CHALMERS
UNIVERSITY OF TECHNOLOGY

Distributed Computing and Systems
Chalmers university of technology

Distributed Systems

Lab 2

Agenda

- Lab 2 Introduction
- Solution cost for lab 1
- Mininet Python API (optional)
- A small note on threads

General notes about the labs (1)

- Lab rooms 3354 and 3358
  - Attendance not mandatory.
- The workload of each lab is designed for 2: Works best with a lab partner!
  - Discussion page in Canvas.

General notes about the labs (2)

- Mininet is used as an infrastructure to test your servers on complex topologies with many nodes.
- Servers are “plain” Python, that can run everywhere.
- You can test them locally, on your own machine.
  - Might need to change the default port and how to contact the neighbors.
  - E.g. Start two servers on localhost, one on port 8080 and one on 8081.
- Then switch to Mininet to test with a bigger topology.

Lab 2 Introduction

A CENTRALIZED BLACKBOARD

Distributed Blackboard

- We have a simple working version so far…
  - But you might get inconsistencies
  - Let’s make it better!
- Consistent Blackboard:
  - All board show messages in the same order
- How? Centralized version!

Distributed Blackboard – Centralized

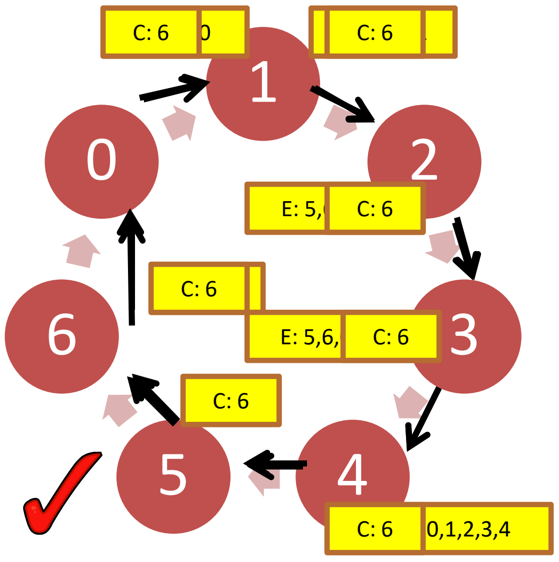
- Each post is sent to the leader which distributes it to the network

- The leader should be able to handle correctly multiple posts from different nodes…

- But who is the leader?
- We will do a leader election:
  - The lecturers will present leader election strategies
  - For this lab, we will use a simple (ring based) algorithm

Leader election on a ring

1. Node 5 initiates the leader election process. It sends ID to its next node in an "Election" (**E**) message.
2. When node 6 receives the message, it appends its ID and forwards the message
3. When the message gets back to the process that started the election:
   i. it elects the node with highest ID as coordinator
   ii. changes the message type to "Coordination" message (**C**) and circulates it in the ring



- This is an example, you can optimize it.
- Also, in our case:
  - Every node will act as an initiator in the beginning .
  - n elections running **concurrently**.
  - Eventually they all agree on the same leader.

Leader Election

- Use the Ring-based Election Algorithm when starting the board in order to decide the leader
  - Find your neighbor (e.g. the node with the next IP number)
  - 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)
- The protocol stars running as soon as the nodes are up .
  - you might have to wait a bit to make sure everyone has booted .
  - e.g. using sleep(1) to wait for 1 second.
- 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

After the election

- The leader is established and everyone agrees on it.
- After the election, nodes send new entries directly to the leader (no ring any more).
- The leader can serve as a centralized sequencer:
  - The decides the correct, global order of all messages.
  - Everybody else follows that order.

Task 1
Leader Election

- Explain your leader election algorithm
- Use a field in the webpage to show who the leader is and what its random number is
- Discuss the *solution cost* of the leader election algorithm that you use
  - see discussion about cost later

Task 2
Blackboard (centralized)

- Show that concurrent **submissions** do not lead to problems anymore
  - all blackboard entries always appear in the same order.
- Demonstrate the cost of your solution (i.e. cost of a post delivered to all nodes)
- Briefly discuss pros + cons of this design

Optional Tasks

- Note: completely optional
  - We still give you up to 10 points even without this extension
  - The optional task can give up to 2 points

- Handle dynamic networks:
  - What happens if the leader fails while the program is running?
  - What happens if a node during the election cannot reach its next neighbor?
- Concurrently delete/modify entries in the blackboard.

Video checklist:

Want we want to you to explain in the video

1. Show that the leader election works and explain how.
2. Show that all nodes have the same ordering of messages. Explain how that works.
3. Discuss the cost of your solution (for the leader election and for the second phase)
4. Pros + Cons of the design.

Optional:

5. Dynamic leader election?
6. Node failures during leader election?
7. Delete/modify?

Deadline:
November 28

For each of the above, <u>show the relevant parts</u> of your code as well!