­­

**ĐẠI HỌC ĐÀ NẴNG  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA CÔNG NGHỆ THÔNG TIN  
\_\_\_\_\*\*\*\_\_\_\_**



**BÁO CÁO ĐỒ ÁN**

**CƠ SỞ NGÀNH MẠNG**

**ĐỀ TÀI:**

*Phần Nguyên lý hệ điều hành:*

**Bài toán Năm triết gia ăn tối**

*Phần Lập trình mạng:*

**Sử dụng Socket trong Java xây dựng ứng dụng chia sẻ file theo mô hình Peer-to-Peer**

**Giáo viên hướng dẫn : ThS. Trần Hồ Thủy Tiên**

**Sinh viên thực hiện : Nguyễn Đình Hoan**

**MSSV : 102150034**

**Lớp : 15T1**

**Nhóm : 15.11A**

**Đà Nẵng, tháng 12 năm 2018**

**Đà Nẵng, ngày 01 tháng 12 năm 2017**

**Đa**

# MỤC LỤC

[MỤC LỤC 1](#_Toc533168358)

[DANH MỤC HÌNH ẢNH 3](#_Toc533168359)

[DANH MỤC BẢNG BIỂU 4](#_Toc533168360)

[LỜI MỞ ĐẦU 5](#_Toc533168361)

[PHẦN I – NGUYÊN LÝ HỆ ĐIỀU HÀNH 6](#_Toc533168362)

[TIÊU ĐỀ: BÀI TOÁN NĂM TRIẾT GIA ĂN TỐI 6](#_Toc533168363)

[TỔNG QUAN VỀ ĐỀ TÀI 6](#_Toc533168364)

[1. Bối cảnh và lý do thực hiện đề tài 6](#_Toc533168365)

[2. Kết cấu của đồ án 6](#_Toc533168366)

[Chương 1. CƠ SỞ LÝ THUYẾT 7](#_Toc533168367)

[1.1. Tiến trình 7](#_Toc533168368)

[1.1.1. Khái niệm 7](#_Toc533168369)

[1.1.2. Các loại tiến trình 7](#_Toc533168370)

[1.1.3. Tuyến 7](#_Toc533168371)

[1.2. Tài nguyên găng và đoạn găng 8](#_Toc533168372)

[1.2.1. Tài nguyên găng (Critical Resource) 8](#_Toc533168373)

[1.2.2. Đoạn găng (Critical Section) 9](#_Toc533168374)

[1.2.3. Yêu cầu của công tác điều độ đoạn găng 9](#_Toc533168375)

[1.3. Giải pháp điều độ tiến trình dùng Semaphore 10](#_Toc533168376)

[1.4. Tắc nghẽn (Deadlock) và chống tắc nghẽn 10](#_Toc533168377)

[1.4.1. Tắc nghẽn 10](#_Toc533168378)

[1.4.2. Điều kiện hình thành tắc nghẽn 11](#_Toc533168379)

[1.4.3. Ngăn chặn tắc nghẽn (Deadlock Prevention) 11](#_Toc533168380)

[Chương 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 13](#_Toc533168381)

[2.1. Phân tích vấn đề 13](#_Toc533168382)

[2.2. Giải quyết vấn đề 13](#_Toc533168383)

[2.2.1. Quản lý vùng găng 13](#_Toc533168384)

[2.2.2. Giải pháp ngăn chặn deadlock 14](#_Toc533168385)

[2.2.3. Chương trình 15](#_Toc533168386)

[2.2.3.1. Class Philosopher 15](#_Toc533168387)

[2.2.3.2. Class Chopstick 16](#_Toc533168388)

[2.2.3.3. Class Dinner 16](#_Toc533168389)

[2.2.3.4. Class DinnerGUI 17](#_Toc533168390)

[2.2.3.5. Enum Status 17](#_Toc533168391)

[Chương 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 19](#_Toc533168392)

[3.1. Ngôn ngữ cài đặt 19](#_Toc533168393)

[3.2. Kết quả chương trình 19](#_Toc533168394)

[3.3. Đánh giá kết quả 19](#_Toc533168395)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 21](#_Toc533168396)

[4.1. Kết luận 21](#_Toc533168397)

[4.2. Hướng phát triển 21](#_Toc533168398)

[PHẦN II – MẠNG MÁY TÍNH 22](#_Toc533168399)

[TIÊU ĐỀ: SỬ DỤNG SOCKET TRONG JAVA XÂY DỰNG ỨNG DỤNG CHIA SẺ FILE THEO MÔ HÌNH PEER-TO-PEER 22](#_Toc533168400)

[TỔNG QUAN VỀ ĐỀ TÀI 22](#_Toc533168401)

[1. Bối cảnh và lý do thực hiện đề tài 22](#_Toc533168402)

[2. Kết cấu của đồ án 22](#_Toc533168403)

[Chương 1. CƠ SỞ LÝ THUYẾT 23](#_Toc533168404)

[1.1. Căn bản về mạng máy tính 23](#_Toc533168405)

[1.1.1. Định nghĩa về mạng máy tính 23](#_Toc533168406)

[1.1.2. Các giao thức mạng 23](#_Toc533168407)

[1.1.2.1. Giao thức TCP 23](#_Toc533168408)

[1.1.2.2. Giao thức UDP 23](#_Toc533168409)

[1.1.3. Các mô hình hoạt động của mạng máy tính 24](#_Toc533168410)

[1.1.3.1. Mô hình Trạm – Chủ (Client – Server) 24](#_Toc533168411)

[1.1.3.2. Mô hình mạng ngang hàng (Peer-to-Peer) 25](#_Toc533168412)

[1.2. Socket và Socket trong Java 26](#_Toc533168413)

[1.2.1. Khái niệm Socket 26](#_Toc533168414)

[1.2.2. Các loại Socket 26](#_Toc533168415)

[1.2.2.1. Socket có hướng kết nối (Stream Socket) 26](#_Toc533168416)

[1.2.2.2. Socket không hướng kết nối 27](#_Toc533168417)

[1.2.3. Socket trong Java 28](#_Toc533168418)

[1.2.3.1. Một số lớp trong lập trình Java Socket 28](#_Toc533168419)

[1.2.3.2. Lớp ServerSocket 29](#_Toc533168420)

[1.2.3.3. Lớp Socket 30](#_Toc533168421)

[1.2.3.4. Lớp InetAddress 32](#_Toc533168422)

[Chương 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 33](#_Toc533168423)

[2.1. Phân tích yêu cầu 33](#_Toc533168424)

[2.2. Mô tả chương trình 33](#_Toc533168425)

[2.3. Phân tích thiết kế 34](#_Toc533168426)

[2.3.1. Chương trình Main Server 34](#_Toc533168427)

[2.3.2. Chương trình Client/Mini Server 35](#_Toc533168428)

[2.3.2.1. Chức năng truy vấn thông tin với Main Server 36](#_Toc533168429)

[2.3.2.2. Chức năng tải file và truyền file cho người dùng khác 38](#_Toc533168430)

[2.3.2.3. Tổng quan hoạt động của chương trình Client/Mini Server 39](#_Toc533168431)

[Chương 3. TRIỂN KHAI VÀ ĐÁNH GIÁ KẾT QUẢ 41](#_Toc533168432)

[3.1. Ngôn ngữ cài đặt 41](#_Toc533168433)

[3.2. Kết quả chương trình 41](#_Toc533168434)

[3.2.1. Giải thích một số control trên giao diện 41](#_Toc533168435)

[3.2.2. Hình ảnh kết quả chương trình 42](#_Toc533168436)

[3.3. Đánh giá kết quả 45](#_Toc533168437)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 46](#_Toc533168438)

[4.1. Kết luận 46](#_Toc533168439)

[4.2. Hướng phát triển 46](#_Toc533168440)

[KẾT LUẬN CHUNG 47](#_Toc533168441)

[TÀI LIỆU THAM KHẢO 48](#_Toc533168442)

# DANH MỤC HÌNH ẢNH

*Hình 1. Mô tả bài toán Năm triết gia ăn tối 13*

*Hình 2. Giao diện chương trình bài toán Năm triết gia ăn tối 19*

*Hình 3. Mô hình Client – Server 25*

*Hình 4. Mô hình mạng ngang hàng 25*

*Hình 5. Cơ chế gọi hàm trong lập trình Socket có hướng kết nối 27*

*Hình 6. Cơ chế gọi hàm trong lập trình Socket không hướng kết nối 28*

*Hình 7. Mô hình mạng đồng đẳng lai 33*

*Hình 8. Sơ đồ tổng thể tương tác của hệ thống 34*

*Hình 9. Sơ đồ Use case đối với người dùng ứng dụng 36*

*Hình 10. Sơ đồ hoạt động của ứng dụng 39*

*Hình 11. Giao diện cơ bản của chương trình 40*

*Hình 12. Hình ảnh demo – Giao diện kết nối với Main Server 41*

*Hình 13. Hình ảnh demo – Giao diện chính của chương trình 41*

*Hình 14. Hình ảnh demo – Giao diện khi nhận yêu cầu tải file từ người khác 42*

*Hình 15. Hình ảnh demo – Giao diện khi download file 42*

*Hình 16. Hình ảnh demo – Giao diện khi download thành công 43*

*Hình 17. Hình ảnh demo – Trường hợp người sở hữu file từ chối yêu cầu 43*

*Hình 18. Hình ảnh demo – Trường hợp file download trùng tên với một file trong*

*thư mục muốn lưu 44*

*Hình 19. Hình ảnh demo – Giao diện Main Server tương ứng 44*

DANH MỤC BẢNG BIỂU

*Bảng 1. So sánh giao thức TCP và UDP 24*

*Bảng 2. So sánh hai mô hình mạng Client – Server và Peer-to-Peer 26*

*Bảng 3. Các hàm dựng của lớp ServerSocket 29*

*Bảng 4. Một số phương thức của lớp ServerSocket 30*

*Bảng 5. Các hàm dựng của lớp Socket 31*

*Bảng 6. Một số phương thức của lớp Socket 31*

*Bảng 7. Một số phương thức của lớp InetAddress 32*

# LỜI MỞ ĐẦU

Đồ án các môn học chính của Khoa Công nghệ thông tin nói chung và Đồ án Cơ sở ngành mạng nói riêng nhằm tạo điều kiện cơ bản giúp cho sinh viên:

* Đi sâu và nắm vững một cách có hệ thống kiến thức đã thu nhận được trong quá trình học lý thuyết, làm bài tập và thực hành;
* Từng bước làm quen với các công tác khoa học có định hướng của giáo viên hướng dẫn và hình thành hành vi nghiên cứu độc lập có sự trợ giúp của tài liệu tham khảo;
* Gắn quá trình học lý thuyết với công tác nghiên cứu thực tế;
* Trình bày rõ ràng và khoa học một vấn đề thuộc lĩnh vực nghiên cứu của mình.

Qua quá trình làm đồ án Cơ sở ngành mạng, em đã có được thêm nhiều kiến thức mới, đồng thời củng cố vững chắc kiến thức đã được học trên trường.

Em xin chân thành cảm ơn cô Trần Hồ Thủy Tiên đã tận tình hướng dẫn để em có thể hoàn thành được đồ án này. Quá trình làm đồ án sẽ không thể tránh khỏi những sai sót, mong cô có thể thông cảm và bỏ qua.

Một lần nữa em xin chân thành cảm ơn.

