TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN MẠNG VÀ TRUYỀN THÔNG

ĐỒ ÁN MÔN HỌC
LẬP TRÌNH MẠNG MÁY TÍNH

ĐỀ TÀI
Sử dụng Socket trong Java xây dựng chương trình CHAT Room theo mô hình Client-Server

GVHD : TRẦN HỒ THỦY TIÊN
Sinh viên : Trần Anh Tuấn
Lớp : 06T3
Nhóm : 09B

Đà Nẵng 2011

MỤC LỤC

DANH MỤC HÌNH

TỔNG QUAN VỀ ĐỀ TÀI

1. Bối cảnh và lý do thực hiện đề tài

Ngày nay, nhu cầu về công nghệ thông tin trong đời sống là đa dạng. Việc mở rộng các hệ thống truyền thông và ngày có nhiều máy vi tính kết nối vào mạng Internet. Với việc ứng dụng giao thức TCP/IP làm cho hệ thống mạng ngày càng rộng hơn và phát triển vượt bật. Vấn đề đề an ninh, bảo mật,…là một thế mạnh của giao thức này đem lại cho công nghệ truyền thông.

Bên cạnh việc phát triển của thư điện tử bằng nhiều dịch vụ khác nhau(gmail, yahoo,MSM trong mạng điện thoại di động…), việc CHAT trực tuyến và gửi flie trực tiếp trên Internet cũng là nhu cầu không thể thiếu.

Với thực tế như vậy, nhóm chúng em đã nghiên cứu và xây dựng mô hình CHAT và truyền file trong mạng LAN.

2. Phương pháp triển khai đề tài

Ngôn ngữ lập trình :

Java

Chương trình soạn thảo và build :

NetBeans IDE

Sử dụng

Giao thức TCP/IP

Sử dụng Socket trong Java

3. Kết cấu của đồ án

Gồm ba chương:

Chương 1: Cơ sở lý thuyết

Chương 2: Thiết kế và xây dựng hệ thống

Chương 3: Triển khai và đánh giá kết quả

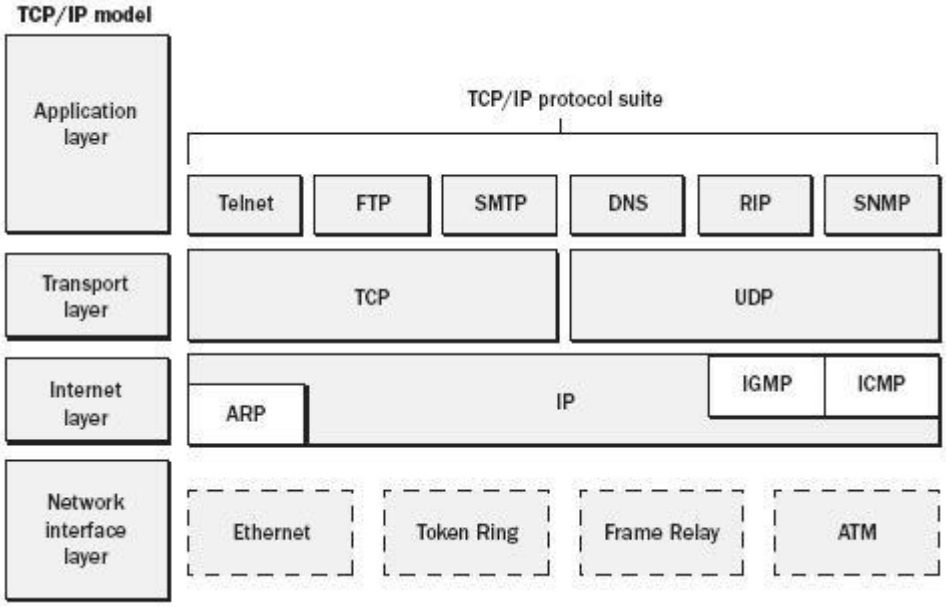
Chương 1. CƠ SỞ LÝ THUYẾT

1.1. Giao thức TCP/IP

TCP/IP là tên chung cho một tập hợp hơn 100 giao thức được sử dụng để kết nối các máy tính vào mạng, trong đó hai giao thức chính là TCP (Transmission Control Protocol) và IP (Internet Protocol).

Trong phạm vi Internet, thông tin không được truyền tải như một dòng riêng biệt từ máy tính này tới máy tính khác. Thay vào đó, dữ liệu được chia thành những gói nhỏ gọi là packet.

Các packet này được gửi trên mạng máy tính. Công việc của IP là chuyển chúng đến các máy tính ở xa. Tại trạm cuối, TCP nhận các packet và kiểm tra lỗi. Nếu một lỗi xuất hiện, TCP yêu cầu gói riêng biệt đó phải được gửi lại. Chỉ khi tất cả các packet đã nhận được là đúng, TCP sẽ sử dụng số thứ tự để tạo lại thông tin ban đầu.



Hình 1: Sơ đồ TCP/IP

1.1.1. Giao thức IP(Internet Protocol - Giao thức Liên mạng)

Là một giao thức hướng dữ liệu được sử dụng bởi các máy chủ nguồn và đích để truyền dữ liệu trong một liên mạng chuyển mạch gói.

Dữ liệu trong một liên mạng IP được gửi theo các khối được gọi là các gói (packet hoặc datagram). Cụ thể, IP không cần thiết lập các đường truyền trước khi một máy chủ gửi các gói tin cho một máy khác mà trước đó nó chưa từng liên lạc với.

Giao thức IP cung cấp một dịch vụ gửi dữ liệu không đảm bảo (còn gọi là cố gắng cao nhất), nghĩa là nó hầu như không đảm bảo gì về gói dữ liệu.

Gói dữ liệu có thể đến nơi mà không còn nguyên vẹn, nó có thể đến không theo thứ tự (so với các gói khác được gửi giữa hai máy nguồn và đích đó), nó có thể bị trùng lặp hoặc bị mất hoàn toàn. Nếu một phần mềm ứng dụng cần được bảo đảm, nó có thể được cung cấp từ nơi khác, thường từ các giao thức giao vận nằm phía trên IP.

Các thiết bị định tuyến liên mạng chuyển tiếp các gói tin IP qua các mạng tầng liên kết dữ liệu được kết nối với nhau. Việc không có đảm bảo về gửi dữ liệu có nghĩa rằng các chuyển mạch gói có thiết kế đơn giản hơn. (Lưu ý rằng nếu mạng bỏ gói tin, làm đổi thứ tự hoặc làm hỏng nhiều gói tin, người dùng sẽ thấy hoạt động mạng trở nên kém đi. Hầu hết các thành phần của mạng đều cố gắng tránh để xảy ra tình trạng đó. Đó là lý do giao thức này còn được gọi là cố gắng cao nhất. Tuy nhiên, khi lỗi xảy ra không thường xuyên sẽ không có hiệu quả đủ xấu đến mức người dùng nhận thấy được.)

Giao thức IP rất thông dụng trong mạng Internet công cộng ngày nay. Giao thức tầng mạng thông dụng nhất ngày nay là IPv4; đây là giao thức IP phiên bản 4. IPv6 được đề nghị sẽ kế tiếp IPv4: Internet đang hết dần địa chỉ IPv4, do IPv4 sử dụng 32 bit để đánh địa chỉ (tạo được khoảng 4 tỷ địa chỉ); IPv6 dùng địa chỉ 128 bit, cung cấp tối đa khoảng 3.4×1038 địa chỉ . Các phiên bản từ 0 đến 3 hoặc bị hạn chế, hoặc không được sử dụng. Phiên bản 5 được dùng làm giao thức dòng (stream) thử nghiệm. Còn có các phiên bản khác, nhưng chúng thường dành là các giao thức thử nghiệm và không được sử dụng rộng rãi.

Địa chỉ IP được chia thành 4 số giới hạn từ 0 - 255. Mỗi số được lưu bởi 1 byte - > IP có kích thước là 4byte, được chia thành các lớp địa chỉ. Có 3 lớp là A, B, và C. Nếu ở lớp A, ta sẽ có thể có 16 triệu địa chỉ, ở lớp B có 65536 địa chỉ. Ví dụ: Ở lớp B chúng ta có tất cả các địa chỉ từ 132.25.0.0 đến 132.25.255.255. Phần lớn các địa chỉ ở lớp A là sở hữu của các công ty hay của tổ chức. Một ISP thường sở hữu một vài địa chỉ lớp B hoặc C. Ví dụ: Nếu địa chỉ IP của bạn là 132.25.23.24 thì bạn có thể xác định ISP của bạn là ai. (có IP là 132.25.x.)

+	Bit 0 - 3	4 - 7	8 - 9	10 - 15	16 - 31
0	Source address				
32	Destination address				
64	Zeros		Protocol		TCP length
96	Source Port				Destination Port
128	Sequence Number				
160	Acknowledgement Number				
192	Data Offset	Reserved		Flags	Window
225	Checksum				Urgent Pointer
257	Options (optional)				
257/289+	Data				

Hình 2 : Cấu trúc Header của IP

1.1.2. Giao thức TCP(Transmission Control Protocol - "Giao thức điều khiển truyền vận")

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

TCP hỗ trợ nhiều giao thức ứng dụng phổ biến nhất trên Internet và các ứng dụng kết quả, trong đó có WWW, thư điện tử và Secure Shell.

