LỜI CẢM ƠN

Lời đầu tiên em xin được bày tỏ lòng biết ơn chân thành tới thầy giáo Ths. Phùng Anh Tuấn - giảng viên khoa CNTT trường ĐHDL Hải Phòng, người thầy đã trực tiếp giảng dậy và tận tình giúp đỡ, chỉ bảo em trong xuốt thời gian qua. Cảm ơn thầy đã luôn động viên, hướng dẫn, định hướng và truyền thụ cho em những kiến thức vô cùng quý báu để em có thể hoàn thành luận án tốt nghiệp này.

Em xin chân thành cảm ơn các thầy giáo, cô giáo trường ĐHDL Hải Phòng và đặc biệt là các thầy cô trong bộ môn tin học, những người đã không ngừng truyền đạt cho chúng em những kiến thức quý báu trong học tập tập cũng như trong cuộc sống xuốt bốn năm học vừa qua.

Và cuối cùng, hơn hết em muốn được bày tỏ lòng biết ơn sâu sắc tới gia đình, bố mẹ, anh chị em cũng như tất cả bạn bè em, những người luôn ở bên động viên, cổ vũ và giúp đỡ em trong học tập cũng như trong cuộc sống.

Dưới đây là những gì em đã tìm hiểu và nghiên cứu được trong thời gian qua. Do tính thực tế và kiến thức còn hạn chế, vì vậy em rất mong nhận được sự chỉ bảo cảu các thầy cô giáo và sự tham gia đóng góp ý kiến của các bạn để em có thể hoàn thành tốt đề tài của mình

Một lần nữa em xin chân thành cảm ơn !

Hải phòng, ngày….tháng ….năm 2009

Sinh viên

Phạm Hồng Thư

MỤC LỤC

LỜI NÓI ĐẦU

Hiện nay vấn đề toàn cầu hoá thông tin và tốc độ phát triển của khoa học công nghệ diễn ra một cách nhanh chóng, một kỷ nguyên mới được mở ra; kỷ nguyên của xã hội hóa thông tin. Công nghệ thông tin và truyền thông phát triển đã đưa thế giới chuyển sang thời đại mới thời đại của công nghệ thông tin. Việc nắm bắt và ứng dụng Công nghệ thông tin trong các lĩnh vực khoa học, kinh tế, xã hội đã đem lại cho các doanh nghiệp và các tổ chức những thành tựu và lợi ích to lớn.

Máy tính đã trở thành công cụ đắc lực và không thể thiếu của con người. Các tổ chức, công ty hay các cơ quan cần phải xây dựng hệ thống mạng máy tính cho riêng mình để trao đổi dữ liệu giữa các bộ phận. Dữ liệu được truyền đi trên mạng phải đảm bảo: dữ liệu được chuyển tới đích nhanh chóng và đúng đắn. Hầu hết dữ liệu được truyền qua mạng là truyền dưới dạng file.

Nhằm tìm hiểu thấu đáo một trong số các phương pháp truyền file qua mạng em chọn đề tài "Tìm hiểu lập trình Socket TCP trong Java và ứng dụng truyền file qua mạng".

Với lập trình socket TCP sẽ bắt buộc các máy đó phải được nối mạng với nhau. Ta đã thấy các máy muốn trao đổi dữ liệu qua mạng, chúng sẽ tạo ra ở mỗi phía một socket và trao đổi dữ liệu bằng cách đọc/ghi từ socket. Khi một chương trình tạo ra một socket, một định danh dạng số (định danh dạng số này còn được gọi là số hiệu cổng) sẽ được gán cho socket. Việc gán số hiệu cổng này cho socket có thể được thực hiện bởi chương trình hoặc hệ điều hành. Trong mỗi gói tin mà socket gửi đi có chứa hai thông tin để xác định đích đến của gói tin:

+ Một địa chỉ mạng để xác định hệ thống sẽ nhận gói tin.
+ Một số định danh cổng để nói cho hệ thống đích biết socket nào trên nó sẽ nhận dữ liệu.

Nội dung đồ án tốt này cố gắng làm rõ về lập trình socket TCP và xây dựng chương trình ứng minh họa về truyền file qua mạng bằng ngôn ngữ lập trình Java.

Đề tài gồm phần mở đầu, bốn chương và kết luận.

Chương 1: Trình bầy những kiến thức căn bản về mạng máy tính : định nghĩa, phân loại, các loại giao thức mạng, các mô hình hoạt động của mạng máy tính.

Chương 2: Giới thiệu về Java, các tính chất, các dạng chương trình ứng dụng của Java, cấu trúc của tệp chương trình Java

Chương 3: Khái niệm về socket, socket trong java và một số lớp trong lập trình java socket.

Chương 4: Xây dựng chương trình ứng dụng truyền file qua mạng: Mô hình và một số giao diện chương trình phía server và phía client.

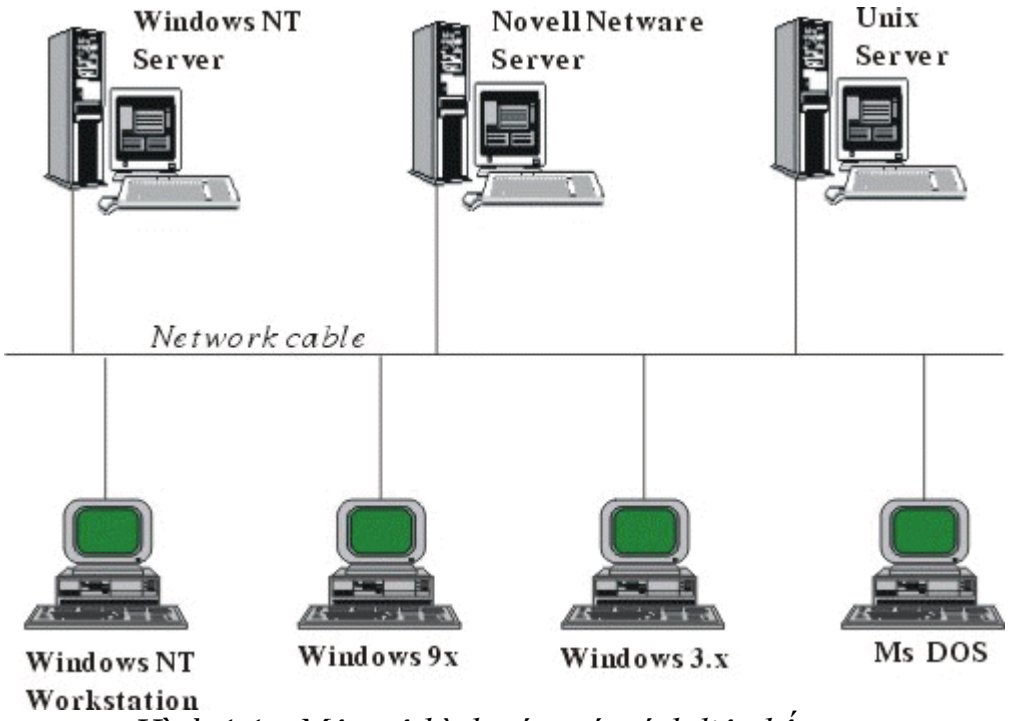
Tiếp theo là kết luận, phụ lục và tài liệu tham khảo.

CHƯƠNG 1: CĂN BẢN VỀ MẠNG MÁY TÍNH

1.1. Định nghĩa mạng máy tính

Mạng máy tính là một tập hợp các máy tính được nối với nhau bởi đường truyền theo một cấu trúc nào đó và thông qua đó các máy tính trao đổi thông tin qua lại cho nhau.

Đường truyền là hệ thống các thiết bị truyền dẫn có dây hay không dây dùng để chuyển các tín hiệu điện tử từ máy tính này đến máy tính khác. Các tín hiệu điện tử đó biểu thị các giá trị dữ liệu dưới dạng các xung nhị phân (on - off). Tất cả các tín hiệu được truyền giữa các máy tính đều thuộc một dạng sóng điện từ. Tùy theo tần số của sóng điện từ có thể dùng các đường truyền vật lý khác nhau để truyền các tín hiệu. Ở đây đường truyền được kết nối có thể là dây cáp đồng trục, cáp xoắn, cáp quang, dây điện thoại, sóng vô tuyến ... Các đường truyền dữ liệu tạo nên cấu trúc của mạng. Hai khái niệm đường truyền và cấu trúc là những đặc trưng cơ bản của mạng máy tính.



Hình 1.1- Một mô hình các máy tính liên kết trong mạng

1.2. Nhu cầu phát triển mạng máy tính

Ngày nay, khi máy tính được sử dụng một cách rộng rãi và số lượng máy tính trong một văn phòng hay cơ quan được tăng lên nhanh chóng thì việc kết nối chúng trở nên vô cùng cần thiết và sẽ mang lại nhiều hiệu quả cho người sử dụng.

Với một lượng lớn về thông tin, nhu cầu xử lý thông tin ngày càng cao, mạng máy tính đã trở nên quá quen thuộc đối với chúng ta trong mọi lĩnh vực như: khoa học, quân sự, quốc phòng, thương mại, dịch vụ, giáo dục...

Người ta thấy được việc kết nối các máy tính thành mạng cho chúng ta những khả năng mới to lớn như:

- *Sử dụng chung tài nguyên:* những tài nguyên (như thiết bị, chương trình, dữ liệu) khi được trở thành các tài nguyên chung thì mọi thành viên của mạng đều có thể tiếp cận được mà không quan tâm tới những tài nguyên đó ở đâu.

- *Tăng độ tin cậy của hệ thống:* người ta có thể dễ dàng bảo trì máy móc, lưu trữ (backup) các dữ liệu chung và khi có trục trặc trong hệ thống thì chúng có thể được khôi phục nhanh chóng. Trong trường hợp có trục trặc trên một trạm làm việc thì người ta cũng có thể sử dụng những trạm khác thay thế.

- *Nâng cao chất lượng và hiệu quả khai thác thông tin:* khi thông tin có thể được sử dụng chung thì nó mang lại cho người sử dụng khả năng tổ chức lại các công việc với những thay đổi về chất như:

+ Đáp ứng những nhu cầu của hệ thống ứng dụng kinh doanh hiện đại.

+ Cung cấp sự thống nhất giữa các dữ liệu.

+ Tăng cường năng lực xử lý nhờ kết hợp các bộ phận phân tán.

+ Tăng cường truy nhập tới các dịch vụ mạng khác nhau đang được cung cấp trên thế giới.

Với nhu cầu đòi hỏi ngày càng cao của xã hội nên vấn đề kỹ thuật trong mạng là mối quan tâm hàng đầu của các nhà tin học. Ví dụ như làm thế nào để truy xuất thông tin một cách nhanh chóng và tối ưu, trong khi việc xử lý thông tin trên mạng quá nhiều, đôi khi có thể làm tắc nghẽn và gây ra mất thông tin một cách đáng tiếc. Hiện nay, việc làm sao có được một hệ thống mạng chạy thật tốt, thật an toàn với lợi ích kinh tế cao đang rất được quan tâm.

1.3. Phân loại mạng máy tính

Do hiện nay mạng máy tính được phát triển khắp nơi với những ứng dụng ngày càng đa dạng cho nên việc phân loại mạng máy tính là một việc rất phức tạp.

Dựa theo phạm vi phân bố của mạng ta có thể phân ra các loại mạng như sau:

- **Mạng cục bộ LAN** (Local Area Network): Mạng LAN là một nhóm máy tính và các thiết bị truyền thông mạng được nối kết với nhau trong một khu vực nhỏ như một tòa nhà cao ốc, khuôn viên trường đại học, khu giải trí…

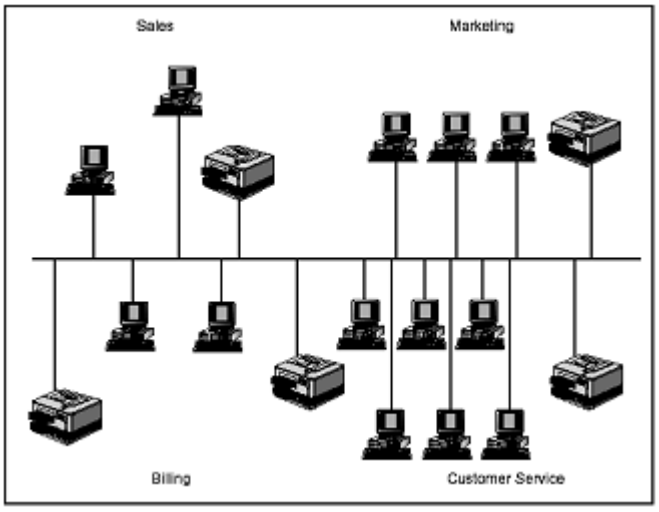
Các mạng LAN thường có đặc điểm sau:

+ Băng thông lớn, có khả năng chạy các ứng dụng trực tuyến như xem phim, hội thảo qua mạng.

+ Kích thước mạng bị giới hạn bởi các thiết bị

+ Chi phí các thiết bị mạng LAN tương đối rẻ

+ Quản trị đơn giản



Hình 1.2- Mô hình mạng cục bộ LAN

- **Mạng đô thị MAN** (Metropolitan Area Network): Mạng MAN gần giống như mạng LAN nhưng giới hạn của nó là một thành phố hay một quốc gia. Mạng MAN nối kết các mạng LAN lạ với nhau thông qua các phương tiện truyền dẫn khác nhau (cáp quang, cáp đồng, sóng….) và các phương thức truyền thông khác nhau.

Đặc điểm của mạng MAN:

+ Băng thông mức trung bình, đủ để phục vụ các ứng dụng cấp thành phố hay quốc gia như chính phủ điện tử, thương mại điện tử, các ứng dụng của các ngân hàng…

+ Do MAN nối kết nhiều LAN với nhau nên độ phức tạp cũng tăng đồng thời công tác quản trị sẽ khó khăn hơn.

+ Chi phí các thiết bị mạng MAN tương đối đắt tiền.

- **Mạng diện rộng WAN** (Wide Area Network): Mạng WAN bao phủ vùng địa lý rộng lớn có thể là một quốc gia, một lục địa hay toàn cầu. Mạng WAN thường là mạng của các công ty đa quốc gia hay toàn cầu, điển hình là mạng internet. Do phạm vi rộng lớn của mạng WAN nên thông thường mạng WAN là tập hợp các mạng LAN, WAN nối lại với nhau bằng các phương tiện như: vệ tinh (satellites), sóng biva (microwave), cáp quang, cáp điện thoại.