Sinh viên  
Nguyễn Đình Hoan

PHẦN I – NGUYÊN LÝ HỆ ĐIỀU HÀNH

TIÊU ĐỀ: BÀI TOÁN NĂM TRIẾT GIA ĂN TỐI

TỔNG QUAN VỀ ĐỀ TÀI

1. Bối cảnh và lý do thực hiện đề tài

Trong môi trường hệ điều hành đa nhiệm – đa chương – đa người sử dụng, việc chia sẻ tài nguyên cho các tiến trình của người sử dụng dùng chung là cần thiết, nhưng nếu hệ điều hành không tổ chức tốt việc sử dụng tài nguyên dùng chung của các tiến trình hoạt động đồng thời, thì không những không mang lại hiểu quả khai thác tài nguyên của hệ thống mà còn làm hỏng dữ liệu của các ứng dụng. Và nguy hiểm hơn là việc hỏng dữ liệu này có thể hệ điều hành và ứng dụng không thể phát hiện được. Việc hỏng dữ liệu của ứng dụng có thể làm sai lệch ý nghĩa thiết kế của nó. Đây là điều mà cả hệ điều hành và người lập trình đều không mong muốn.

Các tiến trình hoạt động đồng thời thường cạnh tranh với nhau trong việc sử dụng tài nguyên dùng chung. Hai tiến trình hoạt động đồng thời cùng ghi vào một không gian nhớ chung (một biến chung) trên bộ nhớ hay hai tiến trình đồng thời cùng ghi dữ liệu vào một file chia sẻ, đó là những biểu hiện của sự cạnh tranh về việc sử dụng tài nguyên dùng chung của các tiến trình. Để các tiến trình hoạt động đồng thời không cạnh tranh hay xung đột với nhau khi sử dụng tài nguyên dùng chung hệ điều hành phải tổ chức cho các tiến trình này được độc quyền truy xuất sử dụng trên các tài nguyên dùng chung này.

Bài toán Năm triết gia ăn tối là bài toán kinh điển về tương tranh và chia sẻ tài nguyên dùng chung. Việc nghiên cứu bài toán giúp ta hiểu thêm về khía cạnh này của hệ điều hành.

1. Kết cấu của đồ án

* Tìm hiểu về tiến trình và các khái niệm liên quan.
* Tìm hiểu về tài nguyên găng, đoạn găng và công tác điều độ đoạn găng, áp dụng đối với bài toán Năm triết gia ăn tối.
* Tìm hiểu deadlock và cách ngăn chặn deadlock, áp dụng đối với bài toán Năm triết gia ăn tối.
* Xây dựng chương trình giải quyết bài toán Năm triết gia ăn tối.

1. CƠ SỞ LÝ THUYẾT
   1. Tiến trình
      1. Khái niệm

Tiến trình là một chương trình đang xử lý, sỡ hữu một con trỏ lệnh, tập các thanh ghi và các biến. Để hoàn thành tác vụ của mình, một tiến trình có thể cần đến một số tài nguyên – như CPU, bộ nhớ chính, các tập tin và thiết bị nhập/xuất.

Cần phân biệt hai khái niệm chương trình và tiến trình. Một chương trình là một thực thể thụ động, chứa đựng các chỉ thị điều khiển máy tính để tiến hành một tác vụ nào đó ; khi cho thực hiện các chỉ thị này, chương trình chuyển thành tiến trình, là một thực thể hoạt động, với con trỏ lệnh xác định chỉ thị kế tiếp sẽ thi hành, kèm theo tập các tài nguyên phục vụ cho hoạt động của tiến trình.

* + 1. Các loại tiến trình

Có thể chia tiến trình thành hai loại: tiến trình tuần tự và tiến trình song song:

* Tiến trình tuần tự: là các tiến trình mà điểm khởi tạo của nó là điểm kết thúc của tiến trình trước đó.
* Tiến trình song song : là các tiến trình mà điểm khởi tạo của tiến trình này mằn ở thân của các tiến trình khác, tức là có thể khởi tạo một tiến trình mới khi các tiến trình trước đó chưa kết thúc.
  + 1. Tuyến

Tuyến là một thành phần của tiến trình sở hữu ngăn xếp và thực thi độc lập ngay trong mã lệnh của tiến trình. Nếu như hệ điều hành có nhiều tiến trình thì trong mỗi tiến trình bạn có thể tạo ra nhiều tuyến hoạt động song song trong hệ điều hành.

Điểm quan trọng nhất cần chú ý là một tuyến có thể làm bất cứ nhiệm vụ gì một tiến trình có thể làm. Tuy nhiên, vì một tiến trình có thể chứa nhiều tuyến, mỗi tuyến có thể coi như là một tiến trình nhỏ. Vậy, điểm khác biệt mấu chốt giữa tuyến và tiến trình là công việc mỗi cái thường phải làm.

Một điểm khác biệt nữa đó là nhiều tuyến nằm trong cùng một tiến trình dùng một không gian bộ nhớ giống nhau, trong khi tiến trình thì không. Điều này cho phép các tuyến đọc và viết cùng một kiểu cấu trúc và dữ liệu, giao tiếp dễ dàng giữa các tuyến với nhau. Giao thức giữa các tiến trình, hay còn gọi là IPC (inter-process communication) thì tương đối phức tạp bởi các dữ liệu có tính tập trung sâu hơn.

Cũng vì các tuyến dùng chung vùng nhớ và không gian địa chỉ của tiến trình, nên điểm yếu của việc dùng tuyến đó là khả năng đổ vỡ của tuyến sẽ ảnh hưởng đến tất cả các tuyến khác và toàn bộ tuyến trình đang hoạt động. Ngược lại, một tiến trình bị đổ vỡ luôn được hệ điều hành cô lập hoàn toàn không gây ảnh hưởng đến các tiến trình khác. Tiến trình có thể chạy trên nhiều máy khác nhau trong khi tuyến chỉ được thực thi trên một máy và trong một tiến trình.

* 1. Tài nguyên găng và đoạn găng
     1. Tài nguyên găng (Critical Resource)

Những tài nguyên được hệ điều hành chia sẻ cho nhiều tiến trình hoạt động đồng thời dùng chung, mà có nguy cơ dẫn đến sự tranh chấp giữa các tiến trình này khi sử dụng chúng, được gọi là tài nguyên găng. Tài nguyên găng có thể là tài nguyên phần cứng hoặc tài nguyên phần mềm, có thể là tài nguyên phân chia được hoặc không phân chia được, nhưng đa số thường là tài nguyên không phân chia được như là: các biến chung, các file chia sẻ…

Ví dụ sau cho thấy hậu quả của việc sử dụng tài nguyên găng trong các chương trình có các tiến trình hoạt động đồng thời:

Ví dụ: Giả sử có một chương trình, trong đó có hai tiến trình P1 và P2 hoạt động đồng thời với nhau. Tiến trình P1 phải tăng biến Count lên 1 đơn vị, tiến trình P2 tăng biến Count lên 1 đơn vị, với mục đích tăng Count lên được 2 đơn vị.

Chương trình có thể thực hiện như sau:

* Tiến trình P1 ghi nội dung biến toàn cục Count vào biến cục bộ L1
* Tiến trình P2 ghi nội dung biến toàn cọc Count vào biến cục bộ L2
* Tiến trình P1 thực hiện L1:=L1 + 1 và Count:=L1
* Tiến trình P2 thực hiện L2:=L2 + 1 và Count:=L2

Như vậy thoạt nhìn ta thấy rằng Count đã được tăng 2 đơn vị, nhưng trong thực tế có thể Count chỉ được tăng 1 đơn vị. Bởi vì nếu P1 và P2 đồng thời nhận giá trị của Count (Giả sử ban đầu Count = 4) vào L1 và L2, sau đó P1 tăng L1 lên 1 và P2 tăng L2 lên 1 (L1 = 5, L2 = 5), rồi sau đó cả P1 và P2 đồng thời ghi giá trị biến L của nó vào lại Count, thì Count chỉ tăng được 1 đơn vị, Count = 5. Đây là điều mà chương trình không mong muốn nhưng cả chương trình và hệ điều hành đều khó có thể phát hiện được.

Nguyên nhân ở trên là do 2 tiến trình P1 và P2 đồng thời truy xuất biến Count, cả khi nhận giá trị của Count, lẫn khi gán giá trị vào Count. Trong trường hợp này nếu hệ điều hành không cho phép hai tiến trình đồng thời truy xuất Count, hoặc hệ điều hành cho phéo mỗi tiến trình được độc quyền truy xuất Count trong đoạn code sau, thì lỗi sẽ không xảy ra:

P1: Begin

L1 := Count;

L1 := L1 + 1;

Count := L1;

End;

P2: Begin

L2 := Count;

L2 := L2+ 1;

Count := L2;

End;

Trong ví dụ này **tài nguyên găng** là biến Count.

* + 1. Đoạn găng (Critical Section)

Đoạn code trong các tiến trình đồng thời, có tác động đến các tài nguyên có thể trở thành tài nguyên găng được gọi là đoạn găng hay miền găng. Tức là, các đoạn code trong các chương trình dùng để truy cập đến các vùng nhớ chia sẻ, các tập tin chia sẻ được gọi là các đoạn găng.

Trong ví dụ ở mục 2.1, có 2 đoạn găng là:

{ L1 := Count và Count := L1 }

Để hạn chế các lỗi có thể xảy ra do sử dụng tài nguyên găng, hệ điều hành phải điều khiển các tiến trình sao cho, tại một thời điểm chỉ có một tiến trình nằm trong đoạn găng, nếu có nhiều tiến trình cùng muốn vào (thực hiện) đoạn găng thì chỉ có một tiến trình được vào, các tiến trình khác phải chờ, một tiến trình khi ra khỏi (kết thúc) đoạn găng phải báo cho hệ điều hành và/hoặc các tiến trình khác biết để các tiến trình này vào đoạn găng,… Các công tác điều khiển tiến trình thực hiện đoạn găng của hệ điều hành được gọi là điều độ tiến trình qua đoạn găng. Để công tác điều độ tiến trình qua đoạn găng được thành công, thì cần phải có sự phối hợp giữa vi xử lý, hệ điều hành và người lập trình. Vi xử lý đưa ra các chỉ thị, hệ điều hành cung cấp các công cụ để người lập trình xây dựng các sơ đồ điều độ hợp lý, để đảm bảo sự độc quyền trong việc sử dụng tài nguyên găng của các tiến trình. Tuy nhiên, dù cung cấp đầy đủ các công cụ điều độ tiến trình cho người lập trình, nhưng các hệ điều hành hiện nay đều tổ chức điều độ tiến trình ngay trong lõi (kernel), nên người lập trình ít quan tâm đến tổ chức điều độ tiến trình khi lập trình.

* + 1. Yêu cầu của công tác điều độ đoạn găng

Trước hết cần lưu ý lại rằng, nhiệm vụ điều độ tiến trình phải là sự phối hợp giữa phần cứng vi xử lý, hệ điều hành, ngôn ngữ lập trình và người lập trình; trong đó nhiệm vụ chính là của hệ điều hành và người lập trình. Cho dù nhiệm vụ điều độ là của thành phần nào, thì tất cả phải đạt được các yêu cầu sau:

* Tại một thời điểm không thể có hai tiến trình nằm trong đoạn găng.
* Nếu có nhiều tiến trình đồng thời cùng xin được vào đoạn găng thì chỉ có một tiến trình được phép vào đoạn găng, các tiến trình khác phải xếp hàng chờ trong hàng đợi.
* Tiến trình chờ ngoài đoạn găng không được ngăn cản các tiền trình khác vào đoạn găng.
* Không có tiến trình nào được phép ở lâu vô hạn trong đoạn găng và không có tiến trình phải chờ lâu mới được vào đoạn găng (chờ trong hàng đợi).
* Nếu tài nguyên găng được giải phóng thì hệ điều hành có nhiệm vụ đánh thức các tiến trình trong hàng đợi ra để tạo điều kiện cho nó vào đoạn găng.

