About rmi

RMI - Remote Method Invocation is a technique of installing distributed objects in Java. RMI is part of the J2SDK suite and is a library function that supports remote method calls and returns values for distributed computing applications. To use rmi, the Java programming language must be used on both the calling and the calling methods.

To solve some problems in communication between Client / Server. RMI does not call directly but through intermediaries. This class exists on both the client and server side. The class on the Client machine is called Stub, the class on the Server machine is called Skel (Skeletion).

**Rmi characteristics**

RMI is Java's distributed object model, RMI makes it easier to communicate between distributed objects in the internet environment.

RMI is a high-level API built on Socket programming.

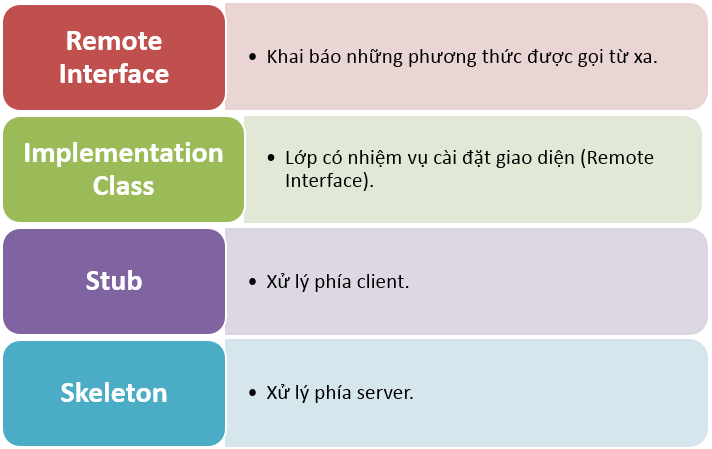
RMI not only allows us to transfer data between objects on different computer systems, but also invokes methods in remote objects (Remote Object).

Data transmission between different machines is handled transparently by the Java virtual machine.

Similar to the Client / Server model, RMI maintains the concepts of Client and Server, but RMI's approach is more flexible than a Client / Server model.

One of the most important advantages of RMI is that it provides a callbacks mechanism, which allows the server to invoke methods on the client.

Rmi architecture



Remote inteface: An interface that declares all the functions that the Server provides. The client can call this function as a regular call

Note: All functions in this interface throw an exception or RemoteException

Impements class: The class is based on the Remote inteface, performing the specified tasks. Usually located on the server

Stub and skeleton: Classes on the server and client for communication. automatically generated by java

**In and out with Rmi**

Another point to note is that when using rmi, the data transmitted is not changed or can be understood as programming with the transmission of value. Therefore all returned results are in the return function. Therefore it is inevitable that the object returned will have a complex structure

In rmi there is support for transferring objects, but this is a lot of restrictions such as: some objects are not implement Serializable class ,It can not be transmitted. Because Rmi will convert the object into an array of bytes to be transmitted and the sum. On the other hand, only those objects that have this class installed can send it

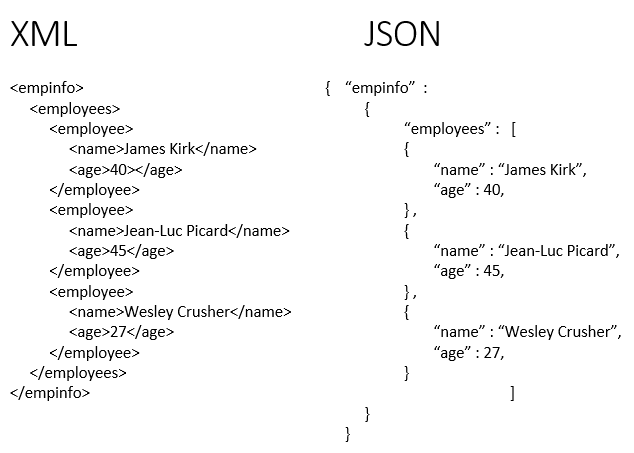
This makes it quite difficult to build software because there are times when it is necessary to send some objects in other libraries but the authors of those libraries often do not implement the Serializable class.

The explanation for this case is that most libraries operate on a single machine environment. And installing the Serializable interface will cause unnecessary difficulties because the internal properties must also be Serializable.

One solution is that we can download the libraries and fix the source code, but this is very time-consuming and many libraries are difficult to find the source code or the author of the library does not allow modification.

