**Báo cáo**

**Công nghệ sử dụng**

**Rmi**

Giới thiệu về rmi

RMI - Remote Method Invocation là một kĩ thuật cài đặt các đối tượng phân tán trong Java. RMI là một phần của bộ J2SDK và là hàm thư viện hỗ trợ các lời gọi phương thức từ xa và trả về giá trị cho các ứng dụng tính toán phân tán. Để sử dụng rmi thì ngôn ngữ lập trình Java phải được sử dụng ở cả hai phía gọi và phía bên phương thức được gọi.

Để giải quyết một số vấn đề trong việc truyền thông giữa Client/Server. RMI không gọi trực tiếp mà thông qua lớp trung gian. Lớp này tồn tại ở cả hai phía Client và Server. Lớp ở máy Client gọi là Stub, lớp ở máy Server gọi là Skel (Skeletion).

Đặc tính rmi

RMI là mô hình đối tượng phân tán của Java, RMI giup cho việc giao tiếp giuẵ các đối tượng phân tán trong môi trường internet trể nên dễ dàng hơn.

RMI là API bậc cao được xây dựng dựa trên lập trình Socket.

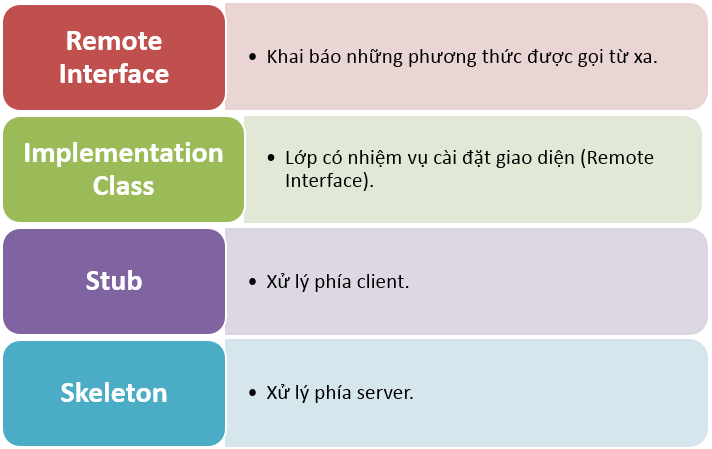
RMI không những cho phép chúng ta truyền dữ liệu giữa các đối tượng trên các hệ thống máy tính khác nhau, mà còn triệu gọi các phương thức trong các đối tượng ở xa (Remote Object).

Việc truyền dữ liệu giữa các máy khác nhau được xử lý một cách trong suốt bởi máy ảo Java (Java virtual machine).

Tương tự như mô hình Client/Server, RMI vẫn duy trì khái niệm của Client và Server, tuy nhiên cách tiếp cận của RMI linh hoạt hơn, mềm dẻo hơn so với môt hình Client/Server.

Một điều thuận lợi quan trong nhất của RMI là nó cung cấp cơ chế callbacks, nó cho phép Server triệu gọi các phương thức ở Client.

Kiến trúc rmi



Remote inteface : là lớp giao diên khai báo tất cả các hàm Server cung cấp . Client có thể gọi hàm này như 1 lời gọi thông thường

Ghi chú : Tất cả các hàm trong giao diện này đều ném ra 1 ngoại lệ hay lỗi RemoteException

Impements class : Lớp được cài đặt dựa trên Remote inteface ,thực hiện các công việc đã được chỉ định. Thường nằm trên server

Stub và skeleton : Các lớp trên server và client dùng để giao tiếp . được java tự động tạo ra

Ngoài ra còn có

RMI registry: Bộ đăng kí này sẽ đăng kí một Remote object với Naming Registry. Giúp các Remote object được chấp nhận khi gọi các method từ xa.

Cách đăng kí rất đơn gian ta chi cần dùng lệnh registry.bind("Name", Object);

Đánh giá về rmi

Ưu điểm :

* Đơn giản ,giúp phát triển ứng dụng nhanh chóng
* Việc thao tác dữ liệu trở nên đễ đang hơn . không phải thao tác thực tiếp với byte nữa

Tăng tính trong suốt .Tạo cảm giác làm việc trên 1 máy

Nhược điểm

Các chi phí về xử lý , tính toán ,.. sẽ cao hơn lập trình socket truyền thống

Bảo mật kém

**Vào ra với Rmi**

1 điểm cần lưu ý nữa là khi sử dụng rmi thì dữ liệu truyền vào không bị thay đổi hay có thể hiểu giống như lập trinh với việc truyền tham trị . Do đó tất cả kết quả trả về nằm trong hàm return . Do đó không thể tránh khỏi việc đối trượng trả vế sẽ có cấu trúc phức tạp

Trong rmi có hỗ trợ việc truyền các object tuy nhiên điều này là có rất nhiều hạn chế như 1 số đối tượng không cài đặt lớp **Serializable** thì không thể truyền được do Rmi sẽ chuyển đổi đối tượng thành mảng byte truyền đi và tổng hợp lại phía nhận , Chỉ nhứng object nào có cài đặt lớp này thì mới có thể gửi đi

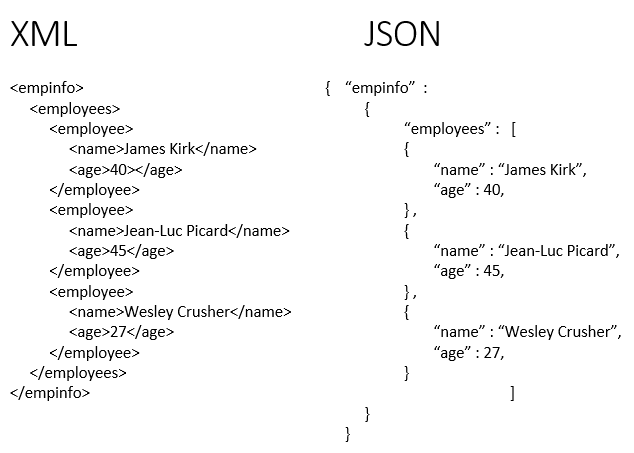
Việc này gây khá nhiểu khó khăn khi xây dựng phần mêm do có những lúc cần phải gửi đi 1 số đối tượng trong các thư viện khác nhưng các tác giả của các thư viện đó thường ko cài đặt lớp **Serializable**

Giải thích cho việc này là do đa số các thư viện đều hoạt động trên môi trường 1 máy Và việc cài đạt giao diện Serializable sẽ gấy ra nhiều khó khăn không cần thiết do các thuộc tính trong cũng phải cài đặt Serializable

1 giải pháp là ta có thể tải các thư viện đó về và sửa mã nguồn tuy nhiên việc này rất tốn thời gian và nhiều thư viện rất khó tìm mã nguồn hoặc tác giả của thư viện đó không cho phép sửa đổi

Phương án được khả thi ở đâý chuyển đối tượng java về các đố tượng có cấu trúc như XML , hay Json , Hai cấu trúc này đươc áp dụng rộng rãi . Có cấu trúc rõ ràng phù hợp với kiểu hướng đối tượng

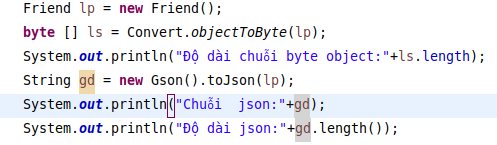
So sánh XML và Json



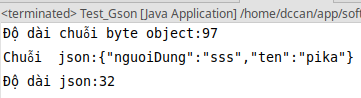
Có thể thấy ngay là json có cấu trúc khá gọn gàng hơn do đó để giảm thiểu lượng thông tin phải truyền ,Em sử dụng json để thực hiện đề tài

Thêm nữa sau khi làm 1 số kiểm tra với thư viến Gson ( thư viện cho phép Chuyển đổi Object java thành Json) thì trong các lần kiểm tra đó độ dài chuỗi thu về đều nhỏ hơn so với phường pháp chuyển đổi đối tượng thành mảng byte

Code



Kêt quả thu về



Trong đồ án của em thường xuyên phải sử dụng các đổi tượng có cấu trúc gần giống mẫu thử trên để truyền thông tin ,nên có thể kết luận việc sử dụng json sẽ làm giảm lượng thông tin cần truyền tải , tuy nhiên lại mất thêm 1 chút chi phí khi chuyển đổi từ Json sang đối tượng java . May mắn là thời gian tính toán của việc này rất bé , thường nhỏ hơn 1ms

**Thêm :**

Và trong quá trình phát triển ứng dụng với java có 1 việc mà ta ít khi chú ý đến đó là các file được biên dịch bỏi bộ JDK thấp hơn có thể chạy bởi JVM đời cao hơn nhưng điều ngược lại là không thể

Câu hỏi đặt ra là có thể giao tiếp rmi giũa hai bộ jvm khác nhau hay không , theo kết quả tiến hành kiểm tra của em thì là có thể

Sau khi thử nghiệm với JDK 8,9,11 thì hệ trống đều hoạt động trơn chu

**SSL, chứng chỉ SSL là gì?**

SSL viết tắt của cum từ Secure Sockets Layer là một tiêu chuẩn trong kỹ thuật bảo mật dùng cho được thiết lập một kết nối được mã hóa giữa một máy chủ web và trình duyệt. Kết nối này được đảm bảo các dữ liệu gửi giữa máy chủ web và trình duyệt là riêng tư và toàn vẹn. SSL là một chuẩn công nghiệp được sử dụng ở hàng triệu các website nhằm bảo vệ thông tin các giao dịch trực tuyến của khách hàng. Để tạo được một kết nối SSL đến một máy chủ web cần có một chứng chỉ SSL.

SSL sử dụng các thuật toán mã hóa công khai, khi máy chủ web được cài đặt SSL, nó sẽ tạo ra hai khóa mã hóa: một khóa công khai (public key) và một khóa bí mật (private key). Các khóa mã hóa được tạo ra khi bạn hoàn thành một số câu hỏi để nhận diện website cũng như công ty của bạn. Khóa công khai không cần thiết phải bảo mật nên được đưa vào file yêu cầu cấp chứng chỉ CSR (Certificate Signing Request). File này sẽ được gửi đến các tổ chức cấp chứng chỉ CA (Certification Authority) để xác thực và phát hành chứng chỉ SSL chứa các thông tin khi tạo CSR (tên website, tên công ty, địa chỉ…) và cho phép máy chủ của bạn được sử dụng SSL.

Thông thường, chứng chỉ SSL sẽ bao gồm một số các thông tin như tên miền, tên công ty, địa chỉ, công ty, quốc gia, ngoài ra thông tin về ngày hết hạn chứng chỉ cũng được đưa vào. Khi một trình duyệt kết nối đến một website được bảo mật, nó sẽ nhận được chứng chỉ SSL, trình duyệt sẽ kiểm tra ngày hết hạn và tổ chức cấp chứng chỉ CA là ai, nếu không xác thực được trình duyệt sẽ thông báo cho người dùng là website này không được bảo mật bằng SSL.

**Kĩ thuật Reflection**

**Java Reflection là gì?**

Java Refelection là một tính năng (gọi là API hay thư viện cũng được) trong Java. Java Reflection cho phép truy cập các thông tin của đối tượng (tên class, các field, các method) và chỉnh sửa các field của đối tượng (kể cả các field private) trong quá trình run time.

Ta có thể áp dụng Java Reflection trong những trường hợp không biết object được xử lý là gì. (tên class là gì, ở package nào, có những field nào, method nào…).

Ví dụ ta muốn viết một hàm copy 2 đối tượng có thể dùng cho loại đối tượng khác nhau. Thì ta cần phải biết 2 đối tượng có cùng kiểu không, có những field nào, lấy và copy giá trị từng field.

Ngoài ra, với các field, method có modifier là private thì ta không thể truy cập ở bên ngoài class đó. Trong những trường hợp bắt buộc phải gọi, truy cập các field, method private ở bên ngoài class đó thì Reflection là một giải pháp.

Một số framework sử dụng Java Reflection:

* Spring
* JUnit
* Tomcat
* Eclipse (dùng để autocomplete)
* …

## **Hạn chế, nhược điểm của Java Reflection**

Trong trường hợp đã biết rõ cấu trúc class, có quyền truy cập các field, method thì ta không nên sử dụng Java Reflection bởi các lý do sau:

* ****Hiệu năng thấp:**** Ví dụ nó phải quét classpath để tìm class.
* ****Các vấn đề bảo mật:**** Việc chỉnh sửa class/object trong quá trình runtime có thể ảnh hưởng tới các thread … khiến cho ứng dụng bị fail.
* ****Khó bảo trì:**** Việc Reflection khá khó hiểu với người mới và không dễ để debug, nên sẽ rất khó để có thể tìm ra lỗi. Ngoài ra chúng ta cũng không thể check được một số lỗi trong quá trình compile (không tìm thấy class, không tìm thấy field…)

## **Các thành phần trong Java Reflection**

Tương ứng với các thành phần trong một class, thì Java Reflection cũng cung cấp các class tương ứng để ta có thể xử lý:

* ****Class:**** Đại diện cho class/interface để lấy ra các thông tin của class (tên class, super class, class modifier, các method, các field …)
* ****Constructor:**** Xử lý các hàm khởi tạo của class
* ****Field:**** Xử lý các field của class (tên, modifier của field, lấy giá trị, thiết lập giá trị cho object…)
* ****Method:**** Xử lý các method của class (liệt kê các method, thực thi các method …)

Ta có thể lấy đối tượng Class thông qua một object với method getClass() hoặc thông qua tên package + tên class với method Class.forName

Để lấy danh cách các hàm khởi tạo, các field, các method ta có thể dùng method getConstructors(), getFields() hoặc getMethods(). Tuy nhiên các method không cho phép lấy các constructor, method, field có modifier là private do đó mình sẽ dùng getDeclaredConstructors(), getDeclaredFields(), getDeclaredMethods().

Method getModifier() trả về modifier của class, method, field nhưng ở dạng số dó đó mình sẽ viết thêm method getModifierName để hiển thị nó dưới dạng String.

Để lấy giá trị hoặc truyền giá trị cho các field của đối tượng bạn có thể dùng method field.set() / field.get()

Nguồn stackjava.com

**Thư viện gson**

[](https://gpcoder.com/3251-huong-dan-su-dung-thu-vien-gson/java-gson/)

****Gson**** là một thư viện Java có thể được sử dụng để ****chuyển đổi các đối tượng Java thành chuỗi JSON****. Nó cũng có thể được sử dụng để ****chuyển đổi một chuỗi JSON thành một đối tượng Java**** tương ứng. Gson có thể làm việc với các đối tượng Java tùy ý bao gồm các đối tượng đã tồn tại từ trước mà bạn không có mã nguồn của nó.

Có một vài thư viện nguồn mở khác có thể chuyển các đối tượng Java sang JSON. Tuy nhiên, hầu hết nó yêu cầu bạn đặt các ****Annotation**** trong các lớp của bạn; hoặc một vài thứ bạn không thể làm nếu bạn không có quyền truy cập vào mã nguồn. Hầu hết cũng không hỗ trợ đầy đủ việc sử dụng ****Generics**** Java.

Mục tiêu Gson:

Cung cấp các phương thức ****toJson()**** và ****fromJson()**** đơn giản để chuyển các đối tượng Java sang JSON và ngược lại.

* Cho phép các ****đối tượng không thể thay đổi**** có sẵn được chuyển đổi sang/ từ JSON.
* Hỗ trợ rộng rãi của Java ****Generics****.
* Cho phép ****tùy chỉnh**** cho các đối tượng để chuyển đổi sang/ từ JSON.
* Hỗ trợ các ****đối tượng phức tạp**** (có phân cấp thừa kế nhiều cấp và sử dụng rộng rãi các kiểu dữ liệu Generics).

Hướng dẫn sử dụng

Các phương thức sử dụng:

* toJson() : phương thức dùng để chuyển Java Object sang chuỗi Json (quá trình này được gọi là Serialization). Phương thức này có 1 đối số là đối tượng cần chuyển sang chuỗi Json.
* fromJson() : phương thức dùng để chuyển chuỗi Json sang Java Object (quá trình này được gọi là Deserialization). Phương thức này có 2 đối số, đối số đầu tiên là chuỗi json, đối số thứ hai là kiểu dữ liệu Java Object ứng với chuỗi json.