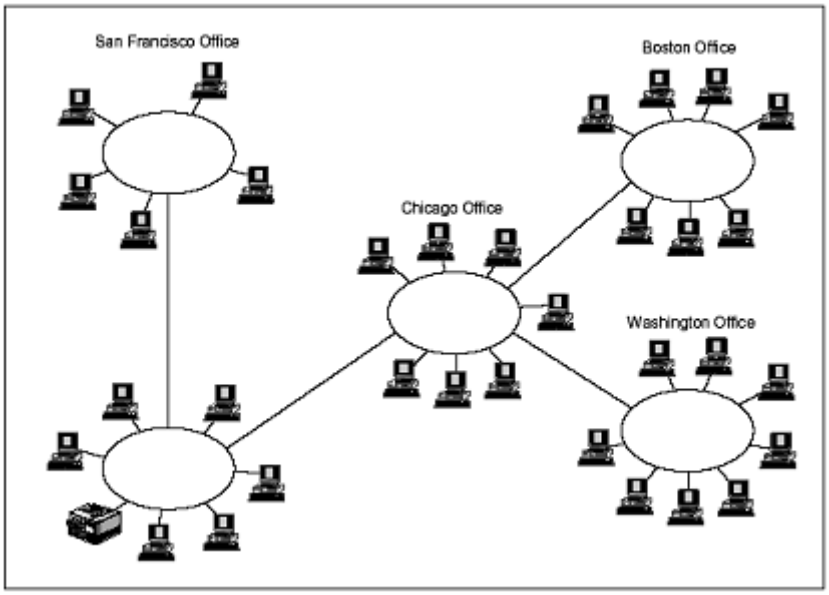
Đặc điểm của mạng WAN:

+ Băng thông thấp, dễ mất kết nối, thường chỉ phù hợp với các ứng dụng offine như e-mail, web, ftp…

+ Phạm vi hoạt động rộng lớn không giới hạn

+ Do kết nối của nhiều LAN, WAN lại với nhau nên mạng rất phức tạp và có tính toàn cầu nên thường là có tổ chức quốc tế đứng ra quản trị

+ Chi phí cho các thiết bị và các công nghệ mạng WAN rất đắt tiền



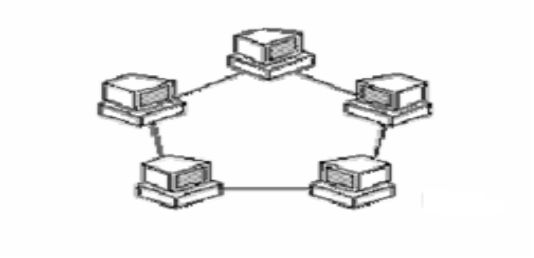
Hình 1.3- Mô hình mạng diện rộng(WAN)

- **Mạng Internet**: Là trường hợp đặc biệt của mạng WAN, nó cung cấp các dịch vụ toàn cầu như mail, web, chat, ftp và phục vụ miễn phí cho mọi người.

1.4. Một số topo mạng thông dụng

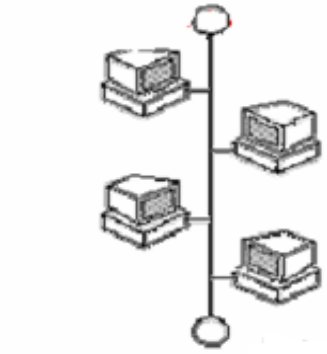
Theo định nghĩa về mạng máy tính, các máy tính được nối với nhau bởi các đường truyền vật lý theo một kiến trúc nào đó, các kiến trúc đó gọi là Topology. Thông thường mạng có ba loại kiến trúc đó là: mạng hình sao (Star Topology), mạng dạng tuyến (Bus Topology), mạng dạng vòng(Ring Topology).

- Ring Topology: Mạng được bố trí vòng tròn, đường dây cáp được thiết kế làm thành một vòng khép kín, tín hiệu chạy theo một chiều nào đó. Các nút truyền tín hiệu cho nhau tại một thời điểm được một nút mà thôi. Mạng dạng vòng có thuận lợi là có thể nới rộng ra xa nhưng đường dây phải khép kín, nếu bị ngắt ở một nơi nào đó thì toàn bộ hệ thống cũng bị ngưng.

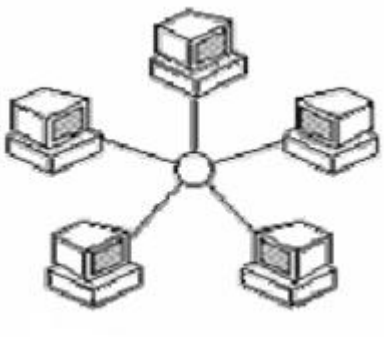


Hình 1.4- Ring Topology

- Bus Topology: Ở dạng Bus tất cả các nút được phân chia một đường truyền chính (bus). Đường truyền này được giới hạn hai đầu bởi một loại đầu nối đặc biệt gọi là Terminator. Khi một nút truyền dữ liệu, tín hiệu được quảng bá trên hai chiều của bus, mọi nút còn lại đều được nhận tín hiệu trực tiếp. Loại mạng này dùng dây cáp ít, dễ lắp đặt. Tuy vậy cũng có những bất lợi đó là sẽ có sự ùn tắc giao thông khi di chuyển với lưu lượng lớn và khi có sự hỏng hóc ở đoạn nào đó thì rất khó phát hiện, nếu một nút ngừng hoạt động sẽ ảnh hưởng tới toàn bộ hệ thống.



Hình 1.5- Bus Topology



Hình 1.6- Star Topology

- Star Topology: Mạng hình sao bao gồm một bộ tập trung và các nút thông tin. Các nút thông tin có thể là các trạm cuối, các máy tính hay các thiết bị khác của mạng. Mạng hoạt động theo nguyên lý nối song song nên nếu có một nút bị hỏng mạng vẫn hoạt động bình thường. Mạng có thể mở rộng hoặc thu hẹp tùy theo yêu cầu của người sử dụng, tuy nhiên mở rộng phụ thuộc và khả năng của trung tâm.

1.5. Giao thức mạng

Giao thức mạng là một tập các quy tắc, quy ước để trao đổi thông tin giữa hai hệ thống máy tính hoặc hai thiết bị máy tính với nhau. Nói một cách hình thức thì giao thức mạng là một ngôn ngữ được các máy tính trong mạng sử dụng để trao đổi dữ liệu với nhau. Có nhiều loại giao thức được sử dụng trong mạng máy tính như: Apple Talk, DLC, NetBEUI,… nhưng hiện nay giao thức được sử dụng phổ biến nhất trong mạng máy tính là giao thức TCP/IP.

1.5.1 Giao thức TCP

Định nghĩa: TCP(Transmission Control Protocol) là giao thức hướng kết nối, nó cung cấp một đường truyền dữ liệu tin cậy giữa hai máy tính. Tính tin cậy thể hiện ở việc nó đảm bảo dữ liệu được gửi sẽ đến được đích và theo đúng thứ tự như khi nó được gửi.

Khi hai ứng dụng muốn giao tiếp với nhau một cách tin cậy, chúng sẽ tạo ra đường kết nối giữa chúng và gửi dữ liệu thông qua đường này. Cách trao đổi dữ liệu này tương tự như cách chúng ta gọi điện thoại. Hãy lấy ví dụ khi bạn nhấc điện thoại lên và quay số của người họ hàng này, lúc đó một kết nối sẽ được tạo ra giữa điện thoại của bạn và người họ hàng, sau đó bạn gửi và nhận dữ liệu (dưới dạng âm thanh) bằng cách nói và nghe qua điện thoại của bạn. Toàn bộ việc thực hiện kết nối và truyền dữ liệu giữa hai máy điện thoại được thực hiện bởi công ty điện thoại thông qua các trạm và đường dây điện thoại, nhiệm vụ duy nhất của bạn là quay số để cung cấp cho nhà cung cấp dịch vụ điện thoại biết số điện thoại bạn muốn liên lạc. Giống như vậy, trong việc truyền dữ liệu qua mạng thì TCP đóng vai trò như nhà cung cấp dịch vụ điện thoại ở ví dụ trên, nó làm nhiệm vụ tạo kết nối và truyền dữ liệu giữa hai điểm giao tiếp để đảm bảo dữ liệu không bị mất và tới đích theo đúng trật tự như khi chúng ta gửi.

Tính tin cậy của đường truyền được thể hiện ở hai điểm sau:

- Mọi gói tin cần gửi sẽ đến được đích. Để làm được điều này thì mỗi lần phía gửi gửi xong một gói tin nó sẽ chờ nhận một xác nhận từ bên nhận rằng đã nhận được gói tin. Nếu sau một khoảng thời gian mà phía gửi không nhận được thông tin xác nhận phản hồi thì nó sẽ phát lại gói tin. Việc phát lại sẽ được tiến hành cho đến khi việc truyền tin thành công, tuy nhiên sau một số lần phát lại max nào đó mà vẫn chưa thành công thì phía gửi có thể suy ra là không thể truyền tin được và sẽ dừng việc phát tin.
- Các gói tin sẽ được trình ứng dụng nhận được theo đúng thứ tự như chúng được gửi. Bởi các gói tin có thể được dẫn đi trên mạng theo nhiều đường khác nhau trước khi tới đích nên thứ tự khi tới đích của chúng có thể không giống như khi chúng được phát. Do đó để đảm bảo có thể sắp xếp lại gói tin ở phía nhận theo đúng thứ tự như khi chúng được gửi, giao thức TCP sẽ gắn vào mỗi gói tin một thông tin cho biết thứ tự của chúng trong cả khối tin chung được phát nhờ vậy bên nhận có thể sắp xếp lại các gói tin theo đúng thứ tự của chúng.

Như vậy có thể thấy TCP cung cấp cho chúng ta một kênh truyền thông điểm- điểm phục vụ cho các ứng dụng đòi hỏi giao tiếp tin cậy như HTTP (HyperText Transfer Protocol), FTP (File Tranfer Protocol), Telnet…. Các ứng dụng này đòi hỏi một kênh giao tiếp tin cậy bởi thứ tự của dữ liệu được gửi và nhận là yếu tố quyết định đến sự thành công hay thất bại của chúng. Hãy lấy ví dụ khi HTTP được sử dụng để đọc thông tin từ một địa chỉ URL, dữ liệu phải được nhận theo đúng thứ tự mà chúng được gửi nếu không thứ mà bạn nhận được có thể là một trang HTML với nội dung lộn xộn hoặc một file zip bị lỗi và không giải nén….

1.5.2 Giao thức UDP

Định nghĩa: UDP (User Datagram Protocol) là giao thức không hướng kết nối, nó gửi các gói dữ liệu độc lập gọi là datagram từ máy tính này đến máy tính khác mà không đảm bảo việc dữ liệu sẽ tới đích.

Ở phần trước chúng ta đã thấy trong giao thức TCP khi hai chương trình muốn giao tiếp với nhau qua mạng chúng tạo ra một kết nối liên kết hai ứng dụng và trao đổi dữ liệu qua kết nối đó. Trái lại ở giao thức UDP khi hai ứng

dụng muốn giao tiếp với nhau chúng không tạo ra kết nối mà chỉ đơn thuần gửi các gói tin một cách độc lập từ máy này tới máy khác.Các gói tin như vậy gọi là các datagram. Việc gửi các gói tin như vậy tương tự như việc chúng ta gửi thư qua đường bưu điện. Các bức thư bạn gửi độc lập với nhau, thứ tự các thư là không quan trọng và không có gì đảm bảo là thư sẽ đến được đích. Trong truyền thông bằng UDP thì các datagram giống như các lá thư, chúng chứa thông tin cần gửi đi cùng thông tin về địa chỉ đích mà chúng phải đến, tuy nhiên chúng khác với các lá thư ở một điểm là nếu như trong việc gửi thư, nếu lá thư của bạn không đến được đích thì nó sẽ được gửi trả lại nơi gửi nêu trên lá thư đó bạn có đề địa chỉ gửi còn UDP sẽ không thông báo gì cho phía gửi về việc lá thư đó có tới được đích hay không.

Vậy nếu UDP là một giao thức không đảm bảo giao tiếp tin cậy thì tại sao người ta lại dùng chúng. Điều đó là bởi nếu như giao thức TCP đảm bảo một kết nối tin cậy giữa các ứng dụng thì chúng cũng đòi hỏi nhiều thời gian để truyền tin do chúng phải kiểm tra các header các gói tin để đảm bảo thứ tự các gói tin cũng như để phát lại các gói tin không đến được đích do đó trong một số trường hợp thì điều này không cần thiết. Dưới đây là một số trường hợp trong đó giao thức không hướng kết nối là thích hợp hơn so với giao thức hướng kết nối:

Khi chỉ một gói dữ liệu cần truyền đi và việc đó đến được đích hay không là không quan trọng, sử dụng giao thức UDP sẽ loại bỏ được các thủ tục tạo và hủy kết nối. So sánh một chút chúng ta sẽ thấy giao thức hướng kết nối TCP phải dùng đến bẩy gói tin để gửi một gói tin do nó cần phát và nhận các gói tin yêu cầu và chấp nhận kết nối cũng như các gói tin yêu cầu và xác nhận việc hủy kết nối, trong khi đó giao thức không kết nối UDP chỉ sử dụng duy nhất một gói tin chính là gói tin chứa dữ liệu cần chuyển đi.

Chúng ta lấy ví dụ về một server đồng hồ, nhiệm vụ của nó là gửi thời gian hiện tại của nó cho các ứng dụng trên client khi có yêu cầu. Nếu gói tin chứa thời gian bị thất lạc trên đường truyền và không tới được đích thì client cũng sẽ không đòi hỏi server phải gửi lại gói tin đó bởi khi gói tin đó được phát lại lần hai và tới được client thì thông tin thời gian chứa trong nó đã không còn đúng nữa. Nếu client tạo ra hai yêu cầu và nhận được các gói tin trả lời không theo đúng thứ tự mà server đã gửi thì client cũng không gặp phải vấn đề gì bởi nó

hoàn toàn có thể suy ra được rằng các gói đã không được chuyển đến đúng thứ tự bằng cách tính thời gian được chứa trong các gói. Trong trường hợp này tính tin cậy của TCP là không cần thiết bởi nó làm giảm hiệu suất và có thể cản trở hoạt động của server.

Trường hợp thứ hai chúng ta xem xét việc sử dụng giao thức UDP là các ứng dụng đòi hỏi chặt chẽ về thời gian như các ứng dụng nghe audio thời gian thực. Trong trường hợp này việc hướng tới một kênh giao tiếp tin cậy không phải là ưu điểm mà ngược lại đó là một nhược điểm bởi nếu việc phải chờ cho khi một gói tin bị mất được nhận có thể gây ra những tác động dễ nhận thấy hoặc khiến chương trình phải tạm ngừng. Với các ứng dụng này giao thức không hướng kết nối đã được phát triển và chúng làm việc tốt hơn hẳn. Chúng ta có thể tham khảo ứng dụng RealAudio, trong đó người ta sử dụng một giao thức không hướng kết nối để truyền các dữ liệu âm thanh qua mạng.

