UBND TỈNH LÂM ĐỒNG
TRƯỜNG CAO ĐẲN ĐÀ LẠT

GIÁO TRÌNH
MÔ ĐUN: LẬP TRÌNH MẠNG

NGÀNH/NGHỀ: CÔNG NGHỆ THÔNG TIN
(ỨNG DỤNG PHẦN MỀM)
TRÌNH ĐỘ: TRUNG CẤP

Ban hành kèm theo Quyết định số: /QĐ-... ngày…….tháng….năm ..... ……..... của……

LƯU HÀNH NỘI BỘ

Đà Lạt, năm 2017

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Đây là tài liệu được xây dựng theo chương trình khung chương trình đào tạo cao đẳng nghề Công nghệ thông tin (Ứng dụng phần mềm)

Trong giai đoạn viết giáo trình chúng tôi cũng đã có những sự điều chỉnh để giáo trình có tính thiết thực và phù hợp hơn với sự phát triển của lĩnh vực công nghệ thông tin.

Lập trình truyền thông là một mô đun đào tạo nghề Công nghệ thông tin (Ứng dụng phần mềm) trình độ cao đẳng nghề trong lĩnh vực Công nghệ thông tin. Ngày nay, chúng ta có thể sản xuất ra các hệ thống phần mềm lớn và hữu ích là nhờ phần lớn vào sự phát triển của lĩnh vực Lập trình truyền thông.

Trong phạm vi giáo trình Lập trình truyền thông này, chúng tôi giới thiệu sơ lược về ngôn ngữ java cho các sinh viên đã có kiến thức căn bản về Lập trình hướng đối tượng với C++. Giới thiệu những vấn đề cơ bản nhất về ngôn ngữ Java. Giới thiệu cơ chế giao tiếp liên quá trình đầu tiên là Pipe và cách sử dụng nó trong Java để làm phương tiện giao tiếp giữa các Thread trong một chương trình. Giới thiệu về cách thức xây dựng ứng dụng Client-Server trên mạng TCP/IP theo cả hai chế độ Có nối kết (TCP) và Không nối kết (UDP). Giới thiệu cách thức xây dựng các ứng dụng phân tán bằng các cơ chế gọi thủ tục từ xa (RPC - Remote Procedure Call và RMI - Remote Method Invocation).

Trong quá trình biên soạn, mặc dù đã cố gắng tham khảo nhiều tài liệu và giáo trình khác nhưng tác giả không tránh khỏi được những thiếu sót và hạn chế. Tác giả chân thành mong đợi những nhận xét, đánh giá và góp ý để cuốn giáo trình ngày một hoàn thiện hơn.

Tài liệu này được thiết kế theo từng mô đun/ môn học thuộc hệ thống mô đun/môn học của một chương trình, để đào tạo hoàn chỉnh nghề Công nghệ thông tin (Ứng dụng phần mềm) ở cấp trình độ cao đẳng nghề và được dùng làm Giáo trình cho sinh viên trong các khoá đào tạo, cũng có

thể được sử dụng cho đào tạo ngắn hạn hoặc cho đào tạo công nhân kỹ thuật, các nhà quản lý và lập trình viên tham khảo.

Đà Lạt, ngày 07 tháng 7 năm 2017

Tham gia biên soạn

1. Chủ biên Ngô Thiên Hoàng
2. Phạm Đình Nam
3. Trương Thị Thanh Thảo
4. Nguyễn Quỳnh Nguyên
5. Phan Ngọc Bảo

MỤC LỤC

CHƯƠNG TRÌNH MÔ ĐUN

Tên mô đun: LẬP TRÌNH MẠNG

Mã mô đun: MĐ38

I. Vị trí, tính chất của mô đun:

1. Vị trí: Mô đun này được bố trí giảng sau môn học Cấu trúc dữ liệu và giải thuật.

2. Tính chất: là mô đun tự chọn thuộc chuyên môn của chương trình đào tạo Cao đẳng Công nghệ thông tin (Ứng dụng phần mềm).

II. Mục tiêu mô đun:

1. Về kiến thức:
   - Trình bày nguyên lý lập trình mạng, cơ chế hoạt động của chương trình thông qua các Giao thức, hàm truy xuất.
   - Mô tả mô hình mạng, Giao thức truy cập thông qua các chương trình được cài đặt.
2. **Về** kỹ năng:
   - Sử dụng thành thạo các công cụ lập trình Windows hoặc Java để lập trình.
   - Xây dựng được các ứng dụng mạng : dịch vụ, hệ thống, dữ liệu để bảo vệ hệ thống, giám sát hệ thống, truy vấn dữ liệu….
3. Về năng lực tự chủ và trách nhiệm:
   - Có khả năng tự nghiên cứu, tự học, tham khảo tài liệu liên quan đến môn học để vận dụng vào hoạt động hoc tập.
   - Vận dụng được các kiến thức tự nghiên cứu, học tập và kiến thức, kỹ năng đã được học để hoàn thiện các kỹ năng liên quan đến môn học một cách khoa học, đúng quy định.

III. Nội dung mô đun

BÀI 1 CÁC KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH MẠNG

Mã bài : MĐ 38.1

Mục tiêu:

- Phân loại hệ thống tin học: tập trung và không tập trung (hệ thống phân tán)
- Trình bày các vấn đề về điều hành mạng : Vấn đề trỏ thông tin, Giao thức, các tầng liên lạc và tính phân cấp của các giao thức, thông điệp.
- Trình bày được các kiểu truyền thông : đối thoại thông qua mạng truyền thông, hội thảo giữa các đối tác, truyền thông tin.
- Hệ thống được các qui ước.
- Thực hiện các thao tác an toàn với máy tính.

1.1. Cơ chế giao tiếp liên quá trình là gì ?

Truyền thông là một khái niệm dùng để chỉ sự giao tiếp, trao đổi thông tin giữa hai hay nhiều thực thể trong một hệ thống nào đó. Nếu hệ thống mà chúng ta xem xét là xã hội loài người, thì truyền thông có thể là quá trình trao đổi thông tin giữa người với người trong cuộc sống thông qua các phương tiện truyền tải thông tin khác nhau như không khí (trong trò chuyện trực tiếp), hệ thống điện thoại, sách, báo, các phương tiện nghe nhìn, mạng máy tính...

Nếu hệ thống mà chúng ta xem xét là một hệ thống máy tính hay một hệ thống mạng thì truyền thông có thể được phân thành hai mức:

- Mức phần cứng: là sự giao tiếp, trao đổi thông tin giữa các bộ phận vật lý cấu thành nên hệ thống máy tính như CPU, bộ nhớ, thiết bị vào ra, card giao tiếp mạng, nhờ vào các phương tiện truyền thông như hệ thống BUS nội, hệ thống BUS vào ra hay các dây cáp mạng . . .
- Mức phần mềm: là sự giao tiếp, trao đổi thông tin giữa các thành phần bên trong của một chương trình hay giữa các chương trình với nhau thông qua các cơ chế truyền thông được hỗ trợ bởi các hệ điều hành, hệ điều hành mạng.

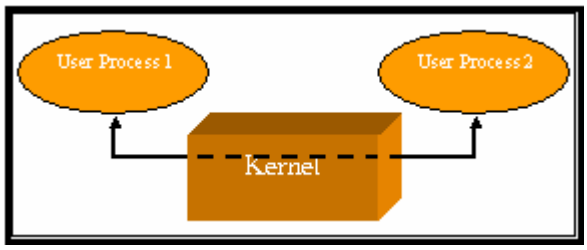
Trong các hệ thống máy tính đơn nhiệm (monotasking) cổ điển, ví dụ MS-DOS, tại một thời điểm chỉ cho phép tồn tại một quá trình. Việc giao tiếp, trao đổi thông tin chỉ diễn ra trong phạm vi của một chương trình. Đó là sự giao tiếp giữa các thủ tục dưới hình thức chia sẻ các biến toàn cục, hay bằng cách truyền các tham số khi gọi hàm, thủ tục hay bằng giá trị trả về của một hàm . . . Ngược lại, trong các hệ thống đa nhiệm

(multitasking) có nhiều quá trình tồn tại song song nhau, mỗi quá trình được thực hiện trong một không gian địa chỉ (Address space) riêng biệt. Việc giao tiếp giữa các quá trình muốn thực hiện được đòi hỏi phải có những tiện ích hỗ trợ bởi hệ điều hành, hệ điều hành mạng. Các tiện ích này thường được gọi với cái tên là **Cơ chế giao tiếp liên quá trình** (IPC - Inter-Process Communication).

1.2. Phân loại cơ chế giao tiếp liên quá trình

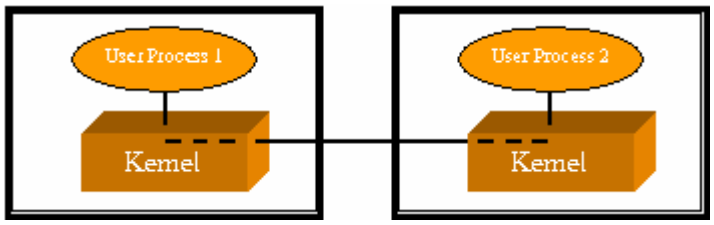
Các cơ chế giao tiếp liên quá trình được hỗ trợ bởi các hệ điều hành đa nhiệm, hệ điều hành mạng được chia ra làm hai loại:

- Loại 1: Cơ chế giao tiếp liên quá trình hỗ trợ giao tiếp giữa các quá trình trên cùng một máy tính. (Hình H1.1)



Hình 1.1 - Cơ chế giao tiếp liên quá trình loại 1

- Loại 2: Cơ chế giao tiếp liên quá trình hỗ trợ giao tiếp giữa các quá trình nằm trên các máy tính khác nhau (Hình H1.2).



Hình 1.2 - Cơ chế giao tiếp liên quá trình loại 2

Trong cơ chế giao tiếp liên quá trình trên cùng một máy, dữ liệu trao đổi qua lại giữa các quá trình phải đi xuyên qua hạt nhân (kernel) của hệ điều hành. Đó có thể là một vùng nhớ dùng chung cho các quá trình đã được qui định trước bởi hệ điều hành, hay một tập tin trên đĩa được quản lý bởi hệ điều hành trong đó một quá trình sẽ ghi dữ liệu vào, quá trình khác đọc dữ liệu ra, . . .

Trong cơ chế giao tiếp liên quá trình trên các máy tính khác nhau, dữ liệu trao đổi giữa các quá trình không những phải đi qua hạt nhân như cơ chế giao tiếp liên quá trình trên một máy tính mà hơn thế các hạt nhân của các máy có liên quan phải hiểu nhau. Nói cách khác các hạt nhân phải thoả thuận trước với nhau về các qui tắc trao đổi thông tin giữa chúng. Thông thường ta gọi các qui tắc này là các giao thức (Protocol).

1.3. Mô hình tham khảo OSI

Để dễ dàng cho việc nối kết và trao đổi thông tin giữa các máy tính với nhau, vào năm 1983, Tổ chức tiêu chuẩn thế giới ISO đã phát triển một mô hình cho phép hai máy tính có thể gởi và nhận dữ liệu cho nhau. Mô hình này dựa trên tiếp cận phân tầng (lớp), với mỗi tầng đảm nhiệm một số các chức năng cơ bản nào đó và được gọi là mô hình OSI.

Để hai máy tính có thể trao đổi thông tin được với nhau cần có rất nhiều vấn đề liên quan. Ví dụ như cần có Card mạng, dây cáp mạng, điện thế tín hiệu trên cáp mạng, cách thức đóng gói dữ liệu, điều khiển lỗi đường truyền ... Bằng cách phân chia các chức năng này vào những tầng riêng biệt nhau, việc viết các phần mềm để thực hiện chúng trở nên dễ dàng hơn. Mô hình OSI giúp đồng nhất các hệ thống máy tính khác biệt nhau khi chúng trao đổi thông tin. Mô hình này gồm có 7 tầng:

1. Tầng vật ký (Physical Layer)

Điều khiển việc truyền tải thật sự các bit trên đường truyền vật lý. Nó định nghĩa các tín hiệu điện, trạng thái đường truyền, phương pháp mã hóa dữ liệu, các loại đầu nối được sử dụng.

2. Tầng liên kết dữ liệu (Data-Link Layer)

Tầng này đảm bảo truyền tải các khung dữ liệu (Frame) giữa hai máy tính có đường truyền vật lý nối trực tiếp với nhau. Nó cài đặt cơ chế phát hiện và xử lý lỗi dữ liệu nhận.

3. Tầng mạng (Network Layer)

Tầng này đảm bảo các gói dữ liệu (Packet) có thể truyền từ máy tính này đến máy tính kia cho dù không có đường truyền vật lý trực tiếp giữa chúng. Nó nhận nhiệm vụ tìm đường đi cho dữ liệu đến các đích khác nhau trong hệ thống mạng.

4. Tầng vận chuyển (Transport Layer)

Tầng này đảm bảo truyền tải dữ liệu giữa các quá trình. Dữ liệu gởi đi được đảm bảo không có lỗi, theo đúng trình tự, không bị mất mát, trùng lắp. Đối với các gói tin có kích thước lớn, tầng này sẽ phân chia chúng thành các phần nhỏ trước khi gởi đi, cũng như tập hợp lại chúng khi nhận được.

5. Tầng giao dịch (Session Layer)

Tầng này cho phép các ứng dụng thiết lập, sử dụng và xóa các kênh giao tiếp giữa chúng (được gọi là giao dịch). Nó cung cấp cơ chế cho việc nhận biết tên và các chức năng về bảo mật thông tin khi truyền qua mạng.

6. Tầng trình bày (Presentation Layer)

Tầng này đảm bảo các máy tính có kiểu định dạng dữ liệu khác nhau vẫn có thể trao đổi thông tin cho nhau. Thông thường các máy tính sẽ thống nhất với nhau về một kiểu định dạng dữ liệu trung gian để trao đổi thông tin giữa các máy tính. Một dữ liệu cần gởi đi sẽ được tầng trình bày chuyển sang định dạng trung gian trước khi nó được truyền lên mạng. Ngược lại, khi nhận dữ liệu từ mạng, tầng trình bày sẽ chuyển dữ liệu sang định dạng riêng của nó.

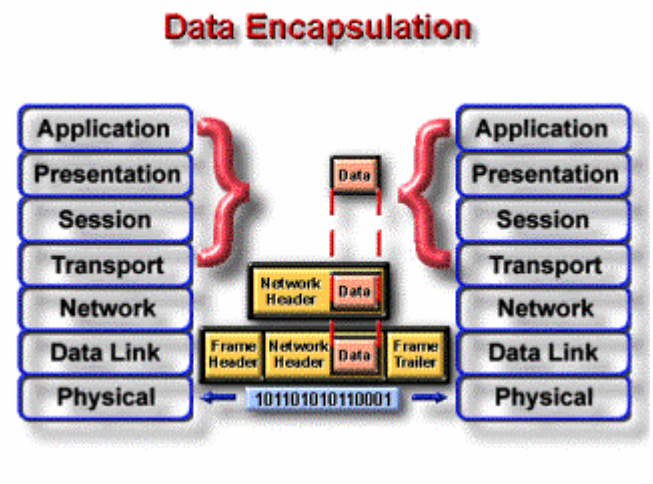
7. Tầng ứng dụng (Application Layer)

Đây là tầng trên cùng, cung cấp các ứng dụng truy xuất đến các dịch vụ mạng. Nó bao gồm các ứng dụng của người dùng, ví dụ như các Web Browser (Netscape Navigator, Internet Explorer), các Mail User Agent (Outlook Express, Netscape Messenger, ...) hay các chương trình làm server cung cấp các dịch vụ mạng như các Web Server (Netscape Enterprise, Internet Information Service, Apache, ...), Các FTP Server, các Mail server (Send mail, MDeamon). Người dùng mạng giao tiếp trực tiếp với tầng này.

Về nguyên tắc, tầng n của một hệ thống chỉ giao tiếp, trao đổi thông tin với tầng n của hệ thống khác. Mỗi tầng sẽ có các đơn vị truyền dữ liệu riêng:

- Tầng vật lý: bit
- Tầng liên kết dữ liệu: Frame
- Tầng mạng: Packet
- Tầng vận chuyển: Segment

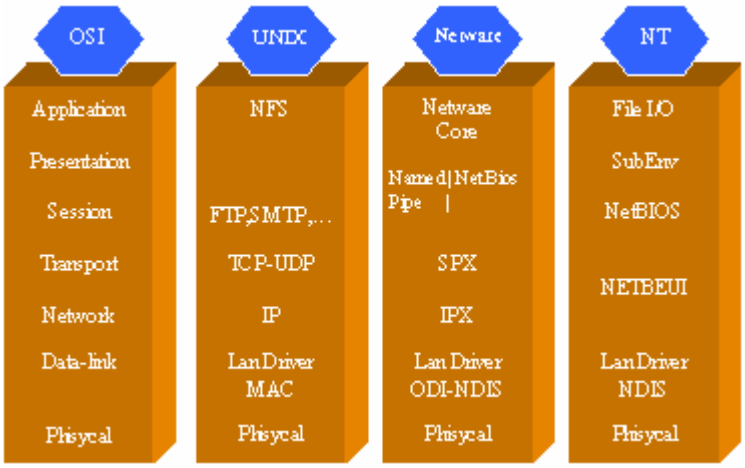
Trong thực tế, dữ liệu được gởi đi từ tầng trên xuống tầng dưới cho đến tầng thấp nhất của máy tính gởi. Ở đó, dữ liệu sẽ được truyền đi trên đường truyền vật lý. Mỗi khi dữ liệu được truyền xuống tầng phía dưới thì nó bị "gói" lại trong đơn vị dữ liệu của tầng dưới. Tại bên nhận, dữ liệu sẽ được truyền ngược lên các tầng cao dần. Mỗi lần qua một tầng, đơn vị dữ liệu tương ứng sẽ được "tháo" ra.



Đơn vị dữ liệu của mỗi tầng sẽ có một tiêu đề (header) riêng, được mô tả trong hình 1.3

Hình 1.3 - Xử lý dữ liệu qua các tầng

OSI chỉ là mô hình tham khảo, mỗi nhà sản xuất khi phát minh ra hệ thống mạng của mình sẽ thực hiện các chức năng ở từng tầng theo những cách thức riêng. Các cách thức này thường được mô tả dưới dạng các chuẩn mạng hay các giao thức mạng. Như vậy dẫn đến trường hợp cùng một chức năng nhưng hai hệ thống mạng khác nhau sẽ không tương tác được với nhau. Hình 1.4 sẽ so sánh kiến trúc của các hệ điều hành mạng thông dụng với mô hình OSI.



Hình 1.4 - Kiến trúc của một số hệ điều hành mạng thông dụng

Để thực hiện các chức năng ở tầng 3 và tầng 4 trong mô hình OSI, mỗi hệ thống mạng sẽ có các protocol riêng:

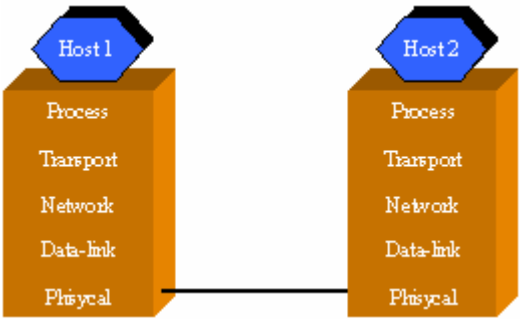
- UNIX: Tầng 3 dùng giao thức IP, tầng 4 giao thức TCP/UDP
- Netware: Tầng 3 dùng giao thức IPX, tầng 4 giao thức SPX

- Windows NT: chỉ dùng 1 giao thức NETBEUI

Nếu chỉ dừng lại ở đây thì các máy tính UNIX, Netware, NT sẽ không trao đổi thông tin được với nhau. Với sự lớn mạnh của mạng Internet, các máy tính cài đặt các hệ điều hành khác nhau đòi hỏi phải giao tiếp được với nhau, tức phải sử dụng chung một giao thức. Đó chính là bộ giao thức TCP/IP, giao thức của mạng Internet.

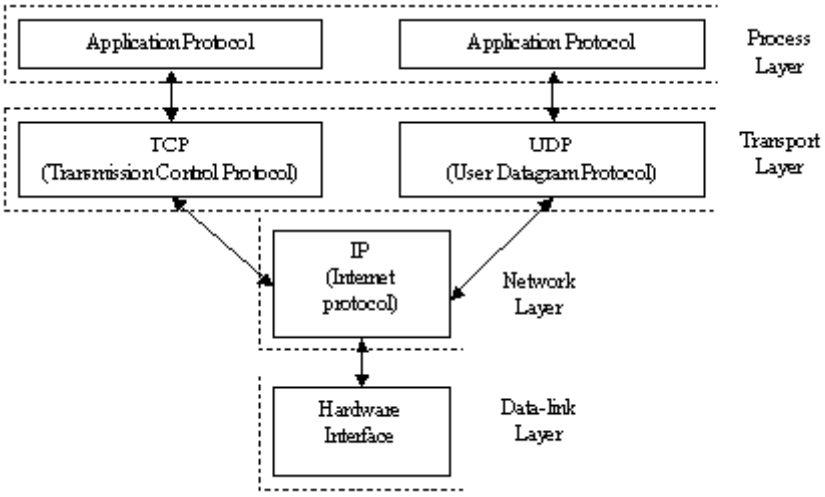
1.4. Mạng TCP/IP

Đây là kiến trúc của mạng Internet, chỉ gồm 5 tầng như hình vẽ dưới đây:



Hình 1.5 - Kiến trúc mạng TCP/IP

Người ta còn gọi mô hình này là mô hình OSI đơn giản. Các giao thức được sử dụng trên mỗi tầng được qui định như sau:



Hình 1.6 – Bộ giao thức TCP/IP

Tầng 3 sử dụng giao thức IP, tầng 4 có thể sử dụng giao thức TCP ở chế độ có nối kết hoặc UPD ở chế độ không nối kết.

Tầng 5 là tầng của các ứng dụng. Mỗi loại ứng dụng phải định nghĩa một giao thức riêng để các thành phần trong ứng dụng trao đổi thông tin qua lại với nhau. Một số ứng dụng phổ biến đã trở thành chuẩn của mạng Internet như:

- Ứng dụng Web: Sử dụng giao thức HTTP để tải các trang web từ Web Server về Web Browser.
- Ứng dụng thư điện tử: Sử dụng giao thức SMTP để chuyển tiếp mail gởi đi đến Mail Server của người nhận và dùng giao thức POP3 hoặc IMAP để nhận mail về cho người đọc.
- Ứng dụng truyền tải tập tin: Sử dụng giao thức FTP để tải (download) các tập tin từ các FTP Server ở xa về máy người dùng hay ngược lại.
- . . . . .

Thông thường các tầng 1,2,3 và 4 được phát triển bởi các nhà sản xuất hệ điều hành, nhà sản xuất các thiết bị phần cứng mạng. Chúng đảm nhận nhiệm vụ truyền tải thông tin cho các quá trình trên tầng ứng dụng. Chúng cài đặt các cơ chế giao tiếp liên quá trình để các quá trình trên tầng ứng dụng có thể truy xuất đến dịch vụ truyền tải thông tin do chúng cung cấp. Trong khi đó, tầng 5 là nơi các nhà sản xuất phần mềm khai thác để tạo ra các ứng dụng giải quyết các vấn đề khác nhau của cuộc sống. Nó được xem như là tầng xử lý thông tin.

1.5. Dịch vụ mạng

Dịch vụ mạng (Net service) là một chương trình ứng dụng thực hiện một tác vụ nào đó trên hệ thống mạng.

Ví dụ:

- Dịch vụ in trên mạng cho phép nhiều máy tính cùng sử dụng một máy in.
- Dịch vụ tập tin trên mạng cho phép chia sẻ chương trình, dữ liệu giữa các máy tính.
- Dịch vụ web cung cấp các trang web cho các máy tính khác nhau

Có nhiều mô hình khác nhau để xây dựng các dịch vụ mạng. Một trong những mô hình được sử dụng khá phổ biến là mô hình Client-Server. Đây là một mô hình cơ bản để xây dựng các dịch vụ mạng.

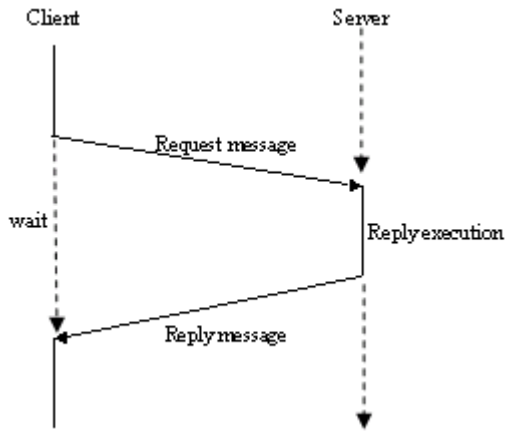
1.6. Mô hình Client – Server

1.6.1. Giới thiệu

Trong mô hình này, chương trình ứng dụng được chia thành 2 thành phần:

- Quá trình chuyên cung cấp một số phục vụ nào đó, chẳng hạn: phục vụ tập tin, phục vụ máy in, phục vụ thư điện tử, phục vụ Web... Các quá trình này được gọi là các trình phục vụ hay **Server**.
- Một số quá trình khác có yêu cầu sử dụng các dịch vụ do các server cung cấp được gọi là các quá trình khách hàng hay **Client**.

Việc giao tiếp giữa client và server được thực hiện dưới hình thức trao đổi các thông điệp (Message). Để được phục vụ, client sẽ gởi một thông điệp yêu cầu (Request Message) mô tả về công việc muốn server thực hiện. Khi nhận được thông điệp yêu cầu, server tiến hành phân tích để xác định công việc cần phải thực thi. Nếu việc thực hiện yêu cầu này có sinh ra kết quả trả về, server sẽ gởi nó cho client trong một thông điệp trả lời (Reply Message). Dạng thức (format) và ý nghĩa của các thông điệp trao đổi giữa client và server được qui định rõ bởi giao thức (protocol) của ứng dụng.



Hình 1.7 – Mô hình Client-Server

———— Quá trình đang được thực thi

…… … Quá trình đang bị nghẽn

1.6.2. Ví dụ về dịch vụ Web.

Dịch vụ web được tổ chức theo mô hình Client -Server, trong đó:

- Web server sẵn sàng cung cấp các trang web đang được lưu trữ trên đĩa cứng cục bộ của mình.

- Web Client, còn gọi là các Browser, có nhu cầu nhận các trang web từ cácWeb Server
- HTTP là giao thức trao đổi thông tin qua lại giữa Web client và Web Server.
- Thông điệp yêu cầu là một chuỗi có dạng sau:

  Command URL HTTP/Ver \n\n
- Thông điệp trả lời có dạng sau:

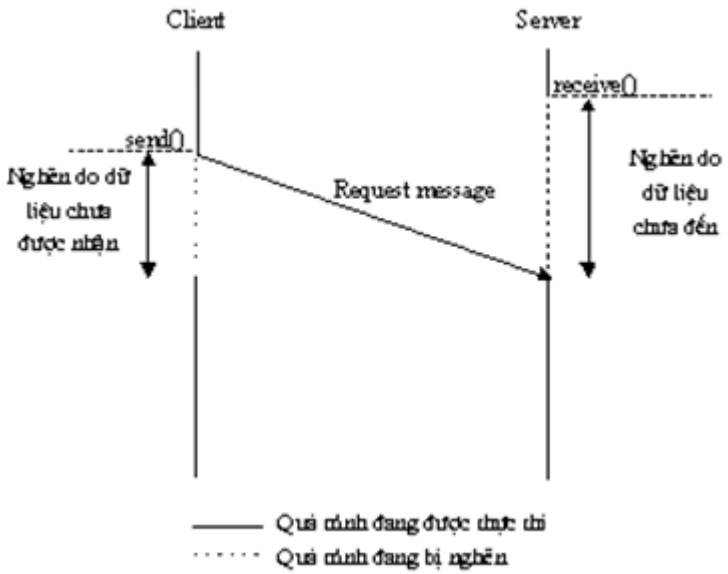
  <HEADER>\n\n

  <CONTENT>

1.6.3 Các chế độ giao tiếp

Quá trình giao tiếp giữa client và server có thể diễn ra theo hai chế độ là nghẽn blocked) hay không nghẽn (Non blocked).

1.6.3.1. Chế độ nghẽn :



Hình 1.8 - Chế độ giao tiếp nghẽn

Trong chế độ này, khi quá trình client hay server phát ra lệnh gởi dữ liệu, (thông thường bằng lệnh send) , sự thực thi của nó sẽ bị tạm dừng cho đến khi quá trình nhận phát ra lệnh nhận số dữ liệu đó (thường là lệnh receive).

Tương tự cho trường hợp nhận dữ liệu, nếu quá trình nào đó, client hay server, phát ra lệnh nhận dữ liệu, mà ở thời điểm đó chưa có dữ liệu gởi đến, sự thực thi của nó cũng tạm dừng cho đến khi có dữ liệu gởi đến

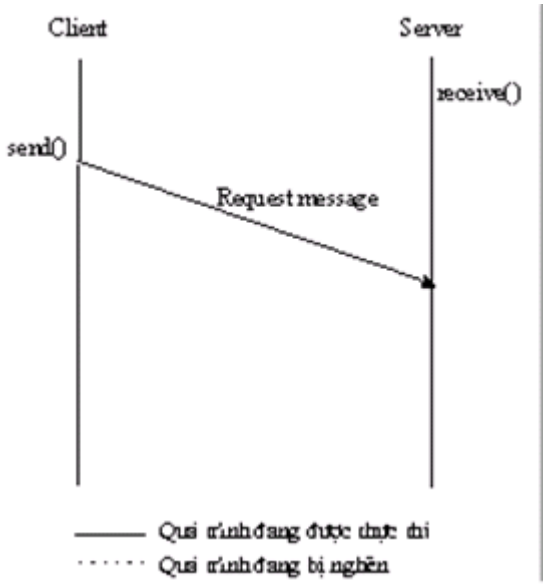
1.6.3.2. Chế độ không nghẽn:

Trong chế độ này, khi quá trình client hay server phát ra lệnh gởi dữ liệu, sự thực thi của nó vẫn được tiếp tục mà không quan tâm đến việc có quá trình nào phát ra lệnh nhận số dữ liệu đó hay không.

Tượng tự cho trường hợp nhận dữ liệu, khi quá trình phát ra lệnh nhận dữ liệu, nó sẽ nhận được số lượng dữ liệu hiện có (bằng 0 nếu chưa có quá trình nào gởi dữ liệu đến). Sự thực thi của quá trình vẫn được tiếp tục.

Trong thực tế cần chú ý đến chế độ giao tiếp nghẽn khi lập trình, vì nó có thể

dẫn đến trường hợp chương trình bị "treo" do số lần gởi và nhận giữ liệu không bằng nhau giữa hai bên giao tiếp.



Hình 1.9 - Chế độ giao tiếp không nghẽn

1.7. Các kiểu kiến trúc chương trình

Ở mức luận lý, các chức năng mà một chương trình ứng dụng thực hiện có thể xếp thành một trong 3 loại sau:

1. Các chức năng thực hiện việc tương tác với người dùng như tạo các giao diện nhập liệu hay in các báo biểu, thông báo ra màn hình. Các chức năng này được gọi chung là **Dịch vụ giao diện người dùng** (User Interface Service).

2. Các chức năng tính toán các dữ liệu, xử lý thông tin theo những qui luật (rule), giải thuật được qui định bởi vấn đề mà ứng dụng giải quyết. Các chức năng này được gọi chung là **Dịch vụ nghiệp vụ** (Business Rule Service).

3. Trong quá trình tính toán, chương trình ứng dụng cần truy vấn đến các thông

tin đã có được lưu trên đĩa cứng hay trong các cơ sỡ dữ liệu. Cũng như cần thiết phải lưu lại các kết quả tính toán được để sử dụng về sau. Các chức năng này được gọi chung là **Dịch vụ lưu trữ** (Data Storage Service).

Ở mức vật lý, các chức năng này có thể được cài đặt vào một hay nhiều tập tin thực thi hình thành các kiểu kiến trúc chương trình khác nhau. Cho đến thời điểm hiện nay, người ta chia kiến trúc của chương trình thành 3 loại được trình bày tiếp theo sau.

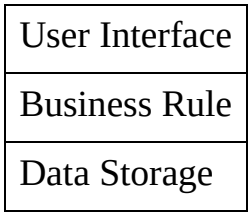
1.7.1. Kiến trúc đơn tầng (Single-tier Architecture)

Trong kiểu kiến trúc này, cả 3 thành phần của chương trình ứng dụng (User

Interface, Business Rule, Data Storage) đều được tích hợp vào một tập tin thực thi.

Ví dụ: BKAV, D2, Winword, . . .

Các ứng dụng kiểu này chỉ được thực thi trên một máy tính.



Hình 1.10 - Kiến trúc chương trình đơn tầng

Ưu điểm:

Dễ dàng trong thiết kế cài đặt ứng dụng kiểu này.

Nhược điểm:

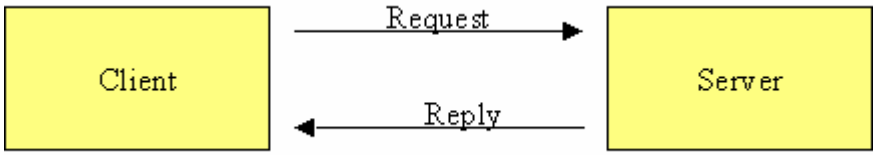
- Bởi vì cả 3 thành phần được cài vào một tập tin thực thi, nên việc sửa lỗi hay nâng cấp chương trình thì rất khó khăn. Toàn bộ chương trình phải biên dịch lại cho dù chỉ sửa đổi một lỗi rất nhỏ trong một thành phần nào đó (User Interface chẳng hạn).
- Việc bảo trì, nâng cấp ấn bản mới là một công việc cực kỳ nặng nề vì ta phải thực hiện việc cài đặt trên tất cả các máy tính.
- Trong kiểu này, mỗi máy tính duy trì một cơ sở dữ liệu riêng cho nên rất khó trong việc trao đổi, tổng hợp dữ liệu.
- Máy tính phải đủ mạnh để có thể thực hiện đồng thời cả 3 loại dịch vụ.

1.7.2. Kiến trúc hai tầng (Two - Tier Architecture)

Kiến trúc này còn được biết đến với tên kiến trúc Client-Server. Kiến trúc này gồm 2 chương trình thực thi: chương trình Client và chương trình Server. Cả

hai chương trình có thể được thực thi trên cùng một máy tính hay trên hai máy tính khác nhau.

Client và Server trao đổi thông tin với nhau dưới dạng các thông điệp (Message) . Thông điệp gởi từ Client sang Server gọi là các thông điệp yêu cầu (Request Message) mô tả công việc mà phần Client muốn Server thực hiện.



Hình 1.11 - Kiến trúc chương trình Client-Server

Mỗi khi Server nhận được một thông điệp yêu cầu, Server sẽ phân tích yêu cầu, thực thi công việc theo yêu cầu và gởi kết quả về client (nếu có) trong một thông điệp trả lời (Reply Message). Khi vận hành, một máy tính làm Server phục vụ cho nhiều máy tính Client.

Mỗi một ứng dụng Client-Server phải định nghĩa một **Giao thức** (Protocol) riêng cho sự trao đổi thông tin, phối hợp công việc giữa Client và Server. Protocol qui định một số vấn đề cơ bản sau:

- Khuôn dạng loại thông điệp.
- Số lượng và ý nghĩa của từng loại thông điệp.
- Cách thức bắt tay, đồng bộ hóa tiến trình truyền nhận giữa Client và Server.
- . . . .

Thông thường phần client đảm nhận các chức năng về User Interface, như tạo các form nhập liệu, các thông báo, các báo biểu giao tiếp với người dùng.

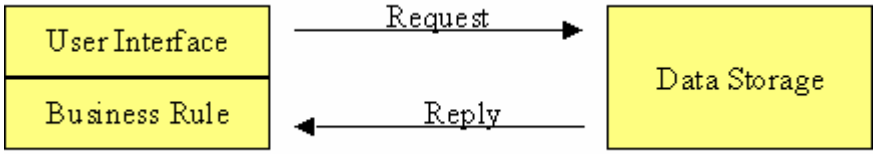
Phần Server đảm nhận các chức năng về Data Storage. Nhờ đó dễ dàng trong việc bảo trì, chia sẻ tổng hợp dữ liệu trong toàn bộ công ty hoặc tổ chức.

Các chức năng về Business Rule có thể được cài đặt ở phần client hoặc ở phần server tạo ra hai loại kiến trúc Client - Server là:

- Fat Client
- Fat Server.

1.7.2.1. Loại Fat Client

Trong loại này Business Rule được cài đặt bên phía Client. Phần Server chủ yếu thực hiện chức năng về truy vấn và lưu trữ thông tin.



Hình 1.12 - Kiến trúc chương trình Client – Server theo kiểu Fat Client

Ưu điểm

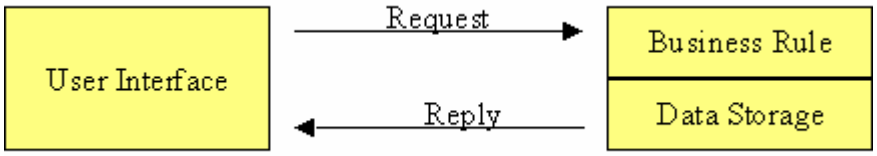
- Tạo ra ít giao thông trên mạng nhờ dữ liệu tạm thời trong quá trình tính toán được lưu tại Client.

Nhược điểm

- Vì Business Rule được cài đặt trên phía Client, đòi hỏi máy tính thực thi phần Client phải đủ mạnh, dẫn đến tốn kém trong chi phí đầu tư phần cứng cho các công ty xí nghiệp.
- Phải cài lại tất cả các máy tính Client khi nâng cấp chương trình.

1.7.2.2. Loại Fat Server

Trong loại này, phần lớn các chức năng về Business Rule được đặt ở phần Server. Phần Client chỉ thực hiện một số chức năng nhỏ của Business Rule về kiểm tra tính hợp lệ của dữ liệu nhập bởi người dùng.



Hình 1.13 - Kiến trúc chương trình Client – Server theo kiểu Fat Server

Ưu điểm

- Vì Business Rule được đặt ở phần Server, các máy tính Client không cần phải có cấu hình mạnh.
- Việc nâng cấp chương trình khi Business Rule thay đổi trở nên nhẹ nhàng hơn vì chỉ phải cài đặt lại phần Server.

Nhược điểm

- Tạo ra nhiều thông điệp trao đổi giữa Client và Server làm tăng giao thông trên mạng.
- Tăng tải trên máy Server vì nó phải đồng thời thực hiện các chức năng của Business Rule và Data Storage làm giảm hiệu năng của chương trình.

1.7.3. Kiến trúc đa tầng (N-Tier Architecture)

Đây là kiến trúc cho các **Ứng dụng phân tán** (Distributed Application). Thông thường là kiến trúc 3 tầng. Chương trình ứng dụng được tách thành 3 phần riêng biệt tương ứng cho 3 chức năng User Interface, Business Rule và Data Storage. Vì các chức năng thuộc về Business Rule được tách thành một phần riêng, nó có thể được thực thi trên một máy tính Server riêng giải quyết được hầu hết các nhược điểm mắc phải của kiến trúc đơn tầng và kiến trúc hai tầng nói trên.

Hình 1.12 - Kiến trúc chương trình đa tầng

Kiến trúc này đáp ứng tốt với những thay đổi về qui luật xử lý dữ liệu của vấn đề mà ứng dụng giải quyết. Việc thay đổi chỉ ảnh hưởng trên tầng Business Rule mà không ảnh hưởng đến hai tầng còn lại.

Thông thường, người ta gọi tên các thành phần trên là:

Client – Application Server – Database Server

1.8. Bài tập

Bài tập 1.1: Protocol HTTP

Tìm đọc và viết một báo cáo không quá 10 trang về giao thức HTTP.

Bài tập 1.2: Chat Protocol

Tìm hiểu về dịch vụ Chat trên mạng Internet. Viết một bảng báo cáo không qua 10 trang trình bày 2 nội dung sau:

- Một bảng mô tả các chức năng thường được hỗ trợ trong một dịch vụ Chat.
- Xây dựng Chat Protocol riêng của bạn trong đó mô tả:
  - Các chức năng hỗ trợ bởi Chat Server.
  - Khuôn dạng (Format) và các loại thông điệp (Message) hỗ trợ bởi
  - Protocol.
  - Sơ đồ trạng thái hoạt động của server và client (giải thuật).
  - Minh họa các kịch bản khác nhau cho từng chức năng của dịch vụ.

Bài tập 1.3: POP3 Protocol

Tìm đọc và viết một báo cáo không quá 10 trang về giao thức POP3.

BÀI 2: I/O TRONG .NET FRAMEWORK

Mã bài: MĐ 38.2

Mục tiêu của bài:

- Trình bày được không gian tên I/O áp dụng cho các mạng truyền dữ liệu.
- Liệt kê các thành phần của không tên System.IO.Streams có sử dụng liên quan đến mạng.
- Mô tả được đối tượng Streams.
- Sử dụng không tên System.IO để ghi và đọc các dữ liệu lên các vùng lưu trữ.
- Sử dụng không tên System.IO để chuyển tải dữ liệu, truy vấn dữ liệu trên mạng.
- Thực hiện các thao tác an toàn với máy tính.