Mã hóa âm thanh

Theo Itu ( tổ chức viễn thông quốc tế ) thì để mã hóa giọng nói cần lấy mẫu ở tần số 8000hz biểu diễn mẫu bằng 16bit .Điều này có liên quan đến việc tần số giọng nói của con người thông thường dưới khoảng 4000hz , Theo lý thuyết Nyquist–Shannon thì cần lấy mẫu ở tần số 8000hz để khôi phục

Tổng kết lại trong 1s ta mất 16x8000 =128kbit/s tương đương với 16kB/s chưa kể các phần tiêu đề gói tin với tốc độ mạng hiện nay có thể thoải nói chuyện nhiều người phía người dùng tuy nhiên phía server sẽ gặp vấn đề về băng thông . Nên là cần có các giải pháp nén thông tin đẻ tăng khả năng phục vụ cho người dùng

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Audio codecs | Tốc độ(KBit) | Độ phức tạp | Chất lượng | Độ trễ |
| G.711 PCM | 64 | Thấp | Rất tốt | Cực thấp |
| G.726 ADPCM | 40, 32, 24 | Thấp | Tốt (40k), Tồi(16k) | Rất thấp |
| G.729 CS-ACELP | 8 | Cao | Tốt | Thấp |
| G.729 A CA-ACELP | 8 | Vừa phải | Khá tốt | Thấp |
| G.723 MP-MLQ | 6.4, 5.3 | Vừa phải | Tốt(6.4k), Tồi(5.3k) | Cao |
| G.723.1 MP-MLQ | 6.4, 5.3 | Vừa phải | Tốt(6.4k), Tồi(5.3k) | Cao |
| G.728 LD-CELP | 16 | Rất cao | Tốt | Thấp |

Sau thời gian tìm hiểu thì em quyết định chọn chuẩn g729 vì các lý do sau

Vẫn còn được áp dụng rộng rãi, chất lương âm thanh đầu ra vẫn nghe rõ tiếng người nói , có kém hơn g711 nhưng không đáng kể

Độ trễ thấp

Hệ số nén cao , gấp 8 lần g711

Có thư viện mã nguồn mở miễn phí

### **Mạng dạng lưới (Mesh Topology)**

### Mesh Topology hay còn gọi là mạng dạng lưới. Sản phẩm có cấu trúc dạng lưới được ứng dụng phổ biến trong các mạng nắm giữ vai trò quan trọng và không thể bị ngừng hoạt động. Điển hình như hệ thống mạng của nhà máy điện nguyên tử hoặc hệ thống mạng an ninh, quốc phòng.Đối với mạng dạng lưới, mỗi một thiết bị máy tính sẽ được kết nối với tất cả cả các máy tính còn lại. Đó cũng là cấu trúc quen thuộc của mạng Internet.



**Xử lý jitter**

Jitter và độ trễ là các đặc tính được quy cho luồng trong lớp ứng dụng. Các jitter và độ trễ được sử dụng làm số liệu để đo hiệu suất của mạng. Sự khác biệt chính giữa jitter và độ trễ nằm trong định nghĩa của chúng trong đó độ trễ không là gì ngoài độ trễ thông qua mạng trong khi jitter là sự thay đổi về độ trễ.

Sự gia tăng độ trễ và jitter có ảnh hưởng xấu đến hiệu suất mạng, do đó, điều cần thiết là phải theo dõi nó định kỳ. Sự gia tăng độ trễ và jitter này xảy ra khi tốc độ của hai thiết bị không khớp nhau; ùn tắc khiến bộ đệm tràn, vỡ giao thông.

### **Biểu đồ so sánh**

|  |  |  |
| --- | --- | --- |
| **Cơ sở để so sánh** | **Jitter** | **Độ trễ** |
| Căn bản | Sự khác biệt về độ trễ giữa hai gói liên tiếp. | Trì hoãn qua mạng. |
| Nguyên nhân | Tắc nghẽn trong mạng. | Độ trễ lan truyền, tuần tự hóa, giao thức dữ liệu, chuyển mạch, định tuyến, đệm các gói. |
| Phòng ngừa | Sử dụng dấu thời gian. | Nhiều kết nối với internet. |

### **Định nghĩa của Jitter**

****Jitter**** là sự khác biệt giữa độ trễ của các gói IP. Nói cách khác, khi độ trễ của độ trễ của phương sai qua mạng, nó gây ra hiện tượng jitter. Có thể giải thích bằng một ví dụ, giả sử bốn gói được gửi tại các thời điểm 0, 1, 2, 3 và nhận được ở 10, 11, 12, 13, độ trễ giữa các gói giống nhau trong tất cả các gói là 10 đơn vị thời gian. Trong trường hợp khác, nếu các gói này đến 11, 13, 11 và 18, thì độ trễ được tạo là 11, 12, 9, 15 sẽ khác với trường hợp trên.

Hình thức trì hoãn đầu tiên sẽ không ảnh hưởng đến các ứng dụng như âm thanh và video, bởi vì tất cả các gói đều có cùng độ trễ. Tuy nhiên, trong trường hợp thứ hai, độ trễ khác nhau cho gói không được chấp nhận và nó cũng dẫn đến sự xuất hiện của các gói không theo thứ tự. Jitter cao biểu thị rằng sự khác biệt giữa các độ trễ là rất lớn trong khi jitter thấp có nghĩa là sự biến đổi đó nhỏ.

### **Định nghĩa độ trễ**

Độ ****trễ**** là thời gian mà gói dữ liệu yêu cầu để đến đích từ nguồn. Trong các điều khoản kết nối mạng, thời gian dành cho việc xử lý yêu cầu truy cập mạng do người dùng tạo và nhận được phản hồi của yêu cầu cho người dùng. Nhìn rộng ra, độ trễ là thời gian trôi qua giữa quá trình thực hiện hai sự kiện.

Độ trễ đơn giản là thời gian cần thiết để xử lý các thông báo ở cả kết thúc nguồn và đích và độ trễ được tạo trong mạng. Có hai cách để đo độ trễ mạng, cách thứ nhất được gọi là độ trễ một chiều trong đó thời gian trôi qua trong nguồn gửi gói và nhận đích, chỉ được đo. Trong khi ở loại khác, độ trễ một chiều từ nút A đến nút B được tổng hợp với độ trễ một chiều từ nút B trở lại nút A và được gọi là chuyến đi khứ hồi.

## **Sự khác biệt chính giữa Jitter và độ trễ**

1. Độ ****trễ**** là thời gian mà gói dữ liệu yêu cầu để đến đích từ nguồn . Ngược lại, jitter nó biến thể của độ trễ được tạo ra bởi việc truyền gói.
2. Sự tắc nghẽn trong mạng có thể gây ra jitter trong khi độ trễ có thể được tạo ra thông qua độ trễ lan truyền, chuyển đổi, định tuyến và bộ đệm.
3. Các jitter có thể được ngăn chặn bằng cách sử dụng dấu thời gian. Ngược lại, độ trễ có thể được giảm bằng cách sử dụng nhiều kết nối với internet.

Nguồn Sự khác biệt giữa Jitter và độ trễ

**Ảnh hưởng đến chất lượng thoại**

Đê khảo sát 1 môi trường truyến tin trong thực tế em đã xây dụng 1 mô hình như sau

Nguồn phát

* Thiết bị thu âm
* Hệ thống gửi các gói tin đi với thời gian giữa mỗi gói tin là 10ms
* Bộ tạo trễ truyền tin

Nguồn thu

* Chương trình tạo ra âm thanh trong với máy tính dử dụng java

Do dùng trong mạng lan là có thể coi môi trường lý tưởng ,việc nghẽn mạng , mất gói tin là không xảy ra

Với trường hợp thời gian truyền tin cố định

Đặc điểm : khoảng cách mỗi lần nhận gói tin ở bên nhận là giống nhau

Dù thây đổi tham số tạo trễ trong nhiểu lần thử nghiệm âm thanh thu được không bị ảnh hưởng

Với thời gian truyền tin không cố định (jitter)

Đặc điểm : khoảng cách mỗi lần nhận gói tin ở bên nhận là khác nhau

Âm thanh được tạo ra sẽ có hiện tượng méo nếu thời gian giữa 2 lần nhận gói tin > 10ms

Còn với các giai đoạn nhận gói tin <10 ms thì không bị ảnh hưởng

Kết luận thời gian truyền gói tin ảnh hưởng rất lớn đến các chất lượng truyền tin do đó cần có biện pháp giảm thiểu ảnh hưởng của jitter

**Giảm ảnh hưởng của jitter**

Jitter là điều không thể tránh khỏi khi truyền tin trên mạng ,do vậy chúng ta chỉ có thể tìm ra các biện pháp để giảm ảnh hưởng của jitter để việc cảm nhận nghe của người dùng

Phương pháp em sử dụng trong đề tài này là tạo bộ đệm chứa các câu thoại

Số lượng chứa mặc định là 10 . có nghĩa là sau ít nhất 10 gói tin thì mới bắt đầu tạo âm thanh và cứ liên tục nhận gói tin và phát thanh , nếu lưu 10 gói tin thì độ trễ so với việc phát liên tục 0.1s có thể chấp nhận

* Việc dùng bộ đêm này có lợi ích là trong 10ms mà chưa có gói tin gửi đến thì chương trình vẫn có dử liệu để phát trong khi chờ gói mới tin đến

Ngoài ra ta có thể thêm các tính năng như điểu chỉnh tốc độ phát gói tin nhanh chậm tùy thuộc vào tốc độ phát gói tin bên nguồn gửi

**Xử lý vấn đề đồng bộ**

Một vấn đề khác trong truyền tin là đồng bộ giữa các nguồn phát . Trong đó là viêc lấy xung nhịp chuẩn cho các nguồn phát .Do thiết bị của từng hãng khác nhau thì có xung nhịp khác nhau

Lấy ví dụ cho sự ảnh hưởng là 1 hệ thống cứ 1s hiền thị 1 bản tin . với 1 nguồn phát 9s tạo ra 10 bản tin . Vậy sau 90s sẽ có 10 bản tin bị thừa ra . Hoặc ngược lại 10s tạo ra 9 bản tin . việc khác nhau giữa tốc độ 2 bên nguồn thu và phát có thể ảnh hưởng chất lượng mong muốn

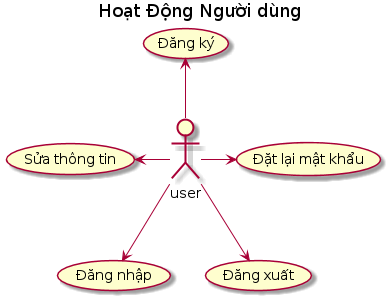
Theo như tài liệu tham khảo thì cần có thiết bị làm đồng hồ chuẩn . Dù có thể dùng máy server làm đồng hồ chuẩn nhưng với lượng kiến thức hiện tại em cũng không xác định được là máy của mình có thể đáp ứng đủ yêu cầu hay không . Thêm nữa là xây dựng các cơ chế để đồng bộ hóa nhịp chuẩn cũng là công việc không hề đơn giản

Tuy nói các dòng máy có sự về khoảng thời gian Lấy mẫu nhưng thật sự mà nói nó rất nhỏ.Nếu giảm đi yêu cầu về chất lượng đầu ra. ta có thể thay đổi tốc độ hiển thị để cho phù hợp đầu vào qua đó ảnh hưởng của sự sai khác do tốc độ lấy mẫu được giảm thiểu

**Các thiết kế use-case**

**Tổng quan vể các chức năng**

**Chức năng chính đối với người dùng**



Đây là hoạt động cơ bản của người dùng thao tác với hế thống gồm các hành động tương ứng với các ca sử dụng

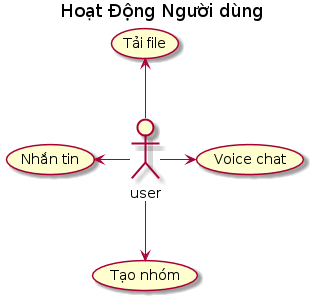
Đăng kí : tạo tài khoản mới

Đăng nhập : Kết nối vào hệ thông

Đăng xuất : thoát tài khoản

Sửa thông tin : sửa thông tin tài khoản

Đặt lại mật khẩu : đổi lại mật khẩu trong trường hợp người dùng



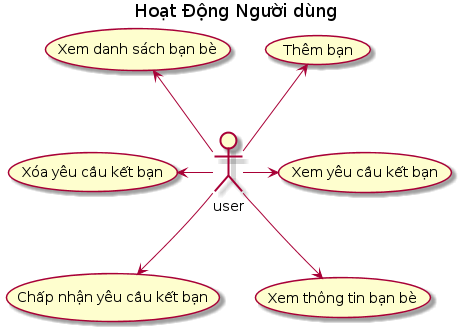
Đây là các hoạt động liên quan đến việc nhắn tin , gồm các hoạt động

Nhắn tin : gửi tin nhắn lên nhóm chat

Tạo nhóm : tạo 1 nhóm chat mới

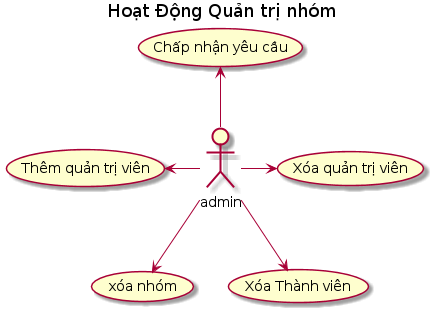
Voicechat : mở 1 cuộc hội thoại bằng giọng nói

Tải tệp : tải tệp lên



Chức năng đại điện cho hoạt động thêm bạn cho người dùng bao gồm các hoạt động hỗ trợ người dùng thêm bạn 1 cách thuận tiện nhất

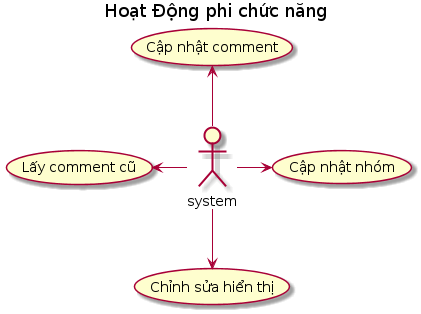
**Các usecase dành cho quản trị viên nhóm**



Chấp nhận yêu cầu vào nhóm

* Thêm 1 số thành viên thành quản trị viên
* Bỏ chức vụ quản trị viên khỏi 1 số thành viên
* Xóa nhóm
* Xóa 1 thành viên

Use case phi chức năng



Cập nhật comment : sau 1 thời gian cập nhật các cmt mới từ các người dùng khác

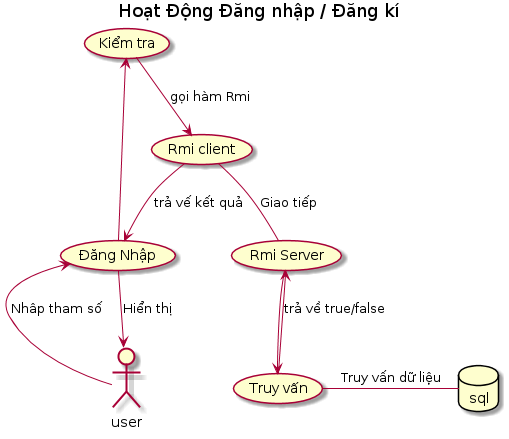
Sắp sếp hiển thi theo trinh tự nhóm nào có commet mới nhất thì xếp lên đầu

Cập nhật danh sách các nhóm đang tham gia

Chỉnh sửa ngôn ngữ hiển thị ( chưa thực hiện )

Lấy các cmt cũ trong nhóm

**1 Chi tiết các chức năng**



Chi tiết hoạt động : người dùng mở phần mềm lên để thực hiện việc đăng nhập / nếu chưa có tài khoản sẽ chuyển sang chức năng đăng ký , tiếp theo đó dữ liêu nhập vào sẽ được kiểm tra cho dúng các yêu cầu cần thiết , nếu sai phải nhập lại

Sau đó sử dụng phương thức login/register rồi thông qua rmi đưa lên server

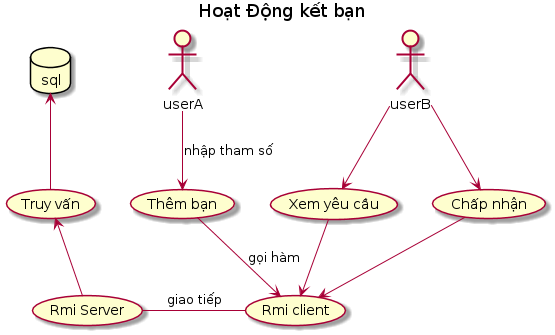
Server sẽ truy vấn và kiểm tra với dự liệu trong hệ quan trị cơ sở dữ liệu rồi trả về kết qủa

Tổng kết các chức năng trong interface Rmi cung cấp

Đăng nhập

Đăng ký

**2 Chức năng kết bạn**



Người dùng A thông quan giao diện nhập tên người muốn thêm thông qua giao diện , sau đó dùng rmi gửi tham số lên server . Server sẽ lưu lại thông tin tin này vào cơ sở dữ liệu

Người dùng b chon chức năng thêm bạn . Phần mềm dùng rmi lấy ra danh sách yêu cầu kết bạn . sau đó người dùng b chọn người muốn kết bạn hoặc xóa yêu cầu kết bạn , sau đó dùng rmi gửi lên server để xử lý

Tổng kết :các chức năng trong interface Rmi cung cấp

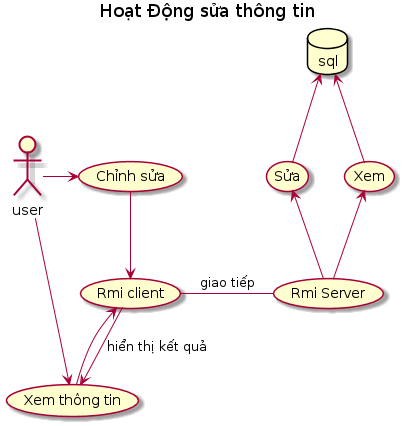
Thêm bạn

Chấp nhận yêu cầu kết bạn

Xóa yêu cầu

Xem yêu cầu kết bạn

**3 Chức năng chỉnh sửa thông tin**



Người dùng chọn chức năng xem thông tin hệ thống lấy thông tin từ server để hiển thị thông qua rmi

Từ giao diện người dùng có thể chỉnh sửa các thông tin mong muốn rồi từ rmi cập nhật lên server

