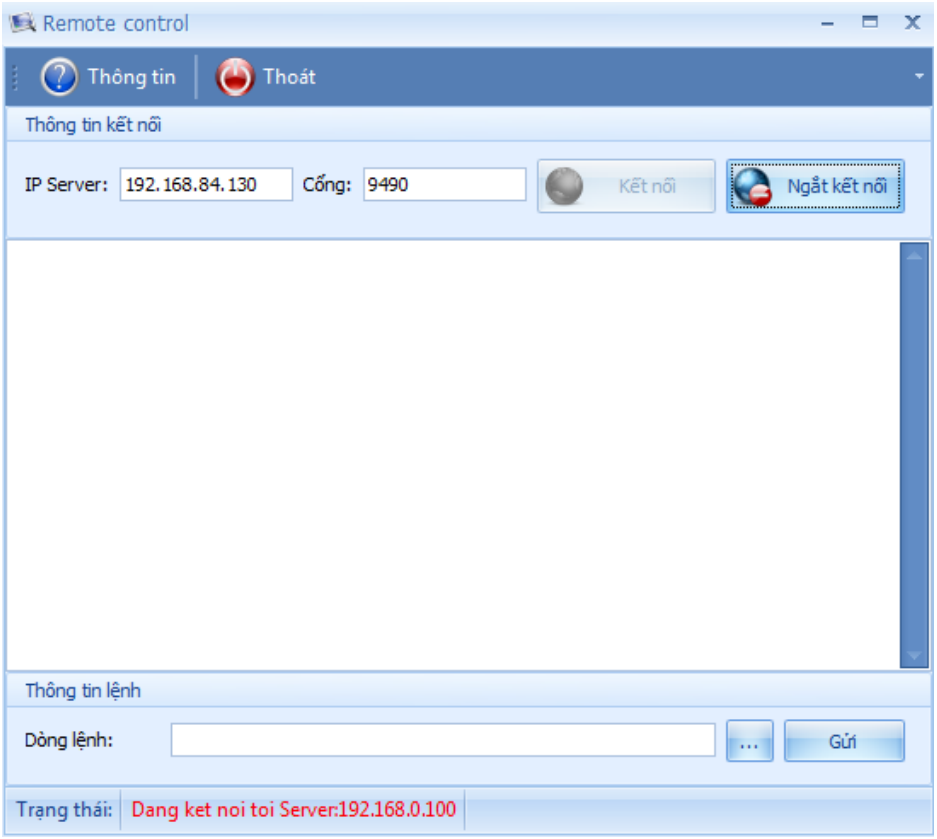
+ Nếu đường dẫn không đúng hoặc chưa chọn file cần tải về thì chương trình sẽ hiển thị thông báo cho người dùng yêu cầu kiểm tra lại.

3.3 Kết quả đạt được

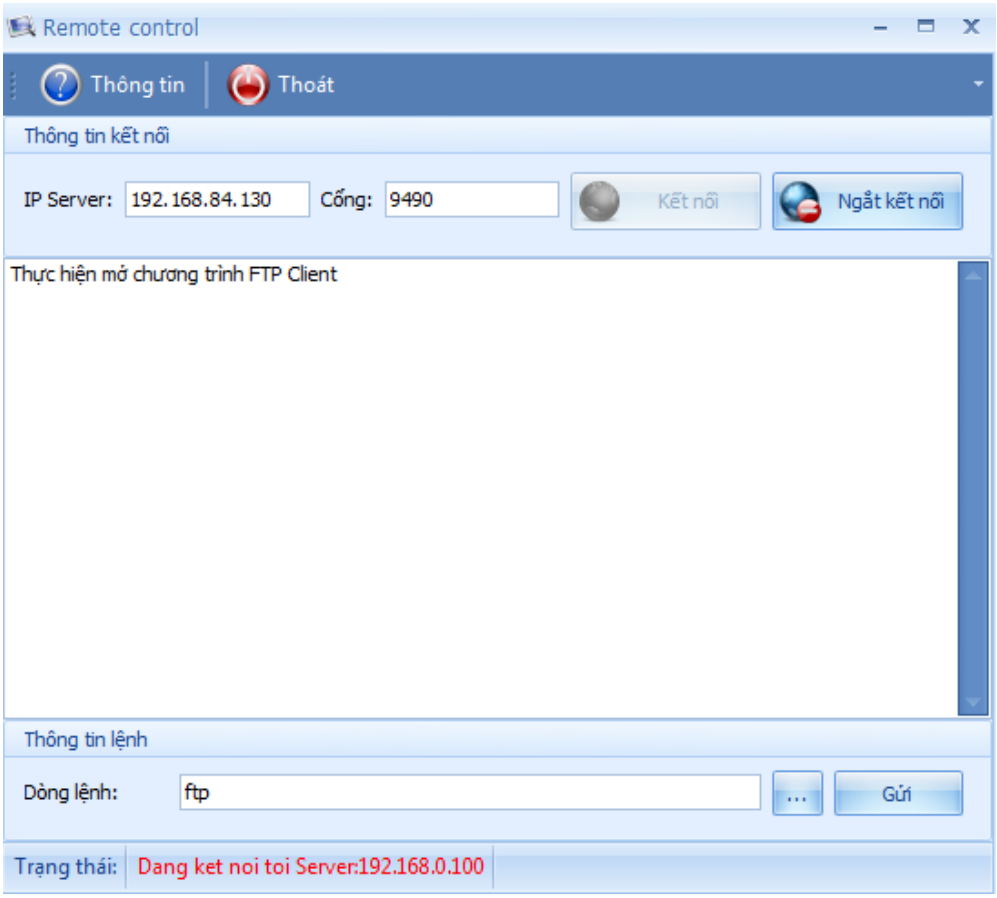
* Một số hình ảnh quá trình chạy chương trình

