Xi He(xh243), Long ma(lm675), Yanjia Li(yl2493) **Project1b**

README

Overall structure

The core of the project is the main class hello_user.java which takes the user request and perform corresponding actions. If it's a request without cookie, it would generate a new session and store it in local session table and find a backup as its backup server. It then generate a cookie to user with primary and backup server ip for next request to retrieve session data. All servers and simpleDB talk to each other to update their view table based on timestamp regularly, and new server can access simpleDB to obtain view information about who is up in the network. All the communication is done through UDP packets.

Cookie format:

Name: "CS5300PROJ1SESSION"
Value: "<SessionId>:::<Version>:::<Server_Location_Primary>,<Server_Location_Backup>"
Expiration(age): The cookie life is set as one minute + delta(3s) =63 secs which will tolerate the network traffic delay.

RPC messages:

To make a RPC, we use UDP which contains content we want to send, destination IP address and destination port. The content of all UDP packets is a class/object called Content.java class which implements serializable and it has 6 fields. Some of them may be null depending on what we send. Those fields include call ID, operation code, session ID, session data, return code and views.

Functionality

The following are source files for our application. We provide an overview for functionality inside those source file and also some explanation for key functions.

AddressResolver.java

This is the class used for server as a utility to resolve its IP address.

Content.java

This is the class used for encapsulating content into a serializable structure for RPC calls. There could be different data and attributes such as session Id, operation code and return code, etc. The constructors(overloading) will create corresponding content according to different kind of input.

GarbageCollection.java

Cleaner Thread:

This is the class used for collecting garbage from local session state table. We use an extra thread to clean those session entry with expired timestamp. The thread will work every 10 seconds to decrease the cost brought by garbage collecting.

Remove in request:
In each request, the server reads the session ID from the cookie string and lookup to see if the session has expired or not (checking expiration time). If expired, it deletes from the table and creates a new page with new cookie. If not, it would process the request as normal.

Cookie Life removes itself
Cookie has life time as mentioned in previous section, when it expires, the request will be a request without cookie, thus, resulting a new session been created.

hello_user.java
This is main class for entire project. It will handle HTTP post and get request. It retrieve the cookie talk to primary and backup servers to retrieve session data. It can generate new session and handle the front end view on client side. Other than doGet and doPost, the following functions or fields are the core:

retriveCookie()
This function is used to find the cookie that comes from this servlet based on cookie name.

getBackupServ()
This function is used to find backup server from the server's view

updateToBackup()
This function is used to update backup server when update comes, which make our system become a 1-resilient system.

constructSessProp()
This function is used to construct session state information / cookie property (class / object). This is for either constructing a new session property or constructing based on the post back cookie value.

Session State Table
The table is implemented as HashTable in Java
Key (String): session ID
Value (session state object/class): session state information
The session state object/class is an inner class that encapsulates session state information including session ID, version number, message, and expiration time.

1. *Session ID:* <ServerID(Ip address), Session num(increment each new session)>
2. *Version Number*: Integer number that increments every time from each request
3. *Message:* String type and it stores the message needs to be shown in the front end. Default as "Hello User!" and then it is whatever typed by the user and replaced by the replace button.

4. *Expiration Time:*
   Timestamp type which stores expiration time for server to check if session expired or not. Each request will then generate a new expiration time (1 minute from current time) if session hasn't expired. Wet our server side timeout as 63 seconds to leave 3 seconds network traffic delay

RPC_Client.java
The class is functioning to send UDP data based on different requests (session read, session write, exchange view) and return the corresponding received the data (received content object, sender's ip) for the same call.

RPC_Code.java
This class used to defined some constant value, such as operation code and return code.

RPC_Server.java
This is class used to receive RPC from other servers. Inside the class there is a thread which continues to run to keep listening to UDP packet at port 5300. Once the server receive an UDP packet, it will convert the bytes data into a Content object. By reading out operation code of that Content object, the server then executes the corresponding functions/actions. There are three handlers/function: SessionRead(), SessionWrite() and exchangeView().

SessionRead()
This function will read session information from session table and return it to client.

sessionWrite()
This function will write updated session information into local session table.

exchangeView()
This function gets the received view table and merge it with its own local view table and return the updated view table back to client

SessionProperty.java
This is the class object used to store session information. It's the value of the session table
This class also helps to generate cookie object for user and generate cookie string for user. Cookie life is 63 seconds as mentioned in previous sections.

SimpleDB.java
This is the class used for database interface. Inside the file, it imports several Amazon AWS libraries to use simpleDB API.

SimpleDB()

The constructor build connection between servlet and simpleDB using our secret access key.

readDBViews()
This function is used for reading views from simpleDB. It returns a list of string which contains views of server.

clearDBViews()
This function is used for clearing views in remote simpleDB. It returns true if clearing operation succeed.

writeSampleViews()
This function is used for debugging the system. It can write sample views into simpleDB.

StatusTuple.java
This is the class used for indicating a server status. This data structure will encapsulate server status information and been stored in the views table as value.

View.java
This is the class for local views inside the server. It is responsible for maintaining local views, exchanging views with other servers or simpleDB with gossip protocol and also initializing views when the server starts. It is also an extra thread which will run every 10+random delta seconds to update its internal views (to avoid traffic burst). This class has functions that can connect to simpleDB, update specific server's status information, and merage views between server or simpleDB.

Elastic Beanstalk setup

To setup an elastic beanstalk environment for our project, just follow these steps.
Step 1: Make sure all other previous servers are stopped and clear the simpleDB.
Step 2: Enter beanstalk dashboard and click "create new application"
Step 3: Select proper app name, server type(web server), environment type(Tomcat/Load Balancing) for new application.
Step 4: Upload "project1b.war" as application source and set proper deployment limit.
Step 5: Finish remaining information for application and click "Launch" after several "Next"
Step 6: Waiting for deployment for few minutes until the Health(status) become Green.
Step 7: Click "Configuration" and open up the setting for scaling. Set the minimum instance to a number you like (But do not choose a very large number) to test our distributed session management.
step 8: change the security group to allow inbound/outbound UDP protocol with any IP.
Step 8: Click the application link besides application title to enter the application.
Step 9: Using "replace", "refresh" and "logout" to explore our application.