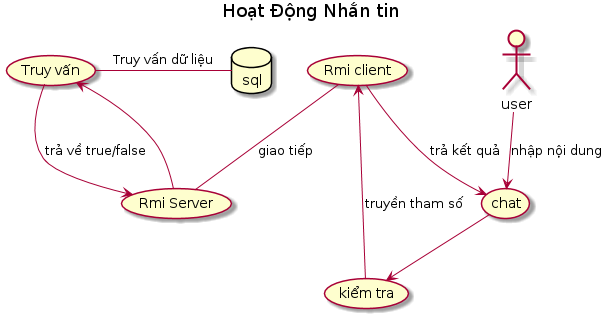
Tổng kết :các chức năng trong interface Rmi cung cấp

Xem thông tin

Sửa tên hiển thị

Sửa email

**4 Chức năng nhắn tin**



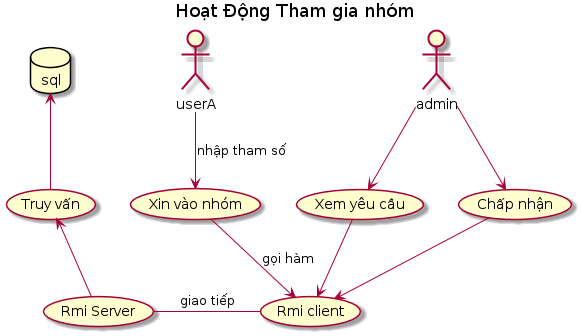
Người dùng nhập tin nhắn vào giao diện của phần mềm sau đó nhấn enter để gửi đi . Hệ thống sẽ kiểm tra nội dung tin nhắn, loại bỏ các kí tự thừa rồi gửi lên server thông qua rmi

Tại server , tin nhắn sẽ được gán nhãn thời gian để thuận tiên cho việc phân biệt tin nhắn cũ và mới , sau đó sẽ lưu vào database

Tổng kết :các chức năng trong interface Rmi cung cấp

Nhắn tin

**5 Tham gia nhóm**



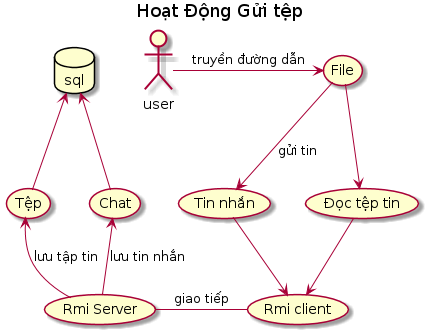
Về cách thức hoạt động cũng không khác gì khi thêm 1 người bạn

Lưu ý ở đây là trong 1 nhóm chỉ có quản trị nhóm mới có quyền xem các yêu cầu và thêm thành viên vào nhóm

Tổng kết :các chức năng trong interface Rmi cung cấp

Yêu cầu vào nhóm

**6 Hoạt động gửi tệp trong nhóm**



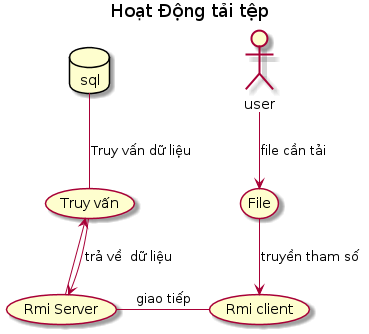
Người dùng truyền đường dẫn đến tệp vào phần mềm , nếu trong trường hợp người dùng gửi tệp lên nhóm chat ,phần mềm sẽ đọc nội dung của file đó chuyển tành mảng byte ,sau đó sẽ gửi tên tệp lên như 1 tin nhắn đồng thời tải dữ liệu của file đó lên server , Trong trường hợp tệp rỗng thì chỉ gửi tin nhắn hoặc không gửi gì cả

Trên server hệ thống sẽ sinh ra id gắn với từng file tải lên rồi đưa lên csdl, thông qua đó đưa ta có thể truy vấn file đó khi cần

Tổng kết :các chức năng trong interface Rmi cung cấp

Upfile

**7 Hoạt động tải file**



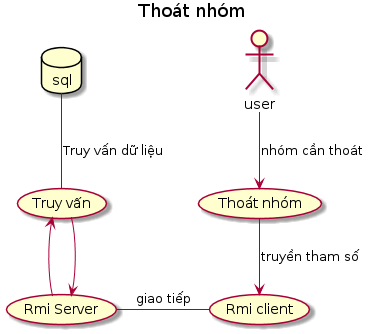
Mô tả hoạt động : người dùng thông qua phần mềm để chọn file cần tải sau đó hiển thị đường dẫn để người dùng chọn nơi lưu trữ tệp tin

Tiếp đó thông qua rmi để lấy dữ liệu dưỡi dạng mảng có kiểu là byte , sau đó phần mềm sẽ ghi ra bộ nhớ file đó

Tổng kết :các chức năng trong interface Rmi cung cấp

Tải tệp

**8 Chức năng thoát nhóm**

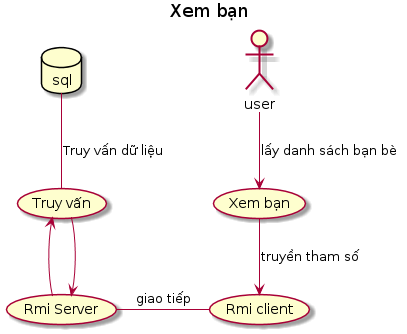


Người dùng nhập vào id nhóm cần thoát , rmi sẽ gửi yêu cầu đó lên server . Sau đố server sẽ xóa người dùng khỏi danh sách các thành viên nhóm , nếu quá trình không thành công , server sẽ gửi thông báo lỗi lại cho người dùng thông qua biến trả và của phương thức rmi

Tổng kết :các chức năng trong interface Rmi cung cấp

Thoát nhóm

**9 Xem danh sách bạn bè**

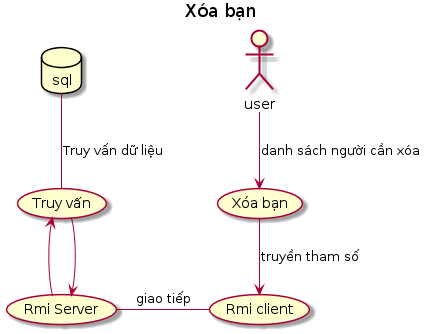


Người dùng kích hoạt chức năng này trên phần mềm . rmi sẽ chuyển thông điệp đến server . server lấy thông tin trong cơ sơ dữ liệu rồi chuyển thành đối tượng rồi trả về

Tổng kết :các chức năng trong interface Rmi cung cấp

Xem danh sách bạn bè

**10 Xóa bạn bè**



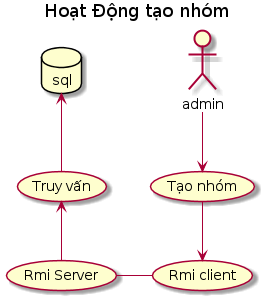
Chức năng này đi kèm với chức năng xem danh sách bạn bè , khi người dùng chọn lựa từ danh sách bạn bè. phần mềm sẽ lưu lại lại người được vào 1 danh sách , sau đó gửi lên server .

Trên server sẽ lần lượt xóa các trường tương ứng

Tổng kết :các chức năng trong interface Rmi cung cấp

Xóa bạn

**11 tạo nhóm**

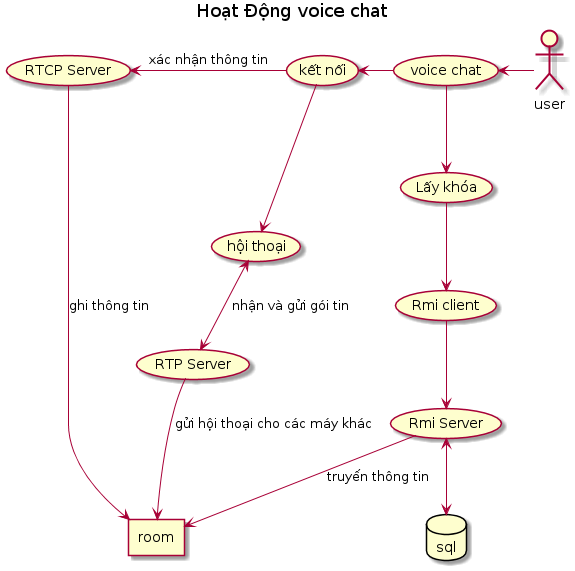


Khi người dùng chọn chức năng này từ giao diện , người dùng sẽ nhập tên nhóm vào phần mềm để gửi lên thông qua rmi

Tổng kết :các chức năng trong interface Rmi cung cấp

Tạo nhóm

**12 trò chuyện bằng giọng nói**

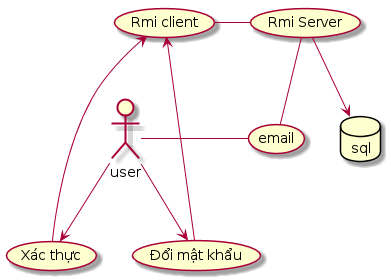


Chi tiết sẽ nói ở phần sau

Tổng kết :các chức năng trong interface Rmi cung cấp

Lấy khóa hội thoại

**13 chức năng đăt lại mật khẩu**



Người dùng sẽ yêu cầu tính năng này khi quên pass . Server sẽ gửi mã xác nhận lên email người dùng . người dùng gửi lại mã này và mật khẩu mới lên để server tiến hành kiểm tra . Nếu đúng thì thay đổi lại mật khẩu và dừng phục vụ các ca đăng nhập trước

Tổng kết :các chức năng trong interface Rmi cung cấp

Yêu cầu đổi mật khẩu

Xác thực đổi

**14 đăng suất**

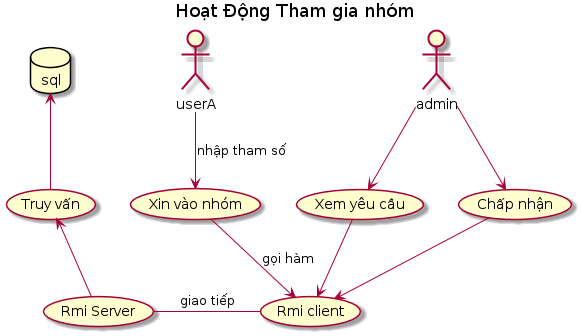
Khi gọi chúc năng này hệ thống sẽ xóa bỏ mã token tương ứng gửi lên , nghĩa là mọi hoạt động liên quan đến mã token này đều không phục vụ nữa

Tổng kết :các chức năng trong interface Rmi cung cấp

Đăng xuất

**Các chức năng phục vụ cho quản trị viên nhóm**

**1 phê duyệt vào nhóm**



Giống như việc thêm bạn ,người quản trị cần xem những người yêu cầu vào nhóm để quyết định cho phép ai được tham gia nhóm , sau đó chọn lựa thêm hoặc xóa những yêu cầu đó

Chức năng xem yêu cầu thông qua rmi sẽ lấy thông tin từ server để hiển thị danh sách các những người yêu cầu

Sau đó thông qua quyết định của quản trị viên cho phép hay là xóa các yêu cầu đó rồi thông qua rmi để thực hiện

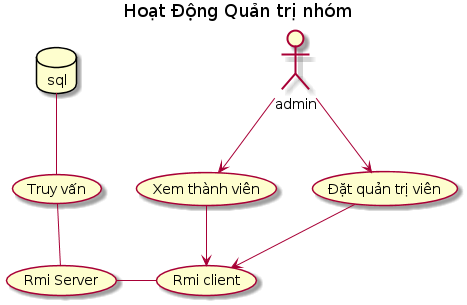
Tổng kết :các chức năng trong interface Rmi cung cấp

Xem yêu cầu vào nhóm

Chấp nhận yêu cầu vào nhóm

Xóa yêu cầu vào nhóm

**Thêm quản trị viên**



Về cách thức thì cũng tương tự việc chấp nhận yêu cầu vào nhóm

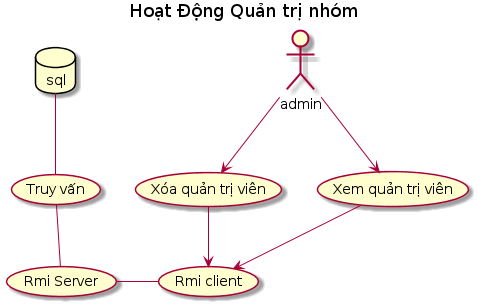
Chức năng :Xem thành viên tức là lấy danh sách các thành viên không có quyền quản trị viên để hiển thị

Tổng kết :các chức năng trong interface Rmi cung cấp

Xem thành viên bình thường

Đặt quản trị viên

**Xóa quản trị viên**



Về cách thức thì cũng tương tự chức năng trên

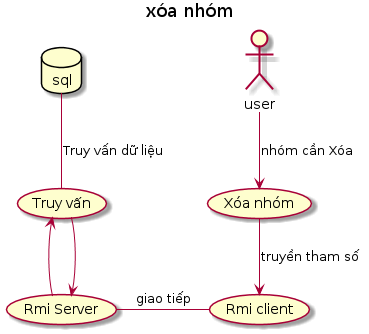
Xem quản trị viên tức là lấy danh sách các thành viên có quyền quản trị viên để hiển thị

Tổng kết :các chức năng trong interface Rmi cung cấp

Xem quản trị viên

Xóa quản trị viên

**Xóa nhóm**



Khi admin yêu cầu chức năng này phần mềm sẽ hiện ra 1 khung để người dùng xác nhận rồi mới gọi các phương thức trong rmi

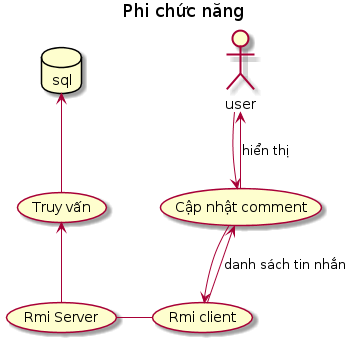
Nếu hoạt động này ko thành công sẽ có thông báo từ sever trả về

Tổng kết :các chức năng trong interface Rmi cung cấp

Xóa các thành viên

**Các phi chức năng**

**Cập nhât tin nhắn mới**



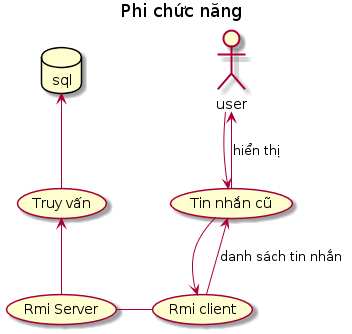
Sau 1 khoảng thời gian phần mềm sẽ phải tự động tải các tin nhắn mới để hiển thị cho người dùng xem

Thông qua rmi hệ thống lấy danh sách các tin nhắn mới và hiển thị cho người dùng

Tổng kết :các chức năng trong interface Rmi cung cấp

Lấy tin nhắn mới

**Lấy tin nhắn cũ**



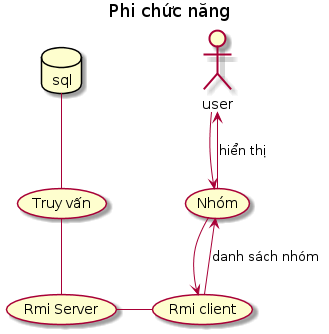
Khi người dùng kéo con lăn lên trên cùng hệ thống sẽ tự động kích hoạt chức năng lấy các tin nhắn cũ

Thông qua rmi hệ thống lấy danh sách các tin nhắn cũ và hiển thị cho người dùng vào cuối danh sách

Tổng kết :các chức năng trong interface Rmi cung cấp

Lấy tin nhắn cũ

**Lấy danh sách nhóm**



Sau 1 khoảng thời gian phần mềm sẽ phải tự động cập nhật lại danh sách nhóm để hiển thị cho người

Tổng kết :các chức năng trong interface Rmi cung cấp

Lấy danh sách nhóm

**Thiết kế**

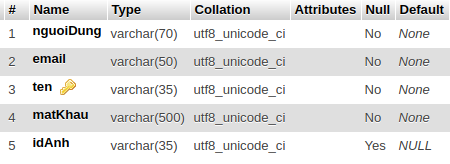
**Thiết kế hệ cơ dử liệu phục vụ việc chat nhóm**

Cơ bản

Để có thể tạo ra 1 nhóm chat thì theo các khảo sát của em trên các sản phẩm tương tự như messenger , Discord thì cần tối thiểu 3 bảng để lưu trự hoạt động

* 1 bảng lưu trữ thông tin người dùng
* 1 bảng lưu trữ thông tin thành viên trong nhóm
* 1 bảng lưu các tin nhắn trong nhóm

Bảng Thông tin thành viên



Giải thích các trường

Người dùng : tên hiển thị

Email : để gửi 1 số thông tin , dùng cho việc như khôi phục tài khoản

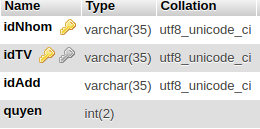
Id : chứa mã của file ảnh

Tên : tên đăng nhập - khóa chính

MatKhau : Mật khẩu đã được băm bằng Bcrypt

Với các nguồn em tham khảo thì còn có thêm nhiều trường nữa phục vụ cho nhiều mục đích khác nhau tuy nhiên với phạm vị của đề tài này thì em chỉ sử dụng các thông tin này

