TRƯỜNG …….

KHOA…….

-----📖-----

Báo cáo tốt nghiệp

Đề tài:

Instant Messenger cho thiết bị di động với chức năng tự động phát hiện sự hiện diện của các nút mạng

Lời cảm ơn

Khóa luận tốt nghiệp này là thành quả lớn nhất mà em hoàn thành trong suốt 4 năm học đại học. Ngoài sự cố gắng nỗ lực của bản thân còn có sự giúp đỡ của rất nhiều người.

Đầu tiên con xin cảm ơn bố mẹ đã nuôi dưỡng chăm sóc con đến ngày hôm nay

Em xin cảm ơn các thầy cô khoa công nghệ thông tin và các thầy cô giảng dạy tại trường đại học Công nghệ đã truyền đạt cho em những kiến thức trong quá trình học tập

Em xin cảm ơn Tiến sĩ Trần Thị Minh Châu – Giảng viên khoa công nghệ thông tin trường đại học công nghệ đã tận tình hướng dẫn và giúp đỡ em để em hoàn thành khóa luận này

Em xin chân thành cảm ơn tất cả.

Hà Nội tháng 5 năm 2010

Tóm tắt khóa luận

Mạng MANET là mạng không dây dạng phi thể thức. Các thiết bị tự động tự cấu hình thành một mạng mà không cần dùng đến các thiết bị định tuyến hoặc thu phát không dây. Mạng MANET rất hữu ích trong việc chia sẻ tài nguyên ở các khu vực nhỏ như các trường đại học, hội nghị,… do đó các ứng dụng trong mạng Manet rất được quan tâm.

Có một bài toán đặt ra là làm sao phát hiện ra được sự hiện diện của một nút mạng trong mạng MANET. Thực tế của bài toán này được đưa ra trong tình huống một nhóm cộng tác đang làm việc tại một nơi không có các hạ tầng cơ sở mạng như Internet hay sóng di động, khi đó họ sẽ liên lạc với nhau thế nào để biết được có người đang ở gần mình để có thể trao đổi trực tiếp. Đây là vấn đề rất đáng quan tâm bởi vì giải quyết được vấn đề này sẽ giúp cho một nhóm làm việc có thể cộng tác với nhau ở bất kì nơi nào dù là vùng sâu vùng xa không biết đến Internet.

Giải pháp cho bài toán này do tiến sĩ Trần Thị Minh Châu đưa ra là sử dụng cấu trúc dữ liệu BloomFilter làm cơ sở để tập hợp và phát tán các thông tin về sự tồn tại của nút mạng. Khóa luận này áp dụng giải pháp đó để nhận diện sự có mặt của các nút mạng trong mạng MANET cho phần mềm Instant Messenger.

Mục lục

Chương mở đầu

Sự phát triển của công nghệ thông tin đã góp phần làm cho xã hội ngày một phong phú thêm. Ở các nước đang phát triển như Việt Nam, công nghệ thông tin đã và đang trở thành một ngành công nghiệp hàng đầu và được sự quan tâm rất lớn của tất cả các bộ ngành. Công nghệ thông tin đã len lỏi vào tất cả các cơ quan từ nhà nước đến tư nhân từ các tập đoàn đến các công ty riêng. Công nghệ thông tin đã ảnh hưởng một cách tích cực đến nhiều hoạt động tại Việt Nam.

Hệ thống mạng không dây WLAN là một phát triển vượt bậc của ngành công nghệ thông tin. Hiện nay nó là sự lựa chọn cho nhiều môi trường văn phòng bởi cùng một lúc có thể kết nối máy in, Internet và các thiết bị máy tính khác mà không cần dây cáp truyền dẫn. Nhờ đó mà ta giảm thiểu được số lượng dây chạy trong phòng, từ phòng này sang phòng khác. Số lượng dây không đáng kể nên không làm thay đổi cảnh quan, thẩm mĩ nơi ở và nơi làm việc, hội họp.

WLAN là công nghệ đang được lựa chọn để ứng dụng rất nhiều trong rất nhiều các lĩnh vực. Và các phần mềm ứng dụng trong WLAN cũng được phát triển một cách rất đáng chú ý. Mục đích chính của khóa luận này là xây dựng một ứng dụng trong mạng WLAN. Đề tài của khóa luận là : *Instant Messenger cho thiết bị di động với chức năng tự động phát hiện sự hiện diện của các nút mạng* sẽ đề cập đến các vấn đề sau:

- WLAN và mạng MANET
- Java và các hỗ trợ về mạng
- Xây dựng phần mềm Instant Messenger

Khóa luận này cài đặt giải pháp của tiến sĩ Trần Thị Minh Châu để nhận diện sự có mặt của các nút mạng trong mạng MANET cho phần mềm Instant Messenger

Khóa luận gồm 5 chương:

- Chương 1: Giới thiệu về thiết bị không dây, giới thiệu về phần mềm (mục tiêu, chức năng, phạm vi hoạt động)

- Chương 2: Các mảng kiến thức cơ sở đề cập đến mạng WLAN và mạng MANET, cấu trúc dữ liệu BloomFilter và Soft State BloomFilter, Java lập trình mạng
- Chương 3: Thiết kế và cài đặt phần mềm đề cập đến các vấn đề liên quan đến việc thiết kế và cài đặt phần mềm
- Chương 4: Thử nghiệm phần mềm đề cập đến việc thử nghiệm phần mềm trong một mạng máy tính
- Chương 5: Kết luận đưa ra những cái đã làm được, chưa làm được và hướng phát triển tiếp theo

Danh sách hình

Chương 1.Giới thiệu

1.1. Giới thiệu về thiết bị không dây

Một thiết bị không dây là thiết bị có khả năng giao tiếp với các thiết bị khác mà không cần phải có dây nối. Ngày nay thiết bị không dây ngày càng phát triển. Với mục tiêu dễ dàng sử dụng mọi lúc mọi nơi nên thiết bị không dây đã trở thành một phần quan trọng trong cuộc sống. Chúng ta có thể dùng điện thoại hoặc một chiếc máy tính xách tay để kết nối Internet đó là một điều rất thuận tiện khi mà những chiếc dây mạng rất loằng ngoằng và không phải lúc nào cũng hữu ích.

Ngày nay công nghệ không dây đang phát triển mạnh mẽ do đó kéo theo sự phát triển mạnh mẽ của thiết bị không dây. Trong tương lai có lẽ chúng ta sẽ có một thế giới toàn các thiết bị không dây và sử dụng hoàn toàn công nghệ không dây.

1.2. Giới thiệu về phần mềm

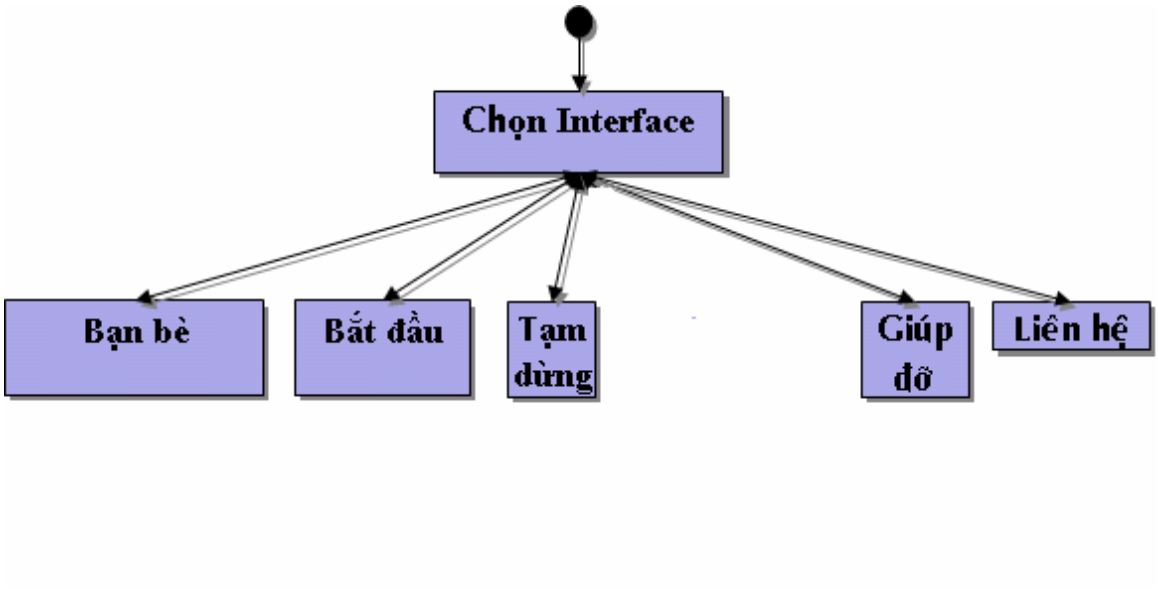
1.2.1. Mục tiêu

Chúng ta đã quen với cách làm việc theo nhóm liên lạc với nhau thông qua mạng internet nhưng chúng ta hãy nghĩ đến một hoàn cảnh khi chúng ta đang ở một vùng sâu vùng xa nào đó không có internet, không có sóng di động có nghĩa là không có hạ tầng cơ sở mạng vậy khi đó chúng ta sẽ liên lạc với nhau bằng cách nào. Một giải pháp là sử dụng phần mềm Instant Messenger này. Phần mềm này sẽ cung cấp dịch vụ liên lạc theo nhóm. Các thành viên trong nhóm sẽ biết các thành viên khác có ở gần mình hay không để có thể làm việc trực tiếp với nhau mà không cần thông qua các dịch vụ internet. Việc cộng tác làm việc vẫn có thể diễn ra trong những môi trường không có hạ tầng cơ sở mạng.

Mục tiêu của chương trình là tạo ra một phần mềm phát hiện ra sự hiện diện của các nút mạng trong mạng MANET. Phần mềm sử dụng các công nghệ hỗ trợ cho thiết bị di động, phần mềm được tối ưu hóa để có thể hỗ trợ một cách tốt nhất cho máy tính xách tay.