The solution is feasible in converting java objects to other structures such as XML, or Json, These two structures are widely applied. Having a clear structure suitable for object-oriented style

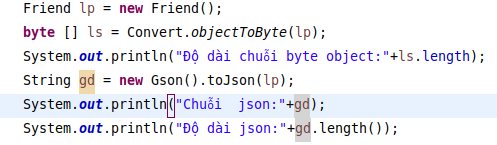
**Compare XML and Json**



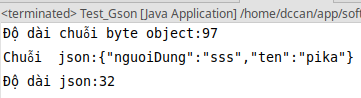
It can be seen immediately that json has a rather tidy structure so to minimize the amount of information to be transmitted, I use json to implement the topic.

In addition, after doing some tests with the Gson library (the library that converts Object java to Json), in those tests, the string length is smaller than the conversion method bytes

Code



Result



In my project, it is often necessary to use objects with the structure similar to the above sample to transfer information, so it can be concluded that using json will reduce the amount of information to be transmitted, but it takes more. A little cost when converting from Json to java object. Fortunately, the computation time for this is very small, usually less than 1ms

**More :**

And in the process of developing applications with java, one thing that we rarely pay attention to is that files compiled by the lower JDK can be run by higher JVMs but the opposite is not possible.

The question is whether it is possible to communicate rmi between two different sets of jvm, according to my test results it is possible

After testing with JDK 8,9,11, the drum system works smoothly

**What is SSL, what is an SSL certificate?**

SSL stands for Secure Sockets Layer is a standard in security technology for establishing an encrypted connection between a web server and a browser. This connection ensures that the data sent between the web server and the browser is private and complete. SSL is an industry standard used on millions of websites to protect customers' online transactions. To create an SSL connection to a web server requires an SSL certificate.

SSL uses public encryption algorithms, when the web server is installed SSL, it will generate two encryption keys: a public key and a secret key. The encryption keys are generated when you complete a number of questions to identify your website as well as your company. The public key that is not necessarily secure should be included in the CSR (Certificate Signing Request) file. This file will be sent to the CA (Certification Authority) to authenticate and issue the SSL certificate containing the information when creating the CSR (website name, company name, address ...) and allowing the server Your is using SSL.

Typically, an SSL certificate will include some information such as domain name, company name, address, company, country, and information about the expiration date of the certificate is included. When a browser connects to a secure website, it will receive an SSL certificate, the browser will check the expiry date and who is the CA certificate issuer, if it fails to verify the browser will notify This website's user is not secured by SSL.

**Reflection technique**

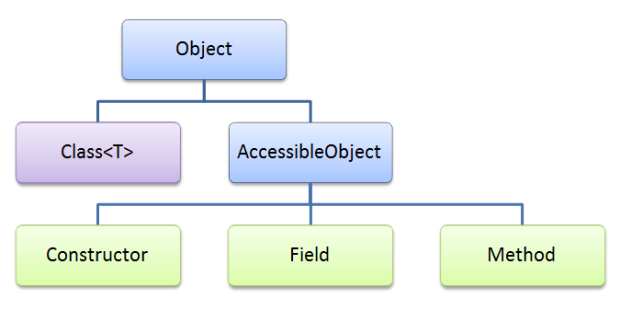
**What is Java Reflection?**

Java Refelection is a feature (called API or library also) in Java. Java Reflection allows access to the object's information (class names, fields, methods) and editing the object's fields (including private fields) during run time.

We can apply Java Reflection in cases where we do not know what the object is handled. (What is the class name, in which package, which fields, which method ...).

For example, I want to write a function to copy 2 objects that can be used for different types of objects. Then I need to know if the two objects are the same type, are there any fields, get and copy the value of each field.

In addition, for fields and methods with modifiers that are private, we cannot access them outside the class. In cases where it is imperative to call and access private fields and methods outside of that class, Reflection is a solution.



Some frameworks use Java Reflection:

* Spring
* JUnit
* Tomcat
* Eclipse (used to autocomplete)
* ...

**Limitations and disadvantages of Java Reflection**

In case you already know the class structure, have access to fields and methods, you should not use Java Reflection for the following reasons:

* Low performance: For example, it has to scan classpath to find class.
* Security Issues: Editing class / object during runtime can affect threads ... causing application to fail.
* Difficult to maintain: Reflection is quite confusing for newbies and not easy to debug, so it will be difficult to find errors. In addition, we also can not check some errors in the compile process (not found class, not found field ...)