Bảng thông tin nhóm



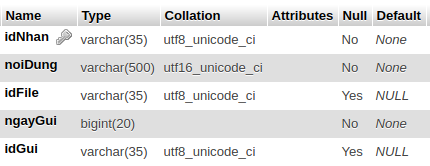
Quyen :thông tin phân quyền cho thành viên 1 là admin , 0 là thành viên

idAdd : lưu thông tin người thêm vào nhóm , dùng khi muốn truy vết thông tin

idTv : thành viên nhóm

idNhom : đinh danh nhóm

Bảng tin nhắn



idNhan : id của nhóm , có khóa ngoài với idNhom ( trong bảng “ nhóm” sẽ giới thiệu sau )

noiDung: nội dung tin nhăn

IdFile : id file gửi dùng để download nếu thành viên gửi lên 1 file

Ngaygui : thời gian gửi file tương ứng với kiểu long trong java

Giải thích thiết kế

1 hệ thống nhắn tin hiện nay thì đều cần có bước đăng nhập thế nên cần 1 bảng chứa thông tin tài khoản

Do các thông tin như Tên hiển thị ,tên đăng nhâp , ảnh đại diện đều được công khai nên việc mã hóa nó là không cần thiết , còn mật khẩu phải băm với hàm băm để bảo vệ phòng trường hợp bị lộ thông tin

Mỗi tin nhắn gửi lên sẽ tương đương 1 bản ghi trong bảng tinNhan , tuy nhiên nếu chỉ hỗ trợ văn bản thì sẽ không đủ cho nhu cầu sử dụng .Do đó em chia tin nhắn ra 2 loại văn bản và đa phương tiện

Về phần văn bản sễ hiển thị nội dung trong khung chat .Còn về đa phương tiện em sẽ thiết kế cho chương trình hiển thị bức ảnh thu nhỏ nếu tệp đó là ảnh . Còn lại , chương trình sẽ hiển thị tên tệp thay thế . Có hỗ trợ việc dowload

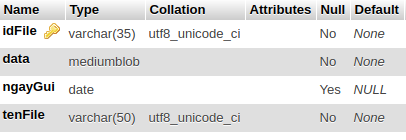
Tuy nhiên để giảm việc sử lý thì phần mềm chỉ cho phép gửi các tệp nhỏ hơn 16MB

Việc bảng thành viên nhóm có thêm trường “quyen” nhằm mục đích kiểm soát việc thêm xóa thành viên trong nhóm . Điều này được rút ra từ phần mềm zoom , học sinh có thể thoải mái thêm thành viên gây ra sự phá hoại

**Các bảng phục vụ cho chức năng nâng cao**

Mục đích các bảng này được tạo ra là nhằm để phục vụ thêm các chức năng gửi tệp , yêu cầu nhóm , ....Các bảng này giúp mở rộng thêm chức năng cho hệ thống , làm tăng độ an toàn cũng như dễ sử dụng

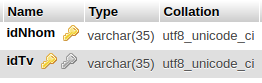
Bảng Tệp Tin



Phục vụ cho mục đích lưu trữ các tệp

* Với khóa chính là idFile
* Data:là trường chứa dữ liệu
* NgayGui : thời gian gửi
* tenFile : tên của file

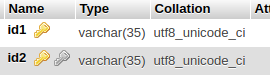
Bảng Yêu Cầu



Chứa các thông tin người yêu cầu vào nhóm

Nhóm được yêu cầu

Bảng bạn bè



Có thiết kế giống bảng yêu cầu tuy nhiên vì nục đích sử dụng khác nhau mà không gộp chung lại

Trong bảng tin nhắn cũng có trường id file tuy nhiên em không gộp chung với bảng tin nhắn là do tăng khả mở rộng cho ứng dụng . Do có thêm việc thêm ảnh cá nhân cho người dùng cũng yêu cầu lưu trữ tệp tin ,..

Mục đích của bảng yêu cầu là là chứa các yêu cầu vào nhóm . Bảng này được tạo ra nhằm kiểm soát việc thêm thành viên khi có mã mời . Tuy rằng có đôi chút bất tiện với các nhóm có tính mở cao , nhưng xét về độ an toàn . cũng như nói chuyện thoải mái với những người mình mong muốn thì việc này là cần thiết

Do mục tiêu là hướng tới các cuộc hội thoại nhóm, việc kết bạn chỉ nhằm mục đích tạo ra 1 nhóm và dễ dàng hơn trong việc tạo nhóm và thêm người . Tuy nhiên 2 người cũng có thể nói chuyện riêng bằng cách tạo thêm 1 nhóm mới , việc này sẽ được hỗ trợ đầy đủ

**Thiết kế quá trình hội thọai**

**Bảo mật**

Để đảm bảo tính bí mật cho cuộc hội thoại thì tính năng bảo mật là không thể thiếu do đó cần phải có cơ chế bảo mật ở đây là mã hóa gói tin

Tuy nhiên 1 số vấn đề đặt ra ở đây là độ dài của bản tin sau khi mã hóa và thời gian giải mã , Chăng hạn nếu dùng thuật toán mã hóa Des với độ dài khóa tiêu chuẩn là 64 bit = 8 byte , 10 byte đầu vào thì thu được 16 byte đầu ra.Tức là tăng thêm 60 % . Các thuật toán mã hóa khóa công khai cũng có vấn đề tương tự chưa kể là thời gian tính toán lâu hơn

Các thuật toán mã hóa đối xứng truyền thông như khóa đối xứng , hoán vị ưu điểm là nhỏ gọn , tuy nhiên lại có độ an toàn thấp dễ dàng dùng máy tính để tìm ra khóa

Các thuật toán mã hóa móc xích (CBC) cũng khó áp dụng trong ứng dụng truyền tin thời thực do 2 nguyên nhân chủ yếu . 1 là trong quá trình truyền tin bằng UDP không thể tránh khỏi việc mất mát gói tin nên là không thể giải mã tại nguồn thu. 2 là các gói tin được gửi liên tục nện để có thể mã hóa các khối đó sẽ gặp nhiều khó khăn trong xây dựng chương trình hơn.

Nhận xét .

Lựa chọn Mã hóa mã công khai thì việc là 1 phương án tồi . Chưa kể đến chi phí tính toán giải thì việc chọn khóa đã là 1 việc hết sức phức tạp . Với cuộc hội thoại nhiều thành viên việc trao đổi từng cặp khóa cho mỗi thành viên là 1 vấn đề hết sức phức tạp do khi thêm 1 người cần phải chuyển giao lại khóa cho các thành viên thêm 1 lần nữa . Còn trường hợp các thành viên dùng chung 1 cặp khóa , tuy là giảm bớt việc chuyển giao các khóa nhưng lại nảy sinh nhiều vấn đề khác. Nếu tất cùng dùng 1 cặp khóa cố định thì việc này gần như là không có tác dụng do hiện nay việc dịch ngược code cho phép đọc mã nguồn hiện nay rất đơn giản . còn nếu tạo cho mỗi nhóm thì với việc người dùng có thể tạo , hủy nhóm hết sức thoải mái thì công việc tạo khóa liên tục như thế sẽ là 1 gánh nặng rất lớn đến hệ thống

Với 64 bit khóa thì tính năng bảo mật của thuật toán DES cũng không phải là tốt cho lắm , việc tăng khích thước khóa lên 128, 256 ... sẽ làm thời gian tính toán lâu hơn , độ dài bản mã sẽ càng to ra ảnh hưởng đến việc truyền tin của hệ thống

Thế nên em quyết định dùng mã hóa Vigenere đê thực hiện , với khích thước 10 byte sẽ có tất cả 256^10 khóa . tức là lớn hơn 10^24 . Thêm nữa đây là kết quả sau khi âm thanh đã đi qua bộ nén theo chuẩn g729 nên việc thám mã sẽ gặp nhiều khó khăn hơn do các cách phá mã cũ sẽ khó mà áp dụng

Do đó em quyết định dùng mã hóa đối xứng với khóa thay đổi mỗi lần để tăng cường khả năng bảo mật

**Lấy khóa**

Khi người dùng yêu cầu chức năng này thì chương trình sẽ tự động kiểm tra danh sách nhóm chat đang có , Nếu không có nhóm sẽ tự động tạo ra đối tượng nhóm chat mới và tạo ra khóa nhóm .Tiếp theo đó là sinh ra chuỗi token để xác nhận người dùng

Sau đó hệ thống sẽ gửi cả khóa lẫn token về cho người dùng thông qua ssl

**Đăng kí**

Tiếp đó người dùng sẽ gửi bản tin Rtcp đăng kí lên máy chủ để xác nhận thông tin .Trong đó có chứa token đã mã hóa . Server sẽ ghi lại thông tin về đị chỉ ip của client , cổng kết nối

Sau đó server sẽ gửi về bản tin xác nhận thành công , chứa mã số nhóm , mã số thành viên

Quá trình gửi tin lên nhằm mục đích đăng kí địa chi ip cho máy tính để sau này gửi tin . Vấn đề này cũng xuất phát từ mô hình Rmi không hỗ trợ cho việc lấy địa chỉ nguồn và không thể băt người dùng mở các cổng trên router của hộ

Do thiết kế chương trình của em là ko cho phép trong 1 cuộc hội thoại có 2 thiết bị trên 1 tài khoản . thiết kế này nhàm chống việc đăng nhập nhiều tài khoản vào phá hoại do đó mới có công đoạn trả về thông tin để xác nhận tài khoản có được tham gia hội thoại hay không

Hệ thống Rtcp

Có tác dụng là để giữ kết nối cho người dùng ở chế độ im lặng do là hệ thống sẽ tự loại bỏ thành viên tham gia hội thọa nếu trong 5s người đó không gủi tin để tiết kệm tài nguyên

Hoặc là xóa người dùng khi người đó gửi gói tin kết thúc trò chuyện

Để xác định người dùng còn hoạt động hay không thì ta phải ghi lại thời gian gửi các gói tin duy trì kết nối . Phương pháp này tuy là phải lặp lại thao tác nhiều lần nhưng ưu điểm là nhanh chóng , không phải tốn băng thông khi gửi gói tin kiểm tra client

Hệ thống Rtp

Nhận và gửi tin cho các thành viên khác Hệ thống Rtp

**Thiết kế giao diện rmi**

Mục tiêu của phần phần này là thiết tham số đầu vào và kết quả trả về của các phương thức

Theo các phần trên , ta xây dựng được các danh sách các hàm quan trọng giúp hệ thống hoạt động

Các chức năng cơ bản

* Đăng nhập
* Đăng ký
* Xác thực đăng ký
* Thêm bạn
* Chấp nhận yêu cầu kết bạn
* Xóa yêu cầu
* Xem yêu cầu kết bạn
* Xem thông tin
* Sửa tên hiển thị
* Sửa email
* Nhắn tin
* Yêu cầu vào nhóm
* Thoát nhóm
* Tạo nhóm
* Upfile
* Tải tệp
* Xóa nhóm
* Xem danh sách bạn bè
* Xóa bạn
* Lấy khóa hội thoại
* Xem yêu cầu vào nhóm
* Chấp nhận yêu cầu vào nhóm
* Xóa yêu cầu vào nhóm
* Xem thành viên bình thường
* Đặt quản trị viên
* Xem quản trị viên
* Xóa quản trị viên
* Xem thành viên
* Xóa các thành viên
* Lấy tin nhắn mới
* Lấy tin nhắn cũ
* Lấy danh sách nhóm
* Đăng xuất
* Đổi mật khẩu
* Xác thực đổi mật khẩu

**Hàm đăng nhập**

Như mọi phần mềm khác , cần có thao tác đăng nhập để xử dụng Thông thường ta cần tên đăng nhập và mật khẩu để kết nối cào hệ thống

Để phân biệt từng người dùng ta có thể dùng tên đăng nhập dể phân biệt tuy nhiên lại hạn chế khả năng người dùng đăng nhập trên nhiều thiết bị . Do đó giải pháp là sử dụng 1 chuỗi là mã token ( One time password) để xác định người dùng

Đầu vào chuỗi mật khẩu , tên đăng nhập

Kết quả trả về chuỗi token

**Hàm đăng kí**

Để đăng kí 1 tài khoản mới thì cần nhập tên đăng nhập , mật khẩu , địa chỉ email , tên hiển thị , tên đăng nhập , mật khẩu phục vụ cho chức năng đăng nhập , email dùng để khôi phục tài khoản , Tên hiển thì là tên mà người dùng mong muốn hiển thị trong cuộc trò chuyện có thể thay đổi tùy ý . Do tên đăng nhập là duy nhất

Đầu vào chuỗi mật khẩu , tên đăng nhập ,địa chỉ email , tên hiển thị

Kết quả trả về: rỗng

**Hàm xác thực đăng ký**

Hàm này có chức năng xác nhận yêu cầu đăng kí của người dùng , server sẽ gửi email cho người dùng nhằm xác minh email đó còn được sử dụng hay không , người dùng cần nhập mã OTP để hoàn tất đăng ký

Đầu vào chuối otp

Đầu ra : true/false

**Thêm bạn**

Muốn thêm 1 người bạn ta cần biết chính xác tên đăng nhập của người bạn đó thì mới thêm được , tiếp đó là mã token để xác định xem ai là ngườ yêu cầu chức năng đó

Trường hợp nếu mã token là rỗng hoặc không tồn tại thì trả về giá trị false

Đầu vào chuỗi token , tên người bạn muốn thêm

Kết quả trả về true /false

**Xem yêu cầu kết bạn**

Đầu vào là mã token để xác định người dùng , không cần thêm các thông tin khác vì đây là thao tác liên quan đến bản thân người dùng

Đầu ra là danh sách tên người dùng dạng danh sách chuỗi (List<String>) có thể gửi qua rmi

Đầu vào :chuỗi token

Đâu ra: danh sách đầu ra

**Chấp nhận yêu cầu kết bạn**

Để kết bạn được thì cần danh sách những người được chấp nhận

Mã token để xác định người sử dụng

Đầu vào là danh sách những người được chấp nhận kết bạn (List<String> có thể gửi được qua rmi ) danh sách này chứa tên đăng nhập do tên đăng nhập là duy nhất dùng để xác định dữ liệu cần liên kết trong csdl

Đầu vào chuỗi token , danh sách

Đâu ra : true / false

**Xóa yêu cầu kết bạn**

Để xóa các yêu câu kết bạn thì cần danh sách những người không được chấp nhận

Mã token để xác định người sử dụng

Đầu vào chuỗi token , danh sách

Đầu ra : true / false

**Xem thông tin**

Chức năng này để hiển thị thông tin của bản thân hay thành viên ,trong nhóm

Để lấy thông tin ta cần truyền tên đăng nhập của đối tượng

Truyền thêm mã token để xác định là người nào đang dùng tránh trường hợp người không có tài khoản có thể dùng chức năng này

Trường hợp nếu mã token là rỗng hoặc không tồn tại thì trả về giá trị false

Đầu ra là đối tượng friend được giới thiệu ở phần khác , chứa các thông tin về id của ảnh đại diện , tên hiển thị , email . Được chuyển về chuỗi json

Đầu vào chuỗi token , tên người dùng

Đầu ra chuỗi json

**Sửa tên hiển thị**

Chức năng này dùng để thay đổi tên hiển thị của người dùng khi cần

Thông tin cần thay đổi là tên hiển thị nên đầu vào cần tên hiển thị , và chuỗi token để xác định đối tượng cần thay đổi

Đầu vào chuỗi token , tên

Đâu ra : kết quả đúng / sai

**Sửa email**

Chức năng này dùng để thay đổi email của người dùng khi cần

Thông tin cần thay đổi là email nên đầu vào cần email mới , và chuỗi token để xác định đối tượng cần thay đổi

Đầu vào chuỗi token , email

Đâu ra : kết quả đúng / sai

**Nhắn tin**

Được sử dụng khi nhắn tin trong 1 nhóm mà người dùng tham gia .tức là cần id nhóm để biết nhóm đang nhắn là nhóm nào , token để biêt người gửi là ai , nội dung của tin nhắn là gì

Đầu vào : token , id nhóm , nội dung tin nhắn

Đầu ra : không

**Yêu cầu vào nhóm**

Được sử dung khi người dùng yêu cầu vào 1 nhóm . do đó cần cung cấp thông tin về nhóm muốn vào , người yêu cầu là ai .Nên thông tin cần là token , id nhóm cần vào

Trường hợp token không tồn tại trong hệ thống thì không xử lý yêu cầu

Đầu vào : token , id nhóm

Đầu ra : true / false

**Thoát nhóm**

Dùng khi người dùng có yêu cầu thoát ra khỏi nhóm nào đó

Nên thông tin cần biêt là nhóm đó là nhóm nào , ai muốn thoát ra khỏi nhóm . Trường hợp token không tồn tại trong hệ thống thì không xử lý yêu cầu

Đầu vào : token , id nhóm

Đầu ra : true / false

**Tạo nhóm**

Để tạo 1 nhóm , ta cần thông tin là tên của nhóm , người tạo ra là ai , các thành viên ban đầu gồm ai ( tham khảo messenger)

Đầu vào token , tên nhóm , danh sách thành viên

Đầu ra : không

**Upfile**

Dùng khi người dùng muốn tải tệp lên server . để tải 1 tệp thì cần thông tin về tên tệp , dữ liệu của tệp , người gửi , nhóm . Do sử dụng rmi thì cần phải đọc dữ liệu của tệp rồi chuyển đổi thành kiểu dữ liệu có thể gửi được trong rmi .theo 1 số ví dụ tìm được thì kiểu dữ liệu nên dùng ở đâu là 1 mảng byte

Đầu vào : token , id nhóm , tên tệp , mảng dữ liệu

Đầu ra : true / false

**Tải tệp**

Đầu vào : token , id của tệp

Đầu ra: mảng byte

**Xóa nhóm**

Để xóa 1 nhóm thì cần phải biết nhóm đó là nhóm nào , tiếp đó là cần xác định xem người xóa là ai để kiểm tra quyền quản trị viên