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

Chế độ có hướng kết nối(TCP)	**Chế độ không hướng kết nối(UDP)**
Tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp	Không tồn tại kênh giao tiếp ảo giữa hai bên giao tiếp
Dữ liệu được gửi 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 được gửi đi theo chế độ không bảo đảm: Không kiểm tra lỗi, không phát hiện không truyền lại gói tin bị lỗi hay mất, không bảo đảm thứ tự đến của các gói tin…
Dữ liệu chính xác, tốc độ truyền chậm	Dữ liệu không chính xác, tốc độ truyền nhanh.
	Thích hợp cho các ứng dụng cần tốc độ, không cần chính xác cao: Truyền âm thanh, hình ảnh

1.6. Các mô hình hoạt động của mạng máy tính

Mô hình hoạt động của mạng máy tính có hai loại:

- Mô hình hoạt động peer to peer

- Mô hình hoạt động clients/server

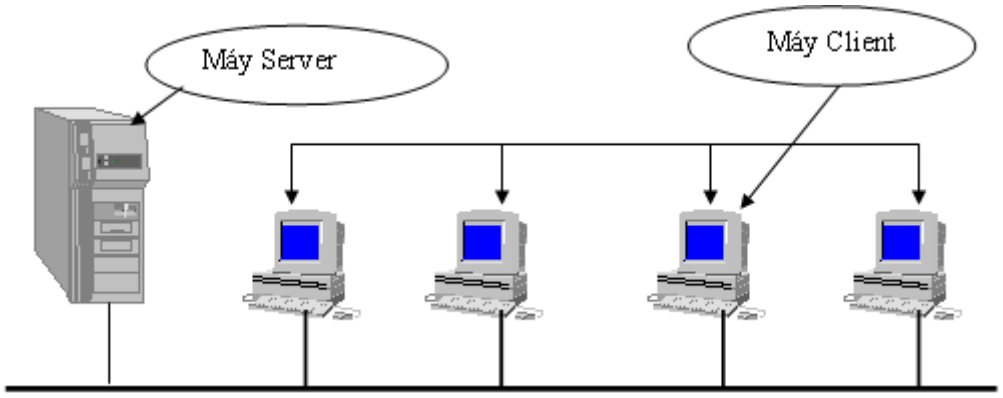
1.6.1. Mô hình hoạt động peer to peer

Không tồn tại bất kỳ máy chuyên dụng hoặc cấu trúc phân cấp giữa các máy tính. Mọi máy tính đều bình đẳng và có vai trò như nhau. Thông thường mỗi máy tính hoạt động với cả vai trò máy khách và máy phục vụ. Vì vậy không có máy nào được chỉ định quản lý toàn mạng. Người dùng ở từng máy tự quyết định dữ liệu nào trên máy của mình sẽ được chia sẻ để dùng chung trên mạng.

Hình 1.7- Mô hình peer to peer

1.6.2. Mô hình hoạt động clients/ server

Trong mạng hoạt động theo mô hình Clients/Server có một hoặc nhiều máy có nhiệm vụ cung cấp một số dịch vụ cho các máy khác ở trong mạng. Các máy này được gọi là server còn các máy tính được phục vụ gọi là máy clients.



Hình 1.8- Mô hình mạng Clients/Server

Đây là mô hình tổng quát, trên thực tế server có thể được nối với nhiều server khác để tăng hiệu quả làm việc. Khi nhận được yêu cầu từ client, server có thể xử lý yêu cầu đó hoặc gửi tiếp yêu cầu vừa nhận được cho một server khác.

Máy server sẽ thi hành các nhiệm vụ do máy client yêu cầu. Có rất nhiều dịch vụ trên mạng hoạt động theo nguyên lý nhận các yêu cầu từ client sau đó xử lý và trả lại các kết quả cho client yêu cầu.

CHƯƠNG 2: CĂN BẢN VỀ NGÔN NGỮ LẬP TRÌNH JAVA

2.1. Giới thiệu Java

Java là một ngôn ngữ lập trình được Sun Microsystems giới thiệu vào tháng 6 năm 1995. Từ đó, nó đã trở thành một công cụ lập trình của các lập trình viên chuyên nghiệp. Java được xây dựng trên nền tảng của C và C++. Do vậy nó sử dụng các cú pháp của C và các đặc trưng hướng đối tượng của C++.

Vào năm 1991, một nhóm các kỹ sư của Sun Microsystems có ý định thiết kế một ngôn ngữ lập trình để điều khiển các thiết bị điện tử như Tivi, máy giặt, lò nướng, … Mặc dù C và C++ có khả năng làm việc này nhưng trình biên dịch lại phụ thuộc vào từng loại CPU.

Trình biên dịch thường phải tốn nhiều thời gian để xây dựng nên rất đắt. Vì vậy để mỗi loại CPU có một trình biên dịch riêng là rất tốn kém. Do đó nhu cầu thực tế đòi hỏi một ngôn ngữ chạy nhanh, gọn, hiệu quả và độc lập thiết bị tức là có thể chạy trên nhiều loại CPU khác nhau, dưới các môi trường khác nhau. “Oak” đã ra đời và vào năm 1995 được đổi tên thành Java. Mặc dù mục tiêu ban đầu không phải cho Internet nhưng do đặc trưng không phụ thuộc thiết bị nên Java đã trở thành ngôn ngữ lập trình cho Internet.

2.2. Một số tính chất của ngôn ngữ Java

- Đơn giản
- Hướng đối tượng
- Độc lập phần cứng và hệ điều hành
- Mạnh
- Bảo mật
- Phân tán
- Đa luồng
- Động

2.2.1. Đơn giản

Những người thiết kế mong muốn phát triển một ngôn ngữ dễ học và quen thuộc với đa số người lập trình. Do vậy Java loại bỏ các đặc trưng phức tạp của C và C++ như:

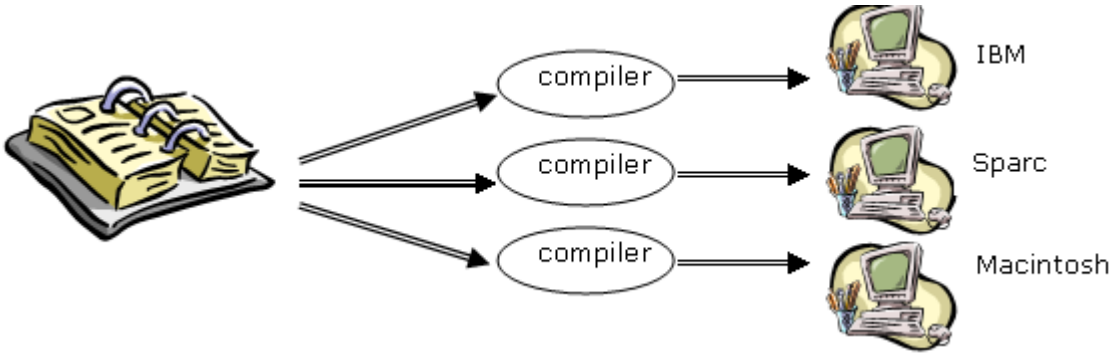
- Loại bỏ thao tác con trỏ, thao tác định nghĩa chồng toán tử
- Không cho phép đa kế thừa mà sử dụng các giao diện
- Không sử dụng lệnh "`goto`" cũng như file header (.h)
- Loại bỏ cấu trúc "`struct`" và "`union`"

2.2.2. Hướng đối tượng

Java là ngôn ngữ lập trình thuần hướng đối tượng. Mọi chương trình viết trên Java đều phải được xây dựng trên các đối tượng. Nếu trong C/ C++ ta có thể tạo ra các hàm (không gắn với đối tượng nào) thì trong Java ta chỉ có thể tạo ra các hàm (phương thức) gắn liền với một đối tượng. Trong Java không cho phép các đối tượng có tính năng đa kế thừa mà thay bằng các giao tiếp

2.2.3. Độc lập phần cứng và hệ điều hành

Đối với các ngôn ngữ lập trình truyền thống như C/ C++, phương pháp biên dịch được thực hiện như sau :

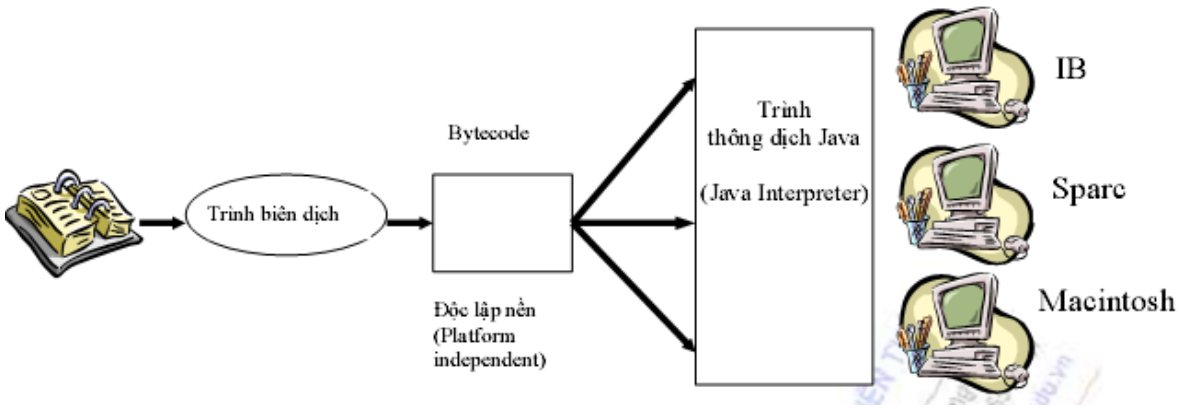


Hình 2.1- Cách biên dịch chương trình truyền thống

Với mỗi nền phần cứng khác nhau, có một trình biên dịch khác nhau để biên dịch mã nguồn chương trình cho phù hợp với nền phần cứng ấy. Do vậy, khi chạy trên một nền phần cứng khác bắt buộc phải biên dịch lại mã nguồn

Đối với các chương trình viết bằng Java, trình biên dịch Java sẽ biên dịch mã nguồn thành dạng bytecode. Sau đó, khi chạy chương trình trên các nền phần cứng khác nhau, máy ảo Java dùng trình thông dịch Java để chuyển mã bytecode

thành dạng chạy được trên các nền phần cứng tương ứng. Do vậy, khi thay đổi nền phần cứng, không phải biên dịch lại mã nguồn Java



Hình 2.2- Dịch chương trình Java

2.2.4. Mạnh mẽ

Java là ngôn ngữ yêu cầu chặt chẽ về kiểu dữ liệu.

- Kiểu dữ liệu phải khai báo tường minh.
- Java không sử dụng con trỏ và các phép toán con trỏ.
- Java kiểm tra tất cả các truy nhập đến mảng, chuỗi khi thực thi để đảm bảo rằng các truy nhập đó không ra ngoài giới hạn kích thước
- Trong các môi trường lập trình truyền thống, lập trình viên phải tự mình cấp phát bộ nhớ. Trước khi chương trình kết thúc thì phải tự giải phóng bộ nhớ đã cấp. Vấn đề nảy sinh khi lập trình viên quên giải phóng bộ nhớ đã xin cấp trước đó. Trong chương trình java, lập trình viên không phải bận tâm đến việc cấp phát bộ nhớ. Quá trình cấp phát, giải phóng được thực hiện tự động, nhờ dịch vụ thu nhặt những đối tượng không còn sử dụng nữa (garbage collection).
- Cơ chế bẫy lỗi của java giúp đơn giản hóa quá trình xử lý lỗi và hồi phục sau lỗi.

2.2.5. Bảo mật

Java cung cấp một môi trường quản lý thực thi chương trình với nhiều mức để kiểm soát tính an toàn:

- Ở mức thứ nhất, dữ liệu và các phương thức được đóng gói bên trong lớp. Chúng chỉ được truy xuất thông qua các giao diện mà lớp cung cấp
- Ở mức thứ hai, trình biên dịch kiểm soát để đảm bảo mã là an toàn, và tuân theo các nguyên tắc của java

- Mức thứ ba được đảm bảo bởi trình thông dịch. Chúng kiểm soát xem bytecode có đảm các quy tắc an toàn trước khi thực thi

- Mức thứ tư kiểm soát việc nạp các lớp vào bộ nhớ để giám sát việc vi phạm giới hạn truy xuất trước khi nạp vào hệ thống.

2.2.6. Phân tán

Java được thiết kế để hỗ trợ các ứng dụng chạy trên mạng bằng các lớp mạng (`java.net`). Hơn nữa, java hỗ trợ nhiều nền chạy khác nhau nên chúng được sử dụng rộng rãi như là công cụ phát triển trên Internet, nơi sử dụng nhiều nền khác nhau

2.2.7. Đa luồng

Chương trình java cung cấp giải pháp đa luồng(Multithreading) để thực thi các công việc đồng thời. Chúng cũng cung cấp giải pháp đồng bộ giữa các luồng. Đặc tính hỗ trợ đa luồng này cho phép xây dựng các ứng dụng trên mạng chạy hiệu quả

2.2.8. Linh động

Java được thiết kế như một ngôn ngữ động để đáp ứng cho những môi trường mở. Các chương trình Java chứa rất nhiều thông tin thực thi nhằm kiểm soát và truy nhập đối tượng lúc chạy. Điều này cho phép khả năng liên kết động mã.

2.3. Các dạng chương trình ứng dụng của Java

2.3.1. Chương trình ứng dụng độc lập (Application)

Chương trình ứng dụng dạng độc lập là một chương trình nguồn mà sau khi dịch có thể thực hiện trực tiếp. Chương trình độc lập trong java bắt đầu thực hiện và kết thúc ở phương thức `main()` giống như hàm `main()` trong chương trình C/ C++

Khi xây dựng một ứng dụng độc lập cần lưu ý:

1. Tạo lập một lớp được định nghĩa bởi người sử dụng có phương thức `main()` gọi là lớp chính và bảo đảm nó được định nghĩa đúng theo đúng nguyên mẫu được quy định bởi java

2. Kiểm tra xem liệu tệp chương trình có tên trùng với tên của lớp chính và đuôi là "`.java`" hay không

3. Dịch tệp chương trình nguồn "`.java`" để tạo ra các tệp mã bytecode có đuôi "`.class`" tương ứng

4. Sử dụng chương trình thông dịch của Java để chạy chương trình đã dịch

2.3.2. Chương trình ứng dụng nhúng(Applet)

Applet là loại chương trình Java đặc biệt mà khi thực hiện mã lệnh của chúng phải được nhúng trong vào một trang web (các file có đuôi HTM hoặc HTML), các thẻ HTML sẽ được trình duyệt Web thực thi (như Netscape hoặc Internet Explorer) còn đoạn mã lệnh của Applet sẽ được máy ảo Java nhúng trong trình duyệt web thực thi. Cũng có thể dùng trình Appletviewer của JDK để thực thi một Applet.