1.2.2. Chức năng

Sơ đồ chức năng:



Hình 2. Sơ đồ chức năng

Phạm vi hoạt động:

Phần mềm thiết kế để sử dụng cho máy tính xách tay và dùng trong mạng MANET do đó phạm vi hoạt động của nó cũng là trong mạng MANET được thiết lập bởi các máy tính xách tay

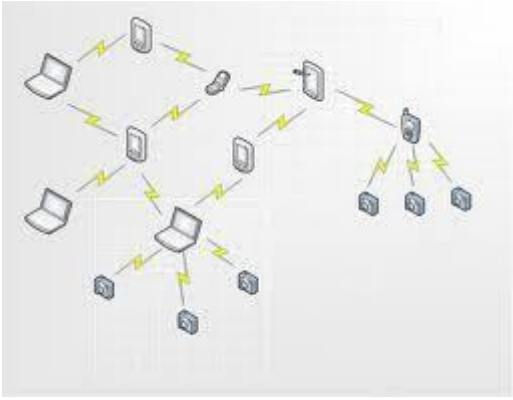
Chương 2.Kiến thức cơ sở

Ngày nay có rất nhiều ngôn ngữ lập trình bậc cao hỗ trợ tốt cho phần lập trình mạng. Java là một ngôn ngữ như vậy. Java rất thuận tiện cho việc lập trình chia sẻ các tài nguyên trên mạng. Và đó là lý do tại sao Java đã được sử dụng để viết các module cho phần mềm Instant Messenger này. Trong chương này sẽ đề cập đến hai mảng kiến thức cơ sở quan trọng của việc thiết kế phần mềm Instant Messenger này đó là cấu trúc dữ liệu Bloom filter cái rất quan trọng trong việc lưu thông tin bạn bè cùng với truy vấn xem người bạn nào đang có mặt trong mạng và đề cập các vấn đề nền tảng của Java cùng những hỗ trợ của nó trong việc lập trình mạng.

2.1. Mạng WLAN và mạng MANET

Mạng WLAN (Wireless Local Area Network – mạng nội bộ không dây) là một hệ thống truyền thông số liệu linh hoạt được thực hiện trên sự mở rộng của mạng LAN hữu tuyến. Mạng WLAN gồm các thiết bị được nối lại với nhau có khả năng giao tiếp thông qua sóng radio hay tia hồng ngoại trên cơ sở sử dụng các giao thức chuẩn riêng của mạng không dây thay vì các đường truyền dẫn bằng dây. Mạng WLAN đang thực sự thay thế cho mạng máy tính có dây, cung cấp khả năng xử lý linh động hơn và tự do hơn cho các hoạt động kinh doanh. Người dùng có thể truy cập vào mạng Intranet của nội bộ công ty hoặc mạng Internet từ bất cứ địa điểm nào trong khuôn viên của công ty mà không bị ràng buộc bởi các kết nối vật lý.

Mạng MANET (Mobile Ad-hoc NETwork - mạng di động phi thể thức): là mạng không dây tự cấu hình của các thiết bị di động. Các máy tính trong mạng này có thể chia sẻ tài nguyên nhưng không thể truy cập tài nguyên của mạng hữu tuyến nếu không cấu hình một máy tính hoạt động như cầu nối tới mạng có dây.



Hình 1.[1] Mô hình mạng ad-hoc.

Không cần dùng đến các thiết bị định tuyến (wireless router), hay thu phát không dây (Wireless Access Point). Mỗi nút mạng có thể truyền thông với nút mạng khác, không cần thiết điểm truy cập điều khiển truy cập môi trường truyền thông.

Trong mạng MANET sự phức tạp của mỗi nút mạng là cao hơn bởi vì mọi nút phải thực thi các cơ chế truy cập môi trường truyền thông, các cơ chế điều khiển ẩn hoặc bộc lộ các vấn đề thiết bị đầu cuối và có lẽ là các cơ chế ưu tiên để cung cấp một dịch vụ đảm bảo chất lượng. Mạng không dây kiểu này tỏ ra mềm dẻo hơn hết, ví dụ: cần thiết cho các hội nghị đột xuất, các sự thay thế nhanh của cơ sở hạ tầng hoặc các kịch bản truyền thông đi xa từ bất kỳ cơ sở hạ tầng nào.

Chế độ IEEE Ad-hoc: Chế độ này thì các nút di động truyền thông trực tiếp với nhau mà không cần tới một cơ sở hạ tầng nào cả. Trong chế độ này thì các liên kết không thể thực hiện qua nhiều chặng.

Ưu điểm của mạng MANET:

Số lượng thiết bị di động lớn. Có thể sử dụng cho rất nhiều thiết bị như là PDA, Máy tính xách tay hay điện thoại di động.

Trong mạng cơ bản thì cơ sở hạ tầng, các trạm trung gian, thu phát sóng là yếu tố quan trọng quyết định chất lượng của mạng, còn trong mạng MANET các nút mạng kết nối thông qua các nút mạng (không cần đến các trạm thu phát), các nút mạng có

[1] Nguồn [5]

thể di chuyển tự do trong cấu trúc mạng do đó nó có tính chất cơ động cao và do đó làm giảm bớt sự phụ thuộc vào cơ sở hạ tầng, làm cho mạng dễ phát triển dễ dàng, tốc độ phát triển của mạng nhanh

Những thách thức đối với mạng MANET:

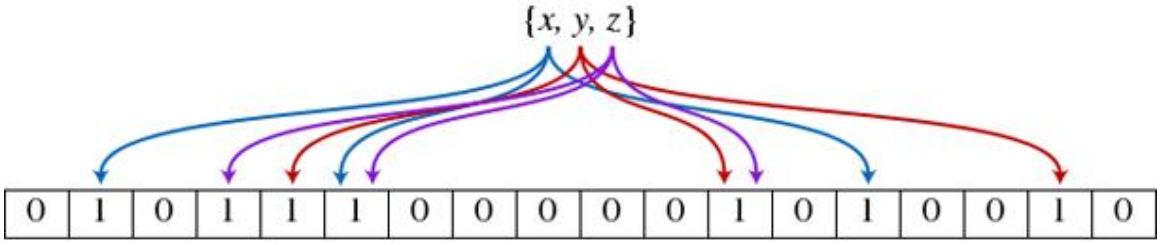
- Chi phí cho việc sử dụng phổ tần số
- Việc định tuyến/ Quản lý các nút mạng:
- Thêm vào mạng
- Thoát khỏi mạng
- Hiệu quả sử dụng nguồn điện
- Giao thức điều khiển truyền
- Tính di động của các Nút
- Băng thông

Trong một mạng MANET các nút di chuyển tự do vì thế tô pô mạng có thể bị thay đổi một cách nhanh chóng và không thể dự đoán đc. Hơn nữa các nút trong mạng ad hoc bị giới hạn phạm vi truyền làm cho một số nút không thể giao tiếp trực tiếp với một nút khác. Vì các tuyến đường trong mạng ad hoc nhiều khả năng sẽ phải trải qua nhiều chặng nên mỗi nút mạng phải đóng vai trò như một router.

Việc thiết kế giao thức cho mạng MANET là không hề đơn giản. Đầu tiên, trong mạng ad hoc các nút có thể di chuyển làm cho các thành phần cũng như tô pô mạng thay đổi thường xuyên. Thứ hai là do tính đa dạng và bất thường của các liên kết không dây làm cho việc mất gói tin xảy ra thường xuyên.

2.2. Bloom Filter

Bloom filter[1] là cấu trúc dữ liệu dùng để kiểm tra xem một phần tử có phải là thành viên của một tập hay không. Một Bloom filter rỗng là một mảng gồm *m* bit.



Hình 3. Bloom filter[2]

Giả sử ta gọi *S* là tập gồm *n* phần tử, Bloom filter gồm *m* phần tử mảng giống như hình trên mỗi phần tử mảng là 1 bit. *k* là số hàm băm độc lập với nhau hoạt động như sau: mỗi lần băm nó cho ra một giá trị nằm trong khoảng từ *0...m-1*. Ban đầu thì *m* phần tử mảng được gán giá trị là 0.

Giả sử ta muốn thêm một phần tử *x* vào Bloom filter này ta sẽ băm *x* bằng *k* hàm băm và cho ta *k* giá trị. Các phần tử mảng tại *k* giá trị đó sẽ được gán bằng 1. Ta hãy xem hình trên. Hình trên thể hiện một Bloom filter có 3 hàm băm. Khi băm phần tử *x* cho ra 3 giá trị là 1,5,14 thì các phần tử mảng tại các giá trị đó được gán giá trị là 1. Với *y* và *z* cũng tương tự như vậy.

Khi ta muốn truy vấn một phần tử nào đó xem nó có là thành viên của tập *S* này hay không ta cũng sẽ băm phần tử đó ra và kiểm tra xem tại các vị trí băm đó thì giá trị phần tử mảng là 0 hay là 1. Tuy nhiên như thế sẽ dẫn đến các khả năng được gọi là False positive tức là giả sử ta kiểm tra một phần tử t nào đó xem nó có tồn tại trong tập S không. Ta băm *t* với 3 hàm băm và giả sử cho giá trị là 1,3,4 khi ta kiểm tra các phần tử mảng tại vị trí này thì thấy giá trị nó là 1 tuy nhiên trên hình trên thì ta thấy t không tồn tại tức là thực ra nó không tồn tại nhưng ta lại tìm được sự tồn tại của nó. Do đó các hàm băm phải được tạo sao cho khả năng False positive là nhỏ nhất. Khi ta đã biết m và thì k thường được chọn là 1 giá trị nguyên và được tính bằng công thức :

$$k = ln2.(\frac{m}{n})$$

[2] Nguồn[5]

Cái khó của một Bloom filter dạng chuẩn là kiểu cấu trúc dữ liệu mà ta chi có thể thêm phần tử vào mà không thể loại phần tử đó ra.

Ta có thể dùng các thuật toán mã hóa *SHA1* hoặc *MD5* để tạo các hàm băm. Có thể sử dụng cách như sau để tạo hàm băm thứ *i*:

$$H_i(x)=MD5(x+i)$$

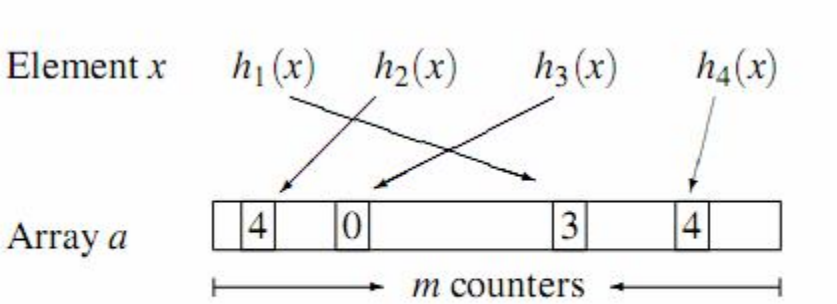
2.3. Soft State Bloom Filter

Trong một mạng máy tính luôn luôn có những lúc một vài nút di chuyển ra khỏi mạng hoặc là khi khoảng cách giữa các nút xa thêm. Có rất nhiều trường hợp ta cần phải xét đến việc một nút không còn tồn tại trong mạng.

