Distributed lock with Lamport clocks

Large Scale Distributed Systems

Objectives

Use Maelstrom (`github.com/jepsen-io/maelstrom`) to observe executions of wrong or limited implementations of a distributed lock. Implement a distributed lock using Lamport clocks. Use Maelstrom to help in debugging tentatative implementations towards a correct version.

Software

Have a JVM and Python installed. Download the Maelstrom version made available, which defines a *workload* named *lock*, used with `-w lock`.

Tasks

1. Study the code of a trivially wrong implementation, in `wrong-lock.py`.
2. Run it in Maelstrom and observe the incorrect behavior, analysing histories and message diagrams.
3. Study now the code of a centralized lock in `lock-single-node.py`.
4. Run this version in Maelstrom and observe the behavior in the following configurations:
   - a single server node, with 3 clients, with options: `--node-count 1 --concurrency 3n`
   - several server nodes, with one client per server, with option: `--node-count 3`
5. Implement a distributed lock based on Lamport clocks, as described in

   `https://lamport.azurewebsites.net/pubs/time-clocks.pdf`

   that with several server nodes, but allowing only one client per node. Test it with `--node-count 3`.
6. Generalize it to allow several clients per server, as the centralized lock provided. First test it with only one server, then the general case of several servers and several clients per server, e.g., `--node-count 3 --concurrency 2n`.