Chapter 3
Leader Election in Rings

Distributed Systems

Summer Term 2020

Fabian Kuhn

Leader Election

General goal: Elect some node as a leader

Leader Election Problem:

Each node eventually whether it is a leader or not subject to the constraint that there is exactly one leader

- *implicit leader election:* the non-leader do not need to know the name of the leader (a.k.a. test-and-set)
- *explicit leader election:* each node knows the name of the leader

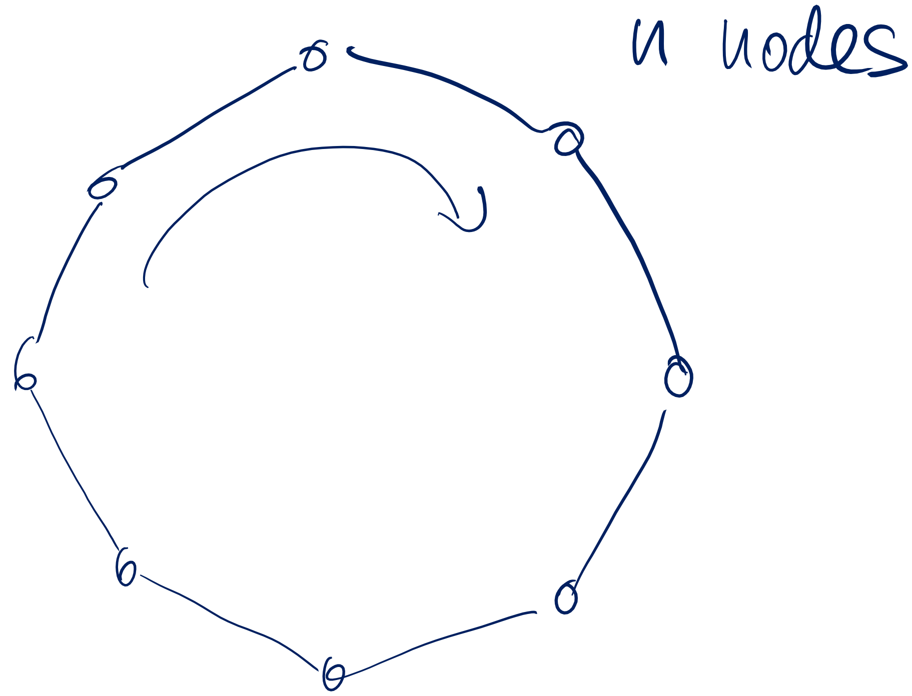
More formally:

- 3 states: undecided, leader, non-leader
- Initially, every node is in the undecided state
- When leaving the undecided state, a node goes into a final state
  - Final state: leader or non-leader
  - Implies termination…

Ring Network

For this lecture, we assume a ring topology

- Many important challenges already reveal on ring networks



Anonymous Systems / Uniform Algorithms

Definition: A distributed system is called **anonymous** if the nodes ***do not have unique identifiers***.

- That is, initially all nodes are indistinguishable from each other

Definition: A distributed algorithms is called **uniform** if the **number of nodes n is not known** to the algorithm (i.e., to the nodes)
If **n is known**, the algorithm is called **non-uniform**.

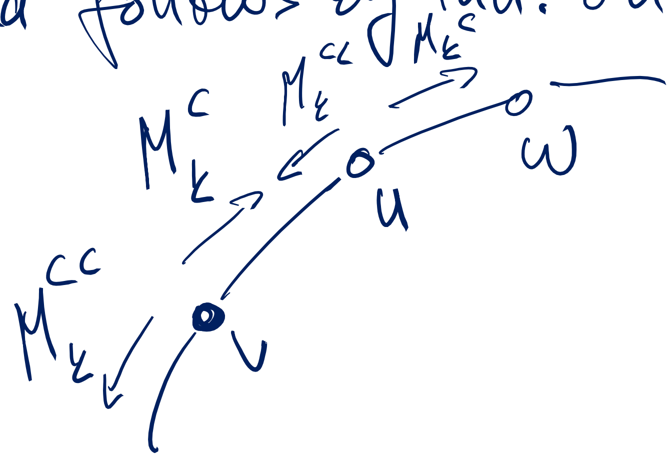
Leader Election in Anonymous Rings

- Is it possible to elect a leader in an anonymous ring?
  - Say if communication is synchronous and the algorithm is non-uniform?

Lemma: After k rounds of any deterministic algorithm on an anonymous ring, every node is in the same state S_k.

anonymous → every node has the same initial state S_0

Lemma follows by ind. on rounds:



<u>after round $k-1$:</u>

all nodes in state S_{k-1}

all nodes send & recv. the same msg. in round k

→ still all in the same state S_k

Leader Election in Anonymous Rings

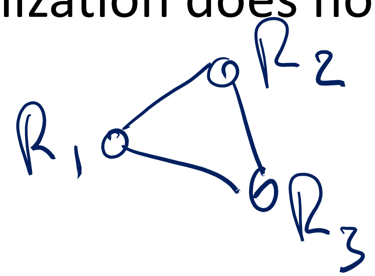
Theorem: Deterministic leader election in anonymous rings is impossible.

