CS5300 PROJECT 1B SCALABLE AND AVAILABLE WEBSITE

Eric Tan(eyt4), Jiabin Dong(jd836), Ray Weng(rw422)

1. Contents of war file:

EnterServlet.java
-constructor defines one socket per servlet
-contains client code - doPost/doGet for form submission and makes Remote Procedure Calls to the server, handles requests and responses, and forwards data to form.jsp for UI/debugging
-method makeRPCRequest(...) handles the actual UDP packet sending/receiving; this function has parameters the actual request, servers to query, as well as an option to wait for all responses or just send all requests. This function handles retrying of request sending and waiting for a response from the other endpoint server. All RPCs use the same anonymous socket
-creates and manages cookies for the browser
- cookie format: sessid_versionNumber_backupservers-primaryserver; where sessid looks like sessNum-svrIdPrimary and backupservers looks like ip1-ip2-ip(k-1),
-creates session IDs using an incrementing session number and a server IP address
-has constant K_RESILIENCY that determines how many backup servers a session has; so depending on this value, sends extra requests (session retrieve/write) to primary server and all backup servers
-when requests are made, if response is received, update local views list to reflect that that server is online; else if socket timeout, set that server in view to offline

RPCServer.java
-startup code for each server to identify its own IP, and attempts to connect to Amazon SimpleDB, exchanging views on startup
-handleRPCRequestion(...) handles the actual requests for session data, managing requests sessionRead, sessionWrite, and exchangeViews along with their respective methods.
-Structure of RPC call is <callID, operation code, sessionID>
-Response structure is <callID, operation code, relevant info returned by the specific operation>
-a second exchangeViews method periodically executes the gossip protocol and exchanges views randomly with another server or simpleDB (called by separate thread GossipProtocolT)
-views are serializable, so views are exchanged as a base64 serialization of the object into a string

RPCListener.java
-constantly listens for RPC requests on port 5300 and responds with relevant data
-uses received UDP packet headers to determine where (IP/Port) to send response

View.java

-Object representation of a View tuple of <Server ID, status, time> and whether the server is up or down, where time is milliseconds since unix epoch
-contains a method mergeViews that combines its own views with another server that it is gossiping with
-holds global server-wide list of known friend servers and status
-implements serializable interface to allow easy sharing of data between instances

GossipProtocolT.java
-threaded gossip protocol, exchanging views periodically using RPCServer.exchangeViews()

MyServletContextListener.java
-initializes supporting threads for the server, gossips, garbage collection

SessionDataTable.java
-Object representation of Sessions, consisting of a HashMap <String, SessionTuple>
-String is the session ID, SessionTuple is all the relevant information about the session including the data

SessionTuple.java
-Tuple representation of session state, in format <sessionID, version, expiration, data>
-defines duration of each session and cookie using SESSION_DURATION constant

In /WebContent/WEB-INF/:
web.xml
-specifies EnterServlet as the welcoming servlet page
-starts MyServletContextListener to start relevant threads

form.jsp
-the web interface for testing things

2. Elastic Beanstalk setup:
Create elastic beanstalk application environment, configured to 2-4 instances. Provide IAM User access and secret key pair with SimpleDB access (you might need to create a SimpleDB access policy for the default role that Elastic Beanstalk assigns). Assign EC2 instances to security group that allows incoming UDP packets on port 5300. Other things can be set to default. It may be necessary wait as much as over 10 minutes for AWS to finish setting things up and getting everything running properly after uploading/deploying the WAR file.

To kill a server instance, go to the EC2 instances and stop the one you are working on, then you can see the how elastic beanstalk replaces the failed instance from user interface.