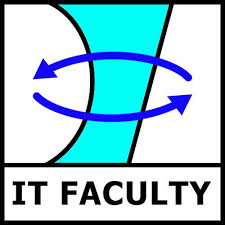
**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỒ ÁN**

**HỆ ĐIỀU HÀNH VÀ MẠNG MÁY TÍNH**

**Đề tài:**

**CHƯƠNG TRÌNH MÔ PHỎNG QUÁ TRÌNH HOẠT ĐỘNG CỦA 3 LOẠI THÔNG ĐIỆP REQ, ACQ VÀ REL**

**SINH VIÊN THỰC HIỆN:**

1. **Nguyễn Lê Tuấn Cương**
2. **Văn Ngọc Đạt**

**LỚP SINH HOẠT: 18TCLC\_DT3**

**Đà Nẵng, 10-2020**

# **LỜI MỞ ĐẦU**

Ngày nay, mạng máy tính và Internet ngày càng phát triển mạnh mẽ, cho phép chúng ta khai thác các nguồn tài nguyên vô cùng rộng lớn ngay cả ở những nơi rất xa. Với hệ tin học phân tán, thông tin không chỉ được lưu trữ độc lập trên một máy chủ mà còn phân tán trên nhiều máy chủ khác và phân bố ở những vị trí địa lý khác nhau. Một trong những lợi ích của việc phân tán dữ liệu là nhằm chia yêu cầu xử lý ra cho nhiều máy chủ nhằm tăng năng lực xử lý thông tin của hệ thống.

Đồ án lần này là sự tích hợp của các bộ môn mạng máy tính và hệ điều hành. Dưới sự hướng dẫn của ThS. Mai Văn Hà cùng các anh chị thuộc công ty Enclave, bọn em đã tiến hành nghiên cứu và hoàn thành đồ án lần này với đề tài “Xây dựng chương trình mô phỏng quá trình hoạt động của ba loại thông điệp REQ, ACQ, REL”. Vì khả năng còn hạn chế nên nhóm không tránh khỏi những thiếu sót trong lúc viết báo cáo và chương trình. Kính mong thầy xem xét, góp ý để chúng em có thể rút kinh nghiệm, hoàn thiện, hiểu rõ hơn về đề tài lần này.

Lời cuối, nhóm xin gửi lời chân thành cảm ơn đến thầy Mai Văn Hà cùng các anh chị thuộc công ty Enclave đã nhiệt tình chỉ dạy, hướng dẫn nhóm em trong việc hoàn thành đồ án. Xin chân thành cảm ơn!

# **MỤC LỤC**

[**LỜI MỞ ĐẦU** 2](#_Toc59875658)

[**MỤC LỤC** 3](#_Toc59875659)

[**DANH MỤC HÌNH VẼ** 4](#_Toc59875660)

[**GIỚI THIỆU ĐỀ TÀI** 5](#_Toc59875661)

[**1 CƠ SỞ LÝ THUYẾT** 6](#_Toc59875662)

[**1.1 Hệ phân tán** 6](#_Toc59875663)

[**1.1.1 Khái niệm hệ phân tán** 6](#_Toc59875664)

[**1.1.2** **Đặc điểm hệ phân tán** 6](#_Toc59875665)

[**1.1.2.1.** **Tính chia sẻ tài nguyên** 6](#_Toc59875666)

[**1.1.2.2.** **Tính trong suốt** 6](#_Toc59875667)

[**1.1.2.3.** **Tính mở** 7](#_Toc59875668)

[**1.1.2.4.** **Tính co giãn** 7](#_Toc59875669)

[**1.2** **Ngôn ngữ Java** 7](#_Toc59875670)

[**1.3** **Lập trình Socket** 8](#_Toc59875671)

[**1.3.1 Tổng quan về lập trình socket** 8](#_Toc59875672)

[**1.3.2 Lập trình TCP Socket với Java** 8](#_Toc59875673)

[**1.4** **Thời gian logic và trật tự sự kiện từng phần** 9](#_Toc59875674)

[**2** **CÁC GIẢI THUẬT** 11](#_Toc59875675)

[**2.1 Gắn thời gian logic với các sự kiện** 11](#_Toc59875676)

[**2.2 Thuật toán đóng dấu thời gian của Lamport** 11](#_Toc59875677)

[**2.2 Giải thuật loại trừ tương hỗ** 12](#_Toc59875678)

[**2.2.1** **Thuật toán loại trừ tương hỗ** 12](#_Toc59875679)

[**2.2.2** **Kiểm tra giải thuật** 13](#_Toc59875680)

[**2.2.3** **Phản ứng sự cố** 14](#_Toc59875681)

[**2.2.4** **Thuyết minh** 14](#_Toc59875682)

[**2.2.5** **Kết luận** 15](#_Toc59875683)

[**3** **CHƯƠNG TRÌNH VÀ KẾT QUẢ** 16](#_Toc59875684)

[**KẾT LUẬN** 18](#_Toc59875685)

[**Tài liệu tham khảo** 19](#_Toc59875686)

# **DANH MỤC HÌNH VẼ**

[Figure 1. Socket trong mô hình TCP/IP và OSI 8](file:///E:\Báo-cáo-đồ-án-PBL-4.docx#_Toc59832356)