**Components in Java Reflection**

Corresponding to the components in a class, Java Reflection also provides the corresponding classes that we can handle:

* Class: Represents the class / interface to retrieve class information (class name, super class, class modifier, methods, fields ...)
* Constructor: Handles the constructor of the class
* Field: Handling the fields of the class (name, modifier of the field, getting values, setting values for objects ...)
* Method: Handling methods of class (listing methods, executing methods, etc.)

We can get the Class object through an object with the method getClass () or via package name + class name with the Class.forName method

To get the names of constructors, fields, and methods, we can use the method getConstructors (), getFields () or getMethods (). However, methods that do not allow to get constructors, methods, fields with modifier are private, so I will use getDeclaredConstructors (), getDeclaredFields (), getDeclaredMethods ().

The getModifier () method returns the modifier of the class, method, field, but in that form,we will write the getModifierName method to display it as String.

To get values or pass values to the object's fields you can use the method field.set () / field.get ()

Source stackjava.com

**Gson library**

Gson is a Java library that can be used to convert Java objects into JSON strings. It can also be used to convert a JSON string into a corresponding Java object. Gson can work with all Java objects including objects that already exist which you don’t source code.

There are several other open source libraries that can convert Java objects to JSON. However, most of it requires you to put Annotations in your classes; or something you can't do if you don't have access to the source code. Most also do not fully support the use of Java generics.

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

Gson goal:

Provide simple toJson () and fromJson () methods to convert Java objects to JSON and vice versa.

*  Allow existing invariable objects to be converted to / from JSON.
*  Extensive support of Java Generics.
*  Allow custom for objects to convert to / from JSON.
*  Support for complex objects (with hierarchies of multiple inheritance and widespread use of Generics data types)

User manual

Methods used:

* toJson (): method used to convert Java Object to Json string (this process is called Serialization). This method has one argument that is the object to convert to the Json string.
* fromJson (): method used to convert Json string to Java Object (this process is called Deserialization). This method has 2 arguments, the first argument is the json string, the second is the Java Object data type corresponding to the json string.

**Audio encoding**

According to Itu (International Telecommunications Union), for voice coding, it is necessary to take samples at the frequency of 8000hz to represent the sample by 16bit. The Nyquist – Shannon theory requires sampling at 8000 Hz to recover

In summary, in 1s, it takes 16x8000 = 128kbit / s, equivalent to 16kB / s, not to mention the packet header with the current network speed, it is easy to talk to many users on the server side, but the server side will have problems. about bandwidth. So there is a need for information compression solutions to increase the ability to serve for user

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

<https://phuongot.wordpress.com/>

After studying, I decided to choose g729 for the following reasons

Still widely applied, the quality of the output sound is still audible, is inferior to the g711 but not significantly

* Low latency
* High compression ratio, 8 times g711
* Free open source library

**Handling jitter**

Jitter and latency are properties attributed to the flow in the application layer. Jitter and latency are used as metrics to measure network performance. The main difference between jitter and latency lies in their definition in that latency is nothing but network delay while jitter is a delay change.

The increase in latency and jitter have a negative effect on network performance, so it is necessary to monitor it periodically. This increase in latency and jitter occurs when the speed of the two devices do not match, congestion causes buffer overflows, traffic disruption.

**Comparison chart**

|  |  |  |
| --- | --- | --- |
| The basis for comparing | Jitter | Latency |
| Basic | The difference in delay between two consecutive packages | delay in package delivery on the internet |
| Reason | Cause of Network congestion | Transmission latency, serialization, data protocol, switching, routing, packet buffering |
| Precautions | Precautions Using timestamps | Many connect to the internet. |

**Definition of Jitter**

Jitter is the difference between the latency of IP packets. In other words, when the latency of the lag of the variance across the network, it causes jitter phenomenon. Can be explained by an example, suppose four packets are sent at times 0, 1, 2, 3 and received at 10, 11, 12, 13, the delay between packets is the same in all packages is 10 time units. In the other case, if these packages come 11, 13, 11 and 18, the generated delay is 11, 12, 9, 15 will be different from the above case.

The first form of delay will not affect applications like audio and video, because all packages have the same latency. However, in the second case, different lags for packets are not accepted and it also leads to the appearance of unordered packages. A high jitter indicates that the difference between lags is huge while a low jitter means that the variation is small.