Nguyên lý cơ bản của điều độ là tổ chức truy xuất độc quyền trên tài nguyên găng, nhưng sự bắt buộc độc quyền này còn tồn tại hai hạn chế lớn:

* Có thể dẫn tới tắc nghẽn (Deadlock) trong hệ thống.
* Các tiến trình có thể bị đói (Stravation) tài nguyên.
  1. Giải pháp điều độ tiến trình dùng Semaphore

Giải pháp này được Dijkstra đề xuất vào năm 1965. Semaphore được định nghĩa để sử dụng trong các sơ đồ điều độ như sau:

* Semaphore S là một biến nguyên, khởi gán bằng một giá trị không âm, đó là khả năng phục vụ của tài nguyên găng tương ứng với nó.
* Ứng với S có một hàng đợi F(s) để lưu các tiến trình đang bị blocked trên S.
* Chỉ có hai thao tác Down và Up được tác động đến Semaphore S. Down giảm S xuống 1 đơn vị, Up tăng S lên 1 đơn vị.
* Mỗi tiến trình trước khi vào đoạn găng thì phải gọi Down để kiểm tra và xác lập quyền vào đoạn găng. Khi tiến trình gọi Down(S) thì hệ thống sẽ thực hiện như sau: S := S – 1, nếu S >=0 thì tiến trình tiếp tục xử lý và vào đoạn găng, nếu S < 0 thì tiến trình phải vào hàng đợi để chờ cho đến khi S >= 0.
* Mỗi tiến trình ngay sau khi ra khỏi đoạn găng phải gọi Up để kiểm tra xem có tiến trình nào đang đợi ở trong hàng đợi hay không, nếu có thì đưa tiến trình trong hàng đợi vào đoạn găng.

Ở đây cần lưu ý rằng: Down và Up là các thủ tục của hệ điều hành, nên hệ điều hành đã được cài đặt cơ chế độc quyền cho nó, tức là các lệnh bên trong nó không thể tách rời nhau.

* 1. Tắc nghẽn (Deadlock) và chống tắc nghẽn
     1. Tắc nghẽn

Tắc nghẽn là hiện tượng trong hệ thống xuất hiện một tập các tiến trình, mỗi tiến trình trong tập này đều chờ được cấp tài nguyên, mà tài nguyên đó đang được một tiến trình khác trong tập này chiếm giữ. Và sự đợi này có thể kéo dài vô hạn nếu không có sự tác động bên ngoài.

Nói cách khác, mỗi tiến trình trong tập hợp đều chờ được cấp phát một tài nguyên hiện đang bị một tiến trình khác cũng ở trạng thái blocked chiếm giữ. Như vậy không có tiến trình nào có thể tiếp tục xử lý , cũng như giải phóng tài nguyên cho tiến trình khác sử dụng, tất cả các tiến trình trong tập hợp đều bị khóa vĩnh viễn.

Trong trường hợp của bài toán là khi tất cả các triết gia đều đói cùng một lúc, tất cả cùng nhấc chiếc nĩa bên tay phải của mình, và cùng chờ đợi chiếc nĩa từ hàng xóm bên tay trái dẫn đến các tiến trình bị khóa chết.

* + 1. Điều kiện hình thành tắc nghẽn

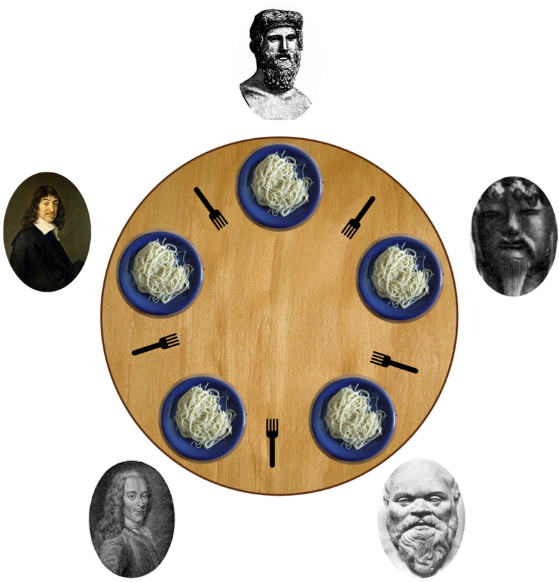
Năm 1971, Coffman đã đưa ra và chứng tỏ được rằng, nếu hệ thống tồn tại đồng thời bốn điều kiện sau đây thì hện thống sẽ xảy ra tắc nghẽn:

* Loại trừ lẫn nhau (mutual excution) hay độc quyền sử dụng: Đối với các tài nguyên không phân chia được thì tại mỗi thời điểm chỉ có một tiến trình sử dụng tài nguyên.
* Giữ và đợi (hold and wait): Một tiến trình hiện tại đang giữ tài nguyên, lại xin cấp phát thêm tài nguyên mới.
* Không ưu tiên (No preemption): Không có tài nguyên nào có thể được giải phóng từ một tiến trình đang chiến giữ nó.
* Sự tắc nghẽn có thể tồn tại với ba điều kiện trên, nhưng cũng có thể không xảy ra chỉ với ba điều kiện đó. Để chắc chắn tắc nghẽn xảy ra cần phải có điều kiện thứ tư:
* Đợi vòng tròn (Circular wait): Đây là trường hợp của ví dụ 1 mà chúng ta đã nêu ở trên. Tức là, mỗi tiến trình đang chiếm giữ tài nguyên mà tiến trình khác đang cần.
* Ba điều kiện đầu là điều kiện cần chứ không phải điều kiện đủ để xảy ra tắc nghẽn. Điều kiện thứ tư là kết quả tất yếu từ ba điều kiện đầu.
  + 1. Ngăn chặn tắc nghẽn (Deadlock Prevention)

Ngăn chặn tắc nghẽn là thiết kế một hệ thống sao cho hiện tượng tắc nghẽn bị loại trừ. Các phương thức ngăn chặn tắc nghẽn đều tập trung giải quyết bốn điều kiện gây ra tắc nghẽn, sao cho hệ thống không thể xảy ra đồng thời bốn điều kiện tắc nghẽn:

* Điều kiện này gần như khơng tránh khỏi, vì sự độc quyền là cần thiết đối với tài nguyên thuộc loại phân chia được như các biến chung, các tập tin chia sẻ, hệ điều hành cần phải hỗ trợ sự độc quyền trên các tài nguyên này. Tuy nhiên, với những tài nguyên thuộc loại khơng phân chia được hệ điều hành có thể sử dụng kỹ thuật SPOOL Smulataneous Peripheral Operation Online để tạo ra nhiều tài nguyên ảo cung cấp cho các tiến trình đồng thời.
* Đối với điều kiện giữ và đợi: Điều kiện này có thể ngăn chặn bằng cách yêu cầu tiến trình yêu cầu tất cả tài nguyên mà nó cần tại một thời điểm và tiến trình sẽ bị khoá blocked cho đến khi yêu cầu tài nguyên của nó được hệ điều hành đáp ứng. Phương pháp này khơng hiệu quả. Thứ nhất, tiến trình phải đợi trong một khoảng thời gian dài để có đủ tài nguyên mới có thẻ chuyển sang hoạt động được, trong khi tiến trình chỉ cần một số ít tài nguyên trong số đó là có thể hoạt động được, sau đó yêu cầu tiếp. Thứ hai, lãng phí tài nguyên, vì có thể tiến trình giữa nhiều tài nguyên mà chỉ đến khi sắp kết thúc tiến trình mới sử dụng, và có thể đây là những tài nguyên mà các tiến trình khác đang rất cần. Ở đây hệ điều hành có thể tổ chức phân lớp tài nguyên hệ thống. Theo đó tiến trình phải trả tài nguyên ở mức thấp mới được cấp phát tài nguyên ở cấp cao hơn.
* Đối với điều kiện No preemption: Điều kiện này có thể ngăn chặn bằng cách, khi tiến trình bị rơi vào trạng thái khố, hệ điều hành có thể thu hồi tài nguyên của tiến trình bị khố để cấp phát cho tiến trình khác và cấp lại đầy đủ tài nguyên cho tiến trình khi tiến trình được đưa ra khỏi trạng thái khóa.
* Đối với điều kiện chờ đợi vòng tròn: Điều kiện này có thể ngăn chặn bằng cách phân lớp tài nguyên của hệ thống. Theo đó, nếu một tiến trình được cấp phát tài nguyên ở lớp L, thì sau đó nó chỉ có thể yêu cầu các tài nguyên ở lớp thấp hơn lớp L.

1. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG
   1. Phân tích vấn đề
      * Bài toán đặt ra: Viết chương trình giải quyết bài toán “Năm triết gia ăn tối”. Chương trình tạo ra năm quá trình con mô phỏng hoạt động của năm triết gia. Sử dụng Semaphore để đồng bộ hoạt động của năm triết gia này.
      * Mô tả bài toán: Bài toán “Năm triết gia ăn tối” được mô tả đầy đủ như sau: Có năm triết gia cùng ngồi ăn tối trên một bàn tròn, trước mặt mỗi triết gia có một đĩa mì Ý, giữa hai triết gia thì có một chiếc nĩa. Mỗi nhà triết học đều liên tục suy nghĩ và ăn khi đói, tuy nhiên, một nhà triết học chỉ có thể ăn mì Ý khi họ cầm được chiếc nĩa bên trái và bên phải của mình. Mỗi chiếc nĩa tại mỗi thời điểm chỉ có thể được lấy bởi một triết gia và vì vậy mội triết gia chỉ có thể sử dụng chiếc nĩa khi nó không được sử dụng bởi một triết gia khác. Sau khi ăn xong, triết gia cần phải đặt hai chiếc nĩa trở lại vị trí cũ và nó trở nên sẵn có cho các triết gia khác. Giả sử việc ăn không bị giới hạn bởi số lượng mì Ý và không gian dạ dày, mỗi triết gia đều không thể biết khi nào những người khác muốn ăn hoặc suy nghĩ. Vấn đề đặt ra là làm thế nào để thiết kế một thuật toán sao cho không có triết gia nào sẽ chết đói, tức là mỗi người có thể mãi mãi luôn phiên giữa suy nghĩ và ăn khi đói.



*Hình 1. Mô tả Bài toán Năm triết gia ăn tối*

Bài toán được đặt ra bởi nhà toán học Edsger W. Dijkstra năm 1965.

* 1. Giải quyết vấn đề

Ta xem mỗi triết gia là một tiến trình, những chiếc nĩa là tài nguyên dùng chung (tài nguyên găng) cần được bảo vệ. Chương trình cần xây dựng sẽ giải quyết hai vấn đề chính, đó là Quản lý vùng găng và Ngăn chặn Deadlock.

* + 1. Quản lý vùng găng

Để quản lý vùng găng, ta xây dựng class Chopstick đại diện cho chiếc nĩa, mỗi Chopstick sẽ có một thuộc tính taken kiểu boolean để đánh dấu trạng thái của chiếc nĩa (đang được sử dụng hay không):

* taken = true: nĩa tương ứng đang được sử dụng.
* taken = false: nĩa tương ứng chưa được sử dụng.

Việc đóng/mở vùng găng được class Chopstick thể hiện qua hai phương thức synchronized put() và get():



Phương thức put() tương ứng với hành động một triết gia đặt một chiếc nĩa xuống khi ăn xong. Trong phương thức này, thuộc tính taken được gán bằng false, tức là đổi trạng thái của nó thành “chưa được sử dụng”. Biến takenBy nhằm xác định triết gia nào đang sử dụng nĩa đó (takenBy mang giá trị từ 0 đến 4 ứng với năm triết gia, và -1 nếu nó đang không được sử dụng), biến này chủ yếu phục vụ cho việc biểu diễn trên giao diện.

Ngược lại, phương thức get() tương ứng với hành động một triết gia lấy một chiếc nĩa khi thấy đói. Trong phương thức này, ta cần kiểm tra trạng thái của chiếc nĩa trước khi cho phép triết gia lấy nó, tức là nếu thuộc tính taken đang là true (nĩa đang được sử dụng) thì ta phải đợi cho đến khi taken chuyển sang giá trị false (có một triết gia khác đã đặt nĩa đó xuống). Sau đó, ta lại gán biến taken = true để đánh dấu rằng nó đang được sử dụng.