[Figure 2. Mô phỏng quá trình trao đổi thông điệp REQ, REL, ACQ 15](file:///E:\Báo-cáo-đồ-án-PBL-4.docx#_Toc59832357)

# **GIỚI THIỆU ĐỀ TÀI**

Ngày nay, trong quá trình triển khai ứng dụng Tin học vào đời sống, các mạng máy tính được phát triển không ngừng, các tài nguyên của các máy tính trong mạng (phần cứng, phần mềm) ngày càng được mở rộng và nâng cấp, giá trị các tài nguyên này ngày càng tăng nhanh dẫn đến sự tăng trưởng vượt bậc nhu cầu chia sẻ tài nguyên và thông tin trong hệ thống thống nhất. Hệ điều hành tập trung và hệ điều hành mạng thuần túy không đáp ứng được nhu cầu đối với sự tăng trưởng đó. Việc giảm giá các trạm làm việc cũng tăng không ngừng mà từ đó làm tăng yêu cầu xử lý phân tán. Điều này tạo ra nhiều vị trí có khả năng xử lý và lưu trữ thông tin hơn mà từ đó cần thiết phải phối hợp để chia sẻ tốt tiềm năng lưu trữ và xử lý các vị trí đó. Trên cơ sở việc kết nối mạng để triển khai hệ điều hành mạng tạo nên một cơ sở kỹ thuật hạ tầng làm nền tảng phát triển hệ điều hành phân tán.

Trên cơ sở đó, nhóm chúng em chọn đề tài như sau: “Viết chương trình mô phỏng quá trình hoạt động của ba loại thông điệp REQ, REL, ACQ trong hệ tin học phân tán. Xây dựng hệ thống ba server thể hiện khả năng phát nhận thông điệp”.

# **1 CƠ SỞ LÝ THUYẾT**

## **1.1 Hệ phân tán**

### **1.1.1 Khái niệm hệ phân tán**

Hệ phân tán - hay hệ điều hành phân tán, là tổ hợp bao gồm các máy tính độc lập, kết nối lẫn nhau bằng mạng máy tính. Các phần mềm trên các máy tính khác nhau có khả năng phối hợp, chia sẻ tài nguyên. Hệ phân tán thực hiện một nhiệm vụ chung, cung cấp dịch vụ một cách thống nhất về giao diện, cách thức truy cập dịch vụ mà người sử dụng không cần phải quan tâm tới các chi tiết của hệ thống.

### **Đặc điểm hệ phân tán**

Hệ phân tán có các đặc điểm cơ bản là:

* Tính chia sẻ tài nguyên
* Tính trong suốt
* Tính mở
* Tính co giãn

#### **Tính chia sẻ tài nguyên**

Thuật ngữ tài nguyên được dùng để chỉ tất cả mọi thứ có thể được chia sẻ trong hệ phân tán, bao gồm từ các thiết bị phần cứng (Đĩa, máy in...) tới các đối tượng (file, các cửa sổ, cơ sở dữ liệu và các đối tượng dữ liệu khác).

Trong hệ phân tán, chia sẻ tài nguyên được hiểu là tài nguyên của hệ thống được các quản trị viên chia sẻ mà không bị hạn chế bởi tình trạng phân tán tài nguyên theo vị trí địa lý.

Việc chia sẻ tài nguyên trên hệ phân tán - trong đó tài nguyên bị lệ thuộc về mặt vật lý với một máy tính nào đó được thực hiện thông qua truyền thông. Để chia sẻ tài nguyên một cách hiệu quả thì mỗi tài nguyên cần phải được quản lý bởi một chương trình có giao diện truyền thông, các tài nguyên có thể truy nhập, cập nhật được một cách tin cậy và nhất quán. Quản lý tài nguyên ở đây bao gồm lập kế hoạch và dự phòng, đặt tên các lớp tài nguyên, cho phép tài nguyên được truy cập từ nơi khác, ánh xạ tên tài nguyên vào địa chỉ truyền thông….

#### **Tính trong suốt**

Đây là tính chất căn bản của hệ phân tán. Tính trong suốt của hệ phân tán được hiểu như là sự che khuất đi các thành phần riêng biệt của hệ thống máy tính (phần cứng và phần mềm) đối với người sử dụng và những người lập trình ứng dụng. Người sử dụng có quyền truy cập đến dữ liệu mà không cần biết đến sự phân tán của tất cả dữ liệu trên mạng. Hệ thống tạo cho người dùng cảm giác là dữ liệu được coi như đặt tại máy tính cục bộ của mình.

#### **Tính mở**

Tính mở của hệ phân tán là tính dễ dàng mở rộng phần cứng (thiết bị ngoại vi, bộ nhớ, các giao diện truyền thông ...) và phần mềm (các mô hình hệ điều hành, các giao thức truyền thông, các dịch vụ chia sẻ tài nguyên...) của nó. Nói một cách khác, tính mở của hệ thống phân tán mang ý nghĩa bao hàm tính dễ dàng cấu hình cả phần cứng lẫn phần mềm của nó.

Tỉnh mở của hệ phân tán được thể hiện là hệ thống có thế được tạo nên từ nhiều loại phần cứng và phần mềm của nhiều nhà cung cấp khác nhau với điều kiện các loại thành phần này phải theo một tiêu chuẩn chung.