2.1 Giới thiệu về không gian tên IO.

2.1.1 Tổng quan về .NET Framework

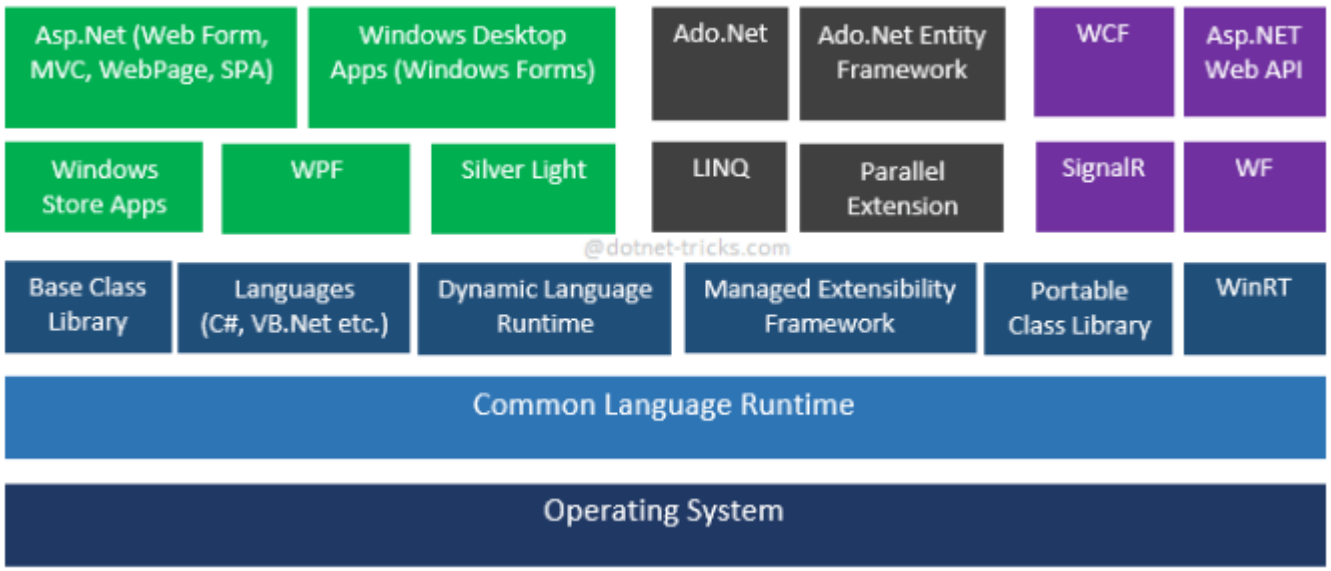
.NET Framework là thành phần nền tảng cho mọi công cụ phát triển ứng dụng .NET

.NET Framework được thiết kế nhằm hỗ trợ cho các trình ứng dụng và các service thế hệ kế tiếp

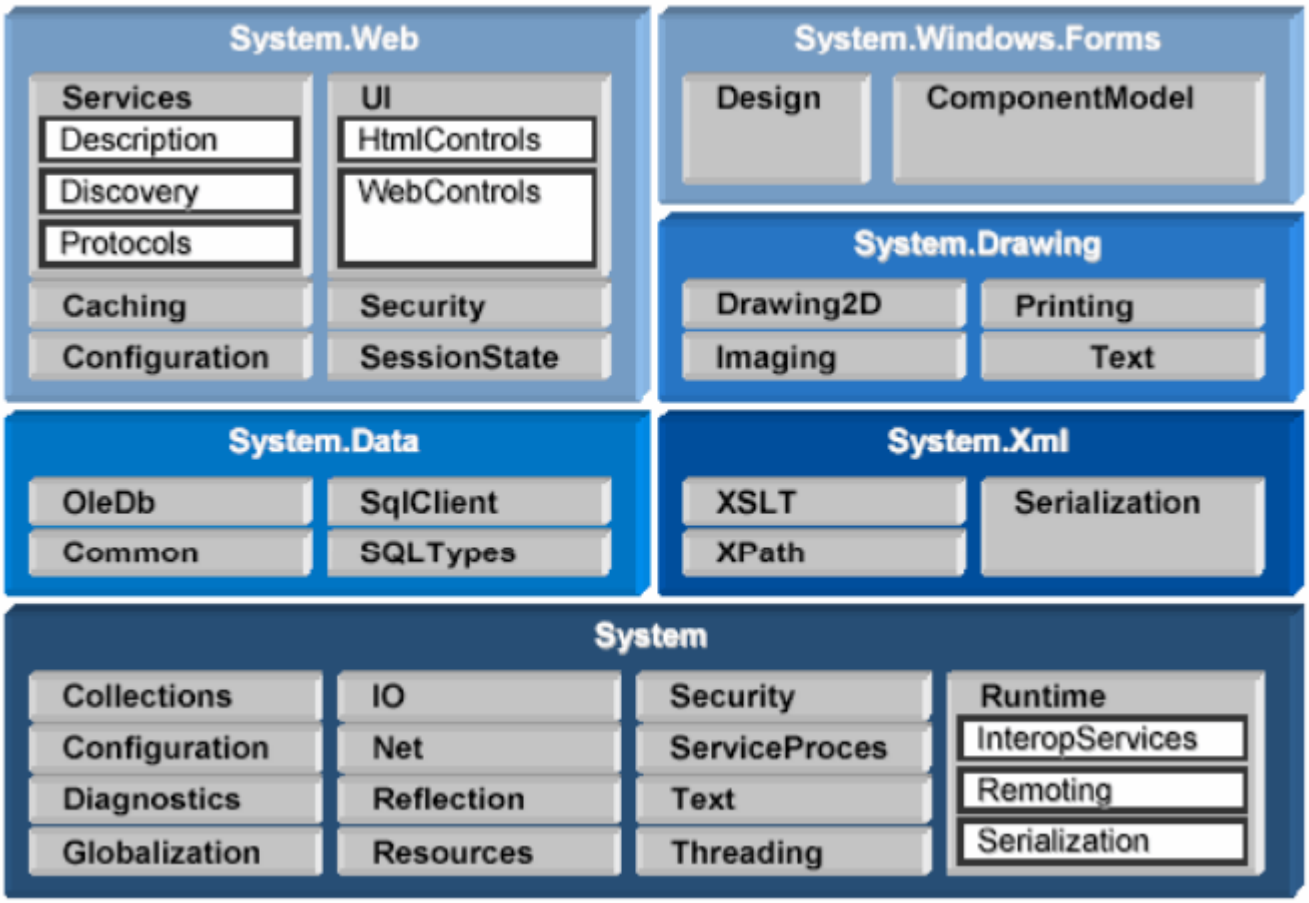
Cung cấp các lớp đối tượng (Class) để có thể gọi thi hành các chức năng mà đối tượng đó cung cấp

Cung cấp hơn 5000 lớp đối tượng để gọi thực hiện đủ các loại dịch vụ từ hệ điều hành

Kiến trúc



Hệ thống namespace trong .NET Framework



Các kiểu dữ liệu

Khái niệm

Trong các ngôn ngữ lập trình cấp cao luôn có các qui định về định nghĩa dữ liệu trong lập trình

Kiểu là khái niệm ám chỉ việc định nghĩa hình thái, cấu trúc và giá trị của dữ liệu, bao gồm cách biểu diễn và xử lý của dữ liệu

Kiểu dữ liệu trong .NET được mô tả chi tiết trong một cấu trúc gọi là Common Type System (CTS)

Kiểu dữ liệu trong .NET được chia thành nhiều loại:

Kiểu giá trị (Value Types)

Kiểu tham chiếu (Reference Types)

Kiểu do người dùng định nghĩa (User-defined Types)

Kiểu liệt kê (Enumerations)

Mỗi kiểu dữ liệu trong .NET là một đối tượng, nghĩa là nó có các thuộc tính và phương thức riêng. Một trong những phương thức thường dùng nhất là Parse và ToString

Cách tạo và sử dụng các kiểu dữ liệu

Các kiểu giá trị (Value Types)

Các kiểu tham chiếu (Reference Types)

Các kiểu giá trị (Value Types)

Bao gồm các kiểu dữ liệu số học ngày giờ, kiểu luận lý, kiểu do người dùng định nghĩa và kiểu liệt kê

Các biến có kiểu giá trị là những biến chứa trực tiếp dữ liệu của chúng thay vì chứa một tham chiếu tới dữ liệu được lưu trữ tại một nơi nào đó trong bộ nhớ (memory)

Các instance của kiểu giá trị được lưu trữ trong một vùng nhớ được gọi là stack, ở đó tại thời điểm vận hành (runtime) có thể tạo (create), đọc (read), cập nhật (update), và loại bỏ (remove) chúng một cách nhanh chóng

Các kiểu giá trị tổng quát:

Các kiểu giá trị được xây dựng sẵn (Built-in types)

Các kiểu do người dùng định nghĩa (User-defined types)

Các kiểu liệt kê (Enumerations)

Kiểu giá trị được xây dựng sẵn (Built-in types)

Là các kiểu cơ sở do NET Framework cung cấp, các kiểu dữ liệu khác đều được xây dựng dựa trên các kiểu dữ liệu cơ sở này

Tất cả các kiểu dạng số (numeric types) được xây dựng sẵn đều thuộc loại kiểu giá trị

Việc sử dụng alias hoàn toàn tương đương với cách sử dụng tên kiểu theo cách đầy đủ, tuy nhiên hầu hết các lập trình viên sử dụng alias để chúng ngắn gọn hơn.

Khi gán các biến kiểu giá trị với nhau, dữ liệu được sao chép từ biến này tới biến kia và được lưu trữ tại hai vị trí khác nhau trên stack

Các kiểu giá trị thường được dùng để trình bày các giá trị đơn giản

Bảng sau liệt kê các kiểu số thường dùng nhất:

Kiểu/Alias C#	Kích thước	Miền giá trị (Range)
System.SByte/sbyte	1 byte	-128 → 127
System.Byte/byte	1 byte	0 → 255
System.Short/short	2 bytes	-32768 → 32767
System.Integer/int	4 bytes	-2147483648 → 2147483647
System.UInteger/uint	4 bytes	0 → 4294967295
System.Long/long	8 bytes	-9223372036854775808→ 9223372036854775807
System.Single/float	4 bytes	-3.402823E+38 → 3.402823E+38
System.Double/double	8 bytes	-1.79769313486232E+308 →1.79769313486232E+308
System.Decimal/decimal	16 bytes	-79228162514264337593543950335 → 79228162514264337593543950335

Kiểu/Alias C#	Kích thước	Miền giá trị (Range)
System.Char/char	2 bytes	N/A
System.Boolean/bool	4 bytes	N/A
System.DateTime (Date/date)	8 bytes	1/1/0001 12:00:00 AM → 12/31/9999 11:59:59 PM

Kiểu do người dùng định nghĩa (User-defined Types)

Kiểu do người dùng định nghĩa còn được gọi là các struct. Giống như các kiểu giá trị khác, các instance của các kiểu do người dùng định nghĩa được lưu trữ trên stack và chứa trực tiếp dữ liệu của chúng

struct là một kiểu cấu trúc hỗn hợp của nhiều kiểu khác nhằm mục đích dễ dàng làm việc với dạng dữ liệu quan hệ

struct có cấu trúc tương tự như class, tuy nhiên struct có kiểu giá trị còn class có kiểu tham chiếu

```
//Khởi tạo
struct Nhan_Vien
{
  public string Ma_so;
  public string Ho_ten;
  public date Ngay_sinh;
  public double He so luong;
}
```

//Khai báo biến nhân viên và gán các giá trị

Nhan_Vien nv;

nv.Ma_so = "MS01"

nv.Ho_ten = "Võ Nhật Nam"

nv.Ngay_sinh = DateTime.Parse("10/10/1989")

nv.He_so_luong = 2.0

Kiểu liệt kê (Enumerations)

Kiểu liệt kê là các ký hiệu có cùng một họ (có liên quan với nhau) có các giá trị cố định.

Kiểu liệt kê được dùng để cung cấp một danh sách chọn lựa cho lập trình viên sử dụng

Lưu ý: khi khai báo kiểu liệt kê chỉ có thể sử dụng các kiểu số nguyên như byte, sbyte, short, ushort, int, uint, long hoặc ulong

enum Titles : int { Mr, Ms, Mrs, Dr };

Nếu bạn tạo một instance của kiểu *Titles* , Visual Studio sẽ hiển thị một danh sách các giá trị có sẵn để gán giá trị cho biến


```
Titles t = Titles.Dr;
Console.WriteLine("{0}.", t); // Displays "Dr."
```

Cách khai báo các kiểu giá trị

Các kiểu giá trị có một khởi tạo (constructor) ngầm định, do vậy bạn không cần phải bao gồm trong phần khai báo từ khóa New như khi khai báo với class

bool b = false;

Trong C# mỗi câu lệnh được kết thúc bởi dấu ";"

.NET cung cấp một kiểu mới đó là kiểu Nullable cho phép kiểm tra liệu một biến đã được gán giá trị hay chưa

Nullable<bool> b = null;

// C# cho phép sử dụng ký hiệu thể hiện nhanh như sau

bool? b = null;

Một biến có kiểu nullable cho phép sử dụng các thành viên HasValue và Value. Bạn có thể sử dụng HasValue để kiểm tra liệu có hay không một giá trị đã được thiết lập

```
if (b.HasValue)
  Console.WriteLine("b is {0}.", b.Value);
else
  Console WriteLine("b is not set ");
```

Các kiểu tham chiếu (Reference Types)

Các kiểu tham chiếu lưu trữ địa chỉ dữ liệu của chúng c , cũng được hiểu như là một con trỏ (pointer), trên stack.

Dữ liệu thực sự mà các tham chiếu địa chỉ tham chiếu tới được lưu trữ trong một vùng bộ nhớ được gọi là heap

Hầu hết các kiểu trong .NET Framework là kiểu tham chiếu. Các kiểu tham chiếu cung cấp rất nhiều cho tính linh động, uyển chuyển, chúng mang đến khả năng thực thi hoàn hảo khi gởi chúng tới các phương thức

Các kiểu tham chiếu được xây dựng sẵn

Thời điểm vận hành (runtime) quản lý bộ nhớ bằng việc sử dụng heap thông qua một tiến trình được gọi là garbage collection. Garbage collection sẽ khởi tạo bộ nhớ theo chu kỳ dựa theo nhu cầu bằng cách loại bỏ các đối tượng không còn được tham chiếu nữa.

Có khoảng 2500 kiểu tham chiếu được xây dựng sẵn trong .NET Framework (không được dẫn xuất từ System.ValueType)

Các kiểu tham chiếu thường dùng:

— Các kiểu tham chiếu thường dùng:

Kiểu	Ý nghĩa sử dụng
System.Object	Kiểu tổng quát
System.String	Dữ liệu dạng Text
System.Text.StringBuilder	Dữ liệu dạng Dynamic text
System.Array	Mảng dữ liệu
System.IO.Stream	Bộ đệm (Buffer) cho tập tin, thiết bị
System.Exception	Kiểm soát hệ thống và trình ứng dụng

Sử dụng string và StringBuilder

Khi có bất kỳ sửa đổi đối với chuỗi tại thời điểm vận hành sẽ tạo nên một chuỗi mới và bỏ đi chuỗi cũ

```
string s = "this is some text to search";
s = s.Replace("search", "replace");
Console.WriteLine(s);
```

Đoạn mã lệnh sau đây sẽ cấp phát bốn chuỗi mới trong bộ nhớ: Chỉ chuỗi sau cùng là có một tham chiếu; ba chuỗi kia sẽ được dispose khi công cụ thu gom rác làm việc

```
string s;
s = "wombat"; // "wombat"
s += " kangaroo"; // "wombat kangaroo"
s += " wallaby"; // "wombat kangaroo wallaby"
s += " koala"; // "wombat kangaroo wallaby koala"
Console.WriteLine(s);
```

Việc ngăn ngừa các loại chuỗi tạm thời này giúp tránh gom rác không cần thiết, cải thiện hiệu suất hoạt động. Có một số cách để ngăn ngừa các chuỗi tạm thời:

Sử dụng các phương thức Concat, Join, hay Format của lớp String ghép nhiều item trong một câu lệnh đơn lẻ

Sử dụng lớp StringBuilder để tạo các chuỗi dạng dynamic (có thể biến đổi - mutable)

```
System.Text.StringBuilder sb = new
System.Text.StringBuilder(30);
sb.Append("wombat"); // Build string.
sb.Append(" kangaroo");
sb.Append(" wallaby");
sb.Append(" koala ); ");
string s = sb.ToString(); // Copy result to string.
Console.WriteLine(s);
```

Sự chuyển đổi giữa các kiểu

Chuyển đổi ngầm định: C# chấp nhận sự chuyển kiểu ngầm định nếu kiểu đích có thể thích hợp với tất cả các giá trị có thể nhận từ kiểu nguồn.

Chuyển đổi tường minh: tất cả các biến trong C# phải được khai báo kiểu một cách rõ ràng, và trình biên dịch kiểm tra tính nhất quán về kiểu của biểu thức.

```
int i = 1;
double d = 1.0001;
d = i; // Sự chuyển đổi được chấp nhận
```

Có thể thực hiện việc chuyển đổi kiểu tường minh bằng cách sử dụng toán tử chuyển kiểu (hay ép kiểu - cast operator), nó có dạng:

```
long L = 32769;
int I = (int) L;
```

Cơ chế Boxing và Unboxing: boxing chuyển đổi một kiểu giá trị tới một kiểu tham chiếu, và unboxing chuyển đổi một kiểu tham chiếu tới một kiểu giá trị.

```
int i = 123;
object o = (object) i; // cơ chế boxing

object j = 123;
int k = (int)j; //cơ chế unboxing
```

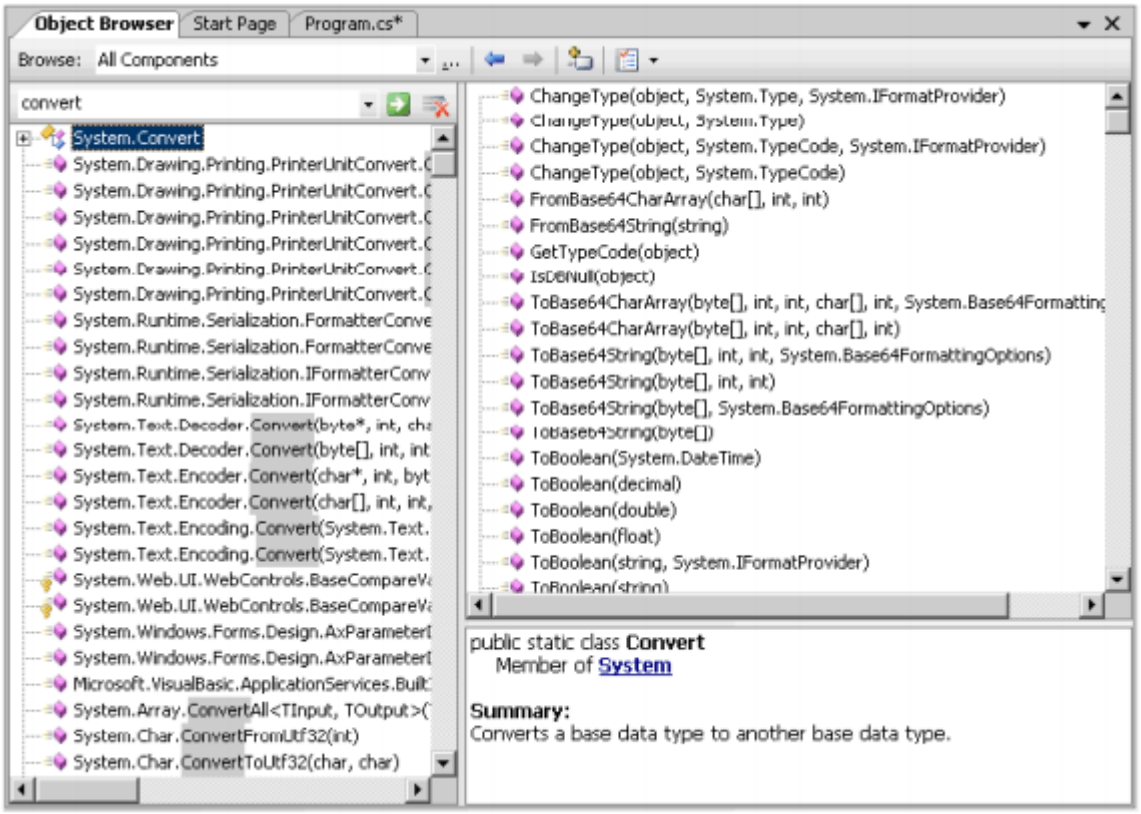
Sử dụng toán tử as: toán tử as cũng được dùng giống như ép kiểu với một số lưu ý:

```
int j;
object i = "35";
j = i as int ;
// Đoạn lệnh trên sẽ xảy ra lỗi do int là kiểu giá trị
//Nếu không thể chuyển đổi kiểu nó s , nó sẽ trả về giá trị null
string j;
object i = "35";
j = i as string;
if (j == null)
  Console.WriteLine("i không phải là chuỗi");
else
  Console.WriteLine("i là chuỗi");
```

Sử dụng các phương thức Convert:

Các thành viên public được chia sẽ của lớp System.Convert có thể được sử dụng để chuyển đổi một kiểu dữ liệu cơ sở tới kiểu dữ liệu cơ sở khác

Sử dụng Object Browser để xem và tìm hiểu những phương thức chuyển đổi sẵn có (Để mở công cụ Object Browser: chọn thực đơn view => Object Browser)




```
//chuyển đổi từ kiểu số thực sang kiểu số nguyên
double SoThuc = 23.15;
try
{
  int SoNguyen = System.Convert.ToInt32(SoThuc);
}
catch (System.OverflowException)
{
  System.Console.WriteLine("Lỗi do số cần chuyển lớn hơn biến chứa kết quả chuyển.");
}
//chuyển đổi từ kiểu số thực sang kiểu chuỗi
double SoThuc = 23.15;
string strNumber = System.Convert.ToString(SoThuc);
//Sử dụng phương thức Parse của kiểu dữ liệu:
//Ví dụ 1: gán giá trị vào cho biến có kiểu ngày
DateTime Ngay = DateTime.Parse("01/25/2008");
```

//Ví dụ 2: chuyển đổi số nguyên sang số thực

double SoThuc;

int SoNguyen = 3;

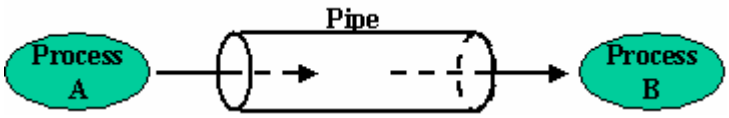
SoThuc = double Parse(SoNguyen);

2.2 Streams

2.2. 1 Giới thiệu về ống dẫn

Ống dẫn là một tiện ích được hỗ trợ trong hầu hết các ngôn ngữ lập trình vận hành trên các hệ thống đa nhiệm. Ống dẫn cho phép hai quá trình nằm trên cùng một máy có thể trao đổi dữ liệu với nhau.

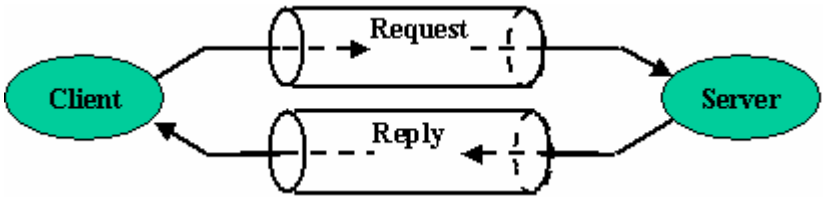
Dữ liệu đi trên ống dẫn theo một chiều nhất định. Khi sử dụng ống dẫn, người ta dùng một đầu cho việc viết dữ liệu vào và một đầu còn lại cho việc đọc dữ liệu ra.



Hình 2.1 Mô hình ống dẫn

Ống dẫn thích hợp cho trường hợp dữ liệu tạo ra của quá trình này sẽ là dữ liệu đầu vào cho quá trình kia.

Tuy nhiên ta cũng có thể sử dụng ống dẫn để xây dựng các ứng dụng theo kiến trúc Client- Server bằng cách sử dụng hai ống dẫn: một ống dẫn để truyền các yêu cầu (request), một ống dẫn để truyền các trả lời (reply).



Hình 3.2 – Dùng ống dẫn trong chương trình Client -Server

Có hai loại ống dẫn:

- Ống dẫn bình thương (Normal Pipe): Giới hạn trong phạm vi không gian địa chỉ của một quá trình mà thôi. Nó chỉ cho phép giao tiếp giữa quá trình cha với các quá trình con hay giữa các quá trình con của một quá trình với nhau. Java hỗ trợ ống dẫn loại này. Trong đó các quá trình con được thay thế bởi các luồng.
- Ống dẫn có tên (Named Pipe): Loại này có thể cho phép hai quá trình có không gian địa chỉ khác nhau (trên cùng một máy) giao tiếp với nhau.

Thực chất nó giống như một tập tin với qui định rằng dữ liệu sẽ được lấy ra ở đầu tập tin và được thêm vào ở cuối tập tin.

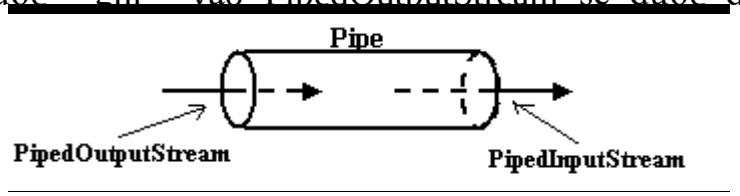
2.2. Ống dẫn trong Java

2.2.1. Giới thiệu

Java hỗ trợ tiên ích ống dẫn thông qua hai lớp java.io.PipedInputStream và java.io.PipedOutputStream. Chúng là hai đầu của một ống dẫn. Trong đó PipedInputStream là đầu đọc dữ liệu và PipedOutputStream là đầu ghi dữ liệu của ống dẫn. PipedInputStream là lớp con của InputStream nên nó có tất cả các thuộc tính của InputStream. PipedOutputStream là lớp con của OutputStream nên nó có tất cả các thuộc tính của OutputStream

2.2.2. Các cách tạo ống dẫn

Để tạo một ống dẫn ta chỉ cần tạo ra hai đối tượng thuộc lớp PipedInputStream và PipedOutputStream và nối chúng lại với nhau. Khi đó dữ liệu được ghi vào PipedOutputStream sẽ được đọc ra ở đầu PipedInputStream:



Hình 2.3 - Tạo ống dẫn trong Java

Cách 1

1. Tạo đầu đọc:
   - PipedInputStream readId = new PipedInputStream();
2. Tạo đầu ghi:
   - PipedOutputStream writeId = new PipedOutputStream();
3. Nối đầu đọc với đầu ghi hay ngược lại
   - readId.connect(writeId);
   - hoặc writeId.connect(readId);

Cách 2

1. Tạo đầu đọc:
   - PipedInputStream readId = new PipedInputStream();
2. Tạo đầu ghi và nối vào đầu đọc đã có:

- PipedOutputStream writeId = new PipedOutputStream(readId);

Hoặc: Ta có thể tạo đầu ghi trước rồi tạo đầu đọc sau.

Lưu ý: Các phương thức khởi tạo của PipedInputStream và PipedOutputStream sử dụng ở trên đòi hỏi phải "bắt" (catch) IOException do chúng có thể "quẳng" ra (throws)..

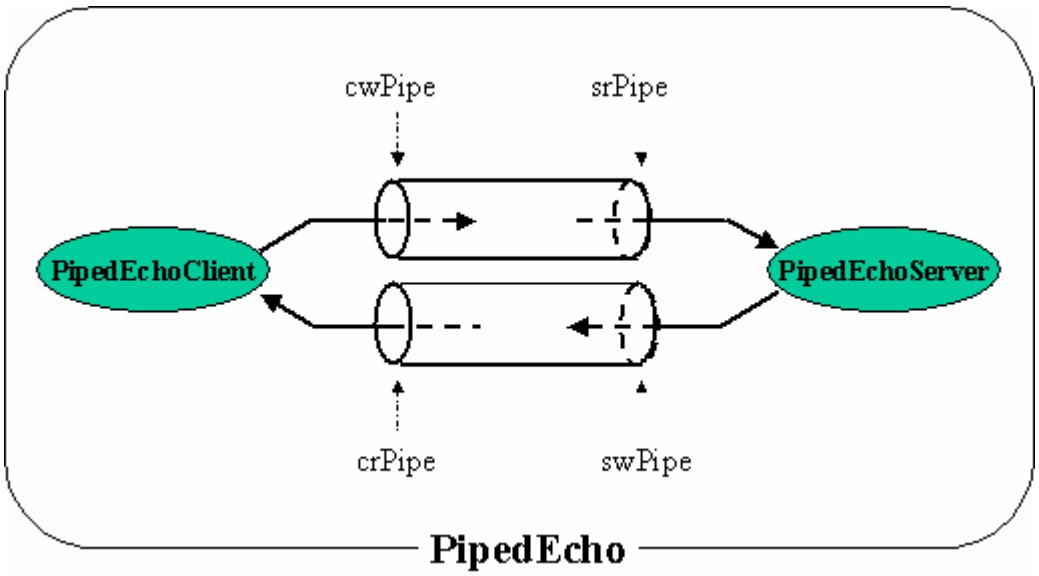
2.3. Dịch vụ phản hồi thông tin (Echo Service)

Dịch vụ phản hồi thông tin là một dịch vụ tồn tại trên hệ thống mạng UNIX. Dịch vụ này được xây dựng theo mô hình Client - Server, cơ chế hoạt động như sau:

- Phần Server: Chờ nhận các byte gởi đến từ Client. Với mỗi byte nhận được, Server sẽ gởi về đúng byte đã nhận trở về Client.
- Phần Client: Gởi các byte sang Server, chờ nhận các byte gởi về từ Server.

2.4. Giả lập dịch vụ phản hồi thông tin bằng Pipe

Phần kế tiếp ta xây dựng một ứng dụng có tên là PipeEcho mô phỏng dịch vụ phản hồi thông tin để minh họa cách sử dụng Pipe làm phương tiện giao tiếp giữa các Thread trong một ứng dụng.



Hình 2.4 – Mô hình ứng dụng PipeEcho

Trong ứng dụng này Client và Server là hai Thread thuộc lớp PipedEchoClient và PipedEchoServer. Việc trao đổi thông tin giữa client và server được thực hiện thông qua 2 Pipe (cwPipe-srPipe) và (swPipe-crPipe).

PipedEchoClient nhận các ký tự từ bàn phím, gởi chúng sang PipedEchoServer và chờ nhận các ký tự gởi về từ PipedEchoServer để in ra màn

hình. PipedEchoServer chờ nhận từng ký tự từ PipedEchoClient, đổi ký tự nhận được thành ký tự hoa và gởi ngược về PipedEchoClient.

2.4.1. Lớp PipedEchoServer

Hãy lưu chương trình sau vào tập tin PipedEchoServer.java

```
import java.io.*;
public class PipedEchoServer extends Thread {
PipedInputStream readPipe;
PipedOutputStream writePipe;
PipedEchoServer(PipedInputStream readPipe, PipedOutputStream writePipe){
this.readPipe = readPipe;
this.writePipe = writePipe;
System.out.println("Server is starting . . ."); start();
}
public void run(){
while(true) {
try{
int ch = readPipe.read();
ch = Character.toUpperCase((char)ch);
writePipe.write(ch);
}
catch (IOException ie) { System.out.println("Echo Server Error: "+ie );
}
}
}
}
```

2.4.2. Lớp PipedEchoClient

Hãy lưu chương trình sau vào tập tin PipedEchoClient.java

```
import java.io.*;
public class PipedEchoClient extends Thread {
PipedInputStream readPipe;
PipedOutputStream writePipe;
```

```
PipedEchoClient(PipedInputStream readPipe, PipedOutputStream writePipe){
this.readPipe = readPipe;
this.writePipe = writePipe;
System.out.println("Client creation");
start();
}
public void run(){
while(true) {
try {
int ch=System.in.read();
writePipe.write(ch);
ch = readPipe.read();
System.out.print((char)ch);
}
catch(IOException ie){
System.out.println("Echo Client Error: "+ie );
}
}
}
}
```

2.4.3. Lớp PipedEcho

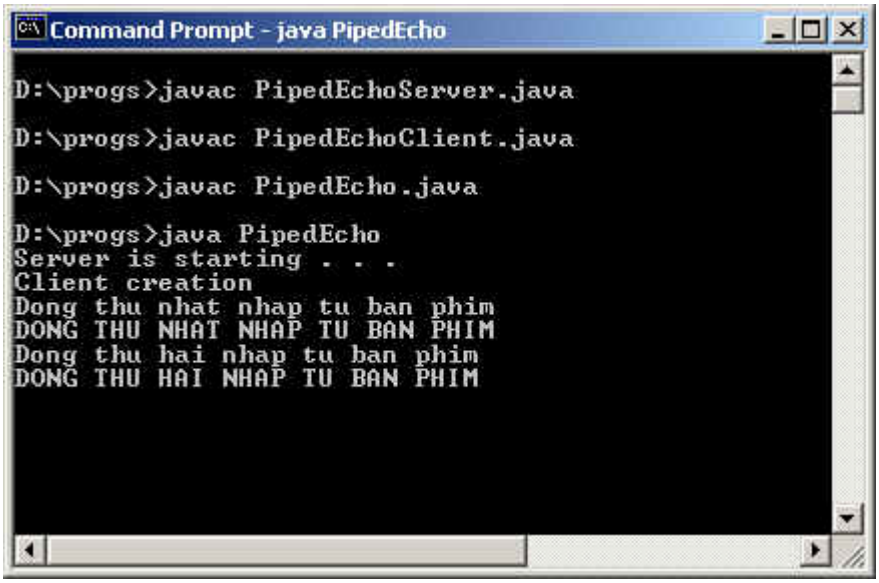
Hãy lưu chương trình sau vào tập tin PipedEcho.java

```
import java.io.*;
public class PipedEcho {
public static void main(String argv[]){
try{
PipedOutputStream cwPipe = new PipedOutputStream();
PipedInputStream crPipe = new PipedInputStream();
PipedOutputStream swPipe = new PipedOutputStream(crPipe);
PipedInputStream srPipe = new PipedInputStream(cwPipe);
PipedEchoServer server = new PipedEchoServer(srPipe,swPipe);
PipedEchoClient client = new PipedEchoClient(crPipe,cwPipe);
```

```
} catch(IOException ie) {
System.out.println("Pipe Echo Error:"+ie);
}
}
}
```

2.4.5. Biên dịch và thực thi chương trình

Biên dịch và thực thi chương trình theo cách sau:



Hình 2.5 Thực thi PipedEchoServer.java, PipedEchoClient.java, PipedEcho.java

Nhập vào bàn phím chuỗi ký tự mà bạn muốn rồi nhấp phím Enter. Bạn sẽ thấy chuỗi ký tự in hoa của chuỗi vừa nhập xuất hiện trên màn hình.

2.5 Câu hỏi và bài tập

Câu 1: Trình bày các đặc điểm của Pipe

Câu 2: Hãy trình bày các cách tạo ống dẫn

Câu 3: Dịch vụ phản hồi thông tin là gì? Trình bày cách Giả lập dịch vụ phản hồi thông tin bằng Pipe.

Câu 4: Xây dựng, biên dịch và thực thi thành chương trình PipedEcho.

BÀI 3: LÀM VIỆC VỚI SOCKETS

Mã bài : MĐ 38.4

Mục tiêu:

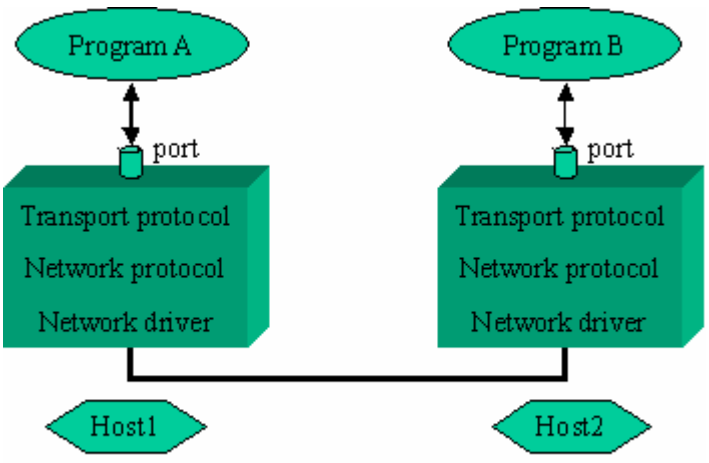
- Mô tả mô hình Client/Server;
- Mô tả lớp Socket;
- Trình bày chế độ làm việc của socket ở Client và Server;
- Viết các ứng dụng trên mạng dùng Socket.
- Thực hiện các thao tác an toàn với máy tính.

3.1. Giới thiệu về socket

3.1.1. Giới thiệu

Socket là một giao diện lập trình ứng dụng (API-Application Programming Interface). Nó được giới thiệu lần đầu tiên trong ấn bản UNIX - BSD 3.2. dưới dạng các hàm hệ thống theo cú pháp ngôn ngữ C (socket(), bind(), connect(), send(), receive(), read(), write(), close() ,..). Ngày nay, Socket được hỗ trợ trong hầu hết các hệ điều hành như MS Windows, Linux và được sử dụng trong nhiều ngôn ngữ lập trình khác nhau: như C, C++, Java, Visual Basic, Visual C++, . . .

Socket cho phép thiết lập các kênh giao tiếp mà hai đầu kênh được đánh dấu bởi hai cổng (port). Thông qua các cổng này một quá trình có thể nhận và gởi dữ liệu với các quá trình khác.



Hình 3.1 – Mô hình Socket

Có hai kiểu socket:

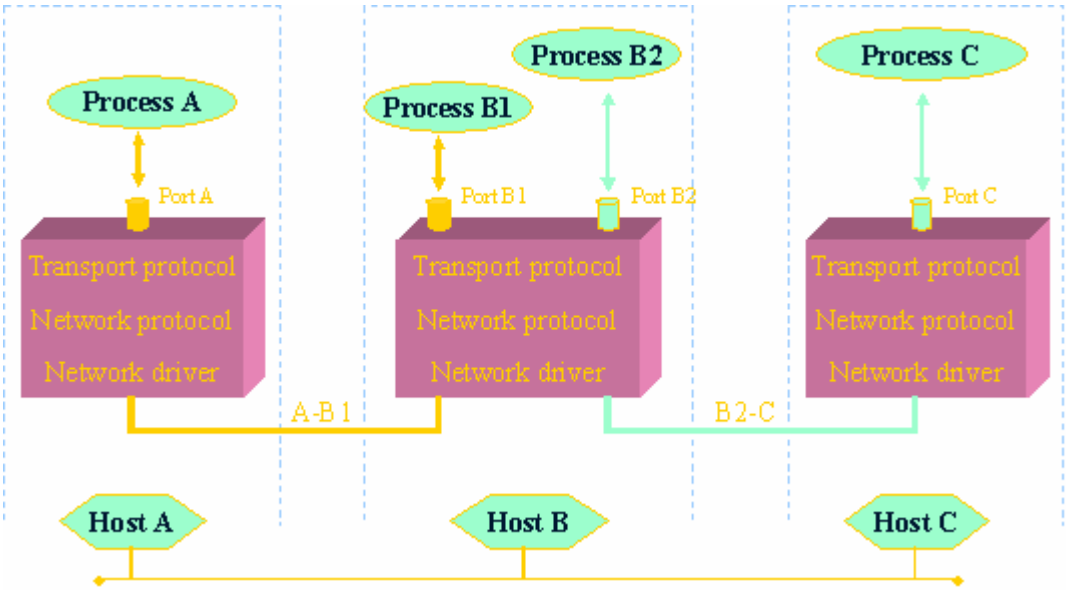
1. Socket kiểu AF_UNIX chỉ cho phép giao tiếp giữa các quá trình trong cùng một máy tính

2. Socket kiểu AF_INET cho phép giao tiếp giữa các quá trình trên những máy tính khác nhau trên mạng.

3.1.2. Số hiệu cổng (Port Number) 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 được yêu cầu tạo ra một socket.

Ngoài số hiệu cổng, hai bên giao tiếp còn phải biết địa chỉ IP của nhau. Địa chỉ IP giúp phân biệt máy tính này với máy tính kia trên mạng TCP/IP. Trong khi số hiệu cổng dùng để phân biệt các quá trình khác nhau trên cùng một máy tính.