Đầu vào : token , id nhóm

Đầu ra : true / false

**Xem danh sách bạn bè**

Do là xem danh sách bạn bè của bản thân nên dữ liệu đầu vào chỉ cần chuỗi token để xác định

Kết quả đầu ra cần danh sách tên của bạn bè, có thể bổ xung vài thông tin nhứ địa chỉ email , ảnh đại diện nên sẽ dùng mảng các đối tượng đã được định nghĩa sẵn . Do vướng phải vấn đề chuyển dữ liệu trong rmi nên kết quả tra về sẽ được chuyển thành chuỗi json

Đầu vào : chuỗi token

Đầu ra : chuỗi json

**Xóa bạn**

Dùng chức năng này khi muốn xóa bỏ những người bạn không mong muốn . Để thuận tiện nên sẽ hỗ trợ xóa cùng lúc nhiều người thế nên đầu vào là danh sách liên kết của tên bạn bè( trong java dùng kiều List sẽ ko phải quan tâm đến kích thước của dãy , diều này thuận tiện vì ta ko biết trước ngươi dùng muốn xóa bao nhiêu người bạn ) . Server báo lại thành công hay thất bại

Đầu vào : chuỗi token , danh sách bạn muốn xóa

Đầu ra : boolean

**Xem yêu cầu vào nhóm**

Sử dụng chức năng này khi admin muốn xét duyệt các yêu cầu do đó cần chuỗi token để xác định người dùng , nhóm mà admin đó đang quản lý

Do tên hiện thị có thể trùng nhau khi hiển thị ra , nên sẽ trả về tên đăng nhập

Đầu vào : chuỗi token , id nhóm

Đầu ra : danh sách tên

**Chấp nhận yêu cầu vào nhóm**

Dùng chức năng này khi muốn cho phép những người khác vào nhóm. Để thuận tiện nên sẽ hỗ trợ chấp nhận cùng lúc nhiều người thế nên đầu vào là danh sách liên kết của tên người( trong java dùng kiều List sẽ ko phải quan tâm đến kích thước của dãy , diều này thuận tiện vì ta ko biết trước ngươi dùng muốn xóa bao nhiêu người bạn ) . Server báo lại thành công hay thất bại

Đầu vào : chuỗi token , danh sách chấp nhận

Đầu ra : boolean

**Xóa yêu cầu vào nhóm**

Dùng chức năng này khi muốn xóa bỏ yêu cầu không phù hợp điều kiện . Để thuận tiện nên sẽ hỗ trợ xóa cùng lúc nhiều người thế nên đầu vào là danh sách liên kết của tên người( trong java dùng kiều List sẽ ko phải quan tâm đến kích thước của dãy , diều này thuận tiện vì ta ko biết trước ngươi dùng muốn xóa bao nhiêu người bạn ) . Server báo lại thành công hay thất bại

Đầu vào : chuỗi token , danh sách muốn xóa , id nhóm

Đầu ra : boolean

**Xem thành viên bình thường**

Chức năng này được dùng khi muốn thêm 1 số người khác thành quản trị viên . Các thành viên lựa chọn ở đây là thành viên không có quyền quản trị viên

Kết quả trả về là

Đầu vào : chuỗi token ,id nhóm

Đầu ra : danh sách thành viên

**Đặt quản trị viên**

Chức năng này được dùng khi muốn thêm 1 số người khác thành quản trị viên , để thuận tiên thì sẽ cung cấp khả năng cho phép nhiều người cùng lúc , nên đầu vào là 1 danh sách Server báo lại thành công hay thất bại

Đầu vào : chuỗi token , danh sách quản trj viên mới , id nhóm

Đầu ra : boolean

**Xem quản trị viên**

Chức năng này được dùng khi muốn xóa bỏ quyền quản trị viên với 1 số thành viên .Nó dùng để hiển thị các quản trị viên trong nhóm. Các thành viên lựa chọn ở đây là các quản trị viên

Kết quả trả về là

Đầu vào : chuỗi token ,id nhóm

Đầu ra : danh sách quản trị viên

**Xóa quản trị viên**

Chức năng này được dùng khi muốn xóa 1 số quản trị viên trong nhóm, để thuận tiên thì sẽ cung cấp khả năng cho phép nhiều người cùng lúc , nên đầu vào là 1 danh sách Server báo lại thành công hay thất bại

Đầu vào : chuỗi token , danh sách quản trị viên muốn xóa , id nhóm

Đầu ra : boolean

**Xem các thành viên**

Áp dụng khi muốn xem thông tin các thành viên trong nhóm , Thông tin các thành viên có thể lưu trữ dưới dạng đối tượng “Friend” , Tuy nhiên , để tiện cho việc chuyển giao dữ liệu sẽ được chuyển về chuỗi json

Đầu vào : id nhóm , token

Đầu ra : chuỗi json

**Xóa các thành viên**

Đầu vào , đầu ra là tương tự các chức năng xóa khác , chỉ khác ở đây danh sách đầu vào là danh sách thành viên muốn xóa

Chỉ áp dụng cho quản trị viên

Đầu vào : chuỗi token , danh sách thành viên muốn xóa , id nhóm

Đầu ra : boolean

**Lấy tin nhắn mới**

Chức năng này lấy tin nhắn có thời gian gửi muộn hơn so với tin nhắn muộn nhất được lưu trong nhóm đang chat trên client

Nên thông tin cần biết là thời gian của tin nhắn muộn nhất , nhóm đang chat , mã token để xác định người dùng có trong hệ thống hay không

Đầu ra là các danh sách các đối tượng “comment “ được chuyển sang chuỗi Json

Thời gian gửi được biểu diện dưới kiểu long để thuận tiện so sanh , truyền dữ liệu

Đầu vào : chuỗi id nhóm , token , thơi gian gửi

Đầu ra chuỗi json

**Lấy tin nhắn cũ**

Chức năng này lấy tin nhắn có thời gian gửi sớm hơn so với tin nhắn sớm nhất được lưu trong nhóm đang chat

Nên cần biết thời gian của tin nhắn muộn nhất , nhóm đang chat , mã token để xác định người dùng có trong hệ thống hay không

Đầu ra là các danh sách các đối tượng “comment “ được chuyển sang chuỗi Json

Thời gian gửi được biểu diện dưới kiểu long để thuận tiện so sanh , truyền dữ liệu

Đầu vào : chuỗi id nhóm , token , thơi gian gửi

Đầu ra chuỗi json

**Lấy danh sách nhóm**

Lấy danh sách các nhóm mà người dùng tham gia

Thông tin 1 nhóm gồm id nhóm , tên nhóm được lưu trong đối tượng nhóm , Các nhóm được lưu thành danh sách và được chuyển thành chuỗi gson

Đầu vào : mã token

Đầu ra : chuỗi json

**Lấy khóa hội thoại**

Đâu là chức năng để lấy thông tin dùng để xác nhận khi tham gia 1 cuộc hội thoại nhóm , Gòm 1 chuỗi là khóa của nhóm , 1 trường là dùng để xác nhận thành viên .Không thể dùng lại chuỗi token vì token nếu dùng token thì phải kiểm tra toàn bộ các token khác mỗi khi có người yêu cầu chức năng này , Nếu tách ra thì việc kiểm tra chi ở phàm vị quy mô nhóm , nhỏ hơn rất nhiều so với kiểm tra toàn bộ

Đầu vào id nhóm , chuôi token

Đầu ra : khóa bảo mật , chuỗ định danh

**Đăng xuất**

Thông tin cần thiết để đăng suất là trường token đang sử dụng . vì thông qua token này ta có thể xác định ai đang sử dụng nó

Đầu vào, chuỗi token

Đầu ra : không

**Đổi password**

Ta cần thông tin là ai là người muốn đổi mật khẩu , mã token chỉ được tạo ra nếu người dùng đăng nhập thành công , tuy nhiên với các trường hợp không đăng nhập được thì không tạo mã token

Đầu vào, chuỗi tên đăng nhâp

Đầu ra : không

**Xác thực đổi password**

Đầu vào, mã xác thực , mật khẩu mới

Đầu ra : true/false

**Xây dựng server**

Trong kiến trúc Rmi ,Server được xây dựng ra thì tối thiểu phải đắp ứng được các hàm trong rmi ,thì hệ thống mới hoạt động được Nên trong phần này em xin trình bày cách xấy dựng từng hàm phía server và các chức năng khác

Ngọai trừ hàm đăng nhập và đăng kí thì các chức năng khác của server đều phải kiểm tra mã token đưa vào có tồn tại trong hệ thống hay không , Việc này nhằm tránh việc giả mạo tài khoản khác để phá hoại

**Đăng nhập**

Để kiểm tra tài khoản có tồn tại trong hệ thống hay không thì ta phải lấy thông tin trong bảng “thongtin” để kiểm tra tên đăng nhập và mật khẩu .lưu ý là password là thông tin đã qua mã hóa để bảo vệ người dùng

**Đăng ký**

Hệ thống ghi lại thông tin tạm thời vào bộ nhớ ,sau đó gửi email chứa mã otp đến email đăng ký của người dùng

**Hàm xác thực đăng ký**

Hệ thống sẽ kiểm tra mã opt nhận được , nếu hợp lệ thì ghi thông tin vào database với thông tin đã lưu trước đó và mật khẩu được mã hóa bởi hàm băm

**Thêm bạn**

Yêu cầu thêm bạn thì cần thêm dữ liệu vào bảng yêu cầu với người yêu cầu kết bạn lưu ở cột idtv , người được yêu cầu lưu tại cột idNhom

**Chấp nhận yêu cầu kết bạn**

Khi chấp nhận yêu cầu kết bạn thị hệ thống sẽ xóa thông tin tương ứng trong bảng yêu cầu với idnhom là người chấp nhận , idTv là người yêu cầu

Tiếp đó sẽ thông thông tin vào cột bạn bè với id1 là người chấp nhận yêu cầu , id2 là người yêu cầu

**Xóa yêu cầu**

Khi xóa yêu cầu kết bạn thị hệ thống sẽ xóa thông tin tương ứng trong bảng yêu cầu với idnhom là người được yêu cầu , id thành viên là người yêu cầu

**Xem yêu cầu kết bạn**

Khi muốn xem hết tất cả yêu cầu kết bạn với người dùng , hệ thống chỉ cần lấy thông tin tương ứng với bảng yêu cầu với điều kiện idNhom bằng tên đăng nhập người dùng

**Xem thông tin**

Khi muốn xem thông tin của người tương ứng với tên nhập vào , ta cần dữ liệu tên hiển thị , email của người dùng . Các thông tin này được lấy trong bảng “thongtin” với điều kiện tên người dùng là tên nhập vào

**Sửa tên hiển thị**

Chức năng này nhằm sửa lại thông tin người dùng mong muốn , cần cập nhật dữ liệu bảng “thongtin” với với nguoiDung bằng tên hiển mới và điều kiện ten là tên đăng nhâp lấy từ token

**Sửa email**

Chức năng này nhằm sửa lại thông tin người dùng mong muốn , cần cập nhật dữ liệu bảng “thongtin” với với email bằng email mới và điều kiện ten là tên đăng nhâp lấy từ token

**Đăng xuất**

Xóa mã token tương ứng trong hệ thống

**Đổi password**

Lấy ra email trong csdl , tạo mã và gửi đến email đã lấy

**Xác thực đổi password**

Kiểm tra mã và cập nhật mật khẩu lên Database

**Nhắn tin**

Khi thêm 1 tin nhắn vào database thì dữ liệu sẽ lưu tại bảng tinNhan với idNhom là nhóm đang chat , idgui là người gửi tin nhắn , noiDung là nội dung tin nhắn , ngayGui là thời điểm khi chức năng này được sử dụng

**Yêu cầu vào nhóm**

Cần thêm bạn thì cần thêm dữ liệu vào bảng yeuCau với người yêu cầu lưu ở cột idtv , nhom được yêu cầu lưu tại cột idNhom

**Thoát nhóm**

Khi một thành viên muốn thoát khỏi nhóm cần xóa dữ liệu tương ứng ở bảng tvNhom với điều kiện idTv là người muốn thoát nhóm , idNhom là nhóm muốn thoát

**Tạo nhóm**

Khi tạo 1 nhóm , hệ thống sẽ tự đông sinh ra một chuỗi dùng để định danh một nhóm gọi là id, sau đó sẽ lưu lại thông tin vào bảng nhóm , Sau đó sẽ lần lượt thêm dữ liệu của từng thành viên vaò bảng tvNhom ,với id đã tạo , thành viên là người trong danh sách đưa lên với quyen bằng 0 và thêm người tạo với quyền bằng 1

**Upfile**

hệ thống sẽ tạo ra 1 chuỗi dùng để định danh cho file được tải lên , sau đó thao tác tương tự như khi nhắn tin với nội dung là tên file , idfile là id đã tạo , tiếp theo đó là lưu file vào bảng tepTin

**Tải tệp**

Dựa vào id file để lấy dữ liệu tương ứng trong bảng tệp tin , chuyển nó thành mang byte rồi trả về kết quả

**Xóa nhóm**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên , thế nên phải kiểm tra thông tin về quyen trong bảng thành viên nhóm

Xóa thông tin trong các bảng tinNhan ,tvNhom, nhom với idNhom bằng chuỗi nhóm trong đầu vào . Thứ tự xóa lần lượt như trên do có liên quan đến khóa ngoài trong bảng

**Xem danh sách bạn bè**

Thông tin chi tiết về từng mối quan hệ bạn bề lưu trong bảng banBe , do cách lưu bạn bè phụ thuộc vào người nào là người thêm bạn nên tên người dùng có thể xuất hiện trong cả hai cột id1, id2

Khi tìm kiếm dùng phép hợp (Union) để lấy đầy đủ kết quả

**Xóa bạn**

Xóa bạn là xóa thông tin trong bạn bè với điều kiện id1 = người xóa , id2 = người bị xóa và ngược lại . Thực hiện lần lượt với tùng người trong danh sách đầu vào

**Xem yêu cầu vào nhóm**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên , thế nên phải kiểm tra thông tin về quyen trong bảng thành viên nhóm

Yêu cầu vào nhóm thì dữ liệu lưu vào bảng yeuCau với người yêu cầu lưu ở cột idtv , nhom được yêu cầu lưu tại cột idNhom . nên muốn lấy ra các yêu cầu vào nhóm thì cần tìm với idNhom = nhóm đang quản lý

**Chấp nhận yêu cầu vào nhóm**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên , thế nên phải kiểm tra thông tin về quyen trong bảng thành viên nhóm

Khi chấp nhận yêu cầu thì hệ thống sẽ xóa thông tin tương ứng trong bảng yêu cầu với idnhom là nhóm chấp nhận , id thành viên là người yêu cầu

Sau đó thêm dữ liệu về thành viên vào bảng tvNhom với idAdd là tên người phê duyệt , quyen bằng 0 đại diện cho thành viên bình thường

**Xóa yêu cầu vào nhóm**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên , hệ thống sẽ xóa bổ các yêu cầu tương ứng trong bảng với idNhom từ dữ liệu đầu vào , idTv là những người trong danh sách

**Xem thành viên bình thường**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên , hệ thống sẽ lấy nhứng thành viên của nhóm trong bảng tvNhom với điều kiện quyen = 0

**Đặt quản trị viên**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên ,hệ thống sẽ đặt lại giá trị quyen của những thành viên của nhóm có trong danh sách đầu vào trong bảng tvNhom là 1

**Xem quản trị viên**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên , hệ thống sẽ lấy những thành viên của nhóm trong bảng tvNhom với điều kiện quyen =1

**Xóa quản trị viên**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên ,hệ thống sẽ đặt lại giá trị quyen của những thành viên của nhóm có trong danh sách đầu vào trong bảng tvNhom là 0

**Xem thành viên**

Chức năng dùng với mọi thành viên trong nhóm, hệ thống sẽ lấy tất cả thành viên của nhóm trong bảng tvNhom

**Xóa các thành viên**

Chức năng này chỉ dùng với với thành viên có quyền quản trị viên ,hệ thống sẽ xóa thông tin của những thành viên của nhóm có trong danh sách đầu vào trên bảng tvNhom

**Lấy tin nhắn mới**

Hệ thống sẽ lấy ra thông tin của những tin nhắn trong bảng tinNhan có thời gian gửi muộn hơn thời gian đầu vào

**Lấy tin nhắn cũ**

Hệ thống sẽ lấy ra thông tin của những tin nhắn trong bảng tinNhan có thời gian gửi sớm hơn thời gian đầu vào

**Lấy danh sách nhóm**

Để lấy danh sách nhóm ta cần lấy thông tin trong bản tvNhom liên kết với bảng nhóm là được , Tuy nhiên để thuận tiện cho người thì ta nên sắp sếp theo thứ tự có thời gian gửi tin nhắn gần nhất thì lên đầu , xa ở cuối .Do đó cần thêm thông tin từ bảng tinNhan . lưu ý là do có 1 số nhóm mới tạo chưa có tin nhắn nên khi liên kết với bảng tinNhan cần dùng liên kết trái (hoặc phải) để đảm bảo đầy đủ dữ liệu

**Thiết kế clients/server trong hội thoại**

Nếu thiết kế với mô hình thông thường thì nó yêu cầu tốc độ xử lý của server cực kì lớn do đây là ứng dụng thời gian thực . Việc tăng kích thời gian phát cũng là 1 cách để giảm áp lực cho server tuy nhiên , nó tăng thêm độ trễ cho người sử dụng , việc kích thước gói tin càng lớn thì khi bị mất sẽ gây ra hiện tượng méo tiếng rõ ràng hơn khi sử nghe

Do đó em đề xuất mô hình giống mô hình mạng hình sao trong đó mỗi thành viên sẽ gửi các thông tin cho các thành viên tiếp theo

Việc khó khăn của cách thiết kế này là mỗi thành viên đều có ip động nên rất khó khăn .Mà dù có thể xác định được địa chỉ ip thì cũng không thể nào kết nối vào nếu chưa Nat port

Ngược lại nếu gửi 1 gói tin ra ngoài thì router sẽ tự động định tuyến cho ta để kết nối internet , qua đó ta có thể lấy được cổng và địa chỉ của router đã nat .Tuy nhiên điểm yếu của phương pháp này là sau 1 thời gian ko sử dụng thì trong bảng dữ liệu ko dùng trong nat sẽ bị xóa đi do đó cần phải gửi các gói tin 1 cách định kỳ để giữ kết nối

Nhược điểm của thiết kế này là client sẽ phải gửi đi nhiều gói tin hơn thông thường tuy nhiên băng thông sử dụng trên server sẽ giảm mạnh , đồng thời gói tin không cần qua server nên giảm được độ trễ

Hoạt động

Các client gửi các gói tin đăng kí lên server , server sẽ lưu lại các thông tin về địa chỉ ip và cổng kết nối

Client sẽ lấy thông tin từ người tham gia nhóm chat để gửi thông tin

Sau 1 khoảng thời gian sẽ gửi lại 1 gói tin lên server để cập nhật lại thông tin về kết nối

**Truyền dữ liệu thời gian thực**

## **So sánh UDP và TCP**

****GIống nhau****: đều là các giao thức mạng TCP/IP, có chức năng kết nối các máy lại với nhau và có thể gửi dữ liệu cho nhau….

