ECE 428/CS 425 Programming Assignment 3 (MP3) – Report

- Amit Jaspal (jaspal2)
- Bharat Thatavarti (thatava2)

Code Design and Implementation Details.

The code is divided into a bunch of modules that perform independent functionalities so that there is minimal dependency among them. These are the following modules -

1. Reader – This is responsible for taking commands like insert, delete etc. from the end user. It processes the command, computes the appropriate nodes that should process that command and transfers the control to the Sender to pass those commands to the respective nodes.

2. Sender – It takes the message and the I.P address of the receiver and sends the message by opening a TCP connection with the receiver.

3. Listener – The Listener performs the bulk of functionalities. Whenever a listener receives a message it checks if that message is stale or not on the basis of timestamp present in the message. If the message should be processed the Listener updates the key value store accordingly and sends the acknowledgement to the sender that updates have been successfully made.

4. ReadRepair – This modules ensures that eventual consistency is maintained in the key-value store. Whenever a get operation is performed the <key, value> pairs from all replicas are queried and if any of the replicas contain stale data for a particular key, the ReadRepair module sends a message to that replica to update the value for that key.

5. Data – This is the class that maintains the state information of the key value store. It encapsulates value and the timestamp information for a particular key.

6. Initializer – This is the initializing module that initiates reader and listener threads so that the process can accept user specified commands.

Let us consider an example to further understand how these modules work in our distributed key value store. Suppose there are 4 nodes and number of replica's is 3. Now when the user gives a command like
>> insert 2 10 9
the following sequence of events happen.

1. The Reader reads the input “insert 2 10 9” from the console.

2. The Reader computes where key = 2 is stored, in this case nodes – 2, 3, 0. Now the readers sends a message “insert_background” to all these nodes via Sender module.

3. A separate Sender thread is initiated with average delay to each respective node. This Sender thread sends the message to the respective node to perform the insert operation.

4. The Listener on nodes 2,3,0 receive the “insert_background” message. The Listener perform a check if the message that is received is stale by comparing the timestamp of the message. After that the listener will update its key value store accordingly and send a message to the Sender “insert_reply” to reply that the insert was successful or not.

5. The Listener on the client receives “insert_reply” from all the three replicas and prints a message on the console that the key was inserted successfully.

   This is how these modules work to make sure that the distributed key value store works properly.