Hình 3.2 – Cổng trong Socket

Trong hình trên, địa chỉ của quá trình B1 được xác định bằng 2 thông tin: (Host B, Port B1):

Địa chỉ máy tính có thể là địa chỉ IP dạng 203.162.36.155 hay là địa chỉ theo dạng tên miền như www.cdndalat.edu.vn

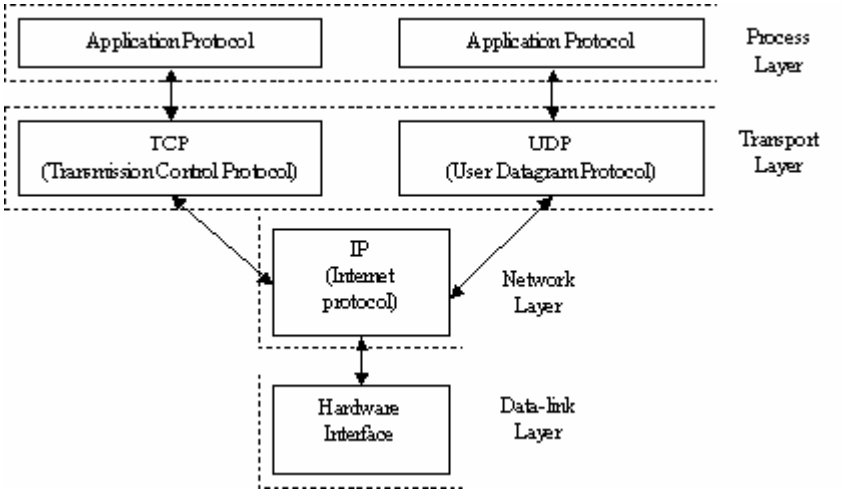
Số hiệu cổng gán cho Socket phải duy nhất trên phạm vi máy tính đó, có giá trị trong khoảng từ 0 đến 65535 (16 bits). Trong đó, các cổng từ 1 đến 1023 được gọi là cổng hệ thống được dành riêng cho các quá trình của hệ thống.

Các cổng mặc định của 1 số dịch vụ mạng thông dụng:

Số hiệu cổng	Quá trình hệ thống
7	Dịch vụ Echo
21	Dịch vụ FTP
23	Dịch vụ Telnet
25	Dịch vụ E-mail (SMTP)
80	Dịch vụ Web (HTTP)
110	Dịch vụ E-mail (POP)

3.1.3. Các chế độ giao tiếp

Xét kiến trúc của hệ thống mạng TCP/IP



Hình 3.3 – Bộ giao thức TCP/IP

Tầng vận chuyển giúp chuyển tiếp các thông điệp giữa các chương trình ứng dụng với nhau. Nó có thể hoạt động theo hai chế độ:

- Giao tiếp có nối kết, nếu sử dụng giao thức TCP
- Hoặc giao tiếp không nối kết, nếu sử dụng giao thức UDP

Socket là giao diện giữa chương trình ứng dụng với tầng vận chuyển. Nó cho phép ta chọn giao thức sử dụng ở tầng vận chuyển là TCP hay UDP cho chương trình ứng dụng của mình.

Bảng sau so sánh sự khác biệt giữa hai chế độ giao tiếp có nối kết và không nối kết:

Chế độ có nối kết (TCP)	Chế độ không nối kết (UDP)
▪ Tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp ▪ Dữ liệu được gởi đi theo chế độ bảo đảm: có kiểm tra lỗi. truyền lại gói tin lỗi hay mất, bảo đảm thứ tự đến của các gói tin . . . ▪ Dữ liệu chính xác, Tốc độ truyền chậm.	▪ Không tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp ▪ Dữ liệu được gởi đi theo chế độ không bảo đảm: Không kiểm tra lỗi, khôngphát hiện không truyền lại gói tin bị lỗi hay mất, không bảo đảm thứ tự đến của các gói tin . . . ▪ Dữ liệu không chính xác, tốc độ truyền nhanh. ▪ Thích hợp cho các ứng dụng cần

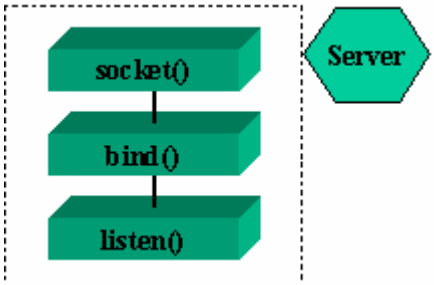
3.2. Xây dựng ứng dụng Client-Server với Socket

Socket là phương tiện hiệu quả để xây dựng các ứng dụng theo kiến trúc Client- Server. Các ứng dụng trên mạng Internet như Web, Email, FTP là các ví dụ điển hình.

Phần này trình bày các bước cơ bản trong việc xây dựng các ứng dụng Client- Server sử dụng Socket làm phương tiện giao tiếp theo cả hai chế độ: Có nối kết và không nối kết.

3.2.1. Mô hình Client-Server sử dụng Socket ở chế độ có nối kết (TCP)

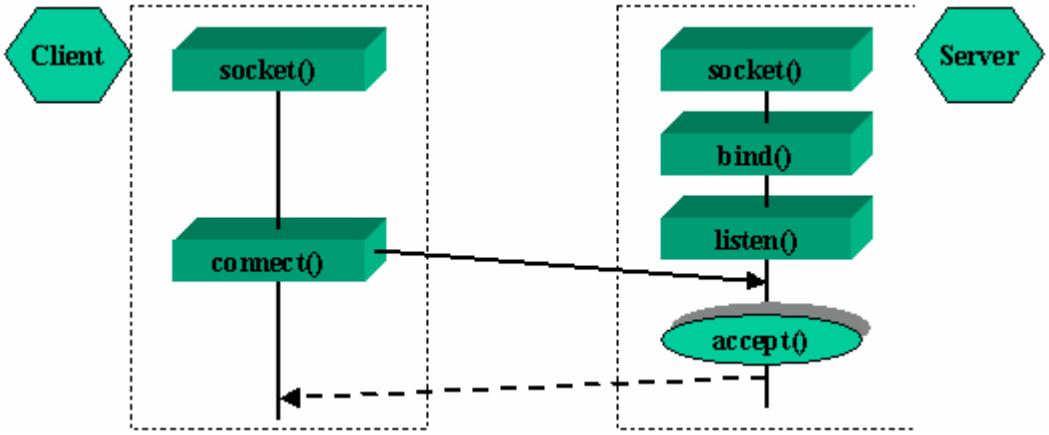
Giai đoạn 1: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết.



Hình 3.4 Server tạo Socket

- socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.
- listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán. Server sẵn sàng phục vụ Client.

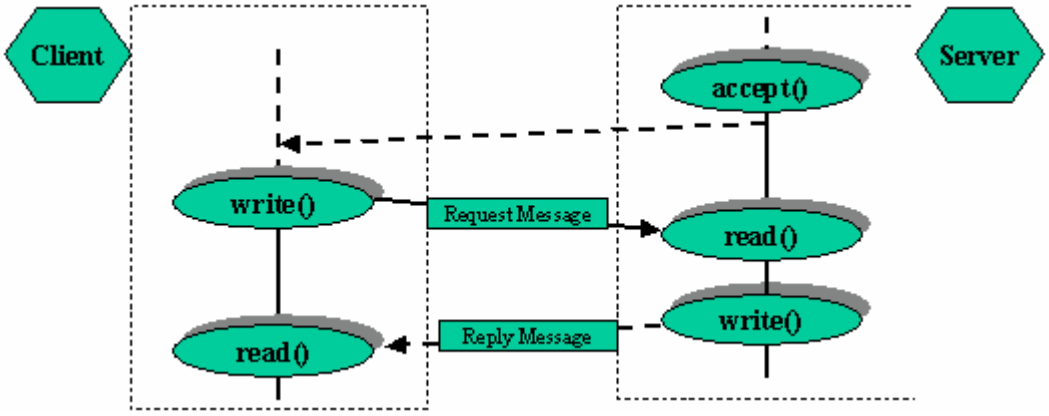
Giai đoạn 2: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.



Hình 3.5 Client tạo Socket

- socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
- connect(): Client gởi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
- accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.

Giai đoạn 3: Trao đổi thông tin giữa Client và Server.



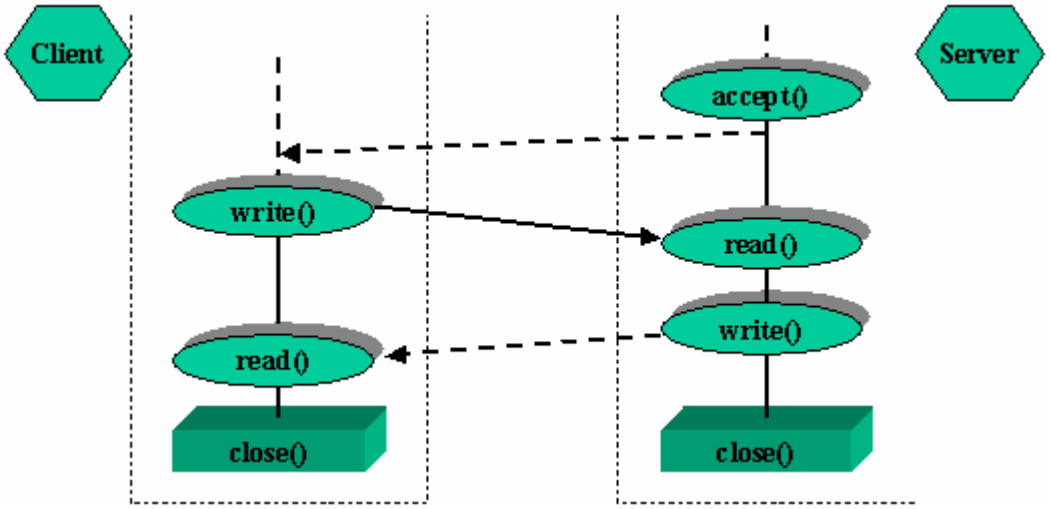
Hình 3.6 Trao đổi thông tin giữa Client và Server.

- Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ client gởi đến.
- Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh write().
- Sau khi gởi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().

Trong giai đoạn này, việc trao đổi thông tin giữa Client và Server phải tuân thủ giao thức của ứng dụng (Dạng thức và ý nghĩa của các thông điệp, qui tắc bắt tay, đồng bộ hóa,...). Thông thường Client sẽ là người gởi yêu cầu đến Server trước.

Nếu chúng ta phát triển ứng dụng theo các Protocol đã định nghĩa sẵn, chúng ta phải tham khảo và tuân thủ đúng những qui định của giao thức. Bạn có thể tìm đọc mô tả chi tiết của các Protocol đã được chuẩn hóa trong các tài liệu RFC (Request For Comments). Ngược lại, nếu chúng ta phát triển một ứng dụng Client-Server riêng của mình, thì công việc đầu tiên chúng ta phải thực hiện là đi xây dựng Protocol cho ứng dụng.

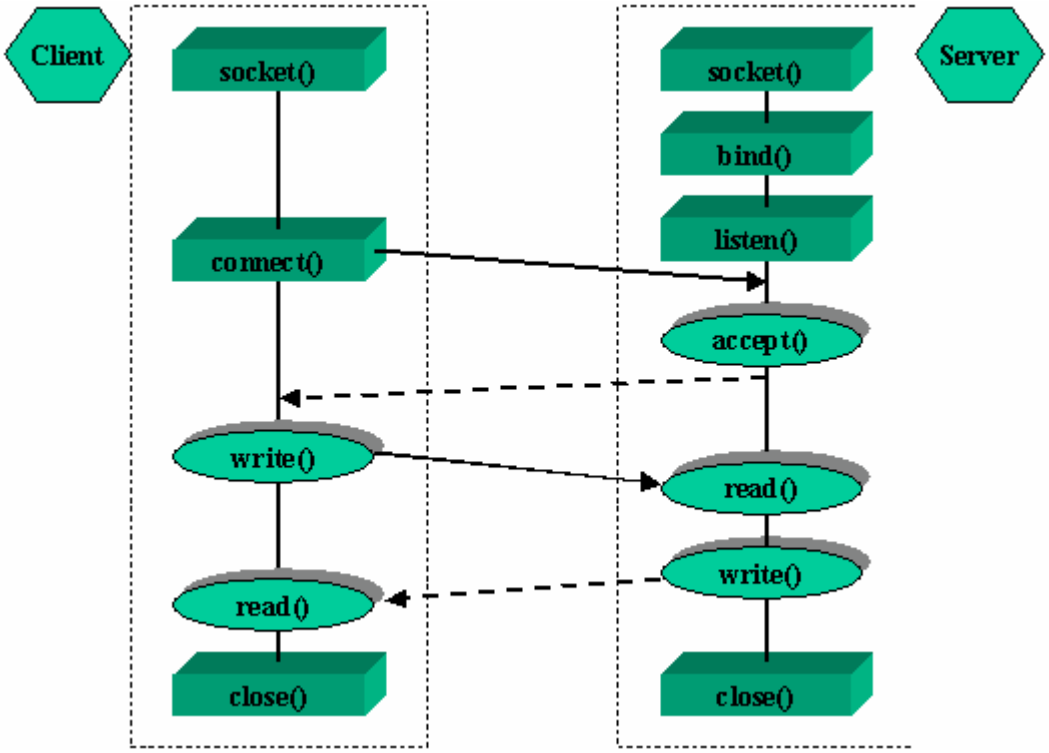
Giai đoạn 3: Kết thúc phiên làm việc.



Hình 3.7 Kết thúc phiên làm việc

- Các câu lệnh read(), write() có thể được thực hiện nhiều lần (ký hiệu bằng hình ellipse).
- Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

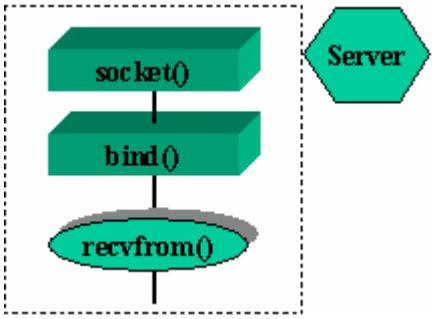
Như vậy toàn bộ tiến trình diễn ra như sau:



Hình 3.8 toàn bộ tiến trình ở chế độ có nối kết

3.2.2. Mô hình Client-Server sử dụng Socket ở chế độ không nối kết (UDP)

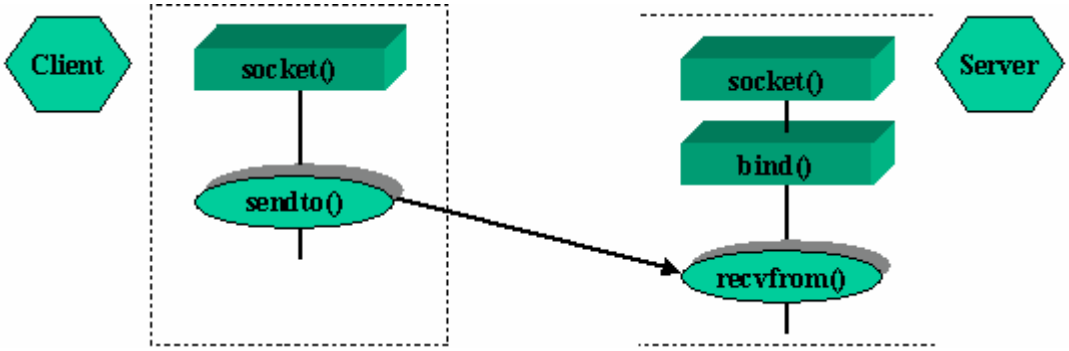
Giai đoạn 1: Server tạo Socket - gán số hiệu cổng.



Hình 3.9 Server tạo Socket

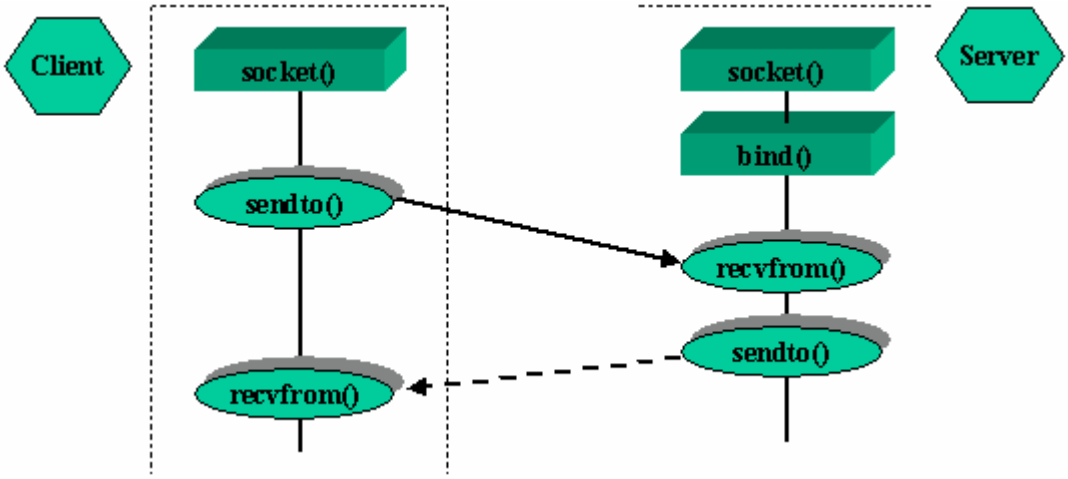
- socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
- bind(): Server yêu cầu gán số hiệu cổng cho socket..

Giai đoạn 2: Client tạo Socket.



Hình 3.10 Client tạo Socket

Giai đoạn 3: Trao đổi thông tin giữa Client và Server.



Hình 3.11 Trao đổi thông tin giữa Client và Server

Sau khi tạo Socket xong, Client và Server có thể trao đổi thông tin qua lại với nhau thông qua hai hàm sendto() và recvfrom(). Đơn vị dữ liệu trao đổi giữa Client và Server là các **Datagram Package** (Gói tin thư tín). Protocol của ứng dụng phải định nghĩa khuôn dạng và ý nghĩa của các Datagram Package. Mỗi Datagram Package có chứa thông tin về địa chỉ người gởi và người nhận (IP, Port).

3.3. Socket dưới ngôn ngữ Java

Java hỗ trợ lập trình mạng thông qua các lớp trong gói **java.net**. Một số lớp tiêu biểu được dùng cho lập trình Client-Server sử dụng socket làm phương tiện giao tiếp như:

InetAddress: Lớp này quản lý địa chỉ Internet bao gồm địa chỉ IP và tên máy tính.

- Socket: Hỗ trợ các phương thức liên quan đến Socket cho chương trình Client ở chế độ có nối kết.
- ServerSocket: Hỗ trợ các phương thức liên quan đến Socket cho chương

trình Server ở chế độ có nối kết.

- DatagramSocket: Hỗ trợ các phương thức liên quan đến Socket ở chế độ không nối kết cho cả Client và Server.
- DatagramPacket: Lớp cài đặt gói tin dạng thư tín người dùng (Datagram Packet) trong giao tiếp giữa Client và Server ở chế độ không nối kết.

3.3.1. Xây dựng chương trình Client ở chế độ có nối kết

Các bước tổng quát:

1. Mở một socket nối kết đến server đã biết địa chỉ IP (hay tên miền) và số hiệu cổng.
2. Lấy InputStream và OutputStream gán với Socket.
3. Tham khảo Protocol của dịch vụ để định dạng đúng dữ liệu trao đổi với

Server.

4. Trao đổi dữ liệu với Server nhờ vào các InputStream và OutputStream.
5. Đóng Socket trước khi kết thúc chương trình.

3.3.1.1. Lớp java.net.Socket

Lớp Socket hỗ trợ các phương thức cần thiết để xây dựng các chương trình client sử dụng socket ở chế độ có nối kết. Dưới đây là một số phương thức thường dùng để xây dựng Client:

public Socket(String HostName, int PortNumber) throws IOException

Phương thức này dùng để nối kết đến một server có tên là HostName, cổng là PortNumber. Nếu nối kết thành công, một kênh ảo sẽ được hình thành giữa Client và Server.

- HostName: Địa chỉ IP hoặc tên logic theo dạng tên miền.
- PortNumber: có giả trị từ 0 ..65535

public InputStream getInputStream()

Phương thức này trả về InputStream nối với Socket. Chương trình Client dùng

InputStream này để nhận dữ liệu từ Server gởi về.

Ví dụ: Lấy InputStream của Socket s: InputStream is = s.getInputStream();

public OutputStream getOutputStream()

Phương thức này trả về OutputStream nối với Socket. Chương trình Client dùng

OutputStream này để gởi dữ liệu cho Server.

Ví dụ: Lấy OutputStream của Socket s: OutputStream os = s.getOutputStream();

public close()

Phương thức này sẽ đóng Socket lại, giải phóng kênh ảo, xóa nối kết giữa Client và Server.

Ví dụ: Đóng Socket s:

s.close();

3.3.1.2. Chương trình TCPEchoClient

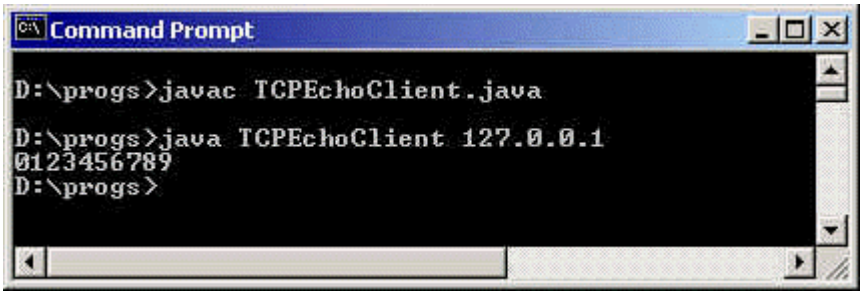
Trên hệ thống UNIX, Dịch vụ Echo được thiết kế theo kiến trúc Client-Server sử dụng Socket làm phương tiện giao tiếp. Cổng mặc định dành cho Echo Server là 7, bao gồm cả hai chế độ có nối kết và không nối kết.

Chương trình TCPEchoClient sẽ nối kết đến EchoServer ở chế độ có nối kết, lần lượt gởi đến Echo Server 10 ký tự từ '0' đến '9', chờ nhận kết quả trả về và hiển thị chúng ra màn hình.

Hãy lưu chương trình sau vào tập tin TCPEchoClient.java

```
import java.io.*;
import java.net.Socket;
public class TCPEchoClient{
public static void main(String args[]){
try {
Socket s = new Socket(args[0],7);// Nối kết đến Server
InputStream is = s.getInputStream();// Lấy InputStream
OutputStream os = s.getOutputStream(); // Lấy OutputStream
for (int i='0'; i<='9';i++){ // Gui '0' ->'9' den EchoServer
os.write(i);// Gởi 1 ký tự sang Server
int ch = is.read();// Chờ nhận 1 ký tự từ Server
System.out.print((char)ch);   // In ký tự nhận được ra màn hình
}
}//try catch(IOException ie){
System.out.println("Loi: Khong tao duoc socket");
}//catch
}//main
}
```

Biên dịch và thực thi chương trình như sau:



Hình 3.12 Thực thi tập tin TCPEchoClient.java

Chương trình này nhận một đối số là địa chỉ IP hay tên miền của máy tính mà ở đó Echo Server đang chạy. Trong hệ thống mạng TCP/IP mỗi máy tính được gán một địa chỉ IP cục bộ là **127.0.0.1** hay có tên là **localhost**. Trong ví dụ trên, chương trình Client nối kết đến Echo Server trên cùng máy với nó.

3.3.2. Xây dựng chương trình Server ở chế độ có nối kết

3.3.2.1. Lớp java.net.ServerSocket

Lớp ServerSocket hỗ trợ các phương thức cần thiết để xây dựng các chương trình Server sử dụng socket ở chế độ có nối kết. Dưới đây là một số phương thức thường dùng để xây dựng Server:

public ServerSocket(int PortNumber);

Phương thức này tạo một Socket với số hiệu cổng là PortNumber mà sau đó Server sẽ lắng nghe trên cổng này.

Ví dụ: Tạo socket cho Server với số hiệu cổng là 7: ServerSocket ss = new ServerSocket(7);

public Socket accept()

Phương thức này lắng nghe yêu cầu nối kết của các Client. Đây là một phương thức hoạt động ở chế độ nghẽn. Nó sẽ bị nghẽn cho đến khi có một yêu cầu nối kết của client gởi đến.

Khi có yêu cầu nối kết của Client gởi đến, nó sẽ chấp nhận yêu cầu nối kết, trả về một Socket là một đầu của kênh giao tiếp ảo giữa Server và Client yêu cầu nối kết.

Ví dụ: Socket ss chờ nhận yêu cầu nối kết: Socket s = ss.accept();

Server sau đó sẽ lấy InputStream và OutputStream của Socket mới **s** để giao tiếp với Client.

3.3.2.2. Xây dựng chương trình Server phục vụ tuần tự

Một Server có thể được cài đặt để phục vụ các Client theo hai cách: phục vụ tuần tự hoặc phục vụ song song.

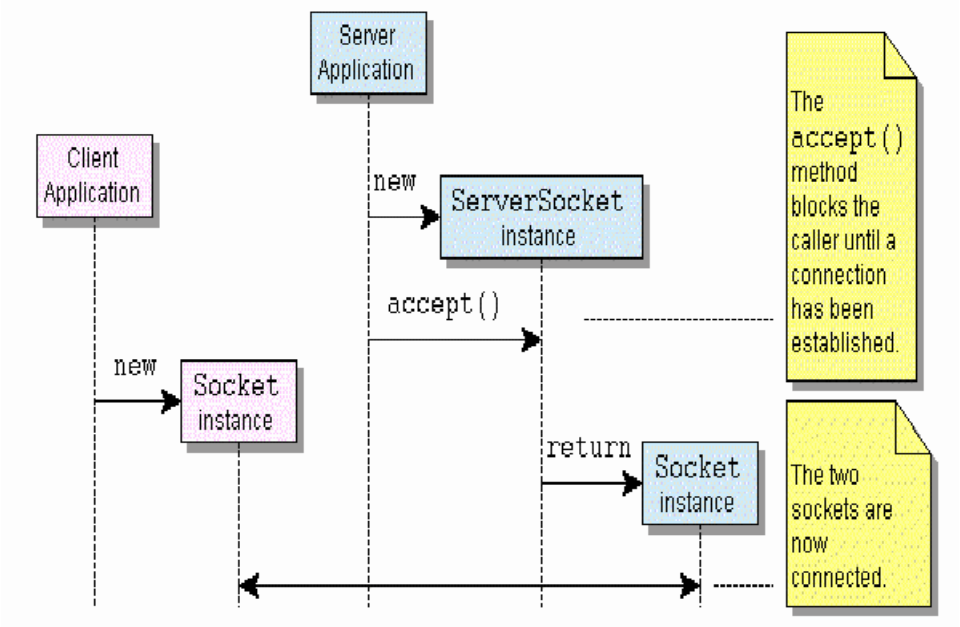
Trong chế độ phục vụ tuần tự, tại một thời điểm Server chỉ chấp nhận một yêu cầu nối kết. Các yêu cầu nối kết của các Client khác đều không được đáp ứng (đưa vào hàng đợi).

Ngược lại trong chế độ phục vụ song song, tại một thời điểm Server chấp nhận nhiều yêu cầu nối kết và phục vụ nhiều Client cùng lúc.

Các bước tổng quát của một Server phục vụ tuần tự

1. Tạo socket và gán số hiệu cổng cho server.
2. Lắng nghe yêu cầu nối kết.
3. Với một yêu cầu nối kết được chấp nhận thực hiện các bước sau:

- Lấy InputStream và OutputStream gắn với Socket của kênh ảo vừa được hình thành.
- Lặp lại công việc sau:
  - Chờ nhận các yêu cầu (công việc).
  - Phân tích và thực hiện yêu cầu.
  - Tạo thông điệp trả lời.
  - Gởi thông điệp trả lời về Client.
  - Nếu không còn yêu cầu hoặc Client kết thúc, đóng Socket và quay lại bước 2.



Hình 3.13 Các bước tổng quát của một Server phục vụ tuần tự

3.3.2.3. Chương trình STCPEchoServer

STCPEchoServer cài đặt một Echo Server phục vụ tuần tự ở chế độ có nối kết. Server lắng nghe trên cổng mặc định số 7.

Hãy lưu chương trình sau vào tập tin STCPEchoServer.java

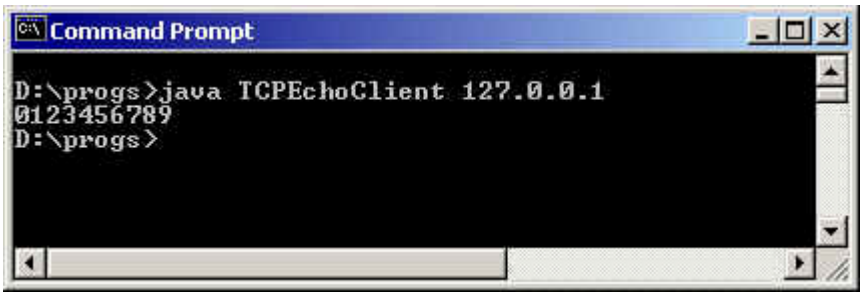
```
import java.net.*;
import java.io.*;
public class STCPEchoServer {
public final static int defaultPort = 7;
public static void main(String[] args) {
try {
ServerSocket ss = new ServerSocket(defaultPort);
while (true) {
try {
Socket s = ss.accept();
OutputStream os = s.getOutputStream();
InputStream is = s.getInputStream();
int ch=0;
while(true) {
ch = is.read();
if(ch == -1) break;
os.write(ch);
```

```
}
s.close();
}
catch (IOException e) {
System.err.println("  Connection Error: "+e);
}
}
}
catch (IOException e) {
System.err.println(" Server Creation Error:"+e);
}
}
}
```

Biên dịch và thực thi chương trình theo cách sau:

Hình 3.14 Thực thi tập tin STCPEchoServer.java

Mở một cửa số DOS khác và thực thi chương trình TCPEchoClient ta có kết quả như sau:



Hình 3.15 Thực thi tập tin TCPEchoServer.java

Hai chương trình này có thể nằm trên hai máy khác nhau. Trong trường hợp đó khi thực hiện chương trình TCPEchoClient phải chú ý nhập đúng địa chỉ IP của máy tính đang chạy chương trình STCPEchoServer.

Xem địa chỉ IP của một máy tính Windows bằng lệnh **ipconfig**.

3.3.2.4. Server phục vụ song song

Các bước tổng quát của một Server phục vụ song song

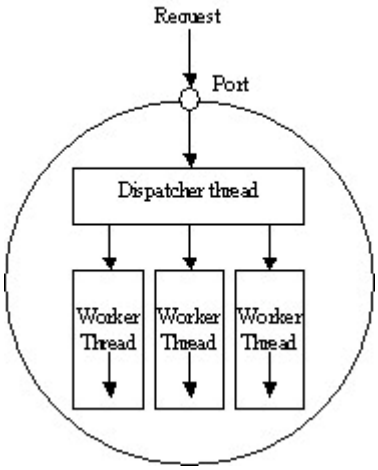
Server phục vụ song song gồm 2 phần thực hiện song song nhau:

- Phần 1: Xử lý các yêu cầu nối kết.
- Phần 2: Xử lý các thông điệp yêu cầu từ khách hàng.

Có cấu trúc như hình sau, trong đó **Phần 1** là (Dispatcher Thread), **Phần 2** là các (Worker Thread)

Phần 1: Lặp lại các công việc sau:

- Lắng nghe yêu cầu nối kết của khách hàng.



Hình 3.16 Server phục vụ song song

- Chấp nhận một yêu cầu nối kết.
  - Tạo kênh giao tiếp ảo mới với khách hàng.
  - Tạo Phần 2 để xử lý các thông điệp yêu cầu của khách hàng.

Phần 2: Lặp lại các công việc sau:

- Chờ nhận thông điệp yêu cầu của khách hàng.
- Phân tích và xử lý yêu cầu.
- Gởi thông điệp trả lời cho khách hàng.

Phần 2 sẽ kết thúc khi kênh ảo bị xóa đi.

Với mỗi Client, trên Server sẽ có một **Phần 2** để xử lý yêu cầu của khách hàng. Như vậy tại một thời điểm bất kỳ luôn tồn tại 1 **Phần 1** và 0 hoặc nhiều **Phần 2**.

Do Phần 2 thực thi song song với Phần 1 cho nên nó được thiết kế là một Thread.

3.3.2.5. Chương trình PTCPEchoServer

PTCPEchoServer cài đặt một Echo Server phục vụ song song ở chế độ có nối kết. Server lắng nghe trên cổng mặc định là 7. Chương trình này gồm 2 lớp:

- Lớp TCPEchoServer, cài đặt các chức năng của Phần 1 - xử lý các yêu cầu nối kết của TCPEchoClient.
- Lớp RequestProcessing, là một Thread cài đặt các chức năng của Phần 2 - Xử lý các thông điệp yêu cầu.

Hãy lưu chương trình sau vào tập tin PTCPEchoServer.java

```
import java.net.*;
import java.io.*;
public class PTCPEchoServer {
public final static int defaultPort = 7; // Cổng mặc định
public static void main(String[] args) {
try {
ServerSocket ss = new ServerSocket(defaultPort); //Tạo socket cho server
while (true) {
try {
Socket s = ss.accept(); // Lắng nghe các yêu cầu nối kết
RequestProcessing rp = new RequestProcessing(s); // Tạo phần xử lý
rp.start(); // Khởi động phần xử lý cho Client hiện tại
}
catch (IOException e) {
System.out.println("Connection Error: "+e);
}
}
}
catch (IOException e) {
System.err.println("Create Socket Error: "+e);
}
}
}

class RequestProcessing extends Thread {
Socket channel; //Socket của kênh ảo nối với Client hiện tại
```

```
public RequestProcessing(Socket s){
channel = s; // Nhận socket của kênh ảo nối với Client
}
public void run() {
try {
OutputStream os = channel.getOutputStream();
InputStream is = channel.getInputStream();
while (true) {
int n = is.read();// Nhận ký tự từ Client
if (n == -1) break;// Thoát nếu kênh ảo bị xóa
os.write(n);//  Gởi ký tự nhận được về Client
}
}
catch (IOException e) {
System.err.println("Request Processing Error: "+e);
}
}
}
```

Biên dịch và thực thi chương trình như sau:

Hình 3.17 Thực thi tập tin PTCPEchoServer.java

Sau đó mở thêm 2 của sổ DOS khác để thực thi chương trình TCPEchoClient nối kết tới PTCPEchoServer. Ta sẽ nhận thấy rằng PTCPEchoServer có khả năng phục vụ đồng thời nhiều Client.

3.3.3. Xây dựng chương trình Client - Server ở chế độ không nối kết

Khi sử dụng socket, ta có thể chọn giao thức UDP cho lớp vận chuyển. UDP viết tắt của User Datagram Protocol, cung cấp cơ chế vận chuyển không bảo đảm và không nối kết trên mạng IP, ngược với giao thức vận chuyển tin cậy, có nối kết TCP.

Cả giao thức TCP và UDP đều phân dữ liệu ra thành các gói tin. Tuy nhiên TCP có thêm vào những tiêu đề (Header) vào trong gói tin để cho phép truyền lại những gói tin thất lạc và tập hợp các gói tin lại theo thứ tự đúng đắn. UDP không cung cấp tính năng này, nếu một gói tin bị thất lạc hoặc bị lỗi, nó sẽ không được truyền lại, và thứ tự đến đích của các gói tin cũng không giống như thứ tự lúc nó được gởi đi.

Tuy nhiên, về tốc độ, UDP sẽ truyền nhanh gấp 3 lần TCP. Cho nên chúng thường được dùng trong các ứng dụng đòi hỏi thời gian truyền tải ngắn và không cần tính chính xác cao, ví dụ truyền âm thanh, hình ảnh . . .

Mô hình client - server sử dụng lớp ServerSocket và Socket ở trên sử dụng giao thức TCP. Nếu muốn sử dụng mô hình client - server với giao thức UDP, ta sử dụng hai lớp java.net.**DatagramSocket** và java.net.**DatagramPacket**.

DatagramSocket được sử dụng để truyền và nhận các DatagramPacket. Dữ liệu được truyền đi là một mảng những byte, chúng được gói vào trong lớp DatagramPacket. Chiều dài của dữ liệu tối đa có thể đưa vào DatagramPacket là khoảng 60.000 byte (phụ thuộc vào dạng đường truyền). Ngoài ra DatagramPacket còn chứa địa chỉ IP và cổng của quá trình gởi và nhận dữ liệu.

Cổng trong giao thức TCP và UDP có thể trùng nhau. Trên cùng một máy tính, bạn có thể gán cổng 20 cho socket dùng giao thức TCP và cổng 20 cho socket sử dụng giao thức UDP.

3.3.3.1. Lớp DatagramPacket

Lớp này dùng để đóng gói dữ liệu gởi đi. Dưới đây là các phương thức thường sử dụng để thao tác trên dữ liệu truyền / nhận qua DatagramSocket.

public DatagramPacket(byte[] b, int n)

- Là phương thức khởi tạo, cho phép tạo ra một DatagramPacket chứa **n** bytes dữ liệu đầu tiên của mảng **b.** (n phải nhỏ hơn chiều dài của mảng b)
- Phương thức trả về một đối tượng thuộc lớp DatagramePacket

Ví dụ: Tạo DatagramPacket để nhận dữ liệu:

```
byte buff[] = new byte[60000];// Nơi chứa dữ liệu nhận
```

được

DatagramPacket inPacket = new Datagrampacket(buff, buff.lenth);

public DatagramPacket(byte[] b, int n, InternetAddress ia, int port)

- Phương thức này cho phép tạo một DatagramPacket chứa dữ liệu và cả địa chỉ của máy nhận dữ liệu.
- Phương thức trả về một đối tượng thuộc lớp DatagramePacket

Các phương thức lấy thông tin trên một DatagramPacket nhận được

Khi nhận được một DatagramPacket từ một quá trình khác gởi đến, ta có thể lấy thông tin trên DatagramPacket này bằng các phương thức sau:

- public synchronized() InternetAddress getAddress() : Địa chỉ máy gởi
- public synchronized() int getPort() : Cổng của quá trình gởi
- public synchronized() byte[] getData() : Dữ liệu từ gói tin
- public synchronized() int getLength() : Chiều dài của dữ liệu trong gói tin

Các phương thức đặt thông tin cho gói tin gởi

Trước khi gởi một DatagramPacket đi, ta có thể đặt thông tin trên DatagramPacket này bằng các phương thức sau:

- public synchronized() void setAddress(IntermetAddress dis) : Đặt địa chỉ máy nhận.
- public synchronized() void setPort(int port) : Đặt cổng quá trình nhận
- public synchronized() void setData(byte buffer[]) : Đặt dữ liệu gởi
- public synchronized() void setLength(int len) : Đặt chiều dài dữ liệu gởi

3.3.3.2. Lớp DatagramSocket

Lớp này hỗ trợ các phương thức sau để gởi / nhận các DatagramPacket

public DatagramSocket() throws SocketException

- Tạo Socket kiểu không nối kết cho Client. Hệ thống tự động gán số hiệu cổng chưa sử dụng cho socket.

Ví dụ: Tạo một socket không nối kết cho Client:

```
try{
    DatagramSocket ds = new DatagramSocket();
   } catch(SocketException se) {
  System.out.print("Create DatagramSocket Error: "+se);
```

}

public DatagramSocket(int port) throws SocketException

- Tạo Socket kiểu không nối kết cho Server với số hiệu cổng được xác định trong tham số (port).

Ví dụ: Tạo một socket không nối kết cho Server với số hiệu cổng là 7:

```
try{
    DatagramSocket dp = new DatagramSocket(7);
  } catch(SocketException se) {
    System.out.print("Create DatagramSocket Error: "+se);
  }
```

public void send(DatagramPacket dp) throws IOException

- Dùng để gởi một DatagramPacket đi.

Ví dụ: Gởi chuỗi "My second UDP Packet", cho quá trình ở địa chỉ www.cdndalat.edu.vn, cổng nhận là 19:

```
try {
   DatagramSocket ds = new DatagramSocket(); //Tạo Socket
   //Địa chỉ Internet của máy nhận
  InetAddress ia = InetAddess.getByName("www.cdndalat.edu.vn");
  int port = 19; //      Cổng của quá trình  nhận
  String s = "My second UDP Packet";// Dữ liệu cần gởi
   byte[] b = s.getBytes();// Đổi sang mảng bytes
  // Tạo gói tin
  DatagramPacket outPacket = new DatagramPacket(b, b.length, ia, port);
  ds.send(outPacket); // Gởi gói tin đi
  }
  catch (IOException e) { System.err.println(e);
  }
```

public synchronized void receive(Datagrampacket dp) throws IOException

- Chờ nhận một DatagramPacket. Quá trình sẽ bị nghẽn cho đến khi có dữ liệu đến.

  Ví dụ:

```
try {
DatagramSocket ds = new DatagramSocket(); //Tạo Socket
byte[] b = new byte[60000];// Nơi chứa dữ liệu nhận được
DatagramPacket inPacket = new DatagramPacket(b, b.length);  // Tạo gói tin ds.receive(inPacket); // Chờ nhận gói tin
}
catch (IOException e) { System.err.println(e);
}
```

3.3.3.3. Chương trình UDPEchoServer

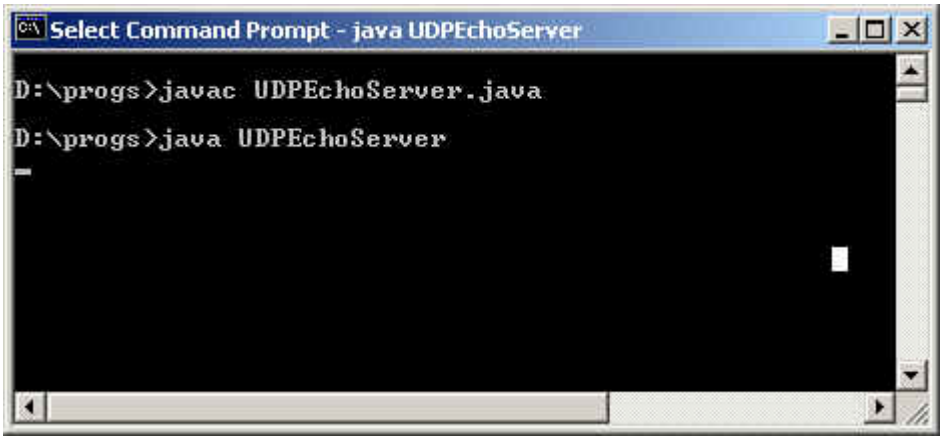
Chương trình UDPEchoServer cài đặt Echo Server ở chế độ không nối kết, cổng mặc định là 7. Chương trình chờ nhận từng gói tin, lấy dữ liệu ra khỏi gói tin nhận được và gởi ngược dữ liệu đó về Client.

