Leader election: motivation

- IBM token ring (1970)
  - For local area network
  - Single token traversing ring
  - Station with token was allowed to send

- How to
  - Start the network?
    - 0 tokens
  - Recover from an error?
    - >1 tokens

Given a graph \( G \) of nodes, design a protocol that will elect a single node as leader

- There is one node with \( u \) leader = true
- For all \( x \in V \), we have \( x \) leader = false

Assumptions
- \( G \) is connected, i.e., nodes can reach each other; we assume a ring network here
- Nodes have unique identifiers \( v \) or
- Eds a MAC address
- Note that nodes do NOT know \( V \) i.e., the set of identities in the graph

Requirements
- Guarantee at most one leader is elected (and once elected stays elected).
- Progress essentially a leader is elected.
- Used to
  - Recover from errors
  - The leader coordinates the repair
Some non-solutions

- Node $i$ with $C[i].id = 0$ becomes leader
  - May not exist, e.g., if identifiers are based on MAC addresses

Consider the following protocol for node $i$

Send clockwise (right) $C[i].id$
Receive counterclockwise (left) id
$C[i].leader = (C[j].id < i)$

This protocol assumes that $C[i].id = i$, i.e., assigned increasing along the ring, this is not necessarily the case

LeLann’s protocol: leader election on a ring

- Assumption
  - FIFO message passing and unique identifiers
  - Note: nodes do not know the size of the ring

- Protocol for node $i$

  ```
  L = 0
  $C[i].leader = \text{false}$
  send right $C[i].id$
  while $C[j].id 
eq C[i].id$
    do receive left id
    $L = \min L, i$
    $C[j].leader = (C[j].id = max \{i\})$
  ```

LeLann

- Why does this work?

- What is the message / round complexity?

- What if message passing is not FIFO?
Why does this work?

- See proof further on

What is the message / round complexity?

- Every node forwards messages until it receives it’s own id back.
- If the size of the ring is $n$, each node sends $n+1$ messages.
- Total number of messages sent if $n\alpha+1$.
- Round complexity is $\alpha$.

What if message passing is not FIFO?

- Homework :)
LeLann proof of correctness (3)

- Progress: eventually a leader is elected.
  - Let node $i$ have the smallest $\ldots i$.
  - Initially node $i$ sends $\ldots i$ to its right-hand neighbour.
  - This means a message $\ldots i$ is either in transit on a link (meaning the next node will eventually receive it) or received by the node (meaning it will be sent out to the right by that node).
  - Whenever this message is sent, it moves one step closer back to node $i$.
  - Eventually node $i$ receives $\ldots i$ (and sends it once more the right) and then stops.
  - It determines that $\ldots i = \min i < 1$ and hence becomes leader as required.

What if nodes do not have unique identifiers?

- Then there exists a symmetric configuration $\ldots$
  - where all nodes have the same state, and all edges have the same state.
  - I.e. either all nodes are leaders, or no node is leader.
- Starting in $\ldots$ let all nodes take a step (the same) in turn, then
  - all steps are local steps (changing the local state to a new state, the same for all nodes).
  - all steps are receive actions (receiving the same message), or
  - all steps are send actions (sending the same message).
- Therefore the resulting configuration $\ldots$ is again symmetric.
- We can repeat this forever, never reaching a state where there is exactly one leader.
- This is called a symmetry argument.

Peterson’s protocol: leader election on a ring

- Bidirectional communication
  - Nodes can send messages clockwise and anticlockwise.
- Idea: algorithm proceeds in rounds
  - First round all $n$ nodes are active and participate.
  - If a round starts with $k$ participants, at least $k/2$ and at most $k - 1$ will be eliminated (and become passive).
  - If a round start with 1 participant, it will declare itself leader at the end of the round.
Peterson's protocol: leader election on a ring

The protocol for node $i$:

```c
if (i.active == true && i.leader == false)
    send left (i.id); send right (i.id);
    receive right (i.id);
    if (i.id == (i.left.id) && (i.left.id == i.right.id))
        i.leader = true;
    else if (i.id < (i.left.id) || (i.id < i.right.id))
        i.role = passive;
    else /* passive or leader */
        receive right (i.id);
        send left (i.id); receive left (i.id);
        send right (i.id);
```

Why does it work?

- Node can only survive (remain active) if both its left and right active neighbour are smaller.
- Therefore at most half of the nodes can survive.

What is the message / round complexity?

- In every round $2r$ messages are sent.
- An active or passive node sends exactly $2$ messages in each round.

So: message complexity is at most $2r \log n$.