**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NÔI**

**Viện Công nghệ Thông tin và Truyền thông**

****

**Báo cáo môn học: Các Hệ Phân Tán**

**Đề tài:**  **Viết ứng dụng phân tán đơn giản chia sẻ và đồng bộ hóa dữ liệu**

Sinh viên thực hiện:

**Lại Trung Kiên MSSV: 20142398**

**Nguyễn Văn Đàn MSSV: 20140914**

**Chu Quang Tỉnh MSSV: 20144517**

Giáo viên hướng dẫn: [**TS. Nguyễn Bình Minh**](http://is.hust.edu.vn/danh-muc/nhan-su/can-bo-co-huu/56-ts--nguyen-binh-minh.html)

**Hà Nội - 2017**

[I. Mô tả bài toán: 3](#_Toc481912458)

[II.Trình bày giải pháp: 3](#_Toc481912459)

[1. Tổng quan về các vấn đề lớn cần phải giải quyết của chương trình 3](#_Toc481912460)

[2. Các phương pháp giải quyết vấn đề trên và các yêu cầu 3](#_Toc481912461)

[2.1 Giải quyết vấn đề 1 ở trên và yêu cầu cho phép nhiều hơn 1 client kết nối tới server 3](#_Toc481912462)

[2.2 Giải quyết vấn đề đồng bộ hóa thời gian 6](#_Toc481912463)

[2.3 Xử lý vấn đề tranh chấp giữa hai client và yêu cầu đồng bộ giữa trên kích thước và thời gian tạo file 7](#_Toc481912464)

[III.Phân chia công việc 7](#_Toc481912465)

[IV. Hướng dẫn chạy chương trình 7](#_Toc481912466)

# Mô tả bài toán:

* Viết ứng dụng phân tán đơn giản chia sẻ và đồng bộ hóa dữ liệu. Yêu cầu:
* Dữ liệu được phía client đưa lên máy Server.
* Server thực hiện việc đồng bộ hóa tự động giữa các máy khách (ít nhất 2 clients) dựa trên 2 yếu tố đồng thời:

+ kích thước của file dữ liệu

+ thời gian thay đổi file dữ liệu.

* Các client tự động upload và download dữ liệu
* Cung cấp giao diện đơn giản (dạng Explorer) để hiển thị dữ liệu dạng phía Client
* Xử lý tương tranh dữ liệu giữa 2 clients.
* Cung cấp giao diện đơn giản (có thể là CLI) để hiển thị dữ liệu phía Server
* Cho phép người sử dụng thoát ứng dụng tự động đồng bộ phía Client

# II.Trình bày giải pháp:

## Tổng quan về các vấn đề lớn cần phải giải quyết của chương trình

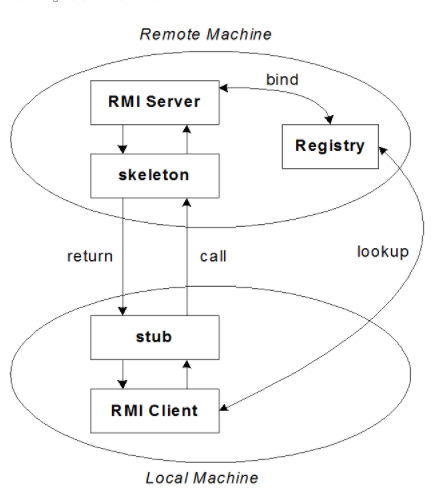
* Làm thế nào để tạo ra server cho phép kết nối nhiều client và lại có thể phân biệt được các dữ liệu đó gửi cho client nào
* Làm sao để thống nhất được thời gian giữa các client và server
* Xử lý tương tranh giữa 2 client thế nào khi cả hai client đều muốn ghi dư liệu vào 1 file.

## Các phương pháp giải quyết vấn đề trên và các yêu cầu

## Giải quyết vấn đề 1 ở trên và yêu cầu cho phép nhiều hơn 1 client kết nối tới server

* Dùng công nghệ java rmi và thư viện commons-logging-1.2.jar
* RMI : (Remote Method Invocation). RMI giúp cho việc truyền thông giữa các đối tượng phân tán được dễ dàng hơn. RMI là API bậc cao được xây dựng dựa trên lập trình Socket. Hệ thống RMI cho phép một đối tượng chạy trên một máy ảo Java này có thể kích hoạt một phương thức của một đối tượng đang chạy trên một máy ảo Java khác. Đối tượng có phương thức được gọi từ xa gọi là các đối tượng ở xa (Remote Object).
* Kịch bản đơn giản của java rmi như sau:
* Đầu tiên server phải đăng ký cổng và tên trên máy bằng 2 hàm LocateRegistry.creatRegistry([port]), rebind(“[url]”, Object) và tạo ra skeleton (tạo mặc định )
* Client tra cứu tên máy chủ để tạo liên kết thực hiện lời gọi từ xa bằng hàm lookup(String name) và tao ra Stub (tạo mặc định)
* Client gọi remote điều khiển từ xa sẽ tuần tự các tham số gửi cho skeleton, skeleton gọi phương thức điều khiển và tuần tự trả về kết quả cho client

Mô tả như trong hình sau:



* Chạy chương trình RMI cần 3 xử lý sau :
* Client
* Server
* Object Registry, rmiregistry, cái như là DNS server cho objects
* Với phía máy chủ
* định nghĩa remote interface