Lưu chương trình sau vào tập tin UDPEchoServer.java

```
import java.net.*;
import java.io.*;
public class UDPEchoServer {
public final static int port = 7; // Cổng mặc định của Server public static void main(String[] args) {
try {
DatagramSocket ds = new DatagramSocket(port); // Tạo Socket với cổng là 7 byte[] buffer = new byte[6000]; // Vùng đệm chứa dữ liệu cho gói tin nhận while(true) {// Tạo gói tin nhận
DatagramPacket incoming = new DatagramPacket(buffer,buffer.length);
ds.receive(incoming); // Chờ nhận gói tin gởi đến
// Lấy dữ liệu khỏi gói tin nhận
String theString = new String(incoming.getData(),0,incoming.getLength());
// Tạo gói tin gởi chứa dữ liệu vừa nhận được
DatagramPacket outsending = new DatagramPacket(theString.getBytes(), incoming.getLength(),incoming.getAddress(), incoming.getPort()); ds.send(outsending);
}
```

```
}
catch (IOException e) { System.err.println(e);
}
}
}
```

Biên dịch và thực thi chương trình như sau



Hình 3.18 Thực thi tập tin UDPEchoServer.java

3.3.3.4. Chương trình UDPEchoClient

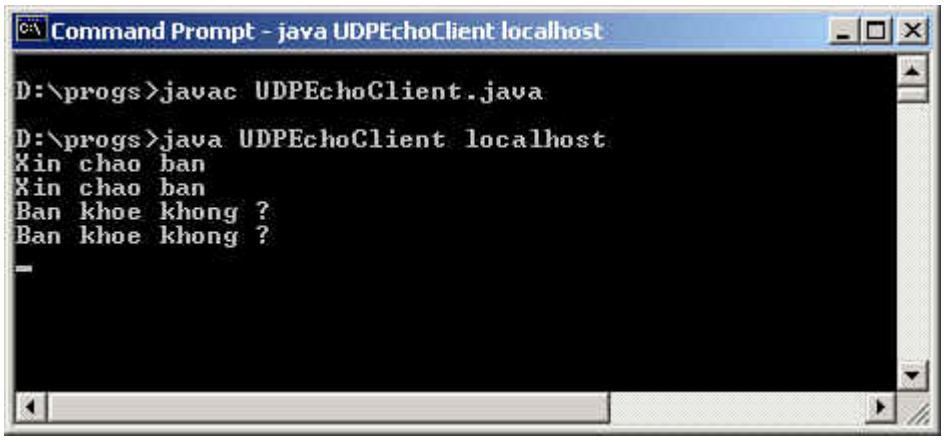
Chương trình này cho phép người sử dụng nhận các chuỗi từ bàn phím, gởi chuỗi sang EchoServer ở chế độ không nối kết ở cổng số 7, chờ nhận và in dữ liệu từ Server gởi về ra màn hình.

Lưu chương trình sau vào tập tin UDPEchoClient.java

```
import java.net.*;
import java.io.*;
public class UDPEchoClient extends Object{
public final static int serverPort = 7; // Cổng mặc định của Echo Server public static void main(String[] args) {
try {
if (args.length ==0) { // Kiểm tra tham số, là địa chỉ của Server
System.out.print("Syntax: java UDPClient HostName");
return;
}
DatagramSocket ds = new DatagramSocket();// Tạo DatagramSocket
InetAddress server = InetAddress.getByName(args[0]);    // Địa chỉ Server while(true) {
```

```
InputStreamReader isr = new InputStreamReader(System.in);// Nhập
BufferedReader br = new BufferedReader(isr);// một chuỗi String
theString = br.readLine();// từ bàn phím
byte[] data = theString.getBytes();// Đổi chuỗi ra mảng bytes
// Tạo gói tin gởi
DatagramPacket dp = new DatagramPacket(data,data.length,server,
serverPort);
ds.send(dp); // Send gói tin sang Echo Server
byte[] buffer = new byte[6000];// Vùng đệm cho dữ liệu nhận
// Gói tin nhận
DatagramPacket incoming = new DatagramPacket(buffer,
buffer.length);
ds.receive(incoming); // Chờ nhận dữ liệu từ EchoServer gởi về
// Đổi dữ liệu nhận được dạng mảng bytes ra chuỗi và in ra màn hình
System.out.println(new String(incoming.getData(), 0,
incoming.getLength()));
}
}
catch (IOException e) { System.err.println(e);
}
}
}
```

Biên dịch và thực thi chương trình như sau:



Hình 3.19 Thực thi tập tin UDPEchoClient.java

Chú ý, khi thực hiện chương trình UDPEchoClient phải đưa vào đối số là

địa chỉ của máy tính đang thực thi chương trình UDPEchoServer. Trong ví dụ trên, Server và Client cùng chạy trên một máy nên địa chỉ của UDPEchoServer là ***localhost*** (hay 127.0.0.1). Nếu UDPEchoServer chạy trên máy tính khác thì khi thực thi, ta phải biết được địa chỉ IP của máy tính đó và cung cấp vào đối số của chương trình. Chẳng hạn, khi UDPEchoServer đang phục vụ trên máy tính ở địa chỉ 172.18.250.211, ta sẽ thực thi UDPEchoClient theo cú pháp sau:

```
java UDPEchoClient 172.18.250.211
```

3.4. Bài tập áp dụng

Chủ đề 1: Client ở chế độ có nối kết

- **Mục đích**:

Viết các chương trình Client nối kết đến các server theo các Protocol chuẩn.

- **Yêu cầu**

Sinh viên thực hiện các bài tập sau

Bài 1: Viết chương trình nhận đối số là một URL. Nối kết đến Web Server trong URL nhận được, lấy trang web về và in ra màn hình theo dạng textfile (html).

Chủ đề 2: Client - Server chế độ có nối kết

- **Mục đích**:

Viết các chương trình Client -Server theo chế độ có nối kết.

- **Yêu cầu**

Sinh viên thực hiện các bài tập sau, với mỗi bài tập hãy thiết kế một Server phục vụ ở chế độ tuần tự và một Server phục vụ ở chế độ song song.

Bài 1: Viết chương trình theo mô hình Client-Server sử dụng dụng Socket ở chế độ có nối kết. Trong đó :

+ Server làm nhiệm vụ đọc một ký tự số từ '0' đến '9'. (Ví dụ : nhận số 0 : trả về "khong" , 1 : trả về "một" ; ... ... 9 : trả về "chín", nếu nhận ký tự khác số thì trả về "Không phải số nguyên").

+ Client sẽ nhập vào 1 ký tự, gửi qua Server, nhận kết quả trả về từ Server và thể hiện lên màn hình

Bài 2: Viết chương trình theo mô hình Client-Server sử dụng Socket ở chế độ có nối kết. Trong đó :

+ Server sẽ nhận các yêu cầu là một chuỗi có khuôn dạng như sau:

"OP Operant1 Operant2\n"

Trong đó:

- OP là một ký tự chỉ phép toán muốn thực hiện: '+','-', '*', '/'.

- Operant1, Operant2 là đối số của phép toán.

- Các thành phần trên cách nhau bởi 1 ký tự trắng ' '.

- Kết thúc yêu cầu bằng ký tự xuống dòng '\n'.

Mỗi khi server nhận được một thông điệp nó sẽ thực hiện phép toán: Operant1 OP Operant2 để cho ra kết quả, sau đó đổi kết quả thành

chuỗi và gởi về Client.

+ Client cho phép người dùng nhập các phép toán muốn tính theo cách thức thông thường. Ví dụ: 100+200. Client tạo ra thông điệp yêu cầu theo đúng dạng do Server qui định, mô tả về phép toán muốn Server thực thi, rồi gởi sang Server, chờ nhận kết quả trả về và in ra màn hình.

Chủ đề 3: Client-Server ở chế độ không nối kết

- **Mục đích**:

Viết các chương trình Client -Server theo chế độ không nối kết.

- **Yêu cầu**

Bài 1 : Viết chương trình Talk theo chế độ không nối kết. Cho phép hai người ngồi trên hai máy tính có thể tán gẫu (chat) với nhau.

BÀI 4: KẾT NỐI VỚI WEB SERVERS

(Remote Procedure Call) và RMI (Method Invocation)

Mã bài : MĐ 38.4

Mục tiêu:

- Trình bày được cách lập trình sử dụng các Giao thức để truy cập với máy chủ Web (Web Server).
- Xây dựng các ứng dụng làm việc với máy chủ Web (WebServer).
- Thực hiện các thao tác an toàn với máy tính.

4.1. Lời gọi thủ tục xa (RPC- Remote Procedure Call)

4.1.1. Giới thiệu

Lời gọi thủ tục xa là một cơ chế cho phép một chương trình có thể gọi thực thi một thủ tục (hay hàm) trên một máy tính khác. Trong chương trình lúc này, tồn tại hai loại thủ tục: thủ tục cục bộ và thủ tục ở xa.

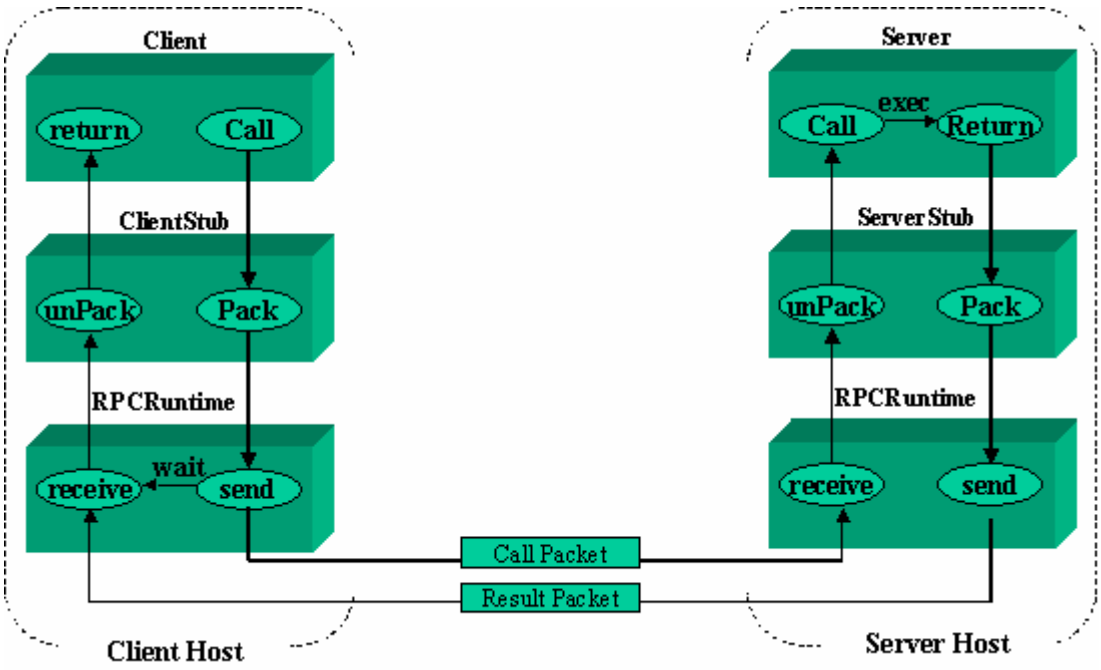
- Thủ tục cục bộ là thủ tục được định nghĩa, cài đặt và thực thi tại máy của chương trình.
- Thủ tục ở xa là thủ tục được định nghĩa, cài đặt và thực thi trên một máy tính khác.

Cú pháp giữa lời gọi thủ tục cục bộ và ở xa thì giống nhau. Tuy nhiên, khi một thủ tục ở xa được gọi đến, một thành phần của chương trình gọi là **Stub** sẽ chuyển hướng để kích hoạt một thủ tục tương ứng nằm trên một máy tính khác với máy của chương trình gọi. Đối với người lập trình, việc gọi thủ tục xa và thủ tục cục bộ thì giống nhau về mặt cú pháp. Đây chính là cơ chế cho phép đơn giản hóa việc xây dựng các ứng dụng Client- Server. Trong hệ thống RPC, Server chính là máy tính cung cấp các thủ tục ở xa cho phép các chương trình trên các máy tính khác gọi thực hiện. Client chính là các chương trình có thể gọi các thủ tục ở xa trong quá trình tính toán của mình.

Một Client có thể gọi thủ tục ở xa của nhiều hơn một máy tính. Như vậy sự thực thi của chương trình Client lúc này không còn gói gọn trên một máy tính của Client mà nó trải rộng trên nhiều máy tính khác nhau. Đây chính là mô hình của ứng dụng phân tán (Distributed Application).

4.1.2. Kiến trúc của chương trình Client-Server cài đặt theo cơ chế lời gọi thủ tục xa

Một ứng dụng Client-Server theo cơ chế RPC được xây dựng gồm có sáu phần như sơ đồ dưới đây:



Hình 4.1 Kiến trúc chương trình kiểu RPC

Phần Client là một quá trình người dùng, nơi khởi tạo một lời gọi thủ tục từ xa. Mỗi lời gọi thủ tục ở xa trên phần Client sẽ kích hoạt một thủ tục cục bộ tương ứng nằm trong phần Stub của Client.

Phần ClientStub cung cấp một bộ các hàm cục bộ mà phần Client có thể gọi. Mỗi một hàm của ClientStub đại diện cho một hàm ở xa được cài đặt và thực thi trên Server.

Mỗi khi một hàm nào đó của ClientStub được gọi bởi Client, ClientStub sẽ đóng gói một thông điệp để mô tả về thủ tục ở xa tương ứng mà Client muốn thực thi cùng với các tham số nếu có. Sau đó nó sẽ nhờ hệ thống RPCRuntime cục bộ gởi thông điệp này đến phần Server Stub của Server.

Phần RPCRuntime quản lý việc truyền thông điệp thông qua mạng giữa máy Client và máy Server. Nó đảm nhận việc truyền lại, báo nhận, chọn đường gói tin và mã hóa thông tin.

RPCRuntime trên máy Client nhận thông điệp yêu cầu từ ClientStub, gởi nó cho RPCRuntime trên máy Server bằng lệnh send(). Sau đó gọi lệnh wait() để chờ kết quả trả về từ Server.

Khi nhận được thông tiệp từ RPCRuntime của Client gởi sang, RPCRuntime bên phía server chuyển thông điệp lên phần ServerStub.

ServerStub mở thông điệp ra xem, xác định hàm ở xa mà Client muốn thực hiện cùng với các tham số của nó. ServerStub gọi một thủ tục tương ứng nằm trên phần Server.

Khi nhận được yêu cầu của ServerStub, Server cho thực thi thủ tục được yêu

cầu và gởi kết quả thực thi được cho ServerStub.

ServerStub đóng gói kết quả thực trong một gói tin trả lời, chuyển cho phần RPCRuntime cục bộ để nó gởi sang RPCRuntime của Client .

RPCRuntime bên phía Client chuyển gói tin trả lời nhận được cho phần ClientStub. ClientStub mở thông điệp chứa kết quả thực thi về cho Client tại vị trí phát ra lời gọi thủ tục xa.

Trong các thành phần trên, RPCRuntime được cung cấp bởi hệ thống. ClientStub và ServerStub có thể tạo ra thủ công (phải lập trình) hay có thể tạo ra bằng các công cụ cung cấp bởi hệ thống.

Cơ chế RPC được hỗ trợ bởi hầu hết các hệ điều hành mạng cũng như các ngôn ngữ lập trình.

4.2. Kích hoạt phương thức xa (RMI- Remote Method Invocation)

4.2.1. Giới thiệu

RMI là một sự cài đặt cơ chế RPC trong ngôn ngữ lập trình hướng đối tượng Java. Hệ thống RMI cho phép một đối tượng chạy trên một máy ảo Java này có thể kích hoạt một phương thức của một đối tượng đang chạy trên một máy ảo Java khác. Đối tượng có phương thức được gọi từ xa gọi là các đối tượng ở xa (Remote Object).

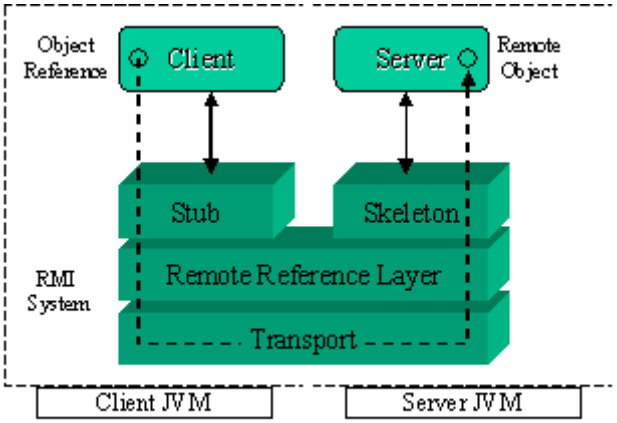
Một ứng dụng RMI thường bao gồm 2 phần phân biệt: Môt chương trình Server và một chương trình Client.

- Chương trình Server tạo một số các Remote Object, tạo các **tham chiếu** (reference) đến chúng và chờ những chương trình Client kích hoạt các phương thức của các Remote Object này.
- Chương trình Client lấy một **tham chiếu** đến một hoặc nhiều Remote Object trên Server và kích hoạt các phương thức từ xa thông qua các tham chiếu.

Một chương trình Client có thể kích hoạt các phương thức ở xa trên một hay nhiều Server. Tức là sự thực thi của chương trình được trải rộng trên nhiều máy tính. Đây chính là đặc điểm của các ứng dụng phân tán. Nói cách khác, RMI là cơ chế để xây dựng các ứng dụng phân tán dưới ngôn ngữ Java.

4.2.2. Kiến trúc của chương trình Client-Server theo cơ chế RMI

Kiến trúc một chương trình Client-Server theo cơ chế RMI được mô tả như hình dưới đây:

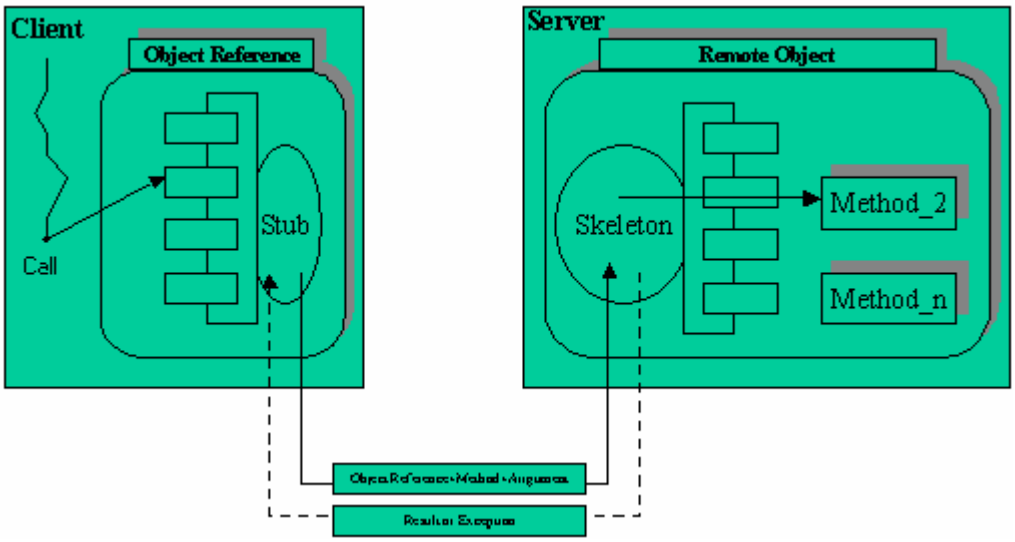

Object
Reference
Client
Server
Remote
Object
Stub
Skeleton
RMI
System
Remote Reference Layer
Transport
Client JVM
Server JVM

Hình 4.2 Kiến trúc chương trình kiểu RMI

Trong đó:

- Server là chương trình cung cấp các đối tượng có thể được gọi từ xa.
- Client là chương trình có tham chiếu đến các phương thức của các đối tượng ở xa trên Server.
- Stub chứa các tham chiếu đến các phương thức ở xa trên Server.
- Skeleton đón nhận các tham chiếu từ Stub để kích hoạt phương thức tương ứng trên Server.
- Remote Reference Layer là hệ thống truyền thông của RMI.

Con đường kích hoạt một phương thức ở xa được mô tả như hình dưới đây:



Hình 4.3 Cơ chế hoạt động của RMI

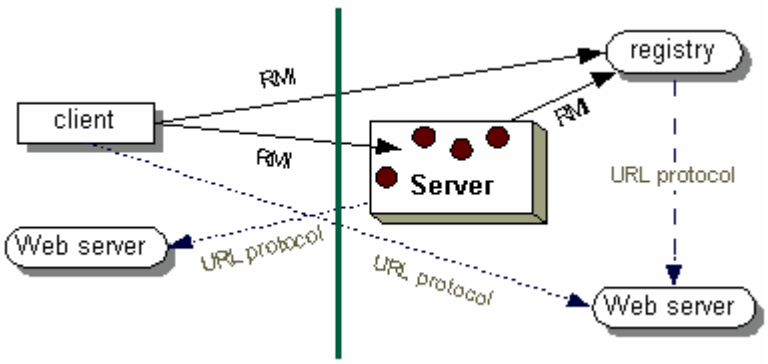
4.2.3. Các cơ chế liên quan trong một ứng dụng đối tượng phân tán

Trong một ứng dụng phân tán cần có các cơ chế sau:

- Cơ chế định vị đối tượng ở xa (Locate remote objects): Cơ chế này xác định cách thức mà chương trình Client có thể lấy được tham chiếu (Stub) đến các đối tượng ở xa. Thông thường người ta sử dụng một Dịch vụ danh bạ (Naming Service) lưu giữ các tham khảo đến các đối tượng cho phép gọi từ xa mà Client sau đó có thể tìm kiếm.
- Cơ chế giao tiếp với các đối tượng ở xa (Communicate with remote objects): Chi tiết của cơ chế giao tiếp với các đối tượng ở xa được cài đặt bởi hệ thống RMI.
- Tải các lớp dạng bytecodes cho các lớp mà nó được chuyển tải qua lại giữa Máy ảo (Load class bytecodes for objects that are passed around): Vì RMI cho phép các chương trình gọi phương thức từ xa trao đổi các đối tượng với các phương thức ở xa dưới dạng các tham số hay giá trị trả về của phương thức, nên

RMI cần có cơ chế cần thiết để tải mã Bytecodes của các đối tượng từ máy ảo này sang máy ảo khác.

Hình dưới đây mô tả một ứng dụng phân tán dưới RMI sử dụng dịch vụ danh bạ để lấy các tham khảo của các đối tượng ở xa. Trong đó:



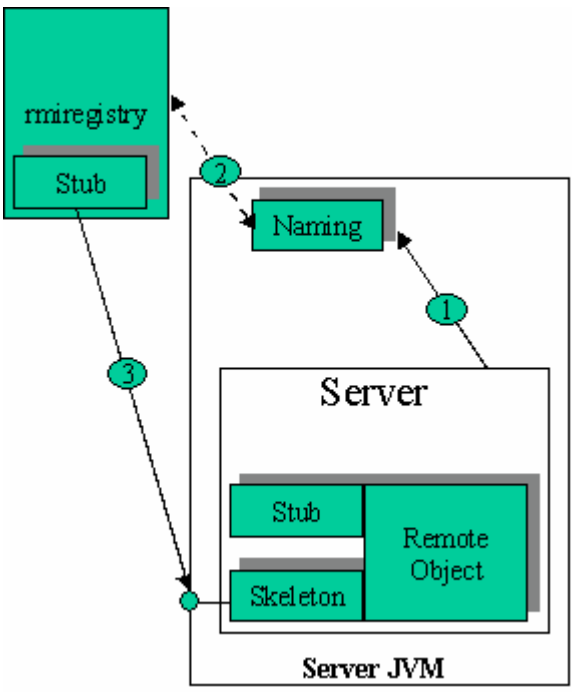
Hình 4.4 Vai trò của dịch vụ tên

- Server đăng ký tên cho đối tượng có thể được gọi từ xa của mình với Dịch vụ danh bạ (Registry Server).
- Client tìm đối tượng ở xa thông qua tên đã được đăng ký trên Registry Server (looks up) và tiếp đó gọi các phương thức ở xa.
- Hình minh họa cũng cho thấy cách thức mà hệ thống RMI sử dụng một WebServer sẵn có để truyền tải mã bytecodes của các lớp qua lại giữa Client và Server.

4.2.4. Cơ chế vận hành của của một ứng dụng Client-Server theo kiểu RMI

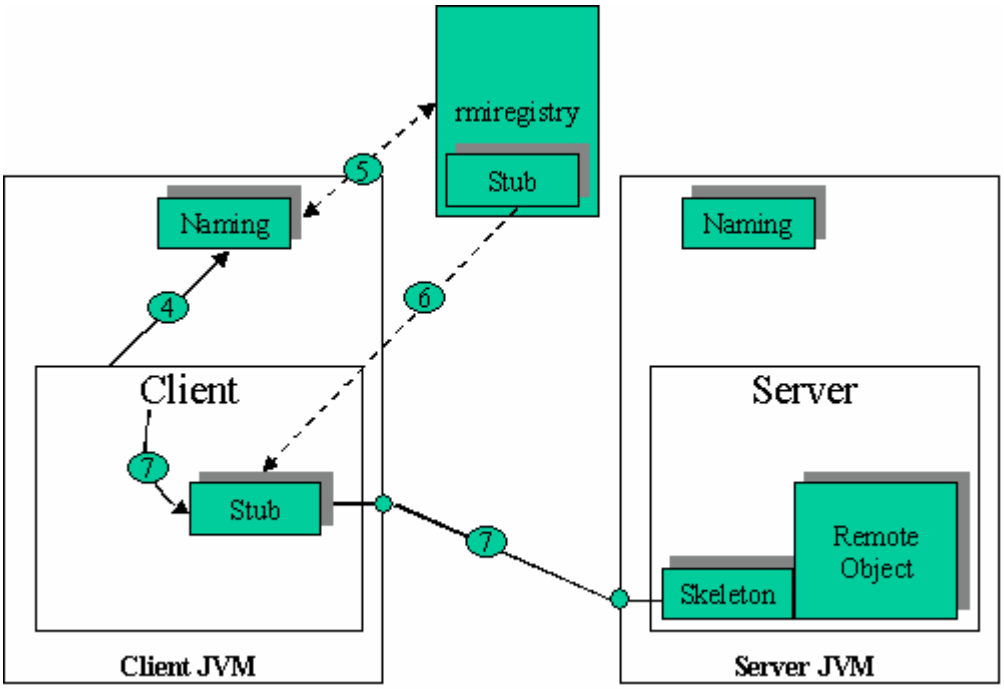
Tiến trình vận hành của một ứng dụng Client-Server theo kiểu RMI diễn ra như sau:

- Bước 1: Server tạo ra các đối tượng cho phép gọi từ xa cùng với các Stub và Skeleton của chúng.
- Bước 2: Server sử dụng lớp Naming để đăng ký tên cho một



đối tượng từ xa (1).

- Bước 3: Naming đăng ký Stub của đối tượng từ xa với Registry Server (2).
- Bước 4: Registry Server sẵn sàng cung cấp tham thảo đến đối tượng từ xa khi có yêu cầu (3).
- Client yêu cầu Naming định vị đối tượng xa qua tên đã được đăng ký (phương thức lookup) với dịch vụ tên (4).
- Naming tải Stub của đối tượng xa từ dịch vụ tên mà đối tượng xa đã đăng ký về Client (5).
- Cài đặt đối tượng Stub và trả về tham khảo đối tượng xa cho Client (6). Hình 4.5 Bước 1, bước 2
- Client thực thi một lời gọi phương thức xa thông qua đối tượng Stub (7).



Hình 4.6 Bước 3, bước 4

4.2.5. Các lớp hỗ trợ chương trình theo kiểu Client-Server trong Java

Java hỗ trợ các lớp cần thiết để cài đặt các ứng dụng Client-Server theo

kiểu RMI trong các gói: java.rmi. Trong số đó các lớp thường được dùng là:

- java.rmi.Naming
- java.rmi.RMISecurityManager
- java.rmi.RemoteException;
- java.rmi.server.RemoteObject
- java.rmi.server.RemoteServer
- java.rmi. server.UnicastRemoteObject

4.3. Xây dựng một ứng dụng phân tán với RMI

Xây dựng một ứng dụng phân tán bằng cơ chế RMI gồm các bước sau:

1. Thiết kế và cài đặt các thành phần của ứng dụng.
2. Biên dịch các chương trình nguồn và tạo ra Stub và Skeleton.
3. Tạo các lớp có thể truy xuất từ mạng cần thiết.
4. Khởi tạo ứng dụng

4.3.1. Thiết kế và cài đặt các thành phần của ứng dụng.

Đầu tiên bạn phải xác định lớp nào là lớp cục bộ, lớp nào là lớp được gọi từ

xa. Nó bao gồm các bước sau:

- *Định nghĩa các giao diện cho các phương thức ở xa (remote interfaces)*: Một remote interface mô tả các phương thức mà nó có thể được kích hoạt từ xa bởi các Client. Đi cùng với việc định nghĩa Remote Interface là việc xác định các lớp cục bộ làm tham số hay giá trị trả về của các phương thức được gọi từ xa.
- *Cài đặt các đối tượng từ xa (remote objects)*: Các Remote Object phải cài đặt cho một hoặc nhiều Remote Interfaces đã được định nghĩa. Các lớp của Remote Object class cài đặt cho các phương thức được gọi từ xa đã được khai báo trong Remote Interface và có thể định nghĩa và cài đặt cho cả các phương thức được sử dụng cục bộ. Nếu có các lớp làm đối số hay giá trị trả về cho các phương thức được gọi từ xa thì ta cũng định nghĩa và cài đặt chúng.
- *Cài đặt các chương trình Client*: Các chương trình Client có sử dụng các Remote Object có thể được cài đặt ở bất kỳ thời điểm nào sau khi các Remote Interface đã được định nghĩa.

4.3.2. Biên dịch các tập tin nguồn và tạo Stubs và Skeleton

Giai đoạn này gồm 2 bước: Bước thứ nhất là dùng chương trình biên dịch javac để biên dịch các tập tin nguồn như các remote interface, các lớp cài đặt cho các remote interface, lớp server, lớp client và các lớp liên quan khác. Kế tiếp ta dùng trình biện dịch rmic để tạo ra stub và skeleton cho các đối tượng từ xa từ các lớp cài đặt cho các remote interface.

4.3.3. Tạo các lớp có thể truy xuất từ mạng

Tạo một tập tin chứa tất cả các file có liên quan như các remote interface stub, các lớp hỗ trợ mà chúng cần thiết phải tải về Client và làm cho tập tin này có thể truy cập đến thông qua một Web server.

4.3.4. Thực thi ứng dụng

Thực thi ứng dụng bao gồm việc thực thi rmiregistry server, thực thi server, và thực thi client.

Tóm lại các công việc phải làm là:

- Tạo giao diện (interface) khai báo các phương thức được gọi từ xa của đối tượng.
- Tạo lớp cài đặt (implement) cho giao diện đã được khai báo.
- Viết chương trình Server.
- Viết chương trình Client.
- Dịch các tập tin nguồn theo dạng RMI để tạo ra các lớp tương ứng và stub cho client, skeleton cho server.
- Khởi động dịch vụ registry.
- Thực hiện chương trình Server.
- Thực thi chương trình Client.

4.3.4. Ví dụ minh họa

Trong ví dụ này chúng ta định nghĩa một phương thức String sayHello() được gọi từ xa. Mỗi khi phương thức này được kích hoạt nó sẽ trả về chuỗi "Hello World" cho Client gọi nó.

Dưới đây là các bước để xây dựng ứng dụng:

Bước 01: Tạo giao diện (interface) khai báo các phương thức được gọi từ

xa của đối tượng.

- Cú pháp tổng quát:

```
import java.rmi.Remote;
```

```
import java.rmi.RemoteException;
public interface InterfaceName extends Remote {
ReturnType remoteMethodOne() throws RemoteException;
ReturnType remoteMethodTwo() throws RemoteException;
….
}
```

- Định nghĩa remote interface có tên là HelloItf, có phương thức được gọi từ xa là String sayHello() như sau:

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface HelloItf extends Remote { String sayHello() throws RemoteException;
}
```

Lưu chương trình này vào tập tin HelloItf.java

Bước 02: Tạo lớp cài đặt (implement) cho giao diện đã được khai báo:

- Cú pháp tổng quát:

```
import java.rmi. server.UnicastRemoteObject;
import java.rmi.RemoteException;
public class RemoteClass extends UnicastRemoteObject implements
InterfaceName {
public RemoteClass() throws RemoteException {
super();
..... // Implement of Method
}
public ReturnType remoteMethodOne() throws RemoteException {
..... // Implement of Method
}
public ReturnType remoteMethodTwo() throws RemoteException {
..... // Definition of Method
}
}
```

- Định nghĩa lớp cài đặt có tên là Hello cài đặt cho remote interface

HelloItf

```
import java.rmi. server.UnicastRemoteObject;
import java.rmi.RemoteException;
public class Hello extends UnicastRemoteObject implements HelloItf
{
public Hello() throws RemoteException {
super();
}
public String sayHello() {
return "Hello World !";
}
}
```

Lưu chương trình này vào tập tin Hello.java

Bước 03: Viết chương trình Server:

- Cú pháp tổng quát:

```
import java.rmi.Naming;
import java.rmi.RemoteException; import java.rmi.RMISecurityManager; public class ServerName {
public static void main(String args[]) {
if (System.getSecurityManager() == null) { // Cài đặt cơ chế bảo mật
System.setSecurityManager(new RMISecurityManager());
}
try {
// Tạo các đối tượng từ xa
RemoteClass remoteObject = new RemoteClass();
// Đăng ký tên cho các đối tượng từ xa
Naming.rebind("RegistryName", remoteObject);
...
}
catch (Exception e) { System.out.println("Error: . . ." + e);
}
}
```

```
}
```

- Tạo server có tên HelloServer chứa một đối tượng từ xa obj thuộc lớp cài đặt Hello. Đăng ký tên cho đối tượng obj là HelloObject

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.RMISecurityManager;
public class HelloServer {
   public static void main(String args[]) {
      if (System.getSecurityManager() == null) {
         System.setSecurityManager(new RMISecurityManager());
      }
try {
            Hello obj = new Hello();
            Naming.rebind("HelloObject", obj);
            System.out.println("HelloObject is registried");
      }
catch (Exception e) {
         System.out.println("Error: " + e);
      }
   }
}
```

Lưu chương trình này vào tập tin HelloServer.java

Bước 04: Viết chương trình Client:

- Cú pháp tổng quát:

```
import java.rmi.Naming;
import java.rmi.RemoteException;
public class Client {
public static void main(String args[]) {
String remoteObjectURL = "rmi://NameServer/RegistryName”;
Interfacename object = null;
try {
object = (InterfaceName)Naming.lookup(remoteObjectURL);
```

```
object.remoteMethodOne();
...
}
catch (Exception e) {
System.out.println(" Error: ”+ e);
}
}
}
```

- Tạo client có tên là HelloClient, tìm đối tượng HelloObject trên rmiregistry chẳng hạn tại địa chỉ 172.18.211.160. Gọi phương thức sayHello() và in kết quả trả về ra màn hình.

```
import java.rmi.Naming;
import java.rmi.RemoteException;
public class HelloClient {
public static void main(String args[]) {
String helloURL = "rmi://172.18.211.160/HelloObject";
HelloItf object = null;
try {
object = (HelloItf)Naming.lookup( helloURL); String message = object.sayHello(); System.out.println(message);
}
catch (Exception e) { System.out.println("Client Error :" + e);
}
}
}
```

Lưu chương trình vào tập tin HelloClient.java

Bước 05: Dịch các tập tin nguồn theo dạng RMI để tạo ra các lớp tương ứng và stub cho client, skeleton cho server:

- Cú pháp tổng quát:

javac InterfaceName.java RemoteClass.java Server.java Client.java

(Tạo ra các lớp InterfaceName.class RemoteClass.class Server.class Client.class)

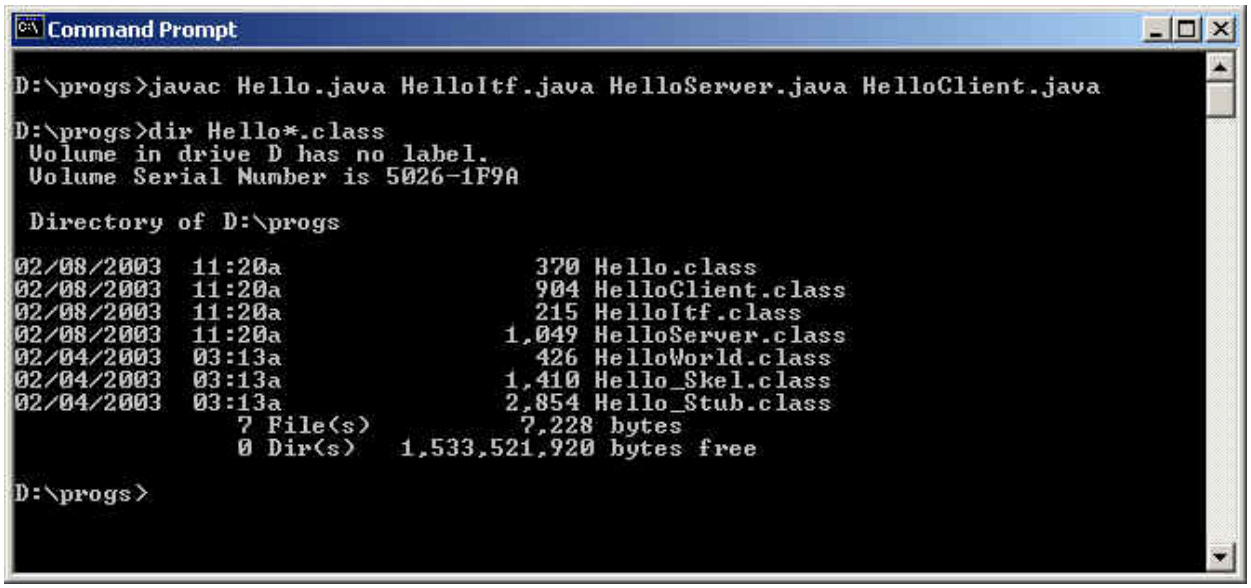
rmic RemoteClass

(Tạo ra các lớp cho Skeleton và Stub: RemoteClass_Skel.class RemoteClass_Stub.class)

- Biên dịch các lớp trong Hello:

javac Hello.java HelloItf.java HelloServer.java HelloClient.java

rmic Hello.class



Hình 4.7 Biên dịch các lớp trong Hello

Bước 06: Khởi động dịch vụ rmiregistry

- Cú pháp tổng quát: start rmiregistry [port] Cổng mặc định là 1099.
- Khởi động dịch vụ rmiregistry trên cổng mặc định như sau:

Hình 4.8 Khởi động dịch vụ rmiregistry trên cổng mặc định

Khi đó rmiregistry server sẽ chạy trên một cửa sổ mới, giữ nguyên cửa sổ này, không đóng nó lại.

Hình 4.9 rmiregistry server chạy trên một cửa sổ mới

Bước 07: Thực hiện chương trình Server

- Cú pháp tổng quát:

java -Djava.security.policy =UrlOfPolicyFile ServerName

Trong đó UrlOfPolicyFile là địa chỉ theo dạng URL của tập tin mô tả chính sách về bảo mật mã nguồn của Server (policy file). Nó qui định "ai" (chương trình, máy tính, quá trình trên) sẽ có quyền download các tập tin của nó trong đó có stub. Để đơn giản trong phần này ta cho phép tất cả mọi người đều có quyền download các tập tin của Server. Khi triển khai các ứng dụng thật sự ta phải có các chính sách bảo mật nghiêm ngặt hơn (Tham khảo tài liệu về Security của Java). File policy có dạng như sau:

```
grant {
// Allow everything for now
permission java.security.AllPermission;
};
```

Lưu nội dung trên vào tập tin có tên **policy.java**

- Thực thi HelloServer với địa tập tin plolicy nằm ở thư mục D:\progs\policy.java

Hình 4.10 Thực thi HelloServer với địa tập tin plolicy nằm ở thư mục D:\progs\policy.java

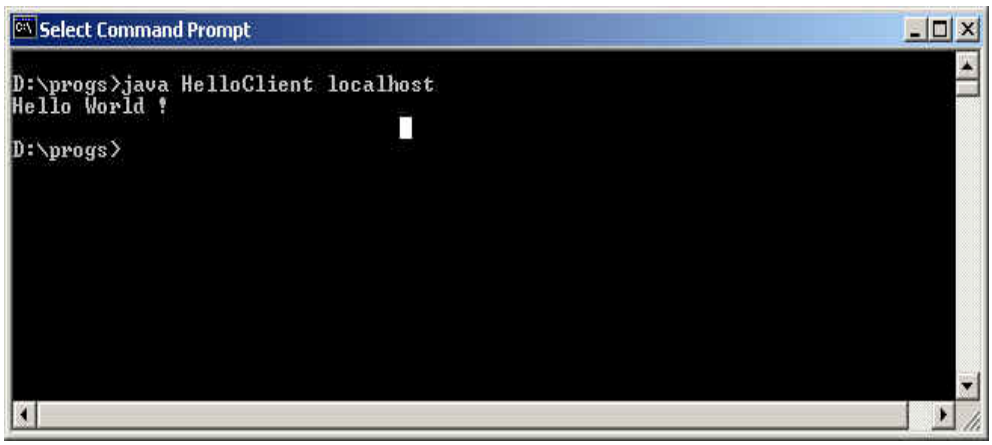
Bước 08: Thực thi chương trình Client:

- Cú pháp tổng quát

  java ClientName

- Thực thi HelloClient với địa chỉ của rmiregistry đưa vào trong tham số

Để thực thi được chương trình HelloClient cần có hai class nằm cùng thư mục với nó là HelloItf.class và Hello_Stub.class.