Tính mở của hệ phân tán được thi hành dựa trên việc cung cấp cơ chế truyền thông giữa các quản trị và công khai các giao diện được dùng để truy cập tài nguyên chung.

#### **Tính co giãn**

Tính co giãn là khả năng của hệ thống có thể đáp ứng được các thay đổi của hạ tầng của môi trường xung quanh. Tính co giãn thường được xem xét dưới 3 góc độ: “co giãn về mặt quy mô, co giãn về mặt địa lý, co giãn về mặt tổ chức”.

* **Co giãn về mặt quy mô**: đảm bảo đáp ứng của hệ thống khi số lượng máy tính, số lượng người sử dụng, số lượng yêu cầu của người sử dụng gửi giữa các máy tính với nhau tăng.
* **Co giãn về mặt địa lý**: đảm bảo trao đổi thông tin trên mạng diện rộng như với mạng cục bộ.
* **Co giãn về mặt tổ chức**: khi tổ chức thay đổi hay máy tính dịch chuyển từ cùng tổ chức này sang cùng tổ chức khác. Khi đó tổ chức hệ thống thành các domain để khi cần thay đổi tổ chức ta chỉ cần thay đổi domain và thay đổi độ tin cậy giữa các domain đó.

## **Ngôn ngữ Java**

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

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

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

Ngôn ngữ lập trình Java có một số đặc trưng tiêu biểu: đơn giản, hướng đối tượng, độc lập phần cứng và hệ điều hành, mạnh mẽ, bảo mật, phân tán, đa luồng và linh động.

## **Lập trình Socket**

### **1.3.1 Tổng quan về lập trình socket**

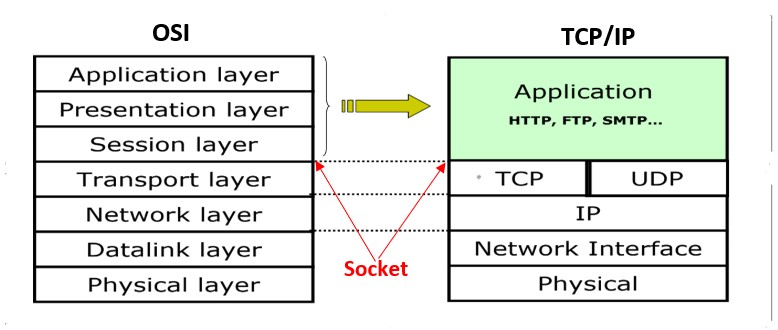
Trong hệ thống mạng máy tính tồn tại những mô hình tham chiếu có kiến trúc phần tầng (OSI, TCP/IP…) nhằm hỗ trợ chức năng trao đôi thông tin giữa các ứng dụng ở nhiều máy tính khác nhau.

Figure . Socket trong mô hình TCP/IP và OSI

Dữ liệu bên gửi sẽ được đóng gói (Encapsulation) từ tầng trên đến tầng cuối là tầng vật lí (Physical Layer), sau đó nhờ tầng vật lí này chuyển dữ liệu đến tầng vật lí máy bên nhận, bên nhận tiến hành giải mã (decapsulation) gói dữ kiện từ tầng dưới lên tầng trên cùng, là tầng ứng dụng (application layer).

Ở đây, Socket chính là cửa giao tiếp giữa tầng ứng dụng và tầng giao vận (Transport layer). Nói cách khác, Socket là giao diện do ứng dụng tạo ra trên máy trạm, quản lí bởi hệ điều hành qua đó các ứng dụng có thể gửi/nhận thông điệp đến/từ các ứng dụng khác. Ở đó, Socket sẽ được ràng buộc với một mã số cổng (Port Number) để giúp tầng giao vận định danh được ứng dụng nhận/gửi thông điệp.

### **1.3.2 Lập trình TCP Socket với Java**

Đúng như tính chất của TCP chúng ta cần có liên kết 2 chiều trước khi server và client có thể trao đổi thông điệp với nhau.

Ban đầu, phía server tạo Socket được ràng buộc với một cổng (port number) để chờ nhận yêu cầu từ phía client.

Tiếp đến phía client yêu cầu server bằng cách tạo một Socket TCP trên máy kèm với địa chỉ IP và port number của tiến tình tương ứng trên máy server. Khi client tạo Socket, client TCP tạo liên kết với server TCP và chờ chấp nhận kết nối từ server.

TCP cung cấp dịch vụ truyền dòng tin cậy và có thứ tự giữa client và server, giữa máy chủ và máy nhận chỉ có 1 địa chỉ IP duy nhất. Thêm vào đó, mỗi thông điệp truyền đi đều có xác nhận trả về.