Một chương trình dạng Applet bao gồm hai tệp: "`.java `" và "`.html `"

Chu trình hoạt động cuả Applet:

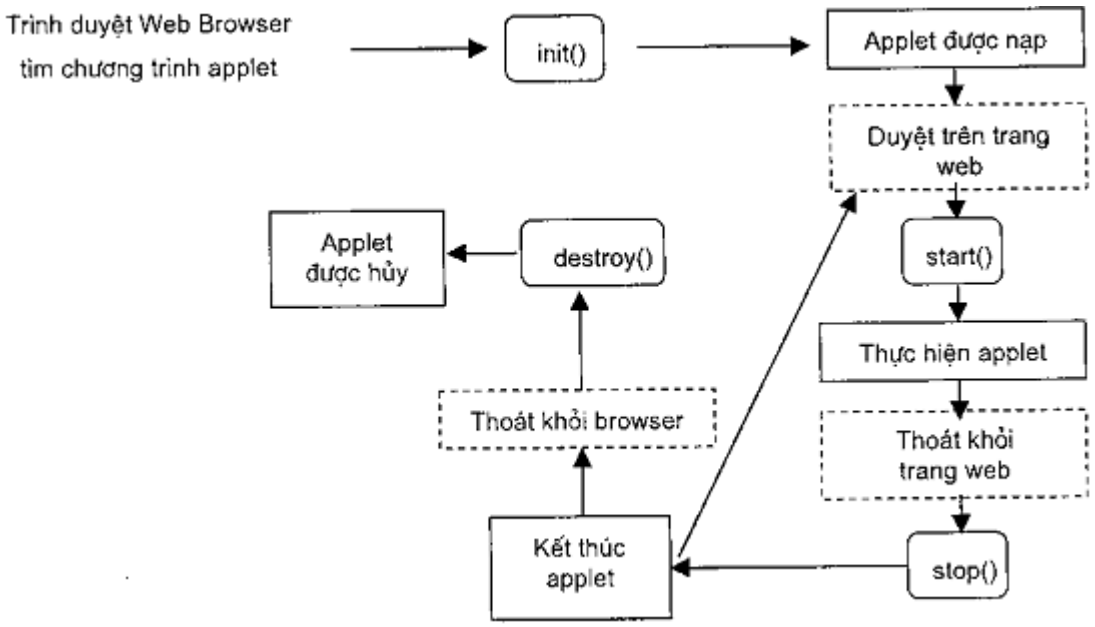
Chương trình ứng dụng Applet được thực hiện như sau:

- Khi một applet được nạp và chạy bởi Web Browser thì nó sẽ gửi thông điệp `init()` cùng với các dữ liệu, kích thước của Window để chương trình applet khởi động.

- Khi bắt đầu thực hiện, Web Browser thông báo cho applet bắt đầu bằng cách gọi phương thức `start()`.

- Khi rời khỏi trang Web có chứa applet thì chương trình applet này nhận được thông điệp `stop()` để dừng chương trình.

Hoạt động của chương trình applet được mô tả như hình dưới:



Hình 2.3- Chu trình hoạt động của applet

Trong đó:

- *init():* phương thức này được gọi khi applet được nạp lần đầu và được xem như là toán tử tạo lập cho applet.
- *start():* được gọi khi applet bắt đầu thực hiện, xuất hiện khi:
  + applet được nạp xuống
  + applet được duyệt lại
- *stop():* được gọi khi applet dừng thực hiện, nhưng chưa bị loại bỏ khỏi bộ nhớ.
- *destroy():* được gọi ngay trước khi applet kết thúc, khi trình duyệt Web được đóng lại và applet bị loại bỏ khỏi bộ nhớ.

2.3.3. Chương trình ứng dụng dạng lai ghép

Java cho phép xây dựng một chương trình có thể chạy được cả ở Web Browser (Applet) cũng như một ứng dụng độc lập (Application). Để xây dựng được một chương trình như thế phải:

- Định nghĩa lớp ứng dụng mở rộng từ lớp Applet
- Trong lớp ứng dụng phải có hàm `main()`

2.4. Cấu trúc của tệp chương trình Java

Tệp chương trình java có thể có các phần được đặc tả như sau:

- Định nghĩa một gói là tùy chọn thông qua định danh của gói (`package`). Tất cả các lớp, các `interface` được định nghĩa trong tệp chứa gói này đều

thuộc gói đó. Nếu bỏ qua định nghĩa gói thì các định nghĩa ở tệp này sẽ thuộc vào gói mặc định.

- Một số lệnh nhập `import`
- Một số định nghĩa lớp và `interface` có thể định nghĩa theo thứ tự bất kỳ. Trong đó thường là lớp `public`

Như vậy, cấu trúc của một tệp chương trình Java có thể khái quát như sau:

```
// Filename: New.java
    // Phần 1: tùy chọn
        // Định nghĩa gói
        package Têngói;
    // Phần 2: 0 hoặc nhiều hơn
        // các gói cần sử dụng
        import java.io.*;
    // Phần 3: 0 hoặc nhiều hơn
        // Định nghĩa các lớp và các interface
        public class New{...}
        class C1 {...}
        interface I1 {...}
        // ...
        class Cn {...}
        interface Im {...}
```

CHƯƠNG 3: LẬP TRÌNH SOCKET TRONG JAVA

3.1 Khái niệm Socket

3.1.1 Lịch sử hình thành

- Khái niệm Socket xuất hiện lần đầu tiên vào khoảng năm 1980 tại trường đại học Berkeley Mỹ. Đó là một chương trình được thiết kế để giúp máy tính nối mạng ở khắp mọi nơi có thể trao đổi thông tin với nhau. Lúc đầu có được sử dụng trên các máy Unix và có tên gọi là Berkeley Socket Interface.

- Tiếp đó cùng với sự phát triển của các ứng dụng mạng, socket được hỗ trợ trong nhiều ngôn ngữ lập trình và chạy trên nhiều nền tảng hệ điều hành khác nhau. Ví dụ như WinSock dùng cho các ứng dụng của Microsoft, Socket++ dùng cho các lập trình viên sử dụng Unix…

- Có câu hỏi đặt ra là tại sao chúng ta lại sử dụng Socket trong truyền thống giữa các máy tính. Để trả lời câu hỏi này chúng ta phải quay lại thời điểm trước khi Socket ra đời:

Trong thời kì này trên hệ thống Unix việc vào/ra dữ liệu được thực hiện theo mô hình 3 bước Open-Read/Write-Close. Để thực hiện việc vào ra dữ liệu trước hết chương trình phải tạo ra một kết nối với tài nguyên mà nó muốn giao tiếp(tài nguyên này có thể là bàn phím, bộ nhớ trong, file…), sau khi kết nối đã được thực hiện, chương trình có thể trao đổi dữ liệu thông qua các thao tác Read- đưa dữ liệu từ tài nguyên đã kết nối vào chương trình để xử lý hoặc Write- đưa dữ liệu đã xử lý từ chương trình ra tài nguyên. Một ví dụ điển hình cho kiểu vào/ra này là thao tác với file dữ liệu mà chúng ta khá quen thuộc trong các ngôn ngữ lập trình: Khi người lập trình muốn thao tác với một file dữ liệu họ tiến hành như sau:

+ Mở file cần sử dụng với các quyền thích hợp trên đó

+ Thực hiện việc đọc dữ liệu từ file để xử lý hay đưa dữ liệu để xử lý để ghi vào file.

+ Đóng file sau khi đã sử dụng xong.

- Khi việc trao đổi dữ liệu giữa các chương trình và kết nối mạng được đưa vào hệ thống Unix người ta mong muốn việc trao đổi dữ liệu giữa các chương trình cũng sẽ được thực hiện theo mô hình ba bước của vào/ra dữ liệu nhằm

tránh cho người lập trình những khó khăn khi giao tiếp với các tầng bên dưới tầng ứng dụng. Để làm được điều đó, socket được sử dụng. Khi hai chương trình muốn giao tiếp với nhau, mỗi chương trình sẽ tạo ra một socket, chúng đóng vai trò là các điểm cuối trong một kết nối và thực hiện trao đổi thông tin giữa hai chương trình. Đối với người lập trình, socket được xem như một tài nguyên hệ thống mà chương trình cần giao tiếp nên chương trình có thể thực hiện giao tiếp với socket theo mô hình ba bước giống như việc vào/ra dữ liệu. Như vậy sự ra đời của socket gắn liền với nhu cầu truyền thông máy tính. Sau đây chúng ta sẽ đưa ra định nghĩa cụ thể về socket.

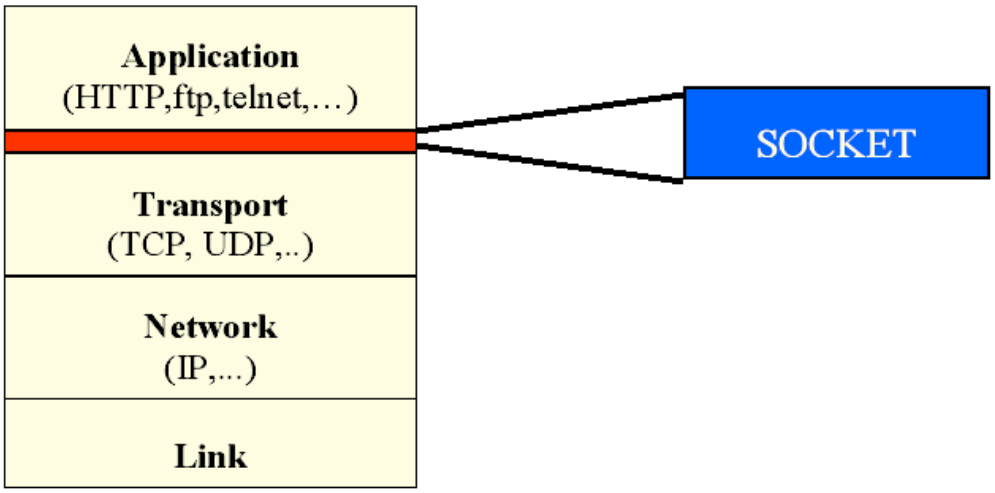
Định nghĩa

- Có nhiều định nghĩa khác nhau về socket tùy theo cách nhìn của người sử dụng.

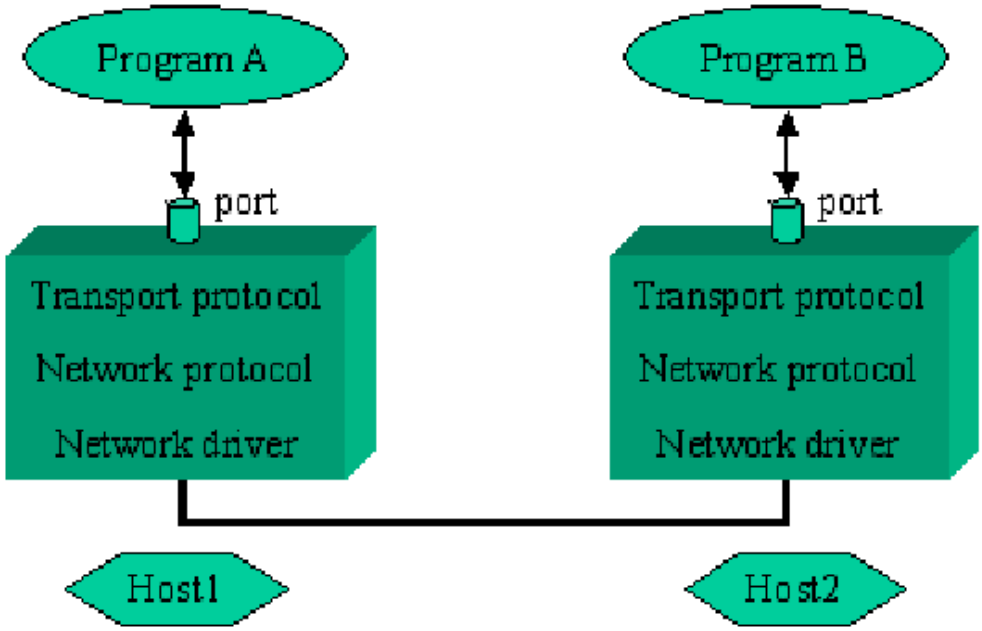
- Một cách tổng quát nhất có thể định nghĩa: *Một Socket là một điểm cuối trong một kết nối giữa hai chương trình đang chạy trên mạng*

- Nhìn trên quan điểm của người phát triển ứng dụng người ta có thể định nghĩa *Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (được gán nhãn là Client) và một chương trình cung cấp dịch vụ (được gán nhãn là server) trên mạng hoặc trên cùng một máy tính.*

- Đối với người lập trình, họ nhìn nhận *Socket như một giao diện nằm giữa tầng ứng dụng và tầng khác* trong mô hình mạng OSI có nhiệm vụ thực hiện việc giao tiếp giữa chương trình ứng dụng với các tầng bên dưới của mạng.



Hình 3.1- Mô hình OSI rút gọn



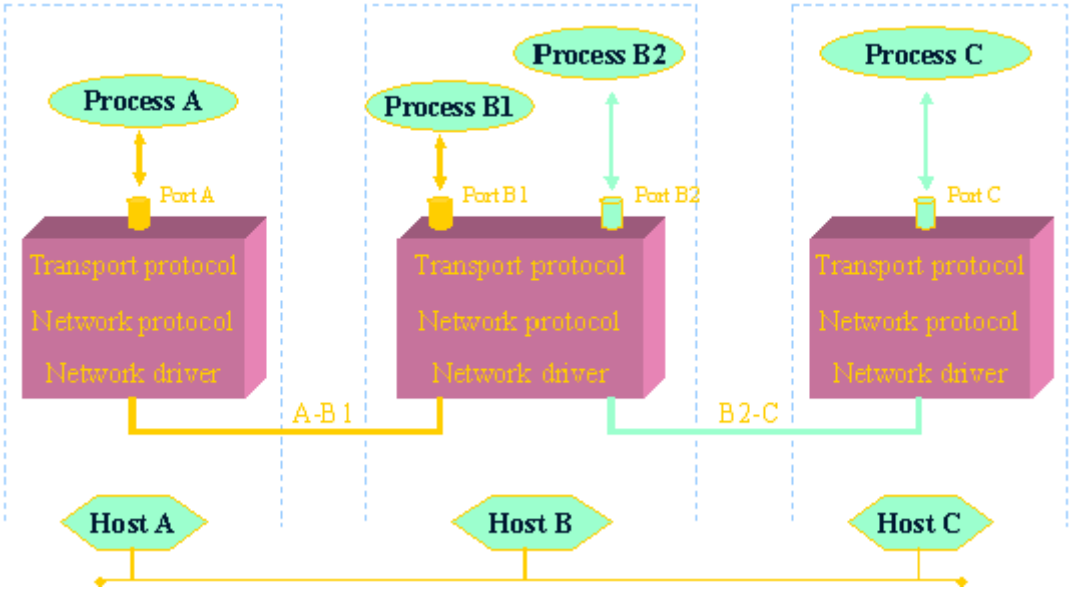
Hình 3.2- Mô hình Socket

- Tuy nhiên, các lập trình viên hiện nay gần như luôn luôn bị ngăn cản tạo socket riêng bằng cách thủ công bởi dù bạn dùng Java, serlet hay CGI, PHP… có thể bạn sẽ không bao giờ mở được cổng một cách tường minh. Thay vào đó các lập trình viên sử dụng thư viện socket được hỗ trợ sẵn bởi các ngôn ngữ lập trình. Như vậy các socket vẫn tồn tại để kết nối các ứng dụng của người dùng, nhưng các chi tiết của socket được ẩn trong những lớp sâu hơn để mọi người không phải động chạm đến.

