Distributed Collaboration - Assignment 4: Distributed Locking with Awareness

Date Assigned: Monday October 20, 2014

Submission Date: Monday Oct 27, 2014(11:55 pm)

Objectives:

- Understand implementation centralized caching-based model
- Understand cache coherence problems
- Understand vetoeable-vetoer design pattern
- Understand implicit and explicit locking/unlocking
- Understand lock awareness

You will extend your implementation of a replicated window system to support a single lock that can be requested explicitly or implicitly. For extra credit you can support implicit tickle and preemptive unlocking, and you can compose your lock implementation not only with the replicated window system of assignment 3 but also the replicated model system of assignment 2.

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.

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.

Locks

Implement a single distributed lock using by supporting cache-based sharing of a centralized lock holder. The lock object provides the grant and release lock operations. Each replica will have a cache of this lock. It will provide a locking user-interface to request and release the lock, providing buttons or menu items for invoking them. The locking user-interface will also show the cached value of the lock holder, showing some special text or nothing if the lock if free. In addition it can show a transcript of lock requests and grants.

In this implementation you can assume that one of the session members holds the master lock copy and the name of this member (e.g. Alice) is known at program writing time.

Shared Window Vetoer and Implicit Locking

Make your replicated lock cache a vetoer of the replicated event queue object of assignment, thereby allowing serial access to the shared windows displayed by an application. Make sure that the locking window is not shared using the window filtering feature implemented in assignment 2.

Support the implicit locking mode wherein a mouse or key event triggers a lock request. Do not send a request to the master if an implicit request is pending . A request is pending if no message has been received from the master since the request was sent.

Add an option to the locking user-interface to switch between the implicit and explicit locking mode.

Implicit Unlocking (extra credit)

Implement preemptive and/or tickle locks and/or their combination, as discussed in class.

Shared Model Vetoer and Implicit Locking (Extra Credit)

Make your replicated model of assignment 2 a vetoeable that talks to the lock cache implemented above, thereby supporting locking semantics in a replicated model system.

Tracing

You should now fire all the tracable events shown in class. As always the events that trigger message sends should be fired before the multicast calls. The String argument of the newCase method in these events take the name of the user who makes a requestor is granted a request. Post to Piazza if there is confusion about it.

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);

LockTracerSetter.traceLock();

The setLockPrintStatus() 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 (extra credit) and 3. Add the tag FunctionTags. CONCURRENCY_CONTROLLED to each main class that supports locking and the IM model if you decide to make it lockable. Add the tag FunctionTags.LOCK_CONTROLS to the master and slave lock classes you define.

Submission

By the submission date, submit a link to a video (with an audio narration) showing your shared lockable window system in action, and if you are doing extra credit, the locked model system in action; 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 hw4videolink. If you are worried about privacy issues, free to use the other means, and post a Piazza message informing me you have done so