Một Bloom filter dạng chuẩn không hỗ trợ việc loại bỏ một phần tử do đó cần phải sửa đổi vì hệ thống cần phải loại bỏ một nút nếu như nó không còn tồn tại ở một thời điểm nào đó. Soft state Bloom filter là dạng Bloom filter để thực hiện yêu cầu đó của hệ thống [2].

Như ta đã tìm hiểu ở bên trên. Mỗi phần tử mảng của Bloom filter bao gồm 1 bit. Bây giờ thay vì sử dụng 1 bit cho mỗi phần tử mảng ta dùng *l* bit cho mỗi phần tử mảng, và để tránh cho việc tràn bộ nhớ thì *l* cần phải được chọn giá trị một cách hợp lý, thường *l* là nhỏ *l =3* hoặc bằng 4 (như trong khóa luận này l được chọn là 3) khi đó mỗi phần tử mảng được lưu trữ giá trị như một bộ đếm, giá trị bộ đếm này như là tuổi của phần tử mảng đó. Thay vì ban đầu ta khởi tạo giá trị cho phần tử mảng là 0 thì ta khởi tạo giá trị lớn nhất cho nó là 2^l-1 , giá trị lớn nhất này thể hiện rằng vị trí này chưa được thiết lập nó giống như việc thiết lập giá trị ban đầu là 0 tại mỗi phần tử mảng ở một Bloom filter chuẩn. Cái bộ đếm này nó hỗ trợ việc loại bỏ một nút nếu như nó không tồn tại quá lâu.

Khi một phần tử bị loại bỏ khỏi Bloom filter thì giá trị phần tử mảng khi băm nó bị giảm.



Hình 4. Soft state Bloom filter[3]

Một False Positive khi truy vấn nút u xảy ra khi tất cả các hàm $h_1, ..., h_k$ đều cho giá trị tại các phần tử mảng mà tại đó giá trị không phải là $2^l - 1$

Xác xuất xảy ra False Positive được tính theo công thức sau:

$$F = (1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-\frac{kn}{m}})^k$$

2.4. Lập trình mạng với Java

2.4.1. Nền tảng Java

Java là ngôn ngữ lập trình hướng đối tượng (tựa C++) do Sun Microsystem đưa ra vào giữa thập niên 90. Chương trình viết bằng ngôn ngữ lập trình Java có thể chạy trên bất kỳ hệ thống nào có cài máy ảo Java (Java Virtual Machine).

Ngày nay, đối với Java người ta không còn nhắc đến như là 1 ngôn ngữ lập trình mà nhắc đến một công nghệ hay một nền tảng phát triển, nó bao gồm các bộ phận:

- Máy ảo Java: JVM
- Bộ công cụ phát triển: J2SDK

[3] Nguồn: [2]

- Các đặc tả chi tiết kĩ thuật (specifications)
- Ngôn ngữ lập trình (programming language)
- Các công nghệ đi kèm như JSP, Servlet, EJB, JDBC, JNDI, JMX, RMI … và framework như Struts, Spring, JSF, Hibernate, JavaFX ...

Java đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất cho tất cả các lập trình viên.

2.4.2. Lập trình mạng với Java

Java là ngôn ngữ rất thích hợp để cài đặt các ứng dụng yêu cầu các giao tiếp giữa các tài nguyên mạng.

Java.net được bắt đầu từ J2SE 1.4 là gói dùng để cung cấp tất cả các Interface và các class cho việc cài đặt các ứng dụng mạng.

Lớp đầu tiên trong gói Java.net mà chúng ta quan tâm đến là lớp InetAddress. Đây là một lớp quan trọng khi lập trình mạng nó dùng để xử lý về địa chỉ IP và hostname của một máy. Ta có thể dùng phương thức static *getByName()* để lấy địa chỉ IP của một máy hoặc cũng có thể dùng phương thức *getLocalHost()* để lấy địa chỉ Ip của máy đang sử dụng. Nó sẽ là rất hữu ích khi lập trình mạng.

Trong mạng thì các chương trình giao tiếp với nhau thông qua các Socket. Java cũng sử dụng cách đó và cung cấp cho chúng ta cả 2 loại Socket là TCP Socket và DatagramSocket.

TCP Socket

Đây là một giao tiếp kiểu TCP/IP tức là giao tiếp hướng kết nối.

Gói Java.net cung cấp 2 lớp hữu ích cho việc thực hiện kết nối bằng TCP Socket. Đầu tiên là Class ServerSocket dùng cho Server và lớp Socket dùng cho Client.

Khởi tạo một ServerSocket

```
ServerSocket ServSock = new ServerSocket(1234);
```

Ở trên Server có tên là servSock sẽ lắng nghe kết nối ở cổng 1234.

Và Server này dùng phương thức accept để chờ đợi kết nối đến từ Client:

```
ServSock.accept();
```

Để khởi tạo một Socket cho Client

```
Socket client = new Socket(1234);
```

Việc giao tiếp giữa Server và Client được thực hiện qua các Stream. OuputStream dùng để ghi dữ liệu vào luồng và InputStream để đọc dữ liệu.

Do khóa luận này không sử dụng TCP Socket nên chúng ta sẽ không đi sâu vào phần này.

UDP Socket

UDP Socket là giao thức quan trọng nhất được sử dụng trong chương trình này. Chúng ta sẽ tìm hiểu sâu về UDP Socket sau đây.

UDP Socket không giống như TCP socket, UDP Socket là kiểu socket không kết nối tức là việc gửi dữ liệu đi không đảm bảo là dữ liệu sẽ được nhận. Kiểu kết nối này thực hiện nhanh hơn kiểu TCP vì nó không cần tạo ra việc bắt tay 3 bước tuy nhiên thì đây là kiểu kết nối không tin cậy.

Gói Java.net cung cấp cho ta lớp DatagramSocket và DatagramPacket dùng cho việc kết nối UDP này. DatagramSocket được tạo ở cả Client và Server để gửi và nhận các DatagramPacket

Về phía Server khởi tạo một Server như sau:

```
DatagramSocket serverSocket = new DatagramSocket(8888);
```

Việc gửi và nhận thông qua các DatagramPacket chứa mảng các byte dữ liệu.

Giả sử muốn gửi đi chuỗi “Toi yeu viet nam”:

```
String s = "Toi yeu viet nam";
```

Chuyển chuỗi này thành mảng byte

```
Byte[] data = s .getByte();
```

Khởi tạo DatagramPacket để gửi đi:

```
DatagramPacket outPacket =
new DatagramPacket(data, data.length(),clientAddress);
```

clientAddress ở đây là địa chỉ Ip của Client mà ta muốn gửi đến.

Và sau đó thực hiện việc gửi:

```
serverSocket.send(outPacket);
```

Đó là việc gửi dữ liệu ở Server. Tại Client muốn nhận được dữ liệu đó,thì cũng khởi tạo một DatagramSocket :

```
DatagramSocket clientSocket = new DatagramSocket(8888);
```

Để thực hiện việc nhận dữ liệu thì phải khởi tạo một buffer là một mảng các byte

```
Byte[] buffer = new byte[256] ;
```

Sau đó khởi tạo một DatagramPacket để chờ :

```
DatagramPacket inPacket =
          new DatagramPacket(buffer, buffer.length);
```

Và thực hiện việc chờ dữ liệu gửi đến:

```
clientSocket.receive(inPacket);
```

Sau đó có thể chuyển dữ liệu lại thành dạng chuỗi đã gửi sau khi nhận:

```
String message =  new String(inPacket.getData(),
       0,inPacket.getLength());
```

UDP là một phương thức không kết nối do đó nó rất hữu ích khi muốn broadcast dữ liệu trong mạng. Broadcast sử dụng UDP là một trong những phần quan trọng nhất trong khóa luận này. Thay vì gửi trực tiếp dữ liệu đến một máy có địa chỉ chính xác nào đó thì ta broadcast dữ liệu trong mạng. Để thực hiện điều này chỉ cần 1 địa chỉ broadcast của máy.

Để thực hiện việc broadcast dữ liệu này thì ở bước khởi tạo DatagramPacket để send thì ta khởi tạo DatagramPacket như sau:

```
     outDataPacket = new DatagramPacket(data,
data.length,broadcastAddress, 8000);
```

Sự khác biệt ở đây là chúng ta đưa vào địa chỉ broadcast của máy chứ không phải là địa chỉ của Client. Việc nhận dữ liệu broadcast cũng giống như việc nhận dữ liệu bình thường.

Chương 3.Thiết kế và cài đặt phần mềm

Như đã trình bài ở chương mở đầu, mục đích chính của phần mềm này là phát hiện sự hiện diện của những người bạn của mình trong mạng MANET và khoảng cách của mình đến người đó là bao nhiêu chặng (hop). Ta biết để biết được một người có hiện diện trong mạng hay không đơn giản nhất là lắng nghe thông điệp phát ra từ người đó, nếu như nghe được thông điệp thì người đó tồn tại trong mạng và nếu như không nghe thấy thông điệp gửi đi từ người đó thì chứng tỏ người đó không tồn tại trong mạng. Từ đó ta thấy một máy muốn biết sự tồn tại của máy khác thì đơn giản nó chỉ cần broadcast dữ liệu và mong muốn dữ liệu đó được gửi đến những máy trong mạng và máy nào trong mạng nhận được sẽ thực hiện công việc tương tự là broadcast để chứng tỏ sự tồn tại của mình. Vấn đề là lưu dữ liệu của một máy gửi đến trong mạng để có thể truy vấn xem đó có phải là bạn của mình hay không. Chương trình sử dụng một Soft State Bloom filter như đã được đặc tả ở trên để lưu thông tin về những máy tồn tại trong mạng và khi cần xem người bạn nào của mình hiện diện trong mạng thì chỉ cần truy vấn thông tin trong Bloom filter này. Và chúng ta hãy xem xét các phần sau:

3.1. Hàm băm (Hash Function)

Chúng ta đã biết một Bloom filter cần phải có *k* hàm băm hoạt động độc lập và việc quan trọng là phải tạo được các hàm băm để tỉ lệ False Positive là nhỏ nhất. Chúng ta cũng đã đề cập đến cách tạo hàm băm sử dụng thuật toán mã hóa MD5. MD5 là một hàm băm dùng để mã hóa với giá trị băm là 128 bit. MD5 là thuật toán mã hóa mà không thể giải mã. Java có gói java.security chứa Class MessageDigest hỗ trợ thuật toán mã hóa MD5 này.