Trong bộ giao thức TCP/IP, TCP là tầng trung gian giữa giao thức IP bên dưới và một ứng dụng bên trên. Các ứng dụng thường cần các kết nối đáng tin cậy kiểu đường ống để liên lạc với nhau, trong khi đó, giao thức IP không cung cấp những dòng kiểu đó, mà chỉ cung cấp dịch vụ chuyển gói tin không đáng tin cậy. TCP làm nhiệm vụ của tầng giao vận trong mô hình OSI đơn giản của các mạng máy tính.

Các ứng dụng gửi các dòng gồm các byte 8-bit tới TCP để chuyển qua mạng. TCP phân chia dòng byte này thành các đoạn (segment) có kích thước thích hợp (thường được quyết định dựa theo kích thước của đơn vị truyền dẫn tối đa (MTU) của tầng liên kết dữ liệu của mạng mà máy tính đang nằm trong đó). Sau đó, TCP chuyển các gói tin thu được tới giao thức IP để gửi nó qua một liên mạng tới mô đun TCP tại máy tính đích. TCP kiểm tra để đảm bảo không có gói tin nào bị thất lạc bằng cách gán cho mỗi gói tin một "số thứ tự" (sequence number). Số thứ tự này còn được sử dụng để đảm bảo dữ liệu được trao cho ứng dụng đích theo đúng thứ tự. Mô đun TCP tại đầu kia gửi lại "tin

báo nhận" (acknowledgement) cho các gói tin đã nhận được thành công; một "đồng hồ" (timer) tại nơi gửi sẽ báo time-out nếu không nhận được tin báo nhận trong khoảng thời gian bằng một round-trip time (RTT), và dữ liệu (được coi là bị thất lạc) sẽ được gửi lại. TCP sử dụng checksum (giá trị kiểm tra) để xem có byte nào bị hỏng trong quá trình truyền hay không; giá trị này được tính toán cho mỗi khối dữ liệu tại nơi gửi trước khi nó được gửi, và được kiểm tra tại nơi nhận.

+	Bit 0 - 3	4 - 9	10 - 15	16 - 31
0	Source Port			Destination Port
32	Sequence Number			
64	Acknowledgement Number			
96	Data Offset	Reserved	Flags	Window
128	Checksum			Urgent Pointer
160	Options (optional)			
160/192+	Data			

Hình 3 : Cấu trúc header cua TCP

1.2. Mô hình Client/Server

Mô hình được phổ biến nhất và được chấp nhận rộng rãi trong các hệ thống phân tán là mô hình client/server. Trong mô hình này sẽ có một tập các tiến trình mà mỗi tiến trình đóng vai trò như là một trình quản lý tài nguyên cho một tập hợp các tài nguyên cho trước và một tập hợp các tiến trình client trong đó mỗi tiến trình thực hiện một tác vụ nào đó cần truy xuất tới tài nguyên phần cứng hoặc phần mềm dùng chung. Bản thân các trình quản lý tài nguyên cần phải truy xuất tới các tài nguyên dùng chung được quản lý bởi một tiến trình khác, vì vậy một số tiến trình vừa là tiến trình client vừa là tiến trình server. Các tiến trình phát ra các yêu cầu tới các server bất kỳ khi nào chúng cần truy xuất tới một trong các tài nguyên của các server. Nếu yêu cầu là đúng đắn thì server sẽ thực hiện hành động được yêu cầu và gửi một đáp ứng trả lời tới tiến trình client.

Mô hình client/server cung cấp một cách tiếp cận tổng quát để chia sẻ tài nguyên trong các hệ thống phân tán. Mô hình này có thể được cài đặt bằng rất nhiều môi trường phần cứng và phần mềm khác nhau. Các máy tính được sử dụng để chạy các tiến trình client/server có nhiều kiểu khác nhau và không cần thiết phải phân biệt giữa chúng; cả tiến trình client và tiến trình server đều có thể chạy trên cùng một máy tính. Một tiến trình server có thể sử dụng dịch vụ của một server khác.

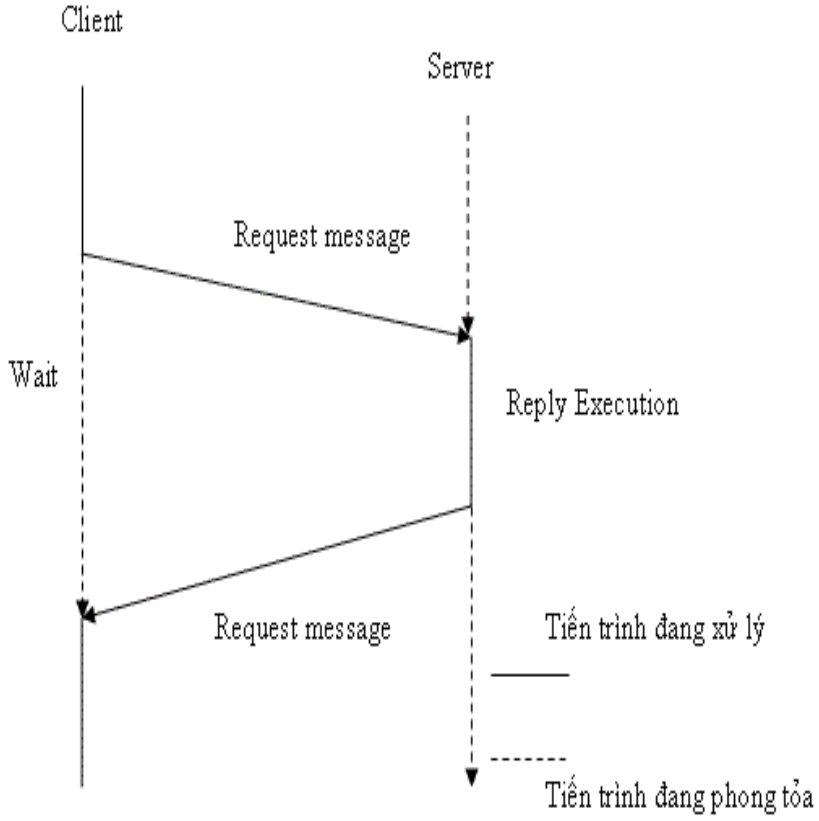
Mô hình truyền tin client/server hướng tới việc cung cấp dịch vụ. Quá trình trao đổi dữ liệu bao gồm:

1. Truyền một yêu cầu từ tiến trình client tới tiến trình server

2. Yêu cầu được server xử lý
3. Truyền đáp ứng cho client

Mô hình truyền tin này liên quan đến việc truyền hai thông điệp và một dạng đồng bộ hóa cụ thể giữa client và server. Tiến trình server phải nhận thức được thông điệp được yêu cầu ở bước một ngay khi nó đến và hành động phát ra yêu cầu trong client phải được tạm dừng (bị phong tỏa) và buộc tiến trình client ở trạng thái chờ cho tớ khi nó nhận được đáp ứng do server gửi về ở bước ba.

Mô hình client/server thường được cài đặt dựa trên các thao tác cơ bản là gửi (send) và nhận (receive)



Hình 4 : Mô hình client/server

Quá trình giao tiếp client và server có thể diễn ra theo một trong hai chế độ: bị phong tỏa (blocked) và không bị phong tỏa (non-blocked).

Chế độ bị phong tỏa (blocked):

Trong chế độ bị phong tỏa, khi tiến trình client hoặc server phát ra lệnh gửi dữ liệu (send), việc thực thi của tiến trình sẽ bị tạm ngừng cho tới khi tiến trình nhận phát ra lệnh nhận dữ liệu (receive).

Tương tự đối với tiến trình nhận dữ liệu, nếu tiến trình nào đó (client hoặc server) phát ra lệnh nhận dữ liệu, mà tại thời điểm đó chưa có dữ liệu

gửi tới thì việc thực thi của tiến trình cũng sẽ bị tạm ngừng cho tới khi có dữ liệu gửi tới.

Chế độ không bị phong tỏa (non-blocked)

Trong chế độ này, khi tiến trình client hay server phát ra lệnh gửi dữ liệu thực sự, việc thực thi của tiến trình vẫn được tiến hành mà không quan tâm đến việc có tiến trình nào phát ra lệnh nhận dữ liệu đó hay không.

Tương tự cho trường hợp nhận dữ liệu, khi tiến trình phát ra lệnh nhận dữ liệu, nó sẽ nhận dữ liệu hiện có, việc thực thi của tiến trình vẫn được tiến hành mà không quan tâm đến việc có tiến trình nào phát ra lệnh gửi dữ liệu tiếp theo hay không.

1.3. Cơ chế Socket trong Java

1.3.1. Khái quát về Socket

Như chúng ta đã biết kết nối URLs và URL cung cấp cho chúng ta một cơ cấu để truy xuất vào các tài nguyên trên Internet ở một mức tương đối cao, nhưng đôi khi chương trình của chúng ta lại yêu cầu một giao tiếp ở tầng mạng mức thấp.Ví dụ khi chúng ta viết một ứng dụng client-server.