**Definition of latency**

Latency is the time it takes a packet to reach its destination from the source. In network connection terms, the time spent processing a user-requested network access request and receiving a response of the request to the user. Broadly speaking, latency is the time elapsed between the execution of two events.

Latency is simply the time it takes to process messages at both the source and destination end and the latency generated in the network. There are two ways to measure network latency, the first is called unidirectional delay in which the time elapsed in the source and destination packets is measured. While in other categories, the one-way delay from node A to node B is aggregated with the one-way delay from node B back to node A and is called a round trip.

**The main difference between jitter and latency**

* Latency is the time that a data packet requires to reach its destination from the source. In contrast, jitter refers to the variation of the delay generated by packet transmission.
* Network congestion can cause jitter while latency can be generated through propagation, conversion, routing and buffer delays.
* The jitter can be prevented by using timestamp. Conversely, latency can be reduced by using multiple connections to the internet.

**Affect voice quality**

In order to explore an information environment in practice, I have built a model as follows

Source

* Recording device
* System of sending packets with the time between each packet is 10ms
* Delay generator

Destination

* The program creates sound in computer using java

Due to use in LAN, the ideal environment can be considered. Network congestion, packet loss does not occur

In case of fixed transmission time

* Characteristics: distance between each receiving packet on the same recipient
* With arbitrary latency. The sound obtained is not affected

With non-fixed transmission time (jitter)

* Characteristics: distance between each receiving packet on the receiver is different
* The sound produced will have distortion phenomenon if the time between receiving packets> 10ms
* As for the period of receiving packets <10 ms, Quality is not affected

Conclusion: packet transmission time greatly affects the transmission quality, therefore, measures need to be taken to minimize the effects of jitter.

**Reduce the effect of jitter**

Jitter is inevitable when transmitting on the network, so we can only find ways to reduce the influence of jitter to the user's listening perception.

The method I use in this topic is to create a buffer that contains dialogues

The default number of containers is 10. This means that after at least 10 packets begin to make sound and continuously receive and broadcast, if saving 10 packets, the delay compared to continuous transmission of 0.1s can be accepted.

The use of this buffer has the benefit that within 10 ms without the incoming packet, the program still has data to broadcast while waiting for the new packet to arrive.

In case there is a packet that comes first to last, using the buffer greatly supports the re-arrangement of the packet so that the sound is seamless.

This can help conceal the jiiter better during conversations

**Mesh Topology**

Mesh Topology is also known as a mesh network. Grid-based products are commonly used in networks that play an important role and cannot be deactivated. Typically, the network of an atomic power plant or the network of security and defense. For a grid, each computer device will be connected to all the other computers. It is also the familiar structure of the Internet.



**Conversation process**

**Security**

To ensure confidentiality of the conversation, the security feature is indispensable so we need to have a security mechanism that encrypts packets

However, some of the issues raised here are the length of the message after encryption and the decryption time, such as using Des encryption algorithm with the standard key length of 64 bits = 8 bytes, 10 bytes. The input gets 16 bytes of output, which is an increase of 60%. The public key encryption algorithm has the same problem, not to mention the longer calculation time

The symmetric encryption algorithms such as symmetric key, advantageous permutation code are compact, but low security easy to use a computer to find the key

Chain Block cipher (CBC) are also difficult to apply in real-time communication applications due to two main reasons. 1 is that during UDP transmission it is inevitable that packet loss should be impossible to decode at the source. 2 is that the packets are constantly being sent to encrypt the blocks, which will make it more difficult to build the program.

Choosing Public Encryption is a bad option. Not to mention the cost of calculating the solution, choosing a key is a very complicated job. With multi-member conversations, exchanging each key pair for each member is a very complicated issue because when adding 1 person, it is necessary to transfer the key to all members once more. In the case of members sharing a pair of keys, while reducing the transfer of keys, many other problems arise. If all use a fixed key pair, this is almost not effective because now the reverse code to read the current source code is very simple. If creating for each group, with the user can create and cancel groups very comfortably, the job of creating a continuous key like that will be a huge burden on the system.

With 64 bit keys, the security feature of the DES algorithm is not very good, increasing the key size to 128, 256 ... will make the calculation time longer, the longer the code will be bigger. affect the transmission of the system

So I decided to use Vigenere encryption to do it, with a size of 10 bytes there would be a total of 256 ^ 10 keys. ie greater than 10 ^ 24. In addition, this is the result after the audio has passed through the g729 compressor so it will be more difficult to detect codes because old ways of decoding will be difficult to apply.