Chúng ta cũng đã nhắc đến cách tạo ra *k* hàm băm tức là hàm băm thứ *i* sẽ lấy theo kiểu :

$$H_i(x)=MD5(x+i)$$

Đây là cách tạo hàm băm đơn giản nhất mà lại hiệu quả. Trong khóa luận này chúng ta phải thực hiện băm một chuỗi với k hàm băm.Việc thực hiện băm một chuỗi sử dụng thuật toán MD5 như sau:

- Mã hóa chuỗi đó bằng thuật toán MD5
- Tính toán chuỗi đó ra giá trị dạng long
- Sử dụng hàm băm theo phương pháp chia lấy dư để lấy giá trị băm được

Sử dụng Class này để tạo ra hàm băm:

```
digestFunction = MessageDigest.getInstance("MD5");
public static long createHash(byte[] data) {
        long h = 0;
        byte[] res;
        digestFunction.update(data, 0, data.length);
        res = digestFunction.digest();
        for (int i = 0; i < 4; i++) {
            h <<= 8;//h*(2^8)==h*256
             h |= ((int) res[i]) & 0xFF;//== h =h|((int) res[i]) &
           0xFF
        }
        return h;
    }
```

Sau đó băm giá trị đầu vào với k hàm băm

```
for (int x = 0; x < k; x++) {
   hash = createHash(valString + Integer.toString(x));
}
```

Đây là giá trị băm được và sau đó dùng cách đơn giản là láy số dư của hash cho số phần tử mảng để có được giá trị lưu vào Bloom filter đây là cách sử dụng hàm băm theo phương pháp chia:

```
hash = hash % (long) sizeofSet;
```

hash ở đây là vị trí phần tử mảng đươc gán giá trị trong Bloom filter

3.1.1. Địa chỉ broadcast

Ta đã biết được một phần của thuật toán là việc liên tục broadcast thông tin mà mình có được đến những nút xung quanh trong mạng. Việc broadcast này là chứng tỏ sự tồn tại của mình trong mạng và cũng để gửi những thông tin mà mình biết về tất cả các nút trong mạng cho những nút có thể nhận.

Như đã tìm hiểu ở trên, khi broadcast dữ liệu trong mang ta cần phải có địa chỉ broadcast của máy. Khi ta có địa chỉ Ip và subnet mask ta có thể dễ dàng tính được địa chỉ broadcast của máy. Tuy nhiên Java hỗ trợ việc lấy địa chỉ broadcast của máy rất đơn giản.

Ta lấy địa chỉ broadcast như sau:

```
public String getBroadcastAddress() {
   String broadcastAddress = "";
   try {
      Enumeration<NetworkInterface> en =
NetworkInterface.getNetworkInterfaces();
                while (en.hasMoreElements()) {
                    NetworkInterface ni = en.nextElement();
                    if (ni.isLoopback()) {
                        continue;
                    }
                    for (InterfaceAddress interfaceAddress :
                            ni.getInterfaceAddresses()) {
                        if
(ni.getName().equalsIgnoreCase(interfaceName)) {
      InetAddress broadcast = interfaceAddress.getBroadcast();
                            broadcastAddress =
broadcast.getHostAddress();
                        }
                    }
                }
            } catch (Exception e) {
            }
            return broadcastAddress;
        }
```

Ở đây interfaceName là tên của một interface và việc lấy địa chỉ Broadcast ở đây là lấy địa chỉ broadcast của một interface xác định

3.1.2. TimeOut và Refresh

Việc broadcast được thực hiện liên tục, 2 lần liên tiếp cách nhau một khoảng thời gian được gọi là TimeOut.

Trong mỗi lần Timeout đó thì sẽ thực hiện 3 bước:

Bước 1:

Tăng giá trị mỗi phần tử mảng lên 1 nếu như giá trị này chưa ở mức 2^l-1 tức là giá trị của phần tử mảng đó đã được thiết lập trước đó rồi

```
public void decay() {
        for (int i = 0; i < bitset.size(); i++) {
            if ((Integer) bitset.get(i) < 7) {
                bitset.set(i, (Integer) bitset.get(i) + 1);
            }
        }
    }
```

Bước 2:

Refresh thông tin về nút bằng cách băm chính mình (hostname của máy) và gán các giá trị tại các vị trí phần tử mảng băm được là 0

Việc này đơn giản chỉ là add lại chính bản thân nó:

```
public Vector add(String valString) {
    long hash = 0;
    Vector hashArray = new Vector();
    for (int x = 0; x < k; x++) {
        hash = createHash(valString + Integer.toString(x));
        hash = hash % (long) sizeofSet;
        hashArray.add(Math.abs((int) hash));
        bitset.setElementAt(0, Math.abs((int) hash));
    }
         return hashArray;
}
```

Bước 3:

Broadcast dữ liệu mới update sang nút hàng xóm

3.2. Merge thông tin

Dữ liệu được broadcast là thông tin về các nút trong mạng mà máy biết do đó khi nhận được thôn tin từ một nút khác thì phải thực hiện việc thêm thông tin đó vào những thông tin mình đã có. Việc thêm thông tin này được gọi là Merge thông tin hay đơn giản là nó chỉ là Merge cái thông tin ở trong Bloom filter của mình với thông tin nhận được.

Việc nối này thực ra là nối hai mảng phần tử mảng của 2 Bloom filter. Việc nối thực ra chỉ là việc so sánh giá trị tại cùng vị trí phần tử mảng của 2 Vector và lấy giá trị nhỏ hơn và lưu vào một Bloom filter

```
public void merge(Vector receive) {
    int size = receive.size();
    for (int i = 0; i < size; i++) {
              if ((Integer) bitset.get(i) > (Integer)
 receive.get(i)) {
            bitset.set(i, receive.get(i));
        }
    }
}
```

Ta có thể thấy ở trên. Thực hiện việc so sánh giá trị các phần tử mảng tương ứng và chọn giá trị nhỏ hơn.

3.3. Tuổi của thông tin

Như chúng ta thấy bước đầu tiên khi broadcast dữ liệu là tăng giá trị phần tử mảng lên 1 nếu nó chưa có giá trị max. Giai đoạn này được gọi là decay (tức là làm già thông tin). Giá trị này chính là tuổi của thông tin

3.4. Kiểm tra sự tồn tại của Friend trong mạng

Để xác định xem một nút u có tồn tại hay không ta lại thực hiện việc băm u k lần bằng k hàm băm của Bloom filter. Sau đó ta lấy giá trị lớn nhất tại mỗi phần tử mảng mà băm được goi là *T(u)*

Nếu như giá trị $T(u)+1=2^1$ thì nó không tồn tại trong mạng, ngược lại $T(u)+1$ chính là khoảng cách từ máy hiện tại đến nút u trong mạng

Ở đây thông tin cần truy vấn là tên của người bạn của mình trong một file danh sách do đó ta phải băm cái chuỗi này bằng k hàm băm của Bloom filter:

```
/*
 * kiem tra xem nick do duoc add chua
 */
public boolean contains(String valString) {
    long hash;
    for (int i = 0; i < k; i++) {
        hash = createHash(valString + Integer.toString(i));
        hash = hash % (long) sizeofSet;
        if (((Integer) bitset.get(Math.abs((int) hash)) == 7)) {
            return false;
        }
    }
    return true;
}
```

Ta thấy hàm trên sẽ kiểm tra xem Chuỗi tên người bạn mà ta muốn kiểm tra xem nó có tồn tại trong Bloom filter hay không

Sau đó ta thực hiện phương thức tiếp theo:

```
public Vector hopQuery(String queryString) {
        Vector vector = new Vector();
        if (contains(queryString)) {
            long hash;
            for (int i = 0; i < k; i++) {
                hash = createHash(queryString +
Integer.toString(i));
                hash = hash % (long) sizeofSet;
                vector.add(bitset.get(Math.abs((int) hash)));
            }
        }
        return vector;
    }
```

`if (contains(queryString))` nếu như Bloom filter này có chứa chuỗi trên thì ta lấy giá trị các phần tử mảng mà các hàm băm băm được khi băm chuỗi này.

Giá trị nhận được là một Vector chứa các giá trị băm được bằng k hàm băm đó. String này không tồn tại trong Bloom filter nếu như Vector là rỗng. Mục đích của việc này đơn giản chỉ là lấy tập các giá trị đó để có thể tìm được giá trị lớn nhất trong tất cả các phần tử mảng đó

```
/*
 * khoang cach
 */

public int hop(String queryString) {
    int max = -1;
    Vector vector = hopQuery(queryString);
    if (vector.isEmpty()) {
        return -1;
    }
    for (int i = 0; i < vector.size(); i++) {
        int temp = (Integer) vector.get(i);
        if (temp > max) {
            max = temp;
        }
    }

    return max + 1;
}
```

Phương thức `hop(String queryString)` trả về khoảng cách của nút đang truy vấn

3.5. Dữ liệu

Dữ liệu của chương trình là tên những người bạn của mình và được tổ chức dưới dạng file. Thông tin được đặt trong file friend.inp. Chương trình cho phép thêm bạn vào danh sách những người bạn của mình.

3.6. Cài đặt

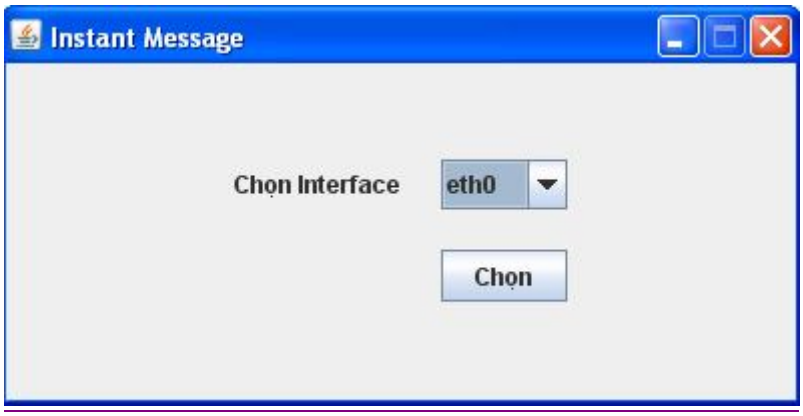
Phần mềm này sử dụng Java là ngôn ngữ chính. Do đó máy chạy phần mềm phải cài Java và thực hiện việc cài đặt đơn giản như cài đặt các phần mềm khác.