Như vậy, class Chopstick đã giải quyết được vấn đề về tài nguyên dùng chung trong bài toán, sở dĩ làm được điều này là nhờ hai hàm của hệ thống: wait() và notify(). Hiểu một cách đơn giản, wait() cho phép tiến trình được loại ra khỏi danh sách tiến trình đang hoạt động cho đến khi một tiến trình khác gọi hàm notify(), notify() thông báo và kích hoạt trở lại tiến trình đầu tiên gọi wait() trên cùng một đối tượng. Tương ứng với việc một triết gia khi cần lấy một chiếc nĩa mà nĩa đó đã được sử dụng, sẽ phải đợi đến khi nĩa đó được đặt xuống, và nếu có hai triết gia cùng muốn dùng một chiếc nĩa, triết gia cần trước sẽ được ưu tiên.

* + 1. Giải pháp ngăn chặn deadlock

Chương trình sẽ giải quyết vấn đề deadlock bằng cách xây dựng hai phương thức Wait() và Signal() trong class Philosopher (đại diện cho triết gia) sao cho vòng tròn đợi không xảy ra.



Cách giải quyết ở đây là những triết gia với số thứ tự chẵn khi đói sẽ ưu tiến lấy nĩa bên phải của mình trước, sau đó đến nĩa bên trái, những triết gia có số thứ tự lẻ thì ngược lại. Như vậy hai triết gia cạnh nhau sẽ có thứ tự lấy nĩa khác nhau, tránh được vòng tròn đợi, bởi vì nếu mỗi triết gia đều lấy nĩa bên trái trước, hoặc bên phải trước thì sẽ có nguy cơ dẫn đến vòng tròn đợi. Đây còn gọi là giải pháp bất đối xứng.

* + 1. Chương trình

Chương trình được xây dựng gồm 4 class và 1 enum:

* Class Philosopher: đại diện cho triết gia (tiến trình).
* Class Chopstick: đại diện cho nĩa (tài nguyên găng).
* Class Dinner: chịu trách nhiệm khởi tạo các class khác.
* Class DinnerGUI: quản lí giao diện.
* Enum Status: định nghĩa các trạng thái của triết gia.
  + - 1. Class Philosopher

Class gồm các thuộc tính:



Vì class Philosopher đại diện cho tiến trình nên nó kế thừa class Thread trong Java, và vì là một Thread nên những tác vụ của nó được đưa vào phương thức run():



Như vậy phương thức run() mô tả trực quan những hành động luân phiên và lặp lại của triết gia, đó là: suy nghĩ – lấy nĩa khi đói – ăn – đặt nĩa xuống khi ăn xong – suy nghĩ (lặp lại).

Hai phương thức Wait() và Signal() đã được đề cập ở mục 2.2.2. Giải pháp xử lý deadlock.

* + - 1. Class Chopstick

Class Chopstick gồm các thuộc tính:



Các thuộc tính và phương thức của Class Chopstick đã được mô tả ở mục 2.2.1. Quản lý vùng găng.

* + - 1. Class Dinner

Class Dinner dùng để khởi tạo các tiến trình và tài nguyên dùng chung, các thuộc tính và hàm dựng của class Dinner như sau:



* + - 1. Class DinnerGUI

Class DinnerGUI kế thừa JFrame, dùng để quản lý giao diện, class này có hai phương thức dùng để cập nhật giao diện khi có sự thay đổi trạng thái của triết gia hoặc của nĩa:



* + - 1. Enum Status

Enum Status định nghĩa các trạng thái của triết gia, bao gồm:

* EATING: Đang ăn
* THINKING: Đang suy nghĩ
* HUNGRY: Đang đói (lúc cần ăn nhưng chưa lấy đủ 2 nĩa)
* GETRIGHT: Đang nắm nĩa bên phải của mình, chờ nĩa bên trái
* GETLEFT: Đang nắm nĩa bên phải của mình, chờ nĩa bên trái

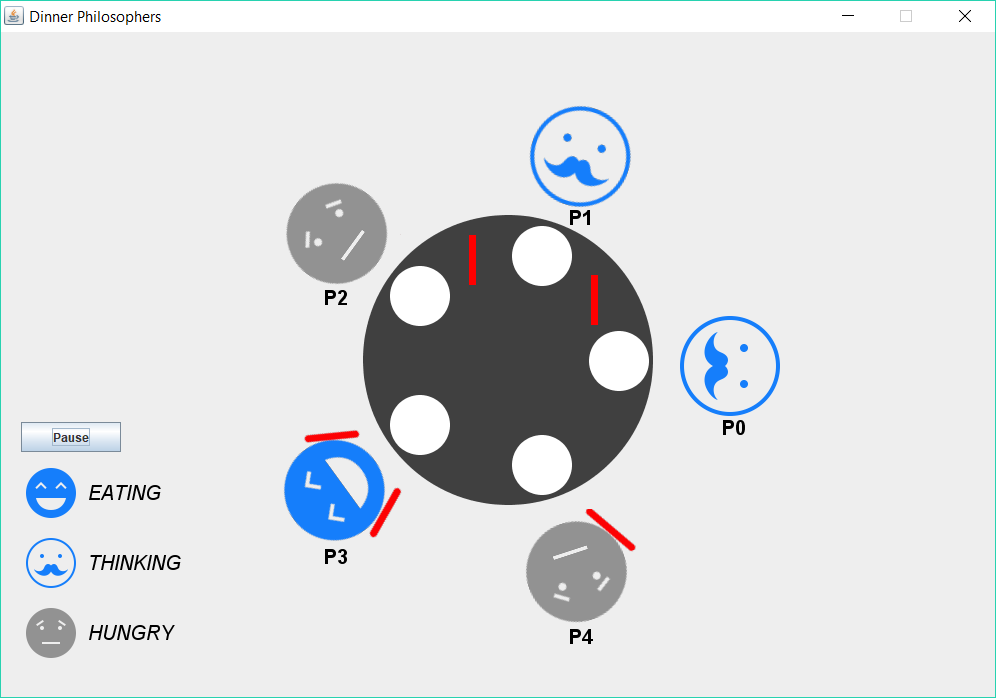
Thực chất hai trạng thái GETLEFT và GETRIGHT cũng là trạng thái lúc triết gia đói, nhưng vì mục đích hiển thị lên giao diện khác nhau, nên ta định nghĩa thêm hai trạng thái này.

1. TRIỂN KHAI   
   VÀ ĐÁNH GIÁ KẾT QUẢ
   1. Ngôn ngữ cài đặt

Chương trình được cài đặt bằng ngôn ngữ Java cơ bản và quen thuộc với sinh viên khoa Công nghệ thông tin, sử dụng công cụ Eclipse.

* 1. Kết quả chương trình

Xét một trạng thái chương trình như sau:



*Hình 2. Giao diện chương trình Bài toán Năm triết gia ăn tối*

Tại thời điểm đang xét, ta thấy:

* Triết gia số 0 và số 1 đang suy nghĩ.
* Triết gia số 3 đang ăn.
* Triết gia số 2 và số 4 đang đói.
* Triết gia số 4 đang đói và đã lấy nĩa bên phải của mình, đang chờ nĩa bên trái (đang được sử dụng bởi triết gia số 3). Còn triết gia số 2 mặc dù đang đói nhưng không lấy nĩa bên trái của mình vì phải đợi lấy chiếc nĩa bên phải trước (triết gia cố số thứ tự chẵn phải lấy nĩa bên phải của mình trước và ngược lại – giải pháp tránh deadlock).
  1. Đánh giá kết quả

Chương trình đã giải quyết được yêu cầu đặt ra của bài toán. Các kết quả thực hiện đều thỏa mãn những tiêu chí ban đầu.

1. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN
   1. Kết luận

Đồ án đã giải quyết được vấn đề đặt ra, bao gồm việc giải quyết Bài toán Năm triết gia ăn tối và tìm hiểu những khái niệm liên quan đến bài toán đó. Công việc điều độ tiến trình của hệ điều hành là vô cùng quan trọng bởi nó giúp các tiến trình trong quá trình hoạt động tránh được các xung đột với các tiến trình khác, từ đó tránh được nguy cơ sai sót hoặc mất mác tài nguyên dùng chung.

* 1. Hướng phát triển

Chương trình tuy đã giải quyết được đầy đủ các yêu cầu cơ bản của bài toán đặt ra, nhưng hẳn vẫn còn nhiều thiếu sót và bất tiện cho người dùng. Vì vậy, chương trình có thể phát triển thêm như sau:

* Giao diện có thể được hiển thị mượt hơn khi có sự thay đổi trạng thái của các triết gia cũng như nĩa.
* Thêm một slide bar để người dùng dễ dàng chỉnh sửa thời gian trung bình của việc ăn và suy nghĩ của triết gia (trong chương trình hiện tại giá trị này là ngẫu nhiên từ 5 đến 10 giây).
* Mô tả hình ảnh triết gia sinh động hơn.

PHẦN II – MẠNG MÁY TÍNH

TIÊU ĐỀ: SỬ DỤNG SOCKET TRONG JAVA XÂY DỰNG ỨNG DỤNG CHIA SẺ FILE THEO MÔ HÌNH PEER-TO-PEER

TỔNG QUAN VỀ ĐỀ TÀI

1. Bối cảnh và lý do thực hiện đề tài

Ngày nay, với sự phát triển không ngừng của khoa học và công nghệ, nhu cầu chia sẻ dữ liệu mà cụ thể là chia sẻ file giữa những người dùng máy tính với nhau càng trở nên phổ biến. Nếu như trong mô hình chia sẻ file Client/Server, vì dữ liệu chỉ được lưu trữ và quản lý ở một Server nên hiệu suất sẽ giảm xuống nếu có nhiều người dùng; thì trong mô hình mạng ngang hàng (Peer-to-Peer), càng nhiều người dùng thì càng làm cho mạng hiệu quả hơn, tuy nhiên nó cũng có những khuyết điểm riêng.

Để tìm hiểu thấu đáo hơn về mô hình truyền file Peer-to-Peer, cũng như nâng cao hiểu biết và khả năng lập trình của mình về Socket trong Java, em đã chọn đề tài “Sử dụng Socket trong Java xây dựng ứng dụng chia sẻ file theo mô hình Peer-to-Peer”.

1. Kết cấu của đồ án

* Tìm hiểu cơ bản về mạng máy tính.
* Tìm hiểu về Socket, Socket trong Java.
* Tìm hiểu về mô hình truyền file Peer-to-Peer.
* Sử dụng Socket trong Java, cũng như kĩ thuật lập trình đa luồng,… để xây dựng ứng dụng truyền file qua mạng theo mô hình Peer-to-Peer.

1. CƠ SỞ LÝ THUYẾT
   1. Căn bản về mạng máy tính
      1. Định nghĩa về mạng máy tính

Mạng máy tính là một hệ thống gồm nhiều máy tính và các thiết bị được kết nối với nhau bởi đường truyền vật lý theo một kiến trúc (Network Architecture) nào đó nhằm thu thập, trao đổi dữ liệu và chia sẽ tài nguyên cho nhiều người sử dụng.

Các máy tính được kết nối với nhau có thể trong cùng một phòng, một tòa nhà, một thành phố hoặc trên phạm vi toàn cầu.

Mạng máy tính bao gồm ba thành phần chính:

* Các máy tính;
* Các thiết bị mạng đảm bảo kết nối các máy tính với nhau;
* Phần mềm cho phép thực hiện việc trao đổi thông tin giữa các máy tính.
  + 1. Các giao thức mạng

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

TCP và UDP là hai giao thức cốt lõi của bộ giao thức TCP/IP.