Hình 4.11 Thực thi HelloClient với địa chỉ của rmiregistry đưa vào trong tham số

4.4. Bài tập áp dụng

- **Mục đích**:

  Xây dựng ứng dụng phân tán theo cơ chế RMI.

- **Yêu cầu**

Sinh viên thực hiện các bài tập sau:

Bài 1 : Xây dựng một ứng dụng phục vụ việc bán vé máy bay cho các đại lý phân tán ở các tỉnh thành khác nhau. Ứng dụng này có các lớp sau:

- Lớp chuyến bay: Đại diện cho một chuyến bay

- Có các thuộc tính: Số hiệu chuyến bay, Ngày giờ bay, Nơi đi, Nơi đến, Thời gian bay, Tổng số ghế, Số lượng ghế đã bán, Số lượng ghế còn trống.
- Các phương thức trên một chuyến bay: phương thức xem thông tin về chuyến bay, phương thức mua vé, phương thức trả vé. Để phục vụ cho nhiều đại lý các phương thức trên thuộc loại được gọi từ xa.

- Lớp Server, tạo ra nhiều chuyến bay và duy trì nó để cho phép các đại lý thực hiện các giao dịch trên chuyến bay cụ thể.
- Client là chương trình cho phép mỗi đại lý được quyền xem thông tin về chuyến bay, mua vé, trả vé theo yêu cầu.

BÀI 5: LÀM VIỆC VỚI SMTP VÀ POP3 - KẾT NỐI VỚI EMAIL SERVERS

Mã bài : MĐ 38.5

Mục tiêu:

- Trình bày được cách sử dụng các lớp trong lập trình với Mail Server
- Xây dựng ứng dụng Mail.
- Thực hiện các thao tác an toàn với máy tính.

5.1 Phương thức gởi và nhận Email

Email là một trong những ứng dụng mạng lâu đời nhất nhưng lại phổ dụng nhất. Thử nghĩ khi bạn muốn gởi thông điệp đến một người bạn ở đầu kia của thế giới, bạn muốn mang thư chạy bộ qua đó hay chỉ đơn giản lên máy tính gõ ít hàng và nhấn nút **Send**? Thật ra, những bậc tiền bối của mạng ARPANET đã không tiên đoán được email sẽ là ứng dụng then chốt chạy trên mạng này, mục tiêu chính của họ là thiết kế hệ thống cho phép truy cập tài nguyên từ xa. Hệ thống email ra đời không mấy nổi bật, để bây giờ lại được sử dụng hằng ngày bởi hàng triệu người trên thế giới.

Mục tiêu của phần này là chỉ ra những nhân vật hoạt động trong hệ thống email, vai trò của họ, giao thức mà họ sử dụng và khuôn dạng thông điệp mà họ trao đổi với nhau.

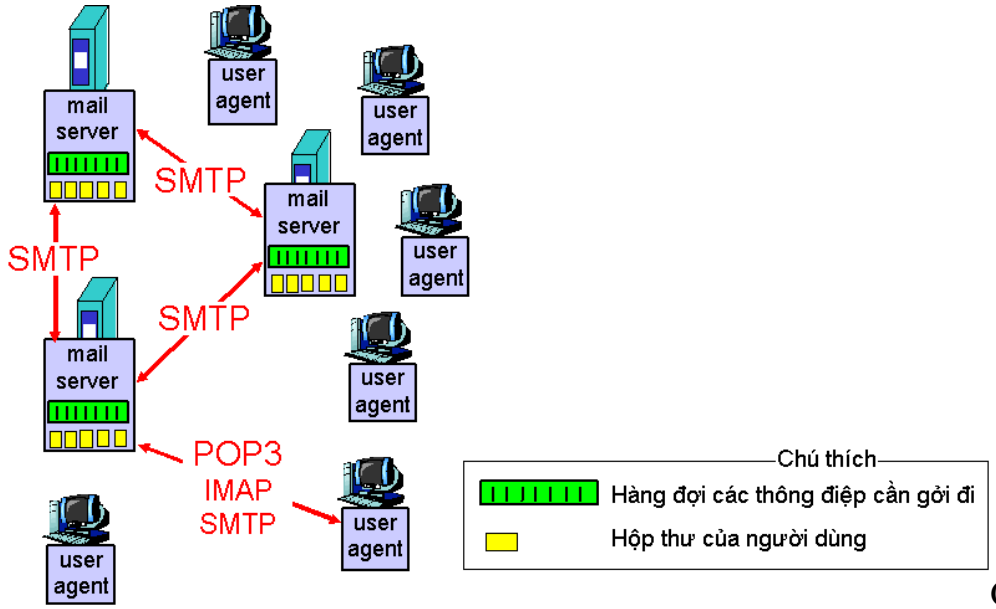
5.1.1 Các thành phần của hệ thống email

Một hệ thống email thường có 3 thành phần chính: Bộ phận trợ giúp người dùng (User Agent), Mail Server và các giao thức mà các thành phần này dùng để giao tiếp với nhau.

Người ta phân loại các giao thức như sau:

- Giao thức giữa các mail servers bao gồm:
  - SMTP (Simple Mail Transfer Protocol): được các server dùng để chuyển thư qua lại với nhau. Ví dụ nôm na, nó giống như cách thức mà các trạm bưu điện dùng để chuyển các thùng thư của khách hàng cho nhau. Thông tin chi tiết về giao thức này được mô tả trong tài liệu RFC 822.
- Giao thức giữa mail server và user agent bao gồm:

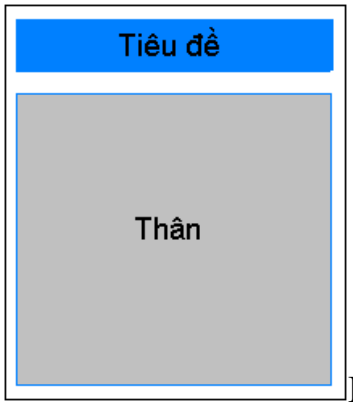
- POP3 (Post Offic Protocol version 3 [RFC 1939]): được user agent sử dụng để lấy thư về từ hộp thư của nó trên server.
- SMTP: được user agent sử dụng để gởi thư ra server.
- IMAP: (Internet Mail Access Protocol [RFC 1730]): Có nhiều tính năng vượt trội hơn POP3. Ngoài ra IMAP còn cho phép gởi mail.



Các thành phần của hệ thống email (H8.6)

5.1.2 Khuôn dạng của một email

RFC 822 định nghĩa một email gồm có hai phần: phần tiêu đề (header) và phần thân (body).



Khuôn dạng của email (H8.7)

Cả hai phần đều được thể hiện dưới dạng ký tự ASCII. Lúc đầu, phần thân được qui định có khuôn dạng văn bản đơn giản. Sau này người ta đề nghị một chuẩn mới gọi là MIME, có thể cho phép phần thân của email chứa bất kỳ loại dữ liệu nào.

Phần tiêu đề bao gồm nhiều dòng thông tin, mỗi dòng kết thúc bằng hai ký tự <CRLF>. Phần tiêu đề được chia khỏi phần thân bởi một hàng rỗng. Mỗi một hàng tiêu đề chứa một cặp "tên" và "giá trị", cách nhau bởi dấu hai chấm (:). Người

dùng có thể rất quen với nhiều hàng tiêu đề vì họ thường phải điền thông tin vào đấy. Ví dụ

Tên	Giá trị
From:	Địa chỉ người gởi
To:	Địa chỉ của người nhận
Subject:	Chủ đề thư
Date:	Ngày gởi

RFC 822 được mở rộng năm 1993 (và được cập nhật lại năm 1996) để cho phép email mang được nhiều loại dữ liệu: audio, video, hình ảnh, tài liệu Word, … MIME (Multipurpose Internet Mail Extensions) về cơ bản có ba phần. Phần đầu tiên là tập các dòng header dùng để bổ túc cho phần header cũ của RFC 822. Theo nhiều cách, những dòng header này mô tả dữ liệu chứa trong phần thân. Cụ thể như sau:

Tên	Giá trị
MIME-Version:	Phiên bản MIME đang sử dụng
Content-Description:	Mô tả trong thư đang có dữ liệu gì
Content-Type:	Mô tả kiểu dữ liệu đang nằm trong thư
Content-Transfer-Encoding:	Mô tả cách thức mã hóa dữ liệu trong thư

Phần thứ hai là các định nghĩa cho một tập các kiểu nội dung (và kiểu con nếu có). Ví dụ một số kiểu mà MIME định nghĩa:

Kiểu	Ý nghĩa
image/gif	Ảnh dạng gif

image/jpeg	Ảnh dạng jpeg
text/plain	Văn bản đơn giản
text/richtext	Văn bản mở rộng (có đặt font chữ, được định dạng đậm, nghiêng hoặc gạch dưới …)
application	Dữ liệu trong thư được xuất ra từ một ứng dụng nào đó. Chẳng hạn:*application/postscript*: tài liệu Postscript (.ps)*application/msword*: tài liệu Microsoft Word (.doc)

MIME cũng định nghĩa kiểu **multipart** để chỉ ra cách mà phần thân của thư mang nhiều loại dữ liệu khác nhau như thế nào. Chỉ có một kiểu con của **multipart** là **mixed** với ý nói rằng trong phần thân của thư có nhiều mảnh dữ liệu khác nhau, độc lập với nhau và được sắp xếp theo một trình tự cụ thể. Mỗi mảnh dữ liệu sẽ có phần tiêu đề riêng để mô tả kiểu dữ liệu của mảnh đó.

Phần thứ ba mô tả cách thức mã hóa các kiểu dữ liệu nói trên để có thể truyền chúng dưới dạng ASCII. Lý do để mọi bức thư phải chứa các ký tự ASCII là vì để đi được đến đích, bức thư đó có thể phải trung chuyển qua nhiều gateway, mà các gateway này đều coi mọi bức thư dưới dạng ASCII. Nếu trong thư chứa bất kỳ ký tự nào khác ASCII thì thư sẽ bị đứt gãy nội dung. MIME sử dụng phương pháp mã hóa trực tiếp dữ liệu nhị phân thành các ký tự nhị phân, gọi là **base64**. Ý tưởng của **base64** là ánh xạ 3 bytes dữ liệu nhị phân nguyên thủy thành 4 ký tự ASCII. Giải thuật đơn giản như sau: tập hợp 3 bytes dữ liệu nhị phân lại thành 24 bits, sau đó chia 24 bits này thành 4 cụm, một cụm 6 bits. Một cụm 6 bits được ánh xạ vào một trong 64 ký tự ASCII hợp lệ; ví dụ 0 ánh xạ thành A, 1 ánh xạ thành B… Nếu nhìn vào bức thư đã được mã hóa dạng **base64**, người dùng sẽ thấy chỉ có 52 chữ cái cả hoa lẫn thường, 10 chữ số từ 0 đến 9 và các ký tự đặc biệt + và /.

Đối với những người dùng chỉ sử dụng trình đọc thư hỗ trợ duy nhất kiểu ký tự thì việc đọc những bức thư có kiểu **base64** sẽ rất là đau khổ. Vì lý do nhân đạo, MIME còn hỗ trợ kiểu mã hóa ký tự thường được gọi là **7-bit**. **7-bit** sẽ giữ nguyên dạng ký tự mà người ta nhập vào.

Tổng hợp lại, ví dụ một bức thư có 2 loại dữ liệu: văn bản thường, một ảnh JPEG, sẽ có hình dáng như sau:

```
From: ptphi@cit.ctu.edu.vn
To: TH27@cit.ctu.edu.vn
Subject: Picture of students.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="-98766789"

--98766789
Content-Transfer-Encoding: 7bit
Content-Type: text/plain

Hi,
Please find a picture of you.
--98766789
Content-Transfer-Encoding: base64
```

5.1.3 Chuyển thư

Kế đến, chúng ta sẽ xem xét giao thức SMTP – giao thức được dùng để chuyển thư từ máy này đến máy kia. Để đặt SMTP vào đúng ngữ cảnh, chúng ta nên nhắc lại các nhân vật then chốt trong hệ thống email. Đầu tiên, người dùng tương tác với trình đọc thư (hay còn gọi là user agent) để soạn, lưu, tìm kiếm và đọc thư của họ. Hiện trên thị trường có nhiều phần mềm đọc thư, cũng giống như hiện cũng đang có nhiều loại trình duyệt Web vậy. Thứ hai, có trình xử lý thư (hay còn gọi là mail server) chạy trên một máy nào đó trong mạng nội bộ của người dùng. Có thể xem mail server như một bưu điện: Người dùng trao cho mail server các bức thư mà họ muốn gởi cho người dùng khác, mail server sử dụng giao thức SMTP trên TCP để chuyển bức các thư này đến mail server bên đích. Mail server bên đích nhận các thư đến và đặt chúng vào hộp thư của người dùng bên đích. Do SMTP là giao thức mà rất nhiều người có thể tự cài đặt, vì thế sẽ có rất nhiều sản phẩm mail server hiện có trên thị trường. Sản phẩm mail server thường được sử dụng nhất là **sendmail**, ban đầu được cài đặt trong hệ điều hành Berkeley Unix.

Tất nhiên mail server bên máy gởi có thể kết nối SMTP/TCP trực tiếp tới mail server bên máy nhận, nhưng trong thực tế, một bức thư có thể đi ngang qua vài mail gateways trước khi đến đích. Cũng giống như máy đích, mỗi mail gateway cũng chạy một mail server. Không phải ngẫu nhiên mà các nút chuyển thư trung gian được gọi là mail gateway. Công việc của chúng cũng giống như các IP gateway là lưu tạm và chuyển phát tiếp các bức thư của người dùng. Điểm khác

nhau duy nhất giữa chúng là, mail gateway trữ tạm các bức thư trong đĩa, trong khi các IP gateway trữ tạm các gói tin IP trong bộ nhớ.

Bạn có thể đặt câu hỏi: tại sao lại cần đến các mail gateways? Tại sao không dùng phương pháp nối kết SMTP/TCP trực tiếp từ bên gởi sang bên nhận? Lý do thứ nhất, người gởi không muốn kèm trong thư địa chỉ của máy đích. Ví dụ, riêng việc nhập vào trong thư địa chỉ đích **ptphi@cit.ctu.edu.vn** đã mất công rồi, không ai thấy thoải mái khi phải nhập thêm địa chỉ máy đích là **machine-of-phi.cit.ctu.edu.vn**. Thứ hai, không chắc lúc bên gởi thiết lập nối kết đến bên nhận, người dùng bên nhận đã bật sẵn máy! Thành thử chỉ cần địa chỉ thư bên nhận là đủ. Khi bức thư đến được mail gateway của Khoa Công Nghệ Thông Tin – Đại học Cần Thơ, nếu người dùng **ptphi** đang mở máy, mail gateway sẽ chuyển thư cho anh ta ngay, nếu không mail gateway sẽ trữ tạm thư trên đĩa của nó đến khi **ptphi** bật máy lên và kiểm tra thư.

Dù có bao nhiêu mail gateways trung gian trên đường đến đích vẫn không đáng lo lắng, bởi vì mỗi mail gateway trung gian sẽ nỗ lực sử dụng một kết nối SMTP độc lập đến gateway kế tiếp trên đường đi nhằm chuyển thư càng ngày càng đến gần người nhận.

5.2. SMTP

SMTP là một giao thức đơn giản dùng các ký tự ASCII. Sau khi thiết lập nối kết TCP đến cổng 25 của máy đích (được coi là server), máy nguồn (được coi là client) chờ nhận kết quả trả về từ server. Server khởi đầu cuộc đối thoại bằng cách gởi một dòng văn bản đến client thông báo danh tính của nó và khả năng tiếp nhận thư. Nếu server không có khả năng nhận thư tại thời điểm hiện tại, client sẽ hủy bỏ nối kết và thử thiết lập lại nối kết sau.

Nếu server sẵn sàng nhận thư, client sẽ thông báo lá thư đó từ đâu đến và ai sẽ là người nhận. Nếu người nhận đó tồn tại, server sẽ thông báo cho client tiếp tục gởi thư. Sau đó client gởi thư và server báo nhận cho thư đó. Sau khi cả hai bên hoàn tất phiên truyền nhận, kết nối sẽ được đóng lại.

Ví dụ một phiên truyền nhận được cho ngay dưới đây. Những dòng bắt đầu bằng **C:** là của phía client gởi đi; bằng **S:** là các câu trả lời của server.

```
S: 220 ctu.edu.vn
C: HELO cit.ctu.edu.vn
S: 250  ctu.edu.vn says hello to cit.ctu.edu.vn
C: MAIL FROM: <ptphi@cit.ctu.edu.vn>
S: 250 Sender ok
C: RCPT TO: <lhly@yale.edu>
S: 250 Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Subject: It's Xmast!
C: So I hope you a merry Xmas and a happy new year!
C: .
```

Như đã thấy trong ví dụ, client gởi đi một lệnh (**HELO, MAIL FROM, RCPT TO, DATA, QUIT**) và server trả lời bằng một mã số (250, 354, 221) có kèm theo lời chú thích có thể đọc được. Client kết thúc thư bằng **<CRLF>.<CRLF>**. Sau đây là bảng giải thích một số lệnh của client và mã số trả lời của server.

LỆNH CỦA CLIENT	
Lệnh	Ý nghĩa
HELO	Câu chào và xưng danh của client
MAIL FROM	Địa chỉ email của người gởi
RCPT TO	Địa chỉ email của người nhận
DATA	Bắt đầu truyền nội dung của thư
QUIT	Hủy nối kết
TRẢ LỜI CỦA SERVER	
Trả lời	Ý nghĩa

250	Yêu cầu hợp lệ
550	Yêu cầu không hợp lệ, không tồn tại hộp thư như client đã chỉ ra.
354	Cho phép bắt đầu nhập thư vào. Kết thúc thư bằng <CRLF>.<CRLF>
221	Server đang đóng kết nối TCP

Vẫn còn nhiều lệnh và mã trả lời chưa được trình bày, xin tham khảo tài liệu RFC 822 để có được đầy đủ thông tin.

Phân phát thư

Như đã trình bày, khi đứng về góc độ người dùng thư, họ sẽ dùng user agent để gởi và nhận thư cho họ. User agent dùng giao thức SMTP để gởi thư đi, dùng giao thức POP3 hoặc IMAP để nhận thư về.

5.3 POP3

Một phiên làm việc theo giao thức POP3 bắt đầu tại user agent. User agent khởi động một nối kết TCP đến cổng 110 của mail server. Khi kết nối thực hiện xong, phiên làm việc POP3 sẽ trải qua theo thứ tự ba kỳ:

1. Chứng thực.
2. Giao dịch dữ liệu.
3. Cập nhật.

Kỳ chứng thực buộc người dùng thực hiện thủ tục đăng nhập bằng cách nhập vào hai lệnh sau:

Lệnh	Ý nghĩa
USER <*tên người dùng*>	Khai báo tên người dùng.
PASS <*mật khẩu*>	Khai báo mật khẩu.

Báo trả của mail server sẽ là một trong hai câu sau:

Trả lời	Ý nghĩa
+OK <*chú thích*>	Khai báo của người dùng là đúng.
+ERR <*chú thích*>	Khai báo của người dùng là sai và lời giải thích.

Trong kỳ giao dịch, người dùng có thể xem danh sách thư chưa nhận về, nhận thư về và xóa thư trong hộp thư của mình khi cần thiết. Các lệnh mà người dùng thường sử dụng để giao dịch với server là:

Lệnh	Ý nghĩa
LIST[<*số thứ tự thư*>]	Nếu dùng LIST không tham số, server sẽ trả về toàn bộ danh sách các thư chưa nhận. Nếu có tham số là số thứ tự thư cụ thể, server sẽ trả về thông tin của chỉ bức thư đó thôi.
RETR <*số thứ tự thư*>	Tải lá thư có số thứ tự <*số thứ tự thư*> về.
DELE<*số thứ tự thư*>	Xóa lá thứ số <*số thứ tự thư*> khỏi hộp thư.
QUIT	Hoàn tất giai đoạn giao dịch và hủy nối kết TCP

Các trả lời của server có thể là các số liệu mà client yêu cầu hoặc các thông báo +OK, -ERR như trong phần đăng nhập.

Sau đây là dàn cảnh một phiên làm việc ví dụ giữa người dùng **ptphi** khi anh ta đăng nhập và làm việc trên hộp thư của mình tại server có địa chỉ **mail.cit.ctu.edu.vn**.

Client	Server	Giải thích
	+OK POP3 server ready	Server sẵn sàng phục vụ client
USER ptphi		

	+OK	Server xác nhận người dùng hợp lệ
PASS godblessus		
	+OK login successfully	Chứng thực thành công
LIST		ptphi kiểm tra hộp thư
	+OK1 10242 2550	Hộp thư của ptphi còn hai thư chưa nhận về, thư thứ nhất có kích thước 1024 bytes, thư thứ hai có kích thước 2550 bytes
RETR 1		ptphi tải thư thứ nhất về
	+OK	server gởi thư thứ 1 cho ptphi
DELE 1		ptphi xóa thư thứ nhất trong hộp thư
	+OK	server xoá thư thứ 1 thành công
QUIT		ptphi hủy nối kết
	+OK Bye-Bye	server hủy nối kết

5.4 IMAP

Với những người dùng có một tài khoản email trên một ISP và người dùng này thường truy cập email trên một PC thì giao thức POP3 hoạt động tốt. Tuy nhiên, một sự thật trong ngành công nghệ máy tính, khi một thứ gì đó đã hoạt động tốt, người ta lập tức đòi hỏi thêm nhiều tính năng mới (và tự chuốc lấy nhiều phiền nhiễu). Điều đó cũng xảy ra đối với hệ thống email. Ví dụ, người ta chỉ có một tài khoản email, nhưng họ lại muốn ngồi đâu cũng truy cập được nó. POP3 cũng làm được chuyện này bằng cách đơn giản tải hết các email xuống máy PC mà người dùng này đang ngồi làm việc. Và dĩ nhiên là thư từ của người dùng này nằm rải rác khắp nơi.

Sự bất tiện này khơi mào cho sự ra đời của giao thức phân phối thư mới, IMAP (Internet Message Access Protocol), được định nghĩa trong RFC 2060. Không giống như POP2, IMAP coi các thông điệp mặc nhiên nằm trên server vô hạn và trên nhiều hộp thư. IMAP còn đưa ra cơ chế cho phép đọc các thông điệp hoặc một phần của thông điệp, một tính năng hữu ích khi người dùng kết nối đến server bằng đường truyền tốc độ chậm như điện thoại nhưng lại đọc các email có âm thanh, hình ảnh… Với quan niệm cho rằng người dùng không cần tải thư về lưu trên PC, IMAP cung cấp các cơ chế cho phép tạo, xóa và sửa đổi nhiều hộp thư trên server.

Cung cách làm việc của IMAP cũng giống như POP3, ngoài trừ trong IMAP có rất nhiều lệnh. IMAP server sẽ lắng nghe trên cổng 143. Cũng nên chú ý rằng, không phải mọi ISP đều hỗ trợ cả hai giao thức POP3 và IMAP.

Bảng sau so sánh các tính năng của POP3 và IMAP

Tính năng	POP3	IMAP
Giao thức được định nghĩa ở đâu?	RFC 1939	RFC 2060
Cổng TCP được dùng	110	143
Email được lưu ở đâu	PC của người dùng	Server
Email được đọc ở đâu	Off-line	On-line
Thời gian nối kết	Ít	Nhiều
Sử dụng tài nguyên của server	Tối thiểu	Nhiều hơn
Nhiều hộp thư	Không	Đúng
Ai lưu phòng hờ các hộp thư	Người dùng	ISP
Tốt cho người dùng di động	Không	Có
Kiểm soát của người dùng đối với việc tải thư về	Ít	Tốt

Tải một phần thư	Không	Có
Quota đĩa có là vấn đề không?	Không	Thỉnh thoảng
Dễ cài đặt	Có	Không
Được hỗ trợ rộng rãi	Có	Đang phát triển

BÀI 6: TRUYỀN FILE

Mã bài : MĐ 38.6

Mục tiêu:

- Trình bày được cách sử dụng các lớp khi sử dụng với FTP;
- Xây dựng ứng dụng trong việc truyền và nhận File.
- Thực hiện các thao tác an toàn với máy tính.

6.1 Khái niệm giao thức FTP

FTP (viết tắt của File Transfer Protocol dịch ra là "Giao thức truyền tập tin") là giao thức truyền file giữa các máy tính. Giao thức này xuất hiện từ những năm 1971 (khi Internet vẫn chỉ là một dự án thử nghiệm) nhưng vẫn còn được sử dụng rộng rãi cho đến tận ngày nay. FTP được đặc tả trong RFC 959. FTP cho phép trao đổi file giữa hai máy tính.

Thông qua giao thức FTP, người dùng có thể tải dữ liệu như hình ảnh, văn bản, các tập tin media (nhạc, video)… từ máy tính của mình lên máy chủ đang đặt ở một nơi khác hoặc tải các tập tin có trên máy chủ về máy tính cá nhân. Giao thức FTP được sử dụng nhiều nhất vào mục đích…

6.2 Hoạt động của FTP

Trong phiên làm việc của FTP, người dùng làm việc trên máy tính của mình và trao đổi file với một máy tính ở chỗ khác. Để truy cập tới máy tính khác, người dùng phải đăng nhập thông qua việc cung cấp định danh người dùng và mật khẩu. Sau khi những thông tin này được kiểm chứng thì công việc truyền file từ hệ thống file trên máy tính của mình đến hệ thống file ở đầu kia mới có thể được thực hiện.

FTP sử dụng hai kết nối TCP song song, một đường truyền thông tin điều khiển (control connection) và một đường truyền dữ liệu (data connection). Các thông tin điều khiển như thông tin định danh người dùng, mật khẩu truy nhập, lệnh thay đổi thư mục, lệnh "PUT" hoặc "GET" file giữa hai máy tính được trao đổi qua đường truyền thông tin điều khiển. Đường truyền dữ liệu để truyền file dữ liệu thực sự. Vì FTP phân biệt luồng thông tin điều khiển với luồng dữ liệu nên nó dược gọi là gửi thông tin điều khiển out-of-band.
FTP gồm 2 kết nối điều khiển và dữ liệu

Khi người dùng bắt đầu một phiên làm việc FTP, đầu tiên FTP sẽ thiết lập một đường kết nối thông tin điều khiển TCP qua cổng 21 . Phía client của giao

thức FTP truyền thông tin về định danh người dùng và mật khẩu cũng như lệnh thay đổi thư mục qua kết nối này. Khi người dùng có một yêu cầu trao đổi file (truyền từ/đến máy người dùng), FTP mở một kết nối TCP để truyền dữ liệu qua cổng 20. FTP truyền đúng một file qua kết nối này và ngay sau khi truyền xong thì đóng kết nối lại. Nếu trong cùng phiên làm việc người dùng có yêu cầu truyền file thì FTP sẽ mở một kết nối khác. Như vậy với FTP, luồng thông tin điều khiển được mở và tồn tại trong suốt phiên làm việc của người dùng, nhưng mỗi kết nối dữ liệu được tạo ra cho mỗi một yêu cầu truyền file (kết nối dữ liệu là không liên tục).

Trong suốt phiên làm việc, FTP server phải giữ lại các thông tin về trạng thái của người dùng, đặc biệt nó phải kết hợp các thông tin điều khiển với tài khoản của người dùng. Server cũng lưu giữ thư mục hiện thời mà người dùng truy cập cũng như cây thư mục của người dùng. Ghi lại các thông tin trạng thái của mỗi phiên làm việc hạn chế đáng kể tổng số phiên làm việc đồng thời.

6.3 Các lệnh FTP (FTP Commands)

Lệnh (yêu cầu) từ client đến server và kết quả (trả lời) từ server tới client được gửi thông qua kết nối điều khiển và được mã hoá bằng bảng mã ASCII 7 bit. Do vậy giống như lệnh HTTP, người ta có thể đọc được lệnh FTP. Trường hợp các lệnh viết liên tục thì cặp ký tự CR (carriage return) và LF (line feed) được sử dụng để phân biệt các lệnh (và trả lời). Mỗi câu lệnh chứa 4 kí tự ASCII in hoa, một số lệnh có tham số. Sau đây là một số câu lệnh hay gặp:

- USER username: sử dụng để gửi thông tin định danh người dùng cho server
- PASS password: dùng để gửi password cho server
- LIST: dùng để yêu cầu server gửi một danh sách các file trong thư mục hiện thời. Danh sách này được gửi thông qua một kết nối dữ liệu TCP
- RETR filename: dùng để lấy một file từ thư mục hiện thời (trên máy ở xa)
- STOR filename: dùng để tải một file vào thư mục hiện thời (trên máy ở xa)

Thông thường có quan hệ 1-1 giữa lệnh của người dùng và lệnh của FTP . Ứng với mỗi lệnh từ client là một trả lời của server. Câu trả lời là một mã 3 chữ số và có thể có một thông báo kèm theo.

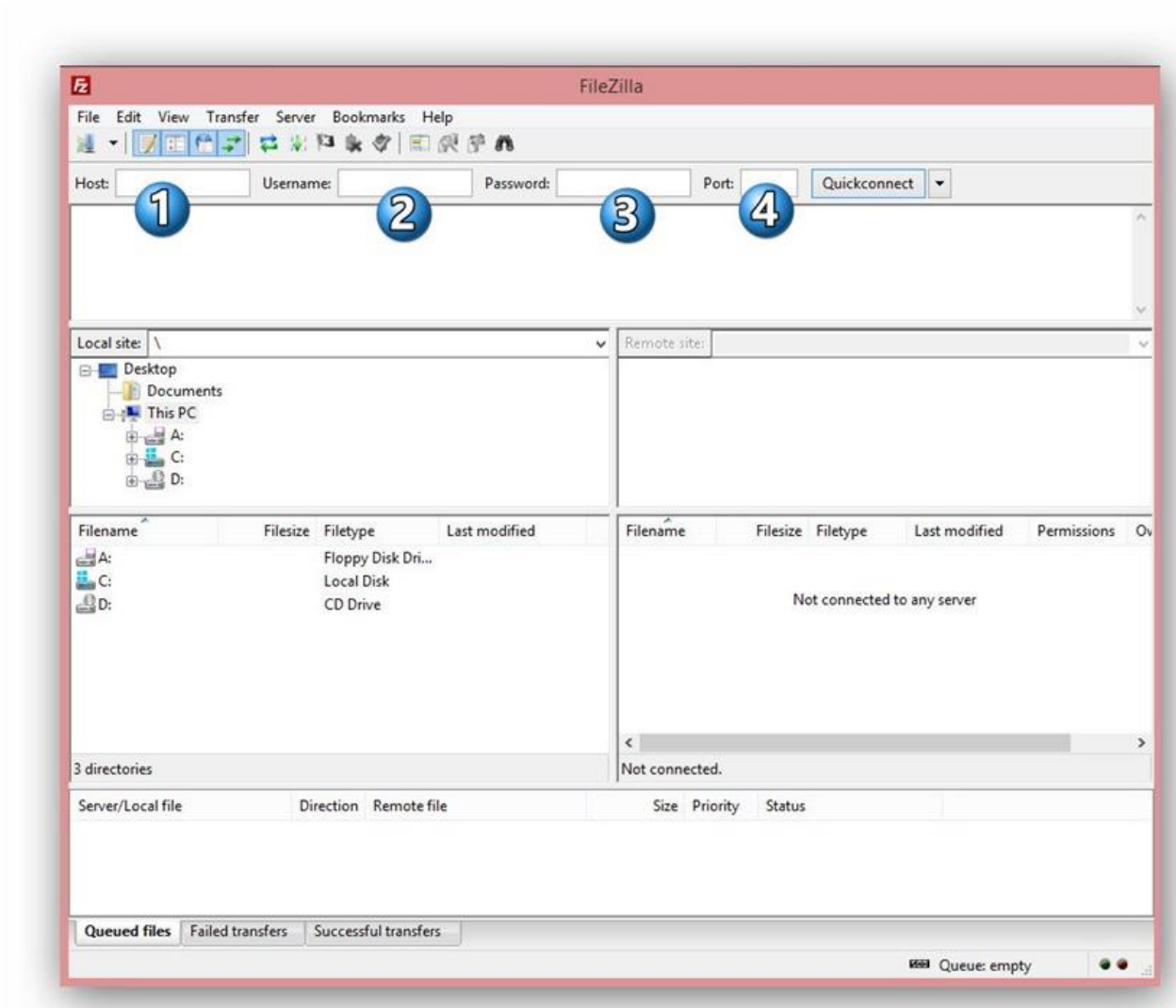
6.4 Cách kết nối vào FTP

Trước khi kết nối vào FTP trên host, mình cần các bạn đã chắc chắn trỏ tên miền về host mặc dù bước này không cần thiết vì bạn có thể kết nối vào cái hostname

của host hoặc dịa chỉ IP của host nhưng ở đây mình sẽ chỉ hướng dẫn kết nối vào FTP qua tên miền trên host để tránh dài dòng.

Để kết nối vào FTP trên host bạn cần phải sử dụng một ứng dụng chuyên làm việc này, nó được gọi là FTP Client. Hiện nay, bạn có thể sử dụng phần mềm **FileZilla** vì đây là FTP Client miễn phí tốt nhất hiện tại, hỗ trợ hầu hết mọi hệ điều hành hiện nay.

Sau khi cài đặt phần mềm FileZilla vào máy, bạn khởi động nó lên sẽ có giao diện như sau:

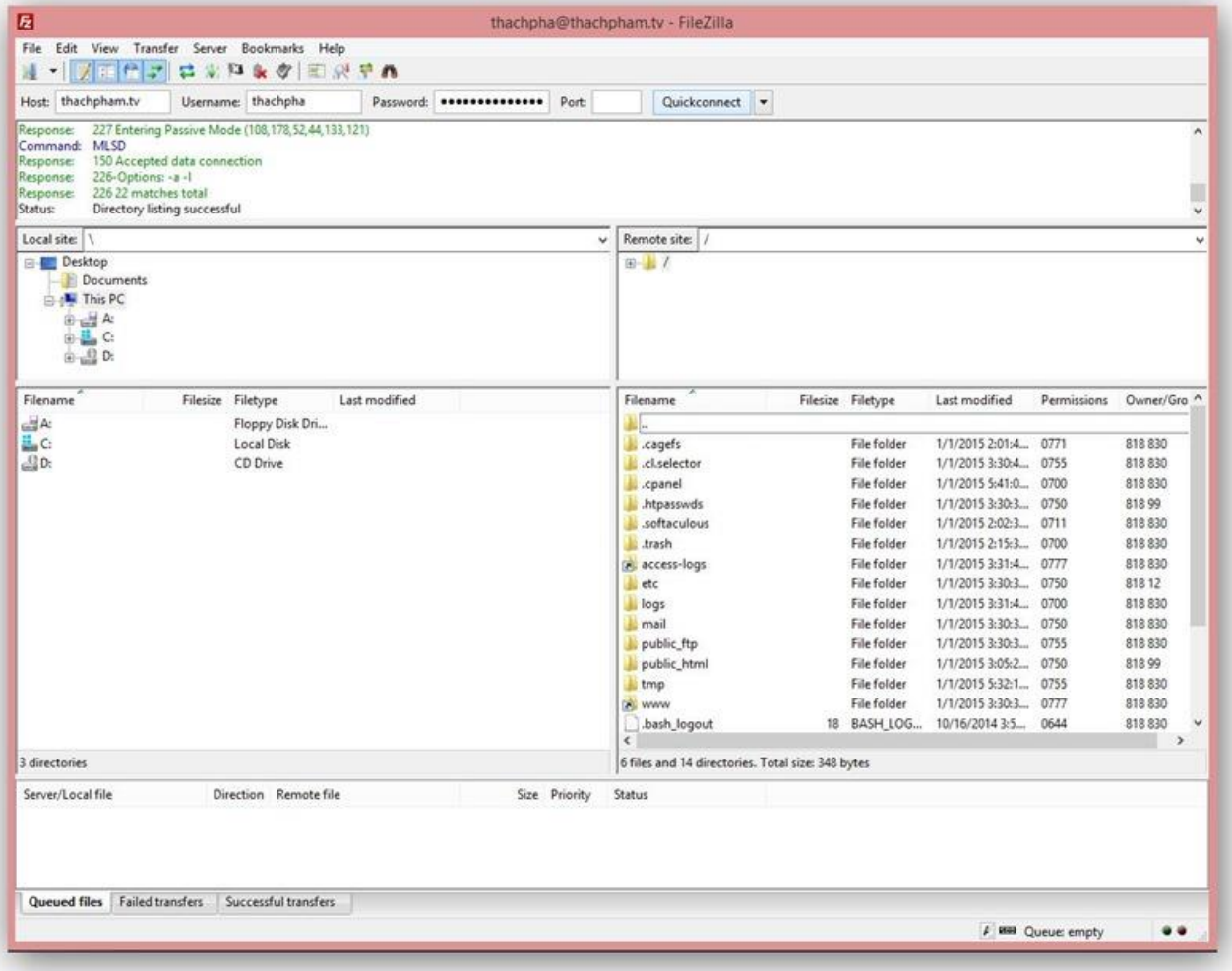


Giải thích:

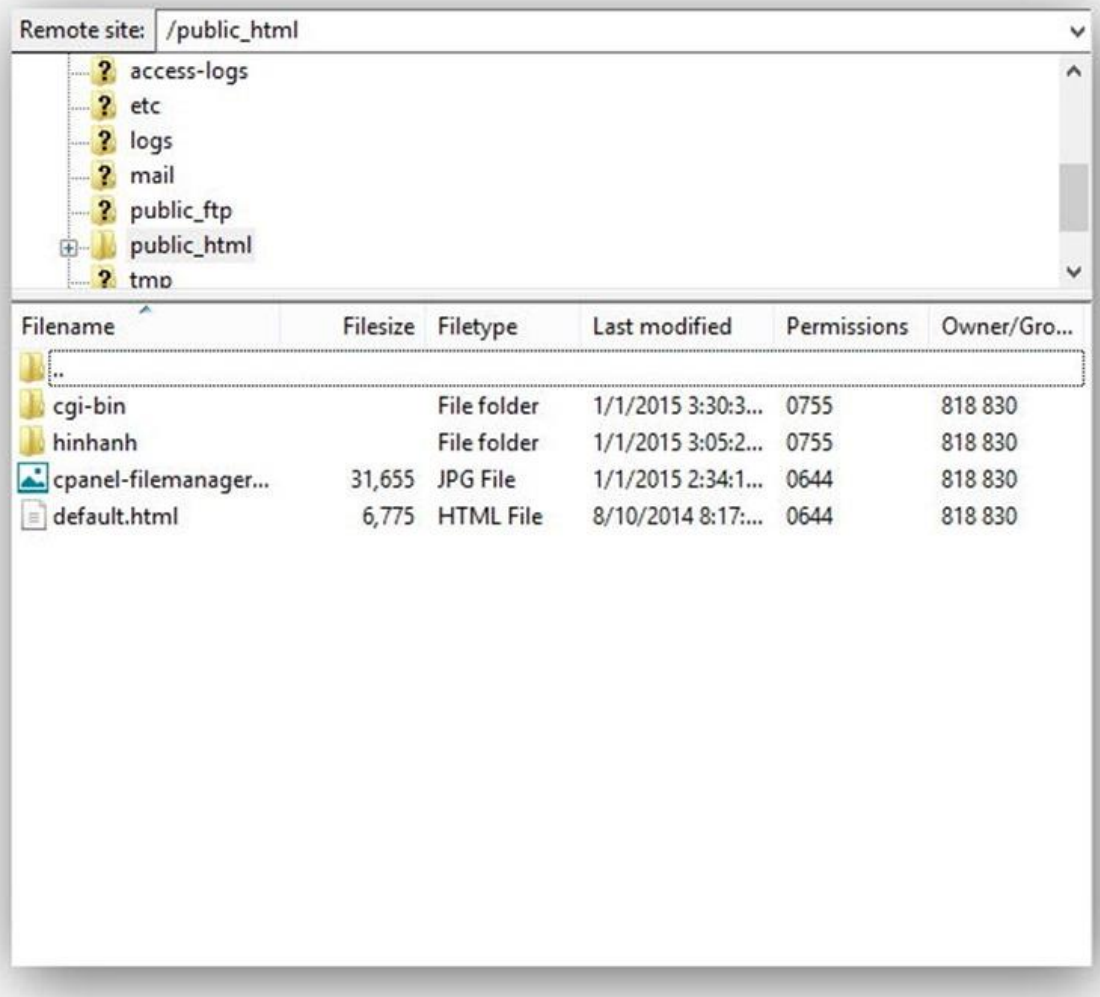
- 1 – Địa chỉ host hoặc IP của host cần truy cập, hãy điền tên miền website của bạn vào.
- 2 – Tên username truy cập vào host
- 3 – Mật khẩu truy cập vào host

- 4 – Cổng kết nối, cổng của giao thức FTP là 21, bạn không cần điền cũng được vì mặc định nó đã là 21

Điền xong các thông tin đó, các bạn ấn vào nút **Quickconnect** để bắt đầu kết nối. Kết nối thành công nó sẽ hiển thị ra như thế này.