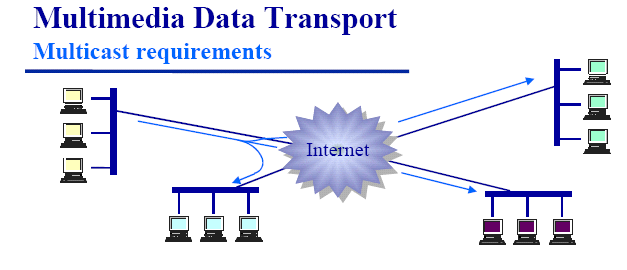
****Khác nhau****

| **TCP** | **UDP** |
| --- | --- |
| Hướng kết nối | Hướng không kết nối |
| Độ tin cậy cao | Độ tin cậy thấp |
| Gửi dữ liệu dạng luồng byte | Gửi đi Datagram |
| Không cho phép mất gói tin | Cho phép mất gói tin |
| Đảm bảo việc truyền dữ liệu | Không đảm bảo việc truyền dữ liệu |
| Có sắp xếp thứ tự các gói tin | Không sắp xếp thứ tự các gói tin |
| Tốc độ truyền thấp hơn UDP | Tốc độ truyền cao |

Lựa chon giao thức cho ứng dụng thời gian thực

Trong những ứng dụng truyền thông đa phương tiện, **yêu cầu đảm bảo khắt khe về thời gian thực** (không cho phép có thời gian trễ lớn, jitter). Việc các gói tin đến không liên tục, đều đặn làm cho chất lượng hình ảnh, hoặc âm thanh thu được thấp. Rất có thể gây ra vấp hình, méo tiếng. Để đáp ứng được những yêu cầu này, một giao thức thời gian thực cần có các yếu tố:

* **Hộ trợ việc định tuyến muticast:** Với các ứng dụng tryền thông đa phương tiện đòi hỏi thời gian thực, có sự phân phối giống dữ liệu từ một nguồn tới nhiều đầu cuối nhận dữ liệu thì việc hỗ trợ multicast là rất cần thiết.Đây là một yêu cầu rất quan trọng. Khi đó, sẽ tồn tại 1 nguồn phát và rất nhiều nguồn thu, một máy chủ xuất luồng dữ liệu thời gian thực đến rất nhiều máy khách. Nếu ta sử dụng truyền unicast, tải trọng tác động lên máy chủ rất lớn. Trong khi đó, nếu mạng có hỗ trợ truyền multicast, ta chỉ việc xuất một luồng duy nhất từ máy chủ tới một địa chỉ multicast. Sau đó tại các nút mạng, luồng dữ liệu sẽ được nhân lên và chuyển tiếp tới những địa chỉ đích.



*Sử dụng Multicast trong truyền dữ liệu đa phương tiện.*

* **Chấp nhận một số gói tin bị lỗi**: Không thể đợi để truyền lại các gói, đoạn, gam dữ liệu bị thất lạc. Việc truyền lại các dữ liệu bị thất lạc hoặc bị lỗi sẽ chiếm khá nhiều thời gian. Nó sẽ làm tăng lượng tải trên đường truyền đồng thời kéo dài thời gian trễ của các gói tin.
* **Cần kết hợp với một thông số về thời gian (nhãn thời gian) kèm theo gói dữ liệu**: Với các tín hiệu thời gian thực, đặc biệt là tín hiệu video, việc khôi phục đồng bộ tại phía thu là rất quan trọng, do đó đòi hỏi nhãn thời gian kèm theo để phục vụ cho việc tái tạo lại dữ liệu tại nơi nhận. Đặc biệt, khi tín hiệu video được mã hoá theo từng khung hình, mỗi khung hình được vận chuyển trong nhiều gói RTP. Khi đó nhãn thời gian sẽ giúp ta phân định từng nhóm gói tin tương ứng với một hình một cách dễ dàng.

Trong những giao thức ở lớp vận chuyển thì giao thức nào có thể đáp ứng được yêu cầu trên:

* TCP:

Đây là một giao thức kiểu có liên kết (Connection – Oriented), tức là phải có giai đoạn thiết lập liên kết giữa một cặp thực thể TCP trước khi truyền dữ liệu. Trong khi truyền dữ liệu giao thức TCP phải đảm bảo các cơ chế xác nhận việc gởi dữ liệu, đảm bảo xắp xếp đúng thứ tự các gói tin tại bên nhận, phát lại các gói tin bị lỗi hoặc thất lạc. Do việc phải đảm bảo những cơ chế này gây lên thời gian trễ lớn, nên giao thức TCP không thể dùng được trong những ứng dụng thời gian thực.

Ngoài ra với tính chất vốn có của mình, TCP là giao thức được sử dụng để truyền dữ liệu theo kiểu điểm tới điểm, hay nói cách khác TCP chỉ được dùng cho truyền unicast, không thể sử dụng cho truyền multicast.

Với những đặc điểm trên, TCP không nên được sử dụng trong việc truyền dữ liệu mang tính thời gian thực.

* UDP:

Đây là một giao thức kiểu không kết nối, được sử dụng trong một số yêu cầu ứng dụng thay thế cho TCP. Tương tự như giao thức IP, UDP không thực hiện các giai đoạn thiết lập và huỷ bỏ liên kết, không có các cơ chế báo nhận như trong TCP. UDP cung cấp các dịch vụ giao vận không đáng tin cậy. Dữ liệu có thể bị mất, bị lỗi hay bị truyền luẩn quẩn trên mạng mà không hề có thông báo lỗi đến nơi gửi hoặc nơi nhận. Do thực hiện ít chức năng hơn TCP nên UDP chạy nhanh hơn, nó thường được sử dụng trong các dịch vụ không đòi hỏi độ tin cậy cao. Ngoài ra, giao thức UDP còn có thể sử dụng cho truyền multicast.

Do vậy UDP có thể được sử dụng để truyền các dữ liệu thời gian thực.

*Trích dẫn từ “Nghiên cứu và ứng dụng Rtp”*

Sơ lược về Rtp được ghi trong phần phụ lục

Kích thước gói tin

Kích thước gói tin rtp tiêu chuẩn là 16 byte rtp header + 10 byte data trên 1 gói tin là 26byte.Tính thêm cả Udp header và ipv4 header là 66 byte . Thông tin về tiêu đề tầng 2 trong mô hình osi thì em chưa nghiên cứu được hết nhưng ước tính tổng kích thước gửi đi nhỏ hơn 100byte . Với tốc độ 1 gói/10 ms thì băng thông tiêu tốn khoảng 10Kb/s . Với hạ tầng mạng hiện nay thì tốc độ này là khá thấp

**1 số thay đổi về rtp**

Rtp là 1 chuẩn được thiết kế cho môi trường truyền dữ liệu chung tuy nhiên với đề tài thì có nhiều chức năng sẽ không dùng đến . Nếu giữ nguyên không thay đổi thì sẽ gây ra nhiểu sự lãng phí không cần thiết với đề tài này. Ngược lại nếu chỉnh sửa cấu trúc gói tin thì sẽ mất đi khả năng tương tác với các hệ thống khác . Hay có thể nói không liên kết được các hệ thống khác

Để tạo thuận lợi cho cho việc nâng cấp thì phần sử lý liên quan đến truyền nhận gói tin sẽ được thiết kế thành packet riêng đảm bảo sự thuận lợi cho việc nâng cấp sau này nếu cần thiết

Thay đổi về rctp

Mục tiêu của Rctp là phản hồi , đánh giá chất lượng mạng từ đó có các biện pháp xử lý .Tuy nhiên do đề tài này làm việc với âm thanh được mã hóa giảm thiểu tối dung lượng cần truyền . Có nghĩa là trong mọi trường hợp kích thước gói tin sẽ không thay đổi .Các yếu tố liên quan đến phản hổi chất lượng mạng ... sẽ chưa được hỗ trợ trong phên bản này

Do đó em sẽ đơn giản hóa các gói tin kiểm soát để đặt được hiệu năng cao nhất

**Gói tin join**

Gói tin này dùng để đăng kí với server rằng có người dùng muốn tham gia vào nhóm

Các trường thông tin trong nhóm

* Trường type : 16 bit chứa kiểu gói tin , có thể dùng để mở rộng sau này
* Trường length: 16bit độ dài của gói tin tính riêng id nhóm và header tiêu chuẩn
* Trường ntp : nhãn thời gian 64bit
* Trường port: cổng nhận dữ liệu máy client
* Trường group : mã id nhóm
* Trường user : chuỗi để xác nhận thành viên
* Type : 1000

**Gói tin bye**

Gói tin này được dùng khi có người muốn thoát ra cuộc hội thoại

Các trường thông tin trong nhóm

* Trường type :chứa kiểu gói tin , có thể dùng để mở rộng sau này 16 bit
* Trường length: độ dài của gói tin 16bit
* Trường ntp : nhãn thời gian 64bit
* Trường group : mã id nhóm 32 bit
* Trường user : id thành viên 32 bit
* Type : 1111

**Gói tin live**

Dùng giữ kết nối khi trong , cập nhật thông tin trong cuộc hội thoại

* Trường type :chứa kiểu gói tin , có thể dùng để mở rộng sau này 16 bit
* Trường length: độ dài của gói tin 16bit
* Trường ntp : nhãn thời gian 64bit
* Trường group : mã id nhóm 32 bit
* Trường user : id thành viên 32 bit

Type : 1001

Gói tin Rtp

Trong gói tin rtp thì trường sequense đảm nhiệm vai trò đếm gói tin . Tuy nhiên vì nguồn phát ra là đều đặn với thời gian tăng dần nên có thể dùng trường timestamp lấy 32 bit cuối để đảm nhận vai trò của trường sequense, 32 bit đủ để biểu diễn thời gian tương ứng với 50 ngày nên có thể dùng tùy ý

Hiện tại chỉ làm việc với âm thanh theo chuẩn ITU-G729 đã mã hóa an tòan với khóa được trao đổi sắn nên có thể cắt bớt các trường crsc count , padding ....

Ý nghĩa 1 số trường

* SSRC: nguồn đồng bộ tương ứng với id nhóm
* CSRC: tương ứng với mã thành viên
* TimeStamp : 32bit cuối của NTP

**Thực hiện đề tài**

**Các lớp hỗ trợ**

Trong mô hình client server thì không thể thiếu được quá trình giao tiếp giữa 2 bên .Với đa số là thông tin là các chuỗi thì việc tổ chức chúng thành các mảng sẽ thuận tiện cho việc truyền tin vì có kích thước gọn nhẹ . Tuy vậy trong quá trình thực hiện thì có nhiều lần em gặp phải sai lầm về vị trí và ý nghĩa trong các mảng đó. Hơn nữa việc tái sử dụng hoặc nâng cấp code sẽ gặp nhiều khó khăn do phải nhớ vị trí của từng thông tin . Và hơn hết lỗi do nhầm vị trí thường xuyên xảy ra làm chương trình có kết quả sai mà không có thông báo lỗi .Thế nên việc gỡ lỗi sẽ rất vất vả , chiếm rất nhiều thời gian .

Theo 1 số nghiên cứu chỉ ra rằng con người nhớ tốt hơn những cụm từ có nghĩa hơn là các con số nên em sẽ không dùng cách này

Lấy ví dụ hệ thống tên miền hâu hết là ko sử dụng số . Ví dụ ta nhớ rất dễ dàng tên miền google.com nhưng với địa chỉ ip 216.58.220.206 thì lại không

Vì lẽ đó em quyết định dùng các đối tượng để chứa dữ liệu .Sẽ trách được tình trạng nhớ sai lệch các vị trí ,Thuận tiện cho việc nâng cấp về sau .Tuy việc này sẽ kiến hệ thống tốn thêm băng thông hoạt động

**Các lớp được cài đặt**

Lớp Comment với các thuộc tính

* idNhan : mã của người nhận
* idGui: mã của người gửi
* noiDung : Nội dung tin nhắn
* idFile : mã tệp gửi đi

Lớp Group với các thuộc tính

* idNhom : mã nhóm
* tenNhom : tên

Lớp Friend

* nguoiDung : tên hiển thị của người dùng
* Ten : tên đăng nhập của người dùng
* idAnh : mã số của file ảnh đại diện
* Email : email của người dùng

Lớp client

* User : tên người dùng
* Id:đinh danh người dùng
* Ip : địa chỉ ip người dùng
* Port : công kết nối

Trong các lớp trên lớp Friend thường sử dụng với nhiều mục đích nhất , chứa thông tin danh sách bạn bè , chứa thông tin cá nhân , chứa thông tin các thành viên trong nhóm , chứa thông thông các thành viên quản trị .... Nếu không sử dụng object thì đó là cả 1 vấn đề do phải nhớ thứ tự từng cái , do mỗi chức năng lại chỉ yêu cầu 1 số thông tin nhất định

Trong các trường hợp trên các thông tin cần thiết sẽ được ghị vào đối tượng . còn lại nhận giá trị null

Ví dụ :Thay vì phải nhớ theo kiểu

Args[10] = .... // i= 10 là tên người dùng

Thì ta có thể ghi p.getTenNguoiDung() ; thuận tiện hơn trong sử dụng , không phải nhớ tùng vị trí một

**Chuyển đổi dữ liệu phía server**

Có thể tóm gọn hoạt động của server là nhận dữ liệu từ hệ quản trị cơ sở dữ liệu chuyển đổi nó sang đối tượng và gửi đi

Việc thực hiện câu truy vấn đã có trong phần khác nên không nhắc lại nữa

Kết quả trả về câu truy vấn là đối tượng ResutSet mà thông qua đó ta có thể lấy thông tin tại cột mình cần bằng cách sử dụng tên cột đó .

Để chuyển dữ liệu sang các đối tượng thì yêu cầu là tên các trường trong đối tượng phải trùng tên trùng kiểu với các bảng trong kết quả

Hạn chế về đặt tên tuy có thể loại bỏ bằng cách dùng notation trong java nhưng do em thấy không cần thiết nên không triển khai

Trong java có 2 cách thực hiện truy vấn với Database là Jdbc và hibernate .Tuy là hibernate có nhiều ưu điểm hơn so với JDBC. Nhưng do bản thân em đã làm việc nhiều với JDBC mà hơn nữa đồ án này cũng chỉ sử dụng Mysql . Nên em thấy lựa chọn là việc với JDBC là điều tốt hơn cả . tôi có thể dùng kĩ thuật Reflection để mô phỏng lại Ưu điểm lớn nhất của Hibernate là nó tự động thêm dữ liệu vào đối tượng . Chưa kể là sử dụng JDBC sẽ áp dụng Connection pool( Hồ kết nối ) đẽ dàng hơn

Thành quả thu được là lớp có khả năng chuyển đổi linh hoạt dữ liệu . Chỉ cần đảm bảo yêu cầu là cùng tên với các trường trong csdl , hơn nữa nhũng trường không có sẽ để giá trị null

Sau đó sẽ thông qua thư viện Gson để chuyển đôi dữ liệu sang chuỗi Json

**Phía client**

Chỉ cần dùng thư viện gson để chuyển thành đối tượng mong muốn

**Hạn chế còn tồn tại**

**Lọc âm thanh đầu vào**

Trong quá trình thu âm đầu vào tín hiệu chứa nhiễu môi trường, hệ thống khiến chất lượng bị ảnh hưởng , tuy nhiên do sự hạn chế về kiến thức về xử lý âm thanh nên chưa sử lý được. Em cũng tìm hiểu về 1 số bộ lọc thường được áp dụng hiện nay là Kalman,LMS ... tuy nhiên lại gặp nhiều vấn đề thời gian tính toán và cũng như áp dụng nó vào môi trường thời gian thực.