* + - 1. Giao thức TCP

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

Giao thức TCP đạt được điều này theo hai cách. Đầu tiên, nó yêu cầu các gói tin bằng cách đánh số chúng. Thứ hai, nó kiểm tra lỗi bằng cách yêu cầu bên nhận gửi phản hồi đã nhận được cho bên gửi. Nếu bên gửi không nhận được phản hồi đúng, nó có thể gửi lại gói tin để đảm bảo bên nhận nhận chúng một cách chính xác.

Như vậy, có thể thấy TCP cung cấp cho chúng ta một kênh truyền thông điểm – điểm phục vụ cho các ứng dụng đòi hỏi giao tiếp tin cậy như HTTP (HyperText Transfer Protocol), FTP (File Transfer Protocol), Telnet,… Các ứng dụng này đòi hỏi một kênh giao tiếp tin cậy bởi thứ tự của dữ liệu được gửi và nhận là yếu tố quyết định đến sự thành công hay thất bại của chúng.

* + - 1. Giao thức UDP

Giao thức UDP hoạt động tương tự như TCP, nhưng nó bỏ qua quá trình kiểm tra lỗi. Khi một ứng dụng sử dụng giao thức UDP, các gói tin được gửi cho bên nhận và bên gửi không phải chờ để đảm bảo bên nhận đã nhận được gói tin, do đó nó lại tiếp tục gửi gói tin tiếp theo. Nếu bên nhận bỏ lỡ một vài gói tin UDP, họ sẽ mất vì bên gửi không gửi lại chúng. Do đó thiết bị có thể giao tiếp nhanh hơn.

UDP được sử dụng khi tốc độ nhanh và không cần thiết sửa lỗi. Ví dụ, UDP thường được sử dụng cho các chương trình phát sóng trực tiếp và game online.

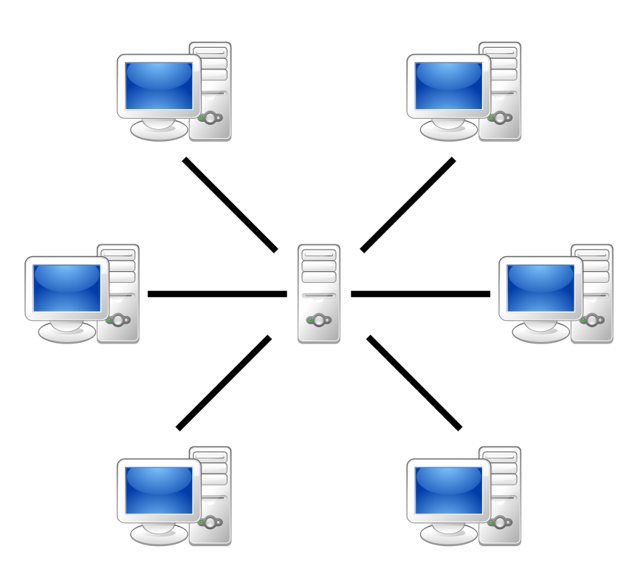
Dưới đây là bảng so sánh sự khác biệt giữa hai giao thức TCP và UDP:

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

Bảng 1. So sánh hai giao thức TCP và UDP

* + 1. Các mô hình hoạt động của mạng máy tính
       1. Mô hình Trạm – Chủ (Client – Server)

Mô hình client-server là một mô hình nổi tiếng trong mạng máy tính, được áp dụng rất rộng rãi. Ý tưởng của mô hình này là máy con (đóng vai trò là máy khách) gửi một yêu cầu (request) để máy chủ (đóng vai trò người cung ứng dịch vụ), máy chủ sẽ xử lý và trả kết quả về cho máy khách.

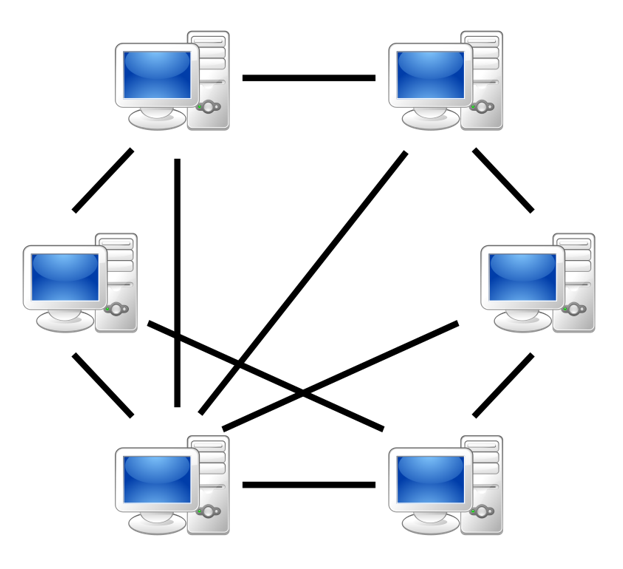


Hình 3. Mô hình Client – Server

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

* + - 1. Mô hình mạng ngang hàng (Peer-to-Peer)

Mạng ngang hàng còn gọi là mạng đồng đẳng, là một mạng máy tính trong đó hoạt động của mạng chủ yếu dựa vào khả năng tính toán và băng thông của các máy tham gia chứ không tập trung vào một số nhỏ các máy chủ trung tâm như các mạng thông thường.



*Hình 4. Mô hình mạng ngang hàng*

Một mạng đồng đẳng đúng nghĩa không có khái niệm máy chủ và máy khách, nói cách khác, tất cả các máy tham gia đều bình đẳng và được gọi là peer, là một nút mạng đóng vai trò đồng thời là máy khách và máy chủ đối với các máy khác trong mạng.

Dưới đây là bảng so sánh hai mô hình mạng Client – Server và Peer-to-Peer:

|  |  |  |
| --- | --- | --- |
| Mô hình mạng | Client – Server | Peer-to-Peer |
| Chỉ tiêu đánh giá |
| Độ an toàn và tính bảo mật thông tin | Có độ an toàn và bảo mật thông tin cao nhất. Quản trị mạng có thể điều chỉnh quyền truy nhập thông tin. | Độ an toàn và bảo mật kém, phụ thuộc vào mức truy cập được chia sẻ. |
| Khả năng cài đặt | Khó cài đặt. | Dễ cài đặt. |
| Đòi hỏi về phần cứng và phần mềm | Đòi hỏi có máy chủ, hệ điều hành mạng và các phần cứng bổ sung. | Không cần máy chủ, hệ điều hành mạng, phần cứng bổ sung rất ít. |
| Quản trị mạng | Phải có quản trị mạng. | Không cần có quản trị mạng. |
| Xử lý và lưu trữ tập trung | Có. | Không. |
| Chi phí cài đặt | Cao. | Thấp. |

*Bảng 2. So sánh hai mô hình mạng Client – Server và Peer-to-Peer*

* 1. Socket và Socket trong Java
     1. Khái niệm Socket

Socket là một giao diện lập trình ứng dụng (API), giúp lập trình kết nối các ứng dụng để truyền và nhận giữ liệu trong môi trường có kết nối internet bằng cách sử dụng phương thức TCP và UDP. Trong quá trình làm việc ta có thể chạy nhiều socket cùng một lúc nên công việc sẽ nhanh hơn, nâng cao hiệu suất làm việc.

Có thể hiểu một cách khác socket là một cổng logic mà một chương trình sử dụng để kết nối với một chương trình khác chạy trên một máy tính khác trên Internet. Chương trình mạng có thể sử dụng nhiều socket cùng một lúc, nhờ đó nhiều chương trình có thể sử dụng Internet cùng một lúc.

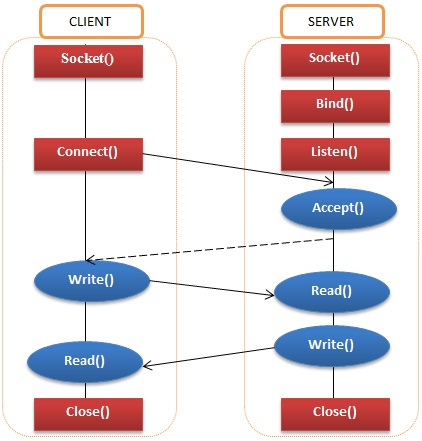
* + 1. Các loại Socket

Có 2 loại socket dựa theo hai giao thức mạng đã trình bày ở mục 1.1.2.

* + - 1. Socket có hướng kết nối (Stream Socket)

Stream Socket dựa trên giao thức TCP( Tranmission Control Protocol), việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình đã thiết lập kết nối. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.

Cơ chế gọi hàm trong lập trình Socket có hướng kết nối:



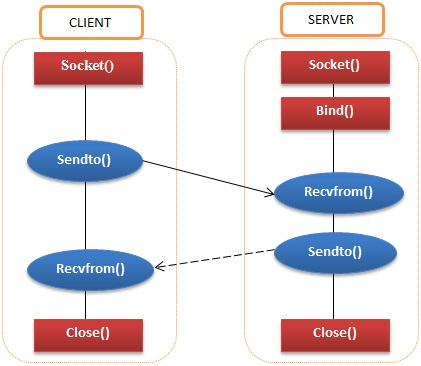
Hình 5. Cơ chế gọi hàm trong lập trình Socket có hướng kết nối

Đặc điểm của Socket hướng kết nối:

* Có một đường kết nối (địa chỉ IP) giữa 2 tiến trình
* Một trong 2 tiến trình kia phải đợi tiến trình kia yêu cầu kết nối
* Mỗi thông điệp gửi phải có xác nhận trả về
* Các gói tin chuyển đi tuần tự.
  + - 1. Socket không hướng kết nối

Datagram Socket dựa trên giao thức UDP( User Datagram Protocol), việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 quá trình. Ngược lại với giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy, có thế không đúng trình tự và lặp lại.

Cơ chế gọi hàm trong lập trình Socket không hướng kết nối:



*Hình 6. Cơ chế gọi hàm trong lập trình Socket không hướng kết nối*

Đặc điểm của Socket không hướng kết nối:

* Hai tiến trình liên lạc với nhau không kết nối trực tiếp
* Thông điệp gửi đi phải kèm theo thông điệp người nhận
* Thông điệp có thể gửi nhiều lần
* Người gửi không chắc chắn thông điệp đến tay người nhận
* Thông điệp gửi sau có thể đến trước và ngược lại.
  + 1. Socket trong Java
       1. Một số lớp trong lập trình Java Socket

Java hỗ trợ lập trình mạng thông qua các lớp trong gói java.net. Một số lớp tiêu biểu sử dụng Socket làm phương tiện giao tiến như:

* InetAddress: Lớp này quản lý địa chỉ Internet bao gồm địa chỉ IP và tên máy tính
* Socket: Hỗ trợ phương thức liên quan đến socket cho chương trình client ở chế độ có kết nối
* ServerSocket: Hỗ trợ các phương thức liên quan đến socket cho chương trình server ở chế độ có kết nối
* DatagramSocket: Hỗ trợ các phương thức liên quan đến socket ở chế độ không kết nối cho cả client và server
* DatagramPacket: Lớp cài đặt gói tin dùng thư tín người dùng trong giao tiếp giữa client và server ở chế độ không kết nối.

Vì độ tin cậy của giao thức TCP, phù hợp với một ứng dụng truyền file nên em sẽ chọn Socket có hướng kết nối để xây dựng ứng dụng; ta sẽ đi sâu tìm hiểu về hai lớp ServerSocket và Socket.

* + - 1. Lớp ServerSocket

Lớp java.net.ServerSocket được sử dụng bởi các ứng dụng máy chủ để tạo ra một một cổng và lắng nghe các yêu cầu của máy khách.