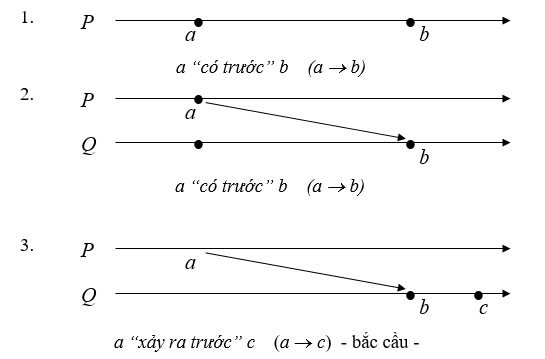
## **Thời gian logic và trật tự sự kiện từng phần**

Trong các hệ tin học tập trung, vấn đề đồng bộ hóa được giải quyết thông qua cơ chế loại trừ tương hỗ. Cơ chế này cho phép xác lập trật tự hoàn toàn các sự kiện. Tuy nhiên, trong thực tết một số hệ thống khi đồng bộ hóa chỉ đòi hỏi trật tự từng phần. Chính vì vậy, trật tự hóa từng phần giữa các sự kiện mà tiến trình của nó cần phải đồng bộ là vấn đề cần phải quan tâm giải quyết.

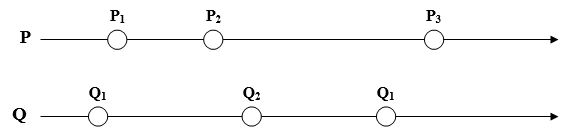
Trong hệ phân tán, việc đồng bộ hóa chủ yếu yêu cầu thiết lập một trật tự giữa các sự kiện. Giữa các trạm khác nhau, trật tự đó có thể hiện thông qua việc trao đổi các thông điệp với nhau.

Năm 1979, Lamport đã đưa ra rằng hai sự kiện từ các trạm khác nhau chỉ có thể trật tự nếu chúng được tách rời với nhau bằng cách gửi và nhận thông điệp. Ngược lại, cho dù một sự kiện xảy ra trước sự kiện khác theo thời gian, không có cách nào cho sự kiện thứ hai có thể quan sát trật tự này và vì vậy, không có sự quan sát bên ngoài của trật tự sự kiện. Nếu sự chính xác của hệ thống phụ thuộc vào trật tự được quan sát bởi con người, thì sự quan sát của Lamport không áp dụng. Tuy nhiên, sự chính xác trật tự chỉ phụ thuộc vào sự đồng bộ bên trong của các sự kiện và vì thế định nghĩa của Lamport có thể áp dụng.

Giả sử rằng, ta có thể xác định một trật tự giữa các sự kiện của hệ phân tán nhờ vào quan hệ “có trước” (→):

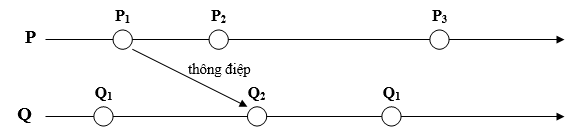


Bây giờ, ta xem xét về trật tự sự kiện:



Ta có thể thấy được trật tự từng phần của các sự kiện:

P1→P2 →P3 và Q1→Q2→Q3

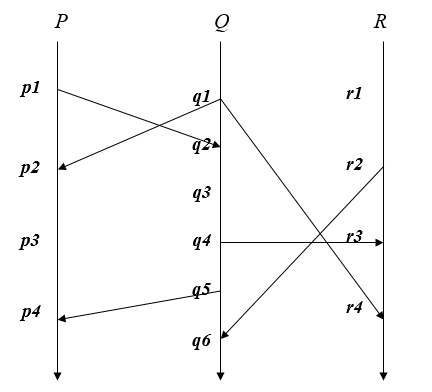


Nếu P1 là sự kiện phát thông điệp và Q2 là sự kiện nhận tương ứng thì P1→Q2

Trật tự các sự kiện được định nghĩa như sau:

* Nếu A và B là hai sự kiện cùng một trạm và A xảy ra trước thì ta có A→B.
* Nếu A phát thông điệp từ một trạm nào đó và B là nhận thông điệp thì ta có A→B.
* Nếu A→B và B→C thì A→C.

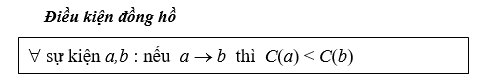
Sơ đồ của “có trước”:



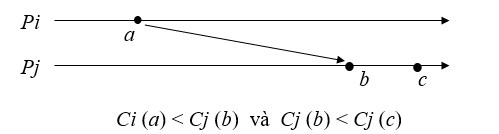
# **CÁC GIẢI THUẬT**

## **2.1 Gắn thời gian logic với các sự kiện**

Các đồng hồ logic: gán một số cho mỗi sự kiện cục bộ nhưng không liên quan đến thời gian vật lý. Hệ thống các đồng hồ logic phải chính xác:



Hay nói cách khác:



Thuật toán đóng dấu thời gian của Lamport – cung cấp một tập hợp các quy luật thực thi đơn giản.

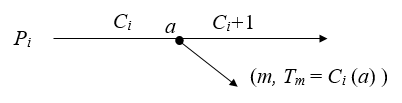
## **2.2 Thuật toán đóng dấu thời gian của Lamport**

Đồng hồ logic là thuật toán cho phép đóng dấu cho từng sự kiện trong hệ phân tán để với mỗi cặp sự kiện A và B ta có: nếu A có trước B(A→B) thì đồng hồ logic của A nhỏ hơn của B. Vì thế, ta có các quy luật khi đóng dấu thời gian như sau:

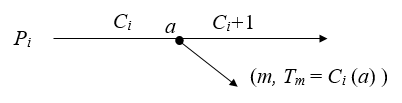
* Quy luật 1: Mỗi tiến trình Pi gia tăng thêm Ci thêm một trị số giữa hai sự kiện thành công.