Chương 4.Thử nghiệm phần mềm

Chương trình được chạy thử trên nền MS Windows sử dụng cho các máy tính xách tay tạo các mạng ad-hoc

4.1. Giao diện của phần mềm:

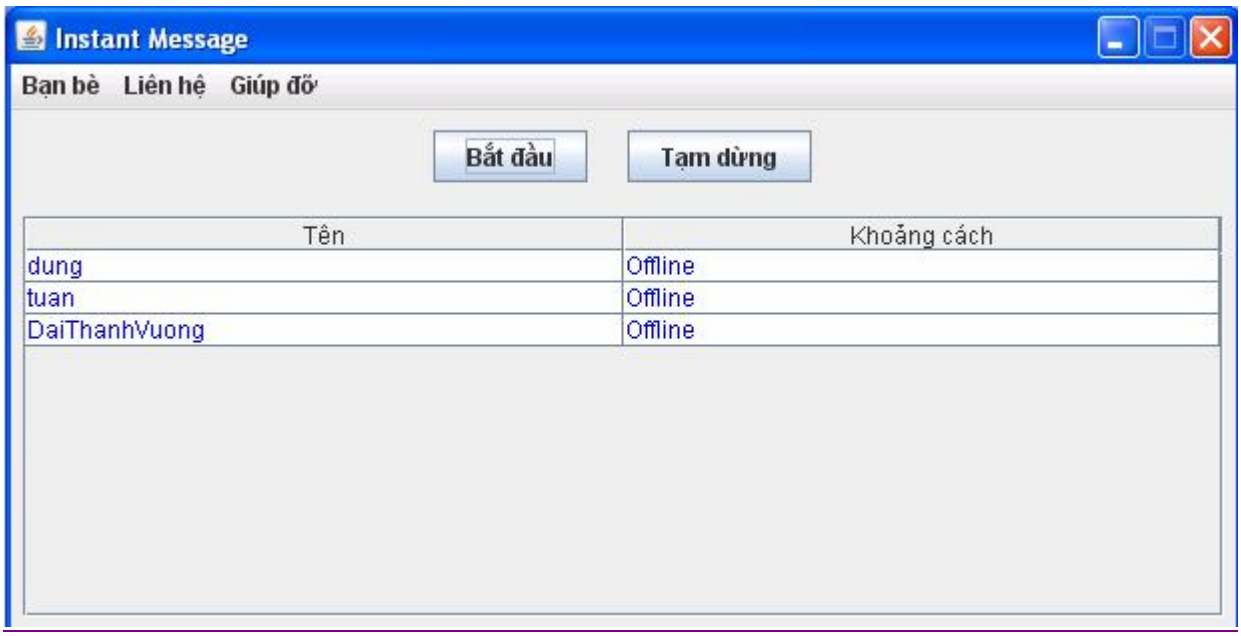
4.2. Chọn Interface:



Hình 5:Chọn interface

Ở đây giá trị cho phép chọn là tên các interface network của máy. Bạn được phép chọn interface nào để thực hiện kết nối. Ở hình trên là interface có tên là eth0

4.3. Giao diện chính: Instant Message



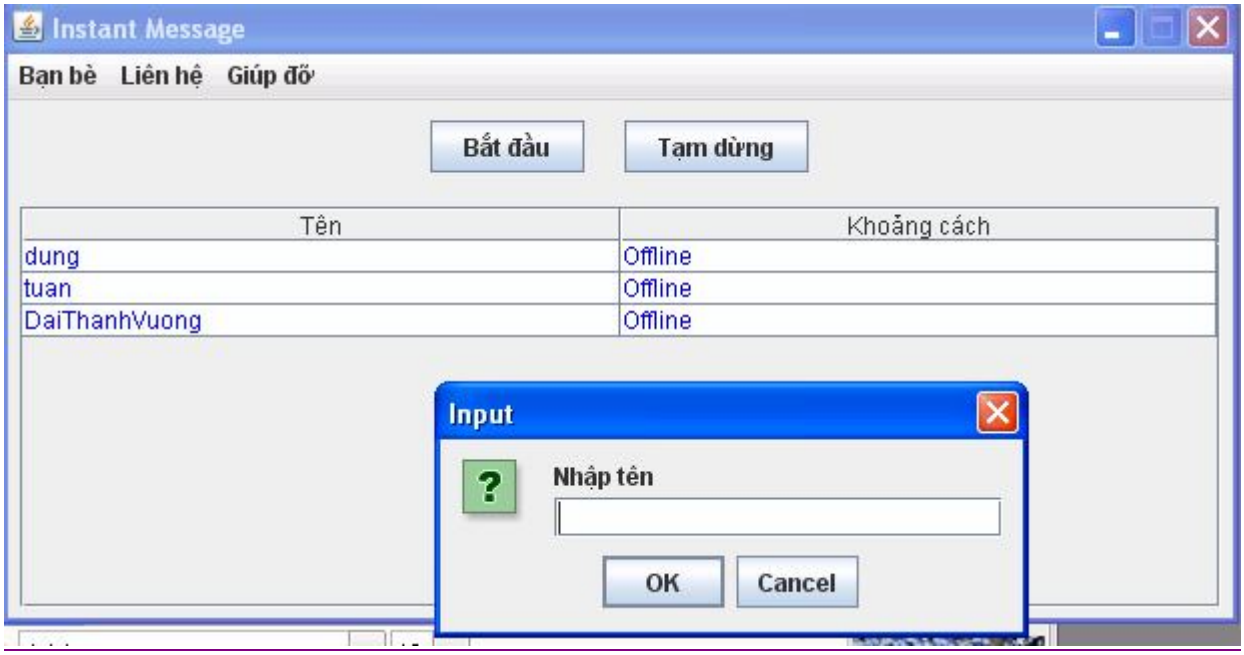
Hình 6:Giao diện chính

Giao diện này bao gồm 1 menubar với các menu Bạn bè, liên hệ và giúp đỡ cho phép người dùng chọn các chức năng tương đương từ các menu này. 2 button bắt đầu và tạm dừng để thực hiện việc bắt đầu chạy chương trình và tạm dừng chương trình. Để bắt đầu chương trình hãy nhấn vào nút "Bắt đầu" và khi muốn tạm dừng chương trình thì nhấn vào nút "Tạm dừng".

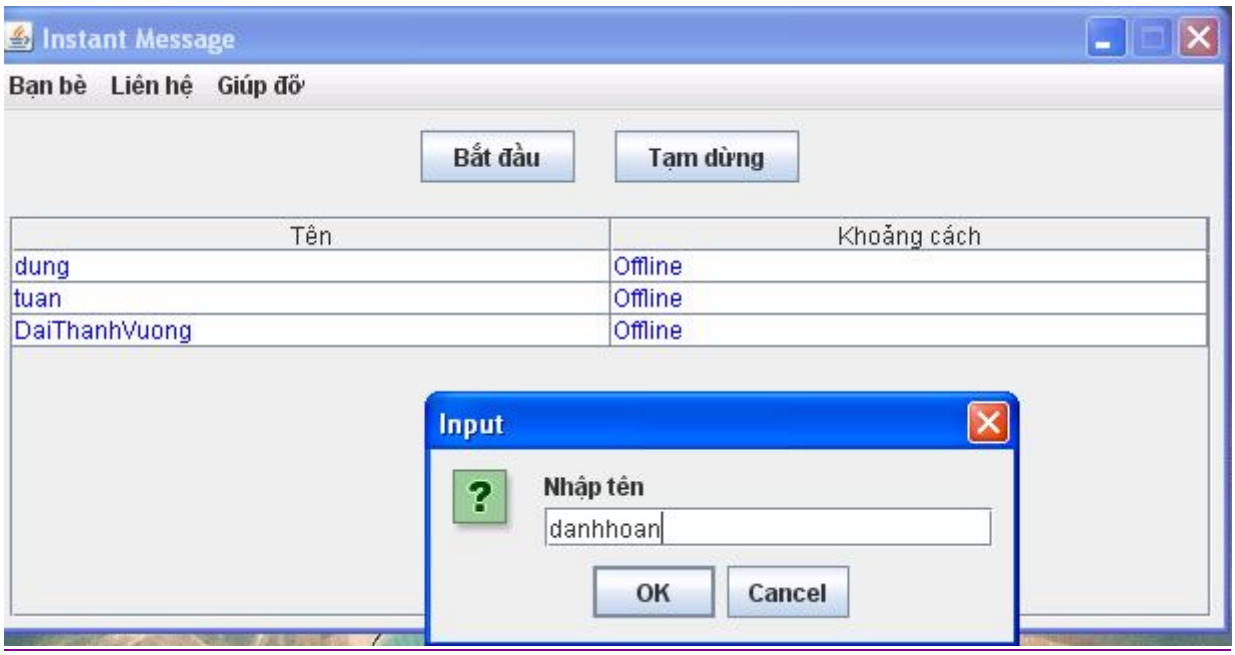
Giao diện này còn có thêm một bảng là. Bảng này bao gồm 2 cột. Một cột là tên của những người bạn trong FriendList của mình. Cột thứ 2 là khoảng cách tới người bạn đó.