Các hàm dựng của lớp ServerSocket:

|  |  |
| --- | --- |
| STT | Hàm dựng và mô tả |
| 1 | public ServerSocket (port int) throws IOException  Cố gắng tạo một ServerSocket bị ràng buộc vào port được chỉ định. Một ngoại lệ xảy ra nếu port đã bị ràng buộc bởi một ứng dụng khác. |
| 2 | public ServerSocket(int port, int backlog) throws IOException  Tương tự như hàm dựng ở trên, tham số backlog xác định có bao nhiêu máy khách đến để lưu trữ trong một hàng đợi. |
| 3 | public ServerSocket(int port, int backlog, InetAddress address) throws IOException  Tương tự như hàm dựng ở trên, tham số InetAddress chỉ định địa chỉ IP cục bộ để ràng buộc. InetAddress được sử dụng cho các máy chủ có thể có nhiều địa chỉ IP, cho phép máy chủ xác định địa chỉ IP nào để chấp nhận yêu cầu của máy khách. |
| 4 | public ServerSocket() throws IOException  Tạo ra một ServerSocket không kết nối. Khi sử dụng Constructor này, sử dụng phương thức bind() khi ta đã sẵn sàng để ràng buộc socket của máy chủ. |

Bảng 3. Các hàm dựng của lớp ServerSocket

Nếu Constructor ServerSocket không ném một ngoại lệ, có nghĩa là ta đã khởi tạo thành công ServerSocket với cổng được chỉ định và đã sẵn sàng cho các yêu cầu của máy khách.

Một số phương thức của lớp ServerSocket:

|  |  |
| --- | --- |
| STT | Phương thức và mô tả |
| 1 | public int getLocalPort()  Trả về cổng mà Server Socket đang lắng nghe. Phương thức này rất có ích, nếu ta đưa vào giá trị cổng bằng 0 trong hàm dựng, nghĩa là nói với server tự tìm một cổng phù hợp, ta có thể sử dụng phương thức này để lấy ra giá trị cổng đó. |
| 2 | public Socket accept() throws IOException  Đợi một yêu cầu kết nối từ client. Phương thức này sẽ khóa ứng dụng cho tới khi có một yêu cầu kết nối từ client đến trên cổng cụ thể hoặc hết thời gian chờ nếu thời gian chờ được thiết lập bằng phương thức setSoTimeout(). Ngược lại phương thức này sẽ khóa chương trình vô thời hạn. |
| 3 | public void setSoTimeout(int timeout)  Thiết lập thời gian chờ tối đa nghĩa là thời gian Server Socket sẽ chờ yêu cầu kết nối từ người dùng trong quá trình accept(). |
| 4 | public void bind(SocketAddress host, int backlog)  Nối Server Socket tới một server cụ thể có cổng định sẵn trong đối tượng SocketAddress. Sử dụng phương thức này nếu ta khởi tạo ServerSocket sử dụng hàm dựng mặc định (không tham số). |

Bảng 4. Một số phương thức của lớp ServerSocket

Khi ServerSocket gọi accept(), phương thức này sẽ không return cho đến khi một client kết nối đến. Sau khi máy khách (client) kết nối, ServerSocket tạo một Socket mới trên một cổng không xác định và trả về một tham chiếu đến Socket mới này. Hiện kết nối TCP giữa máy khách và máy chủ và có thể tương tác với nhau.

* + - 1. Lớp Socket

Lớp java.net.Socket đại diện cho socket mà cả máy khách và máy chủ sử dụng để liên lạc với nhau. Máy khách nhận được một đối tượng Socket bằng cách khởi tạo một, trong khi máy chủ lấy một đối tượng Socket từ giá trị trả về của phương thức accept().

Các hàm dựng của lớp Socket:

|  |  |
| --- | --- |
| STT | Hàm dựng và mô tả |
| 1 | public Socket(String host, int port) throws UnknownHostException, IOException  Hàm dựng này tạo Socket kết nối tới Server cụ thể tại cổng xác định trong tham số. Nếu nó không ném ra ngoại lệ, nghĩa là việc kết nối thành công, client đã kết nối với server. |
| 2 | public Socket(InetAddress host, int port) throws IOException  Hàm dựng này giống hệt với hàm dựng ở trên, ngoại trừ Server được biểu thị bằng một đối tượng InetAddress. |
| 3 | public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException  Kết nối tới host, port cụ thể, tạo ra một Socket trên host địa phương, và cổng địa phương được chỉ định trong tham số. |
| 4 | public Socket(InetAddress host, int port, InetAddress localAddress, int localPort) throws IOException  Hàm dựng này giống hệt với hàm dựng trên, ngoại trừ server được biểu thị bằng một đối tượng InetAddress thay vì một String. |
| 5 | public Socket()  Tạo một Socket chưa kết nối. Sử dụng phương thức connect() để kết nối socket này tới server. |

Bảng 5. Các hàm dựng của lớp Socket

Một số phương thức của lớp Socket:

|  |  |
| --- | --- |
| STT | Phương thức và mô tả |
| 1 | public void connect(SocketAddress host, int timeout) throws IOException  Phương thức này kết nối socket với máy chủ được chỉ định, cần thiết chỉ khi ta khởi tạo Socket bằng cách sử dụng Constructor không có đối số. |
| 2 | public InetAddress getInetAddress()  Phương thức này trả về địa chỉ của máy tính khác mà socket này được kết nối. |
| 3 | public int getPort()  Trả về cổng mà socket bị ràng buộc trên từ xa. |
| 4 | public int getLocalPort()  Trả về cổng mà socket bị ràng buộc trên máy local. |
| 5 | public SocketAddress getRemoteSocketAddress()  Trả về địa chỉ của socket từ xa. |
| 6 | public InputStream getInputStream() throws IOException  Trả về input stream của socket. Input stream được kết nối với output stream của socket từ xa. |
| 7 | public OutputStream getOutputStream() throws IOException  Trả về output stream của socket. Output stream được kết nối với input stream của socket từ xa. |
| 8 | public void close() throws IOException  Đóng socket, làm cho đối tượng Socket này không còn có khả năng kết nối về với bất kỳ máy chủ nào. |

Bảng 6. Một số phương thức của lớp Socket

* + - 1. Lớp InetAddress

Một số phương thức thường dùng trong lập trình mạng với Java, thuộc lớp InetAddress:

|  |  |
| --- | --- |
| STT | Phương thức và mô tả |
| 1 | static InetAddress getByAddress(byte[] addr)  Trả về đối tượng InetAddress cho bởi địa chỉ IP thô. |
| 2 | static InetAddress getByAddress(String host, byte[] addr)  Trả về đối tượng InetAddress dựa trên tên máy chủ được cung cấp và địa chỉ IP. |
| 3 | static InetAddress getByName(String host)  Xác định địa chỉ IP của host, cho bởi tên host. |
| 4 | String getHostAddress()  Trả về địa chỉ IP. |
| 5 | String getHostName()  Trả về host name. |
| 6 | static InetAddress getLocalHost()  Trả về địa chỉ địa phương. |

Bảng 7. Một số phương thức của lớp InetAddress

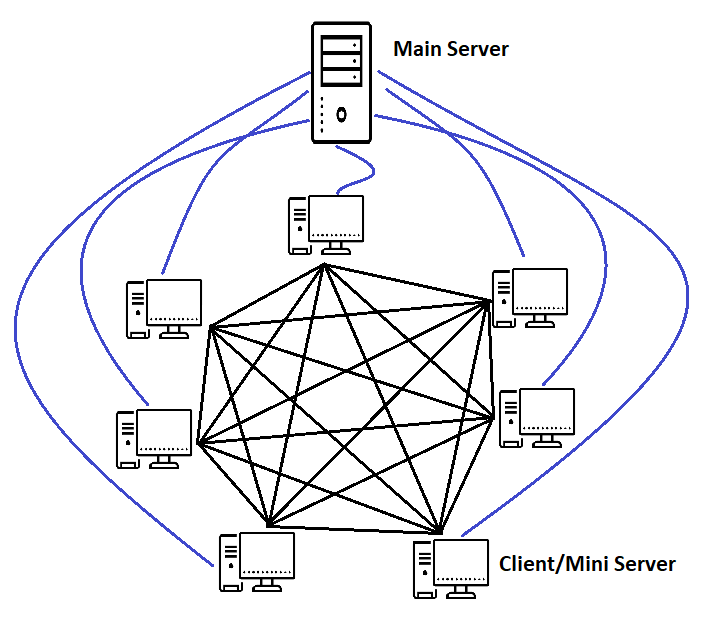
1. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG
   1. Phân tích yêu cầu

Bài toán đặt ra: Sử dụng Socket trong Java xây dựng ứng dụng chia sẻ file theo mô hình Peer-to-Peer.

Như vậy, ta sẽ tiến hành xây dựng một chương trình cho phép người dùng có thể chia sẻ file qua lại trực tiếp với nhau, mỗi người dùng cũng đóng vai trò như một kho chứa file để chia sẻ cho người khác. Giao tiếp qua lại giữa các người dùng thông qua Socket.

* 1. Mô tả chương trình

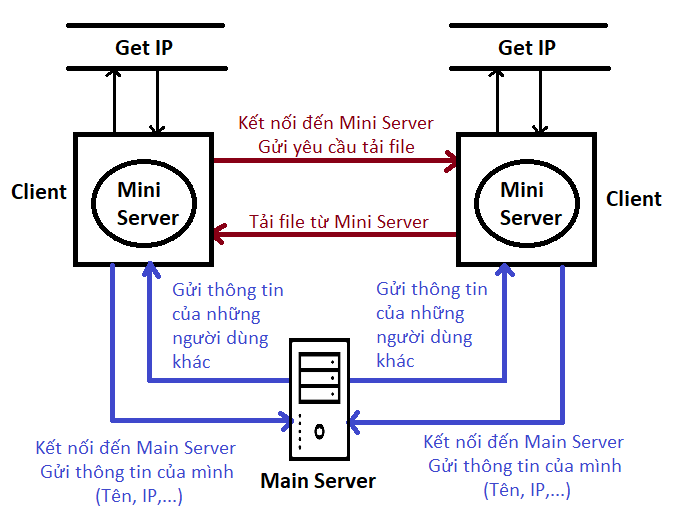
Trong chương trình, ta sử dụng mô hình mạng ngang hàng tập trung thế hệ thứ nhất, tức là vẫn còn dựa vào một máy chủ trung tâm (sau đây em gọi tắt là Main Server). Các máy khách tham gia mạng được kết nối với Main Server, truy vấn thông tin về những máy khách khác hoặc gửi thông tin của mình để Main Server quản lý. Các máy khách cũng đóng vai trò là một Server đối với những máy khách còn lại (sau đây em gọi tắt mà Mini Server). Như vậy, quá trình truyền tải file vẫn được thực hiện trực tiếp qua lại giữa các máy khách, không phụ thuộc Main Server.



*Hình 7. Mô hình mạng đồng đẳng lai*

Mô hình này còn được gọi là mạng đồng đẳng lai, mạng ngang hàng không thuần túy,…

Hoạt động của hệ thống được khái quát trong sơ đồ sau:



*Hình 8. Sơ đồ tổng thể tương tác của hệ thống*

* 1. Phân tích thiết kế
     1. Chương trình Main Server

Xây dựng Main Server đáp ứng các yêu cầu sau:

* Nhiều người dùng kết nối cùng một lúc
* Có thể nhận thông tin của từng người dùng (Tên, địa chỉ IP, danh sách File chia sẻ), quản lý thông tin đó trong một danh sách các người dùng đang kết nối, đồng thời có thể gửi những thông tin đó cho từng người dùng.