Không giống với lọc trên file âm thanh sẵn có , môi trường thời gian thực có rất nhiều hạn chế như số lượng mẫu thấp , chỉ có các mẫu ở thời điểm hiện tại , yêu cầu tính toán nhanh .Tuy rằng đã có 1 số tác giả đã nói qua về các áp dụng tuy nhiên với một người mới bắt đầu tìm hiểu như em thì nó quá trừu tượng

**Đảm bảo nguồn phát**

Điều kiện lý tưởng nhất của đề tài này là cứ là sau khoảng thời gian có T sẽ phát đi 1 gói tin âm thanh . Trong quá trình thực hiện đề tài thì em nhận ra điều này là không thể đạt được . chẳng hạn với quy trình tuần tự như sau : thu âm -> xử lý ->gửi -> thu âm ...hoặc quy trình song song thu âm -> lưu vào hàng đợi -> thu âm .Rõ ràng khi qua bước thu âm ta đã mất khoảng thời gian Ts này rồi chưa kể là xử lý sau đó nên là dù có truyền tin trong môi trường ý tưởng thì cũng không thể tránh khỏi jitter được

**Java**

Do mục tiêu là xây dụng các ứng dụng chạy trên đa nền tảng nên ngôn ngữ java là trong những lựa chọn hàng đầu do tình phổ biến của nó cũng như trong quá trình học tập trên lớp em được dạy nhiều môn về java . Tuy nhiên do là ngôn ngữ chạy trên da nền tảng nên việc thao tác với các thiết bị trên máy tính còn gặp nhiều hạn chế .

**Phụ lục**

**Các dạng cấu trúc**

**XML**

Xml là gì

XML được viết tắt bởi cụm từ Extensible Markup Language với nghĩa là ngôn ngữ đánh dấu mở rộng. Định dạng này được tổ hợp web toàn cầu [W3C](https://www.w3.org/) đề nghị tạo ra với mục đích phát triển và mở rộng các ngôn ngữ đánh dấu khác ngoài định dạng .txt, ASCII,… cách lưu trữ file XML này khá đơn giản, nhưng nó có thể mô tả được nhiều dữ liệu khác nhau. Chính vì vậy, XML tương đối hữu ích trong việc chia sẻ dữ liệu giữa các hệ thống, chương trình. Ví dụ: Định dạng file XML có thể là ngôn ngữ giao tiếp giúp trao đổi dữ liệu giữa 2 ứng dụng ASP và PHP.

Muốn vậy thì những đặc tả dữ liệu XML đều phải tuân theo một quy luật cũng như cú pháp nhất định. Bởi lẽ, hầu hết các file XML đều tuân thủ nghiêm khắc trong việc biên dịch.Tuy nhiên, tỷ lệ sai sót trong quá trình thao tác và truyền tải dữ liệu vẫn có, khoảng 5 – 7%. Mặc dù con số này không cao nhưng cũng đáng để cân nhắc trước khi sử dụng.

Mặc dù vậy,ta cũng có thể yên tâm khi mà ngôn ngữ đánh dấu siêu văn bản HTML cũng sử dụng cú pháp XML để tạo nên. Nó có các bộ phận phần tử, thuộc tính không mềm dẻo, nên chỉ có tác dụng trong việc trình bày dữ liệu trên [trình duyệt Browser](https://monamedia.co/trinh-duyet-web-la-gi/) mà thôi.

Tóm lại

XML là ngôn ngữ đánh dấu được dùng để miêu tả dữ liệu.

Các thẻ tag trong file XML chưa được xác định trước. Do đó, người dùng phải tự định nghĩa trong quá trình tạo file XML.

**Tác dụng của file XML**

Mục đích chính yếu của định dạng XML là chia sẻ tài nguyên dữ liệu giữa các platform trên các hệ thống khác nhau. Đặc biệt, các hệ thống được kết nối mạng, file XML sẽ được chia sẻ nhanh chóng hơn. Bởi vậy, XML thường được dùng vào việc trao đổi, chia sẻ giữa các hệ thống với nhau.

Khi có sự trao đổi dữ liệu giữa các hệ thống thì những dữ liệu đó sẽ được tổ chức dưới dạng XML. Ví dụ cụ thể: Nếu một hệ thống quản lý nhà hàng muốn lấy thông tin khách du lịch từ hệ thống công ty lữ hành thì giữa hệ thống cần đảm bảo thực hiện các thao tác công việc sau:

* Các hệ thống cần có sự thống nhất về cấu trúc file
* Công ty lữ hành cần có trích xuất dữ liệu từ hệ thống của mình, đóng gói dữ liệu dưới dạng file XML theo cấu trúc đã thỏa thuận trước đó.
* Hệ thống phần mềm của nhà hàng sẽ dựa trên các quy ước, tiến hành phân tích, trích xuất dữ liệu từ XML nhận từ hệ thống của công ty lữ hành để có nguồn data khách hàng theo mong muốn.

**Đặc điểm của XML**

Một số đặc điểm nổi bật file XML:

* XML chỉ sử dụng cho dữ liệu có cấu trúc cụ thể.
* XML có cấu trúc tương tự với file HTML
* XML mặc dù là một dạng file văn bản nhưng không dùng để đọc.
* Cấu trúc file XML thường rất dài
* File XML được biết đến là một module
* XML được ví như một nền tảng cho web ngữ nghĩa và RDF.
* XML được coi như một file cầu nối giữa hệ thống HTML và XHTML
* File XML được miễn phí bản quyền, có platform độc lập và được hỗ trợ rất tốt cho người dùng.

**Ưu điểm**

* XML là dữ liệu độc lập. Đây cũng là ưu điểm lớn nhất của file XML. Nó được sử dụng để mô tả dữ liệu dưới dạng text. Vì vậy, hầu hết các phần mềm hay chương trình bình thường đều có thể đọc được nó.
* File XML có thể dễ dàng đọc và phân tích các nguồn dữ liệu. Do đó, nó được sử dụng chính vào mục đích trao đổi dữ liệu giữa các chương trình, hệ thống với nhau.
* File XML được tạo một cách dễ dàng chỉ với vài thao tác đơn giản.
* File XML được sử dụng dành cho Remote Procedure Calls với mục đích phục vụ các dịch vụ trong thiết kế website.

**Nhược điểm**

Tỷ lệ sai sót trong khi sử dụng định dạng file XML lên tới 5 – 7%. Tuy nhiên đa phần các lỗi khi sử dụng file XML đều xuất phát từ người dùng cấu hình sai, vì vậy nếu bạn thật sự cẩn thận trong các thiết lập file của mình thì sai sót này có thể còn thấp hơn.

**Cấu trúc của XML**

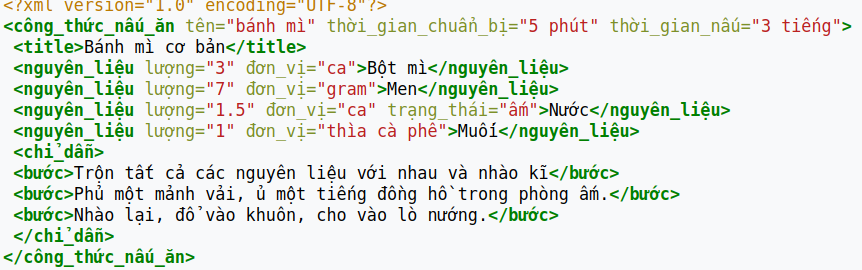
Cấu trúc của file XML được thể hiện như sau:

Trong đó:

* Content: là tên thuộc tính, thẻ mở, nội dung, tên của thẻ sẽ do người dùng tự xác định.
* Content: Nội dung chính của thẻ.
* Content: Thẻ đóng, tên, nội dung thẻ đóng và thẻ mở bắt buộc phải trùng nhau.

Nguồn <https://monamedia.co/file-xml-la-gi/>

Ví dụ XML



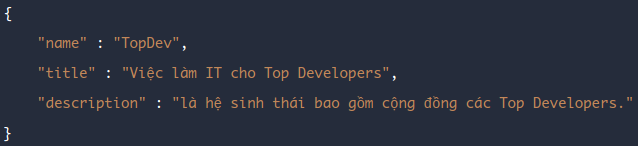
**JSON**

**JSON là gì**

JSON là viết tắt của JavaScript Object Notation, là một kiểu định dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được. JSON là một tiêu chuẩn mở để trao đổi dữ liệu trên web.

**Định nghĩa**

Định dạng JSON sử dụng các cặp key – value để dữ liệu sử dụng. Nó hỗ trợ các cấu trúc dữ liệu như đối tượng và mảng. Ví dụ một tập tin có tên topdev\_info.json với nội dung như ở dưới đây sử dụng format kiểu JSON để lưu trữ thông tin:



Ta có thể thấy cú pháp của JSON có 2 phần đó là key và value:

* Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}
* Các key, valuecủa JSON bắt buộc phải đặt trong dấu nháy kép {“}, nếu bạn đặt nó trong dấu nháy đơn thì đây không phải là một chuỗi JSON đúng chuẩn. Nếu trường hợp trong value của bạn có chứa dấu nháy kép " thì hãy dùng dấu (\) để đặt trước nó, ví dụ  \"json là gì\".
* Nếu có nhiều dữ liệu thì dùng dấu phẩy , để ngăn cách.
* Các key của JSON bạn nên đặt chữ cái không dấu hoặc số, dấu \_ và không có khoảng trắng., ký tự đầu tiên không nên đặt là số.

File json có thể được lưu với bất kỳ phần mở rộng nào, tuy nhiên thông thường thì nó được lưu dưới phần mở rộng là .json hoặc .js.

JSON ban đầu được phát triển để dành phục vụ cho ứng dụng viết bằng JavaScript. Tuy nhiên vì JSON là một định dạng dữ liệu nên nó có thể được sử dụng bởi bất cứ ngôn ngữ nào mà không bị giới hạn.

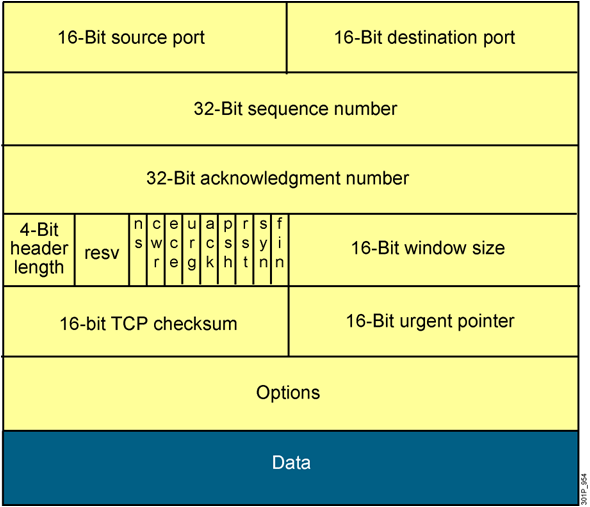
Giá trị key trong JSON có thể là chuỗi (string), số (numner), rỗng (null), mảng (array), hoặc đối tượng (object).

# **Giao thức TCP (Tranmission Control Protocol)**

* TCP là giao thức truyền tải hướng kết nối (connection-oriented), nghĩa là phải thực hiện thiết lập kết nối với đầu xa trước khi thực hiện truyền dữ liệu. Tiến trình thiết lập kết nối ở TCP được gọi là tiến trình ****bắt tay 3 bước**** (threeway handshake).
* Cung cấp cơ chế báo nhận (Acknowledgement) :Khi A gửi dữ liệu cho B, B nhận được thì gửi gói tin cho A xác nhận là đã nhận. Nếu không nhận được tin xác nhận thì A sẽ gửi cho đến khi B báo nhận thì thôi.
* Cung cấp cơ chế đánh số thứ tự gói tin (sequencing) cho các đơn vị dữ liệu được truyền, sử dụng để ráp các gói tin chính xác ở điểm nhận và loại bỏ gói tin trùng lặp.
* Có các cơ chế điều khiển luồng thích hợp (flow control) để tránh nghẽn xảy ra.
* Hỗ trợ cơ chế full-duplex ( truyền và nhận dữ liệu cùng một lúc)
* Phục hồi dữ liệu bị mất trên đường truyền ( A gửi B mà không thấy xác nhận sẽ gửi lại) .

### **Cấu trúc gói tin TCP**

Do là giao thức tin cậy nên cấu trúc gói tin của TCP rất phức tạp



* ****Source port và destination port (đều dài 16 bit)****: được sử dụng để định danh cho session của giao thức nào đó trên lớp ứng dụng đang được truyền tải trong TCP segment đang xét
* ****Sequence number (32 bit)****: dùng để đánh số thứ tự gói tin (từ số sequence nó sẽ tính ra được số byte đã được truyền).
* ****Acknowledge number (32 bit)****: : dùng để báo đã nhận được gói tin nào và mong nhận được byte mang số thứ tự nào tiếp theo.
* ****Header length (4 bit)****: cho biết toàn bộ header dài bao nhiêu tính theo đơn vị word(1 Word = 4 byte).
* ****Các bit reserverd (4 bit)****: đều được thiết lập bằng 0
* ****Các bit control (9 bit)****: các bit dùng để điều khiển cờ (flag) ACK, cờ Sequence ...
* ****Window size (16 bit)****: số lượng byte được thiết bị sẵn sàng tiếp nhận
* ****Checksum (16 bit)****: kiểm tra lỗi của toàn bộ TCP segment
* ****Urgent pointer (16 bit)****: sử dụng trong trường hợp cần ưu tiên dữ liệu
* ****Options (tối đa 32 bit)****: cho phép thêm vào TCP các tính năng khác
* ****Data****: dữ liệu của lớp trên

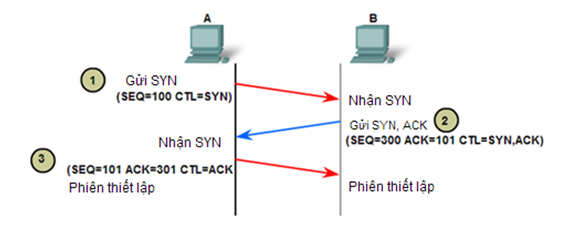
**Cách thức hoạt động**

Giả sử host A muốn truyền dữ liệu cho host B thông qua một kết nối TCP. Trước khi thực hiện truyền , host A cần phải thiết lập kết nối TCP với host B việc này được tiến hành thông qua quá trình bắt tay 3 bước như s:

* ****Bước 1****: Host A gửi cho B một gói tin có cờ SYN được bật lên, với số thứ tự được đánh là 100. Segment đầu tiên này không chứa phần dữ liệu nên không có phần data, tuy nhiên số lượng byte dữ liệu vẫn được tính là một byte cho hoạt động gửi cờ SYN.
* ****Bước 2****: Host B nhận được gói tin thì B gửi lại gói tin có cờ SYN được bật lên, kèm theo đó là cờ ACK để xác nhận.

Giả sử host B thiết lập segment có số thứ tự là 300. Segment trả lời từ Host B này cũng không có dữ liệu nhưng vẫn được tính là 1 byte cho phần data. Khi phản hồi lại host A, host B cũng cần phải chỉ rõ trong trường ACK sequence số thứ tự của byte kế tiếp mà nó muốn nhận từ host A. Do segment SYN do A gửi qua được tính là 1 byte nên B sẽ mong muốn nhận byte tiếp theo là byte thứ 101 từ A , do đó ACK sequence được đánh số là 101. (SEQ=300, ACK=101)

* ****Bước 3****: Sau khi kết nối đã được thiết lập thì A gửi lại gói tin để đáp ứng nhu cầu của B. Gói tin được đánh số SEQ = 101 để đáp ứng nhu cầu của B. ACK =301 dùng để báo là đã nhận được gói tin có SEQ = 300.

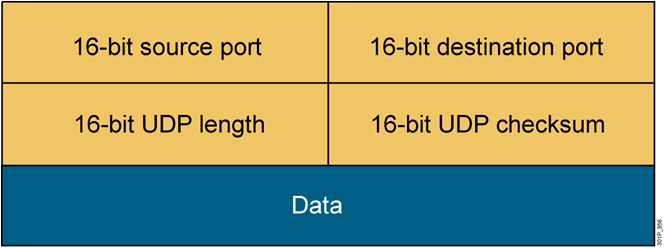


# **Giao thức UDP (User Datagram Protocol)**

* Ngược lại với giao thức TCP thì UDP là giao thức truyền tải hướng không kết nối (connectionless). Nó sẽ không thực hiện thao tác xây dựng kết nối trước khi truyền dữ liệu mà thực hiện truyền ngay lập tức khi có dữ liệu cần truyền (kiểu truyền best effort) => truyền tải rất nhanh cho dữ liệu của lớp ứng dụng.
* Không đảm bảo tính tin cậy khi truyền dữ liệu và không có cơ chế phục hồi dữ liệu ( nó không quan tâm gói tin có đến đích hay không, không biết gói tin có bị mất mát trên đường đi hay không) => dễ bị lỗi.
* Không thực hiện các biện pháp đánh số thứ tự cho các đơn vị dữ liệu được truyền…
* Nhanh và hiệu quả hơn đối với các dữ liệu có kích thước nhỏ và yêu cầu khắt khe về thời gian.
* Bản chất không trạng thái nên UDP hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.

