Report - Server Side Events

In this report, I have described code along with explanation of the implementation both locally and remotely.

Joining the nodes in distributed hash table to communicate between them.

Used listenOn, listenOff and listeners to communicate between them,

Adding and retrieving bindings at both locally and remotely.

CODE:

In POM file, server.name property has been defined (local/home) and jar file generates after run the project at specified location.

```
properties>
        project.build.sourceEncoding>
        <!-- HTTP server listens on this port. -->
        <server.port.http>8080</server.port.http>
        <!-- External DNS for AWS. -->
        <server.external-dns>localhost</server.external-dns>
        <!-- Where to distribute jar files (using mvn package). -->
        <server.home>${user.home}</server.home>
        <server.testdir>${server.home}/tmp/cs549/dht-test</server.testdir>
        <server.dist>${server.testdir}</server.dist>
        <server.name>dht</server.name>
</properties>
In Application file, register the server side events.
public class Application extends ResourceConfig {
        public Application() {
                super(NodeResource.class, SseFeature.class);
                packages("edu.stevens.cs549.dhts.resource");
                // TODO register SseFeature
        }
}
```

In WebClient class file, we put listenForBinding and listenOff methods. ListenForBinding tells another node that whatever changes happen to any particular key, whichever node binds to that key will be notified.

ListenOff method is useful to get disconnect from node to given key. ListenOff to node will no longer eligible to get notified.

```
public void listenOff(NodeInfo node, int id, String skey) throws
DHTBase.Failed {
// TODO listen for SSE subscription requests on http://.../dht/listen?key=<key>
// On the service side, don't expect LT request or response headers for this request.
        String uri = String.format(node.addr + UriApi.LISTEN, id, skey);
        try {
                 deleteRequest(new URI(uri));
        } catch (Exception e) {
                 throw new DHTBase.Failed("listenOff error. message: " + e);
        }
}
deleteRequest and putRequest methods:
public Response putRequest(URI uri, Entity<?> entity) {
// TODO Complete.
        try{
                 Response cr = client.target(uri)
                 .request(MediaType.APPLICATION_XML_TYPE)
                 .header(Time.TIME_STAMP, Time.advanceTime())
                 .put(entity);
                 processResponseTimestamp(cr);
                 return cr;
        } catch (
                 Exception e) {
                 error("Exception during Put request: " + e);
                 return null;
        }
}
public Response deleteRequest(URI uri) {
        Response cr = client.target(uri)
                         .request(MediaType.APPLICATION_XML_TYPE)
                         .header(Time.TIME STAMP, Time.advanceTime())
                         .delete();
        processResponseTimestamp(cr);
        return cr;
}
In State class file added addlisteners and remove listeners methods. Which adds if map contains the key
into the node.
@Override
        public void addListener(int id,String key,EventOutput eventopt) {
                 if(outputs.containsKey(id)){
                         outputs.get(id).put(key, eventopt);
                 } else {
                         Map<String,EventOutput> map = new HashMap<String,EventOutput>();
```

```
map.put(key,eventopt);
    outputs.put(id,map);
}
if(listeners.containsKey(key)){
    listeners.get(key).add(eventopt);
} else {
    Map<String,EventOutput> map = new HashMap<String,EventOutput>();
    SseBroadcaster b = new SseBroadcaster();
    b.add(eventopt);
    listeners.put(key,b);
}
```

Simply remove nodes form the node in hash table.

listenOn and listenOff methods in Dht class file.

This method register a new node for new bindings. Client send request to identify this node.

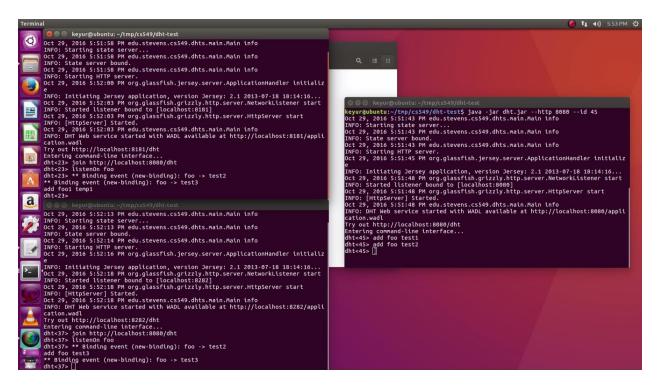
```
public void listenOn(String key, EventListener listener) throws DHTBase.Failed {
    int id = NodeKey(key);
    NodeInfo succ = this.findSuccessor(id);
    EventSource eventsrc = client.listenForBindings(succ, info.id, key);
    state.addCallback(key, eventsrc);
    eventsrc.register(listener);
    eventsrc.open();
}
```

This method stops listening to binding the node.

```
public void listenOff(String key) throws DHTBase.Failed {
    int id = NodeKey(key);
    NodeInfo succ = this.findPredecessor(id);
    client.listenOff(succ, info.id, key);
    state.removeCallback(key);
}
```

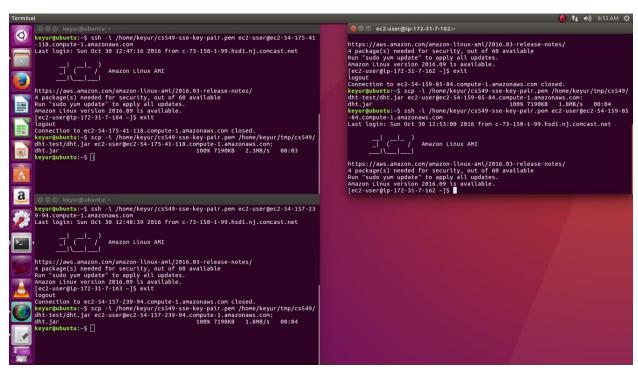
Local Test:

ListenOn, listenOff and listeners command to communicate in has table between nodes.



Remote Test:

Transfer dht.jar file on amazon ec2 instance using Secure Copy (Scp) and connect that instance using Secure Shell (SSH).



On Ec2 instances, three nodes perform DHT commands.