Do hoạt động theo giao thức TCP, Main Server khi khởi chạy sẽ tạo một Server Socket để lắng nghe các kết nối từ người dùng:



Server Socket này lắng nghe trên một địa chỉ IP và port cụ thể, tham số backlog (10) có nghĩa là số yêu cầu kết nối tối đa được phép chờ trong hàng đợi để Server xử lý (không phải số kết nối tối đa đồng thời của Server). Server Socket không kết nối đến một client cụ thể nào, kết nối cụ thể này chỉ được trả về từ hàm accept() khi có một client nào đó kết nối đến. Mỗi lần chấp nhận một kết nối, Main Server tạo một luồng mới để tương tác với client vừa chấp nhận:



Như vậy, giao tiếp giữa Main Server với từng client sẽ được thực hiện ở từng luồng riêng biệt, các loại thông điệp qua lại giữa Main Server và client phụ thuộc vào yêu cầu mà client gửi lên:



* *GET:* Ứng với client muốn lấy thông tin của những người dùng khác đang kết nối, Server sẽ gửi những thông tin này về cho client.
* *POST:* Ứng với client muốn gửi thông tin của mình lên để Server quản lý, Server sẽ lưu thông tin này vào một mảng các người dùng đang kết nối.
* *CONNECT:* Ứng với client yêu cầu được tham gia vào mạng chung, nếu kiểm tra người dùng chưa kết nối trước đó, Server sẽ cho phép client tham gia, ngược lại, Server gửi thông điệp từ chối kết nối và đóng socket ứng với client này.
* *DISCONNECT:* Ứng với client muốn ngắt kết nối, Server sẽ đóng socket ứng với client này và xóa client này ra khỏi danh sách đang kết nối.

Main Server quản lý những người dùng đang kết nối bằng một ArrayList:



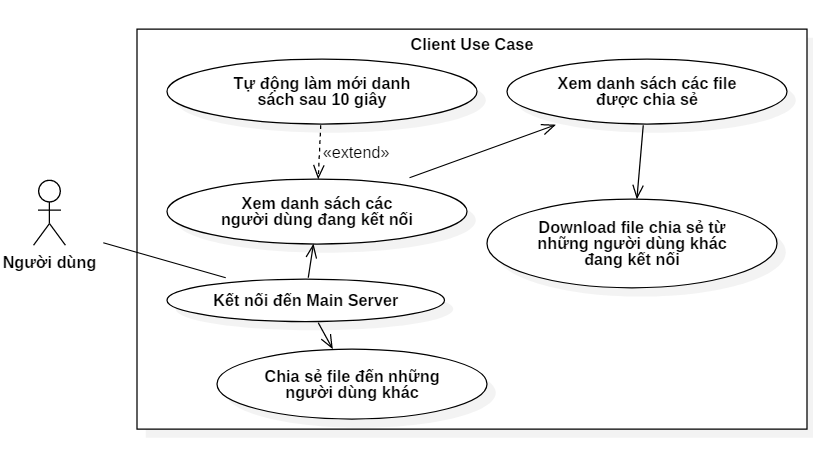
Với Peer là một class đại diện cho một người dùng, với những thuộc tính Tên, Địa chỉ IP, và danh sách các File chia sẻ:



MyFile là class kế thừa từ *java.io.File*, ta kế thừa để sửa đổi một số phương thức nhằm phục vụ cho việc hiển thị file trên giao diện được tường minh hơn.

* + 1. Chương trình Client/Mini Server

Các chức năng chính của chương trình phía người dùng được mô tả trong sơ đồ dưới đây :



*Hình 9. Sơ đồ use case đối với người dùng ứng dụng*

* + - 1. Chức năng truy vấn thông tin với Main Server

Để kết nối với Main Server, đầu tiên chương trình yêu cầu người dùng nhập vào địa chỉ IP của Main Server, sau đó dùng địa chỉ đó tạo một đối tượng socket kết nối đến Main Server:



Sau khi đã kết nối, client sẽ gửi đến Main Server thông điệp CONNECT, kèm theo tên và địa chỉ IP của mình, đợi Main Server kiểm tra, nếu trong danh sách đang kết nối ở Server chưa có thông tin trùng với tên và IP này, Main Server sẽ cho phép client tham gia vào mạng ngang hàng, lúc này client mới chính thức tham gia vào mạng, một giao diện khác hiện lên để người dùng bắt đầu sử dụng các chức năng khác của ứng dụng.



Sau khi đăng nhập, client gửi danh sách file chia sẻ của mình lên Main Server bằng cách: gửi thông điệp POST, sau đó gửi danh sách các file mà mình chia sẻ lên Main Server (chỉ gửi danh sách file, không gửi dữ liệu của file):



Client nhận thông tin về những người dùng đang kết nối khác bằng cách gửi thông điệp GET, sau đó nhận lại dữ liệu từ Main Server, lưu lại trong một danh sách, đồng thời hiển thị lên giao diện:



* + - 1. Chức năng tải file và truyền file cho người dùng khác

Để có thể chia sẻ file với người dùng khác, client đóng vai trò là một Mini Server, Mini Server này chạy trên một luồng song song với luồng hiện tại, nhằm đảm bảo một người dùng có thể vừa tải file, vừa truyền file cho người khác cùng một lúc:



Mini Server cũng được tạo bằng cách sử dụng Server Socket, với địa chỉ IP chính là địa chỉ IP của client đó, port được quy định cụ thể:



Mỗi khi chấp nhận kết nối từ một người dùng khác, Mini Server tạo một luồng mới để trao đổi dữ liệu với người dùng đó, điều này cho phép một người dùng có thể cùng lúc cho phép nhiều người dùng khác tải file của họ:



Trong luồng Mini Server Thread này, Mini Server nhận thông tin về file muốn tải, đọc file đó từ máy tính và truyền cho người dùng khác thông qua socket tương ứng.

Mặt khác, để tải file từ một người dùng khác, client cũng tạo một luồng mới chạy song song với luồng hiện tại, điều này cho phép người dùng có thể cùng một lúc tải được nhiều file, từ nhiều người khác nhau…

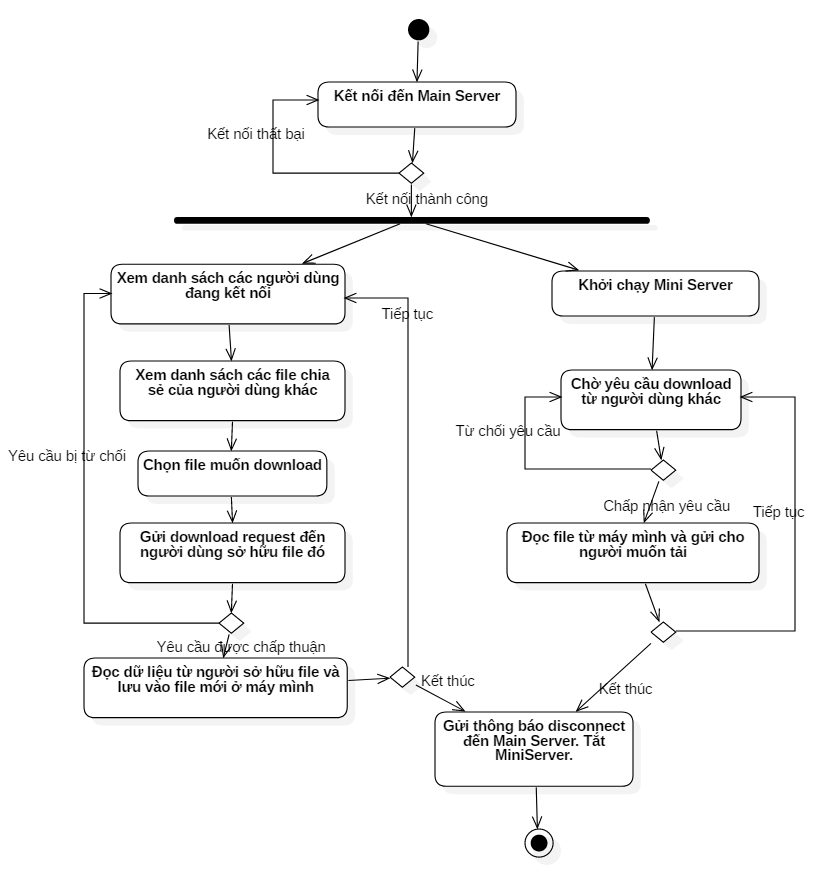


Luồng Download Thread tạo một socket kết nối đến Mini Server của người sở hữu file, sau đó gửi thông tin về file muốn tải, thông tin về người tải,… đến Mini Server đó, chờ đợi được chấp thuận và tiến hành đọc file từ Mini Server gửi đến, lưu file vào máy mình:



* + - 1. Tổng quan hoạt động của chương trình Client/Mini Server

Tổng quan hoạt động của chương trình Client/Mini Server được biểu diễn trong sơ đồ dưới đây:



Hình 10. Sơ đồ hoạt động của ứng dụng

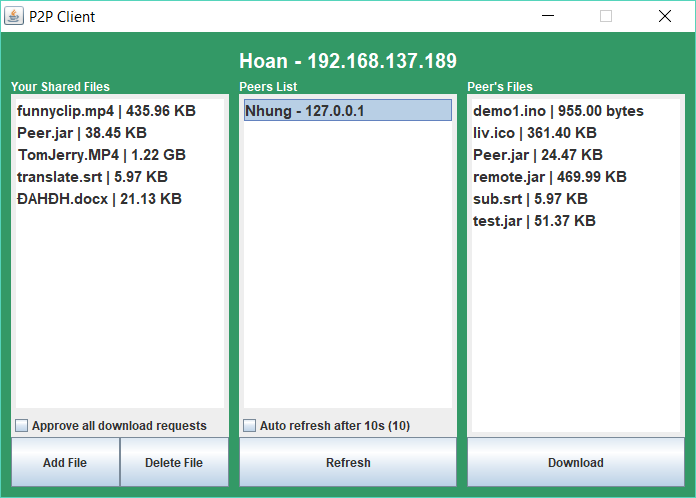
Ngoài ra chương trình còn có một số chức năng bổ sung như:

* Tự động làm mới danh sách người dùng đang kết nối sau mỗi 10 giây;
* Tùy chọn chấp nhận tất cả các yêu cầu download của người dùng khác;
* Trực tiếp thêm file vào folder chia sẻ;
* Trực tiếp xóa file khỏi folder chia sẻ.

1. TRIỂN KHAI   
   VÀ ĐÁNH GIÁ KẾT QUẢ
   1. Ngôn ngữ cài đặt

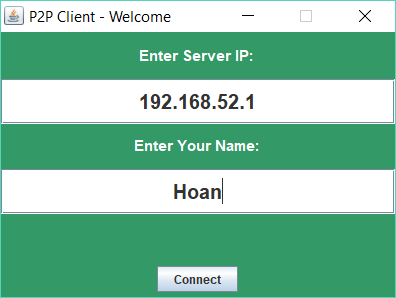
Chương trình được cài đặt bằng ngôn ngữ Java cơ bản và quen thuộc với sinh viên khoa Công nghệ thông tin, sử dụng công cụ Eclipse.