Số hiệu cổng của Socket

- Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng. Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống. Khi quá trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gửi 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 3.3- Cổng trong Socket

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

Địa chỉ máy tính có thể là địa chỉ IP dạng 203.162.88.162 hay là địa chỉ cho dạng trên miền như www.hpu.edu.vn

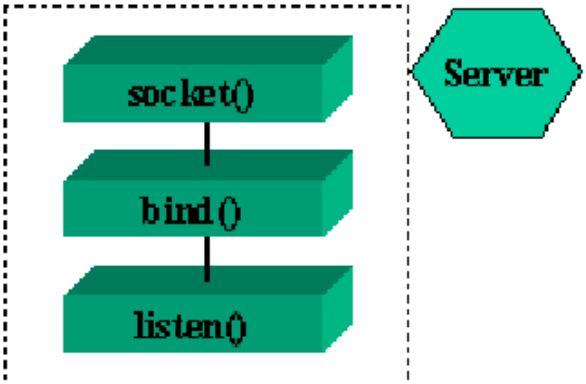
Số hiệu cổng gán cho Socket phải duy nhất trên phạm vi máy tính đó, có giá trị trong khoảng từ 0 đến 65535 (16 bit). Trong thực tế thì các số hiệu cổng từ 0 đến 1023 (gồm có 1024 cổng) đã dành cho các dịch vụ nổi tiếng như: http: 80, telnet:21, ftp:23,…. Nếu chúng ta không phải là người quản trị thì nên dùng từ cổng 1024 trở lên.

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

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

Mô hình Clients/Server sử dụng Socket ở chế độ hướng nối kết TCP

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

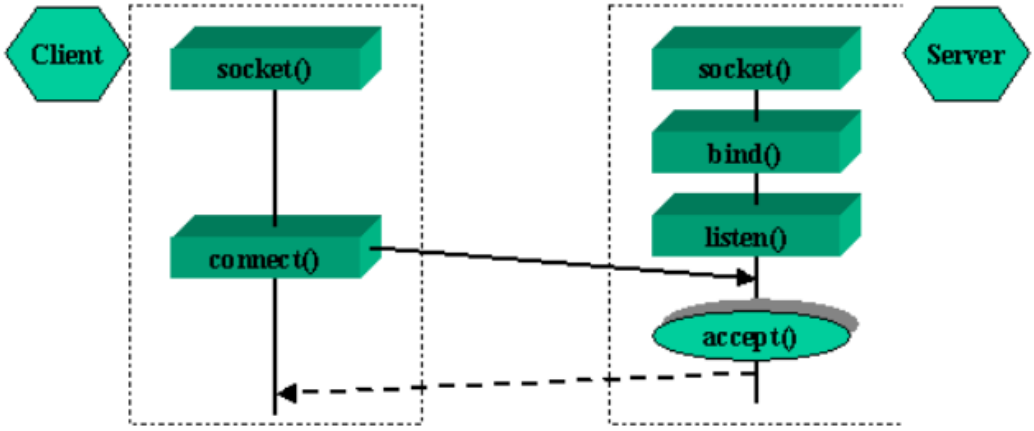


`Socket()`: Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.

`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.

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

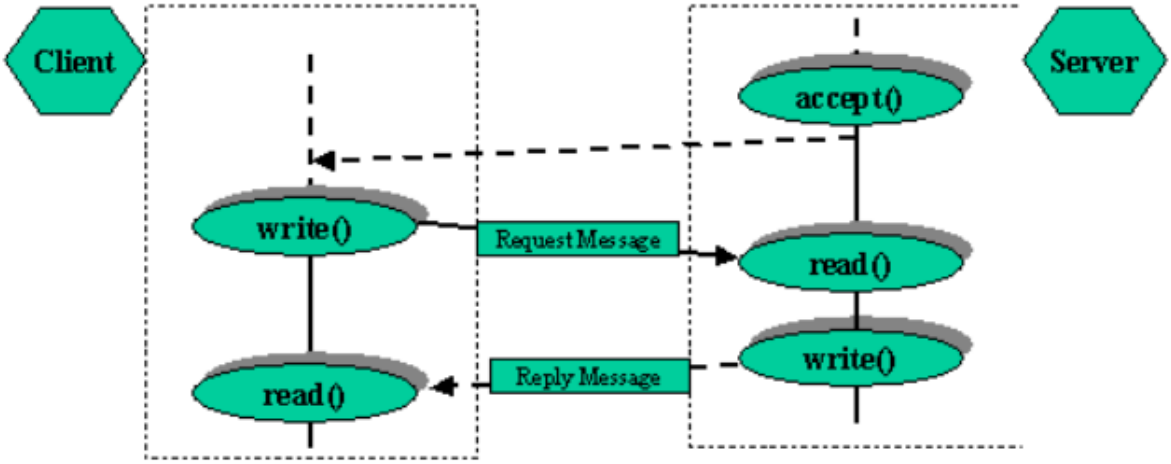


`Socket()`: Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng chưa sử dụng cho socket của Client.

`Connect()`: Client gửi 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.



Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh `read()` và nghẽ cho đến khi có thông điệp yêu cầu(Reaquest Message) từ client gửi đến.

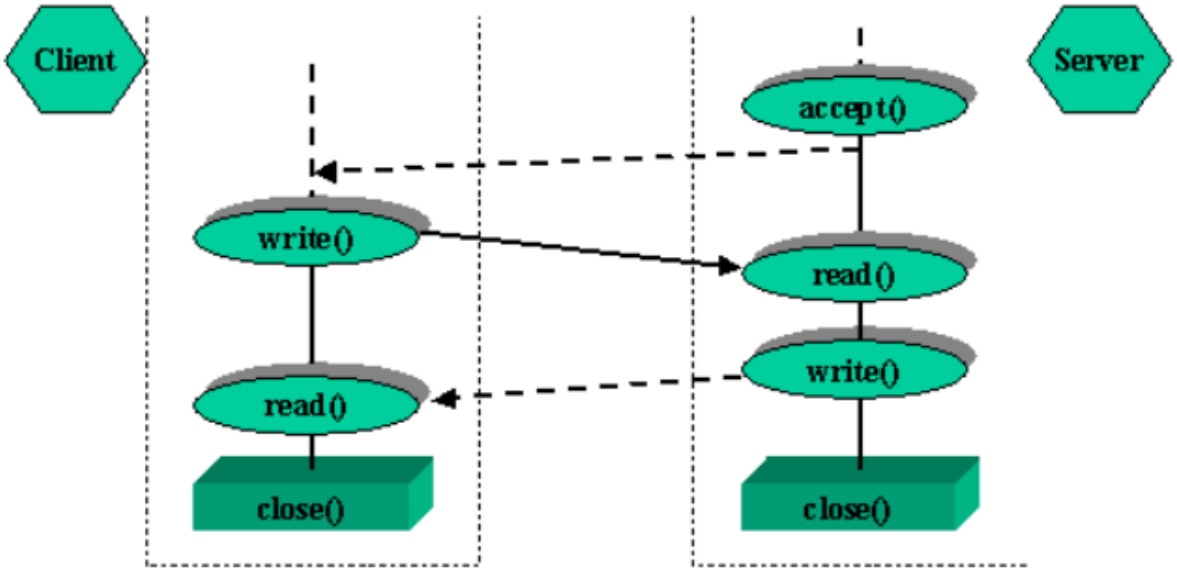
Server phân tích và thực thi yêu cầu. Kết quả sẽ được gửi về client bằng lệnh `write()`.

Sau khi gửi yêu cầu bằng lệnh `write()`, client chờ nhận thông điệp kết quả (Reply Message) 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 các thông điệp, quy tắc bắt tay, đồng bộ hóa…). Thông thường client sẽ là người gửi 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 quy định của giao thức. Ngược lại, nếu chúng ta phát triển một ứng dụng clients/server riêng của mình, thì công việc đầu tiên chúng ta phải thực hiện là đi xây dựng protocol cho ứng dụng.

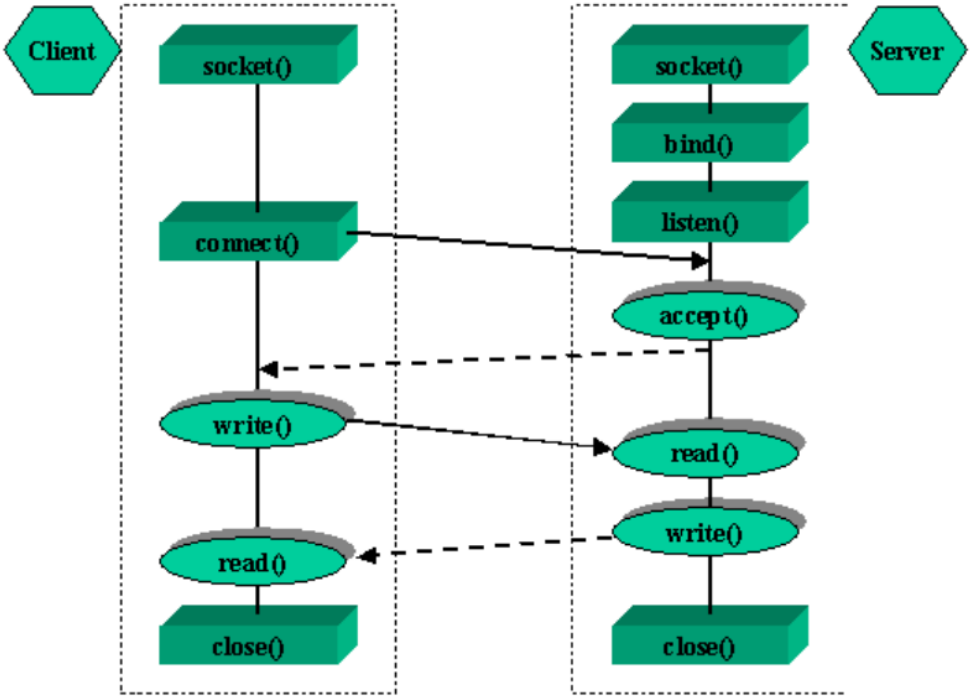
Giai đoạn 4: Kết thúc phiên làm việc



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:



3.1.2 Nguyên lý hoạt động

- Ta đã thấy khi hai ứng dụng muốn trao đổi dữ liệu qua mạng, chúng sẽ tạo ra ở mỗi phía một socket và trao đổi dữ liệu bằng cách đọc/ghi từ socket. Để hiểu rõ cách thức socket trao đổi dữ liệu chúng ta hãy xem xét nguyên lý hoạt động của chúng.

- Trước hết chúng ta hãy xem xét làm thế nào các socket có thể xác định được nhau. Khi một chương trình tạo ra một socket, một định danh dạng số (định danh dạng số này còn được gọi là số hiệu cổng) sẽ được gán cho socket. Việc gán số hiệu cổng này cho socket có thể được thực hiện bởi chương trình hoặc hệ điều hành tùy theo cách socket được sử dụng như thế nào. Trong mỗi gói tin mà socket gửi đi có chứa hai thông tin để xác định đích đến của gói tin:

+ Một địa chỉ mạng để xác định hệ thống sẽ nhận gói tin.

+ Một số định danh cổng để nói cho hệ thống đích biết socket nào trên nó sẽ nhận dữ liệu.

- Nhờ hai thông tin này mà gói tin có thể đến được đúng máy tính chứa socket mà nó cần đến (nhờ địa chỉ mạng) và được phân phối đến đúng socket đích (nhờ địa chỉ cổng của socket đích).

- Dưới góc độ lập trình các socket thường làm việc theo cặp, một socket đóng vai trò làm server còn các socket khác đóng vai trò như clients. Socket phía server xác định một cổng cho giao tiếp mạng, sau đó chờ nghe yêu cầu mà client gửi tới nó bằng client socket. Do đó các cổng cho server socket phải được biết bởi các chương trình client. Ví dụ server FTP sử dụng một socket để nghe tại cổng 21 do đó nếu một chương trình client muốn giao tiếp với server FTP nó cần phải kết nối đến socket có số hiệu cổng 21.

- Như vậy số hiệu cổng của socket phía server được xác định bởi chương trình, ngược lại cổng cho client socket được xác định bởi hệ điều hành. Khi một socket phía client gửi một gói tin tới socket phía server thì trong gói tin đã có chứa thông tin về địa chỉ của hệ thống client và cổng của socket phía client nên server hoàn toàn có thể gửi thông tin phản hồi cho client.

- Chúng ta có thể khái quát quá trình trao đổi dữ liệu thông qua các socket như sau:

+ Chương trình phía server tạo ra một socket, socket này được chương trình gắn với một cổng trên server. Sau khi được tạo ra socket này (ta gọi là socket phía server) sẽ chờ nghe yêu cầu từ phía clients.

+ Khi chương trình phía clients cần kết nối với một server, nó cũng tạo ra một socket, socket này cũng được hệ điều hành gắn với một cổng. Chương trình client sẽ cung cấp cho socket của nó (ta gọi là socket phía client) địa chỉ mạng và cổng của socket phía server và yêu cầu thực hiện kết nối (nếu chương trình định sử dụng giao thức hướng kết nối) hoặc truyền dữ liệu (nếu chương trình sử dụng giao thức không hướng kết nối)

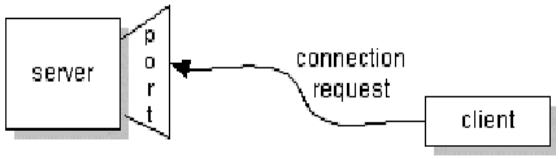
+ Chương trình phía server và chương trình phía clients trao đổi dữ liệu với nhau bằng cách đọc từ socket hoặc ghi vào socket của mình. Các socket ở hai phía nhận dữ liệu từ ứng dụng và đóng gói để gửi đi hoặc nhận các dữ liệu được gửi đến và chuyển cho chương trình ứng dụng bởi socket ở cả hai phía đều biết được địa chỉ mạng và địa chỉ cổng của nhau.

- Ở bước thứ hai chúng ta thấy chương trình ứng dụng phải lựa chọn giao thức mà nó định sử dụng để trao đổi dữ liệu. Tuỳ theo việc chúng ta sử dụng giao thức nào (TCP hay UDP) mà cách thức xử lý trước yêu cầu của clients có thể khác.

