Lab 2

Reliable Blackboard – Centralized

Distributed Blackboard

- We have a simple working version so far… Let's make it better!
- Reliable and consistent
  - No message gets lost
  - Every board shows messages in the same order
- How? Centralized version!

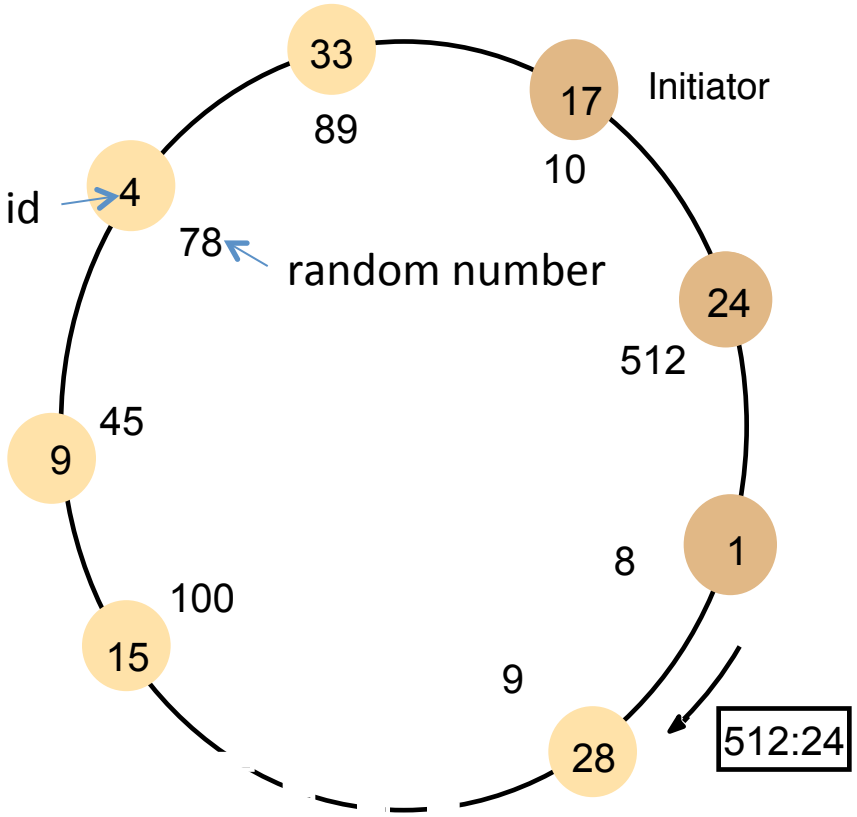
Distributed Blackboard – Centralized

- Each post is sent to the leader, which distributes it to the network
- The leader should be able to correctly handle multiple posts from different nodes
- But who is the leader?

Leader Election

- Use the Ring-based Election Algorithm (see lecture slides) when starting the board in order to decide the leader
  - Define a ring topology
  - Every node should send **only** to their next neighbor
  - Use a locally generated random number as a criterion for selecting the leader (e.g. highest wins)
- Every node acts as an initiator in the beginning (n elections running concurrently)
- Every node starts sending their updates only to the leader, when one exists
- Simplifications (but feel free to impress us):
  - Not dynamic – only run election in the initialization of the protocol
  - Assume that communication between neighbors is reliable

Snapshot of one election



Task

- Explain your leader election algorithm
- Use a field in the webpage to show who the leader is and what its random number is
- Show that concurrent submissions do not lead to problems anymore
  - with multiple browsers submitting to the same vessel concurrently, and…
  - with multiple vessels submitting concurrently
- Briefly discuss pros + cons of this design

Notes

- Note that besides electing a leader you need to make sure that the ordering of blackboard entries is the same on every vessel, even in corner cases
  - The concurrency functionality that we introduced during the lab introduction will help you solve these problems
- Also keep an eye on the following:
  - Once you register a callback function with waitforconn(ip, port, function), every time it is triggered a new thread is spawned that runs the function
  - If the function accesses shared variables or data structures, you need to take care of any synchronization needed (i.e. at this point you should assume that Python is not thread-safe when accessing shared data)

Optional extension

- Note: completely optional
  - We still give you up to 10 points even without this extension
- Handle dynamic networks:
  - What happens if the leader fails while the program is running?
  - What happens if a node during election cannot reach its next neighbor?