\_ Phải là public

\_ Phải kế thừa từ java.rmi.Remote

\_ Mọi phương thức khai báo đều phải throws Java.rmi.RemoteException

* Thiết kế remote object bằng cách kế thừa remote interface

\_Phải kế thừa Remote interface

\_Nên kế thừa java.rmi.server.UnicastRemoteObject

\_Có thể tao ra các phương thức truy cập đia phương không có trong Remote interface

* Thiết kế chương trình server hoàn thiện

\_Tạo ra 1 server để Remote ObjectServer nên dùng hàm rebind

* Về phía client
  + Thết kế remote interface và remote object tương tự như server
* Thiết kế chương trình client
  + Dùng hàm lookup để remote object vào server đã được đăng ký
* Dùng RMI ta đã giải quyết được vấn đề client có thể kết nối tới server. Để cho nhiều client kết nối đến server cho các Object được kế thừa các Thread trong java mỗi client là 1 luồng. Đến đây lại có vấn đề nảy sinh ra là phân biệt giữa các client khi các client truyền dữ liệu phải mở InputStream và OuputStream truyền dữ liệu làm thế nào phân biệt được InputStream và OutputStream cho luồng nào thì thư viện commons-logging giải quyết vấn đề đó

Phần trên tham khảo trên trang web <http://www.ejbtutorial.com/category/java-rmi>

## 2.2 Giải quyết vấn đề đồng bộ hóa thời gian

* Giải pháp sử dụng NTP và thuật toán Christian và thư viện joda-time-2.7.jar
* NTP (Network Time Protocol - Giao thức đồng bộ thời gian mạng) : NTP sử dụng kiến trúc phân cấp, phân lớp cho các cấp nguồn đồng bộ, mỗi một cấp trong phân cấp này được gọi là một "statum' và được gán một số của cấp bắt đầu từ 0 là cấp cao nhất. Cấp stratum chỉ ra nó đã qua bao nhiêu trung gian để đến được cấp tham chiếu và cấp stratum cũng giúp tránh tham chiếu vòng trong phân cấp. Chú ý rằng cấp stratum không có ý nghĩa chỉ chất lượng hay độ ổn định, dễ dàng tim thấy một nguồn đồng bộ "stratum 3" có chất lượng tốt hơn một nguồn "stratum 2" khác. Định nghĩa về statum này cũng khác với stratum dùng trong đồng bộ viễn thông. Xem chi tiết trên <https://vi.wikipedia.org/wiki/NTP> và slide của thầy Nguyễn Bình Minh còn source code tham khảo tại <https://commons.apache.org/proper/commons-net/examples/ntp/NTPClient.java>
* Thuật toán Christian :

Đầu tiên ta tính độ lệch bằng độ trên trên đường truyền mạng từ lúc gửi đi đến lục trả về

Nếu khác 0 thì cộng thêm vào thời gian hiện tại của hệ thống

Xem chi tiết trong slide thầy Nguyễn Bình Minh

* Thư viện joda-time-2.7.jar : Thư viện là thư viện nâng cao của thư viên đơn giản ngày và giờ của java nó cung cấp các nhiều định dạnh thời gian, cung cấp cho thời gian dựa trên múi giờ, và nó có tạo ra một mô phỏng local time trên máy tiện lợi cho việc test các hệ thống phân tán. Đối với chương trình này dùng để tao ra nhiều client có thời gian khác nhau để kiểm tra tính đồng bộ của thời gian

## 2.3 Xử lý vấn đề tranh chấp giữa hai client và yêu cầu đồng bộ giữa trên kích thước và thời gian tạo file

* Việc đồng bộ giữa client và server được tham khảo trên trang <http://www.java2s.com/Tutorial/Java/0180__File/Utilityclassforsynchronizingfilesdirectories.htm>
* đã viết ngon rồi ta chỉ cần thêm vài câu lệnh if else để xử lý theo hai tiêu chí kích thước và thời gian tạo file rồi gọi đến hàm Syschronization([paramater], [paramater]);

Còn vấn đề xử lý tranh chấp giữa 2 client thì RMI có hỗ trợ ta chỉ cần implement Serializable và khai báo thêm từ khóa synchronized nhưng RMI chỉ thực sự hiêu quả khi ta truyền dữ liệu ít hay chỉ là truyền url hay đường dẫn cho file. Nếu truyền dữ liệu quá nhiều sẽ dẫn đến ngoại lệ OutOfMemoryError để giải quyết vấn đề này ta dùng thư viện rmiio-2.0.5.jar

# III.Phân chia công việc

1. Lại Trung Kiên xử lý vấn đề 1
2. Phạm Văn Đàn xử lý vấn đề 2
3. Chu Quang Tỉnh xử lý vấn đề 3

# IV. Hướng dẫn chạy chương trình

1. Nếu trên máy window chưa có NTP thì chạy ntp-4.2.8p10-win32-setup.exe chứa trong CD còn trên linux dùng sudo apt-get install ntp
2. Mở NetBeans rồi open project đó
3. Cài đặt lại đường dẫn tới file security.policy trong mã nguồn System.setProperty() tại RunServer.java dòng 175 và RunClient.java dòng 43
4. Rồi chạy file RunServer.java và RunClient.java

**Tài liệu tham khảo**

1. slide thầy Nguyễn Bình Minh
2. <http://www.java2s.com/Tutorial/Java/0180__File/Utilityclassforsynchronizingfilesdirectories.htm>
3. <https://vi.wikipedia.org/wiki/NTP>
4. <http://www.ejbtutorial.com/category/java-rmi>
5. <https://commons.apache.org/proper/commons-net/examples/ntp/NTPClient.java>