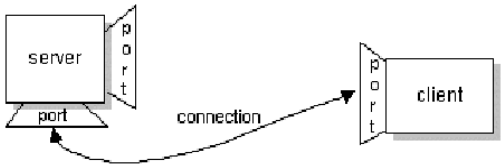
- Sau đây chúng ta sẽ xem xét chi tiết cách thức trao đổi dữ liệu của socket với từng loại giao thức.

Socket hỗ trợ TCP

a. Ở phía Server: Khi một ứng dụng trên server hoạt động nó sẽ tạo ra một socket và đăng ký với server một cổng ứng dụng và chờ đợi yêu cầu kết nối từ phía clients qua cổng này.



Hình 3.4-Clients gửi yêu cầu đến server



Hình 3.5- Server chấp nhận yêu cầu và tạo một socket để phục vụ clients

b. Ở phía clients: Nó biết địa chỉ của máy trên đó server đang chạy vào cổng và server đang chờ nghe yêu cầu. Do đó khi muốn kết nối đến server, nó cũng tạo một socket chứa địa chỉ máy client và cổng của ứng dụng trên máy clients đồng thời clients sẽ cung cấp cho socket của nó địa chỉ và cổng của server mà nó cần kết nối và yêu cầu socket thực hiện kết nối.

Khi server nhận được yêu cầu kết nối từ clients, nếu nó chấp nhận thì server sẽ sinh ra một socket mới được gắn với một cổng khác với cổng mà nó đang nghe yêu cầu. Sở dĩ server làm như vậy bởi nó cần cổng cũ để tiếp tục nghe yêu cầu từ phía clients trong khi vẫn cần một kết nối với clients.

Sau đó chương trình ứng dụng phía server sẽ gửi thông báo chấp nhận kết nối cho clients cùng thông tin về địa chỉ cổng mới của socket mà nó dành cho clients.

c. Quay lại phía clients, nếu kết nối được chấp nhận nghĩa là socket của nó đã được tạo ra thành công và nó có thể sử dụng socket để giao tiếp với server bằng cách viết và ghi tới socket theo cách giao tiếp với một tài nguyên trên máy tính thông thường.

Socket hỗ trợ UDP

a. Ở phía Server: Khi một ứng dụng trên server hoạt động nó sẽ tạo ra một socket và đăng ký với server một cổng ứng dụng và chờ đợi yêu cầu kết nối từ phía clients qua cổng này.

b. Ở phía Clients: Nó biết địa chỉ của máy trên đó server đang chạy vào cổng và server đang chờ nghe yêu cầu. Do đó khi muốn giao tiếp với server, nó cũng tạo ra một socket chứa địa chỉ máy clients và cổng của ứng dụng trên máy clients đồng thời clients sẽ cung cấp cho socket của nó địa chỉ và cổng của server mà nó cần kết nối. Khi clients muốn gửi tin đế server nó sẽ chuyển dữ liệu cho socket của mình, socket này sẽ chuyển thẳng gói tin mà client muốn gửi tới server dưới dạng một datagram có chứa địa chỉ máy server và cổng mà server đang chờ nghe yêu cầu. Như vậy không hề có một kết nối nào được thực hiện giữa client với server và server cũng không cần tạo ra một socket khác để kết nối với clients thay vào đó server dùng ngay cổng ban đầu để trao đổi dữ liệu.

3.2 Socket trong Java

* Xây dựng chương trình Client ở chế độ có nối kết

Các bước tổng quát:

- Mở một socket nối kết đến server đã biết địa chỉ IP(hay tên miền) và số hiệu cổng.
- Lấy `InputStream` và `OutputStream` gán với socket.
- Tham khảo protocol của dịch vụ để định dạng đúng dữ liệu trao đổi với server.
- Trao đổi dữ liệu với server nhờ vào các `InputStream` và `OutputStream`.
- Đóng socket trước khi kết thúc chương trình.

3.2.1 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 clients.

`public Socket(String HostName, int PortNumber) throws IOException`

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

HostName: Địa chỉ IP hoặc tên logic theo dạng tên miền.

PortNumber: Có giá trị từ 0....65535

public InputStream getInputStream()

Phương thức này trả về `InputStream` nối với Socket. Chương trình clients dùng `InputStream` này để nhận dữ liệu từ server gửi về

public OutputStream getOutputStream()

Phương thức này trả về `OutputStream` nối với socket. Chương trình client dùng `OutputStream` này để gửi dữ liệu cho server.

public close()

Phương thức này sẽ đóng socket lại, giải phóng kênh ảo, xóa nối kết giữa clients và server.

3.2.2 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 kết nối và không kết nối.

Chương trình TCPEchoClient sẽ kết nối đến EchoServer ở chế độ có kết nối, lần lượt gửi đến EchoServer ở chế độ có kết nối, lần lượt gửi đến EchoServer 10 kí tự từ ”0” đến “9” chờ nhận kết quả trả về và hiển thị chúng ra màn hình

Hãy lưu chương trình sau vào tệp tin TCPEchoClient.java

```
Import java,io.*
Import java.net.socket;
Public class TCPEchoClient{
    Public static void main(String arg[]){
    Try{
    Socket s=new socket(args[0],7);  // Nối kết đến Server
    InputStream is =s.getInputStream();  // Lấy InputStream
    OutputStream os = s.getOutputStream();
    For(int i= 0; i<=9;i++){
        Os.write(i);
        Int ch= is.read();
        System.out.print((char)ch);
    }
    }
    Catch(IOException ie){
```

```
            System.out.println("Loi khong tao duoc socket");
            }
    }
}
```

Biên dich và thực thi chương trình như sau:

Chương trình này nhận một đối số là địa chỉ IP hay tên miền của máy tính mà ở đó EchoServer 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 clients kết nối đến EchoServer trên cùng với máy đó.

3.3 Một số lớp trong lập trình Java Socket

Java hỗ trợ lập trình mạng thông qua các lớp trong gói `java.net`. Một số lớp tiêu biểu cần dùng cho lập trình clients/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 chươnng trình clients ở chế độ có kết nối.

`ServerSocket`: Hỗ trợ các phương thức liên quan đến socket cho chương trình server ở chế độ có kết nối.

`DatagramSocket`: Hỗ trợ các phương thức liên quan đến socket ở chế độ không kết nối cho cả clients và server.

`DatagramPacket`: Lớp cài đặt gói tin dùng thư tín người dùng (`Datagram Packet`) trong giao tiếp giữa clients và server ở chế độ không kết nối.

CHƯƠNG 4: XÂY DỰNG CHƯƠNG TRÌNH ỨNG DỤNG

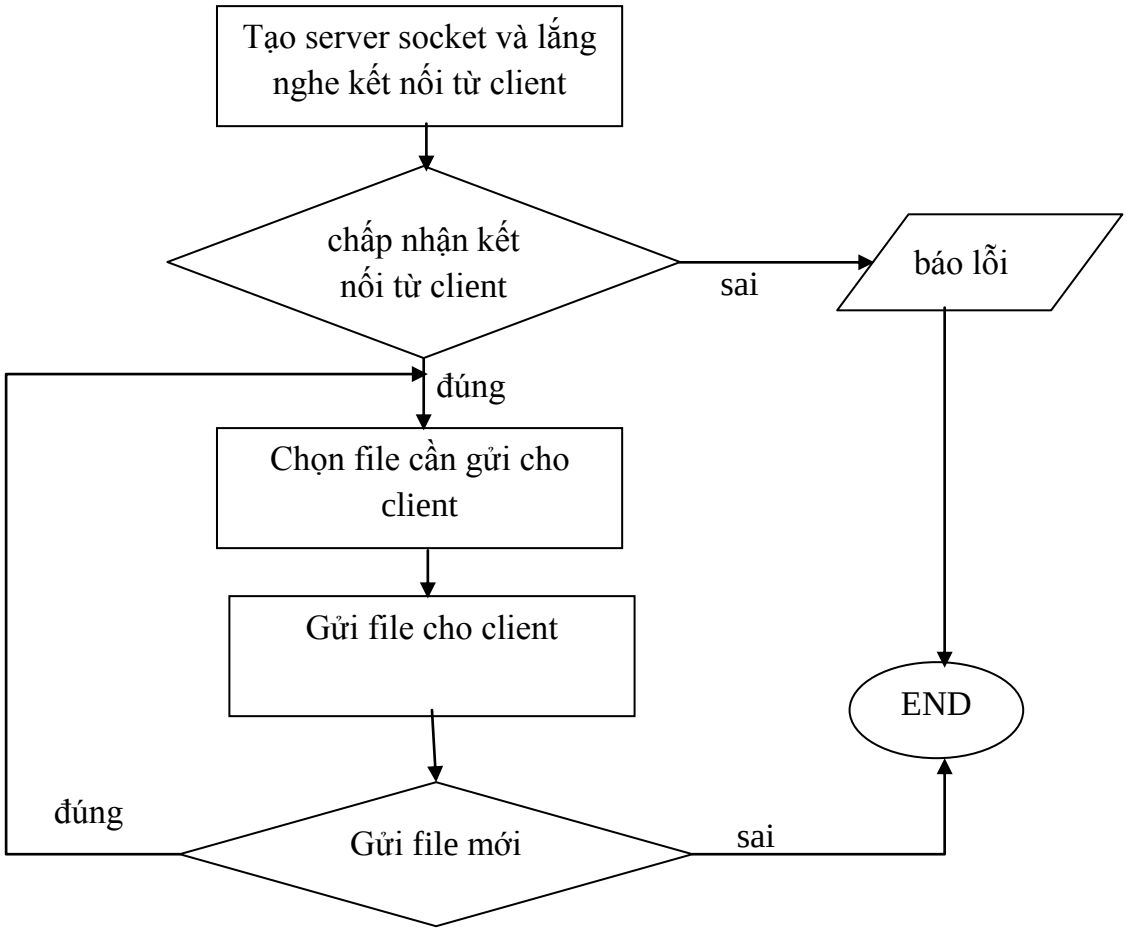
4.1. Giới thiệu

Trao đổi dữ liệu giữa hai máy tính trong mạng thực chất là sự trao đổi dữ liệu giữa hai chương trình ứng dụng chạy trên hai máy tính đó. Trong đó, một chương trình được gán nhãn là server và một chương trình được gán nhãn là client, có nhiều phương pháp để xây dựng chương trình ứng dụng mạng nhưng phương pháp sử dụng phổ biến là lập trình ứng mạng dựa trên cơ chế socket. Trong chương này sẽ trình bầy một ứng dụng của lập trình socket TCP là xây dựng chương trình truyền file qua mạng giữa hai máy tính bằng Java Socket TCP.

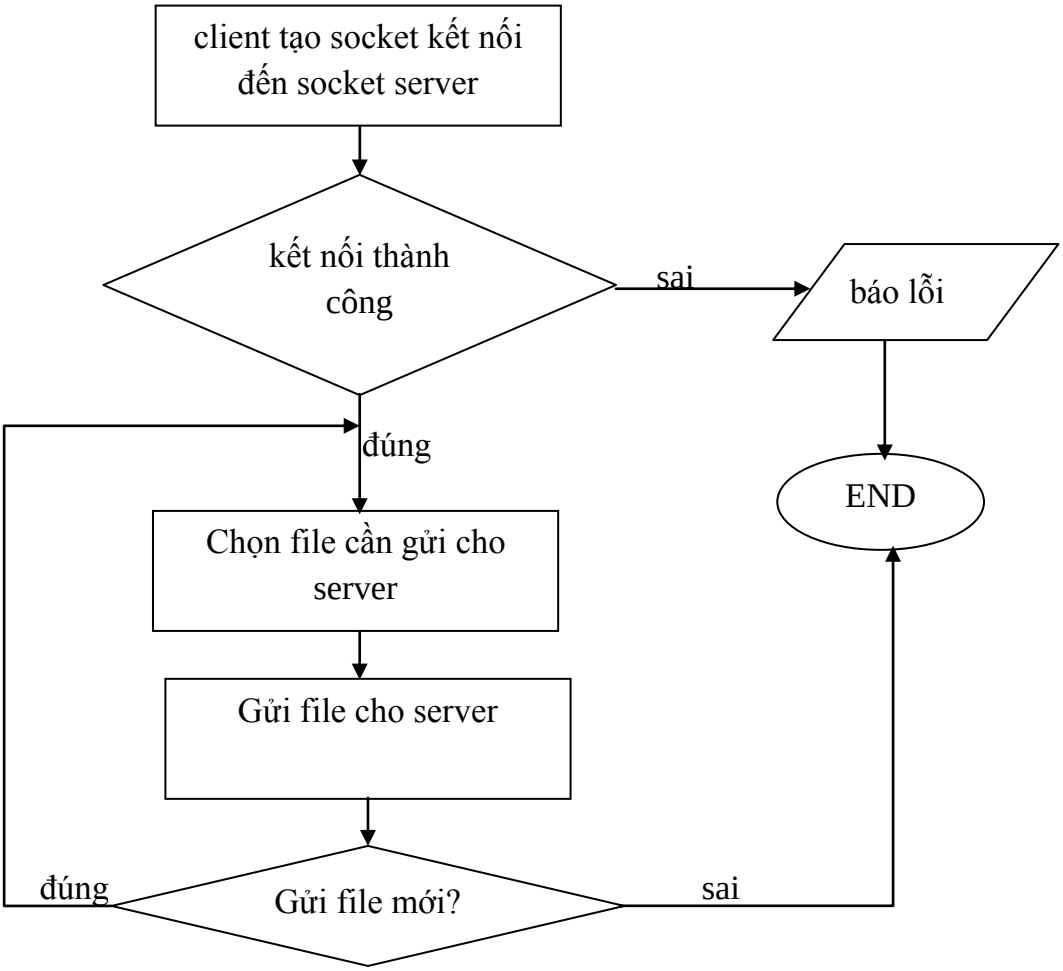
4.2. Phân tích chương trình

Chương trình ứng dụng được xây dựng theo mô hình clients/server. Chương trình bao gồm hai mô đun server và client. Người sử dụng có thể truyền file từ phía client cho server hoặc ngược lại.