Trong một ứng dụng client-server thì phía server sẽ cung cấp một số dịch vụ, như: xử lí cơ sở dữ liệu, các yêu cầu bên phía client đưa ra, sau đó sẽ gửi lại cho phía client. Sự giao tiếp như vậy gọi là tin cậy bởi vì dữ liệu sẽ không bị mất mát, sai lệch trong quá trình truyền, server gửi cho client thông điệp gì thì phía client sẽ nhận được thông điệp nguyên như vậy. Giao thức TCP sẽ cung cấp cho chúng ta một cách thức truyền tin cậy. Để có thể nói chuyện được trên TCP thì chương trình client và chương trình server phải thiếp lập một đường truyền, và mỗi chương trình sẽ phải kết nối lại với socket là điểm cuối để kết nối, client và server muốn nói chuyện với nhau thì sẽ phải thông qua socket, mọi thông điệp sẽ phải đi qua socket. Chúng ta cứ mường tượng socket ở đây là một cái cửa mọi người muốn đi ra hay đi vào đều phải thông qua cái cửa này.

1.3.2. Cơ chế Socket

Một socket là một điểm cuối của thông tin hai chiều liên kết giữa hai chương trình đang chạy trên mạng. Những lớp socket được dùng để đại diện cho kết nối giữa một chương trình client và một chương trình server. Trong Java gói Java.net cung cấp hai lớp Socket và ServerSocket để thực hiện kết nối giữa client và server.

Thông thường thì server sẽ chạy trên một máy đặc biệt và có một socket giới hạn trong 1 Portnumber đặc biệt.

Phía client: client được biết hostname của máy mà server đang chạy và port number mà server đang lắng nghe. Để tạo một yêu cầu kết nối client sẽ

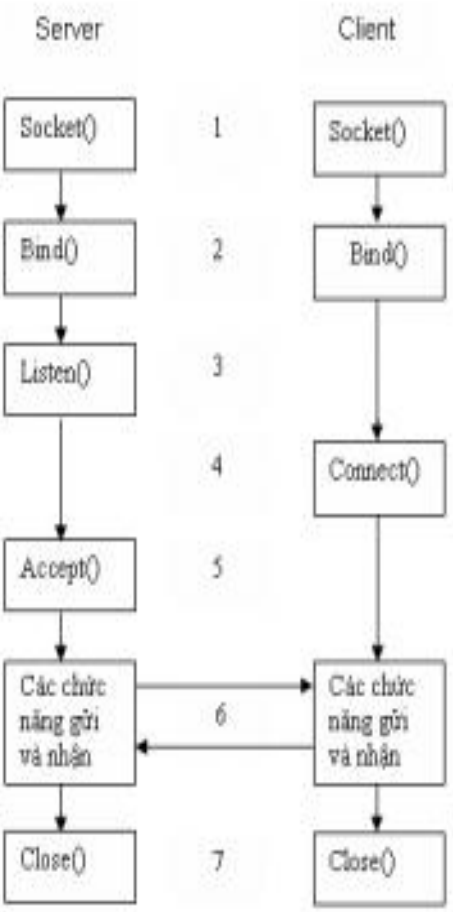
thử hẹn gặp server ở trên máy của server thông qua port number. Client cũng cần xác định chính nó với server thông qua local port number.

Hình 5 : Client gửi yêu cầu kết nối tới Server

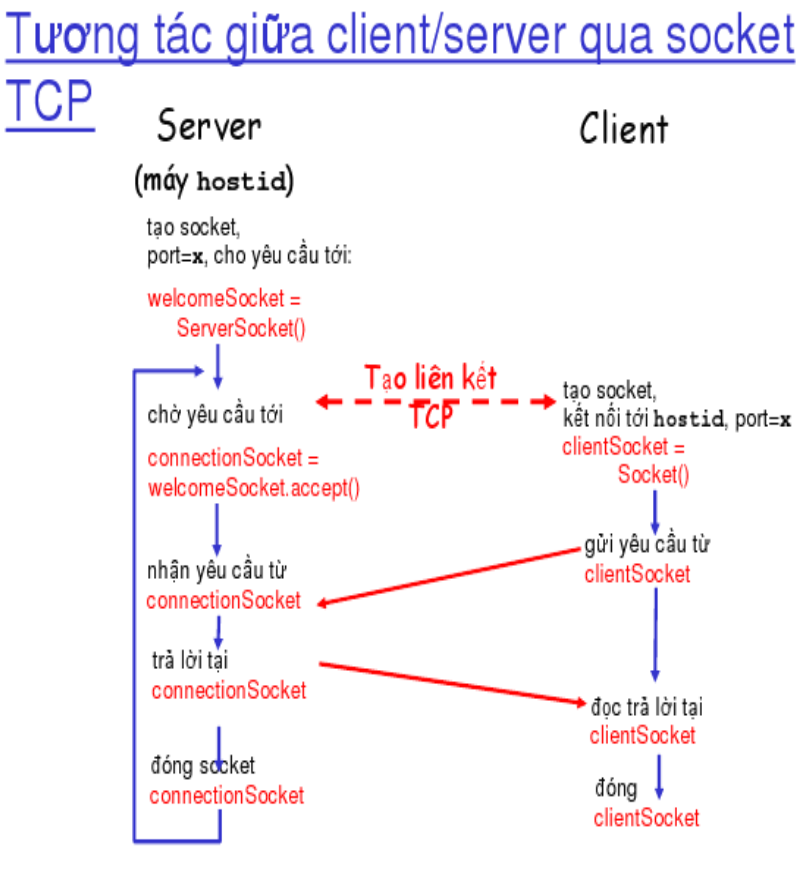
Nếu mọi thứ tốt đẹp thì server sẽ đồng ý kết nối. khi đồng ý kết nối thì server sẽ tạo ra một socket mới để nói chuyện với client và cũng tạo ra một socket khác để tiếp tục lắng nghe.

Hình 6 : Server đồng ý kết nối và tiếp tục lắng nghe.

1.3.3. Mô hình truyền tin socket



Hình 7 : Mô hình truyền tin socket



Hình 8 : Mô hình tương tác giữa client/server qua socket TCP

Một socket có thể thực hiện bảy thao tác cơ bản:
Kết nối với một máy ở xa (ví dụ, chuẩn bị để gửi và nhận dữ liệu)
Gửi dữ liệu
Nhận dữ liệu
Ngắt liên kết
Gán cổng
Nghe dữ liệu đến
Chấp nhận liên kết từ các máy ở xa trên cổng đã được gán

Lớp Socket của Java được sử dụng bởi cả client và server, có các phương thức tương ứng với bốn thao tác đầu tiên. Ba thao tác cuối chỉ cần cho server để chờ các client liên kết với chúng. Các thao tác này được cài đặt bởi lớp ServerSocket. Các socket cho client thường được sử dụng theo mô hình sau:

Một socket mới được tạo ra bằng cách sử dụng hàm Socket().
Socket cố gắng liên kết với một host ở xa.

Mỗi khi liên kết được thiết lập, các host ở xa nhận các luồng vào và luồng ra từ socket, và sử dụng các luồng này để gửi dữ liệu cho nhau. Kiểu liên kết này được gọi là song công (full-duplex)-các host có thể nhận và gửi dữ liệu đồng thời. Ý nghĩa của dữ liệu phụ thuộc vào giao thức.

Khi việc truyền dữ liệu hoàn thành, một hoặc cả hai phía ngắt liên kết. Một số giao thức, như HTTP, đòi hỏi mỗi liên kết phải bị đóng sau mỗi khi yêu cầu được phục vụ. Các giao thức khác, chẳng hạn FTP, cho phép nhiều yêu cầu được xử lý trong một liên kết đơn.

1.3.4. Một số hàm cơ bản trong socket

Class mô tả về socket

- Tạo một socket

Socket(InetAddress address, int port)
Socket(String host, int port)
Socket(InetAddress address, int port, InetAddress, localAddr, int localPort)
Socket(String host, int port, InetAddress, localAddr, int localPort)
Socket()
–

- Lấy thông tin về một socket

InetAddress getInetAddress() : trả về địa chỉ mà socket kết nối đến
int getPort() : trả về địa chỉ mà socket kết nối đến.
InetAddress getLocalAddress() : trả về địa chỉ cục bộ.
int getLocalPort() : trả về địa chỉ cục bộ.

- Sử dụng Streams

public OutputStream getOutputStream() throws IOException: Trả về một output stream cho việc viết các byte đến socket này.

public InputStream getInputStream() throws IOException : Trả về một input stream cho việc đọc các byte từ socket này.