### **Cấu trúc gói tin UDP**

Cấu trúc gói tin UDP thì đơn giản hơn rất nhiều so với TCP



* ****source port**** và ****destination port(đều 16 bit)****: cho phép định danh một session của một ứng dụng nào đó chạy trên UDP. Có thể coi port chính là địa chỉ của tâng Transport
* ****UDP length(16 bit)****: cho biết chiều dài của toàn bộ UDP datagram tổng cộng bao nhiêu byte. (16 bit thì sẽ có tổng cộng 2^16 byte = 65536 giá trị (từ 0 -> 65535 byte)).
* ****UDP checksum(16 bit)****: sử dụng [thuật toán mã vòng CRC](https://vi.wikipedia.org/wiki/Cyclic_Redundancy_Check" \t "/home/dccan/Documents\\x/_blank) để kiểm lỗi cho toàn bộ UDP datagram và chỉ kiểm tra một cách hạn chế
* ****Data****: dữ liệu tầng trên được đóng gói vào UDP datagram đang xét.

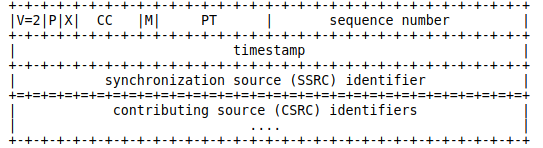
### **Cách hoạt động của UDP**

UDP hoạt động tương tự như TCP nhưng nó không cung cấp kiểm tra lỗi khi truyền gói tin.

Khi một ứng dụng sử dụng UDP, các gói tin chỉ được gửi đến người nhận. Người gửi không đợi để đảm bảo người nhận nhận được gói tin hay không, mà tiếp tục gửi các gói tiếp theo. Nếu người nhận miss mất một vài gói tin UDP thì gói tin đó coi như bị mất vì người gửi sẽ không gửi lại chúng. Các thiết bị có thể giao tiếp nhanh hơn.

**RTP**

**Cấu trúc Tiêu đề RTP**



1. **version** (V): 2 bits

Trường này dùng để xác định Verson của RTP. Hiện nay trong truyền Video và Audio RTP đang sử dụng Verson 2. Verson 1 là Verson được sử dụng đầu tiên. Còn Verson 0 chỉ là giao thức được cài đặt thêm các chức năng cho Audio.

1. **padding** (P): 1 bit

Nếu bit đệm này được đặt giá trị 1, báo rằng gói tin có chứa 1 số byte điều kiện phụ ở phần cuối (cuối phần payload). Byte cuối cùng trong các byte đệm sẽ chứa số các byte đệm đã được thêm, kể cả chính byte đó. Các byte đệm này có thể được dùng để mã hoá mật gói tin, hoặc dùng trong trường hợp đóng gói nhiều gói RTP trong 1 gói của lớp dưới.

1. **extension** (X): 1 bit

Khi bit này được gán giá trị 1 tức là sau phần tiêu đề cố định sẽ là phần tiêu đề mở rộng. Việc mở rộng thêm phần tiêu đề nhằm tăng thêm lượng thông tin cho gói RTP khi cần thiết.

1. **CSRC count** (CC): 4 bits

Phần này chứa số lượng các bộ nhận dạng CSRC sẽ được thêm vào sau phần tiêu đề cố định. Dùng để xác định số các phần tử 32 bit được chứa trong phần CSRC.

1. **Marker** (M): 1 bit

Bit này được dùng với mục đích báo hiệu. Ta có thể dùng nó để làm sự kiện báo hiệu khung trong trường hợp ta truyền các gói tin thành dòng. Bit này có thể được sử dụng hoặc không. Nếu không sử dụng ta có thể thay đổi số lượng bit trong trường payload type.

1. **Payload type** (PT): 7 bits

Trường này dùng để xác định dạng của phần tải để chọn lựa các ứng dụng phù hợp. Giá trị của phần định dạng tải này có thể cố định trong một phiên RTP nếu ta sử dụng phương pháp mã hoá tĩnh. Nó sẽ có giá trị biến đổi nếu như trong phiên RTP đó ta sử dụng cơ chế định dạng động.

Một nguồn RTP có thể thay đổi định dạng tải trong một phiên truyền, tuy nhiên ta không nên dùng 1 phiên RTP để truyền đồng thời các luồng media có định dạng khác nhau, theo khuyến cáo của RFC1890.

Về phía nhận, nếu nhận được gói RTP có định dạng tải mà nó không hiểu, gói này sẽ phải được bỏ qua.

1. **Sequence number**: 16 bits

Số thứ tự được đánh tăng dần theo số lượng các gói RTP được phát đi. Phía nhận sẽ sử dụng số thứ tự này để khôi phục lại trật tự các gói, hoặc dùng để phát hiện số lượng gói đã bị mất.

Việc khởi tạo các giá trị này nên được thực hiện theo cơ chế ngẫu nhiên, nhằm tăng tính bảo mật, bởi nó có thể được kết hợp với khoá mã. Chúng ta sẽ đề cập rõ hơn ở phần sau.

1. **Timestamp**: 32 bits

Nhãn thời gian được tính theo thời điểm lấy mẫu của byte đầu tiên trong gói RTP. Thời gian được sử dụng theo chuẩn thời gian NTP.

Nhãn thời gian phải được lấy từ đồng hồ nhịp chuẩn, có độ chính xác cao, nhằm đảm bảo cho việc kiểm tra đồng bộ và xác định độ Jitter giữa các gói tin khi đến đích. Điều này rất quan trọng, nếu ta truyền tín hiệu Video thì Jitter có thể gây ra hiện tượng vấp hình.

Tần số nhịp của nhãn thời gian phụ thuộc vào từng trường hợp cụ thể, thường do loại định dạng tải quyết định. Với ứng dụng thoại, ta lấy mẫu với tần số 8 KHz. Các gói tin sẽ được truyền đi theo từng khối sau mỗi khoảng thời gian 20ms, tương ứng với 160 mẫu, . Do vậy mỗi nhãn thời gian liên tiếp sẽ có giá trị cách nhau 160 đơn vị, không cần quan tâm gói dữ liệu trước có được nhận hay không.

Tương tự như số thứ tự, giá trị khởi tạo của nhãn thời gian cho mỗi phiên truyền là ngẫu nhiên.

1. **SSRC**: 32 bits

Trường này dùng cho việc định danh một nguồn đồng bộ. Giá trị của trường này được chọn một cách ngẫu nhiên (có kiểm tra xung đột) để tránh trường hợp trong một phiên RTP có nhiều hơn một nguồn đồng bộ sử dụng cùng một giá trị SSRC.

Khi một nguồn thay đổi địa chỉ truyền tải nguồn (*source transport address*), thì nó cũng phải chọn một giá trị SSRC mới để tránh trường hợp xung đột.

1. **CSRC**:

Danh sách này được dùng vào do bộ Mixer. Tại phía người nhận, nó được dùng để xác định rõ xem thông tin nào của nguồn nào gởi.

Danh sách này sẽ có từ 0 đến 15 phần tử. Mỗi phần tử chiếm 32 bit. Nó được dùng để xác định số nguồn tin tạo ra nội dung trong phần tải. Do danh sách chỉ chứa được tối đa 16 phần tử, nên khi có nhiều hơn 16 nguồn tới thì một số nguồn sẽ bị loại bỏ, hoặc sử dụng cơ chế gán vòng.

**Các loại gói tin Rctp**

* SR (Sender report): Bản tin phía gởi, dùng để thông tin về trạng thái truyền và nhận của phía gởi tin.
* RR (Receiver report): Bản tin người nhận, dùng để thông tin về trạng thái nhận của phía nhận tin.
* SDES (Source description items): Thông tin mô các về nguồn gởi, bao gồm cả CNAME.
* BYE: Dùng khi thành viên nào đó thoát khỏi hội thảo.
* APP (Application specific functions): Xác định các chức năng phụ thuộc vào từng ứng dụng cụ thể.

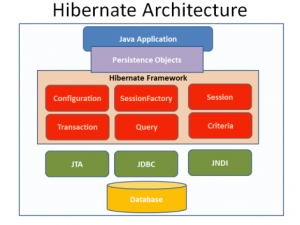
**Hibernate**

# **1. Hibernate framework là gì?**

Hibernate là 1 [ORM Framework](https://stackjava.com/uncategorized/orm-la-gi-tong-quan-ve-orm-framework.html): thực hiện mapping cơ sở dữ liệu quan hệ sang các object trong ngôn ngữ hướng đối tượng.

Hibernate là 1 Framework cho persistence layer: Thực hiện giao tiếp giữa tầng ứng dụng với tầng dữ liệu (kết nối, truy xuất, lưu trữ…)

# **2. Kiến trúc hibernate framework.**



Mỗi table trong database là một object trong Hibernate. Do đó, bạn cần có một java bean cho mỗi table trong database. Các java bean này sẽ có các getters/setters và một số ngoại lệ theo quy ước của Hibernate.

Mỗi file mapping (ánh xạ) có dạng \*\*\*.hbm.xml có nhiệm vụ đặc tả mỗi quan hệ giữa các thuộc tính của đối tượng và các trường trong bảng CSDL (cột nào ứng với thuộc tính nào của class, kiểu giá trị là gì, quan hệ giữa các bảng chuyển thành quan hệ giữa các lớp ra sao…).

File Hibernate.cfg.xml : Đây là tập tin được load đầu tiên khi khởi chạy ứng dụng Hibernate. Nó chứa các thông tin về kết nối (database kết nối là gì (MySQL, Oracle, SQL server…), username và password kết nối với database là gì, tên database được kết nối), driver kết nối…

# **3. Ưu/nhược điểm khi dùng Hibernate**

## **3.1. Ưu điểm**

* Dễ sử dụng: dễ dàng quản lí các kết nối database và dễ fix bug, cung cấp sẵn nhiều API truy vấn
* Tính độc lập: không cần quan tâm tới cơ sở dữ liệu sử dụng khi viết câu lệnh SQL.
* Tính hướng đối tượng: tập trung xử lý theo hướng đối tượng, phù hợp sử dụng trong các case CRUD(Create, Read, Update, Delete)
* Tính tin cậy: Hibernate đã được kiểm thử và khá an toàn trong các truy vấn

## **3.2. Nhược điểm**

* Không hỗ trợ các câu truy vấn phức tạp
* Một số trường hợp vẫn phải dùng native SQL do Hibernate không thể cover hết tất cả các cú pháp của các hệ quản trị cơ sử dữ liệu.
* Bị hạn chế sự can thiệp vào câu lệnh SQl do nó được tự động sinh ra.

Nguồn <https://stackjava.com/>

**Nat là gì?**

Nat ([Network Address Translation](https://vi.wikipedia.org/wiki/Bi%C3%AAn_d%E1%BB%8Bch_%C4%91%E1%BB%8Ba_ch%E1%BB%89_m%E1%BA%A1ng)) là một kỹ thuật cho phép chuyển đổi từ một địa chỉ IP này thành một địa chỉ IP khác. Thông thường, NAT được dùng phổ biến trong mạng sử dụng địa chỉ cục bộ, cần truy cập đến mạng công cộng (Internet). Vị trí thực hiện NAT là router biên kết nối giữa hai mạng.

Địa chỉ private và địa chỉ public

Địa chỉ private:

[Địa chỉ private](https://www.totolink.vn/article/71-4-dieu-ban-can-biet-ve-ip-private.html) được định nghĩa trong RFC 1918

10.0.0.0 - 10.255.255.255

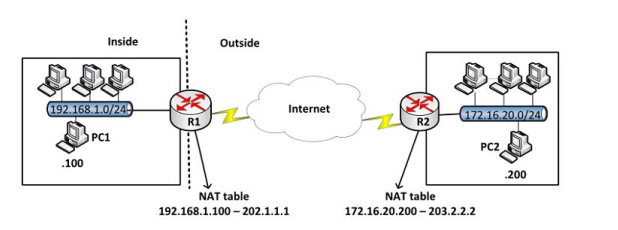
172.16.0.0 - 172.31.255.255

192.168.0.0 - 192.168.255.255

Địa chỉ public:

Là các địa chỉ còn lại. Các địa chỉ public là các địa chỉ được cung cấp bởi các tổ chức có thẩm quyền.

Một số thuật ngữ cần biết



Địa chỉ inside local: là địa chỉ IP gán cho một thiết bị ở mạng bên trong. Địa chỉ này hầu như không phải địa chỉ được cung cấp bởi NIC (Network Information Center) hay nhà cung cấp dịch vụ.

Địa chỉ inside global: là địa chỉ đã được đăng ký với NIC, dùng để thay thế một hay nhiều địa chỉ IP inside local.

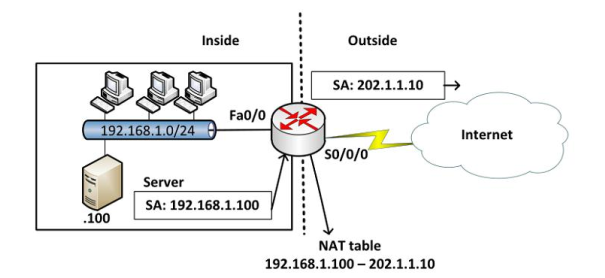
Địa chỉ outside local: là địa chỉ IP của một thiết bị bên ngoài khi nó xuất hiện bên trong mạng. Địa chỉ này không nhất thiết là địa chỉ được đăng ký, nó được lấy từ không gian địa chỉ bên trong.

Địa chỉ outside global: là địa chỉ IP gán cho một thiết bị ở mạng bên ngoài. Địa chỉ này được lấy từ địa chỉ có thể dùng để định tuyến toàn cầu từ không gian địa chỉ mạng.

**Static Nat**

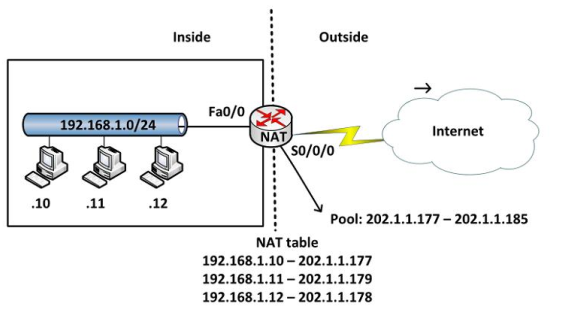
Static NAT được dùng để chuyển đổi một [địa chỉ IP](https://www.totolink.vn/article/74-ip-la-gi-cach-xac-dinh-dia-chi-ip-tren-may-tinh.html) này sang một địa chỉ khác một cách cố định, thông thường là từ một địa chỉ cục bộ sang một địa chỉ công cộng và quá trình này được cài đặt thủ công, nghĩa là địa chỉ ánh xạ và địa chỉ ánh xạ chỉ định rõ ràng tương ứng duy nhất.

Static NAT rất hữu ích trong trường hợp những thiết bị cần phải có địa chỉ cố định để có thể truy cập từ bên ngoài Internet. Những thiết bị này phổ biến là những Server như Web, Mail,...



**Dynamic NAT**

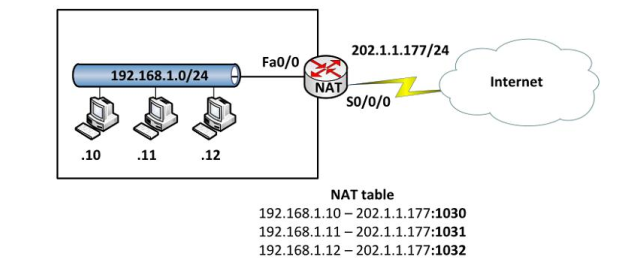
Dynamic NAT được dùng để ánh xạ một địa chỉ IP này sang một địa chỉ khác một cách tự động, thông thường là ánh xạ từ một địa chỉ cục bộ sang một địa chỉ được đăng ký. Bất kỳ một địa chỉ IP nào nằm trong dải địa chỉ IP công cộng đã được định trước đều có thể được gán một thiết bị bên trong mạng.



**NAT Overload**

Nat Overload là một dạng của Dynamic NAT, nó thực hiện ánh xạ nhiều địa chỉ IP thành một địa chỉ (many - to - one) và sử dụng các địa chỉ số cổng khác nhau để phân biệt cho từng chuyển đổi. NAT Overload còn có tên gọi là PAT (Port Address Translation).

Chỉ số cổng được mã hóa 16 bit, do đó có tới 65536 địa chỉ nội bộ có thể được chuyển đổi sang một địa chỉ công cộng.



**Tài liệu tham khảo**

Nghiên cứu và ứng dụng giao thức RTP

<http://luanvan.co/luan-van/nghien-cuu-va-ung-dung-giao-thuc-rtp-35670/>

Bộ giao thức Rtp/Rctp

<https://voer.edu.vn/m/bo-giao-thuc-rtprtcp/3d1c93bb>

RTP: A Transport Protocol for Real-Time Applications

<https://tools.ietf.org/html/rfc1889>

CÁC GIAO THỨC TRONG VOIP

<https://voer.edu.vn/c/cac-bien-phap-dam-bao-chat-luong-dich-vu/5cfab6af/bb08aafe>

Bảng mã âm thanh

<https://phuongot.wordpress.com/2011/04/19/m%E1%BB%99t-s%E1%BB%91-chu%E1%BA%A9n-ma-hoagi%E1%BA%A3i-ma-codecs-tin-hi%E1%BB%87u-audio/>

Sự khác biệt giữa Jitter và độ trễ

<https://vi.gadget-info.com/difference-between-jitter>

Java reflection

<https://stackjava.com/java/java-reflection-la-gi-huong-dan-java-reflection-api.html>

Hirbernate

<https://stackjava.com/hibernate/series-hibernate-phan-1-gioi-thieu-ve-hibernate-framework.html>

G729

<https://github.com/deruelle>