Mô đun phía server



Mô đun phía client



4.3 Cơ chế hoạt động của chương trình

Chương trình gồm hai mô đun. Phía server là file chương trình có tên là `FileTransferServer.java`, phía client là file chương trình có tên là `FileTransferClient.java`. Sau khi biên dịch file `.java` này ta nhận được các file `.class` tương ứng. Chương trình được thực thi tại dấu nhắc hệ thống theo cú pháp

```
c:\>java FileTransferServer
c:\>java FileTransferClient
```

Chạy chương trình ở server mode:

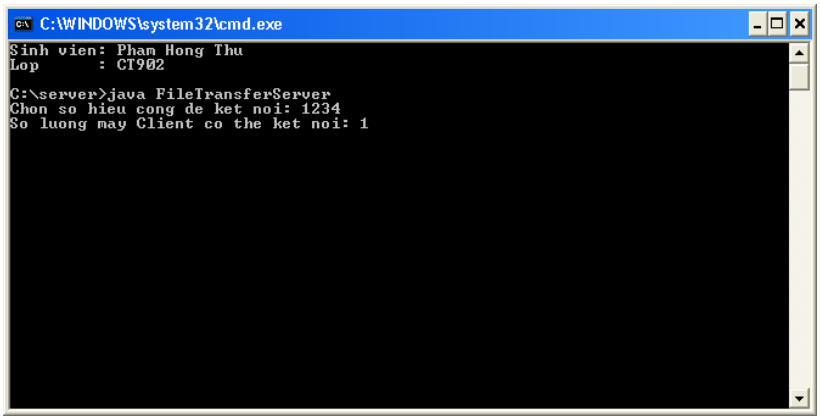
Chương trình chạy phía server đã được chỉ định chạy server mode, sau đó máy tính này sẽ chờ đợi các kết nối từ phía clients đến nó. Ta phải nhập tiếp port number, ứng với server mode này thì ta có thể chọn bất cứ port number nào lớn hơn 1024, vì những port number dưới 1024 đã bị giữ trước và sử dụng bởi hệ thống (well-known ports). Chương trình này cũng có thể thực hiện giao nhận file giữa một server và nhiều máy clients đồng thời trong một hệ thống mạng nên nó sẽ yêu cầu ta nhập vào số lượng máy client lớn nhất có thể kết nối đến server này.

Chạy chương trình ở client mode:

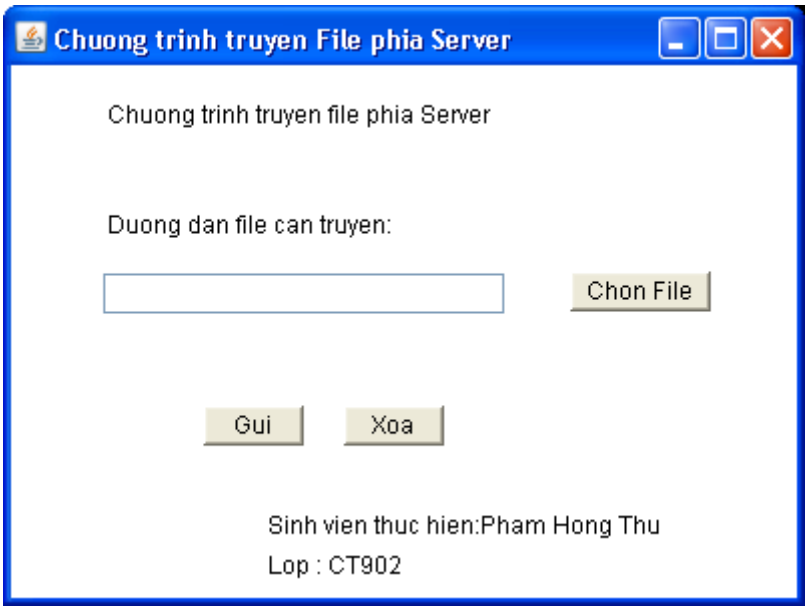
Tương tự cho phía bên client, chương trình sẽ yêu cầu nhập vào địa chỉ của server (host address), ta có thể nhập địa chỉ IP hay nhập vào tên của máy chạy server mode đều được (trong trường hợp trên hình bên dưới thì tên máy chạy server mode có địa chỉ IP trong mạng LAN là 192.168.1.43). Tiếp tục ta sẽ nhập port number (số hiệu cổng) của server socket (đã biết) cần kết nối đến.

4.4. Giao diện chương trình

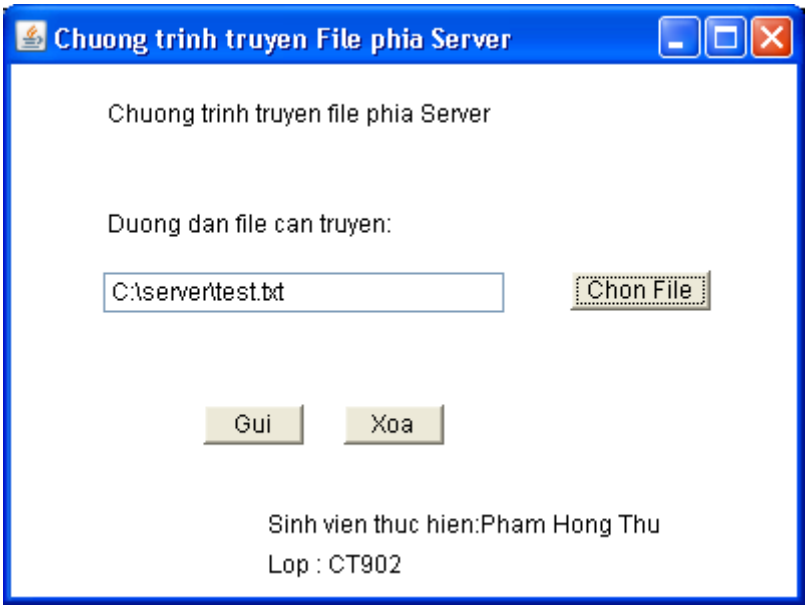
4.4.1 Giao diện phía Server



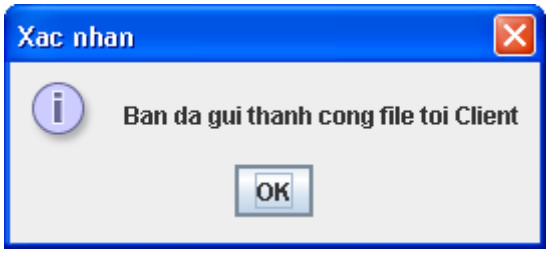
Hình 4.1- Lựa chọn số hiệu cổng của Server và số lượng Client cần kết nối



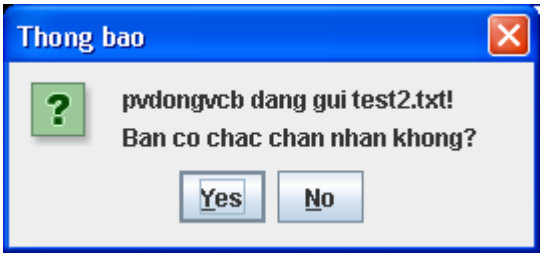
Hình 4.2- Giao diện chính của chương trình phía Server



Hình 4.3- Giao diện phía server khi chọn file truyền đi



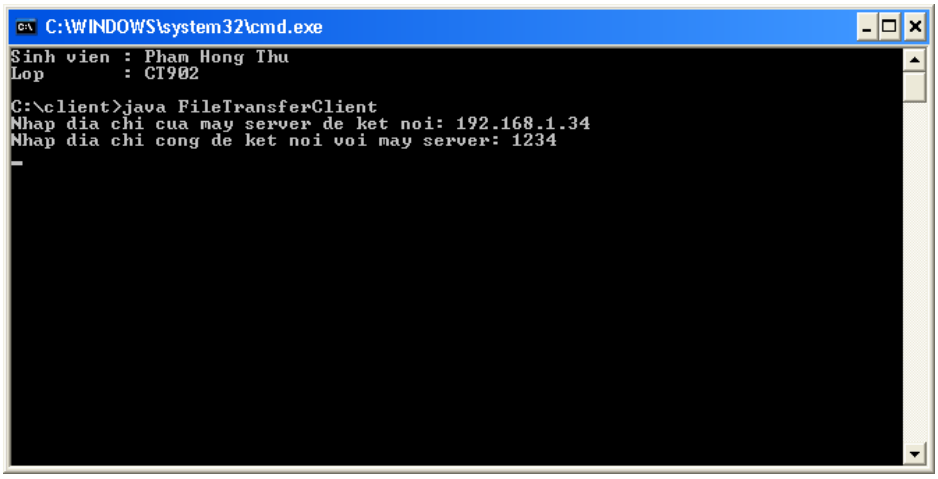
Hình 4.4- Thông báo gửi file thành công từ Server đến Client



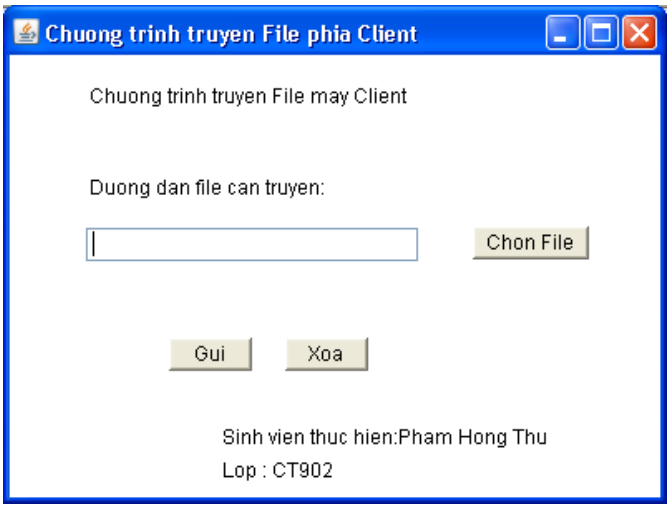
Hình 4.5- Thông báo bên Clien đang gửi file tới cho Server

Hình 4.6- Thông báo bạn đã nhận thành công file được gửi từ Client

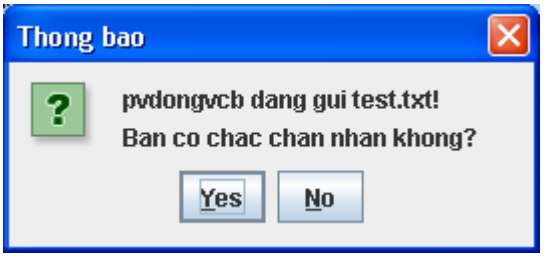
4.4.2 Giao diện phía Client



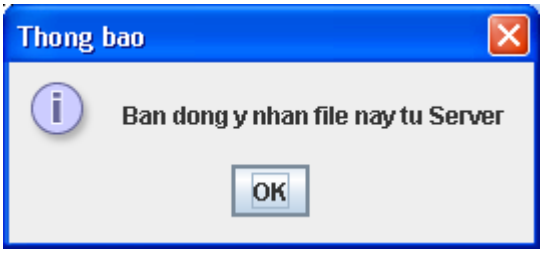
Hình 4.7- Nhập địa chỉ máy Server và số hiệu cổng của Server để kết nối



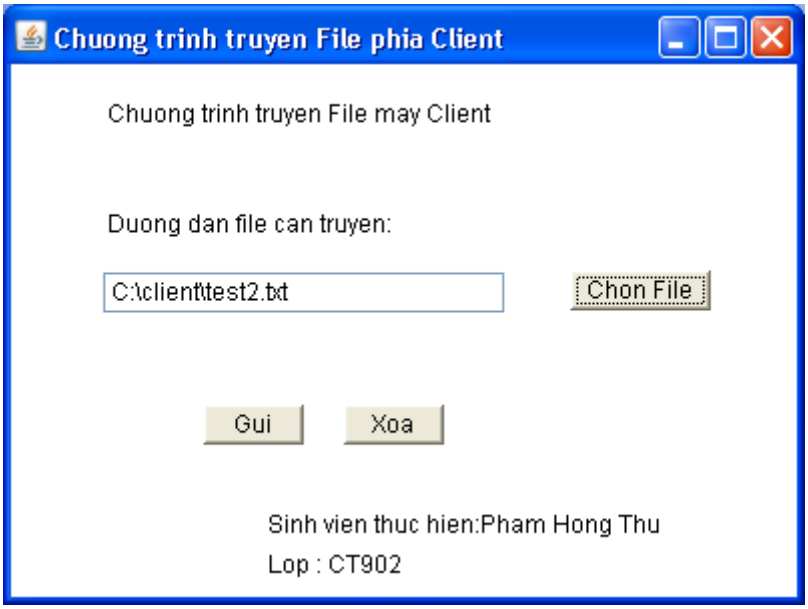
Hình 4.8- Giao diện chính của chương trình phía Client



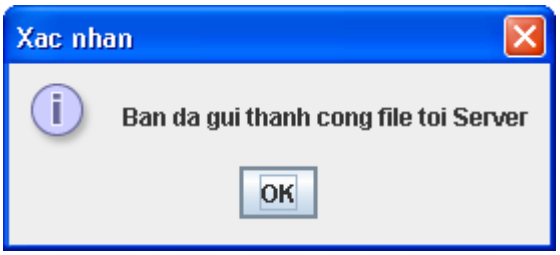
Hình 4.9- Thông báo bên Server đang gửi file tới cho Client



Hình 4.10- Thông báo bạn đã nhận thành công file được gửi từ Server



Hình 4.11- Giao diện phía server khi chọn file truyền đi



Hình 4.12- Thông báo gửi file thành công từ Client đến Server

4.5 Nhận xét

Chương trình ứng dụng được lựa chọn viết bằng ngôn ngữ Java, ta chọn Java vì những lý do sau đây:

Thứ nhất, những ứng dụng mạng kiểu này sẽ gọn gàng hơn khi viết bằng Java, với Java sẽ có ít dòng mã hơn, và mỗi dòng có thể dễ dàng giải thích với cả những người lập trình mới bắt đầu.

Thứ hai, những chương trình ứng dụng mô hình clients/server lập trình bằng Java đã gia tăng ngày càng thông dụng và nó có thể trở thành tiêu chuẩn cho lập trình mạng trong vài năm tới. Java là một ngôn ngữ độc lập nền, nó có cơ chế bẫy lỗi (điều quản các ngoại lệ), có thể giải quyết hầu nó các lỗi xảy ra trong quá trình xuất/nhập và những hoạt động mạng và khả năng phân luồng (thread) mạnh cung cấp những phương pháp đơn giản để xây dựng những server mạnh mẽ.

KẾT LUẬN

Truyền file qua mạng là một trong những ứng dụng phổ biến trên mạng LAN và Internet như: tải xuống các file từ một máy chủ file ở xa, gửi/nhận thư điện tử, ... Truyền file qua mạng dựa trên Socket TCP là một phương pháp truyền file có độ tin cậy cao bởi vì trước khi truyền nó cần thiết lập thành công kênh truyền dữ liệu

Không phải là phương pháp thay thế hoàn toàn những phương pháp truyền file khác mà ta đã từng sử dụng. Bản chất của phương pháp truyền file dựa vào Socket TCP là nhằm tăng thêm hiệu xuất làm việc. Đề tài "Tìm hiểu lập trình socket TCP trong java và ứng dụng truyền file qua mạng"đã đạt được kết quả nhất định.

Về cơ sở lý thuyết, đồ án đã trình bầy được các nội dung về mạng máy tính, sơ lược về ngôn ngữ Java, lập trình Socket TCP nói chung và lập trình Socket TCP trong Java nói riêng; các nội dung liên quan đến truyền file quua mạng.

