Distributed Collaboration - Assignment 6: Operation Transformation

Date Assigned: Wednesday November 23, 2014

Submission Date: Tuesday December 9, 2014 4 pm (Final Exam Time)

Objectives:

- Understand insert-insert operation transformation functions
- Understand insert-delete and delete-delete operation transformation functions (extra)
- Understand server-based centralized control algorithms
- Understand architecture of OT algorithms supporting multiple lists of elements of different types.

You will extend your implementation of a replicated model system to support centralized operation transformation of both the message history and edited text.

For regular credit, assume only insert operations. For extra credit, allow both inserts and deletes.

You have to support N users and M sequences. If it is easier for you, you can start with 2 users and 1 sequence. However, as I have given the detailed N-user M-sequence algorithm, you may find it easier to directly do the more general case.

Feel free to use your own architecture as long as you use filters to keep OT code separate from application and communication code, and fire the traceables mention in class. You are free to fire additional traceables provided, but they are harder to use.

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).

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

Insert Transformation Function

Implement the transformation function for transforming two concurrent inserts at both the client and server. You will need to tell the function which operation has been executed by the server so that consistent transformations can be executed by both clients and the server.

Extra Credit: Transformation Functions for Delete

Assume both insert and delete operations and now create additional transformation functions for the delete-delete and insert-delete case. You can design them yourself or search the web for them. I recommend the former.

Centralized Control Algorithm

Implement the control algorithms at the server and client using client and server filters.

Tracing

You should now fire the traceable events set by trace.ot.OTTraceSetter . As always a trace of a sending event should be before the event. As mentioned in class, the user name associated with edits traced indicate the name of the user who issued the edit.

Make sure the tracing is on in your clients and the server for the events you fire. To do so, you should execute the following code before you display your user interface in a client and before you call register in the server:

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

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

Dynamic OT

As with causality, let your code provide a GUI and API to turn OT on and off for a particular (client or server) site, as I do in my demo. Turning OT on from off at a site resets the time stamp(s) and buffer(s) at that site. Turning OT off from on handles all subsequent messages without any special filtering.

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

Delays and Test Scenarios

To show that you transform both local and remote operations, run the scenario of slide 97 on the shared text.

To show that you can support N users and M sequences, run the scenario of slide 110 on both data structures, (shared text and history) that is, assume an element is concurrently added to both sequences at three sites. To set the delay to sever, call the setMinimumDelayToServer(newValue) method on the communicator.

For extra credit you will need to show additional scenarios.

Main Class and Tags

Follow the tags specification for assignment 2 and 3 (extra credit). Add the tag DistributedTags.OT to each main class that supports OT. This means you must support this tag in all clients and the server.

OT Server

This class will be like the class SessionManagerServerStarter except that it will call the tracer setter and set up the filters before calling the register method.

Submission

By the submission date, submit a link to a video (with an audio narration) showing the scenarios with and without OT 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 **hw6videolink.** If you are worried about privacy issues, free to use the other means, and post a Piazza message informing me you have done so