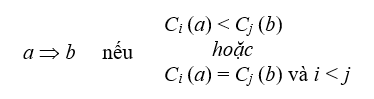
* Quy luật 2: Mỗi tiến trình Pi đóng dấu thời gian cho các thông điệp vừa gửi đi.



* Quy luật 3: Khi nhận được thông điệp m, tiến trình Pj đặt lại giá trị Cj:  
   Cj = max(Cj, Tm) +1



* Trật tự sự kiện toàn bộ: giả sử là các đồng hồ thỏa mãn Điều kiện Đồng hồ, ta định nghĩa quan hệ sau:



## **2.2 Giải thuật loại trừ tương hỗ**

Loại trừ tương hỗ (Mutual Exclusion): là quá trình truy cập đồng thời của các tiến trình với một tài nguyên hoặc dữ liệu được chia sẻ, thực hiện theo cách loại trừ lẫn nhau. Trong hệ thống phân tán không có các biến chia sẻ có thể sử dụng để thực hiện loại trừ lẫn nhau.

Loại trừ tương hỗ có thể được thực hiện trên một trạm trung tâm có nhiệm vụ nhận tất cả các thông điệp và khuyến nghị giải phóng. Trạm này duy trì một hàng đợi, sắp xếp các yêu cầu theo trật tự đến và phục vụ cho từng thông điệp một trong trật tự này.

Phân tán giải thuật này kéo theo việc phân tán các chức năng cung cấp mà cần phải điều khiển hàng đợi trên trạm. Do vậy, một trạm chuyên cho việc nhận các yêu cầu và khuyến nghị giải phóng trên tất cả các trạm còn lại. Một trật tự giống nhau trên các trạm chỉ đạt được nếu ta áp dụng dấu trong các thông điệp bởi các đồng hồ logic truyền và đánh số các trạm. Quan hệ trật tự toàn bộ được định nghĩa, thêm vào đó, để cho một trạm có thể ra quyết định bằng việc tham chiếu duy nhất vào hàng đợi của mình, nó còn cần phải nhận một thông điệp của từng trạm khẳng định rằng không có thông điệp nào trước các thông điệp khác mà còn đang quá cảnh trên đường.

### **Thuật toán loại trừ tương hỗ**

1. Trạm i của mạng có thể gửi thông điệp cho các trạm khác thông điệp có dạng (T, Hi, i), trong đó Hi là dấu của thông điệp có nghĩa là đồng hồ logic của nó và T có thể nhận một trong 3 giá trị REQ, REL, ACQ.

|  |  |  |
| --- | --- | --- |
| STT | Tên thông điệp | Chức năng |
| 1 | REQ | Thông điệp REQ gửi yêu cầu đến các trạm, khi trạm i muốn vào trong đoạn găng |
| 2 | REL | Thông điệp REL giải phóng được phát đi cho tất cả các trạm, khi trạm i đã rời khỏi đoạn găng |
| 3 | ACQ | Thông điệp ACQ phản hồi được gửi bởi trại j cho trạm i, khi trạm j đã nhận được từ trạm i thông điệp REQ |

1. Mỗi trạm quản lý một hàng đợi các thông điệp được sắp xếp theo quan hệ <thời gian, số> của từng thông điệp. Theo cấu trúc, hàng đợi luôn luôn chứa một thông điệp và chỉ một thường trực trên mỗi trạm, bao gồm trạm cục bộ. Khi có một thông điệp được gửi đi bởi trạm i, đồng thời nó cũng được ghi trong hàng đợi của trạm này.   
   Giả sử rằng mỗi hàng đợi bann đầu chứa các thông điệp:

Mi = (REL, Hinit, i)

Trong đó, i, Hinit là thời điểm khởi sự giống nhau cho tất cả các trạm.

1. Trên mỗi trạm, khi nhận được một thông điệp dạng (REQ, Hi, i) hay(REL, Hi, i), thông điệp này thay thế thông điệp Mi bất chấp nó là gì. Khi nhận thông điệp loại (ACQ, Hi, i), thông điệp này thay thế Mi ngoại trừ Mi là một yêu cầu mà trong trường hợp đó ACQ bị bỏ qua. Do vậy, ta có thể tiết kiệm việc gửi thông điệp ACQ cho trạm i khi trạm này đã gửi một thông điệp REQ và không còn thông điệp REL.
2. Trạm i được quyền vào đoạn găng khi thông điệp REQ của nó đến trước theo nghĩa của quan hệ <thời gian, số> của tất cả các thông điệp khác trong hàng đợi của nó.

### **Kiểm tra giải thuật**

Các yêu cầu vào đoạn găng được xử lý theo trật tự FIFO và theo quan hệ =>. Để chứng minh điều đó, ta lưu ý rằng khi trạm i quyết định vào đoạn găng, nó không thể đưa vào trong mạng yêu cầu REQ nào trước bối cảnh sau. Thực tế, khi trạm i vào đoạn găng có nghĩa là nó đã nhận thông điệp từ tất cả các trạm khác và tất cả các thông điệp đều sau thông điệp REQ của riêng nó.