4.4. Giao diện thêm bạn:

Khi chọn chức năng bạn bè sẽ xuất hiện giao diện này



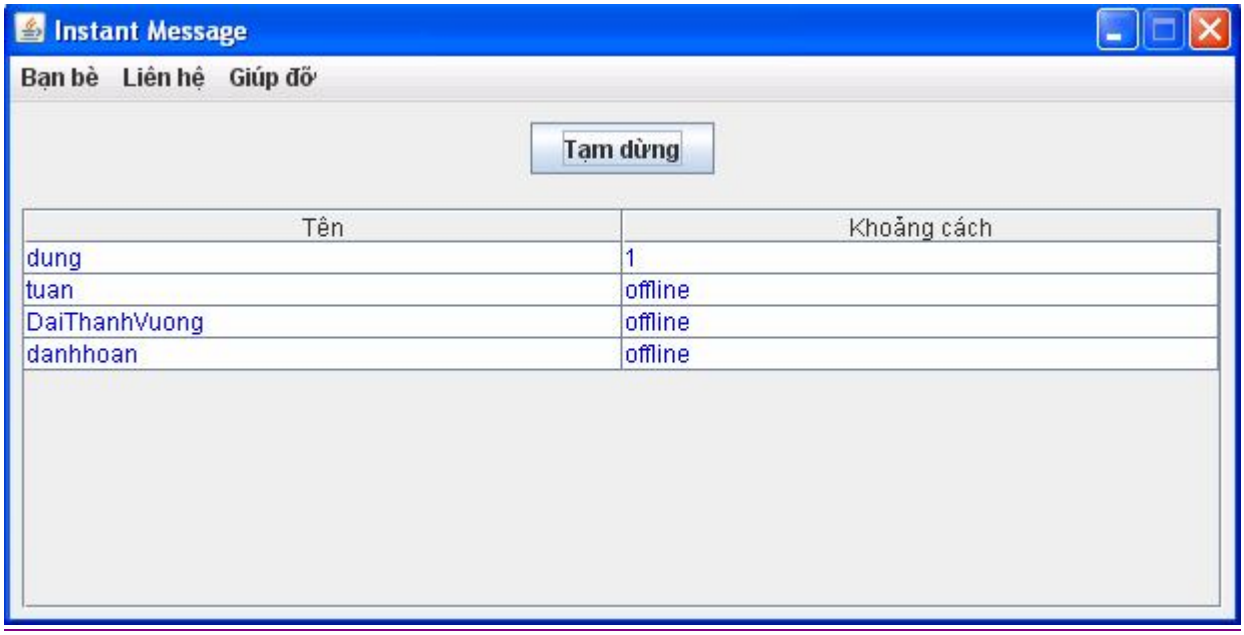
Hình 7:Giao diện thêm bạn



Hình 8:Giao diện thêm bạn

Giao diện xuất hiện một Dialog input cho phép người dùng nhập vào một chuỗi. Ở đây cho phép nhập với số kí tự không giới hạn. Tên nhập ở đây là một hostname của một máy.

4.5. Giao diện chạy phần mềm:

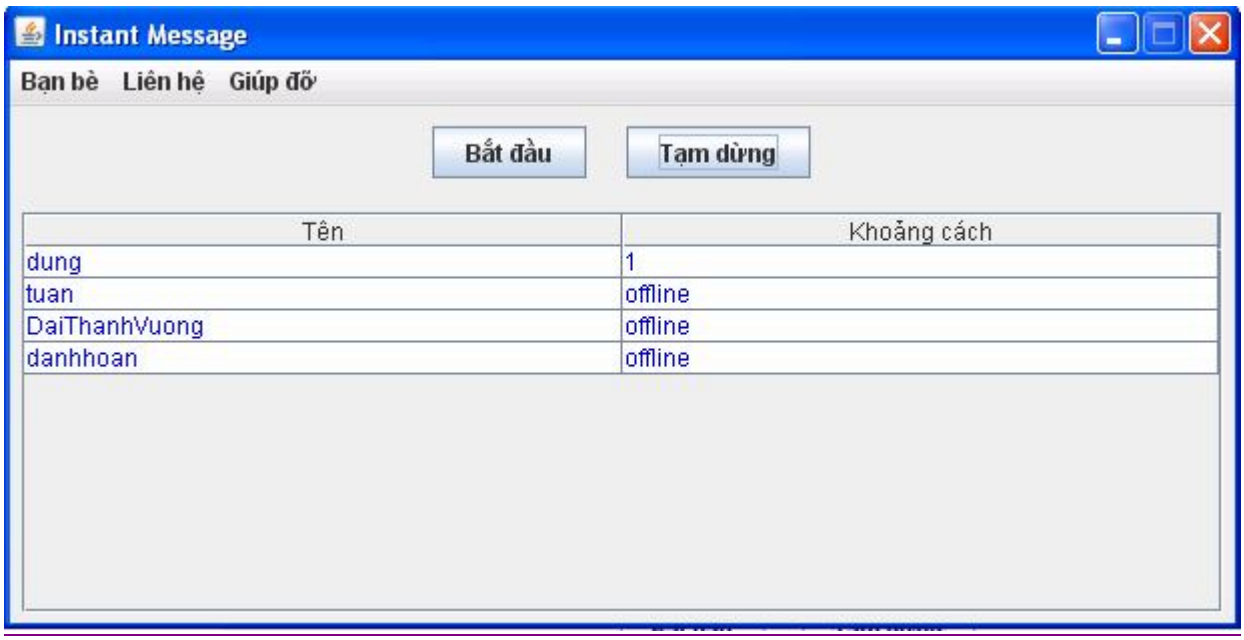


Hình 9: Giao diện chạy phần mềm

Chạy phần mềm sẽ có giao diện như trên với giá trị ở cột khoảng cách được thay đổi với giá trị tính toán được trong chương trình. Như hình trên khoảng cách từ máy này đến máy có hostname "dung" là 1 hop còn các máy khác thì không online sẽ có giá trị là offline.

Khi chương trình chạy thì nút "Bắt đầu" sẽ ẩn đi và giao diện chỉ còn lại nút "Tạm dừng"

4.6. Giao diện tạm dừng:



Hình 10: Giao diện tạm dừng

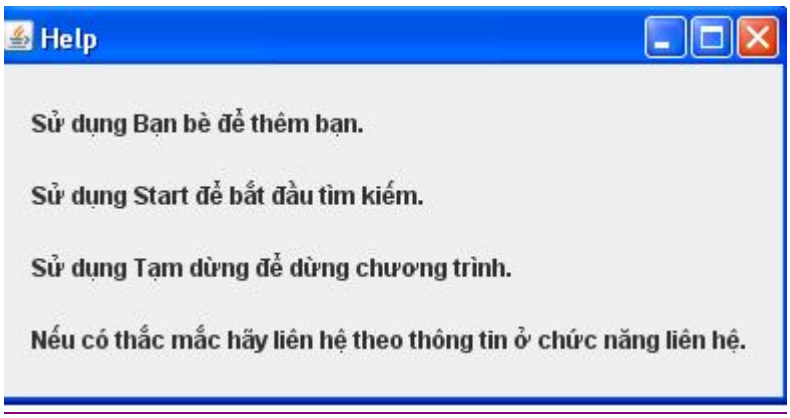
Khi nhấn nút "Tạm dừng" chương trình sẽ dừng lại và nút "Bắt đầu " xuất hiện cùng với các giá trị khoảng cách đo được khi chạy chương trình.

4.7. Giao diện liên hệ:

Hình 11: Giao diện liên hệ

Giao diện này đưa ra thông tin cần liên hệ khi có thắc mắc gì về phần mềm.

4.8. Giao diện giúp đỡ:



Hình 12: Giao diện giúp đỡ

Giao diện này chỉ ra cách sử dụng phần mềm này.

Chương 5.Kết luận

Với việc sử dụng công nghệ Java thì khóa luận này đã xây dựng được phần mềm Instan Messenger. Phần mềm Instant Messenger này được sử dụng với thiết bị là Máy tính xách tay và được dùng trong mạng MANET

Tuy nhiên hạn chế là chương trình mới chỉ sử dụng cho Máy tính xách tay và được dùng trong Win

Hướng phát triển tiếp theo là phát triển để chương trình có thể sử dụng trên các thiết bị di động cầm tay như điện thoại di động hoặc là PDA và có thể chạy trên nhiều hệ điều hành như Linux hoặc các hệ điều hành khác.

Tài liệu tham khảo

[1]. Bloom, Burton H. (1970), Space/time trade-offs in hash coding with allowable errors, Communications of the ACM 13 (7): tr. 422–426.

[2]. Thi Minh Chau Tran, Björn Scheuermann, Martin Mauve: Detecting the presence of nodes in MANETs. Challenged Networks 2007: 43-50

[3]. Thi Minh Chau Tran, Björn Scheuermann, Martin Mauve: Lightweight detection of node presence in MANETs. Ad Hoc Networks 7(7): 1386-1399 (2009)