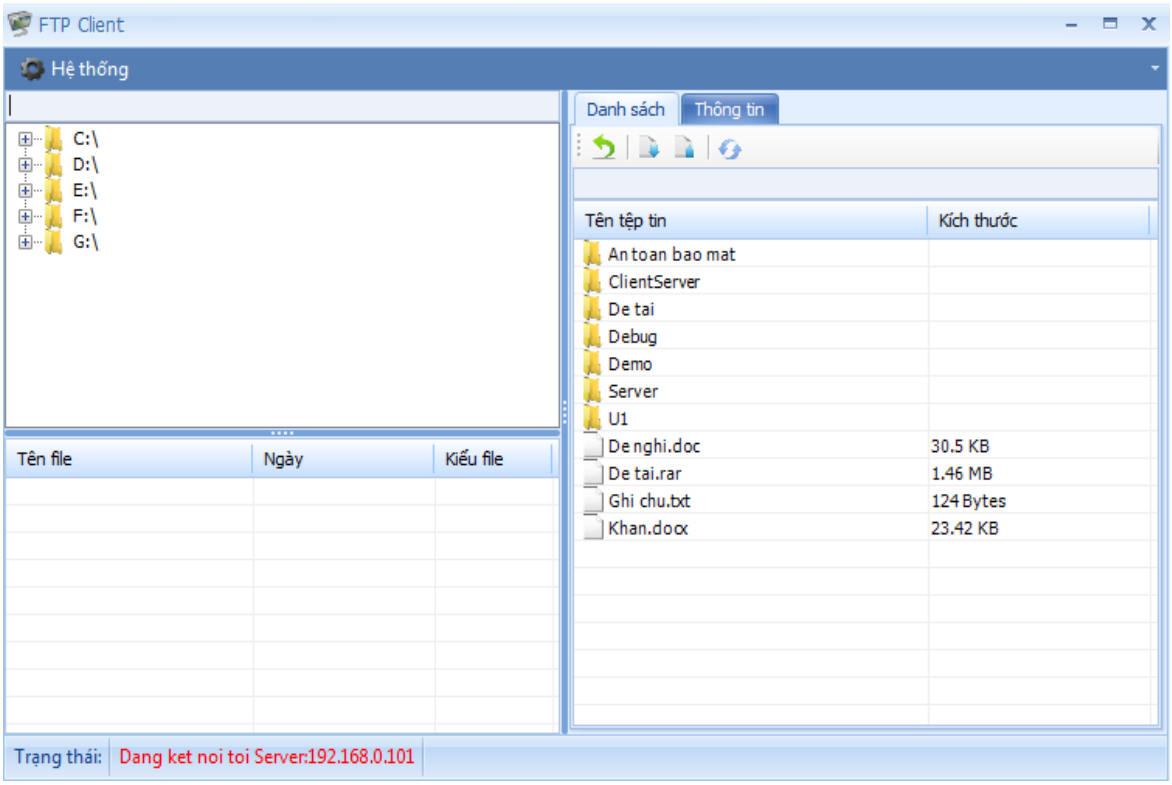


Giao diện khi chưa kết nối tới server

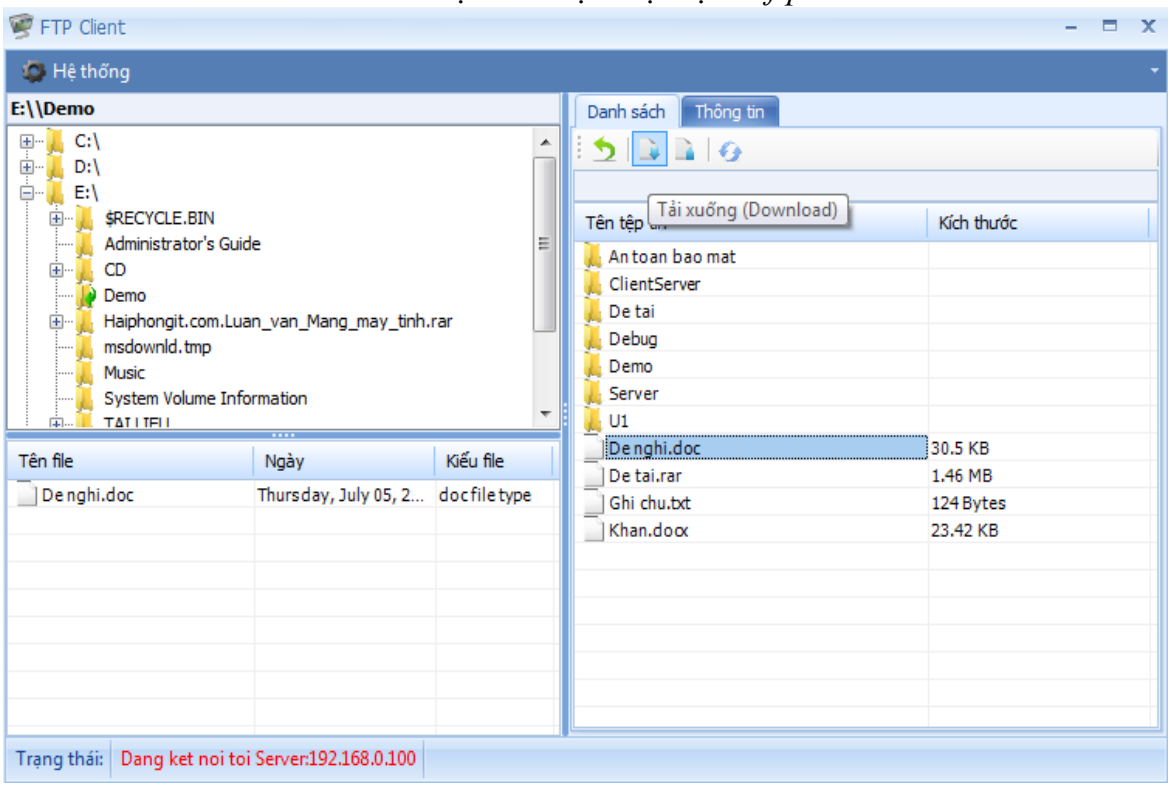


Giao diện khi nhấn nút kết nối tới server

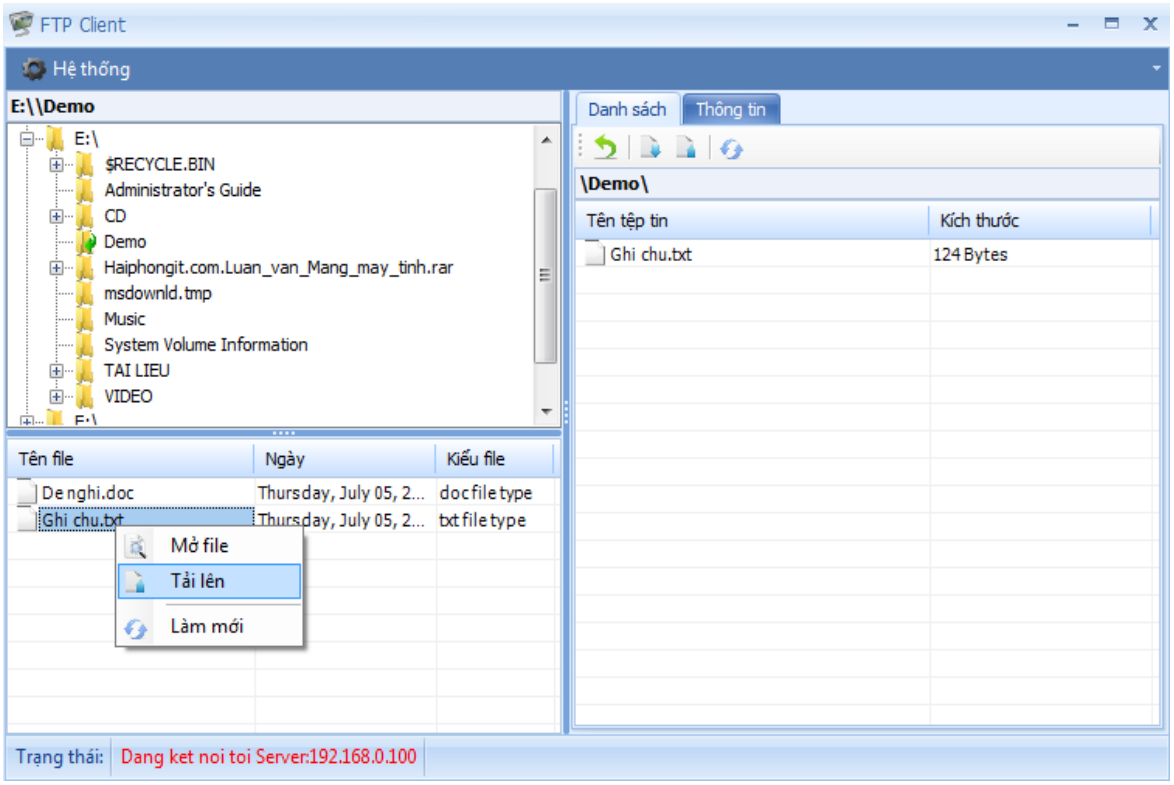




Giao diện khi thực hiện lệnh "ftp"



Giao diện khi nhấn nút Tải xuống, file được lưu theo đường dẫn: E:\Demo



Giao diện khi nhấn nút Tải lên. File "Ghi chu.txt" được tải lên thư mục \Demo của server

KẾT LUẬN

Với đề tài "**Tìm hiểu kỹ thuật lập trình Network service**". Em đã mang nhưng kiến thức được học ở nhà trường đem vận dụng vào thực tế để xây dựng bài toán này. Qua đó em có điều kiện trau dồi, nâng cao kiến thức đã học.