ServerSocket class

- Class mô tả ServerSocket

Tạo một ServerSocket

ServerSocket(intport) throws IOException ServerSocket(intport, intbacklog) throws IOException ServerSocket(intport, intbacklog, InetAddressbindAddr) throws IOException

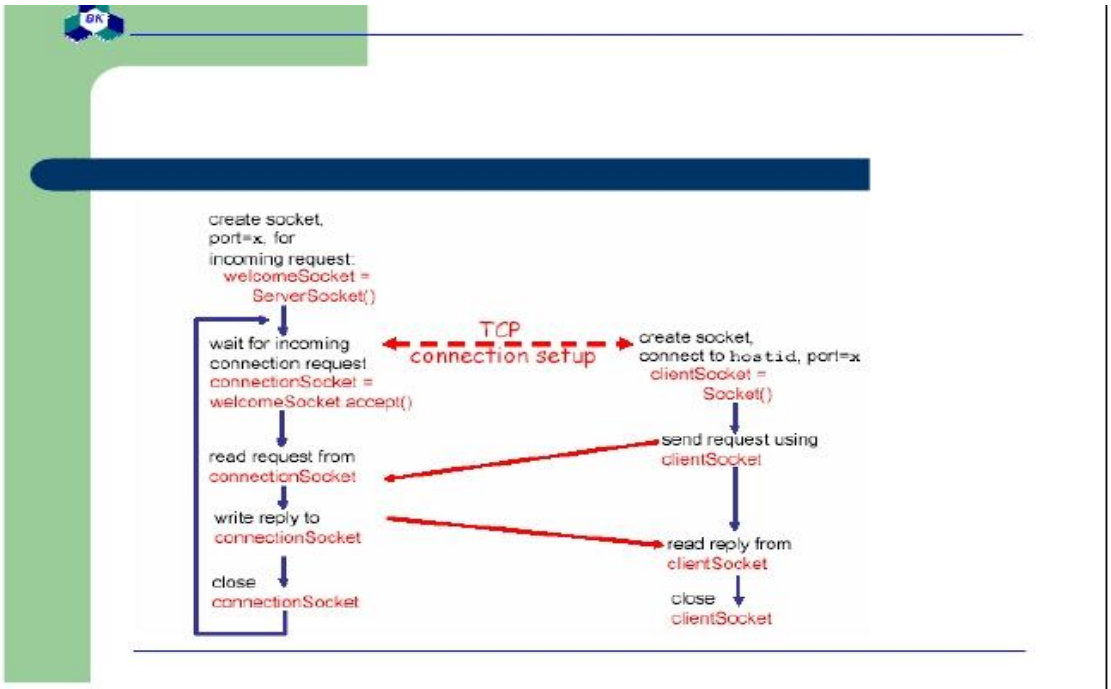
- Các phương thức trong ServerSocket

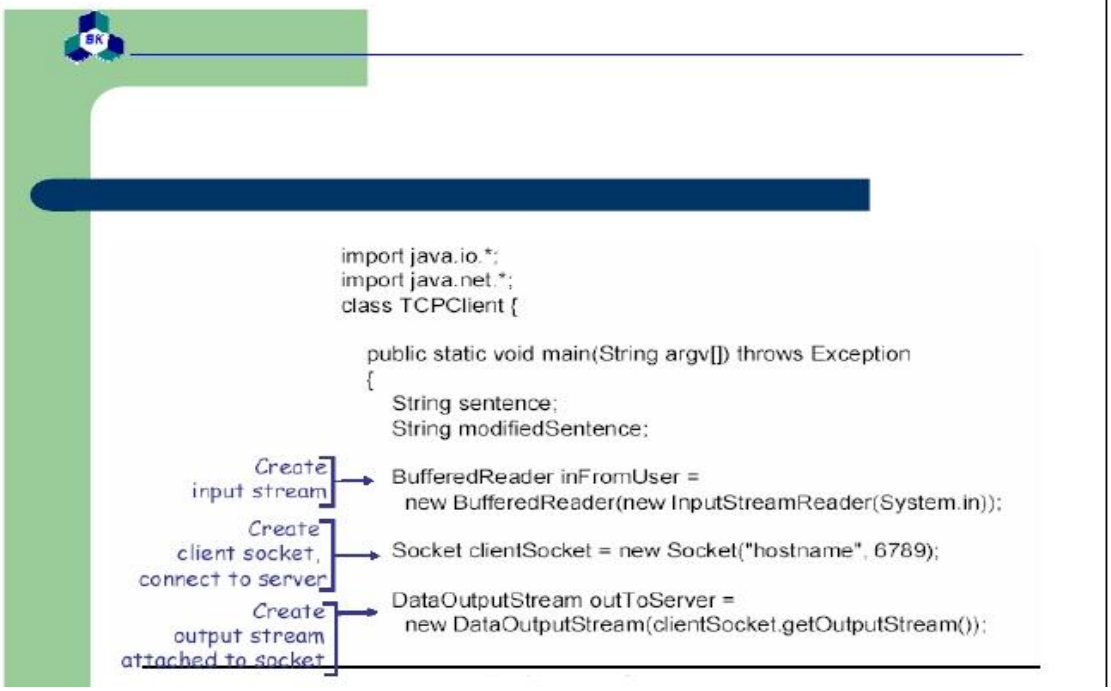
Socket accept() throws IOException: lắng nghe một kết nối đến socket này và có chấp nhận nó hay không

void close() throws IOException: Đóngsocket.

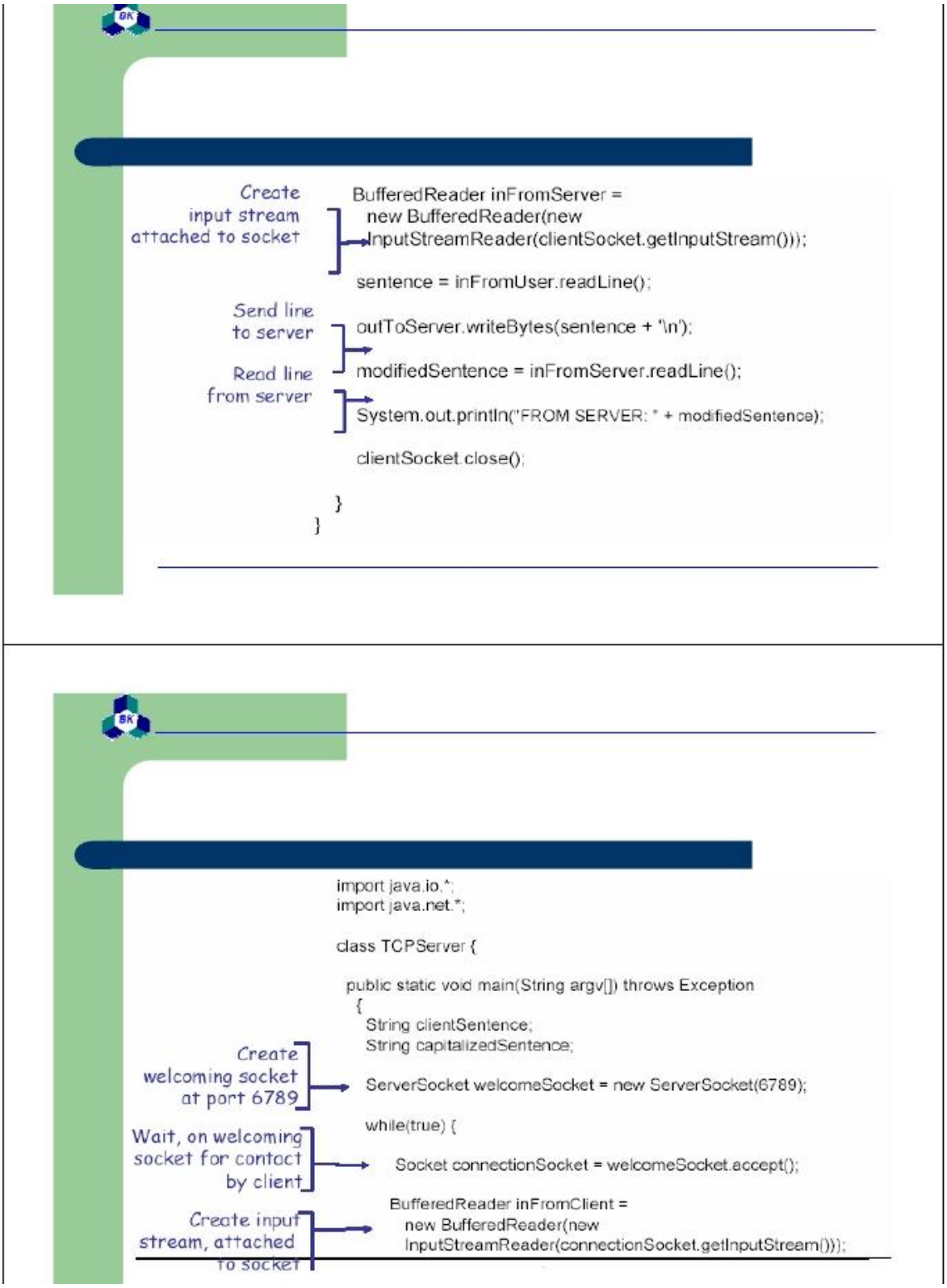
InetAddress getInetAddress() : trả về địa chỉ cục bộ của socket

int getLocalPort() : trả về port mà server đang lắng nghe





Hình 9 : Ví dụ về một chương trình socket TCP



Hình 10 : Ví dụ về một chương trình socket TCP

Chương 2. THIẾT KẾ VÀ XÂY DỰNG HỆ THỐNG

2.1. Phân tích yêu cầu

- Yêu cầu của bài toán: Xây dựng mô hình Client – Server ứng dụng CHAT
- Xây dựng chương trình Server

Tạo một TCP Socket và gắn vào một cổng

Xây dựng một chương trình Server đa tuyến (Threaded Server) để cho phép nhiều Client kết nối tới Server.Mỗi tuyến đảm nhận liên lạc với Client.

Chờ và lắng nghe yêu cầu kết nối từ Client

Chấp nhận kết nối và nhận Socket tương ứng.

Truyền nhận thông tin qua các luồng nhận/gửi dữ liệu của socket.

Khi một user login vào Server thì Server sẻ cập nhật user đó và gửi tới các Client đang kết nối

Khi một user logout ra khỏi Server thì Server hiện thông báo user đó đã logout và gửi tới Client.

Đóng kết nối.

- Xây dựng chương trình Client.