[4]. http://en.wikipedia.org/wiki/Bloom_filter

[5]. http://www.ics.uci.edu/~keldefra/manet.htm

Các module xử lý

BloomFilter.java

```
import java.security.MessageDigest;
import java.util.Vector;

public class BloomFilter {

    private Vector bitset;// mang chua cac entry
    private int sizeofSet;// so entry dung de luu cac phan tu
    private int l;// so bit danh cho moi entry
    private int maxElements; // so phan tu lon nhat
    private int numberElements; // so phan tu add vao
    private int k; // so ham bam
    static MessageDigest digestFunction;// doi tuong Disgest

    public BloomFilter() {
    }

    public BloomFilter(int sizeofSet, int maxElements) {
        this.sizeofSet = sizeofSet;
        this.l = 3;//fix so bit moi entry la 3
        this.maxElements = maxElements;
        this.k = 4;//fix so ham bam la 4
        numberElements = 0;
        bitset = new Vector();
        //ban dau tat ca cac entry duoc thiet lap gia tri la 2^3-1=7
        for (int i = 0; i < sizeofSet; i++) {
            bitset.add(7);
        }
        try {
            digestFunction = MessageDigest.getInstance("MD5");//
khoi tao doi tuong Digest
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    /*
     * tao 1 hash value tu mang Byte
     */
    public static long createHash(String val) {
        return createHash(val.getBytes());
    }

    /*
     * tao 1 hash value tu mang Byte
     */
    public static long createHash(byte[] data) {
        long h = 0;
        byte[] res;
        digestFunction.update(data, 0, data.length);
```

```
        res = digestFunction.digest();
        for (int i = 0; i < 4; i++) {
            h <<= 8;
            h |= ((int) res[i]) & 0xFF;
        }
        return h;
    }

    /*
     * add a String o day la hostname
     */
    public Vector add(String valString) {
        long hash = 0;
        Vector hashArray = new Vector();

        for (int x = 0; x < k; x++) {
            hash = createHash(valString + Integer.toString(x));
            //hashArray.add(hash);
            hash = hash % (long) sizeofSet;
            hashArray.add(Math.abs((int) hash));
            //bitset[Math.abs((int) hash)] = 0;
            bitset.setElementAt(0, Math.abs((int) hash));
        }
        numberElements++;
        return hashArray;
    }

    /*
     * kiem tra xem nick do duoc add chua
     */
    public boolean contains(String valString) {
        long hash;
        for (int i = 0; i < k; i++) {
            hash = createHash(valString + Integer.toString(i));
            hash = hash % (long) sizeofSet;
            if (((Integer) bitset.get(Math.abs((int) hash)) == 7)) {
                return false;
            }
        }
        return true;
    }
    /*
     * Lay gia tri query
     */

    public Vector hopQuery(String queryString) {
        Vector vector = new Vector();
        if (contains(queryString)) {
            long hash;
            for (int i = 0; i < k; i++) {
                hash = createHash(queryString +
Integer.toString(i));
                hash = hash % (long) sizeofSet;
                vector.add(bitset.get(Math.abs((int) hash)));
```

```
            }
        }
        return vector;
    }
    /*
     * khoang cach
     */

    public int hop(String queryString) {
        int max = -1;
        Vector vector = hopQuery(queryString);
        if (vector.isEmpty()) {
            return -1;
        }
        for (int i = 0; i < vector.size(); i++) {
            int temp = (Integer) vector.get(i);
            if (temp > max) {
                max = temp;
            }
        }

        return max + 1;
    }
    /*
     * Lam gia thong tin
     * Nhung entry nao` da duoc set thi duoc tang them 1
     */

    public void decay() {
        for (int i = 0; i < bitset.size(); i++) {
            if ((Integer) bitset.get(i) < 7) {
                bitset.set(i, (Integer) bitset.get(i) + 1);
            }
        }
    }

    public void merge(Vector receive) {
        int size = receive.size();

        for (int i = 0; i < size; i++) {
            if ((Integer) bitset.get(i) > (Integer) receive.get(i))
{
                bitset.set(i, receive.get(i));
            }
        }
    }

    /*
     * so phan tu trong bloom
     */
    public int numberElement() {
        return numberElements;
    }
```

```
    /*
     * so phan tu lon nhat
     */
    public int maxElement() {
        return maxElements;
    }
    /*
     * So bit cho moi entry
     */

    public int getBitOfEntry() {
        return l;
    }

    public Vector getEntry() {
        return bitset;
    }

    public byte[] getByte() {

        return bitset.toString().getBytes();

    }

    public void setEntry(Vector temp) {
        bitset.clear();
        for (int i = 0; i < temp.size(); i++) {
            bitset.add(temp.get(i));
        }
    }
}
```

NinterfaceNames.java

```
import java.net.*;
import java.util.*;
import java.util.Enumeration;

public class NInterfaceNames {

    public NInterfaceNames() {
    }

    public Vector getNameInterfaces() {
        Vector vector = new Vector();
        try
        {
            Enumeration<NetworkInterface> en =
NetworkInterface.getNetworkInterfaces();
            while (en.hasMoreElements()) {
                NetworkInterface ni = en.nextElement();
```

```
                if (ni.isLoopback()) {
                    continue;
                }
                for (InterfaceAddress interfaceAddress :
                        ni.getInterfaceAddresses()) {
                            vector.add(ni.getName());
                        }

            }
        }
        catch(Exception e){}

        return vector;
    }
}
```

FriendList.java

```
import java.util.*;
import java.io.*;

public class FriendList {

    private Vector list;//danh sach cac Friend
    private int count;//so Friend

    public FriendList() {
        list = new Vector();
        count = 0;
    }

    public int getSize() {
        return this.count;
    }

    /*
     * Add new Friend
     */
    public void addFriend(Friend newFriend) {
        list.addElement(newFriend);
    }
    /*
     * Khi add them 1 friend moi thi ta thuc hien viec append vao
file friend.txt
     */

    public static void append(File aFile, String content) {
        try {
            PrintStream p = new PrintStream(new
BufferedOutputStream(new FileOutputStream(aFile, true)));
```

```
            p.println(content);
            p.close();

        } catch (Exception e) {
            e.printStackTrace();
            System.err.println(aFile);
        }
    }
    /*
     * doc file ra mang vector
     */

    public Vector getList(File file) {
        Vector vector = new Vector();
        FileInputStream fis = null;
        BufferedInputStream bis = null;
        DataInputStream dis = null;
        try {
            fis = new FileInputStream(file);

            bis = new BufferedInputStream(fis);
            dis = new DataInputStream(bis);

            while (dis.available() != 0) {
                String str = dis.readLine();
                //System.out.println(dis.readLine());
                Vector temp = new Vector();
                temp.add(str);
                vector.add(temp);
            }
        } catch (Exception e) {
        }
        return vector;
    }
}
```

InstantMessage.java

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.util.*;
import java.io.*;
import javax.swing.table.*;
import java.net.*;

public class InstantMessage extends JFrame {

    static boolean flag = false;//co cho vong lap Broadcast va
Receive
    JMenuBar menubar = new JMenuBar();
    JMenu menu = new JMenu("Bạn bè");
    JMenu menuContact = new JMenu("Liên hệ");
    JMenu menuHelp = new JMenu("Giúp đỡ");
    JMenuItem addFriend = new JMenuItem("Thêm bạn");
```

```
    JMenuItem contact = new JMenuItem("Liên hệ");
    JMenuItem help = new JMenuItem("Giúp đỡ");
    JButton jbStart = new JButton("Bắt đầu");
    JButton jbStop = new JButton("Tạm dừng");
    JPanel panelButton = new JPanel();
    JPanel panelTable = new JPanel();
    Vector colName = new Vector();//ten cac cot cua Table
    DefaultTableModel model;
    JTable table;
    operation oper = null;
    String interfaceName;
    java.util.Timer bkTimer;

    public InstantMessage(String title, final String interfaceName)
{
        super(title);
        this.interfaceName = interfaceName;
        menu.add(addFriend);
        menuHelp.add(help);
        menuContact.add(contact);
        menubar.add(menu);
        menubar.add(menuContact);
        menubar.add(menuHelp);
        this.setJMenuBar(menubar);

        table = new JTable();

        colName.add("Tên");
        colName.add("Khoảng cách");

        panelButton.setLayout(new GridBagLayout());
        GridBagConstraints grid = new GridBagConstraints();
        grid.fill = grid.BOTH;//can deu tat ca cac dong
        setBackground(new Color(205, 175, 142));

        grid.gridx = 0;
        grid.gridy = 0;
        grid.insets = new Insets(10, 10, 10, 10);
        panelButton.add(jbStart, grid);

        grid.gridx = 1;
        grid.gridy = 0;
        grid.insets = new Insets(10, 10, 10, 10);
        panelButton.add(jbStop, grid);

        this.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });

        //bat dau chay chuong trinh
        jbStart.addActionListener(new ActionListener() {
```

```
            public void actionPerformed(ActionEvent e) {
                flag = true;
                addFriend.setVisible(false);

                operation oper = new operation(interfaceName);

                if (bkTimer == null) {
                    bkTimer = new java.util.Timer();
                }
                bkTimer.schedule(oper, 200);
                try {
                    Thread.sleep(2100);
                } catch (InterruptedException exc) {
                    System.out.println(exc);
                }
                jbStart.setVisible(false);
            }
        });

        //bat su kien tam dung
        jbStop.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                if (bkTimer != null) {
                    bkTimer.cancel();
                    bkTimer = null;
                }
                flag = false;
                jbStart.setVisible(true);
            }
        });
        contact.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                new Contact("Contact Me");

            }
        });

        help.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                new Help("Help");

            }
        });

        //them ban be
        addFriend.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                String name = JOptionPane.showInputDialog("Nhập
tên");//Tao Dialog de nhap ten
```

```
                if (name != null) {//neu ten duoc nhap

                    Vector data1 = new FriendList().getList(new
File("friend.txt"));// doc file ra mot Vector
                    Vector temp1 = new Vector();
                    temp1.add(name);
                    data1.add(temp1);

                    //tao bang chua ten va ban dau gia tri khoang
cach la Offline
                    final JTable table2;//
                    for (int j = 0; j < data1.size(); j++) {
                        Vector temp = (Vector) data1.get(j);
                        temp.add("Offline");
                        data1.set(j, temp);

                    }
                    table2 = new JTable(data1, colName) {

                        public boolean isCellEditable(int rowIndex,
int colIndex) {
                            return false;
                        }
                    };
                    table2.setForeground(Color.BLUE);
                    table2.setOpaque(false);

                    JScrollPane pane2 = new JScrollPane(table2);
                    pane2.setPreferredSize(new Dimension(600, 200));

pane2.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_
ALWAYS);

pane2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWA
YS);
                    panelTable.add(pane2, BorderLayout.NORTH);
                    panelTable.repaint(0);
                    panelTable.updateUI();
                    panelTable.getParent().repaint(0);
                    FriendList list = new FriendList();
                    list.append(new File("friend.inp"), name);//them
vao danh sach
                }

                System.out.println(name);

            }
        });

        panelTable.removeAll();//remove tat cac cac thanh phan tren
Frame
        panelTable.repaint();
```

```
        Vector data = new FriendList().getList(new
File("friend.inp"));
        for (int i = 0; i < data.size(); i++) {
            Vector temp = (Vector) data.get(i);
            temp.add("Offline");
            data.set(i, temp);
        }
        model = new DefaultTableModel(data, colName);
        table = new JTable(model) {

            public boolean isCellEditable(int rowIndex, int
colIndex) {
                return false;

            }
        };
        table.setForeground(Color.BLUE);
        table.setOpaque(false);
        JScrollPane pane = new JScrollPane(table);
        pane.setPreferredSize(new Dimension(600, 200));
        panelTable.add(pane, BorderLayout.NORTH);


        getContentPane().add(panelTable, BorderLayout.SOUTH);
        getContentPane().add(panelButton, BorderLayout.NORTH);
        pack();
        setVisible(true);
        this.setResizable(false);
    }

//-----------------------------------------------
    //Thuc hien viec broadcast va receive lien tuc
//-----------------------------------------------
    class operation extends TimerTask {

        BloomFilter info;
        ListenerThread listener;
        BroadcasterThread broad;
        String interfaceName;

        public operation() {
            info = new BloomFilter(40, 10);//khoi tao BloomFilter
        }

        public operation(BloomFilter info) {
            this.info = info;
        }
        //Ham khoi tao voi doi la ten Interface

        public operation(String interfaceName) {
            info = new BloomFilter(40, 10);
            this.interfaceName = interfaceName;

        }
```

```
        public void run() {
            broad = new BroadcasterThread(interfaceName, info,
2000);//khoi tao Thread cho viec Broadcast
            listener = new ListenerThread(8000, info);//khoi tao
Thread cho viec lang nghe Broadcast
        }
    }