Với yêu cầu của bài toán đặt ra thì đồ án của em bước đầu đã đạt được một số kết quả sau:

- Đã triển khai được mô hình Client-Server ở mức đơn giản.
- Tìm hiểu được các kiểu dữ liệu, một số lớp, đối tượng trong lập trình Socket
- Tìm hiểu được một số lớp, đối tượng trong lập trình service cho windows.
- Cài đặt được ứng dụng dưới dạng dịch vụ.

Tuy nhiên, chương trình vẫn còn một số hạn chế như:

- Chương trình chạy dễ phát trình lỗi, ngoại lệ không kiểm soát hết được.

-Mới chỉ xây dựng cho một client đơn giản, chưa xây dựng được một chương trình client – server có tính chuyên nghiệp cao.

- Dịch vụ đơn gian, khó theo dõi quá trình hoạt động.

Hướng phát triển trong tương lai:

- Xây dựng một chương trình client – server hoàn thiện.
- Xây dựng chương trình có khả năng kiểm soát lỗi tốt, kiểm soát các ngoại lệ

Do kiến thức còn hạn chế nên đồ án tốt nghiệp của em chắc chắn không tránh khỏi những thiếu sót. Em rất mong có được những ý kiến đánh giá, đóng góp của các thầy cô và các bạn để nội dung đồ án thêm hoàn thiện.

TÀI LIỆU THAM KHẢO

[1]. Đề tài " Xây dựng chương trình quản lý máy tính trong mạng LAN", Tồn Thất Tư

[2]. Đồ án " Lập trình mạng nâng cao", Quốc Nhựt – Trọng Việt – Ngọc Trí, Cao đẳng Công nghệ thông tin Hữu nghị Việt – Hàn.

[3]. http://www.tenouk.com/ModuleDD.html

[4]. http://www.codeproject.com

[5]. http://www. google.com