CSCI 5105 - Project 1

PubSub

Achal Shantharam (shant012@umn.edu)
Aditya Balanarayan (balan016@umn.edu)
Aronee Dasgupta (arone008@umn.edu)

Implementation:

Our implementation consists of three packages: client, server and remote object. The remote object consists of an interface “PubSubService” which is implemented in the server as PubSubServiceImpl. The core functionality of the project is implemented in PubSubServiceImpl.

In the client package, the client side functionality is implemented in the Client class. This includes functionalities such as join, publish, subscribe, unsubscribe and leave. Each client also runs a UDPSubscriptionReceiver thread that constantly listens on a designated port for incoming subscriptions.

In the server package, the core functionality of publish-subscribe is implemented in the PubSubServiceImpl class. The GroupServer class includes a main method where the remote object is bound to the PubSubServiceImpl class. This is the class that needs to be run in order to test the code on the server side. Further details are provided in the “How to Run” section of this document.

The system built is multithreaded and asynchronous in a way that the packets that need to be sent to the clients are added to a sendQueue. Multiple threads from a thread pool read from the sendQueue and dispatch the packets to their respective clients. As future work, to make this more scalable, we can add multiple queues and have each thread read from a queue and dispatch the packets. This functionality is provided by the SenderThreadExecutorService and PacketSenderWorkerThread classes.

How to run:

This project is packaged as an Idea project. Since all CSE machines have IntelliJ installed, it is recommended to build it using that. In order to run the project, ensure that rmi registry is

running. Once it is running, run the GroupServer.java class. This will bind the remote object to its implementation.

Once GroupServer is running, we can run each of the ClientTest classes. These test cases simulate different scenarios with different number of clients performing different operations. All the operations are meaningfully printed onto the console.

If you wish to run the project from the command line, cd into the src folder and then compile client and server packages.

```
cd client
javac -cp ../ *.java
cd ..
cd server
javac -cp ../ *.java
cd..
```

Next start the rmiregistry in the src folder (or the folder with the .class files with you have not stored them elsewhere).

Start the server using,

```
java server.GroupServer
```

Next run the various test cases using

```
java client.TestCase<number> (Ex: java client.TestCase1)
```

Negative test cases are also included in some of the test cases.

Comparison with Google PubSub:

The PubSub system built by us and Google PubSub work on the same principles of an asynchronous Publisher System system. The primary functionality of both systems are the same but they greatly differ in development effort.

The system built by us uses Java RMI and UDP to communicate between client and server. The client and server classes were implemented from scratch. Implementing client and server functionality included setting up the network, and successfully parsing received messages. Client also had to be made multi-threaded to handle making RMIs and receiving subscriptions at the same time.

On the other hand, Google PubSub does not need a user to explicitly code a server and client. Google PubSub permits users to host a fake-PubSub server on a machine of our choice. Once

the server is hosted, we can instantiate multiple clients in a programming language of our choice and see the sytem in action. This is much simpler as most of the error handling and network set up is already implemented. Google PubSub also permits users to add topics that can be subscribed to. Our system has set of valid topics that the client has to conform with.

We ran Google PubSub using Python. “pip” was used to install the google-pubsub library and we tested GooglePubSub using the example code given in the google pubsub python page (link).