Fall17 CS271 Project1

October 4, 2017

Social media are an important part of our social life these days. In this first project, you will implement a common function in most apps: LIKE. To simplify the model, we only have one post in the whole system which can be seen by all users. Each user could LIKE the post and you will need to keep track of the number of LIKEs. In other words, a user needs to get mutual exclusion over the counter for LIKE. You should develop the application logic that uses Lamport's Distributed Solution to achieve mutual exclusion.

1 Application Component

We will assume multiple clients. The life of a client is boring:

- Read the post.
- Like the post.

Clients are always curious about the number of likes associated with the post, and do so by reading *numOfLikes*. Clients need to communicate with each other to maintain the correct number of LIKEs.

2 Implementation Detail Suggestions

1. Each client should store its own view of *numOfLikes*.

2. Each client should maintain a Lamport logical clock. As discussed in the lecture, we should use the Totally-Ordered Lamport Clock, ie,$\langle(Lamportclock, Processid\rangle$ to break ties and each client should maintain its request queue.

3 User Interface

1. When starting a client, it should connect to all the other clients. You can provide a client's IP, or other identification info that can uniquely identify each client. Or this could be done via a configuration file or other methods that are appropriate.

2. A client should read the content of the post (not relevant for the purposes of this assignment). It should display the content and the current number of LIKEs (set to 0 when starting; kept up to date during the processing of requests) on the screen.

3. You should log all necessary information on the console for the sake of debugging and demonstration, e.g. Message sent to client XX. Message received from client YY. When the local clock value changes, output the current clock value. When the LIKE count changes, output the current LIKE count.

4. You should add some delay (e.g. 5 seconds) when sending a message. This simulates the time for message passing and makes it easier for demoing concurrent events.

5. Use message passing primitives TCP/UDP. You can decide which alternative and explore the trade-offs. We will be interested in hearing your experience.

4 Demo case

For the demo, you should have 3 to 5 clients. Initially, they should read from the same content file and display the following initial information:
"TESTCASE CONTENT" Like:0
Then each client can send requests and LIKE the post.
You will need to maintain the correct number of LIKE and display the order of requests.

5 Deadlines, Extension and Deployment

This project will be due 10/18/2017. We will have a short demo for each project For this project's demo, you can deploy your code on several machines. However it is also acceptable if you just use several processes in the same machine to simulate the distributed environment.

6 Teams

Projects should be done in team of 2. You can use Piazza to form teams.