Tạo một TCP Socket với địa chỉ IP và số cổng mà chương trình Server đang chạy

Thiết lập kết nối tới Server

Trao đổi dữ liệu với Server.

Cập nhật các user khác vừa logint/logout.

Gửi/ nhận thông điệp tới tất cả mọi người có trong phòng chát

Đóng kết nối

Thiết kế giao diện hiển thị khung chat phía Client.

- Sử dụng giao thức TCP/IP
- Yêu cầu giao diện:

Vùng thao tác chuỗi(CHAT):

CHAT trực tiếp giữa Client và Server khi được kết nối.

Trao đổi dữ liệu với nhau nhờ vào Stream nhập và Stream xuất.

2.2. Phân tích các chức năng

Chức năng của chương trình được thể hiện qua chương trình Client và Server.

Chức năng Chat trong mạng LAN:

Chuỗi ký tự nhập từ bàn phím.

Ấn nút "Send" để gửi chuỗi ký tự vừa nhập để giao tiếp với nhau giữa Client và Server.

Xây dựng chương trình Client: nối kết đến Server trong mạng để trao đổi thông tin và truyền tải dữ liệu.

Xây dựng chương trình Server đa tuyến để kết nối với nhiều Client:

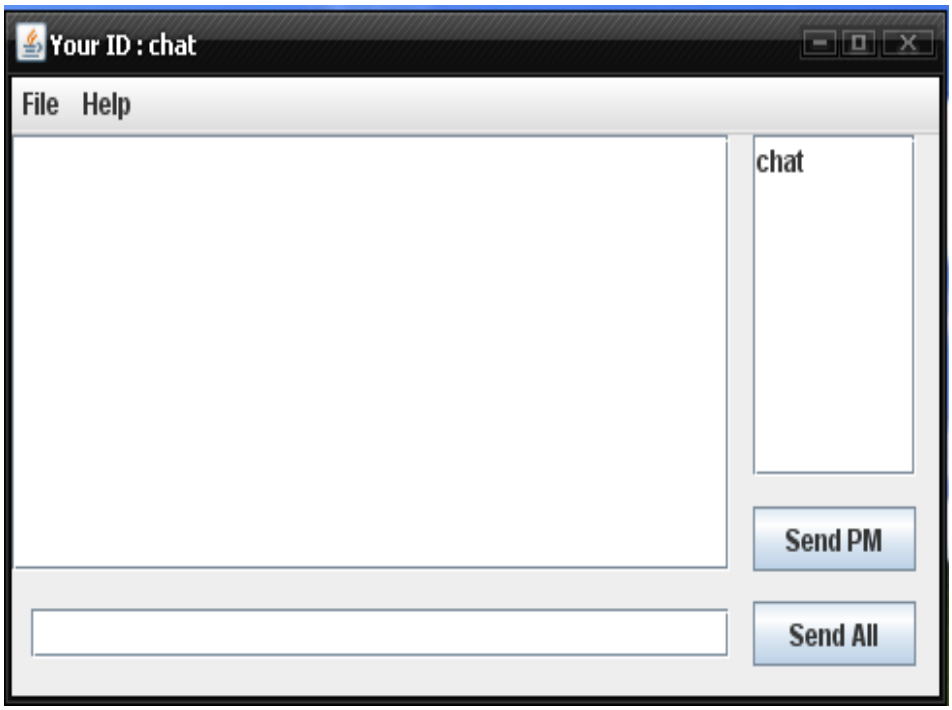
Xử lý các yêu cầu nối kết: lắng nghe yêu cầu nối kết của Client.

Xử lý các thông điệp yêu cầu từ Client: chấp nhận hoặc từ chối nối kêt.

2.3. Thiết kế kế chương trình

2.3.1. Thiết kế giao diện

Hình 11: Giao diện login



Hình 12: Giao diện Chat room

Server ip/host name: địa chỉ ip hay hostname của Server (mặc định là địa chỉ localhost).

Port : số hiệu cổng kết nối

Chat ID: nickname dùng để đăng nhập

Send All : thông tin Chat sẽ được gửi tới tất cả các user trong phòng Chat

Send PM: thông tin Chat sẽ được gửi tới user được chỉ định trong phòng Chat

File/Exit : thoát khỏi phòng Chat

2.3.2. Xây dựng các chức năng

Xây dựng chương trình Client(sử dụng lớp java.net.Socket):

Mở một socket nối kết đến Server đã biết địa chỉ IP/localhost và số hiệu cổng(9999).

Lấy Stream nhập và Stream xuất được gán với socket.

Trao đổi dữ liệu với Server nhờ vào các Stream nhập và Stream xuất.

Dùng giao thức TCP/IP để kiểm tra dữ liệu trao đổi với Server.

Xây dựng chương trình Server đa tuyến(Server phải luôn giữ kết nối)

Xử lý các yêu cầu nối kết:

Lắng nghe yêu cầu nối kết.

Chấp nhận một yêu cầu nối kết.

Tạo kênh giao tiếp ảo với các Client.

Xử lý các thông điệp yêu cầu từ Client.

Xử lý các thông điệp yêu cầu từ Client:

Chờ nhận thông điệp của các Client.

Phân tích và xử lý yêu cầu.

Gửi thông điệp trả lời cho Client.

Chương 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ

3.1. Môi trường triển khai

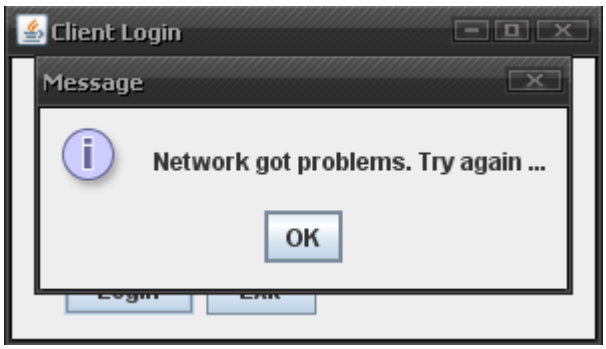
Hệ điều hành Microsoft Windowns 7 profesional, Microsoft Windowns. XP.

Eclipse , Netbeans

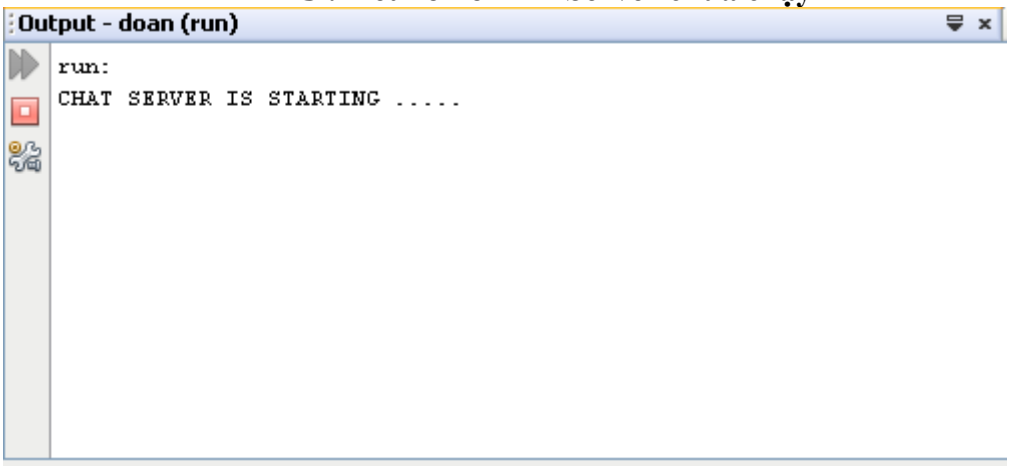
JDK 1.5

Mạng LAN

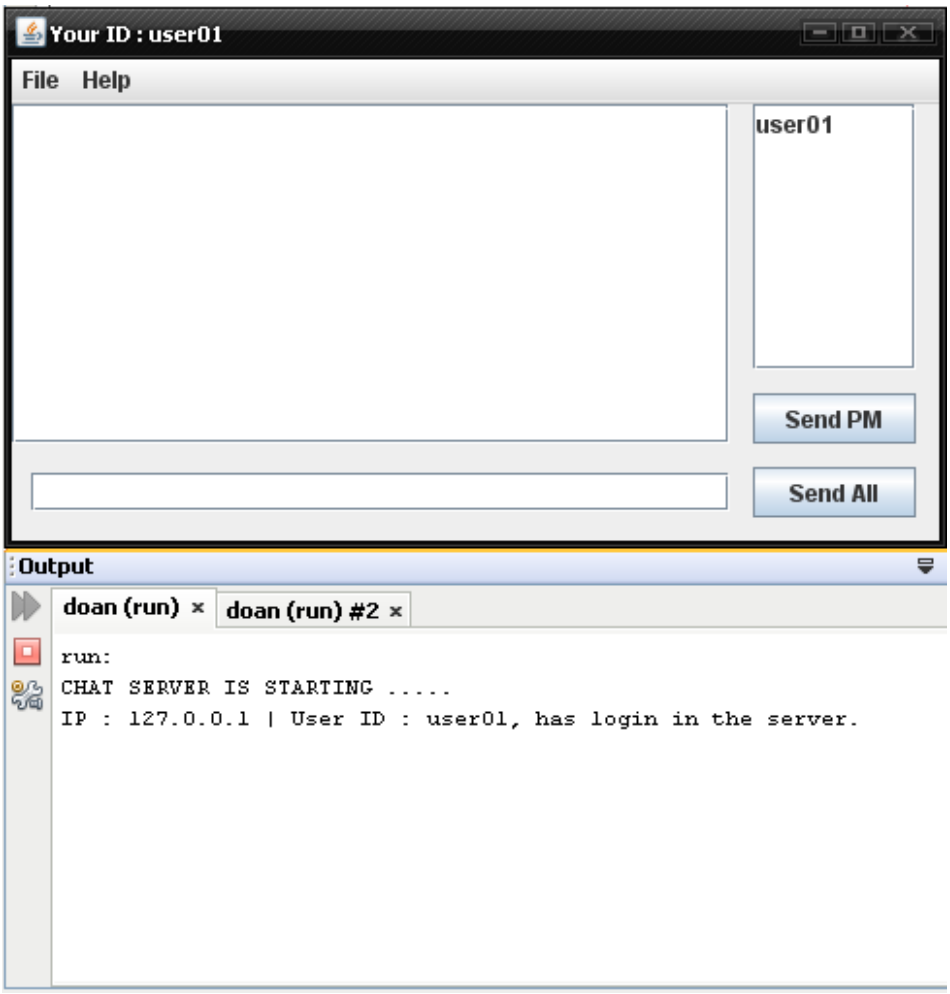
3.2. Kết quả các chức năng của chương trình



