

Programming Project 1

This is a first programming project using the JGroups toolkit. To get started, read and follow the instructions in

<http://www.zurich.ibm.com/~cca/sft09/proj1/README>

We provide two artificially simple protocol stacks that add a random delay to the messages: `udp1.xml` using UDP and `tcp1.xml` using TCP as basic transport. We use these stacks because the default stacks (these are in `udp.xml` and `tcp.xml`) have large message retransmission buffers, do not time out fast enough, and have extra functionality that is not needed here. The custom stacks specify small buffers and do not support state transfer between group members.

1 Group Membership

To demonstrate the effects of group membership, run the Draw demo with the protocol stack in `tcp1.xml`. Type multiple times in separate shells:

```
./jgroups org.jgroups.demos.Draw -props udp1.xml
```

Delay one instance (by typing `^Z` at the shell to suspend the process) and observe how the other instances deliver a new group membership (view) that excludes the delayed one. Continue drawing in the others. Then resume the delayed instance (by typing `fj` at the shell to resume the process) and watch it join the group again.

2 Total Order

The installation instructions mention the Chatter program. Familiarize yourself with the code. Chatter supports an automatic mode where group members repeatedly send messages. Automatic mode is enabled by specifying the parameter `-auto <no_seconds>` with a non-zero argument.

In automatic mode, a member starts sending messages as soon as the group contains enough members, as specified by the parameter `-size <no_members>`. Chatter stops sending after the specified number of seconds given with the `-auto <no_seconds>` parameter. There is also a hard-coded `Thread.sleep()` statement that inserts an artificial delay between two broadcasts.

Chat program. Run several copies of Chatter in automatic mode with one of the stacks `udp1.xml` or `tcp1.xml` and identify the differences in the output between the three parties. For example:

```
./jgroups Chatter -user alice -props udp1.xml -auto 5 -size 3 \  
> /tmp/alice &  
./jgroups Chatter -user bob -props udp1.xml -auto 5 -size 3 \  
> /tmp/bob &  
./jgroups Chatter -user charlie -props udp1.xml -auto 5 -size 3 \  
> /tmp/charlie &
```

You should start the instances quickly after each other and you may get errors during shutdown. The choice of the UDP- or TCP-based stack depends on the network between your machines. If all parties run on the same host and communicate via the loopback interface, it doesn't matter much; but when there is a WAN between the parties, UDP is subject to message loss.

Total Order. Total ordering of messages is provided by the SEQUENCER protocol. Insert the SEQUENCER protocol at the top the stacks that you use (that means, insert the string `<SEQUENCER/>` in the last line of the XML configuration file). Confirm that the messages are now delivered in the same order by all parties.