Ta kiểm tra các đặc tính sau đây:

1. Trạm i đang ở trong đoạn găng là trạm duy nhất nằm trong đoạn găng ấy. Thực tế cho thấy thông điệp REQ được phát bởi i vẫn tiếp tục tồn tại trong tất cả các hàng đợi cho đến khi nó được thay thế bởi thông điệp REL.

2. Trạm đã yêu cầu vào đoạn găng phải đảm bảo thời hạn và phải ra khỏi đoạn găng sau một khoản thời gian xác định. Thuật toán thể hiện tính đồng đều và tránh được tổn thất.

### **Phản ứng sự cố**

Sự cố xảy ra với trạm chưa vào được trong đoạn găng không làm rối loạn hoạt động của giải thuật với điều kiện là nó gây ra việc truyền thông điệp đặc biệt vang\_mat (vắng) cho việc chuyển tải ở mạng giao vận. Do có trang bị như thế, việc vào đoạn găng trở nên không được nhanh chóng và dễ dàng cho các trạm khác. Nếu trạm có sự cố đã gửi yêu cầu, thì nó kết thúc với lý do trở thành trước đối với tất cả các trạm khác.

Khi một trạm lại được đưa vào trong mạng sau khi đã có sự cố và đã khắc phục, nó cần phải kiến tạo lại trạng thái hiện hành của các yêu cầu. Để đảm bảo điều đó, nó phát đi thông điệp vao\_lai (xin vào lại) và để trả lời, các trạm gửi hoặc thời gian của yêu cầu cuối cùng của nó REQ không được thõa mãn (nếu có tồn tại một REQ) hoặc một thông điệp REL. Mạng cần phải bổ khuyết cho các trạm bị sự cố bằng cách gửi thông điệp vang\_mat. Khi nó đã nhận tất cả các trả lời cho thông điệp vao\_lai, trạm vừa đưa vào đó có thể bắt đầu lại bằng các yêu cầu.

### **Thuyết minh**

Để mô tả cho giải thuật này, điều kiện đủ là một trạm cần biết gần đúng trạng thái của các trạm khác trên mạng. Sự xấp xỉ này không đặt ra trạng thái mặc định cho loại trừ tương hổ. Nó có thể làm chậm một tí việc vào đoạn găng của một trạm.

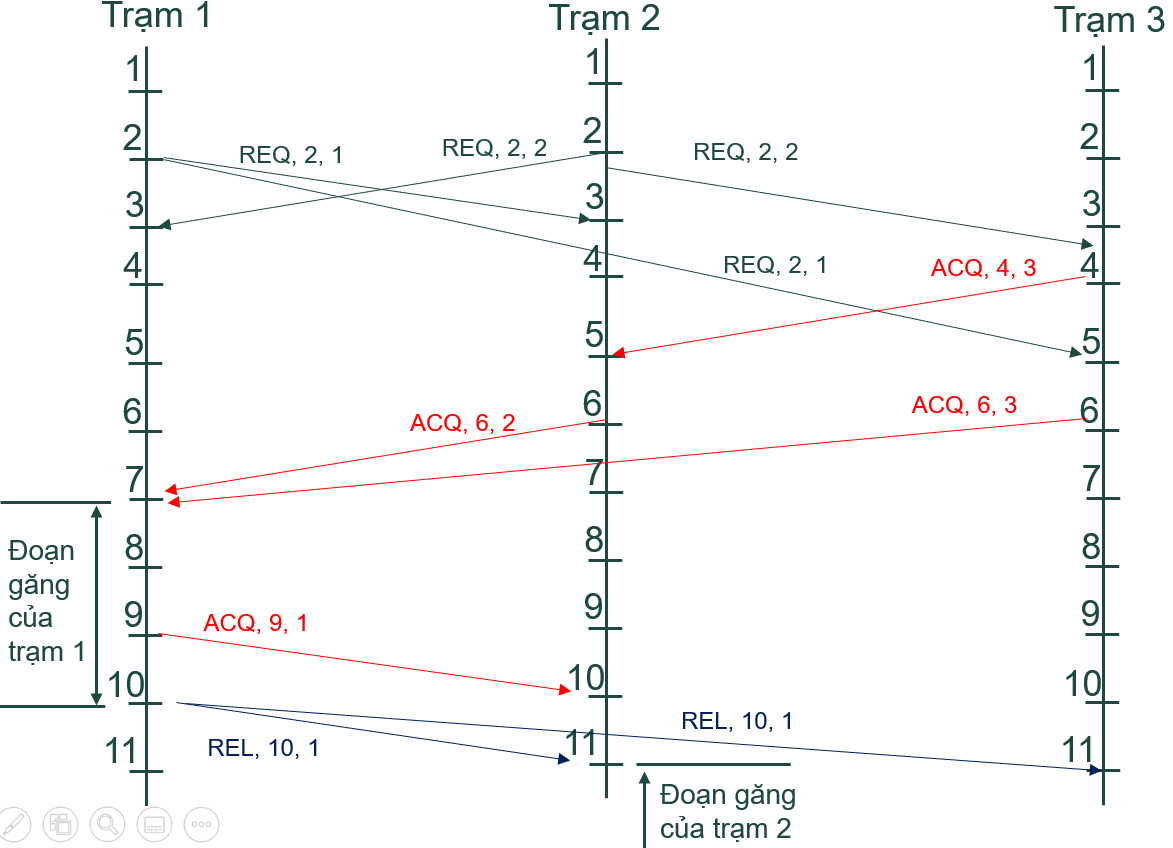
Ví dụ: Xét một mạng gồm 3 trạm, trong số đó trạm 1 và trạm 2 yêu cầu vào đoạn găng tại thời điểm 2 của đồng hồ Logic của chúng. Tập hợp các thông điệp được truyền đi giữa chúng với nhau được mô tả theo hình vẽ sau và ta cũng có thể tìm thấy một số trạng thái của các hàng đợi thông điệp:

Figure . Mô phỏng quá trình trao đổi thông điệp REQ, REL, ACQ

Ta nhận thấy rằng:

* Trạm 1 vào đoạn găng tại = 7
* Trạm 2 vào đoạn găng tại = 11

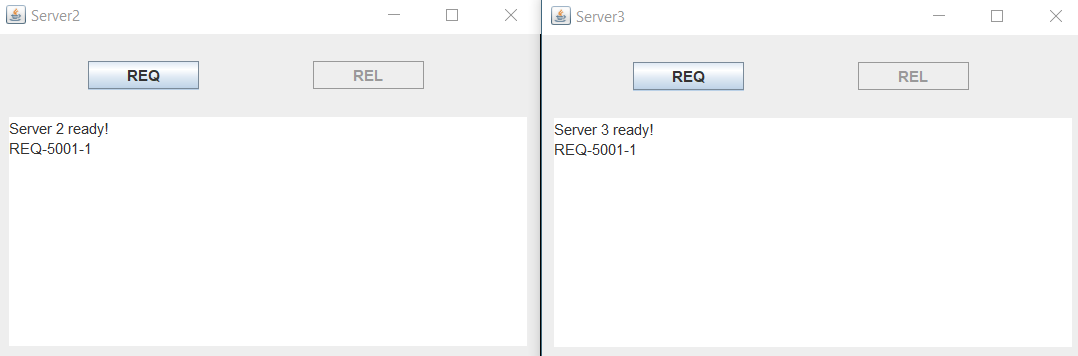
### **Kết luận**

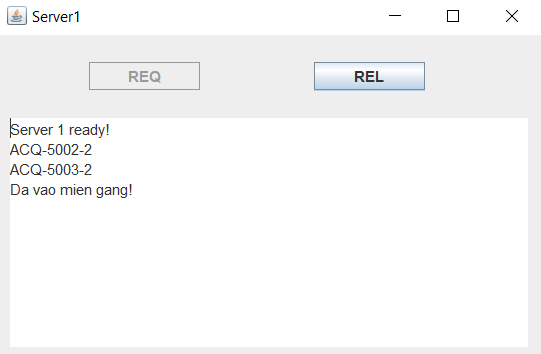
Các dấu được cung cấp bởi đồng hồ Logic cho phép đánh dấu các sự kiện và xác định một tổng quát chặt chẽ. Nhưng tại đây không có quan hệ nào giữa các sự kiện và các giá trị của dấu. Trên một trạm cho trước, việc nhận một thông điệp có đóng dấu không thể cho nó biết còn sự kiện nào trước sự kiện đó đang ở trên đường. Như thế, ta còn phải nhận thông điệp từ các trạm khác còn lại.

# **CHƯƠNG TRÌNH VÀ KẾT QUẢ**

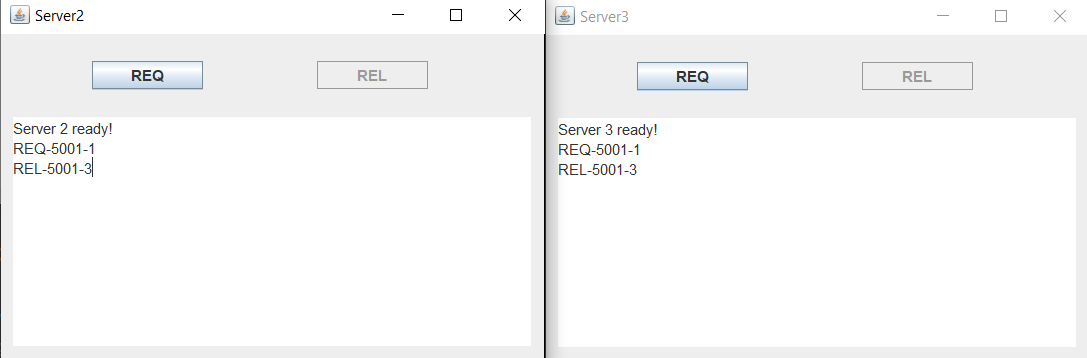
 Khởi động 3 server:

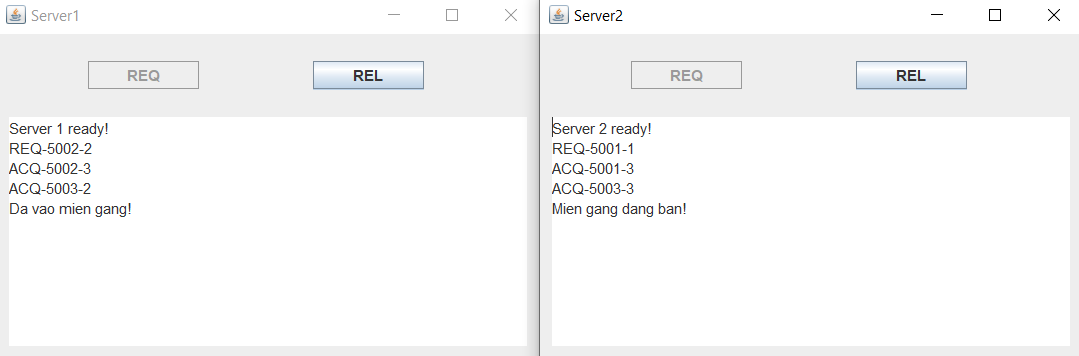
Server 1 gửi thông điệp REQ:



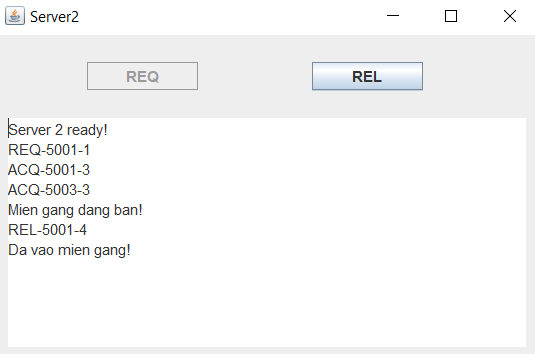
Server 2 và server 3 nhận thông điệp và phản hồi thông điệpACQ, server 1 vào miền găng:

Server 1 gửi thông điệp REL cho server 2 và server 3 khi rời miền găng:



Trường hợp server 1 và server 2 cùng gửi thông điệp điệp REQ, vì server gửi trước nên server 2 vào trạng thái chờ:

Sau khi server 1 gửi thông điệp REL rời khỏi đoạn găng, server 2 nhận được thông điệp ACQ từ server 1 và server 3 và tiến vào đoạn găng:



# **KẾT LUẬN**

Với đề tài xây dựng chương trình mô phỏng quá trình hoạt động của ba loại thông điệp REQ, REL, ACQ trong hệ tin học phân tán, sau quá trình nghiên cứu cơ sở lý thuyết, giải thuật cùng với sự hướng dẫn của thầy Mai Văn Hà và công ty Enclave, nhóm chúng em đã rút ra được một số kết luận như sau:

* Về mặt lý thuyết, ta có thể thấy được ứng dụng của hệ phân tán trong việc yêu cầu sử dụng tài nguyên dùng chung qua việc trao đổi các thông điệp giữa các server với nhau. Bằng các giải thuật đóng dấu thời gian logic của lamport hay loại trừ tương hỗ, ta có thể xử lý được bài toán tranh chấp tài nguyên giữa các server và từ đó tránh được xảy ra tắc nghẽn trong các luồng xử lý. Ở các hệ thống phân tán nhỏ như mạng LAN, ta thường dùng Socket Communication. Nhưng đối với các hệ thống lớn hơn mạng mạng WAN, thì người ta sẽ dùng các lời gọi từ xa (Remote Invacation) như RPC, Java RMI.
* Về mặt thực tế, từ bài toán lý thuyết kể trên, hệ phân tán đã được áp dụng để giải quyết các bài toán lớn hơn như hệ thống ATM, hệ thống gửi – rút tiền tại các ngân hàng, hệ thống đặt vé máy bay…

Vai trò của hệ phân tán đối với sự phát triển của máy tính là không thể bàn cãi, đặc biệt khi quy mô các hệ thống máy tính ngày càng lớn như hiện nay. Các đặc tính của hệ phân tán giúp nó có điều kiện thuận lợi cho việc phát triển. Tuy nhiên, bên cạnh các đặc tính thuận lợi của hệ phân tán, vẫn còn tồn tại rất nhiều trong việc xây dựng hệ thống để đảm bảo các tính chất tốt đó của hệ. Do đó, việc nghiên cứu hệ phân tán ngày càng đóng vai trò quan trọng trong việc xây dựng các mạng máy tính.

Trong các đặc tính của hệ phân tán, tính trong suốt là đặc tính quan trọng, tạo điều kiện để khai thác hiệu quả hệ phân tán đồng thời đảm bảo tính tiện dụng cho người dùng. Việc nghiên cứu tính trong suốt tạo điều kiện để phát triển và nâng cao hiệu năng của hệ thống, đồng thời vẫn đảm bảo hệ phân tán là hệ thống phục vụ người dùng.

# **Tài liệu tham khảo**

[1] Tiểu luận hệ tin học phân tán, Nguyễn Đăng Huy, ĐẠI HỌC BÁCH KHOA – ĐẠI HỌC ĐÀ NẴNG

[2] Song song hóa thuật toán lamport trong loại trừ tương hỗ phân tán, Đặng Hùng Vĩ, ĐẠI HỌC SƯ PHẠM – ĐẠI HỌC ĐÀ NẴNG

[3] Tài liệu lập trình Java Core, Java-Notes