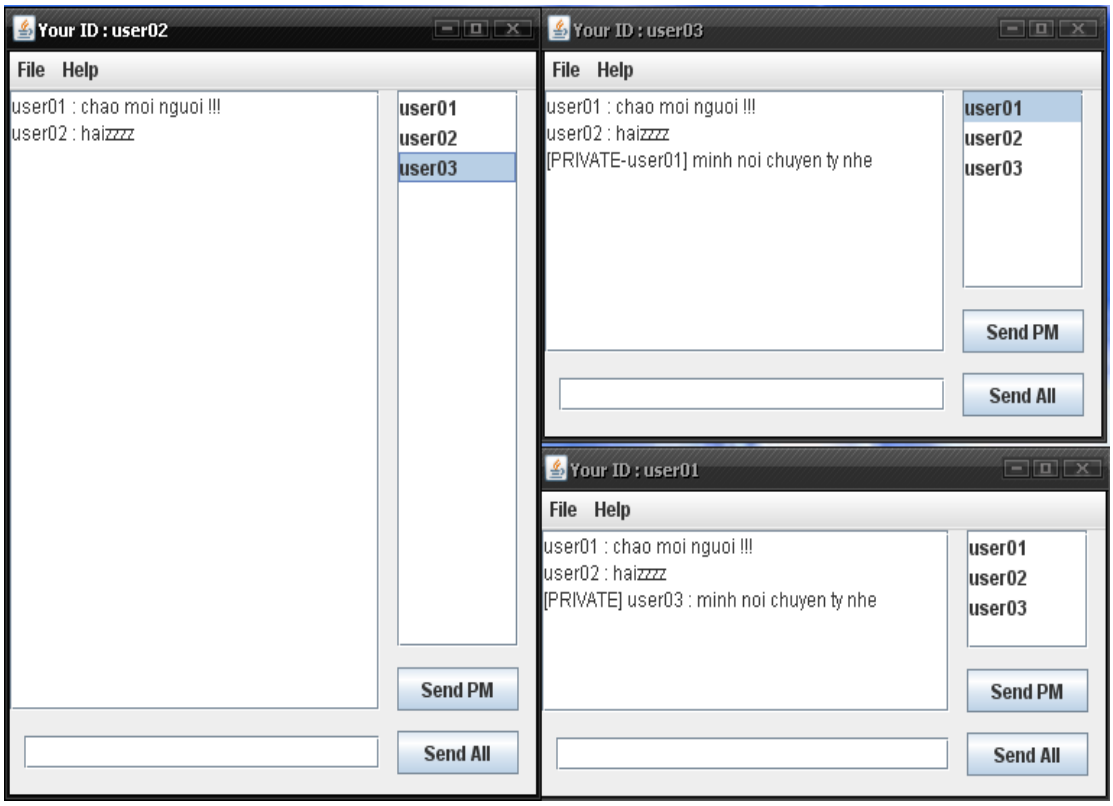
Hình 13 : Kết nối lỗi khi Server chưa chạy



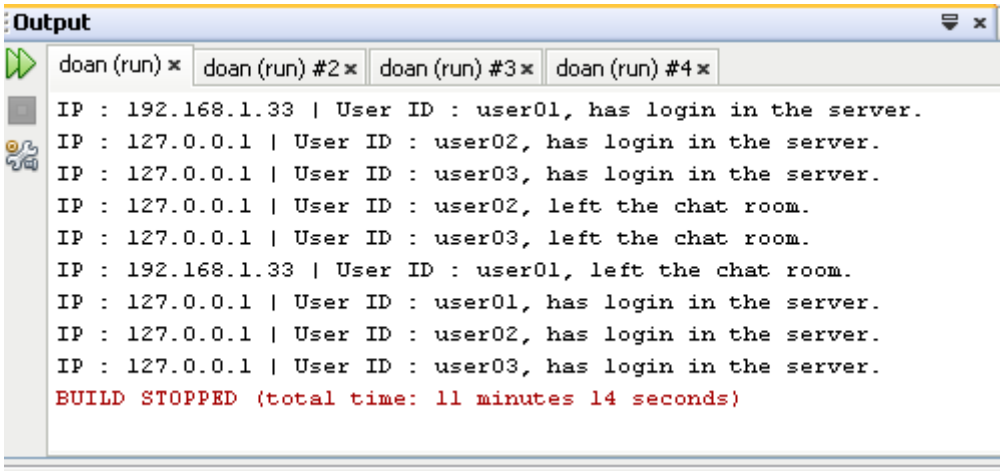
Hình 14 : Chạy chương trình server



Hình 15 : Login thành công vào phòng Chat



Hình 16 : Thử nghiệm Chat private



Hình 17 : Hiển thị thông báo login và logout trên Server

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Những kết quả đạt được

- Mô phỏng được mô hình Client-Server trong mạng LAN.
- Thực hiện được yêu cầu bài đề ra(CHAT ROOM).
- Dùng giao thức TCP/IP trong truyền dữ liệu.
- Thực hiện được việc kiểm tra trong nối kêt giữa Client-Server.

2. Những vấn đề tồn tại

- Chương trình Chat còn đơn giản
- Giao diện sơ sài
- Chỉ thực hiện được nối kết giữa Client-Server(theo mô hình).

3. Hướng phát triển

- Hoàn thiện giao diện đẹp và phù hợp với người dùng.
- Mở rộng ứng dụng trong chương trình Chat như :
  + Xử lý truyền file thông qua chương trình Chat
  + Xây dựng hệ cơ sở dữ liệu quản lý thông tin đăng nhập (username, password …)
- Phát triển trên mạng rộng ngoài LAN.

PHỤ LỤC

1. Chương trình Server:

1.1. Chatserver class:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package server;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.*;
import java.util.logging.Level;
import java.util.logging.Logger;
/**
 *
 * @author anhtuan
 */
public class chatserver implements Runnable{

   serverSocket sThread[] = new serverSocket[50];
   int maxClient = 0;

   public synchronized void send(String username,String message){
      if ( username.equals("send_to_all")) {
         for(int i=0; i<maxClient; i++){
            if (sThread[i].active){
               sThread[i].send(message);
            }
         }
      }
      else{
         for(int i=0; i<maxClient; i++){
            if
((sThread[i].active)&&(sThread[i].userID.equals(username))){
               sThread[i].send(message);
            }
```

```
            }
        }
    }

    public synchronized  void sendToOther(String username, String
message){
        for(int i=0; i<maxClient; i++){
            if
((sThread[i].active)&&(!sThread[i].userID.equals(username))){
                sThread[i].send(message);
            }
        }
    }

    public String getAliveHost(){
        String result = "";
        for ( int i=0; i<maxClient; i++) {
            if ( sThread[i].active ) {
                result += sThread[i].userID + "|";
            }
        }
        String myresult = "";
        for (int i =0; i<result.length()-1; i++) {
            myresult = myresult + result.charAt(i);
        }
        // System.out.println("Return : " + myresult);
        return myresult;
    }


    public void run() {
        System.out.println("CHAT SERVER IS STARTING ..... ");
        try {
            ServerSocket ss = new ServerSocket(9999);
            boolean result;
            while (true){
                Socket socket = ss.accept();
                DataInputStream input = new
DataInputStream(socket.getInputStream());
                DataOutputStream output= new
DataOutputStream(socket.getOutputStream());
                String inputString = input.readUTF();
```

```
                //
                // Command :
                // request_login|username
                // send_all|message
                // send_pm|username|message
                // logout|username
                // list_request
                String st[] = inputString.split("\\|");
                result = true;
                if ((st[0].equals("request_login"))&&maxClient>0) {
                   for ( int i =0; i< maxClient; i++ ){
                      if ( sThread[i].userID.equals(st[1]) )  {
                         result = false;
                         break;
                      }
                   }
                }
                if (result){
                   output.writeUTF("OK");
                   sThread[maxClient] = new serverSocket(this,socket,st[1]);
                   sThread[maxClient].start();
                   maxClient++;
                   System.out.println("IP :
"+socket.getInetAddress().getHostAddress() + " | User ID : " + st[1] + ", has
login in the server.");
                   sendToOther(st[1], "updateList|");
                }
                else
                {
                   output.writeUTF("Duplicated ID");
                   System.out.println("IP :
"+socket.getInetAddress().getHostAddress() + " | User ID : " + st[1] + ",
cannot login as duplicated of username.");
                }
             }
          }
          catch (IOException ex) {

