Lab 3

Eventually Consistent Blackboard

Consistency Trade-offs

- Balance between strictness of consistency and efficiency/scalability
  - How “much” consistency we need, depends on the application

Consistency in our Blackboard so far

- Simple board (Lab 1):
  - Inconsistent
- With a central leader (Lab 2):
  - Consistent
  - Not scalable
- In Lab 3: *eventual consistency*
  - Blackboards on different vessels can be inconsistent for a while
  - But will synchronize over time

Eventual Consistency

- All replicas *eventually converge* to the same value
- A protocol for eventual consistency:
  - *Writes* are eventually applied in total order
    - same order on all replicas
    - lead to the same value
    - eventual consistency
  - *Reads* might not see most recent writes in total order

Design Considerations

- In eventually consistent (large) data-stores:
  - Write-write conflicts are rare (but **not** in our board!!!)
    - 2 processes writing the same data is a rare case
    - It can be handled through simple mutual exclusion
  - Read-write conflicts are more frequent
    - 1 process reading a value while another process is writing a value to the same variable
    - Eventually Consistent designs should efficiently resolve such conflicts

How-to hints!

- Use logical clocks
  - Each post has a sequence number:
    - Sequence number of a new post:
      the last sequence number received + 1
  - On a vessel:
    - Posts are ordered by sequence numbers
    - If two posts have the same sequence number, break ties with some rule (e.g. prioritize highest IP address)

Assumptions/Requirements

- Boards are distributed:
  - No centralized leader, no ring topology
  - You can base on Lab 1's code (if you want)
- Each post is updated to the local board, then propagated to other boards
- All boards are eventually consistent

Measurements

- Time for the Blackboard to reach consistency:
  - Each vessel receives a fixed number of posts
  - Measure time to reach consistency state:
    - at a vessel (simplified): duration from when the first post is received until all posts are received
    - of the board: the longest time among all replicas.

Measurements (cont.)

- How to measure?
  - Post a message to vessels at (almost) the same time
    - Write a shell script or a program to automatically post messages
    - An example bash command to send a POST request:

      ```
      curl --request POST '<ip>:port' –data
      ‘post_message’
      ```
  - Record the time for the Blackboard to reach consistency
    - Time in Repy: using elapsed time (RepyTutorial example 1.4)

Task 1: Video

- Video: demonstrate that your Blackboard is eventually consistent
  - Inconsistent for a while
  - Then becomes consistent
- Briefly discuss pros + cons of this design

Task 2: Latency

- Run a simple scenario where each vessel receives a fixed number of posts
- Measure the time for the Blackboard to reach consistency, when using 3, 6 and 9 vessels (get vessels from different locations/ continents if possible)
- Plot a graph: time to reach consistency as a function of the number of vessels
- You can use any tool of choice such as openoffice, excel, google docs, matlab, matplotlib, …
- Submit the graph, as a pdf file

Task 3 - Optional: “Partitions” in the network

- Separate the nodes in 3 groups {A,B,C}
  - each of the pairs of groups (A,B) and (B,C) should have one node in common
- Every node propagates the updates it receives to the members of the sets it belongs
- Then measure the several “consistency latencies”, i.e. the time for the Blackboard to reach consistency:
  - within one set
    (i.e. start sending messages to a node that belongs only in set A, and measure when another node in A gets consistent)
  - all sets
    (i.e. start sending messages to a node that belongs only in set A, and measure when another node that belongs only in C gets consistent)
- Submit the related graphs, and compare also with the results of task 2