Distributed Collaboration - Assignment 5: Causality

Date Assigned: Wednesday November 5, 2014

Submission Date: Wednesday November 12, 2014(11:55 pm)

Objectives:

- Understand causal vs. non causal communication
- Implement logical/vector timestamps
- Understand the difference between vector time stamps and global scalar time stamp and last sender time stamp we saw in class
- Implement filters
- Implement factories and use abstract factories
- Implement causal broadcast

You will extend your implementation of a replicated model system to support causal communication and allow causality to be turned off and on. You will use filters and factories provided by GroupMessages in your implementation.

For extra credit you will compose your filters with replicated window system and deliver concurrent messages while honoring causality concerns.

In all cases you will use P2P communication among processes.

You will need to pull the latest changes from ColabTeaching for this project to get the appropriate traceables. Before each assignment, be sure to pull any changes that have been made. You will also need to pull util (and maybe oeall in case you do not have util in your classpath).

The assignment has been broken up for you in parts to make it easy to implement.

All assignments will most probably be changed after they are posted to clarify them or change constraints, and features. The changes will be highlighted using track changes.

Part 1: Delays

Run assignment 2 with P2P communication and delays (see slide 51) that creates: (a) the scenario of slide 10 and also my demo; (b) the scenario of slide 14; and (c) the scenario of slide 16. In these scenarios the message ordering is important; the contents of course are up to you.

Part 2: Causality

Implement causal broadcast using filters and run the same three scenarios. Print a message if you detect a concurrent message and process it.

Verify that the first one scenario runs to completion, and the second and third ones result in conflicts being detected. Observe what happens with the other two scenarios.

Part 3: Dynamic Causality

Let your code provide a GUI and API to turn causality on and off for a particular site, as I do in my demo. Turning causality on from off at a site resets the vector time stamp and buffer at that site. Turning causality off from on processes all buffered messages and all subsequent messages without any checks.

In your demo trace what happens in the three scenarios with causality first off and then on.

Part 4: Extra Credit: Recovering from Concurrent Messages

Allow your system to function even if concurrent message are received by processing concurrent messages without trying to order them while ensuring that a message is not processed before its causes. This might be a complicated or impossible problem but look at slides 58, 59 (these might move when I add review slides) on an outline of an algorithm that I myself have not implemented. If you find something in the literature that supports these semantics, please implement it and perhaps explain to the class how it works.

Part 5: Extra Credit: Causal Replicated Windows

Implement part 3 for replicated window systems.

Tracing

You should now fire the tracable events set by trace.causal.CausalTraceSetter – there are a few extra ones relevant only if you recover from concurrent messages. As always a trace of a sending event should be before the event.

Make sure the tracing in on in your clients for the events you fire. To do so, you should execute the following code before you display your user interface:

```
Tracer.showInfo(true);
CausalTraceSetter.traceCausal();
```

The setCausalPrintStatus() method in LockTracerSetter enumerates all the events you should trace, which are hopefully consistent with the PPT.

Main Class and Tags

Follow the tags specification for assignment 2 and 3 (extra credit). Add the tag DistributedTags.CAUSAL to each main class that supports causality. There are four client tags to support the three scenarios.

Submission

By the submission date, submit a link to a video (with an audio narration) showing the three scenarios with and without causality and any additional scenarios you identify as being useful; and also submit your code to Sakai. Please post the link as a private message in Piazza rather than an email (which is hard to keep track of) or a Sakai submission (the overhead of logging in to Sakai is high). Tag the link as **hw5videolink.** If you are worried about privacy issues, free to use the other means, and post a Piazza message informing me you have done so