Logger.getLogger(chatserver.class.getName()).log(Level.SEVERE, null, ex);
          }
       }
```

```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
          new server.chatserver().run();
    }


}
```

1.2. serverSocket class:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package server;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.SocketException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author anhtuan
 */
public class serverSocket extends Thread{

    Socket socket = null;
    String userID = null;
    boolean active = true;
    chatserver cs = null;


    public serverSocket(chatserver _cs, Socket incoming, String userID){
        this.socket = incoming;
        this.userID = userID;
        this.active = true;
```

```
            this.cs = _cs;
        }

        DataOutputStream output = null;

        public void send(String message){
            try {
                output = new DataOutputStream(socket.getOutputStream());
                output.writeUTF(message);
            } catch (IOException ex) {

Logger.getLogger(serverSocket.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }

        @Override
        public void run(){
            DataInputStream input = null;
            try {
                input = new DataInputStream(socket.getInputStream());
                output = new DataOutputStream(socket.getOutputStream());
                while ((true)&&(this.active)){
                    String inputString = input.readUTF();
                    String[] st = inputString.split("\\|"); //  Special character to
split - Hechay
                    // System.out.println(st[0]);
                    ////////// list_request ////////////////
                    if ( st[0].equals("list_request") ) {
                        String sendSt = cs.getAliveHost();
                        // System.out.println(sendSt);
                        output.writeUTF(sendSt);
                    }
                    ////////////////////////////////////////
                    if ( st[0].equals("send_all") ) {
                        cs.send("send_to_all", "normal|" + this.userID + " : " +st[1]
+ "\n");
                    }
                    ////////////////////////////////////////

                    ////////////////////////////////////////
                    if ( st[0].equals("send_pm") ) {
```

```
                    cs.send(st[1], "normal|" + "[PRIVATE] " + this.userID + " :
" + st[2] + "\n");
                }
                ////////////////////////////////////

                ////////////////////////////////////
                if ( st[0].equals("logout") ) {
                    this.active = false;
                    System.out.println("IP :
"+socket.getInetAddress().getHostAddress() + " | User ID : " + this.userID +
", left the chat room.");
                    this.userID = "Anhtuan-NO-USED-!@#$%^&*()_+";
                    cs.send("send_to_all", "updateList|");
                    break;
                }
                ////////////////////////////////////
            }
        } catch (IOException ex) {
            //
Logger.getLogger(serverThread.class.getName()).log(Level.SEVERE, null,
ex);
            System.out.println("Someone is disconnected but I haven't
received the logoff messsage");
        }


    }
}
```

2. Chương trình Client

2.1. Client class:

```
 /*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package client;
import java.io.*;
import java.net.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JList;
import javax.swing.JOptionPane;


/**
 *
 * @author anhtuan
 */
public class Client extends javax.swing.JFrame implements Runnable {

   Socket socket = null;
   String username = null;

   public void run() {
     DataOutputStream output = null;
      try {
         output = new DataOutputStream(socket.getOutputStream());
         DataInputStream input = new
DataInputStream(socket.getInputStream());
         String sendSt = "list_request";
         output.writeUTF(sendSt+"|");
         String st[] = new String[50];
         String receive = input.readUTF();
         st = receive.split("\\|");
         jList1.setListData(st);
          } catch (IOException ex) {
         Logger.getLogger(Client.class.getName()).log(Level.SEVERE,
null, ex);
             }
```

```
            }

        Thread thread = null;

        public Client(Socket socket, String username) {
           initComponents();
           this.setLocationRelativeTo(null);
           jList1.removeAll();
           this.socket = socket;
           this.username = username;
           new Thread(this).start();
           thread  = new Thread(new clientSocket(socket,jTextArea1,jList1));
           thread.start();
           this.setTitle("Your ID : " + username);
           jTextField2.requestFocus();
        }

        /** This method is called from within the constructor to
         * initialize the form.
         * WARNING: Do NOT modify this code. The content of this method
is
         * always regenerated by the Form Editor.
         */
        @SuppressWarnings("unchecked")
        // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
        private void initComponents() {

           jScrollPane1 = new javax.swing.JScrollPane();
           jList1 = new javax.swing.JList();
           jTextField2 = new javax.swing.JTextField();
           jButton1 = new javax.swing.JButton();
           jButton2 = new javax.swing.JButton();
           jScrollPane2 = new javax.swing.JScrollPane();
           jTextArea1 = new javax.swing.JTextArea();
           jMenuBar1 = new javax.swing.JMenuBar();
           jMenu1 = new javax.swing.JMenu();
           jMenuItem2 = new javax.swing.JMenuItem();
           jMenu2 = new javax.swing.JMenu();
           jMenuItem1 = new javax.swing.JMenuItem();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOS
E);
            addWindowListener(new java.awt.event.WindowAdapter() {
               public void windowClosing(java.awt.event.WindowEvent evt) {
                  formWindowClosing(evt);
               }
            });

            jList1.setModel(new javax.swing.AbstractListModel() {
               String[] strings = { "Item 1", "Item 2", "Item 3", "Item 4", "Item
5" };
               public int getSize() { return strings.length; }
               public Object getElementAt(int i) { return strings[i]; }
            });

jList1.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECT
ION);
            jScrollPane1.setViewportView(jList1);

            jTextField2.addActionListener(new
java.awt.event.ActionListener() {
               public void actionPerformed(java.awt.event.ActionEvent evt) {
                  jTextField2ActionPerformed(evt);
               }
            });
            jTextField2.addKeyListener(new java.awt.event.KeyAdapter() {
               public void keyPressed(java.awt.event.KeyEvent evt) {
                  jTextField2KeyPressed(evt);
               }
            });

            jButton1.setText("Send All");
            jButton1.addActionListener(new java.awt.event.ActionListener() {
               public void actionPerformed(java.awt.event.ActionEvent evt) {
                  jButton1ActionPerformed(evt);
               }
            });

            jButton2.setText("Send PM");
            jButton2.addActionListener(new java.awt.event.ActionListener() {
               public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
                jButton2ActionPerformed(evt);
            }
        });

        jTextArea1.setColumns(20);
        jTextArea1.setEditable(false);
        jTextArea1.setRows(5);
        jScrollPane2.setViewportView(jTextArea1);

        jMenu1.setText("File");

        jMenuItem2.setText("Exit");
        jMenuItem2.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem2ActionPerformed(evt);
            }
        });
        jMenu1.add(jMenuItem2);

        jMenuBar1.add(jMenu1);

        jMenu2.setText("Help");

        jMenuItem1.setText("About");
        jMenuItem1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jMenuItem1ActionPerformed(evt);
            }
        });
        jMenu2.add(jMenuItem1);

        jMenuBar1.add(jMenu2);

        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                  .addGroup(layout.createSequentialGroup()
                     .addGap(10, 10, 10)
                     .addComponent(jTextField2,
javax.swing.GroupLayout.DEFAULT_SIZE, 360, Short.MAX_VALUE))
                  .addComponent(jScrollPane2,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 370, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELA
TED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                  .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 73, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                     .addComponent(jButton2,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                     .addComponent(jButton1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
               .addContainerGap())
         );
         layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                  .addComponent(jScrollPane2,
javax.swing.GroupLayout.DEFAULT_SIZE, 171, Short.MAX_VALUE)
```

```
.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                        .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 137, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELA
TED)
                        .addComponent(jButton2)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELA
TED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jButton1))
                .addContainerGap())
            );

            pack();
        }// </editor-fold>//GEN-END:initComponents

        private void
jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jMenuItem2ActionPerformed
            formWindowClosing(null);
            System.exit(0);
        }//GEN-LAST:event_jMenuItem2ActionPerformed

        private void jButton2ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jButton2ActionPerformed
            if
((!jList1.isSelectionEmpty())&&(!jTextField2.getText().equals(""))) {
                String selected = (String)jList1.getSelectedValue();
                if (  !selected.equals(this.username) ) {
                    DataOutputStream output = null;
                    try {
                        //send_pm|username|message
```

```
                    String sendSt = "send_pm|";
                    sendSt += (String) jList1.getSelectedValue() + "|";
                    sendSt += jTextField2.getText();
                    output = new
DataOutputStream(socket.getOutputStream());
                    output.writeUTF(sendSt);
                    jTextArea1.append("[PRIVATE-"+selected+"] " +
jTextField2.getText() );
                    jTextField2.setText("");

                } catch (IOException ex) {

Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
                }
              }
              else{
                JOptionPane.showMessageDialog(null, "You cannot send PM
to yourself!");
              }

            }
            else
            {
              JOptionPane.showMessageDialog(null, "No User ID Is Selected
or No Message!");
            }
        }//GEN-LAST:event_jButton2ActionPerformed

        private void jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jButton1ActionPerformed
            if (!jTextField2.getText().equals(""))
            {
              try {
                // TODO add your handling code here:
                DataOutputStream output = new
DataOutputStream(socket.getOutputStream());
                output.writeUTF("send_all|"+jTextField2.getText());
                jTextField2.setText("");
              } catch (IOException ex) {

Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
              }
```

```
            }
            else
            {
               JOptionPane.showMessageDialog(null, "No message !");
            }
         }//GEN-LAST:event_jButton1ActionPerformed

         private void formWindowClosing(java.awt.event.WindowEvent evt)
{//GEN-FIRST:event_formWindowClosing
            // TODO add your handling code here:
            if (this.socket != null) {
               DataOutputStream output = null;
               try {
                  output = new DataOutputStream(socket.getOutputStream());
                  output.writeUTF("logout|");
                  thread.interrupt();
               } catch (IOException ex) {

Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
               }
            }
         }//GEN-LAST:event_formWindowClosing

         private void jTextField2KeyPressed(java.awt.event.KeyEvent evt)
{//GEN-FIRST:event_jTextField2KeyPressed
            // TODO add your handling code here:

         }//GEN-LAST:event_jTextField2KeyPressed

         private void jTextField2ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jTextField2ActionPerformed
            // TODO add your handling code here:
            jButton1ActionPerformed(null);
         }//GEN-LAST:event_jTextField2ActionPerformed

         private void
jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jMenuItem1ActionPerformed
            // TODO add your handling code here:
            new About().setVisible(true);

         }//GEN-LAST:event_jMenuItem1ActionPerformed
```

```
    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {

    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JList jList1;
    private javax.swing.JMenu jMenu1;
    private javax.swing.JMenu jMenu2;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JMenuItem jMenuItem1;
    private javax.swing.JMenuItem jMenuItem2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration//GEN-END:variables

}
```