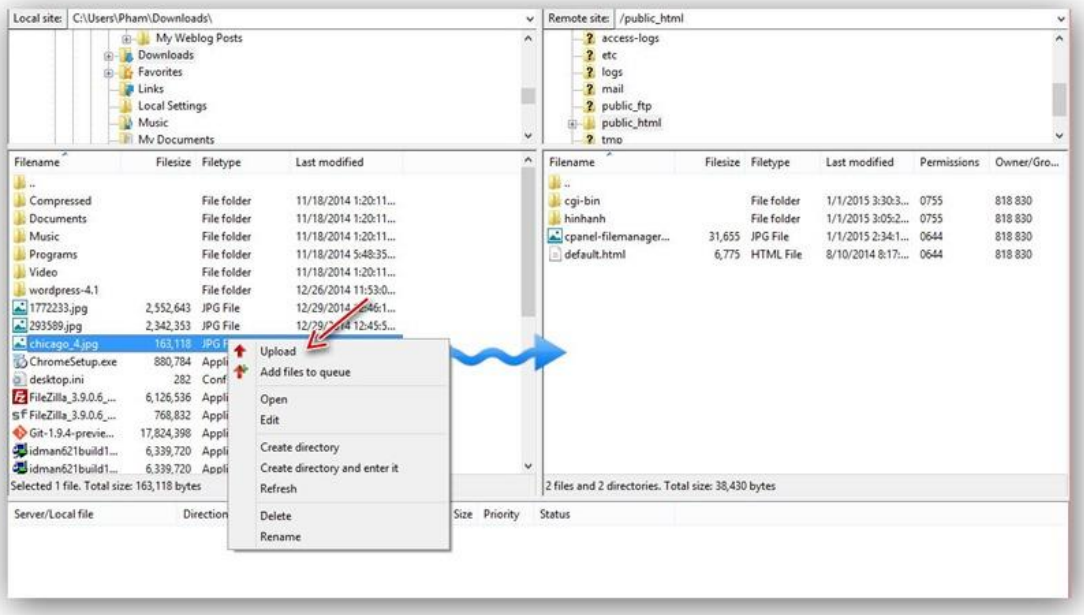


Trong đó, bên tay trái là các dữ liệu trên máy tính và bên tay phải là các dữ liệu trên host. Bạn thấy thư mục public_html chứ, đó là thư mục gốc trên host của bạn đó, nhấp vào đi rồi bạn sẽ thấy các tập tin và thư mục trên host của mình hiện ra.

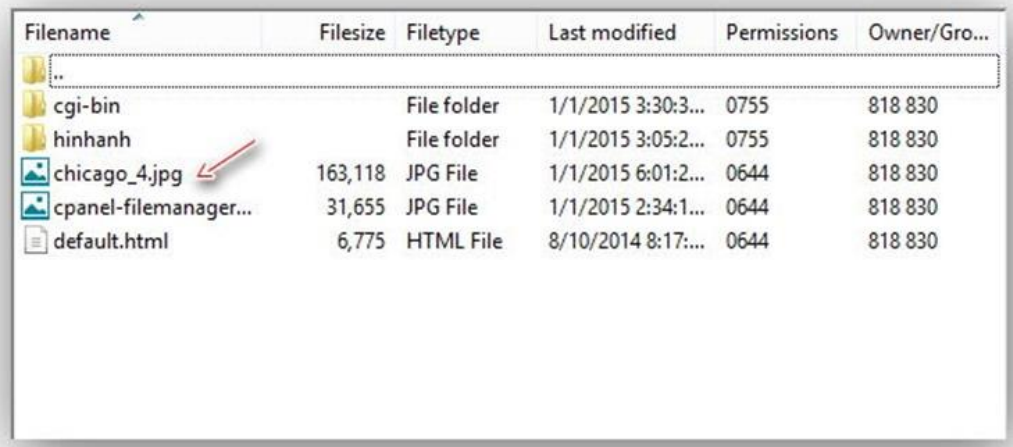


Đối với các tập tin, bạn có thể ấn chuột phải và chọn Edit để sửa nội dung, ngoại trừ hình ảnh, chắc chắn rồi.

Nếu bạn muốn upload cái gì đó lên host qua FTP, bạn cứ truy cập vào thư mục cần upload vào trên FTP và di chuyển thư mục ở máy tính đến vị trí của tập tin cần upload, ấn chuột phải và chọn Upload.



Chờ đợi đến khi nào trên host xuất hiện cái tập tin bạn vừa upload thì thôi.



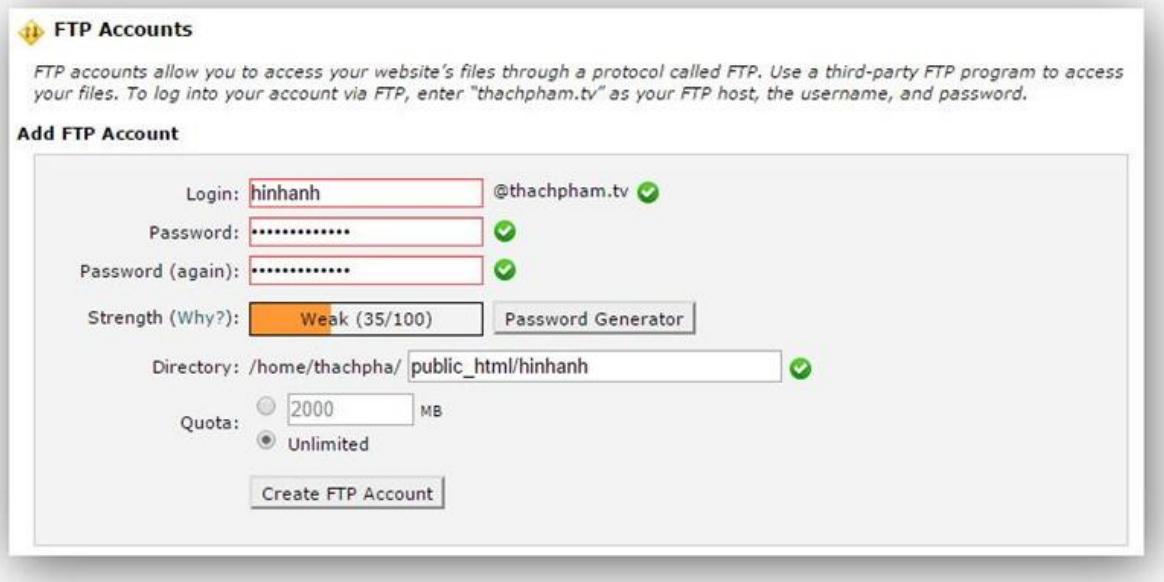
Tương tự, bạn có thể ấn chuột phải vào file nào đó trên host và chọn Download để tải về máy tính.

Tạo tài khoản FTP riêng

FTP có một cái hay là bạn có thể tạo ra nhiều tài khoản FTP riêng và có thể chỉ định tài khoản đó chỉ có thể quản lý một thư mục nào đó trên host, rất có ích nếu bạn muốn cho người khác có quyền upload dữ liệu lên host của bạn mà không sợ người ta đụng chạm tới các dữ liệu khác.

Để làm được việc này, bạn đăng nhập vào control panel của host và tìm đến **FTP Accounts**.

Và mình thiết lập username và mật khẩu FTP cần tạo ra, đồng thời bạn có thể khai báo thư mục được áp dụng cho tài khoản FTP này và dung lượng tối đa tài khoản FTP này được phép sử dụng.



Lưu ý là với các tài khoản FTP được chính bạn tạo ra, username của nó sẽ có dạng là username@domain-của-bạn.com nên khi đăng nhập, bạn phải nhập username là như thế. Như ví dụ với ảnh trên, mình tạo ra một tài khoản FTP với username

là hinhanh@thachpham.tv và nó được quyền sử dụng thư mục /public_html/hinhanh/. Ấn nút Create FTP Account để kết thúc.

Bây giờ bạn có thể đăng nhập vào FTP bằng FileZilla, phần host bạn vẫn để là domain-của-bạn.com, chỉ khác username mà thôi.

BÀI 7: AN NINH MẠNG (FIREWALLS, PROXY SERVERS, AND ROUTERS)

Mục tiêu:

- Mô tả cách xây dựng mạng lưới bảo vệ mạng;
- Trình bày được các cạm bẩy mạng;
- Thực hiện được xây dựng ứng dụng đơn giản bảo đảm an toàn hệ thống mạng trong doanh nghiệp.
- Thực hiện các thao tác an toàn với máy tính.

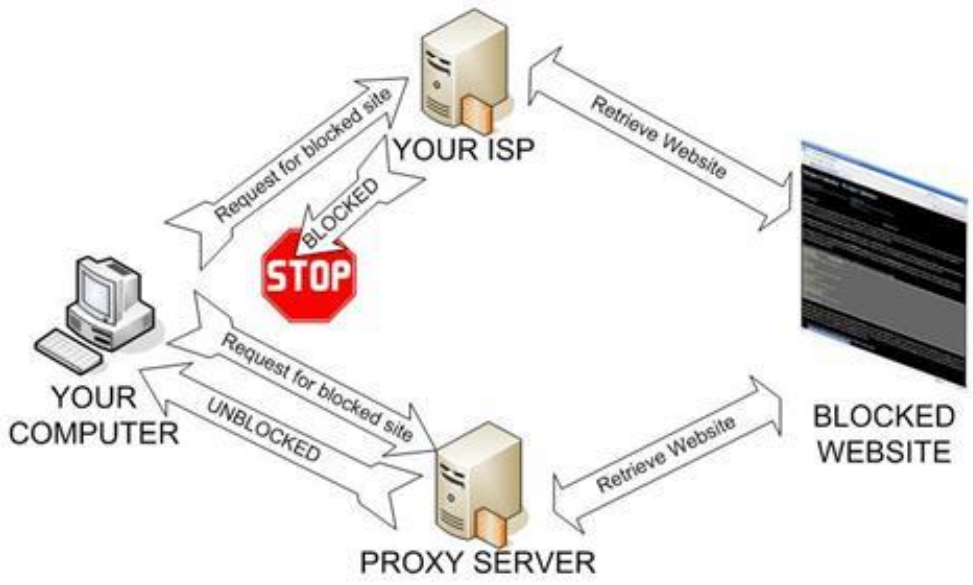
7.1 Khái niệm Proxy

Proxy là một Internet server làm nhiệm vụ chuyển tiếp thông tin và kiểm soát tạo sự an toàn cho việc truy cập Internet của các máy khách, còn gọi là khách hàng sử dụng dịch vụ internet. Trạm cài đặt proxy gọi là proxy server. Proxy hay trạm cài đặt proxy có địa chỉ IP và một cổng truy cập cố định. Ví dụ: 123.234.111.222:80 Địa chỉ IP của proxy trong ví dụ là 123.234.111.222 và cổng truy cập là 80.

7.2 Chức năng của proxy

Đối với một só hãng, công ty người ta sử dụng proxy vào việc:

-Proxy chia sẽ đường truyền: giúp nhiều máy truy cập Internet thông qua 1 máy, mà máy này gọi là Proxy server. Chỉ duy nhất máy Proxy này cần modem và account truy cập internet, các máy client (các máy trực thuộc) muốn truy cập internet qua máy này chỉ cần nối mạng LAN tới máy Proxy và truy cập địa chỉ yêu cầu.Những yêu cầu của người sử dung sẽ qua trung gian proxy server thay thế cho server thật sự mà người sử dụng cần giao tiếp, tại điểm trung gian này công ty kiểm soát được mọi giao tiếp từ trong công ty ra ngoài intrnet và từ internet vào máy của công ty. Sử dụng Proxy, công ty có thể cấm nhân viên truy cập những địa chỉ web không cho phép, cải thiện tốc độ truy cập nhờ sự lưu trữ cục bộ các trang web trong bộ nhớ của proxy server và giấu định danh địa chỉ của mạng nội bộ gây khó khăn cho việc thâm nhập từ bên ngoài vào các máy của công ty.



Đối với các nhà cung cấp dịch vụ đường truyền internet:

-Do trên mạng internet có lượng thông tin rất phong phú, theo quan điểm của từng quốc gia, của từng chủng tộc hay địa phương, các nhà cung cấp dịch vụ internet khu vực đó sẽ phối hợp proxy với kỹ thuật tường lửa để tạo ra một bộ lọc gọi là firewall proxy nhằm ngăn chặn các thông tin độc hại hoặc trái thuần phong mỹ tục đối với quốc gia, đói với chủng tộc hay địa phương đó. Địa chỉ các websit mà khách hàng yêu cầu truy cập sẽ được lọc tại bộ lọc này, nếu địa chỉ không bị cấm thì yêu cầu của khách hàng tiếp tục được gửi đi, tới các DNS server của các nhà cung cấp dịch vụ. Firewall proxy sẽ lọc tất cả các thông tin từ internet gửi vào máy của khách hàng và ngược lại.

7.3 Ý nghĩa của proxy

Proxy không chỉ có giá trị bởi nó làm được nhiệm vụ của một bộ lọc thông tin, nó còn tạo ra được sự an toàn cho các khách hàng của nó, firewal Proxy ngăn chặn hiệu quả sư xâm nhập của các đối tượng không mong muốn vào máy của khách hàng. Proxy lưu trữ dược các thông tin mà khách hàng cần trong bộ nhớ, do đó làm giảm thời gian truy tìm làm cho việc sử dụng băng thông hiệu quả.

Proxy server giống như một vệ sĩ bảo vệ khỏi những rắc rối trên Internet. Một Ps thường nằm bên trong tường lửa, giữa trình duyệt web và server thật, làm chức năng tạm giữ những yêu cầu Internet của các máy khách để chúng không giao tiếp trực tiếp Internet. Người dùng sẽ không truy cập được những trang web không cho phép (bị công ty cấm).

Mọi yêu cầu của máy khách phải qua Ps, nếu địa chỉ IP có trên proxy, nghĩa là Website này được lưu trữ cục bộ, thì trang này sẽ đươc truy cập mà không cần phải kết nối Internet, nếu không có trên Ps và trang này không bị cấm yêu cầu sẽ được

chuyển đến server thật, DNS server... và ra Internet. Ps lưu trữ cục bộ các trang Web thường truy cập nhất trong bộ đệm giảm chi phí, tốc độ hiển thị trang Web nhanh.

Proxy server bảo vệ mạng nội bộ khỏi bị xác định bởi bên ngoài bằng cách mang lại cho mạng hai định danh: một cho nội bộ, một cho bên ngoài. Điều này tạo ra một "bí danh" đối với thế giới bên ngoài gây khó khăn đối với nếu người dùng "tự tung tự tác" hay các tay bẻ khóa muốn xâm nhập trực tiếp máy nào đó.

7.4 Cách sử dụng proxy hiệu quả

Do các proxy có quy mô bộ nhớ khác nhau và số lượng người đang sử dụng proxy nhiều-ít khác nhau, Proxy server hoạt động quá tải thì tốc độ truy cập internet của khách hàng có thể bị chậm. Mặt khác một số websit khách hàng có đầy đủ điều kiện nhân thân để đọc, nghiên cứu nhưng bị tường lửa chặn không truy cập được thì biện pháp đổi proxy để truy cập là điều cần thiết nhằm đảm bảo công việc. Do đó người sử dụng có thể chọn proxy server để sử dụng cho riêng mình. Có các cách chọn lựa cho người sử dụng. Sử dụng proxy mặc định của nhà cung cấp dịch vụ (internet), trường hợp này người sử dụng không cần điền địa chỉ IP của proxy vào cửa sổ internet option của trình duyệt trong máy của mình. Sử dụng proxy server khác (phải trả phí hoặc miễn phí) thì phải điền địa chỉ IP của proxy server vào cửa sổ internet option của trình duyệt.

Một số Proxy miễn phí tham khảo

IP address	Anonymity level	Checked time	Country
203.252.226.215:8001	transparent	Dec-26, 18:26	Korea, Republic of
81.20.173.65:3128	transparent server	Dec-26, 10:08	Russian Federation
81.92.147.62:3128	transparent	Dec-26, 18:15	Czech Republic
189.15.68.99:3128	transparent	Dec-26, 16:14	Brazil
141.223.175.140:8080	transparent proxy server	Dec-26, 15:57	Korea, Republic of
85.226.20.158:3128	transparent proxy	Dec-26, 14:49	Sweden
66.29.36.95:554	hight-anonymous ($)	Dec-27, 06:01	United States
88.243.48.144:3128	transparent	Dec-26, 18:22	Turkey
213.144.14.66:3128	transparent server	Dec-26, 20:44	Germany
212.62.102.247:80	transparent proxy	Dec-26, 14:43	Saudi Arabia
61.19.237.202:8080	transparent	Dec-26, 09:55	Thailand
61.50.217.230:1080	transparent proxy	Dec-26, 09:55	China
202.206.100.39:3128	transparent proxy	Dec-26, 15:59	China
61.142.249.116:8080	transparent proxy	Dec-26, 09:59	China
118.98.212.242:3128	transparent proxy	Dec-26, 19:05	Indonesia
203.73.180.16:3128	transparent	Dec-25, 18:26	Taiwan
200.129.25.3:8080	transparent proxy	Dec-26, 18:29	Brazil
203.190.51.206:8080	transparent server	Dec-25, 14:21	Indonesia
212.62.102.134:80	transparent server	Dec-25, 14:23	Saudi Arabia

7.5 Nguyên lý hoạt động của Firewall

Khái niệm tường lửa (Firewall)

Trong ngành mạng máy tính, bức tường lửa (tiếng Anh: firewall) là rào chắn mà một số cá nhân, tổ chức, doanh nghiệp, cơ quan nhà nước lập ra nhằm ngăn chặn người dùng mạng Internet truy cập các thông tin không mong muốn hoặc/và ngăn chặn người dùng từ bên ngoài truy nhập các thông tin bảo mật nằm trong mạng nội bộ.

Tường lửa là một thiết bị phần cứng và/hoặc một phần mềm hoạt động trong một môi trường máy tính nối mạng để ngăn chặn một số liên lạc bị cấm bởi chính sách an ninh của cá nhân hay tổ chức, việc này tương tự với hoạt động của các bức tường ngăn lửa trong các tòa nhà. Tường lửa còn được gọi là Thiết bị bảo vệ biên giới (Border Protection Device - BPD), đặc biệt trong các ngữ cảnh của NATO, hay bộ lọc gói tin (packet filter) trong hệ điều hành BSD - một phiên bản Unix của Đại học California, Berkeley.

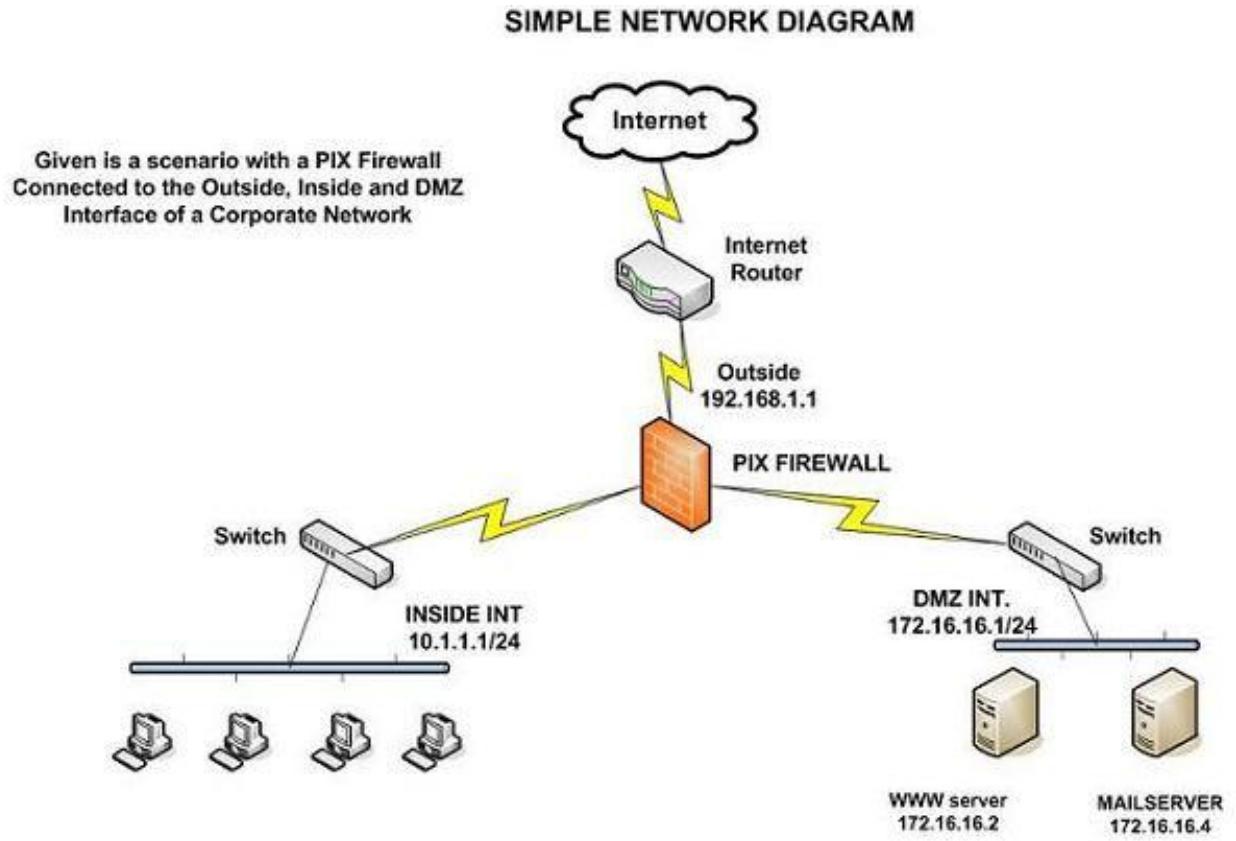
Nhiệm vụ cơ bản của tường lửa là kiểm soát giao thông dữ liệu giữa hai vùng tin cậy khác nhau. Các vùng tin cậy (zone of trust) điển hình bao gồm: mạng Internet (vùng không đáng tin cậy) và mạng nội bộ (một vùng có độ tin cậy cao). Mục đích

cuối cùng là cung cấp kết nối có kiểm soát giữa các vùng với độ tin cậy khác nhau thông qua việc áp dụng một chính sách an ninh và mô hình kết nối dựa trên nguyên tắc quyền tối thiểu (principle of least privilege).

Cấu hình đúng đắn cho các tường lửa đòi hỏi kĩ năng của người quản trị hệ thống. Việc này đòi hỏi hiểu biết đáng kể về các giao thức mạng và về an ninh máy tính. Những lỗi nhỏ có thể biến tường lửa thành một công cụ an ninh vô dụng.

Lịch sử phát triển Firewall

Công nghệ tường lửa bắt đầu xuất hiện vào cuối những năm 1980 khi Internet vẫn còn là một công nghệ khá mới mẻ theo khía cạnh kết nối và sử dụng trên toàn cầu. Ý tưởng đầu tiên được đã hình thành sau khi hàng loạt các vụ xâm phạm nghiêm trọng đối với an ninh liên mạng xảy ra vào cuối những năm 1980. Năm 1988, một nhân viên tại trung tâm nghiên cứu NASA Ames tại California gửi một bản ghi nhớ qua thư điện tử tới đồng nghiệp rằng: "Chúng ta đang bị một con VIRUS Internet tấn công! Nó đã đánh Berkeley, UC San Diego, Lawrence Livermore, Stanford, và NASA Ames." Con virus được biết đến với tên Sâu Morris này đã được phát tán qua thư điện tử và khi đó đã là một sự khó chịu chung ngay cả đối với những người dùng vô thưởng vô phạt nhất. Sâu Morris là cuộc tấn công diện rộng đầu tiên đối với an ninh Internet. Cộng đồng mạng đã không hề chuẩn bị cho một cuộc tấn công như vậy và đã hoàn toàn bị bất ngờ. Sau đó, cộng đồng Internet đã quyết định rằng ưu tiên tối cao là phải ngăn chặn không cho một cuộc tấn công bất kỳ nào nữa có thể xảy ra, họ bắt đầu cộng tác đưa ra các ý tưởng mới, những hệ thống và phần mềm mới để làm cho mạng Internet có thể trở lại an toàn.



Năm 1988, bài báo đầu tiên về công nghệ tường lửa được công bố, khi Jeff Mogul thuộc Digital Equipment Corp. phát triển các hệ thống lọc đầu tiên được biết đến với tên các tường lửa lọc gói tin. Hệ thống khá cơ bản này đã là thế hệ đầu tiên của cái mà sau này sẽ trở thành một tính năng kỹ thuật an toàn mạng được phát triển cao. Từ năm 1980 đến năm 1990, hai nhà nghiên cứu tại phòng thí nghiệm AT&T Bell, Dave Presetto và Howard Trickey, đã phát triển thế hệ tường lửa thứ hai, được biến đến với tên các tường lửa tầng mạch (circuit level firewall). Các bài báo của Gene Spafford ở Đại học Purdue, Bill Cheswick ở phòng thí nghiệm AT&T và Marcus Ranum đã mô tả thế hệ tường lửa thứ ba, với tên gọi tường lửa tầng ứng dụng (application layer firewall), hay tường lửa dựa proxy (proxy-based firewall). Nghiên cứu công nghệ của Marcus Ranum đã khởi đầu cho việc tạo ra sản phẩn thương mại đầu tiên. Sản phẩm này đã được Digital Equipment Corporation's (DEC) phát hành với tên SEAL. Đợt bán hàng lớn đầu tiên của DEC là vào ngày 13 tháng 9 năm 1991 cho một công ty hóa chất tại bờ biển phía Đông của Mỹ.

Tại AT&T, Bill Cheswick và Steve Bellovin tiếp tục nghiên cứu của họ về lọc gói tin và đã phát triển một mô hình chạy được cho công ty của chính họ, dựa trên kiến trúc của thế hệ tường lửa thứ nhất của mình. Năm 1992, Bob Braden và Annette DeSchon tại Đại học Nam California đã phát triển hệ thống tường lửa lọc gói tin thế hệ thứ tư. Sản phẩm có tên “Visas” này là hệ thống đầu tiên có một giao diện

với màu sắc và các biểu tượng, có thể dễ dàng cài đặt thành phần mềm cho các hệ điều hành chẳng hạn Microsoft Windows và Mac/OS của Apple và truy nhập từ các hệ điều hành đó. Năm 1994, một công ty Israel có tên Check Point Software Technologies đã xây dựng sản phẩm này thành một phần mềm sẵn sàng cho sử dụng, đó là FireWall-1. Một thế hệ thứ hai của các tường lửa proxy đã được dựa trên công nghệ Kernel Proxy. Thiết kế này liên tục được cải tiến nhưng các tính năng và mã chương trình cơ bản hiện đang được sử dụng rộng rãi trong cả các hệ thống máy tính gia đình và thương mại. Cisco, một trong những công ty an ninh mạng lớn nhất trên thế giới đã phát hành sản phẩm này năm 1997.

Thế hệ FireWall-1 mới tạo thêm hiệu lực cho động cơ kiểm tra sâu gói tin bằng cách chia sẻ chức năng này với một hệ thống ngăn chặn xâm nhập.

Các loại tường lửa

Có ba loại tường lửa cơ bản tùy theo:

- Truyền thông được thực hiện giữa một nút đơn và mạng, hay giữa một số mạng.
- Truyền thông được chặn tại tầng mạng, hay tại tầng ứng dụng.
- Tường lửa có theo dõi trạng thái của truyền thông hay không.

Phân loại theo phạm vi của các truyền trông được lọc, có các loại sau:

- Tường lửa cá nhân, một ứng dụng phần mềm với chức năng thông thường là lọc dữ liệu ra vào một máy tính đơn.
- Tường lửa mạng, thường chạy trên một thiết bị mạng hay máy tính chuyên dụng đặt tại ranh giới của hai hay nhiều mạng hoặc các khu phi quân sự (mạng con trung gian nằm giữa mạng nội bộ và mạng bên ngoài). Một tường lửa thuộc loại này lọc tất cả giao thông dữ liệu vào hoặc ra các mạng được kết nối qua nó.

  Loại tường lửa mạng tương ứng với ý nghĩa truyền thống của thuật ngữ "tường lửa" trong ngành mạng máy tính.

  Khi phân loại theo các tầng giao thức nơi giao thông dữ liệu có thể bị chặn, có ba loại tường lửa chính:

- Tường lửa tầng mạng. Ví dụ iptables.
- Tường lửa tầng ứng dụng. Ví dụ TCP Wrappers.
- Tường lửa ứng dụng. Ví dụ: hạn chế các dịch vụ ftp bằng việc định cấu hình tại tệp /etc/ftpaccess.

Các loại tường lửa tầng mạng và tường lửa tầng ứng dụng thường trùm lên nhau, mặc dù tường lửa cá nhân không phục vụ mạng, nhưng một số hệ thống đơn đã cài đặt chung cả hai.

Cuối cùng, nếu phân loại theo tiêu chí rằng tường lửa theo dõi trạng thái của các kết nối mạng hay chỉ quan tâm đến từng gói tin một cách riêng rẽ, có hai loại tường lửa:

- Tường lửa có trạng thái (Stateful firewall)
- Tường lửa phi trạng thái (Stateless firewall)

Lý do sử dụng tường lửa

Mạng internet ngày càng phát triển và phổ biến rộng khắp mọi nơi, lợi ích của nó rất lớn. Tuy nhiên cũng có rất nhiều ngoại tác không mong muốn đối với các cá nhân là cha mẹ hay tổ chức, doanh nghiệp, cơ quan nhà nước... như các trang web không phù hợp lứa tuổi, nhiệm vụ, lợi ích, đạo đức, pháp luật hoặc trao đổi thông tin bất lợi cho cá nhân, doanh nghiệp... Do vậy họ (các cá nhân, tổ chức, cơ quan và nhà nước) sử dụng tường lửa để ngăn chặn.

Một lý do khác là một số quốc gia theo chế độ độc tài, độc đảng áp dụng tường lửa để ngăn chận quyền trao đổi, tiếp cận thông tin của công dân nước mình không cho họ truy cập vào các trang web hoặc trao đổi với bên ngoài, điều mà nhà cầm quyền cho rằng không có lợi cho chế độ đó.

Cách thức ngăn chặn

Để ngăn chặn các trang web không mong muốn, các trao đổi thông tin không mong muốn người ta dùng cách lọc các địa chỉ web không mong muốn mà họ đã tập hợp được hoặc lọc nội dung thông tin trong các trang thông qua các từ khóa để ngăn chặn những người dùng không mong muốn truy cập vào mạng và cho phép người dùng hợp lệ thực hiện việc truy xuất.

Bức tường lửa có thể là một thiết bị định hướng (Router, một thiết bị kết nối giữa hai hay nhiều mạng và chuyển các thông tin giữa các mạng này) hay trên một máy chủ (Server), bao gồm phần cứng và/hoặc phần mềm nằm giữa hai mạng (chẳng hạn mạng Internet và mạng liên kết các gia đình, điểm kinh doanh internet, tổ chức, công ty, hệ thống Ngân hàng, cơ quan nhà nước.

Cơ quan nhà nước có thể lập bức tường lửa ngay từ cổng Internet quốc gia hoặc yêu cầu các nhà cung cấp dịch vụ đường truyền (IXP) và cung cấp dịch vụ Internet (ISP) thiết lập hệ thống tường lửa hữu hiệu hoặc yêu cầu các đại lý kinh doanh internet thực hiện các biện pháp khác như Thông tư liên tịch số 02/2005/TTLT về quản lý đại lý Internet có hiệu lực vào đầu tháng 8-2005 ở Việt Nam.

Vượt tường lửa

Các trang web bị chặn nhất là các trang web sex thường rất linh động thay đổi địa chỉ để tránh sự nhận diện hoặc nhanh chóng thông báo địa chỉ mới một cách hạn chế với các đối tượng dùng đã định.

Người dùng ở các nước có hệ thống tường lửa có thể tiếp cận với nội dung bị chặn qua các ngõ khác bằng cách thay đổi địa chỉ Proxy, DNS hoặc qua vùng nhớ đệm cached của trang tìm kiếm thông dụng như Google, Yahoo..., hoặc sử dụng phần mềm miễn phí Tor. Nói chung người dùng mạng hiểu biết nhiều về máy tính thì biết nhiều kỹ xảo vượt tường lửa.

Hiệu quả khi sử dụng tường lửa

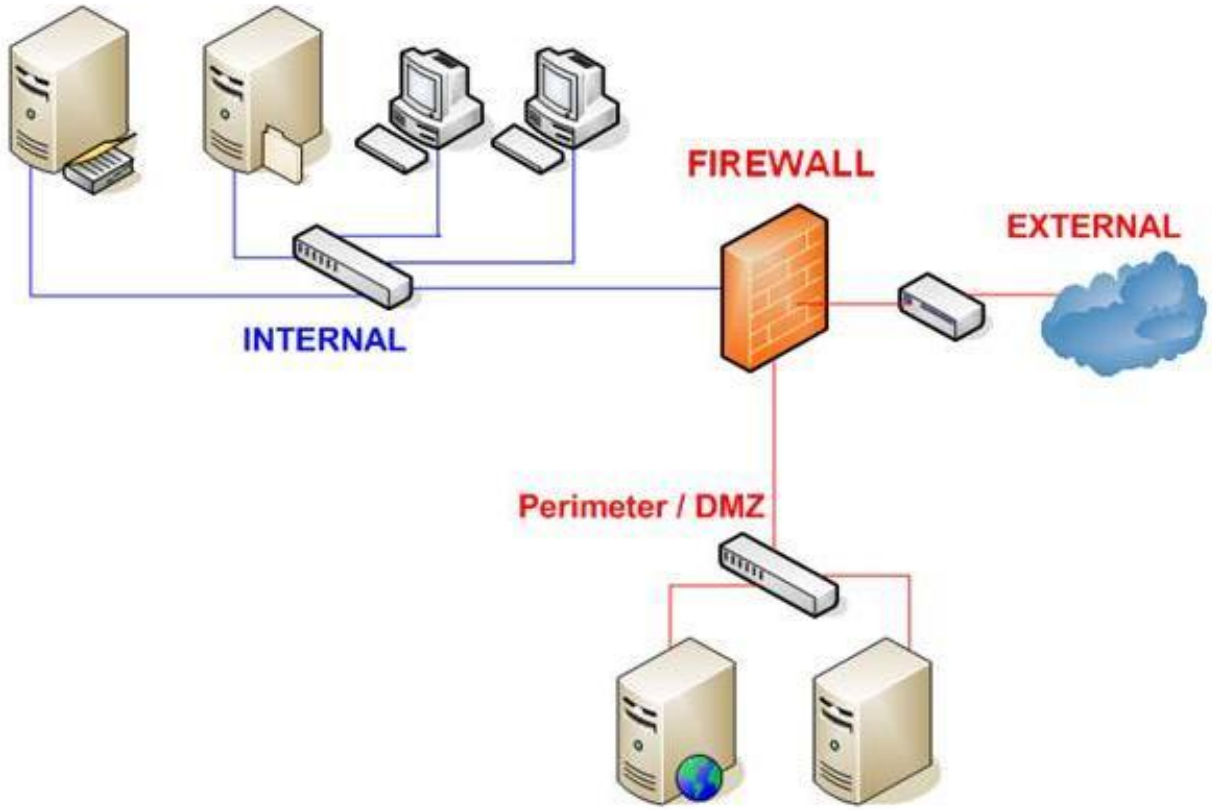
Bức tường lửa chỉ có hiệu quả tốt một thời gian sau đó các trang web bị chặn cũng như người sử dụng dùng mưu mẹo, kỹ xảo, kỹ thuật để né và vuợt tường, vì vậy phải luôn luôn cập nhật kỹ thuật, nhận điện các địa chỉ mới để thay đổi phương thức hoạt động, điều này làm tốc độ truy cập chung bị giảm và đòi hỏi phải nâng cấp trang thiết bị, kỹ thuật.

Nhược điểm khi sử dụng tường lửa

Sử dụng tường lửa cần phải xử lý một lượng lớn thông tin nên việc xử lý lọc thông tin có thể làm chậm quá trình kết nối của người kết nối.

Việc sử dụng tường lửa chỉ hữu hiệu đối với những người không thành thạo kỹ thuật vượt tường lửa, những người sử dụng khác có hiểu biết có thể dễ dàng vượt qua tường lửa bằng cách sử dụng các proxy không bị ngăn chặn.

7.6 Triển khai xây dựng hệ thống tường lửa cho doanh nghiệp



Mô tả sơ đồ hệ thống:

- Gồm 01 PC đóng vai trò Domain Controller (DC)
- Mạng LAN thuộc dải IP 192.168.1.0/24
- DMZ thuộc dải IP 172.16.1.0/24
- External có dải IP 10.0.0.0/30
- Firewall có 03 Fast Ethernet tương ứng 03 phân vùng LAN (Internal), DMZ và External

Yêu cầu:

- Các PC join vào Domain (DC)
- File Server và Web Server thuộc vùng DMZ cho phép các PC thuộc LAN truy cập vào
- Các PC thuộc LAN có thể truy cập Internet theo sự cho phép của Firewall

7.7 Thiết lập tường lửa và các mặc định bảo mật kết nối IPsec

Windows Server 2003 đã có một tường lửa khá cơ bản cho phép bảo vệ máy tính chống lại các kết nối bên ngoài mà bạn không muốn chúng kết nối với máy chủ. Vấn đề này rất hữu dụng trong việc bảo vệ các máy Windows Server 2003, tuy nhiên nó khá đơn giản và không cho phép điều khiển hạt nhân cho cả truy cập

đi vào và gửi đi đối với các máy Windows Server 2003. Thêm vào đó, tường lửa trong Windows Server 2003 lại không được tích hợp chặt chẽ với các dịch vụ đã được cài đặt, chính vì vậy bạn phải cấu hình tường lửa bất cứ khi nào thêm vào một máy chủ mới hoặc dịch vụ mới.

Windows Server 2008 giới thiệu một tường lửa mới và có nhiều cải thiện đáng kể; Windows Firewall với Advanced Security. Tường lửa mới này trong Windows có nhiều cải thiện và cũng rất giống với tường lửa được giới thiệu trong Windows Vista. Các tính năng mới có trong tường lửa mới này về vấn đề bảo mật nâng cao gồm có:

- Nhiều điều khiển truy cập đi vào
- Nhiều điều khiển truy cập gửi đi
- Tích hợp chặt chẽ với Windows Server 2008 Server Manager, với cấu hình tự động của tường lửa khi các dịch vụ được cài đặt bằng Server Manager.
- Cấu hình và việc quản lý chính sách IPsec được cải thiện mạnh mẽ, bên cạnh đó là còn có cả sự thay đổi tên. Các chính sách IPsec hiện được khai báo như các rule bảo mật kết nối (**Connection Security Rules)**.
- Kiểm tra chính sách tường lửa được cải thiện
- Kiểm tra các chính sách IPsec được cải thiện (giờ đây được gọi là các Rule bảo mật kết nối)
- Kiểm tra tập trung các trong việc kết hợp chế độ bảo mật Main và Quick được cải thiện

Có nhiều tùy chọn cấu hình có trong tường lửa mới này, chính vì vậy trong bài này chúng tôi sẽ chia làm ba phần, phần đầu là giới thiệu về các tùy chọn cấu hình cơ bản cho tường lửa và cho các chính sách IPsec. Phần hai sẽ tập trung đến cách tạo các rule đi vào và gửi đi, còn trong phần ba sẽ giới thiệu về cách tạo các rule bảo mật kết nối.

Giao diện điều khiển Windows Firewall with Advanced Security có thể được mở từ menu **Administrative Tools.** Khi mở giao diện điều khiển này, bạn sẽ thấy được phần panel bên trái như những gì thể hiện trong hình dưới đây.

Hình 1

Panel ở giữa của giao diện điều khiển sẽ cung cấp cho bạn các thông tin về profile Domain, Private và Public. Mặc định các giá trị này cho mỗi profile là:

- Windows Firewall là "on"
- Kết nối gửi vào không hợp lệ với rule sẽ bị khóa
- Kết nối gửi đi không hợp lệ với rule sẽ được cho phép

Với những người có quyền quản trị tường lửa thì thành phần cuối cùng dường như có phần hơi lộn xộn vì trên các tường lửa của mạng, nếu không có rule nào cho phép một kết nối thì mặc định rule "clean up" được kích hoạt và kết nối sẽ bị khóa. Tuy vậy trong phần nội dung của Windows Firewall trên host với Advanced Security, kết nối gửi đi không hợp lệ với rule sẽ được cho phép có nghĩa rằng không có kiểm soát truy cập gửi đi được thiết lập mặc định.