Về ứng dụng đồ án đã phân tích một cách khá chi tiết cơ chế hoạt động của chương trình ở phía clients, phía server và đã cài đặt thành công chương trình. Java là một ngôn ngữ mạnh mẽ, tính bảo mật cao và độc lập với nền, do đó chương trình ứng dụng của đồ án có thể dễ dàng chạy trên các hệ thống khác nhau mà không phải lập trình lại.

Tuy nhiên, do hạn chế về thời gian và trình độ nên nhiều tính năng của chương trình chưa được hoàn thiện. Trong thời gian tới, chương trình sẽ được hoàn thiện theo hướng bổ sung các chức năng cho phù hợp yêu cầu đặc thù của việc truyền file qua mạng, có thể áp dụng vào thực tế cuộc sống.

Tài liệu tham khảo

Tài liệu tiếng Việt

[1]. *Giáo trình Lập trình Hướng đối tượng JAVA* - Ngọc Anh Thư Press- NXB Thống Kê

[2]. *JAVA Lập trình mạng* - Nguyễn Phương Lan và Hoàng Đức Hải - NXB Giáo Dục

[3]. *Lập trình Socket với TCP (bản điện tử)*

[4]. *Giáo trình Hệ thống mạng CCNA*- Nguyễn Hồng Sơn - NXB Giáo dục năm 2001

Tài liệu tiếng Anh

[1]. *Computer Networking* - By James F. Kurose and Keith W. Ross - Addison Wesley

[2]. *IP Network Address Translation* - Michael Hasenstein -1997

Tài liệu khác

[1]. Website: *http://quantrimang.com*/

[2]. Website: *http://www.planet-source-code.com/*

[3]. Website: http://www.javavietnam.org/

Phụ lục

Mã nguồn chương trình ứng dụng

1. Mã nguồn chương chình phía Server

```
//FileTransferServer

import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class FileTransferServer extends Frame {
public static String strHostAddress = "";
public static int intPortNumber = 0, intMaxClients = 0;
public static Vector vecConnectionSockets = null;
public static FileTransferServer objFileTransfer;
public static String strFileName = "",strFilePath = "";
public static Socket clientSocket = null;
public static ObjectOutputStream outToServer = null;
public static ObjectInputStream inFromServer = null;
public static void main (String [] args) throws IOException {
BufferedReader stdin = new BufferedReader(new InputStreamReader
(System.in));
System.out.print("Chon so hieu cong de ket noi: ");
System.out.flush();
intPortNumber = Integer.parseInt(stdin.readLine());
System.out.print("So luong may Client co the ket noi: ");
System.out.flush();
intMaxClients = Integer.parseInt(stdin.readLine());
objFileTransfer = new FileTransferServer();
}
public Label lblSelectFile;
public Label lblTitle;
public Label lblStudentName;
public Label lblStudentClass;
public TextField tfFile;
public Button btnBrowse;
public Button btnSend;
public Button btnReset;
public FileTransferServer () {
```

```
setTitle("Chuong trinh truyen File phia Server");
setSize(400 , 300);
setLayout(null);
addWindowListener(new WindowAdapter () { public void windowClosing
(WindowEvent e) { System.exit(0); } } );
lblTitle = new Label("Chuong trinh truyen File qua mang dua tren
Socket TCP ");
add(lblTitle);
lblTitle.setBounds(50,30,450,50);
lblSelectFile = new Label("Duong dan file can truyen:");
add(lblSelectFile);
lblSelectFile.setBounds(50,100,200,20);
lblStudentName = new Label("Sinh vien thuc hien:Pham Hong Thu");
add(lblStudentName);
lblStudentName.setBounds(130,250,200,20);
lblStudentClass = new Label("Lop : CT902");
add(lblStudentClass);
lblStudentClass.setBounds(130,270,100,20);
tfFile = new TextField("");
add(tfFile);
tfFile.setBounds(50,134,200,20);
btnBrowse = new Button("Chon File");
btnBrowse.addActionListener(new buttonListener());
add(btnBrowse);
btnBrowse.setBounds(283,133,70,20);
btnSend = new Button("Gui");
btnSend.addActionListener(new buttonListener());
add(btnSend);
btnSend.setBounds(100,200,50,20);
btnReset = new Button("Xoa");
btnReset.addActionListener(new buttonListener());
add(btnReset);
btnReset.setBounds(170,200,50,20);
show();
vecConnectionSockets = new Vector();
try {
ServerSocket welcomeSocket = new
ServerSocket(intPortNumber,intMaxClients);
while (true) {
vecConnectionSockets.addElement(new
ThreadedConnectionSocket(welcomeSocket.accept()));
Thread.yield();
```

```
}
} catch (IOException ioe) {System.out.println(ioe);}
}
public static String showDialog () {
FileDialog fd = new FileDialog(new Frame(),"Select
File...",FileDialog.LOAD);
fd.show();
return fd.getDirectory()+fd.getFile();
}
private class buttonListener implements ActionListener {
public void actionPerformed (ActionEvent ae) {
byte[] arrByteOfSentFile = null;
if (ae.getSource() == btnBrowse) {
strFilePath = showDialog();
tfFile.setText(strFilePath);
int intIndex = strFilePath.lastIndexOf("\\");
strFileName = strFilePath.substring(intIndex+1);
}
if (ae.getSource() == btnSend) {
try {
FileInputStream inFromHardDisk = new FileInputStream (strFilePath);
int size = inFromHardDisk.available();
arrByteOfSentFile = new byte[size];
inFromHardDisk.read(arrByteOfSentFile,0,size);
for (int i=0;i<vecConnectionSockets.size();i++)
{
ThreadedConnectionSocket tempConnectionSocket =
(ThreadedConnectionSocket)vecConnectionSockets.elementAt(i);
tempConnectionSocket.outToClient.writeObject("IsFileTransfered");
tempConnectionSocket.outToClient.flush();
tempConnectionSocket.outToClient.writeObject(strFileName);
tempConnectionSocket.outToClient.flush();
tempConnectionSocket.outToClient.writeObject(arrByteOfSentFile);
tempConnectionSocket.outToClient.flush();
}
JOptionPane.showMessageDialog(null,"Ban da gui thanh cong file toi
Client","Xac nhan",JOptionPane.INFORMATION_MESSAGE);
} catch (Exception ex) {}
}
if (ae.getSource() == btnReset) {
tfFile.setText("");
}}}
```

```
class ThreadedConnectionSocket extends Thread {
public Socket connectionSocket;
public ObjectInputStream inFromClient;
public ObjectOutputStream outToClient;
public ThreadedConnectionSocket (Socket s) {
connectionSocket = s;
try {
outToClient = new
ObjectOutputStream(connectionSocket.getOutputStream());
outToClient.flush();
inFromClient = new
ObjectInputStream(connectionSocket.getInputStream( ));
} catch (Exception e) {System.out.println(e);}
start();
}
public void run () {
try {
int intFlag = 0;
String strFileName = "";
while (true) {
Object objRecieved = inFromClient.readObject();
switch (intFlag) {
case 0:
if (objRecieved.equals("IsFileTransfered")) {
intFlag++;
}
break;
case 1:
strFileName = (String) objRecieved;
int intOption =
JOptionPane.showConfirmDialog(null,connectionSocket.getInetAddress()
.getHostName()+" dang gui "+strFileName+"!\nBan co chac chan nhan
khong?","Thong
bao",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
if (intOption == JOptionPane.YES_OPTION) {
intFlag++;
} else {
intFlag = 0;
}
break;
case 2:
byte[] arrByteOfReceivedFile = (byte[])objRecieved;
```

```
FileOutputStream outToHardDisk = new FileOutputStream(strFileName);
outToHardDisk.write(arrByteOfReceivedFile);
intFlag = 0;
JOptionPane.showMessageDialog(null,"Ban da nhan thanh cong file tu
Client","Xac nhan",JOptionPane.INFORMATION_MESSAGE);
break;
}
Thread.yield();
}
} catch (Exception e) {System.out.println(e);}
}} }
```

2. Mã nguồn phía Clients

```
//FiletransferClient
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class FileTransferClient extends Frame {
public static String strHostAddress = "";
public static int intPortNumber = 0, intMaxClients = 0;
public static Vector vecConnectionSockets = null;
public static FileTransferClient objFileTransfer;
public static String strFileName = "",strFilePath = "";
public static Socket clientSocket = null;
public static ObjectOutputStream outToServer = null;
public static ObjectInputStream inFromServer = null;
public static void main (String [] args) throws IOException {
BufferedReader stdin = new BufferedReader(new InputStreamReader
(System.in));
System.out.print("Nhap dia chi cua may server de ket noi: ");
System.out.flush();
strHostAddress = stdin.readLine();
System.out.print("Nhap dia chi cong de ket noi voi may server: ");
System.out.flush();
intPortNumber = Integer.parseInt(stdin.readLine());
objFileTransfer = new FileTransferClient();
}
public Label lblSelectFile;
public Label lblTitle;
```

```
public Label lblStudentName;
public Label lblStudentClass;
public TextField tfFile;
public Button btnBrowse;
public Button btnSend;
public Button btnReset;
public FileTransferClient () {
setTitle("Chuong trinh truyen File phia Client");
setSize(400 , 300);
setLayout(null);
addWindowListener(new WindowAdapter () { public void windowClosing
(WindowEvent e) { System.exit(0); } } );
lblTitle = new Label("Chuong trinh truyen File may Client ");
add(lblTitle);
lblTitle.setBounds(50,30,450,50);
lblSelectFile = new Label("Duong dan file can truyen:");
add(lblSelectFile);
lblSelectFile.setBounds(50,100,200,20);
lblStudentName = new Label("Sinh vien thuc hien:Pham Hong Thu");
add(lblStudentName);
lblStudentName.setBounds(130,250,200,20);
lblStudentClass = new Label("Lop : CT902");
add(lblStudentClass);
lblStudentClass.setBounds(130,270,100,20);
tfFile = new TextField("");
add(tfFile);
tfFile.setBounds(50,134,200,20);
btnBrowse = new Button("Chon File");
btnBrowse.addActionListener(new buttonListener());
add(btnBrowse);
btnBrowse.setBounds(283,133,70,20);
btnSend = new Button("Gui");
btnSend.addActionListener(new buttonListener());
add(btnSend);
btnSend.setBounds(100,200,50,20);
btnReset = new Button("Xoa");
btnReset.addActionListener(new buttonListener());
add(btnReset);
btnReset.setBounds(170,200,50,20);
show();

try {
```

```
clientSocket = new Socket (strHostAddress,intPortNumber);
outToServer = new
ObjectOutputStream(clientSocket.getOutputStream());
outToServer.flush();
inFromServer = new ObjectInputStream(clientSocket.getInputStream());
int intFlag = 0;
while (true) {
Object objRecieved = inFromServer.readObject();
switch (intFlag) {
case 0:
if (objRecieved.equals("IsFileTransfered")) {
intFlag++;
}
break;
case 1:
strFileName = (String) objRecieved;
int intOption =
JOptionPane.showConfirmDialog(this,clientSocket.getInetAddress().get
HostName()+" dang gui "+strFileName+"!\nBan co chac chan nhan
khong?","Thong
bao",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);//xac
nhan
if (intOption == JOptionPane.YES_OPTION) {
intFlag++;
} else {
intFlag = 0;
}
break;
case 2:
byte[] arrByteOfReceivedFile = (byte[])objRecieved;
FileOutputStream outToHardDisk = new FileOutputStream(strFileName);
outToHardDisk.write(arrByteOfReceivedFile);
intFlag = 0;
JOptionPane.showMessageDialog(this,"Ban dong y nhan file nay tu
Server","Thong bao",JOptionPane.INFORMATION_MESSAGE);//file dc
nhan;su chung thuc, su xac thuc
break;
}
Thread.yield();
}
} catch (Exception e) {System.out.println(e);}
}
```

```
public static String showDialog () {
FileDialog fd = new FileDialog(new Frame(),"Select
File...",FileDialog.LOAD);
fd.show();
return fd.getDirectory()+fd.getFile();
}
private class buttonListener implements ActionListener {
public void actionPerformed (ActionEvent ae) {
byte[] arrByteOfSentFile = null;
if (ae.getSource() == btnBrowse) {
strFilePath = showDialog();
tfFile.setText(strFilePath);
int intIndex = strFilePath.lastIndexOf("\\");
strFileName = strFilePath.substring(intIndex+1);
}
if (ae.getSource() == btnSend) {
try {
FileInputStream inFromHardDisk = new FileInputStream (strFilePath);
int size = inFromHardDisk.available();
arrByteOfSentFile = new byte[size];
inFromHardDisk.read(arrByteOfSentFile,0,size);
outToServer.writeObject("IsFileTransfered");
outToServer.flush();
outToServer.writeObject(strFileName);
outToServer.flush();
outToServer.writeObject(arrByteOfSentFile);
outToServer.flush();
JOptionPane.showMessageDialog(null,"Ban da gui thanh cong file toi
Server","Xac nhan",JOptionPane.INFORMATION_MESSAGE);
} catch (Exception ex) {}
}
if (ae.getSource() == btnReset) {
tfFile.setText("");
}}}
}
class ThreadedConnectionSocket extends Thread {
public Socket connectionSocket;
public ObjectInputStream inFromClient;
public ObjectOutputStream outToClient;
public ThreadedConnectionSocket (Socket s) {
connectionSocket = s;
try {
```

```
outToClient = new
ObjectOutputStream(connectionSocket.getOutputStream());
outToClient.flush();
inFromClient = new
ObjectInputStream(connectionSocket.getInputStream( ));
} catch (Exception e) {System.out.println(e);}
start();
}
public void run () {
try {
int intFlag = 0;
String strFileName = "";
while (true) {
Object objRecieved = inFromClient.readObject();
switch (intFlag) {
case 0:
if (objRecieved.equals("IsFileTransfered")) {
intFlag++;
}
break;
case 1:
strFileName = (String) objRecieved;
int intOption =
JOptionPane.showConfirmDialog(null,connectionSocket.getInetAddress()
.getHostName()+" dang gui "+strFileName+"!\nBan co chac chan nhan
khong?","Thong
bao",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE);
if (intOption == JOptionPane.YES_OPTION) {
intFlag++;
} else {
intFlag = 0;
}
break;
case 2:
byte[] arrByteOfReceivedFile = (byte[])objRecieved;
FileOutputStream outToHardDisk = new FileOutputStream(strFileName);
outToHardDisk.write(arrByteOfReceivedFile);
intFlag = 0;
JOptionPane.showMessageDialog(null,"Ban da gui thanh cong file toi
Server","Xac nhan",JOptionPane.INFORMATION_MESSAGE);
break;
}
```

```
Thread.yield();
}
} catch (Exception e) {System.out.println(e);}
}
}
```