//------------------------------------------------
    // Thuc hien viec lien tu lang nghe thong tin broadcast tu cac
node hang xom
//------------------------------------------------
    class BroadcasterThread implements Runnable {

        String interfaceName;
        int sleepTime;
        BloomFilter info;
        Thread t;

        BroadcasterThread(String interfaceName, BloomFilter info,
int sleepTime) {
            this.sleepTime = sleepTime;
            this.info = info;
            this.interfaceName = interfaceName;
            t = new Thread(this, "Broadcaster Thread");
            t.start(); // Bat dau Thread
        }

        public String getBroadcastAddress() {
            String broadcastAddress = "";
            try {
                Enumeration<NetworkInterface> en =
NetworkInterface.getNetworkInterfaces();//duyet cac interface
                while (en.hasMoreElements()) {
                    NetworkInterface ni = en.nextElement();
                    if (ni.isLoopback()) { //bo qua loopback
                        continue;
                    }
                    for (InterfaceAddress interfaceAddress :
                            ni.getInterfaceAddresses()) {
                        //lay dia chi BroadCast cua Interface co ten
la interfaceName
                        if
(ni.getName().equalsIgnoreCase(interfaceName)) {
                            InetAddress broadcast =
interfaceAddress.getBroadcast();
                            broadcastAddress =
broadcast.getHostAddress();
                        }
                    }
                }
            } catch (Exception e) {
                System.out.print(e.getMessage());
```

```
            }
            return broadcastAddress;
        }
        /*
         * Lay hostname cua may
         */

        public String getHostName() {
            String hostname = "";
            try {
                InetAddress address =
InetAddress.getLocalHost();//lay inetAddress
                hostname = address.getHostName();//lay hostname
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
            return hostname;
        }

        public void run() {
            try {


                String hostName = getHostName();//hostname cua may
                info.add(hostName);//refresh
                //khoi tao DatagramSocket
                final DatagramSocket datagramSocket = new
DatagramSocket();
                datagramSocket.setBroadcast(true);

                // Broadcast address
                final InetAddress broadcastAddress =

InetAddress.getByName(getBroadcastAddress());

                try {
                    while (flag) {
                        // Tao 1 DatagraPacket de Broadcast o port
la 8000
                        DatagramPacket outDataPacket; // Datagram
packet to the server
                        byte[] data = info.getByte();
                        outDataPacket =
                                new DatagramPacket(data,
data.length, broadcastAddress, 8000);
                        // broadcast.
                        datagramSocket.send(outDataPacket);
                        System.out.println("send");

                        Thread.currentThread().sleep(sleepTime);//
thoi gian TimeOut
                    }
                } catch (Exception e) {
                    e.printStackTrace();
```

```
                } finally {
                    //dong socket
                    datagramSocket.close();
                }

            } catch (Exception e) {
                System.out.println("Exception occured with
socket.");
                System.out.println(e);
            }

        }
    }
//---------------------------------------
//---------------------------------------

    class ListenerThread implements Runnable {

        Thread t;
        BloomFilter info;
        final private int listeningPort;

        ListenerThread(int port, BloomFilter info) {
            listeningPort = port;
            this.info = info;
            t = new Thread(this, "Listener Thread");
            t.start();
        }

        // Vong lap de lang nghe
        public void run() {
            try {
                DatagramSocket datagramSocket;
                // Datagram packet
                DatagramPacket inDataPacket;
                //  buffer.
                byte[] msg = new byte[256];

                // khoi tao DatagramSocket voi port la
listeningPort.
                datagramSocket = new DatagramSocket(listeningPort);


                // Loop forever
                while (flag) {

                    // Khoi tao DatagramPacket cho
                    inDataPacket = new DatagramPacket(msg,
msg.length);

                    // Nhan thong tin.
                    datagramSocket.receive(inDataPacket);//cho thong
tin
```

```
                    byte[] receive = inDataPacket.getData();//nhan
du lieu
                    String rec = new String(receive, 0,
inDataPacket.getLength());
                    Vector bitSetReceive = getEntry(rec);//lay noi
dung bitSet tu thong tin nhan duoc
                    info.merge(bitSetReceive);//merge

                    //thuc hien viec update bang thong tin
                    panelTable.removeAll();//remove bang
                    panelTable.repaint();
                    Vector data1 = new FriendList().getList(new
File("friend.inp"));// lay danh sach ten
                    final JTable table1;//khoi tao bang moi
                    //kiem tra gia tri moi ten trong file friend.inp
xem no co ton tai hay khong
                    //neu ton tai thi khoang cach la bao nhie
                    for (int j = 0; j < data1.size(); j++) {
                        Vector temp = (Vector) data1.get(j);
                        String str = (String) temp.get(0);
                        int hop = info.hop(str);//khoang cach
                        if (hop == -1) {
                            temp.add("offline");
                        } else {
                            temp.add(hop);
                        }
                        data1.set(j, temp);

                    }
                    //tao bang moi khong cho phep sua noi dung
                    table1 = new JTable(data1, colName) {

                        public boolean isCellEditable(int rowIndex,
int colIndex) {
                            return false;
                        }
                    };
                    table1.setForeground(Color.BLUE);
                    table1.setOpaque(false);

                    //tao ScrollPane
                    JScrollPane pane1 = new JScrollPane(table1);
                    pane1.setPreferredSize(new Dimension(600, 200));
                    panelTable.add(pane1, BorderLayout.NORTH);

                    panelTable.updateUI();

                }
            } catch (IOException e) {
                System.out.println("IOException occured with
socket.");
                System.out.println(e);

            }
```

```
        }

        //lay noi dung bitSet tu thong tin nhan duoc la mot chuoi
        public Vector getEntry(String rec) {
            String a = rec.substring(1, rec.length() - 1);
            Vector b = new Vector();
            int j = 0;

            //thuc hien viec tach chuoi de lay cac so luu vao 1
Vector
            for (int i = 0; i < a.length(); i++) {

                if (rec.charAt(i) == ',') {
                    b.add(a.substring(j, i - 1));
                    j = i + 1;
                    if (j == a.length() - 1) {
                        b.add(a.substring(j, j + 1));
                    }
                }
            }
            Vector c = new Vector();
            for (int i = 0; i < b.size(); i++) {
                String s1 = (String) b.elementAt(i);
                c.add(Integer.parseInt(s1));

            }
            return c;
        }
    }
}
```

Contact.java

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Contact extends JFrame
{
    JLabel name=new JLabel("Họ tên: Hoàng Anh Dũng");
    JLabel phoneNumber = new JLabel("Số điện thoại:
0988352716");
    JLabel email = new
JLabel("Email:Hoang_dung1171987@yahoo.com");

    public Contact(String title)
    {
        super(title);

        this.setLayout(new GridBagLayout());
        GridBagConstraints grid = new GridBagConstraints();
```

```
        grid.fill = grid.BOTH;//can deu tat ca cac dong
        this.setBackground(new Color(205, 175, 142));

        grid.gridx = 0;
        grid.gridy = 0;
        grid.insets = new Insets(10, 10, 10, 10);
        getContentPane().add(name,grid);

        grid.gridx = 0;
        grid.gridy = 1;
        grid.insets = new Insets(10, 10, 10, 10);
        getContentPane().add(phoneNumber,grid);

        grid.gridx = 0;
        grid.gridy = 2;
        grid.insets = new Insets(10, 10, 10, 10);
        getContentPane().add(email,grid);

        setSize(350,200);
        setVisible(true);

        this.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });
        this.setResizable(false);
    }
}
```

Help.java

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class Help extends JFrame
{
    JLabel help1 =new JLabel("Sử dụng Bạn bè để thêm bạn.");
    JLabel help2=new JLabel("Sử dụng Start để bắt đầu tìm kiếm.");
    JLabel help3=new JLabel("Sử dụng Tạm dừng để dừng chương
trình.");

    public Help(String title)
    {
        super(title);

        this.setLayout(new GridBagLayout());
        GridBagConstraints grid = new GridBagConstraints();
        grid.fill = grid.BOTH;//can deu tat ca cac dong
```

```
        this.setBackground(new Color(205, 175, 142));

        grid.gridx = 0;
        grid.gridy = 0;
        grid.insets = new Insets(10, 10, 10, 10);
        getContentPane().add(help1,grid);

        grid.gridx = 0;
        grid.gridy = 1;
        grid.insets = new Insets(10, 10, 10, 10);
        getContentPane().add(help2,grid);

        grid.gridx = 0;
        grid.gridy = 2;
        grid.insets = new Insets(10, 10, 10, 10);
        getContentPane().add(help3,grid);

        setSize(350,200);
        setVisible(true);
        this.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });
    }
}
```

index.java

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.util.*;

public class index extends JFrame {

    JLabel txtChoice = new JLabel("Chọn Interface");
    JComboBox jcbInterface = new JComboBox();
    JButton jbOk = new JButton("Chọn");

    public index(String title) {
        super(title);
        Vector vc = new NInterfaceNames().getNameInterfaces();
        if (!vc.isEmpty()) {
            for (int i = 0; i < vc.size(); i++) {
                jcbInterface.addItem((String) vc.get(i));
            }
        }
        setLayout(new GridBagLayout());
        GridBagConstraints grid = new GridBagConstraints();
        grid.fill = grid.BOTH;//can deu tat ca cac dong
        setBackground(new Color(205, 175, 142));

        grid.gridx = 0;
```

```
        grid.gridy = 0;
        grid.insets = new Insets(10, 10, 10, 10);
        this.getContentPane().add(txtChoice, grid);

        grid.gridx = 1;
        grid.gridy = 0;
        grid.insets = new Insets(10, 10, 10, 10);
        this.getContentPane().add(jcbInterface, grid);

        grid.gridx = 1;
        grid.gridy = 1;
        grid.insets = new Insets(10, 10, 10, 10);
        this.getContentPane().add(jbOk, grid);

        this.addWindowListener(new WindowAdapter() {

            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });
        jbOk.addActionListener(new ActionListener() {

            public void actionPerformed(ActionEvent e) {
                String interfaceName = (String)
jcbInterface.getSelectedItem();
                new InstantMessage("Instant Message",
interfaceName);
                dispose();
            }
        });
        setSize(400, 200);
        setVisible(true);
        this.setResizable(false);
    }

    public static void main(String[] args) {
        new index("Instant Message");
    }
}
```