So I decided to use symmetric encryption with the key changes every time to enhance security

**Get the key**

When the user requests this function, the program will automatically check the list of existing chat groups. If no group will automatically create a new group chat object and create a group key. token for user confirmation

The system will then send both the key and token to the user via ssl

**Sign up**

The user will then send the registered Rtcp message to the server to confirm the information, which contains encrypted tokens. The server will record information about the client's IP address, connection port

The server will then send a confirmation message, containing the group number and member number

The posting process aims to register the ip address for the server to send later. This problem also stemmed from the Rmi model which does not support source address retrieval and cannot force users to open ports on their routers.

Because the design of my program is not allowed in a conversation with 2 devices on 1 account. This design makes it difficult to log in to multiple accounts, so it is only necessary to return the information to confirm whether the account can join the conversation.

Rtcp system

It works to keep the connection to the user in silent mode because the system will automatically remove participants in the conference if the person does not send messages to save resources

Or delete the user when he or she sends the packet to end the chat

To determine whether the user is still active or not, we must record the time to send the connection maintenance packets. This method, although it has to be repeated many times, the advantage is fast, no bandwidth is required when sending client test packets.

Rtp system

Receive and send messages to other members

**Design clients / server in the conversation**

If designed with a normal model, it requires extremely high server processing speed because this is a real-time application. Increasing the sending time between packets is also a way to reduce the pressure on the server, but it increases the latency for the user, the larger the packet size, the more lost it will cause distortion. The sound is clearer when used to listen

Therefore, I propose a model similar to a grid pattern in which each member will send information to the next members.

The difficulty of this design is that each member has dynamic ip so it is very difficult. Even if you can determine the ip address, it is not possible to connect to it without Nat port.

On the other hand, if you send a packet out, the router will automatically route for us to connect to the internet, so that we can get the port and address on the router corresponding to the packet. is that after a period of inactivity, the data table that is not used in nat will be deleted so it needs to send packets periodically to stay connected.

The downside of this design is that the client will have to send more packets than usual, but the bandwidth used on the server will decrease sharply, and the packet will not need to go through the server, reducing latency.

Work

The client sends the registration packet to the server, the server will store information about the ip address and connection port

The client will get information from the chat group participants to send information

After a period of time, a packet will be sent back to the server to update the connection information

**Real time data transmission**

**Compare UDP and TCP**

Same: all network protocols TCP / IP, function to connect the machines together and can send data to each other ....

**Different**

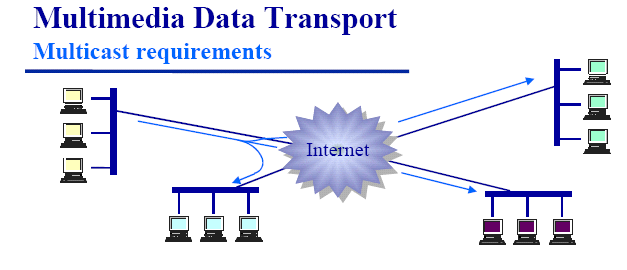
| **TCP** | **UDP** |
| --- | --- |
| Connection direction | Direction is not connected |
| High reliability | Low reliability |
| Send data stream as byte stream | Send to Datagram |
| Do not allow packet loss | Allow packet loss |
| Ensure data transmission | No Guarantee for data transmission |
| Has ordered packets | Unordered packets |
| The transmission speed is lower than UDP | High transmission speed |

Select the protocol for real-time application

Quote verbatim

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.

*Source “Nghiên cứu và ứng dụng Rtp”*

An overview of Rtp is included in the appendix

Packet size

The standard rtp packet size is 16 bytes rtp header + 10 bytes of data per 1 packet is 26 bytes. The addition of Udp header and ipv4 header is 66 bytes. Information about the 2nd floor title in the osi model, I have not fully studied but estimated the total size sent is less than 100 bytes. At 1 packet / 10 ms, the bandwidth consumes about 10Kb / s. With the current network infrastructure, this speed is quite low

**Some changes on rtp protocol**

Rtp is a standard designed for a common data transfer environment, but with many topics there are many features that will not be used. If left unchanged, there will be a lot of unnecessary waste on this subject. Conversely, modifying the packet structure will lose the ability to interact with other systems. Or it could be said that it could not link to other systems

In order to facilitate the upgrade, the processing related to packet transmission will be designed into a separate packet to ensure the convenience of future upgrades if necessary.

