Lab 4: Quorum based blackboard

Or consistency through voting

Quorum: voting based protocol

- Read the blackboard:
  - From N/2 nodes (including the interface node)
  - Display the most recent version (some logical timestamp might be needed);
- Write to the blackboard:
  - Not only to our local copy but also to N/2 nodes (N/2 + 1 in total)

Pitfalls & design choices in the write protocol

- Do you need to send in only the new message or the entire blackboard?
- Do you need to exclude others from writing at the same time or not?

Answering such questions is a design choice that might affect the read protocol also. Make sure your protocols are correct.

Assumptions - Suggestions

- We believe you are experienced enough with Seattle to handle 2 open connections ;)
- A suggested algorithm (W=N/2 +1, R=N/2):
  - Read:
    - Request blackboard copies from R nodes
    - Show the most recent one
  - Write:
    - *Remotely lock* W nodes
    - *Read* from R nodes (out of the W locked ones) to get the most recent blackboard
    - *Write* the new blackboard to the W locked vessels (you can start by sending the entire blackboard)
    - *Unlock* the W nodes
- Hint! Beware of deadlocks! Lock in ascending or descending IP/id order [cf. Dining Philosophers Problem]

Task 1

- Implement the described protocol in the following setups:
  - Setup 1: 4 Vessels (W=3, R=2)
  - Setup 2: 8 Vessels (W=5, R=4)
- Demonstrate your working quorum based blackboard (in the video)
- Present, explain and motivate your design choices (in the video)

Task 2 (Optional, +2 points)

- Compare latency with previous lab solutions (leader election, eventual consistency, event. consistency with partitions if you have done it)
  - 8 vessels
  - Submit a graph/table

Task 3 (Optional, +2 points)

- Lock nodes in random order. In order to avoid deadlocks look for:
  - Timeouts (on acquiring a lock)
  - Dynamic back-off (according to #locks already acquired)
  - Compare your performance to the results in task 2 (if you also implemented task 2)