Project 2
CS505
3/30/2014

Reliable Broadcast: Java Implementation

Derek Schatzlein and Josh Reese

Structure –

For this project, we have implemented the interfaces provided in the handout in the following files: RBroadcast.java and FRBroadcast.java. In our implementation, rblisten() boots up a thread implemented in BReceiveThread.java which handles receiving for the reliable broadcast and fifo broadcast. init() boots up the channels and threads needed for reliable communication between specific processes, which uses our ReliableChannel implementation. Both init() and rblisten() must be called for the broadcast to work. There will be four threads running for a given broadcast instance: ReceiveThread (which handles channel-level receiving), SendThread (which sends messages and ACKS for the channel), BReceiveThread (which handles messages handed to the broadcast level by a channel), and the main program thread.

Important: to interface between ReliableChannel and ReliableBroadcast, we have used the ReliableChannel callback, RChannelReceiver. Do not use our callback if you plan to just make a channel on its own.

To be reliable, messages that are sent in our broadcast, when received, are then resent to all other processes (except the sender) one time. We have implemented a header class (implemented in Header.java) which contains all the information necessary to route the messages (the sender, receiver, and originator of the message, as well as sequence numbers). The various ReliableChannels involved will hand messages up to the broadcast layer, using our callback and a shared collection.

The SRBroadcast implementation involves global variables in FRBroadcast.java which must be changed to turn SRB on and off, as well as to change the timeout. Since no interface was defined, these should be changed by hand.

Usage –

For compilation, we have used ant to make everything simple. Simply type:

```
ant jar
```

and everything will build. We have included a simple testing suite (in BcastNode.java) which runs a simple broadcast between several processes (integers increasing from zero). The program takes one command line argument: passing in “fifo” will make the broadcast a fifo broadcast. Anything else will make it not fifo. Look in the tests/ directory to find the output files from each process, which are written using our callback.