2.2. ClientLogin class:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package client;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author anhtuan
```

```
 */
public class ClientLogin extends javax.swing.JFrame {

    /** Creates new form ClientLogin */
    public ClientLogin() {
        initComponents();
        this.setLocationRelativeTo(null);
        jTextField3.requestFocus();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. ssThe content of this
method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jTextField1 = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();



setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOS
E);
        setTitle("Client Login");

        jTextField1.setText("127.0.0.1");

        jLabel1.setText("Server IP/Hostname");

        jLabel2.setText("Port");
```

```
            jTextField2.setEditable(false);
            jTextField2.setText("9999");

            jLabel3.setText("Chat ID");

            jButton1.setText("Login");
            jButton1.addActionListener(new java.awt.event.ActionListener() {
               public void actionPerformed(java.awt.event.ActionEvent evt) {
                  jButton1ActionPerformed(evt);
                   }
            });

            jButton2.setText("Exit");
            jButton2.addActionListener(new java.awt.event.ActionListener() {
               public void actionPerformed(java.awt.event.ActionEvent evt) {
                  jButton2ActionPerformed(evt);
               }
            });

            javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
            getContentPane().setLayout(layout);
            layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addGroup(layout.createSequentialGroup()
                  .addGap(26, 26, 26)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                     .addGroup(layout.createSequentialGroup()
                        .addComponent(jButton1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
)
                        .addComponent(jButton2))

                     .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                           .addComponent(jLabel1)
```

```
                        .addComponent(jLabel2)
                        .addComponent(jLabel3))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                        .addComponent(jTextField1,
javax.swing.GroupLayout.DEFAULT_SIZE, 118, Short.MAX_VALUE)
                        .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jTextField3,
javax.swing.GroupLayout.DEFAULT_SIZE, 118, Short.MAX_VALUE))))
                .addGap(22, 22, 22))
            );
            layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                        .addComponent(jLabel1)
                        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                        .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel2))
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton2))


.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            );

            pack();
        }// </editor-fold>//GEN-END:initComponents

        private void jButton2ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jButton2ActionPerformed
            // TODO add your handling code here:
            System.exit(0);
        }//GEN-LAST:event_jButton2ActionPerformed

        private void jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jButton1ActionPerformed
            if (jTextField3.getText().equals("")){
                JOptionPane.showMessageDialog(null, "User ID cannot be
blank !");
            }
            else
            {
                try {
                    // TODO add your handling code here:
```

```
                // Send the request to the server, if the userid is duplicate,
program will ask to change userID
                String serverIP = jTextField1.getText();
                int serverPort = Integer.parseInt(jTextField2.getText());
                Socket socket = new Socket(serverIP, serverPort);
                String request = "request_login|"+jTextField3.getText();
                DataInputStream input = new
DataInputStream(socket.getInputStream());
                DataOutputStream output = new
DataOutputStream(socket.getOutputStream());
                output.writeUTF(request);
                String receive = input.readUTF();
                if (receive.equals("OK")){
                   new Client(socket,jTextField3.getText()).setVisible(true);
                   dispose();
                }
                else
                {
                   JOptionPane.showMessageDialog(null, receive, "Error", 0);
                }
             } catch (UnknownHostException ex) {

//Logger.getLogger(ClientLogin.class.getName()).log(Level.SEVERE, null,
ex);
                JOptionPane.showMessageDialog(null, "Network got
problems. Try again ...");
             } catch (IOException ex) {

//Logger.getLogger(ClientLogin.class.getName()).log(Level.SEVERE, null,
ex);
                JOptionPane.showMessageDialog(null, "Network got
problems. Try again ...");
             }
          }
       }//GEN-LAST:event_jButton1ActionPerformed

       /**
       * @param args the command line arguments
       */
       public static void main(String args[]) {
          java.awt.EventQueue.invokeLater(new Runnable() {
             public void run() {
```

```
                new ClientLogin().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    // End of variables declaration//GEN-END:variables

}
```

2.3. clientSocket

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package client;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JList;
import javax.swing.JTextArea;

/**
 *
 * @author anhtuan
 */
public class clientSocket implements Runnable  {

    Socket socket = null;
    JTextArea chatArea = null;
```

```
        JList list = null;

        public clientSocket(Socket socket, JTextArea chatArea,JList list){
           this.socket = socket;
           this.chatArea = chatArea;
           this.list = list;
        }
        public void run()  {
           try {
              DataInputStream input = new
DataInputStream(socket.getInputStream());
              while(true){
                    if (!socket.isInputShutdown()){
                       String receive = input.readUTF();
                       String st[] = receive.split("\\|");

                       // Normal message ////////////
                       if (st[0].equals("normal")) {
                          chatArea.append(st[1]);
                       }
                       ////////////////////////////

                       //  List Command
                       if (st[0].equals("updateList")) {
                          DataOutputStream output = new
DataOutputStream(socket.getOutputStream());
                          String sendSt = "list_request";
                          output.writeUTF(sendSt+"|");
                          String st1[] = new String[50];
                          String receive1 = input.readUTF();
                          st1 = receive1.split("\\|");
                          list.removeAll();
                          list.setListData(st1);
                       }
                       ////////////////////////////
                    }
              }
           } catch (IOException ex) {

Logger.getLogger(clientSocket.class.getName()).log(Level.SEVERE, null,
ex);
           }
```

```
        }

    }
```

About class

```
    /*
     * To change this template, choose Tools | Templates
     * and open the template in the editor.
     */

    package client;

    /**
     *
     * @author anhtuan
     */
    public class About extends javax.swing.JFrame {

        /** Creates new form About */
        public About() {
            initComponents();
            this.setLocationRelativeTo(null);
        }

        /** This method is called from within the constructor to
         * initialize the form.
         * WARNING: Do NOT modify this code. The content of this method
is
         * always regenerated by the Form Editor.
         */
        @SuppressWarnings("unchecked")
        // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
        private void initComponents() {

            jLabel1 = new javax.swing.JLabel();
            jButton1 = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOS
E);
```

```
            setTitle("About");

            jLabel1.setText("This product was developed by Tran Anh Tuan");

            jButton1.setText("OK");
            jButton1.addActionListener(new java.awt.event.ActionListener() {
               public void actionPerformed(java.awt.event.ActionEvent evt) {
                  jButton1ActionPerformed(evt);
               }
            });

            javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
            getContentPane().setLayout(layout);
            layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                     .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel1))
                     .addGroup(layout.createSequentialGroup()
                        .addGap(90, 90, 90)
                        .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE, 108,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            );
            layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addGroup(layout.createSequentialGroup()
                  .addContainerGap()
                  .addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED
)
```

```
                .addComponent(jButton1)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
            );

            pack();
        }// </editor-fold>//GEN-END:initComponents

        private void jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_jButton1ActionPerformed
            // TODO add your handling code here:
            this.dispose();
        }//GEN-LAST:event_jButton1ActionPerformed

        /**
        * @param args the command line arguments
        */
        public static void main(String args[]) {
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
                    new About().setVisible(true);
                }
            });
        }

        // Variables declaration - do not modify//GEN-BEGIN:variables
        private javax.swing.JButton jButton1;
        private javax.swing.JLabel jLabel1;
        // End of variables declaration//GEN-END:variables

    }
```

TÀI LIỆU THAM KHẢO

http://www.songmay.com/index.php?fs=main&loai=14&sub=2&tieude=Mi4gVENQL0lQIGzDoCBnw6w/&chude=RkFRcyAvIEludGVybmV0.

http://vi.wikipedia.org/wiki/Client-server

http://vi.wikipedia.org/wiki/TCP#C.E1.BA.A5u_tr.C3.BAc_g.C3.B3i_tin

http://hpcc.hut.edu.vn/forum/index.php?topic=213.0

http://www.ebook.edu.vn/?page=1.39&view=179

http://www.softechaptech.com/forums/viewtopic.php?f=83&t=95