* 1. Kết quả chương trình
     1. Giải thích một số control trên giao diện

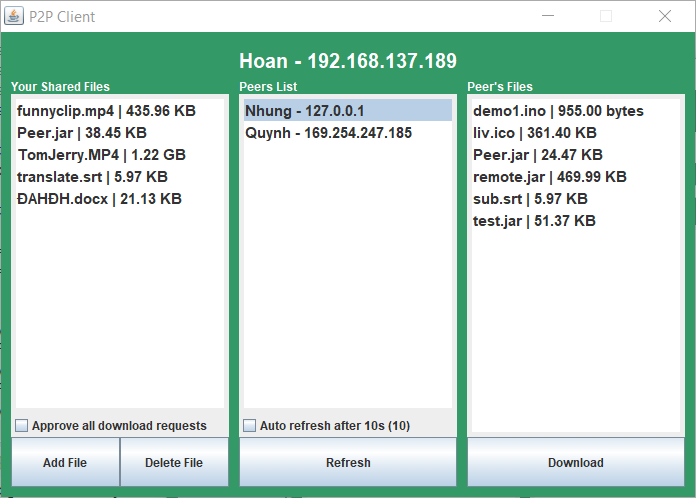


*Hình 11. Giao diện cơ bản của chương trình*

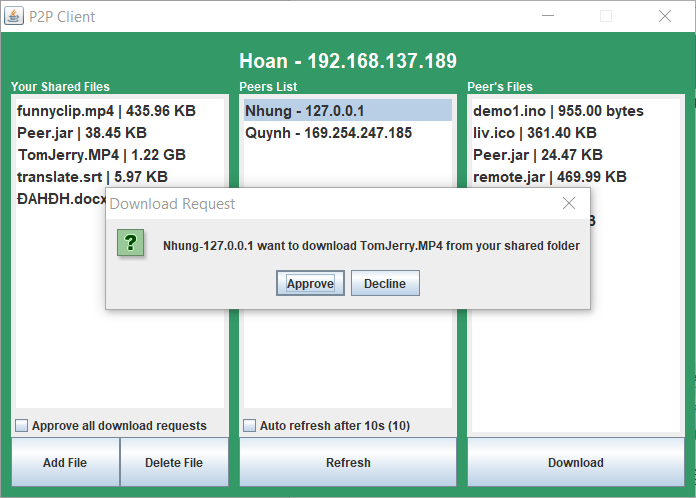
* Your Shared Files: Danh sách các file chia sẻ của người dùng;
* Peers List: Danh sách những người dùng khác đang kết nối;
* Peer’s Files: Danh sách các file chia sẻ của người dùng tương ứng được chọn ở Peer List;
* Add File: Nút cho phép thêm file vào folder chia sẻ;
* Delete File: Nút cho phép xóa file khỏi folder chia sẻ;
* Refresh: Nút cho phép làm mới danh sách người dùng đang kết nối;
* Download: Nút cho phép download file từ người dùng khác;
* Approve all…: Checkbox cho phép người dùng chấp nhận tất cả các yêu cầu tải file từ người khác;
* Auto refresh…: Checkbox cho ứng dụng tự động làm mới danh sách người dùng đang kết nối sau mỗi 10 giây.
  + 1. Hình ảnh kết quả chương trình



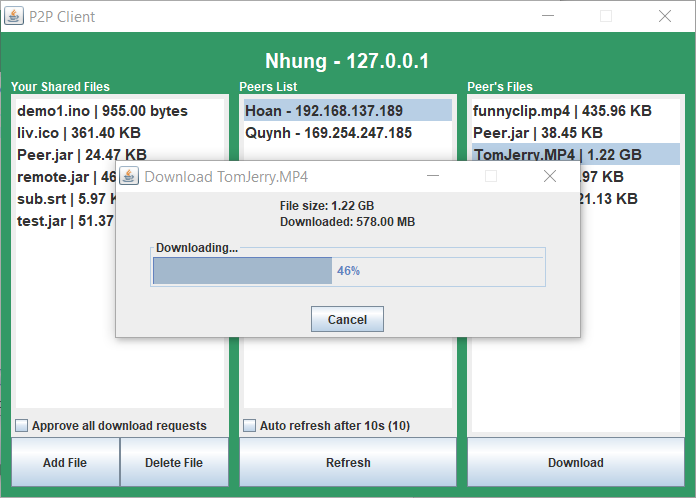
*Hình 12. Hình ảnh demo – Giao diện kết nối với Main Server*



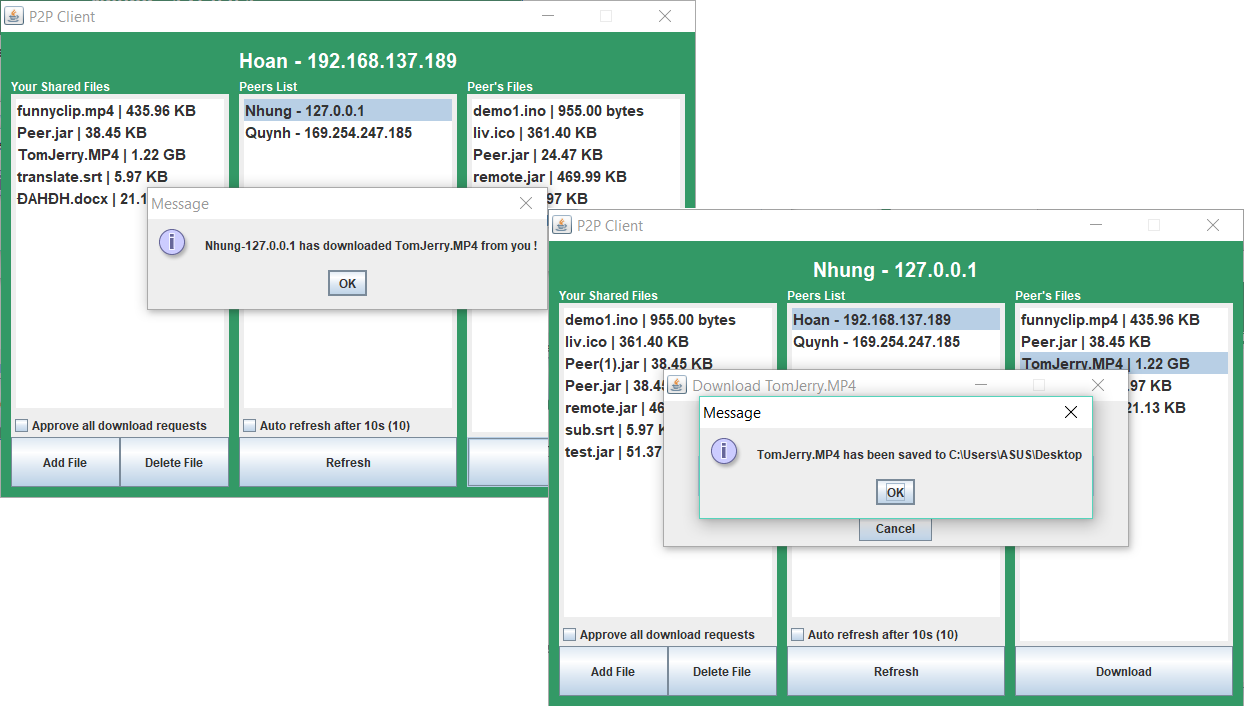
*Hình 13. Hình ảnh demo – Giao diện chính của chương trình*



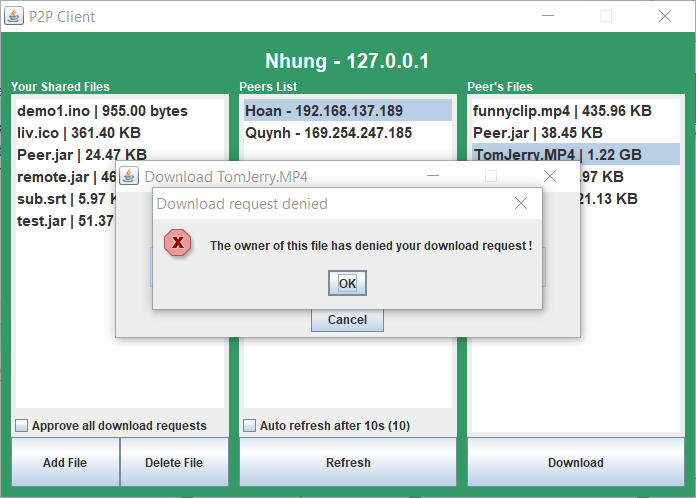
*Hình 14. Hình ảnh demo – Giao diện khi nhận yêu cầu tải file từ người khác*



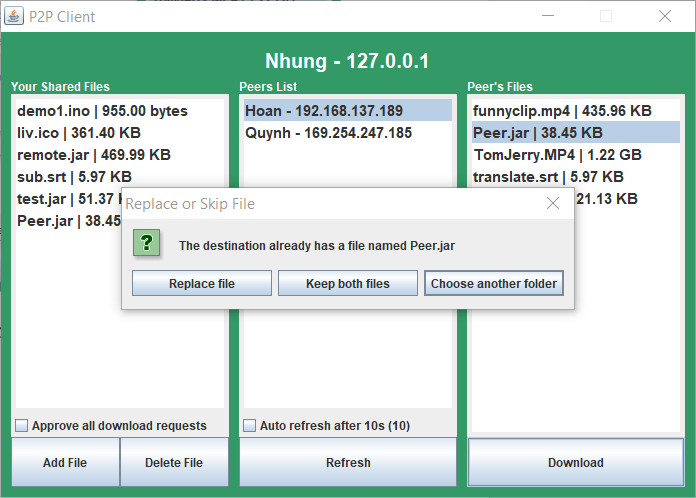
*Hình 15. Hình ảnh demo – Giao diện khi download file*



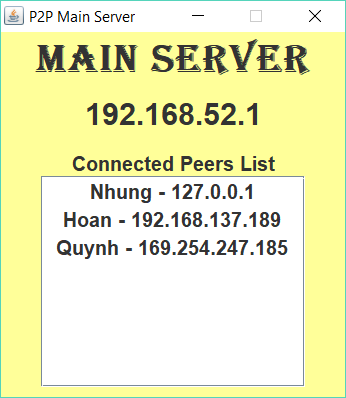
*Hình 16. Hình ảnh demo – Giao diện khi download thành công ở hai bên người dùng*



*Hình 17. Hình ảnh demo – Trường hợp người sở hữu file từ chối yêu cầu*



*Hình 18. Hình ảnh demo – Trường hợp file download trùng tên với một file trong thư mục muốn lưu*



*Hình 19. Hình ảnh demo – Giao diện Main Server tương ứng*

* 1. Đánh giá kết quả

Chương trình đã giải quyết được các yêu cầu đặt ra. Người dùng có thể chia sẻ file qua lại trực tiếp với nhau. Main Server đóng vai trò quản lý và cung cấp thông tin cho người dùng.

1. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN
   1. Kết luận

Đồ án đã giải quyết được vấn đề đặt ra, bao gồm việc tìm hiểu các khái niệm cơ bản của mạng máy tính, tìm hiểu về Socket, mô hình truyền file Peer-to-Peer và đã xây dựng được ứng dụng.

* 1. Hướng phát triển

Chương trình tuy đã giải quyết được đầy đủ các yêu cầu cơ bản của bài toán đặt ra, nhưng hẳn vẫn còn nhiều thiếu sót và bất tiện cho người dùng. Vì vậy, chương trình có thể phát triển thêm như sau:

* Chương trình không cố định folder chia sẻ mà có thể cho phép người dùng linh động lựa chọn;
* Chương trình có thể có chức năng tìm kiếm, cho phép người dùng tìm file muốn tải, Main Server sẽ nhận tên file và tìm trong từng danh sách file chia sẻ của từng người dùng đang kết nối;
* Chương trình có thể cho phép người dùng chat qua lại với nhau, phục vụ cho việc trao đổi giữa người dùng được thuận tiện hơn;
* Giao diện có thể bắt mắt hơn.

# KẾT LUẬN CHUNG

Qua quá trình nghiên cứu và tìm hiểu đề tài, em đã có thêm nhiều kiến thức lý thuyết và thực tiễn về hệ điều hành, mạng máy tính nói chung và các khái niệm về tiến trình và điều độ tiến trình, mô hình mạng ngang hàng, socket… nói riêng, đồng thời rèn luyện khả năng lập trình mạng, lập trình đa luồng với ngôn ngữ Java của mình. Có thêm những kiến thức như vậy là rất bổ ích đối với một sinh viên Công nghệ thông tin như em.

Cuối cùng, em xin gửi lời cảm ơn chân thành nhất đến cô Trần Hồ Thủy Tiên đã tận tình hướng dẫn để giúp em hoàn thành đồ án này.

TÀI LIỆU THAM KHẢO

[1] **Đặng Vũ Tùng**, Giáo trình Nguyên lý Hệ Điều Hành, NXB Hà Nội (2005).

[2] http://monhoc.vn/tai-lieu/giao-trinh-he-dieu-hanh-dong-bo-hoa-qua-trinh-1250/

[3] https://vi.scribd.com/doc/103929820/Peer2Peer-File-Sharing-System-With-Chat-Using-Java

[4] https://stackoverflow.com/