Hình 2

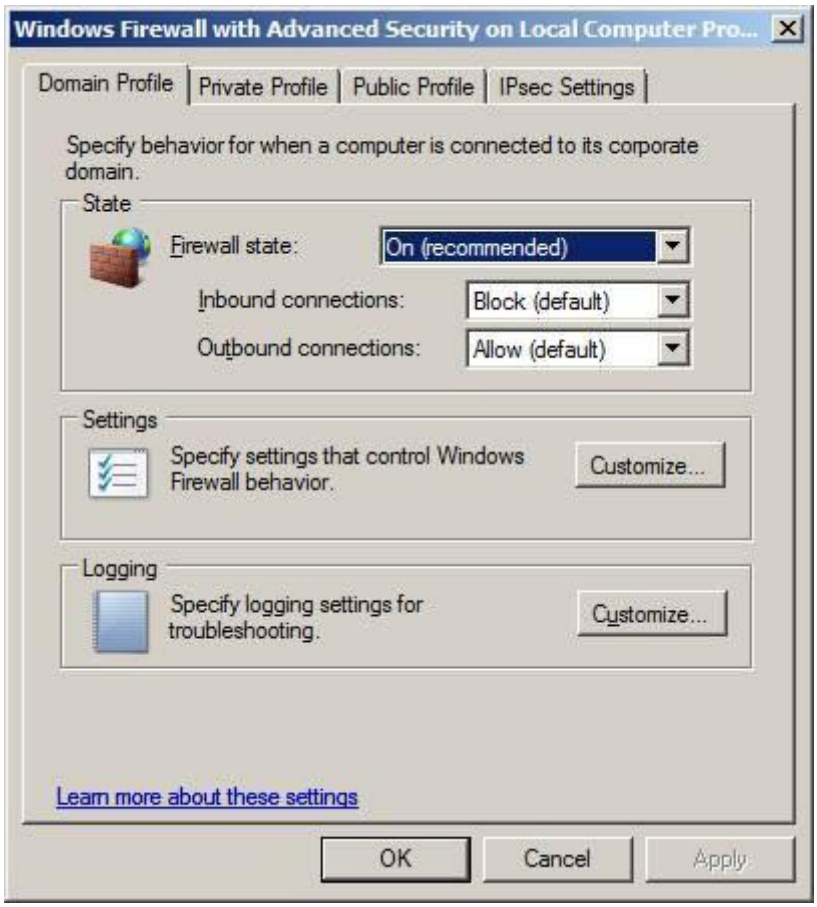
Trong phần panel phái bên trái của giao diện điều khiển **Windows Firewall with Advanced Security,** kích chuột phải vào nút **Windows Firewall with Advanced Security** ở phần trên của panel trái của giao diện điều khiển và kích vào **Properties**. Thao tác này sẽ mở ra hộp thoại **Windows Firewall with Advanced Security Properties**

Domain Profile là tab đầu tiên xuất hiện trong hộp thoại **Windows Firewall with Advanced Security Properties**. Domain Profile áp dụng khi máy tính được kết nối với mạng công ty và có thể liên lạc với miền. Vì các máy chủ không chuyển từ mạng này sang mạng khác nên chỉ có Domain Profile có thể áp dụng trong đại đa số các trường hợp. Ngoại từ khi máy chủ không phải là thành viên miền, trong trường hợp đó thì **Private Profile** sẽ áp dụng nó.

Khung **State** bạn cấu hình như sau:

- **Firewall state**. Trạng thái này có thể off hoặc on. Nó được mặc định và nên để như vậy.
- **Inbound connections**. Các thiết lập mặc định để khóa. Điều này có nghĩa rằng các kết nối không có một rule cho phép sẽ bị khóa. Có hai tùy chọn khác ở đây: **Allow**, tùy chọn này sẽ cho phép tất cả các kết nối gửi đến và **Block all connections**, sẽ khóa tất cả các kết nối gửi đi. Bạn nên cẩn thận với cả hai thiết lập có thể thay đổi này, vì tùy chọn **Block all connections** có thể khóa tất cả các kết nối gửi đến, điều đó sẽ làm khó khăn hơn trong việc quản lý máy tính từ mạng.
- **Outbound connections.** Thiết lập mặc định là **Allow (default)**, cho phép kết nối gửi đi. Một tùy chọn khác ở đây là khóa các kết nối gửi đi. Chúng tôi khuyên bạn nên chọn mặc định bằng không máy tính sẽ không thể kết nối với các máy tính khác. Có một số ngoại lệ như các thiết bị được kết nối Internet đang chỉ nên xử lý các kết nối gửi đến và không nên thiết lập các kết nối gửi đi mới.

Trong khung **Settings** bạn có thể cấu hình các thiết lập để điều khiển một số hành vi của tường lửa. Kích nút **Customize**.



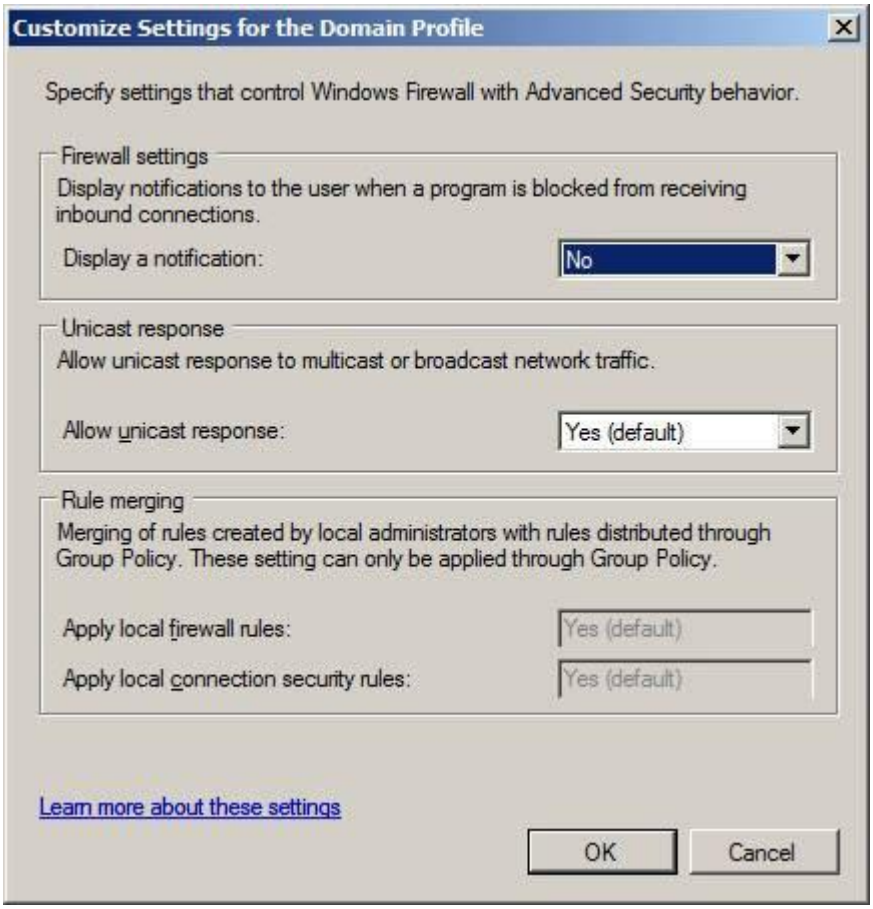
Hình 3

Khi đó hộp thoại **Customize Settings for the Domain Profile** sẽ xuất hiện. Trong khung **Firewall settings,** bạn cấu hình muốn hay không muốn có thông báo được hiển thị khi kết nối gửi đến bị khóa. Thiết lập này được mặc định là **No** và bạn nên để lại lựa chọn mặc định đó. Vì nếu chọn yes thì sẽ gặp phải rất nhiều thông báo cho các kết nối được gửi đi đến máy chủ.

Trong khung **Unicast response**, bạn cấu hình cách máy tính đáp trả đối với lưu lượng mạng multicast và broadcast. Giá trị mặc định là **Yes (default)**, giá trị này cho phép máy chủ cung cấp các đáp trả unicast (đơn) với các yêu cầu multicast (đa). Nếu bạn ở trong môi trường không phụ thuộc vào các thư tín multicast hoặc broadcast (bạn không có các ứng dụng multicast trên máy chủ và máy chủ không phụ thuộc vào các giao thức dựa trên broadcast như NetBIOS trên TCP/IP), sau đó bạn có thể điều chỉnh về **No**.

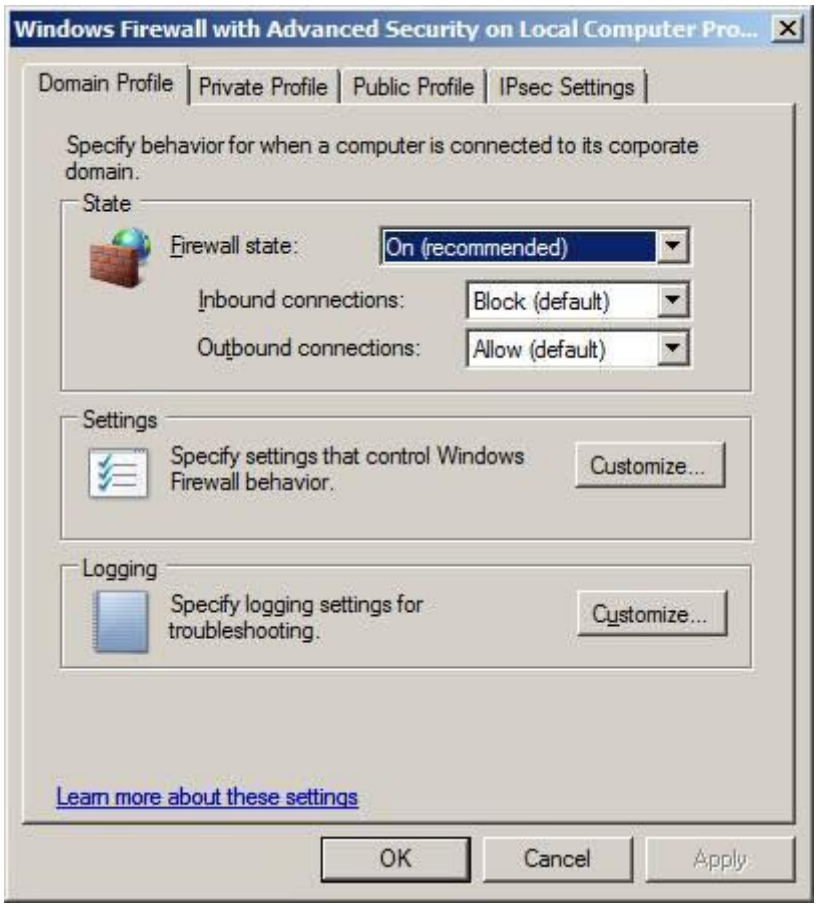
Khung cuối cùng không thể cấu hình thông qua giao diện và phải được cấu hình thông qua Group Policy. Khung **Merging** thể hiện cho bạn cách các client sẽ xử lý như thế nào với các rule đến từ tập rule tường lửa nội bộ và rule tường lửa đã được cấu hình thông qua Group Policy ra sao. Các thiết lập mặc định trong Group Policy

đều áp dụng cả hai rule tường lửa nội bộ và rule bảo mật kết nối nội bộ. Như với các thiết lập Group Policy khác, thứ tự ưu tiên được xác định bởi LSDOU.



Hình 4

Trên hộp thoại **Windows Firewall with Advanced Security Properties**, trong khung **Logging** bạn có thể cấu hình một số tùy chọn cho việc **Logging** cho Windows Firewall. Kích **Customize**.



Hình 5

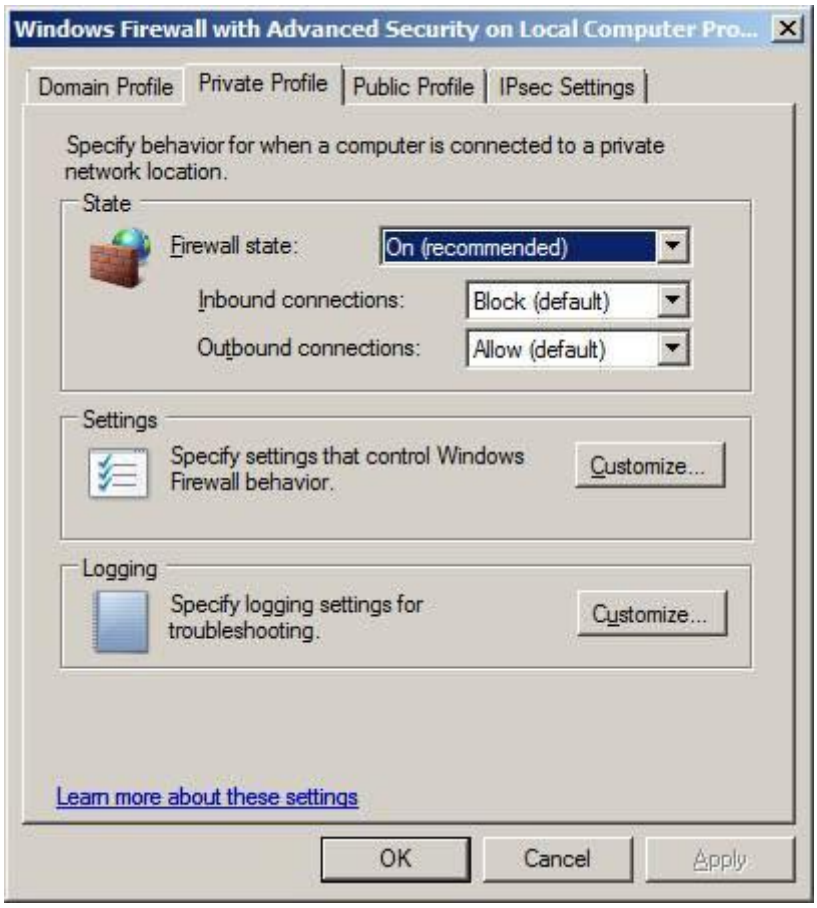
Khi đó hộp thoại **Customize Logging Settings for the Domain Profile** sẽ xuất hiện. Tên của file bản ghi mặc định sẽ là **pfireall.log** và được lưu trong vị trí mặc định trên đĩa cứng nội bộ. Bạn có thể thay đổi vị trí này nếu thích bằng cách đánh vào đó một đường dẫn mới vào hộp Name hoặc kích vào nút **Browse.**

Giá trị **Size limit (KB)** được mặc định cho kích thước file bản ghi là 4 MB (**4096 KB)**. Bạn có thể giảm hoặc tăng kích thước này nếu muốn. Sau khi bản ghi được điền, các entry cũ sẽ bị xóa và entry mới sẽ được bổ sung.

Mặc định, **Log dropped packets** và **Log successful connections** được thiết lập là **No (default)**. Lưu ý nếu bạn cấu hình cả hai thiết lập này thì sẽ không có gì để ghi cho file bản ghi .J

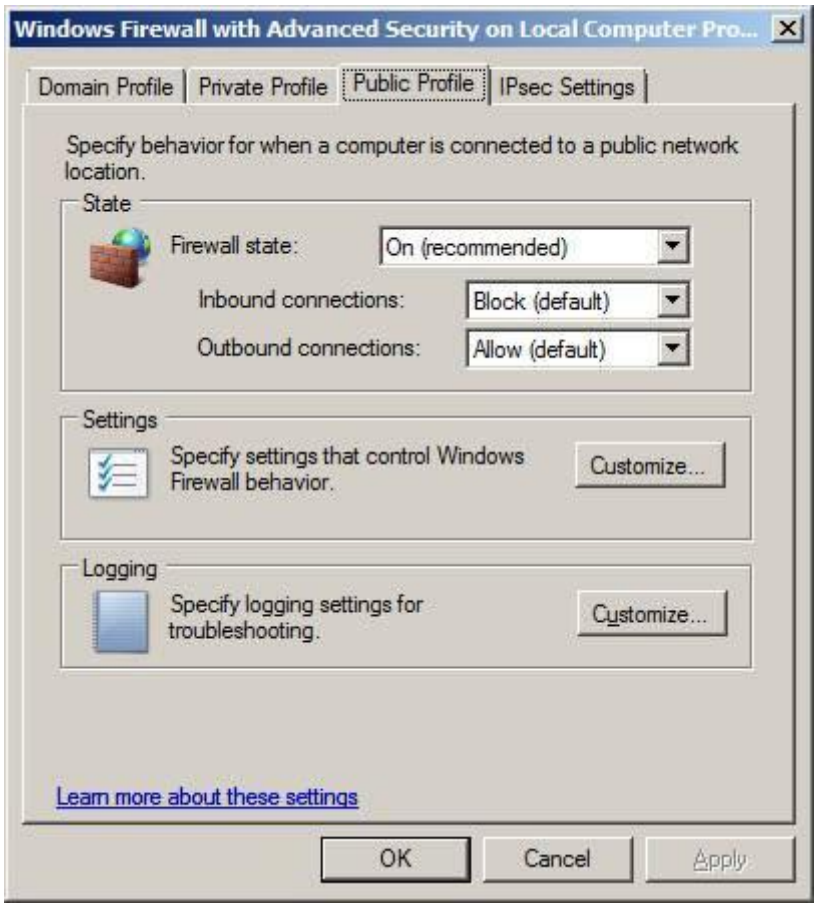
Hình 6

Trong tab, bạn có thể cấu hình các thiết lập tường lửa giống với thiết lập mà bạn đã thực hiện trên tab, tuy nhiên các thiết lập này sẽ chỉ ảnh hưởng khi máy tính của bạn được kết nối với mạng riêng, mạng riêng này không được kết nối với miền. Các thiết lập này không được áp dụng khi một máy chủ thành viên miền là một thành viên của miền, vì sẽ không được xóa on và off cho mạng, chính vì vậy nó luôn là miền được kết nối, bằng không nó sẽ không thực hiện tất cả các chức năng.



Hình 7

Trong tab **Public Profile**, bạn cấu hình các thiết lập áp dụng khi máy tính kết nối với mạng công cộng. Các cấu hình này sẽ không áp dụng cho các máy chủ vì chúng được sử dụng khi máy tính được kết nối với một mạng công cộng.

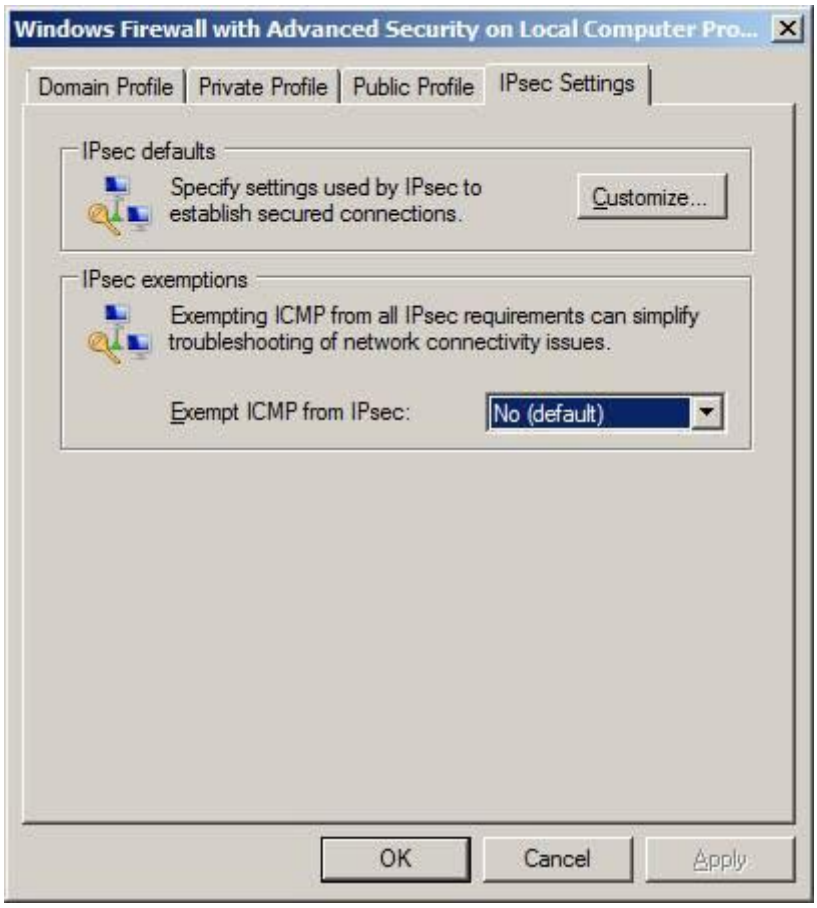


Hình 8

Trong tab **IPsec Settings**, có hai khung:

- **IPsec defaults**. Có các thiết lập IPsec mặc định được áp dụng khi bạn tạo các rule bảo mật kết nối (tên mới cho các chính sách IPsec). Lưu ý rằng khi tạo các Rule bạn sẽ có tùy chọn để thay đổi các thiết lập trên mỗi rule từ các thiết lập mặc định.
- **IPsec exemptions**. Mặc định, các IPsec exemptions được vô hiệu hóa. Tuy vậy bạn có thể khắc phục sự cố mạng bằng cách sử dụng Ping, tracert và các công cụ ICMP khác dễ dàng hơn rất nhiều nếu thay đổi nó từ chế độ mặc định **No (default)** thành **Yes**.

Kích nút **Customize** trong khung **IPsec defaults**.



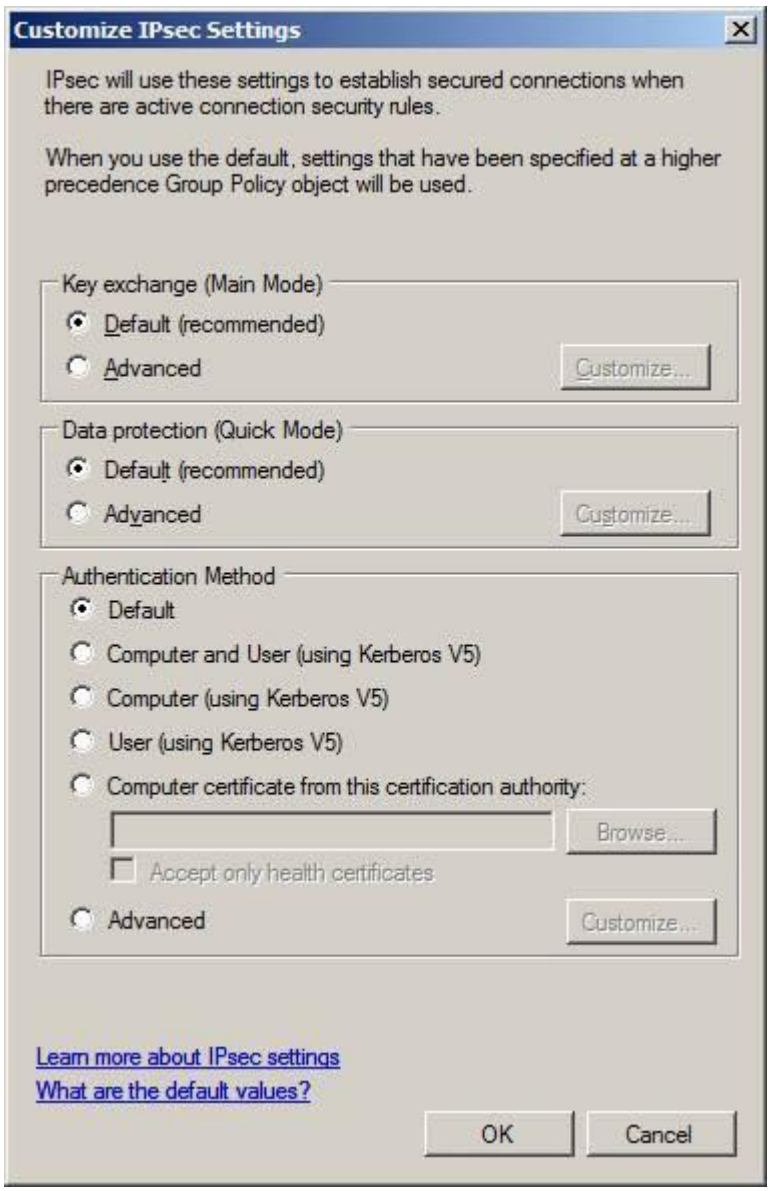
Hình 9

Trong hộp thoại **Customize IPsec Settings** , bạn có thể cấu hình như dưới đây:

- **Key Exchange (Main Mode)**
- **Data Protection (Quick Mode)**
- **Authentication Method**

Mỗi một tùy chọn này đều được cấu hình bằng một tập các giá trị mặc định mà Microsoft đã cân nhắc từ trước. Mặc dù vậy chúng đều có thể được thiết lập lại. Với **Key exchange (Main Mode)** và **Data Protection (Quick Mode)** bạn cần chọn tùy chọn **Advanced.** Còn với các tùy chọn **Authentication Method** bạn có thể chọn tùy chọn khác hoặc sử dụng tùy chọn **Advanced** để điều khiển tinh chỉnh hơn các phương pháp chứng thực.

Trong khung **Key exchange (Main Mode)**, bạn hãy kích vào tùy chọn **Advanced**, sau đó kích nút **Customize**.

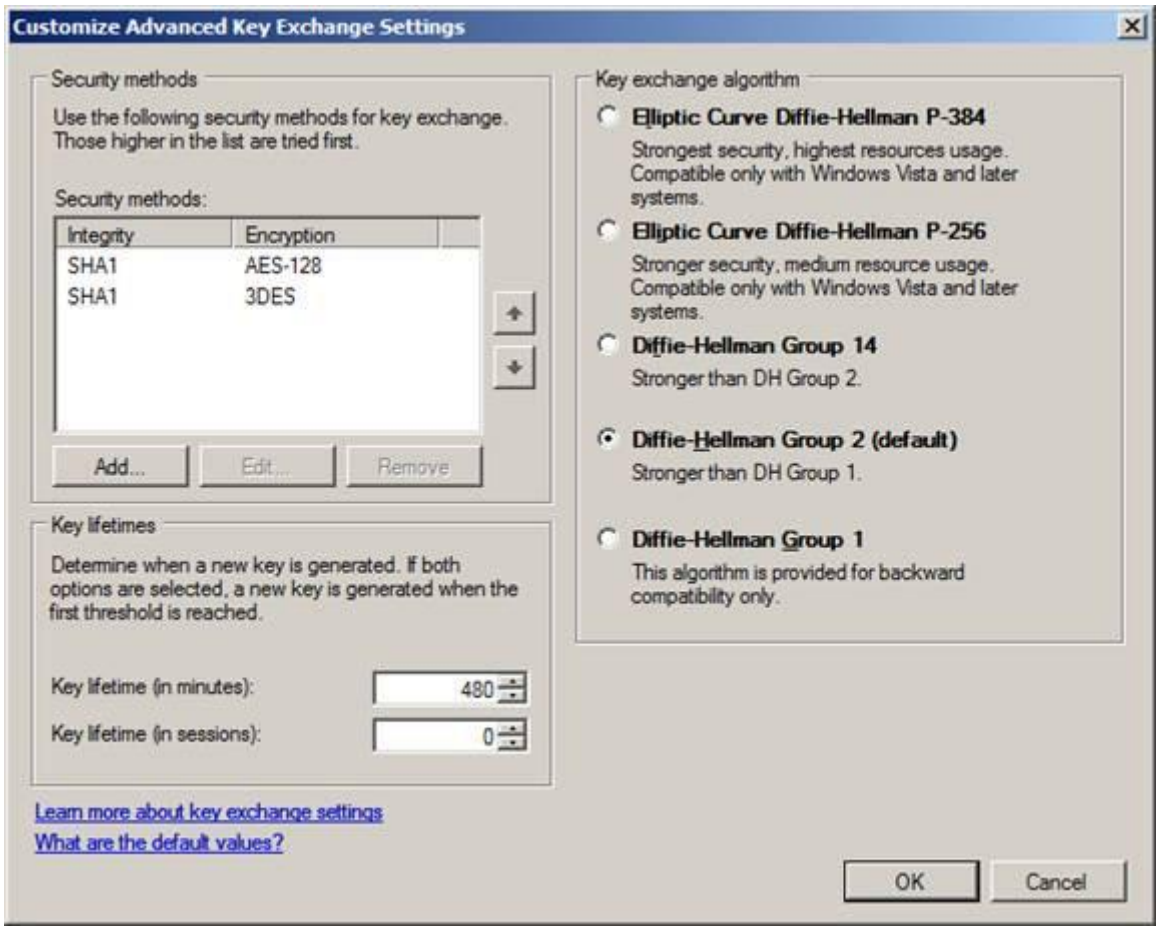


Hình 10

Khi đó hộp thoại **Customize Advanced Key Exchange Settings** sẽ xuất hiện. Các thiết lập mặc định được thể hiện ở đây. Như những gì bạn thấy, **AES-128** là phương pháp ưu tiên được sử dụng cho key exchange và nếu không có sẵn ở đây thì nó sẽ là **3DES**. Key lifetimes cũng được cấu hình trên trang này. **Key exchange algorithm** được thiết lập mặc định là **Diffie-Hellman Group 2.** Group 1 là 768 bits, Group 2 là 1024 bits và Group 14 là 2048 bits.

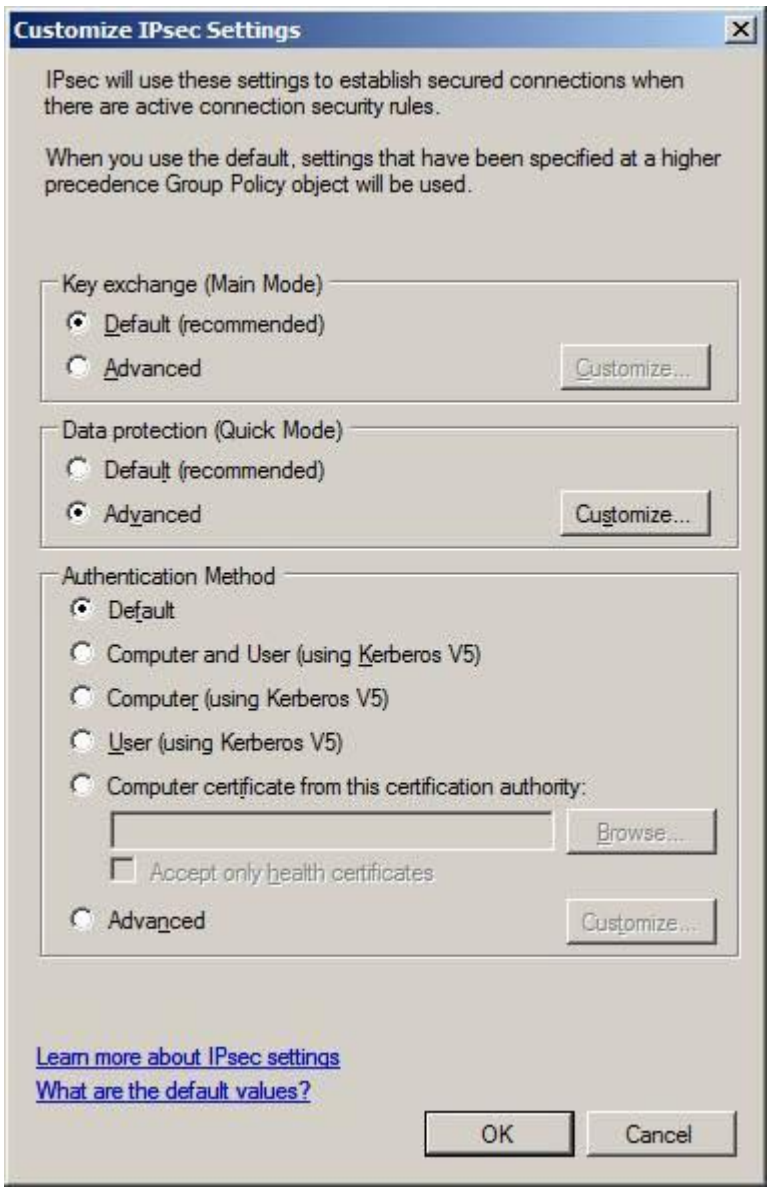
Lưu ý các thuật toán Elliptic Curve và Group 14 sẽ không làm việc với các phiên bản trước đó của Windows. Chúng chỉ làm việc với Windows Vista và Windows Server 2008.

Kích **Cancel** trong trang **Customize Advanced Key Exchange Settings**.



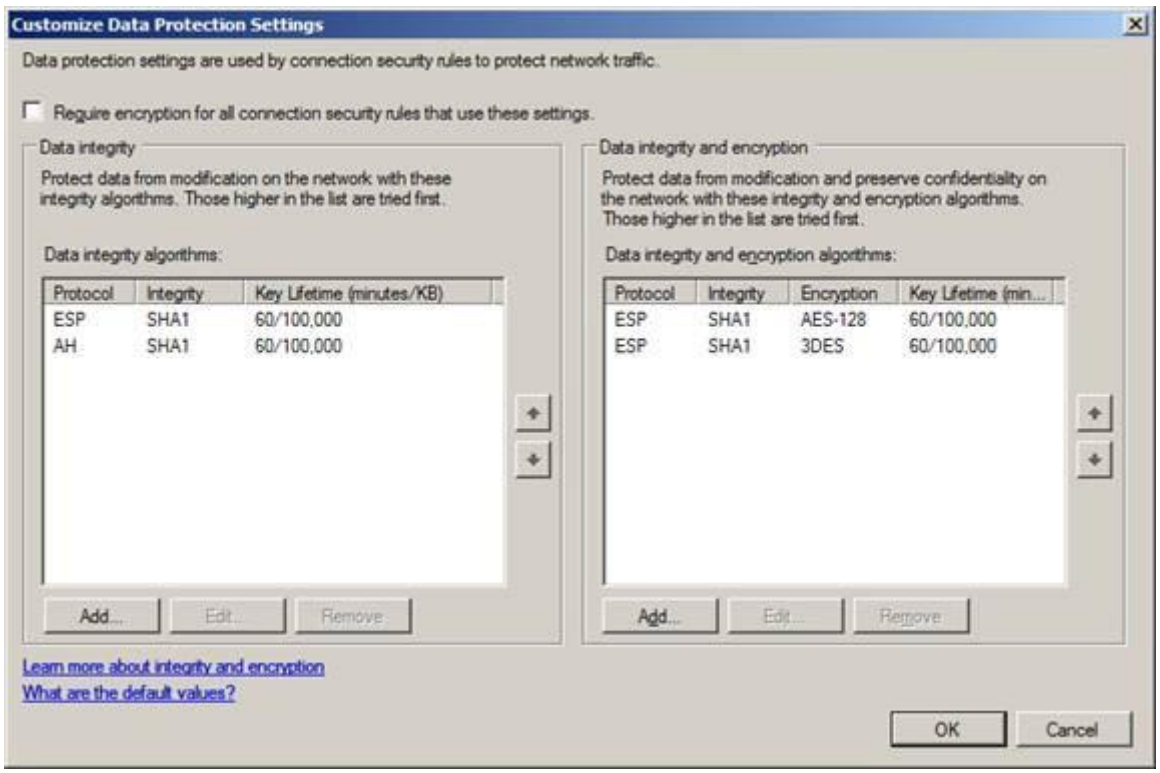
Hình 11

Chúng ta sẽ trở về với trang **Customize IPsec Settings.** Trong khung **Data Protection**, chọn tùy chọn **Advanced** và kích **Customize**.



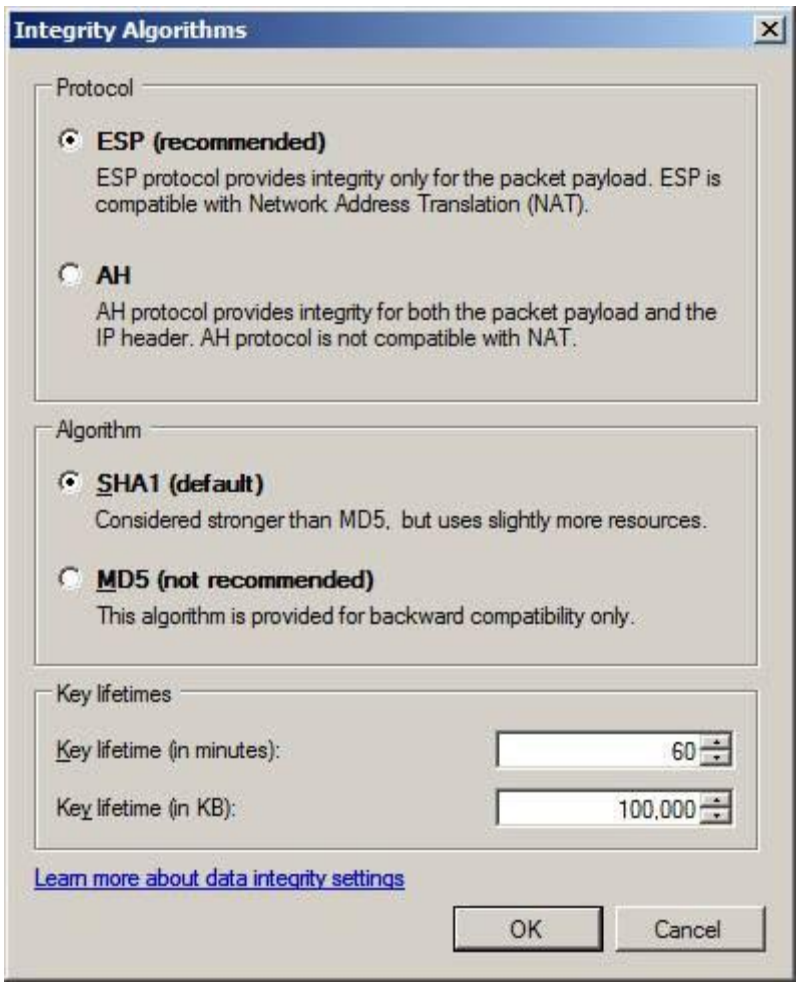
Hình 12

Trong trang **Customize Data Protection Settings**, bạn cấu hình các tùy chọn mã hóa và toàn vẹn dữ liệu. Mặc định, ESP được sử dụng cho sự toàn vẹn dữ liệu và ESP với mã hóa AES-128 được sử dụng cho việc mã hóa dữ liệu. Lưu ý rằng AES-128 không được hỗ trợ trong các phiên bản trước của Windows, chính vì vậy các thiết lập cấu hình có thể sử dụng với DES (3DES).



Hình 13

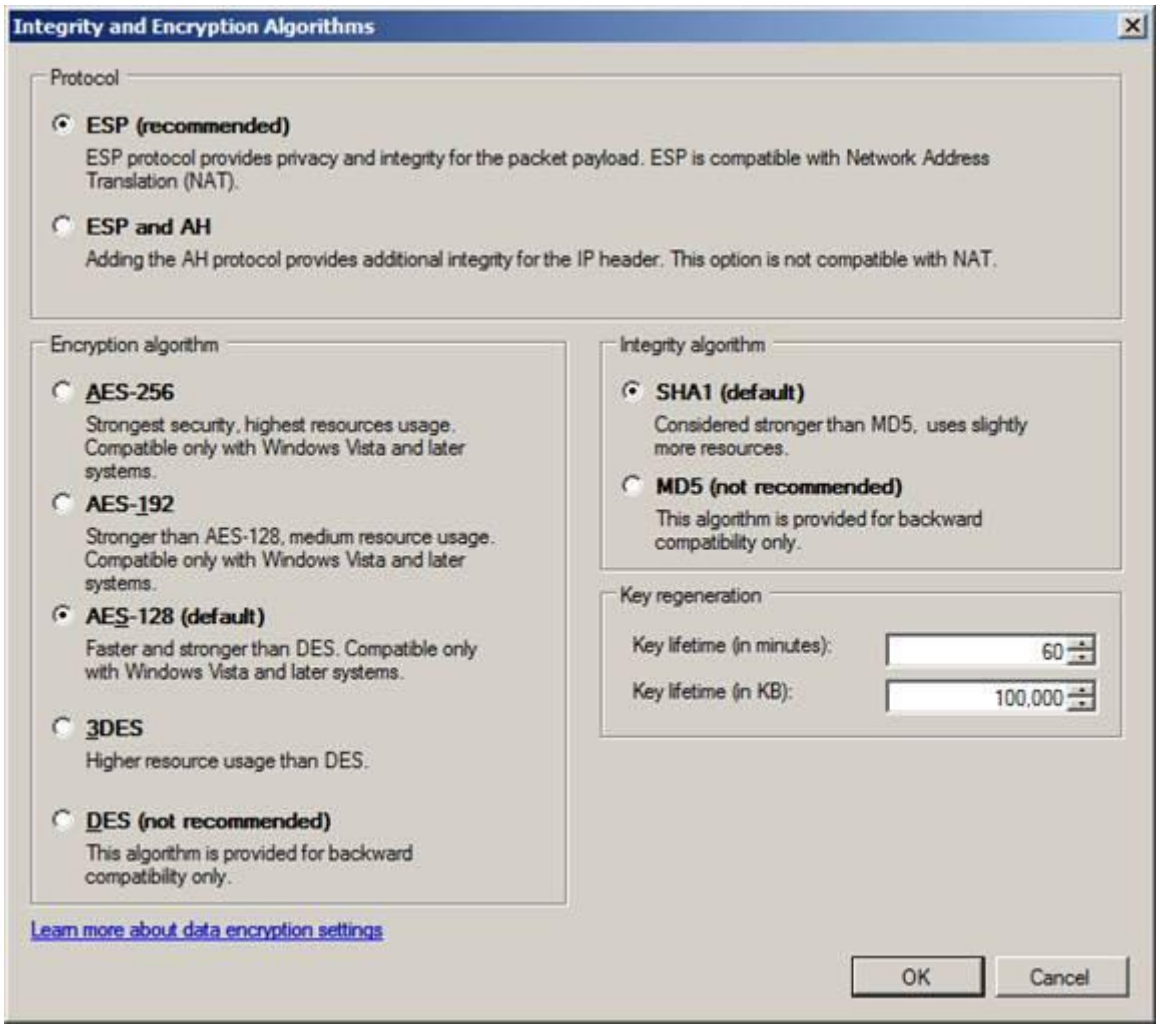
Trong bất kỳ giao thức mã hóa và toàn vẹn dữ liệu nào, bạn cũng có thể kích nút **Edit** sau khi chọn giao thức để xem xem các thiết lập giao thức. Khi kích đúp vào giao thức ESP integrity bạn sẽ thấy ESP đã được chọn và là giao thức được khuyên dùng. Lý do là nó có thể đi qua các thiết bị NAT khi IPsec NAT traversal được kích hoạt trên cả hai thiết bị. Thuật toán SHA1 hash được sử dụng mặc định vì nó an toàn hơn MD5.



