Wearefamulei Group

COMP90015 Distributed Systems

Project 2 Report

Name	Student id	Login Name
Siming Gu	775686	simingg
Chao Yang	795047	chaoy3
Jie He	746388	hej3
Xing Han	748181	xingh2

1. Security Implementation

1.1 Definition of SSL protocol

In this project, we used the SSL (security socket layer) protocol to maintain the security in this system. SSL is a security protocol that it can provide the security in network communication. It used to authenticate between client and server or server and server; encrypt information and prevent the data are stolen. In addition, it can maintain the data integrity that there have no changes during the transmission.

1.2 Details are changed and processes in system

In this system, we create the keystores and certifications to support the SSL, and we adjust some protocols to implement it. What's more, we used dual-way authentication to ensure the security in system.

In client side:

- Creating
  kclient.keystore: store client's private key
  tclient.keystore: store server's certification
- Creating SSL socket and import the keys and certification
- Using the port number and ip address to communication

In server side:

- Creating
  kserver.keystore: store server's private key
  tserver.keystore: store client's certification
- Creating SSL server socket and import the keys and certification
- Importing the server's certification into the trustkeystore in server side. (it aims to support the communication with SSL protocol between server and server)
- Using the port number to communication

The following protocols' examples are used in the new system in transmission.

-
  ```
  {
  "command" : "HELLO",
  "supportSSL" :true
  }
  ```

-
  ```
  {
  "command" : "REPLY",
  "supportSSL" :true,
  "SSLPort":123513
  }
  ```

-
  ```
  {
  "command" : "FINISH_HELLO"
  }
  ```

-
  ```
  {
  "command" : "REDIRECT",
  "hostname" :22.22.22.222 ,
  "port": 3570
  "supportSSL": true
  "SSLPort": 12133
  }
  ```

-
  ```
  {
  "command" :"SERVER_ANNOUNCE",
  "id" :"fmnmpp3ai91qb3gc2bvs14ge",
  "load" : 5,
  "hostname" : "128.250.13.46",
  "port" : 3570
  "server_num": 5
  "supportSSL":true
  "SSLPort":12133
  }
  ```

At first, client would send "HELLO" to server that confirms it supports the SSL protocol. Then, sever side would get response - "REPLY" back to client side. If both of them support the SSL protocol, the client would send "FINISH_HELLO" to server and create the connection to transmit the data. Between sever and server, we already put the certification in server's trustkeystore, so they are already secure. In addition, the attributes "supportSSL" and "SSLPort" in "SERVER_ANNOUNCE" and "REDIRECT" are used in the function of redirect.

1.3 Benefits and limitations with SSL protocol

System used SSL protocol has three benefits

- Integrity: SSL protocol would check the integrity of information when the system gets transmission.
- Confidentiality: SSL protocol provides the authorized key encryption/decryption with the data.
- Reliability: Communication with approved certification can prevent the sensitive data in system.

In this project, we used SSL protocol because of the easier implementation. We contend that our system provides services that keep data integrity and security. Integrity protection in SSL protocol makes sure the data in communication cannot be changed; authentication in SSL protocol promises that the client and server or server and server are the correct object to communication; confidentiality in SSL protocol can confirm the transmit data are encrypted.

However, it also brings some limitations. It would cause the slow speed in transmission. Because it needs more processes to exchange handshake than other protocols. It indicates before transmitting encrypted data, it already has more pre-process information to address. And it also needs to deal with ore resources in encryption and decryption. For example, there have addition SSL information are generated that may reduce the download speed in communication. We hold that improvement may through simplify the encryption/decryption algorithm to address, but in our project, it is a good protocol to use.

2. Improvement of system

2.1 Load balance

Since we assume that the server connected in the form of tree, how the clients linked to sever will have huge impact on the behavior of the system. Figure 1 shows that if client connected to top server wants to send an activity to others, a lot of messages are transmitted to the bottom server even if no clients connected to the server on that path. When we get rid of this, the messages transmitted will be decreased a lot. Briefly speaking, we move client to more "busy" server and do not send message to its children server if no clients connected to the server under it.

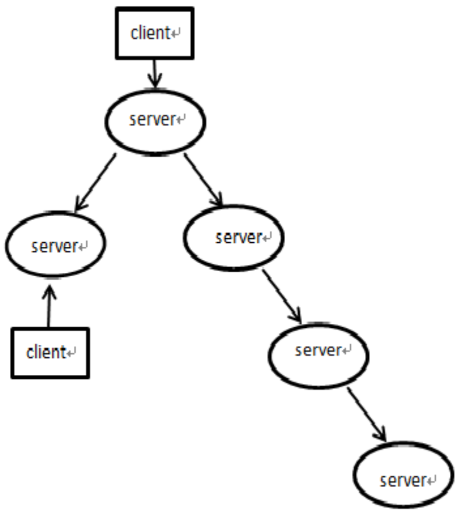

Figure 1

The method we use is first adding an extra attribute "serverNum" for each server, which contains how many servers connected to it. This attribute is put in "serverAnnounce" and will be transmitted to other server every few seconds. When clients needed to be redirected, we choose the server that has biggest "serverNum" in the system, and then pass the hostname and port number to client and connect the clients to that server.