**Change to rctp**

The goal of Rctp is to respond and evaluate the quality of the network from which the measures are taken. However, because this topic works with encrypted audio to minimize the amount of transmission needed. This means that in every case the packet size will not change. Elements related to network quality feedback ... will not be supported in this release.

Therefore I will simplify the control packet to set the highest performance

**Join Package**

This packet is used to register with the server that a user wants to join the group

Information fields in the group

* The type: 16 bit field contains the packet type, which can be extended later
* Field length: 16bit length of packet separately from group id and standard header
* Ntp field: 64bit time stamp
* Port field: port to receive client data
* Group field: group id code
* User: string field for member verification

Type: 1000

**Bye packet**

This packet is used when someone wants to log off the conversation

Information fields in the group

* The type field contains the packet type, which can be extended later 16 bits
* Field length: the length of the 16bit packet
* Ntp field: 64bit time stamp
* Group field: 32-bit group id
* User field: 32-bit member id
* Type: 1111

**Live packet**

Use to stay connected when in, update information in the conversation

* The type field contains the packet type, which can be extended later 16 bits
* Field length: the length of the 16bit packet
* Ntp field: 64bit time stamp
* Group field: 32-bit group id
* User field: 32-bit member id

Type: 1001

**Rtp packet**

In the rtp packet, the sequense field is responsible for counting packets. However, since the emitting source is regular with increasing time, it is possible to use the last 32 bit timestamp field to take on the role of the sequense field, 32 bits is sufficient to represent the time corresponding to 50 days, so it can be used. optional

Currently only works with ITU-G729 audio that has been safely encrypted with cassava exchange key, so it can cut the crsc count, padding fields ...

Meaning of some schools

* SSRC: sync source corresponding to group id
* CSRC: corresponds to the membership code
* TimeStamp: Last 32bit of NTP

**Support classes**

In the client server model, the communication between the two parties is indispensable. With most information being strings, organizing them into arrays will be convenient for transmission because of their compact size. However, during the implementation process, there were many times I encountered mistakes in position and meaning in those areas. Moreover, reuse or upgrade code will be difficult due to having to remember the location of each information. And above all, the error due to the wrong location that often happens causes the program to have the wrong result without the error message. Therefore, the debugging will be hard, taking a lot of time.

Because of that, I decided to use objects to store data. It will avoid the problem of memory location errors, Convenient for future upgrades. Although this will help the system consume more bandwidth to operate.

The classes

The Comment class with attributes

* idNhan: the recipient's code
* idGui: the sender's code
* noiDung: Message content
* idFile: code file sent

Group class with attributes

* idNhom: group code
* username: name

Class Friend

* nguoiDung: display name of the user
* Ten: the username of the user
* id: Image number of the avatar file
* Email: the email of the user

In the above classes, the Friend class is most often used for many purposes, containing friends list information, containing personal information, containing information of group members, and information of administrators. .. If you do not use an object then it is a whole problem because you have to remember the order of each one, because each function only requires a certain amount of information.

In the above cases the necessary information will be pinned to the object. The rest receive null values

For example: Instead of having to remember in style

Args [10] = .... // i = 10 is the username

Then we can write p.getTenNguoiDung (); More convenient in use, not having to remember spare part one

**Data transfer server side**

We can summarize the operation of the server as it receives data from the database management system and converts it to the object and sends it.

Executing the query is already in another section, so I don't repeat it

The result of the query is the ResutSet object through which we can get the information in the column we need by using that column name.

To convert data to objects it is required that the field names in the object must have the same name and the same type as the tables in the result.

Restrictions on naming can be eliminated by using notation in java, but because I found it unnecessary, it was not implemented.

In java, there are 2 ways to perform queries with Database: Jdbc and hibernate. However, hibernate has many advantages over JDBC. But because I have worked hard with JDBC, this project only uses Mysql. So I think the choice is that it is better with JDBC. I can use the Reflection technique to simulate the biggest advantage of Hibernate is that it automatically adds data to the object. Not to mention that using JDBC will apply the Connection pool more easily

The result is a class that can convert data flexibly. Just make sure the request is the same name as the fields in the database, and that none of the fields will have a null value

It will then be through the Gson library to convert the data into a Json sequence

In Client side

Just use the gson library to convert to the desired object