Hình 14

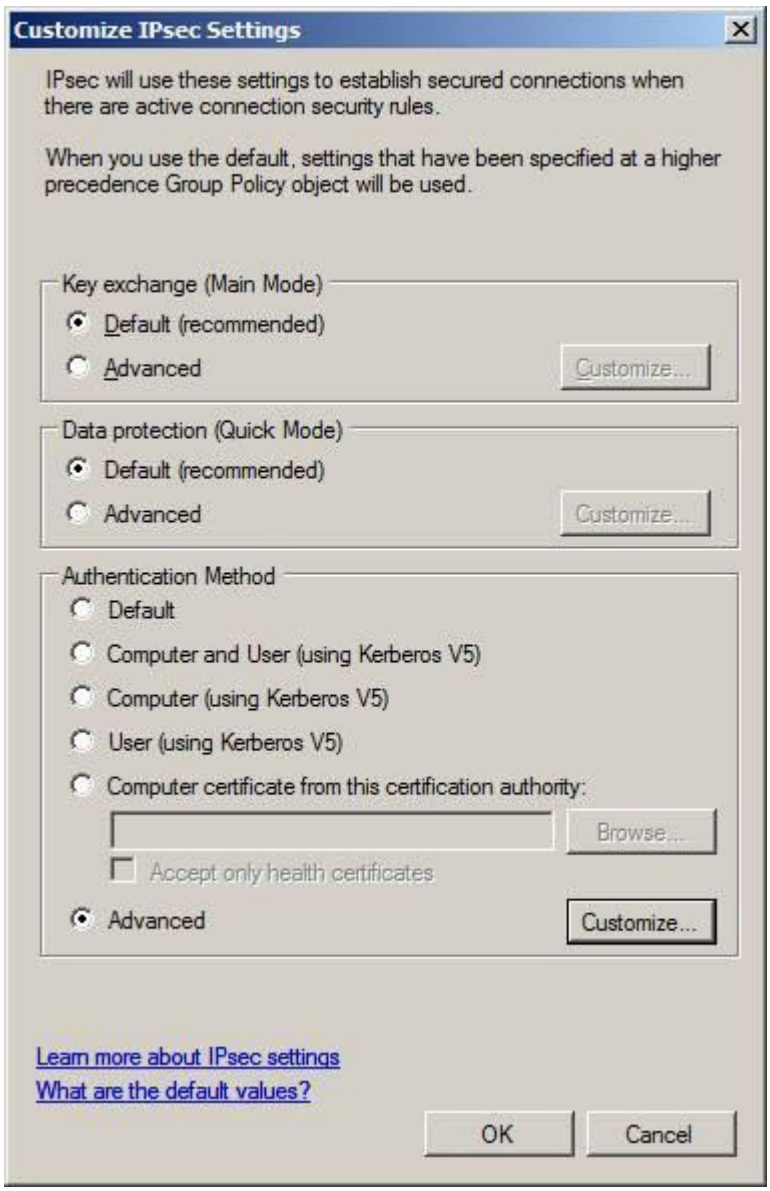
Nếu kích đúp vào mục mã hóa ESP, bạn sẽ thấy hộp thoại cấu hình cho tùy chọn đó. Ở đây bạn có thể thấy được rằng ESP chỉ được chọn một cách mặc định, vì sự bất lực của AH trong việc đi qua các thiết bị NAT. Tuy nhiên cần phải lưu ý rằng nếu bạn không có các thiết bị NAT trong đường dẫn giữa IPsec được kết nối ngang hàng thì có thể add các rule bảo mật cho kết nối bằng cách kích hoạt AH. Mặc dù vậy đây dường như là vấn đề bạn nên thực hiện một cách tùy chỉnh khi tạo các rule kết nối.

Mặc định được thiết lập là AES-128, nhưng như những gì bạn thấy trong khung, còn có cả thiết lập DES nếu cần kết nối với các máy khách và máy chủ Windows phiên bản trước.



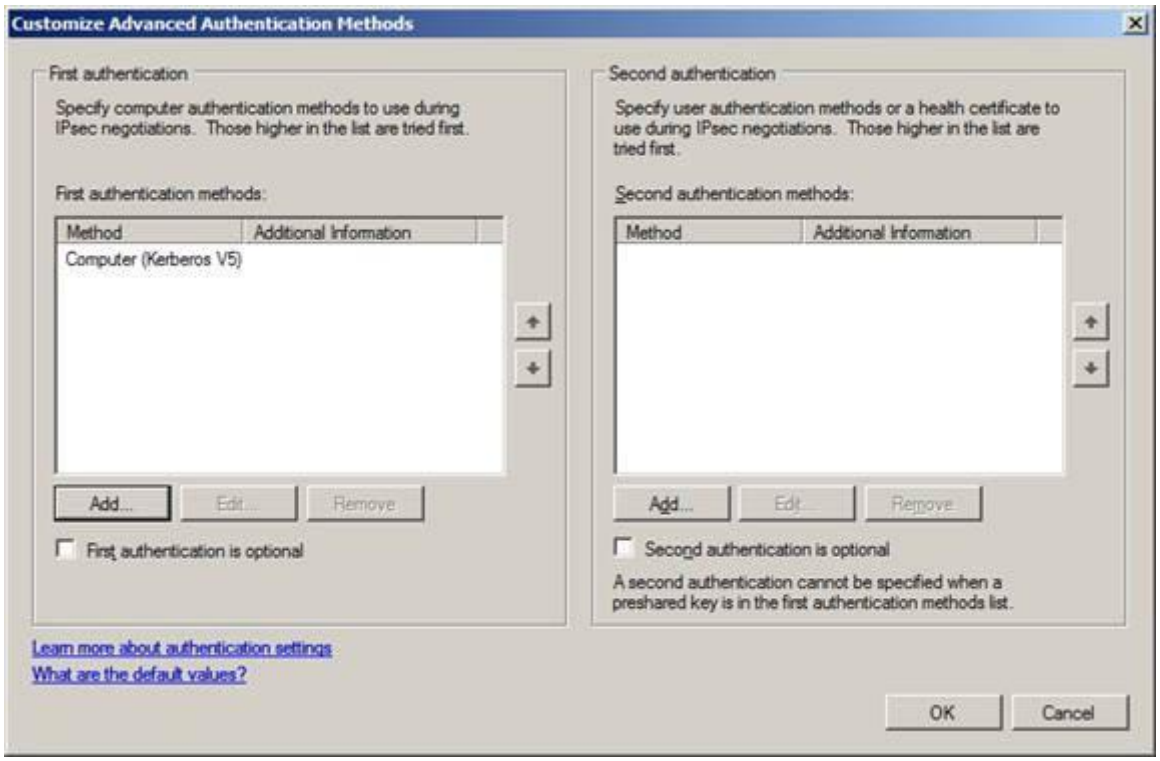
Hình 15

Tùy chọn cuối cùng bạn có thể cấu hình các mặc định là **Authentication Method**. Để xem các thông tin chi tiết về các phương pháp chứng thực có sẵn, kích tùy chọn **Advanced** và sau đó kích **Customize**.



Hình 16

Ở đây bạn sẽ thấy hộp thoại **Customize Advanced Authentication Methods**. Bạn có thể thấy các thiết lập mặc định được kích hoạt chỉ cho chứng thực **Computer (Kerberos V5)**. Nó được quy vào đây như một chứng thực đầu tiên **First Authentication.** Bạn cũng có thể chọn kích hoạt User Authentication cho một chứng thực thứ hai **Second Authentication**. Bạn sẽ thấy được điều này khi chúng ta tạo các chính sách bảo mật kết nối, bạn có thể thiết lập chứng thực cho máy tính, cho người dùng, hoặc cho cả người dùng và máy tính.


Customize Advanced Authentication Methods
First authentication
Specify computer authentication methods to use during IPsec negotiations. Those higher in the list are tried first.
First authentication methods:
Method
Additional Information
Computer (Kerberos V5)
Add...
Edit...
Remove
First authentication is optional
Second authentication
Specify user authentication methods or a health certificate to use during IPsec negotiations. Those higher in the list are tried first.
Second authentication methods:
Method
Additional Information
Add...
Edit...
Remove
Second authentication is optional
A second authentication cannot be specified when a preshared key is in the first authentication methods list.
Learn more about authentication settings
What are the default values?
OK
Cancel

Hình 17

BÀI 8: BẢO VỆ DỮ LIỆU

Mà bài: MĐ38.8

Mục tiêu:

- Trình bày các cách mã hoá dữ liệu, cách mã hoá
- Xây dựng ứng dụng đơn giản để mã hoá dữ liệu trong qua các cách mã hoá.
- Thực hiện thao tác an toàn với máy tính.

8.1 Giới thiệu

Trước đây khi công nghệ máy tính chưa phát triển, khi nói đến vấn đề an toàn bảo mật thông tin (Information Security), chúng ta thường hay nghĩ đến các biện pháp nhằm đảm bảo cho thông tin được trao đổi hay cất giữ một cách an toàn và bí mật. Chẳng hạn là các biện pháp như:

• Đóng dấu và ký niêm phong một bức thư để biết rằng lá thư có được chuyển nguyên vẹn đến người nhận hay không.

• Dùng mật mã mã hóa thông điệp để chỉ có người gửi và người nhận hiểu được thông điệp. Phương pháp này thường được sử dụng trong chính trị và quân sự (xem chương 2).

• Lưu giữ tài liệu mật trong các két sắt có khóa, tại các nơi được bảo vệ nghiêm ngặt, chỉ có những người được cấp quyền mới có thể xem tài liệu.

Với sự phát triển mạnh mẽ của công nghệ thông tin, đặt biệt là sự phát triển của mạng Internet, ngày càng có nhiều thông tin được lưu giữ trên máy vi tính và gửi đi trên mạng Internet. Và do đó xuất hiện nhu cầu về an toàn và bảo mật thông tin trên máy tính. Có thể phân loại mô hình an toàn bảo mật thông tin trên máy tính theo hai hướng chính như sau:

1) Bảo vệ thông tin trong quá trình truyền thông tin trên mạng (Network Security)

2) Bảo vệ hệ thống máy tính, và mạng máy tính, khỏi sự xâm nhập phá hoại từ bên ngoài (System Security)

Phần tiếp theo sau sẽ lần lượt trình bày các đặc điểm chính của hai mô hình trên.

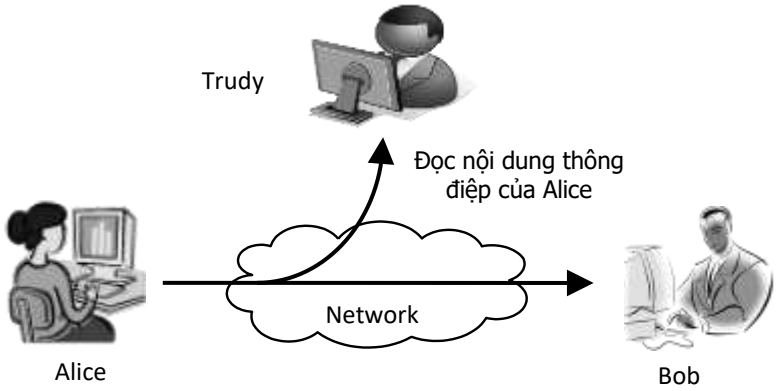
8.2 Bảo vệ thông tin trong quá trình truyền thông tin trên mạng

8.2.1 Các loại hình tấn công

Để xem xét những vấn đề bảo mật liên quan đến truyền thông trên mạng, chúng ta hãy lấy một bối cảnh sau: có ba nhân vật tên là Alice, Bob và Trudy, trong đó Alice và Bob thực hiện trao đổi thông tin với nhau, còn Trudy là kẻ xấu, đặt thiết bị can thiệp vào kênh truyền tin giữa Alice và Bob. Sau đây là các loại hành động tấn công của Trudy mà ảnh hưởng đến quá trình truyền tin giữa Alice và Bob:

1) Xem trộm thông tin (Release of Message Content)

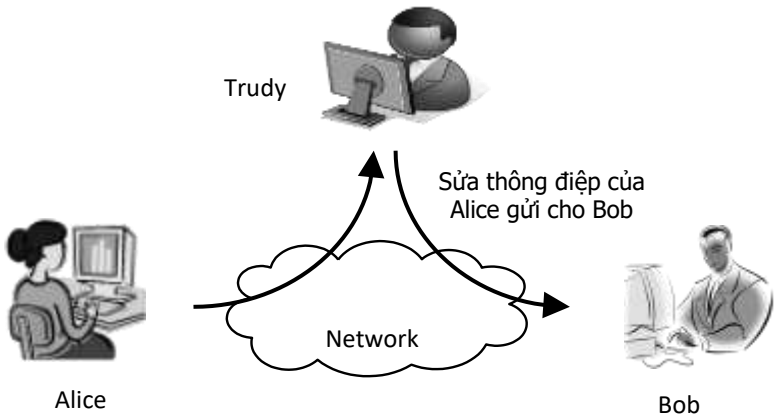
Trong trường hợp này Trudy chặn các thông điệp Alice gửi cho Bob, và xem được nội dung của thông điệp.



Hình 1-1. Xem trộm thông điệp

2) Thay đổi thông điệp (Modification of Message)

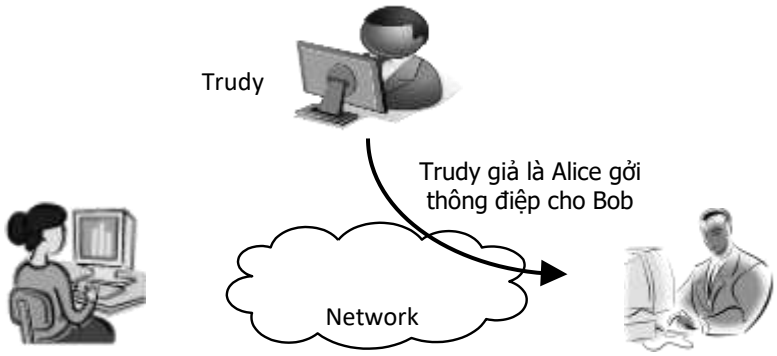
Trudy chặn các thông điệp Alice gửi cho Bob và ngăn không cho các thông điệp này đến đích. Sau đó Trudy thay đổi nội dung của thông điệp và gửi tiếp cho Bob. Bob nghĩ rằng nhận được thông điệp nguyên bản ban đầu của Alice mà không biết rằng chúng đã bị sửa đổi.



Hình 1-2. Sửa thông điệp

3) Mạo danh (Masquerade)

Trong trường hợp này Trudy giả là Alice gửi thông điệp cho Bob. Bob không biết điều này và nghĩ rằng thông điệp là của Alice.

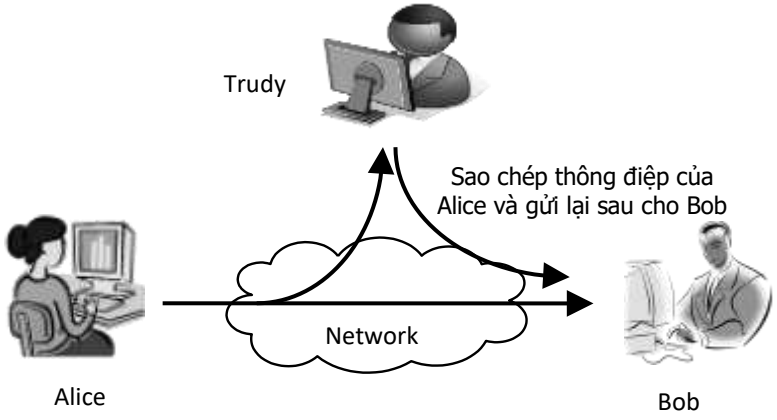


Alice Bob

Hình 1-3. Mạo danh

4) Phát lại thông điệp (Replay)

Trudy sao chép lại thông điệp Alice gửi cho Bob. Sau đó một thời gian Trudy gửi bản sao chép này cho Bob. Bob tin rằng thông điệp thứ hai vẫn là từ Alice, nội dung hai thông điệp là giống nhau. Thoạt đầu có thể nghĩ rằng việc phát lại này là vô hại, tuy nhiên trong nhiều trường hợp cũng gây ra tác hại không kém so với việc giả mạo thông điệp. Xét tình huống sau: giả sử Bob là ngân hàng còn Alice là một khách hàng. Alice gửi thông điệp đề nghị Bob chuyển cho Trudy 1000$. Alice có áp dụng các biện pháp như chữ ký điện tử với mục đích không cho Trudy mạo danh cũng như sửa thông điệp. Tuy nhiên nếu Trudy sao chép và phát lại thông điệp thì các biện pháp bảo vệ này không có ý nghĩa. Bob tin rằng Alice gửi tiếp một thông điệp mới để chuyển thêm cho Trudy 1000$ nữa.



Hình 1-4. Phát lại thông điệp

8.2.2 Yêu cầu của một hệ truyền thông tin an toàn và bảo mật

Phần trên đã trình bày các hình thức tấn công, một hệ truyền tin được gọi là an toàn và bảo mật thì phải có khả năng chống lại được các hình thức tấn công trên. Như vậy hệ truyền tin phải có các đặt tính sau:

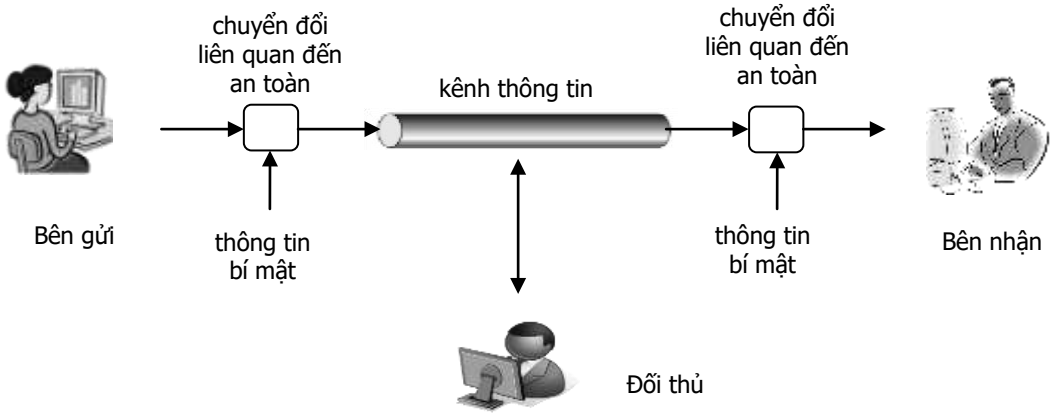
1) Tính bảo mật (***Confidentiality***): Ngăn chặn được vấn đề xem trộm thông điệp.

2) Tính chứng thực (***Authentication***): Nhằm đảm bảo cho Bob rằng thông điệp mà Bob nhận được thực sự được gửi đi từ Alice, và không bị thay đổi trong quá trình truyền tin. Như vậy tính chứng thực ngăn chặn các hình thức tấn công sửa thông điệp, mạo danh, và phát lại thông điệp.

3) Tính không từ chối (***Nonrepudiation***): xét tình huống sau:

Giả sử Bob là nhân viên môi giới chứng khoán của Alice. Alice gởi thông điệp yêu cầu Bob mua cổ phiếu của công ty Z. Ngày hôm sau, giá cổ phiếu công ty này giảm hơn 50%. Thấy bị thiệt hại, Alice nói rằng Alice không gửi thông điệp nào cả và quy trách nhiệm cho Bob. Bob phải có cơ chế để xác định rằng chính Alice là người gởi mà Alice không thể từ chối trách nhiệm được.

Khái niệm chữ ký trên giấy mà con người đang sử dụng ngày nay là một cơ chế để bảo đảm tính chứng thực và tính không từ chối. Và trong lĩnh vực máy tính, người ta cũng thiết lập một cơ chế như vậy, cơ chế này được gọi là chữ ký điện tử.



Hình 1-5. Mô hình bảo mật truyền thông tin trên mạng

8.2.3 Vai trò của mật mã trong việc bảo mật thông tin trên mạng

Mật mã hay mã hóa dữ liệu (cryptography), là một công cụ cơ bản thiết yếu của bảo mật thông tin. Mật mã đáp ứng được các nhu cầu về tính bảo mật

(confidentiality), tính chứng thực (authentication) và tính không từ chối (non-repudiation) của một hệ truyền tin.

Tài liệu này trước tiên trình bày về *mật mã cổ điển*. Những hệ mật mã cổ điển này tuy ngày nay tuy ít được sử dụng, nhưng chúng thể hiện những nguyên lý cơ bản được ứng dụng trong mật mã hiện đại. Dựa trên nền tảng đó, chúng ta sẽ tìm hiểu về *mã hóa đối xứng* và *mã hóa bất đối xứng*, chúng đóng vai trò quan trọng trong mật mã hiện đại. Bên cạnh đó chúng ta cũng sẽ tìm hiểu về *hàm Hash*, cũng là một công cụ bảo mật quan trọng mà có nhiều ứng dụng lý thú, trong đó có chữ ký điện tử.

Các chương 2, 3, 4, 5 sẽ lần lượt trình bày những nội dung liên quan đến mật mã.

8.2.4 Các giao thức (protocol) thực hiện bảo mật.

Sau khi tìm hiểu về mật mã, chúng ta sẽ tìm hiểu về cách ứng dụng chúng vào thực tế thông qua một số giao thức bảo mật phổ biến hiện nay là:

• Keberos: là giao thức dùng để chứng thực dựa trên mã hóa đối xứng. Chuẩn chứng thực X509: dùng trong mã hóa khóa công khai.

• Secure Socket Layer (SSL): là giao thức bảo mật Web, được sử dụng phổ biến trong Web và thương mại điện tử.

• PGP và S/MIME: bảo mật thư điện tử email.

Mô hình lý thuyết và nội dung các giao thức trên được trình bày trong chương 6 và chương 7.

8.3 Bảo vệ hệ thống khỏi sự xâm nhập phá hoại từ bên ngoài

Ngày nay, khi mạng Internet đã kết nối các máy tính ở khắp nơi trên thế giới lại với nhau, thì vấn đề bảo vệ máy tính khỏi sự thâm nhập phá hoại từ bên ngoài là một điều cần thiết. Thông qua mạng Internet, các hacker có thể truy cập vào các máy tính trong một tổ chức (dùng telnet chẳng hạn), lấy trộm các dữ liệu quan trọng như mật khẩu, thẻ tín dụng, tài liệu… Hoặc đơn giản chỉ là phá hoại, gây trục trặc hệ thống mà tổ chức đó phải tốn nhiều chi phí để khôi phục lại tình trạng hoạt động bình thường.

Để thực hiện việc bảo vệ này, người ta dùng khái niệm "kiểm soát truy cập" (Access Control). Khái niệm kiểm soát truy cập này có hai yếu tố sau:

• Chứng thực truy cập (Authentication): xác nhận rằng đối tượng (con người hay chương trình máy tính) được cấp phép truy cập vào hệ thống. Ví dụ: để sử dụng máy tính thì trước tiên đối tượng phải logon vào máy tính bằng username và

password. Ngoài ra, còn có các phương pháp chứng thực khác như sinh trắc học (dấu vân tay, mống mắt…) hay dùng thẻ (thẻ ATM…).

• Phân quyền (Authorization): các hành động được phép thực hiện sau khi đã truy cập vào hệ thống. Ví dụ: bạn được cấp username và password để logon vào hệ điều hành, tuy nhiên bạn chỉ được cấp quyền để đọc một file nào đó. Hoặc bạn chỉ có quyền đọc file mà không có quyền xóa file.

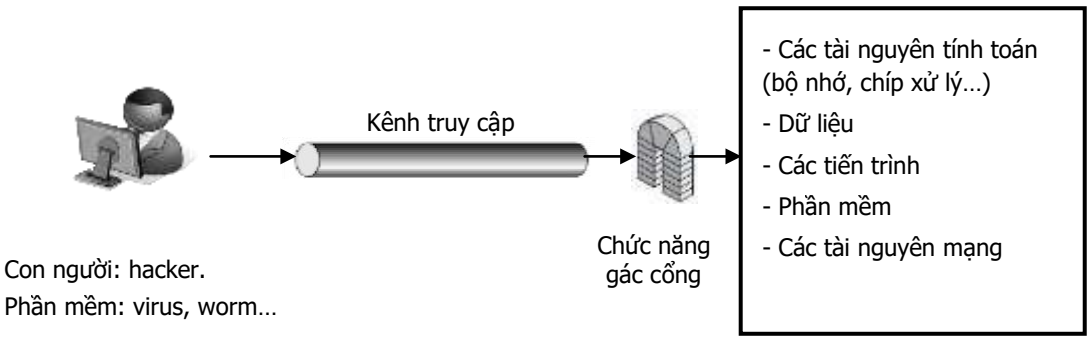
Với nguyên tắc như vậy thì một máy tính hoặc một mạng máy tính được bảo vệ khỏi sự thâm nhập của các đối tượng không được phép. Tuy nhiên thực tế chúng ta vẫn nghe nói đến các vụ tấn công phá hoại. Để thực hiện điều đó, kẻ phá hoại tìm cách phá bỏ cơ chế Authentication và Authorization bằng các cách thức sau:

• Dùng các đoạn mã phá hoại (Malware): như virus, worm, trojan, backdoor… những đoạn mã độc này phát tán lan truyền từ máy tính này qua máy tính khác dựa trên sự bất cẩn của người sử dụng, hay dựa trên các lỗi của phần mềm. Lợi dụng các quyền được cấp cho người sử dụng (chẳng hạn rất nhiều người login vào máy tính với quyền administrator), các đoạn mã này thực hiện các lệnh phá hoại hoặc dò tìm password của quản trị hệ thống để gửi cho hacker, cài đặt các cổng hậu để hacker bên ngoài xâm nhập.

• Thực hiện các hành vi xâm phạm (Intrusion): việc thiết kế các phần mềm có nhiều lỗ hổng, dẫn đến các hacker lợi dụng để thực hiện những lệnh phá hoại. Những lệnh này thường là không được phép đối với người bên ngoài, nhưng lỗ hổng của phần mềm dẫn đến được phép. Trong những trường hợp đặc biệt, lỗ hổng phần mềm cho phép thực hiện những lệnh phá hoại mà ngay cả người thiết kế chương trình không ngờ tới. Hoặc hacker có thể sử dụng các cổng hậu do các backdoor tạo ra để xâm nhập.

Để khắc phục các hành động phá hoại này, người ta dùng các chương trình có chức năng gác cổng, phòng chống. Những chương trình này dò tìm virus hoặc dò tìm các hành vi xâm phạm đển ngăn chặn chúng, không cho chúng thực hiện hoặc xâm nhập. Đó là các chương trình chống virus, chương trình firewall… Ngoài ra các nhà phát triển phần mềm cần có quy trình xây dựng và kiểm lỗi phần mềm nhằm hạn chế tối đa những lỗ hổng bảo mật có thể có.

Hệ Thống Thông Tin



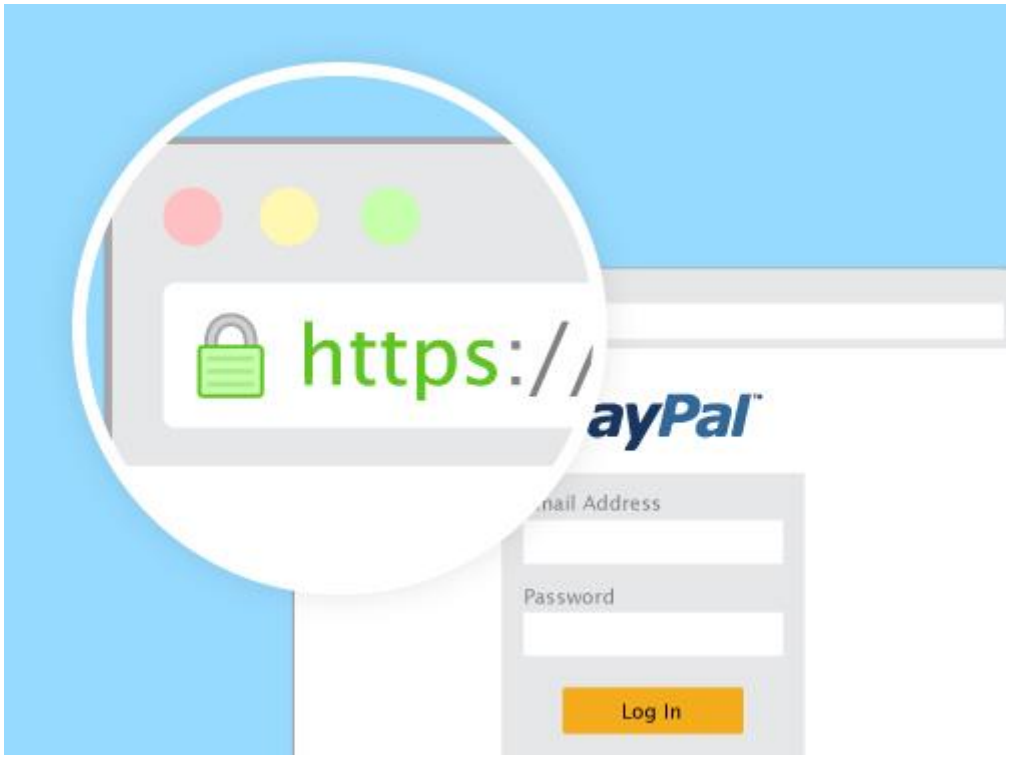
Hình 1-6.Mô hình phòng chống xâm nhập và phá hoại hệ thống

8. 4 Sơ lược về mã hoá đối xứng và bất đối xứng.

Hầu như ai có tìm hiểu về an toàn thông tin đều biết đến hai loại mã hoá phổ biến là mã hoá đối xứng (symmetric cryptography) và mã hoá bất đối xứng (asymmetric cryptography). Về cơ bản thì:

- Mã hoá đối xứng (hay còn gọi là mã hoá bí mật): Nói đơn giản là người ta dùng cùng một chìa khoá để khoá và mở thông tin cần được giữ bí mật. Và cả hai bên gửi và nhận thông tin đều phải có chìa khoá này.
- Mã hoá bất đối xứng (hay còn gọi là mã hoá công khai): Có thể hiểu là người ta dùng hai chìa khoá khác nhau để khoá và mở khoá thông tin bí mật. public key sẽ được công khai, và được gửi đi đến đối tượng cần mã hoá thông tin, còn private key được giữ bí mật, và nó đóng vai trò như chìa khoá vạn năng có thể mở được tất cả thông tin được khoá bằng public key.

8.5 Tại sao cần má hoá bất đối xứng?



Front-end Engineer | Salary from $1,200 to $1,400

JavaScriptFront-EndReactJSWebMeteorJSVueJS

Pangara

Xem thêm

Nói ngắn gọn thì hầu như các ứng dụng bạn dùng hàng ngày hiện nay như Facebook, Gmail, Amazon, PayPal v.v đều sử dụng giao thức HTTPs. Có thể hiểu là giao thức HTTPs an toàn hơn HTTP vì toàn bộ thông tin truyền đi giữa client và server được bảo vệ bởi bộ mã hoá SSL/TSL. SSL/TSL này hoạt động dựa trên cả hai loại mã hoá đối xứng và bất đối xứng. Nhờ nó mà chúng ta có thể đảm bảo bí mật khi thực hiện những giao dịch có chứa thông tin nhạy cảm trên Internet mà không bị đánh cắp thông tin trong suốt quá trình truyền nhận dữ liệu. Có thể nói, nếu không có mật mã, đặc biệt là mã hoá bất đối xứng thì **không có thương mại điện tử**.

Về cơ bản, HTTP truyền dữ liệu dưới dạng plain text, nghĩa là nếu ai đó nghe lén dữ liệu bạn truyền và nhận với server thì có thể đọc và can thiệp được nội dung (man-in-the-middle attack). Ngay cả khi dùng mã hoá đối xứng để encrypt và decrypt thông tin truyền và nhận thì cũng có lỗ hổng là hai bên phải trao đổi key mới mã hoá và giải mã được, như vậy attacker vẫn có thể tóm được key và đọc được thông tin như thường.

Điểm yếu của mã hoá đối xứng được khắc chế trong mã hoá bất đối xứng. Ý tưởng là thay vì gửi chìa khoá cho phía client, thì server sẽ gửi ổ khoá, để client khoá thông điệp bí mật trong một chiếc hộp, và chỉ có server có thể giải mã được. Cho nên các client sẽ**không** đọc được thông điệp của nhau, và chỉ có sever với private key mới mở khoá được những chiếc hộp này. (Trên thực tế thì public key vừa dùng để mã hoá vừa dùng để giải mã thông tin nhận và gửi lên server!)

8.6 Về RSA

RSA là một trong những hệ thống mã hoá bất đối xứng được sử dụng rộng rãi. Nó được đặt theo tên của 3 nhà khoa học MIT thiết kế ra nó là: Ron **Rivest**, Adi **Shamir**, và Leonard **Adleman**. Ý tưởng then chốt để đảm bảo tính an toàn của RSA là dựa trên sự khó khăn trong việc phân tích nhân tử của 2 số nguyên tố lớn. (a x b = c, tìm ngược lại a, b từ c là phân tích nhân tử).

Hệ thống mã hoá RSA bao gồm 4 bước: **key generation**, **key distribution**, **encryption**và **decryption**. Vì để đảm bảo tính bí mật, nên mỗi hệ

thống khác nhau cần tạo ra các public, và private key khác nhau. Sau qúa trình handshake và public key được gởi tới phía client thì thông tin mới chính thức được mã hoá khi server và client giao tiếp với nhau.

8.7 Mã hoá và giải mã

Tạm thời bỏ qua bước public key và private được tạo ra như thế nào. Chúng ta có công thức để mã hoá và giải mã dữ liệu như sau:

- Encryption: memodn=cmemodn=c
- Decryption: cdmodn=mcdmodn=m

Trong đó:

- m là message ban đầu
- e, n là public key
- c là dữ liệu đã được mã hoá
- d là private key thường là một số rất lớn, tích của 2 số nguyên tố, và được giữ an toàn tuyệt đối

Ví dụ cho e = 17, n = 3233, d = 2753 và cho thông điệp cần đc mã hoá là m = 42

- Mã hoá: 4217mod3233=25574217mod3233=2557

Số 2557 này khi được giải mã thì nó trở về 42 như cũ:

- Giải mã: 25572753mod3233=4225572753mod3233=42

private key d được tạo ra dựa vào 2 prime factor của n. Trong thực tế n được tạo ra bằng cách nhân hai số nguyên tố 2048 bits cho nên tính ra d thì dễ, còn từ n tính ngược lại 2 số đó để tìm private key là gần như bất khả thi với máy tính hiện giờ.

8.8 Cách tạo public và private key

Phần này thiêng về toán, hầu như chúng ta không cần quan tâm nếu như sử dụng các gói Cipher có sẵn của Java hoặc JavaScript.

1. Chọn hai số nguyên tố ngẫu nhiên phân biệt p và q (trong thực tế là càng lớn càng tốt, cỡ 2048 bits hay 617 chữ số).
    - Ví dụ: p = 61 và q = 53
2. Tính tích n=p∗q=61∗53=3233n=p∗q=61∗53=3233
3. Tính kết quả hàm số Euler (totient): Φ(n)=(p−1)(q−1)Φ(n)=(p−1)(q−1)Φ(3233)=(61−1)∗(53−1)=3120Φ(3233)=(61−1)∗(53−1)=3120
4. Chọn một số bất kỳ 1<e<31201<e<3120 và là số nguyên tố cùng nhau của 3120

- Chọn e=17e=17

5. Tính d là nghịch đảo modular của e(modΦ(n))e(modΦ(n)):

e∗dmodΦ(n)=117∗dmod3120=1e∗dmodΦ(n)=117∗dmod3120=1

Hoặc là dùng cách brute force để tính d (có thể được vì chúng ta chọn các số nhỏ), hoặc dùng thuật toán Euclid mở rộng, ta có d=2735d=2735

Cách brute force thì như sau (chạy chưa tới 15 bước là ra):

```
<span class="k">def</span> <span class="nf">compute_d</span><span class="p">(</span><span class="n">phi_n</span><span class="p">,</span> <span class="n">e</span><span class="p">):</span>
    <span class="k">for</span> <span class="n">i</span> <span class="ow">in</span> <span class="nb">range</span><span class="p">(</span><span class="mi">1</span><span class="p">,</span> <span class="mi">1000</span><span class="p">):</span>
        <span class="n">x</span> <span class="o">=</span> <span class="p">((</span><span class="n">i</span> <span class="o">*</span> <span class="n">phi_n</span><span class="p">)</span> <span class="o">+</span> <span class="mi">1</span><span class="p">)</span> <span class="o">/</span> <span class="n">e</span>
        <span class="n">y</span> <span class="o">=</span> <span class="p">(</span><span class="n">e</span> <span class="o">*</span> <span class="n">x</span><span class="p">)</span> <span class="o">%</span> <span class="n">phi_n</span>
        <span class="k">if</span> <span class="n">y</span> <span class="o">==</span> <span class="mi">1</span><span class="p">:</span>
            <span class="k">print</span> <span class="n">x</span>
            <span class="k">break</span>

<span class="n">compute_d</span><span class="p">(</span><span class="mi">3120</span><span class="p">,</span> <span class="mi">17</span><span class="p">)</span>
```

Như vậy cuối cùng chúng ta tính toán được public key: e = 17, n = 3233 vàprivate key: d = 2735

8.9 Tính an toàn của RSA

Tính an toàn của RSA chủ yếu dựa vào bộ tạo số ngẫu nhiên sinh ra 2 số nguyên tố p và q ban đầu. Việc tính ngược lại p và q từ n là chuyện hầu như không thể với hai số nguyên tố 2048 bits như đã đề cập ở trên. Nhưng việc tính ra d từ từ p và q là việc rất dễ dàng. Do đó nếu như một bên nào đó đoán ra được hoặc tìm ra lỗ hổng của bộ sinh số ngẫu nhiên đó thì coi RSA bị hoá giải. Gần đây có ý kiến cho rằng Bộ An ninh Nội địa Hoa Kỳ (NSA) đã cài một back door vào bộ tạo số ngẫu nhiên Dual Elliptic Curve để giúp NSA có thể crack RSA nhanh hơn 10,000 lần. Và điều đáng quan tâm là bộ tạo số ngẫu nhiên này được công ty RSA (được thành lập bởi 3 đồng tác giả của hệ thống RSA) cài đặt mặc định trong rất nhiều ứng dụng khác nhau.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Đoàn Văn Ban (2005), *Lập trình hướng đối tượng với Java,* Nhà xuất bản Khoa học và Kỹ Thuật.
2. Ngô Bá Hùng - Nguyễn Công Huy (2003), *Lập Trình Truyền Thông*, Khoa Công Nghệ Thông Tin - Đại Học Cần Thơ.

Tiếng Anh

3. Elliotte Rusty Harold, *Java Network Programming, 2nd Edition,* Published by O'Reilly.
4. Patrick Naughton, Michael Morrison (1996), *The Java Handbook*, Osborne/McGraw-Hill
5. Stephanie Bodoff, Dale Green, Kim Haase,Eric Jendrock,Monica Pawlan, Beth Stearns (2002), *The J2EE Tutorial*, Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, CA 94383 USA. All rights reserved.

Tài liệu tham khảo từ Internet

6. http://www.ibm.com/developerworks/vn/edu/j-introjava/index.html, truy nhập ngày 01/09/2011.
7. http://www.javavietnam.org/javavn/mvnforum/index, truy nhập ngày 15/09/2011.
8. http://www.oracle.com/technetwork/java/javase/downloads/index.html, truy nhập ngày 01/09/2011.
9. http://www.oracle.com/technetwork/java/javase/downloads/jdk6-jsp-136632.html, truy nhập ngày 01/09/2011.
10. http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html, truy nhập ngày 15/09/2011.