We also add a method “clientChange” to the system. When new client logs in or logs out, the server connected to it will send “client_change” to other servers which tells them a new client comes in (or disconnected), the other servers who receives “client_change” will add (or delete) that new connection into a hash map named “connectionInfo” by using “processClientChange” method. When doing activity broadcast, if one connection doesn’t exist in “connectionInfo”, which means no clients connected to the server on this connection, it does not need to be broadcasted.

This two method works well under the condition that the number of log in and log out is much smaller than the number of activity broadcast, since every time clients come in and out messages have to be sent to other nodes connected to it. Assuming we have m servers and each server connected to an average of n servers, each time a client log in or out, about $O(mn)$ messages have to be transmitted. But if the system process more activity broadcast than log in and log out, it will work well, because only the connection who has clients will receive activities.

2.2 Merge broadcast activities

To reduce the complexity of the system, we design the new system that would merge the several broadcast activities during a short period time and send it as one activity broadcast. This way is very good for the system with large number of broadcast message. First each server would have one long type field named lastBroadcastTime to store the time stamp of server last broadcast and a Hashmap to cache the broadcast activity. And the new broadcast activity example:

```
{
"command" : "ACTIVITY_BROADCAST",
"activity1" : { ... }
"activity2" : { ... }
"activity3" : { ... }
"activity_num" : 3
}
```

The server and the client would accept the new structure of activity object and respond as same as old protocol.

The detailed steps are as follow:

1 when the server receives a new activity message from client or activity broadcast from server, it would split each activity and invoke the new method called activityMessagePacking.

2 the method activityMessagePacking would first check the field lastBroadcastTime find out whether it passed 5s since last broadcast to others:

If yes, it would broadcast this activity object immediately then update the lastBroadcastTime;

If no, it would check whether the cache is empty:

If yes, it would start a timer task: broadcast in 5 secs;

If no indicates that there is already a timer task, do nothing

then store activity object into the cache Hashmap;

3 Then if there is a timer task, it would merge all the cached activity objects into one package and broadcast then update lastBroadcastTime.

4. In order to display activity broadcast packages from server, we add a new arraylist to store these activities. First take out each activity from

received packages and rename them to “activity”, because all the activities we received is labelled with activity followed by its order such as “activity2”. Rename will make it easy for us to display. Then we put each activity into an arraylist. Remove the first element and display it every 5 seconds. If the arraylist is empty, the output thread “PringActivity” will wait, until the next package coming.

Here we can find if the system has little data currency, the server won’t wait 5 secs (this time could be adjusted by hand as the system currency and data load, for our system it is 5 secs). This would make the system adapt to the various situation: busy or not busy. When the system is busy it would merge several packages into one to reduce the network congestion. And when the network is not busy it would broadcast the message immediately to reach the delay. But how to choose a proper time is very important as it trade off the network complexity and delay. As far as we concerned, 5 secs are a proper choice to balance the network and delay.

3. Backwards compatibility

Our system’s backwards compatibility does well for each situation: new system connects to an old one and an old system connects to a new one. It sounds very tricky to manage it but at the matter of fact, it is a big challenge. And for each server there would be a field with type ArrayList to store all the connections which support SSL connections. And the new server would have two ports: one for the SSL connections, the other for the old connections.

The detailed process is as follow:

Situation 1: when a new server connects to an old server, it would first send a hello message before all other message (detailed in SSL part), then if it gets an invalid massage, that means it is an old server. Then the new server would automatically set a new the connection without SSL. And all the features we mentioned in this report of the new server would not support any more

Situation 2: when an old server connects to a new server, the new server would detect this old server by the port which is bonded with none secure connection and after authorize new server would send the entire message without SSL. And the new server would send all the activity broadcast cached one by one.

Situation 3: when a new client connects to old server, it would first send a hello message before all other message (detailed in SSL part), then if it gets an invalid massage, that means it is an old server. Then it would automatically change the connection without SSL

Situation 4: when an old client connects to new server, the new server gets a connection without SSL from the port which is bonded with none secure connection. And for all the activity broad cast, the new server won’t merge the objects and send it one by one as old server does.

By cope above 4 situations, our group managed the backwards compatibility well