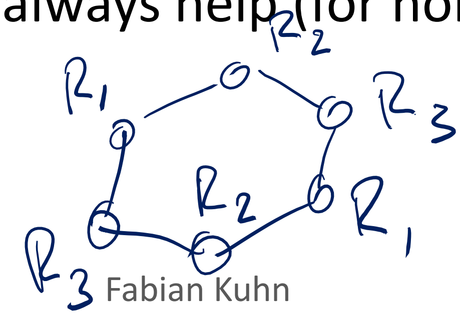
Proof:

- All nodes are always in the same state (previous lemma)
  → at the end either one or all nodes are in the leader state

Remarks:

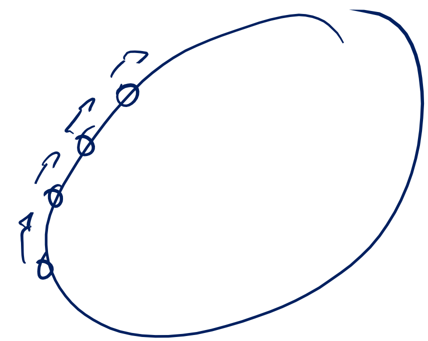
- Holds for synchronous algorithms and thus also for asynchronous ones
- Holds for non-uniform algorithms and thus also for uniform ones
- Sense of direction does not help
  - Sense of direction: distinguish clockwise from counter-clockwise direction
- Randomization might help (can be used to break the symmetry)
- Randomization does not always help (for non-uniform alg.)





Leader Election in Asynchronous Rings

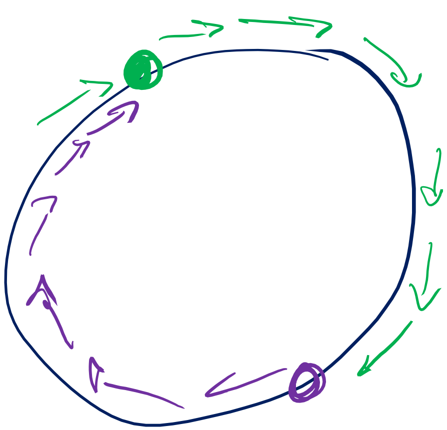
- For simplicity: assume sense of direction



Algorithm 1 (Clockwise leader election):

Each node v executes the following code:

1. Node v stores largest known ID in m_v
2. Initialize $m_v := \mathrm{ID}(v)$ and send $\mathrm{ID}(v)$ to clockwise neighbor
3. **if** v receives message with $\mathrm{ID}(w) > m_v$ **then**
4. $\quad$ v forwards $\mathrm{ID}(w)$ to clockwise neighbor and sets $m_v := \mathrm{ID}(w)$
5. $\quad$ v decides not to be the leader if it has not done so already
6. **else** if v receives message with $ID(v)$ **then**
7. $\quad$ v decides to be the leader



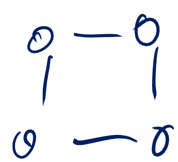
Clockwise Leader Election: Analysis

Theorem: The clockwise leader election algorithm correctly solves the leader election problem in $O(n)$ time with message complexity $O(n^2)$.

correctness: largest ID makes it completely around the ring

all smaller IDs will be stopped at some node → they do not make it around the ring

time compl.: largest ID gets back to its node in n time steps

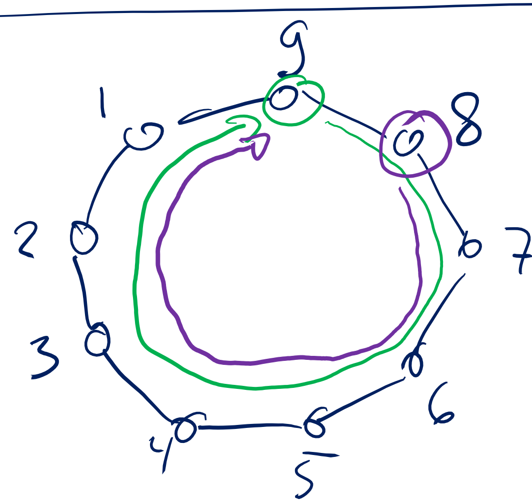


Clockwise Leader Election: Analysis

Theorem: The clockwise leader election algorithm correctly solves the leader election problem in $O(n)$ time with message complexity $O(n^2)$.

msg. compl. $O(n^2)$ trivial

also $\Omega(n^2)$ in worst case:



Clockwise Leader Election: Analysis

Theorem: The clockwise leader election algorithm correctly solves the leader election problem in $O(n)$ time with message complexity $O(n^2)$.

Remarks:

- Time complexity is optimal, message complexity maybe not?
- Algorithm distinguishes clockwise and counter-clockwise neighbors
  - This is not really necessary

each node can sent its ID in one direction

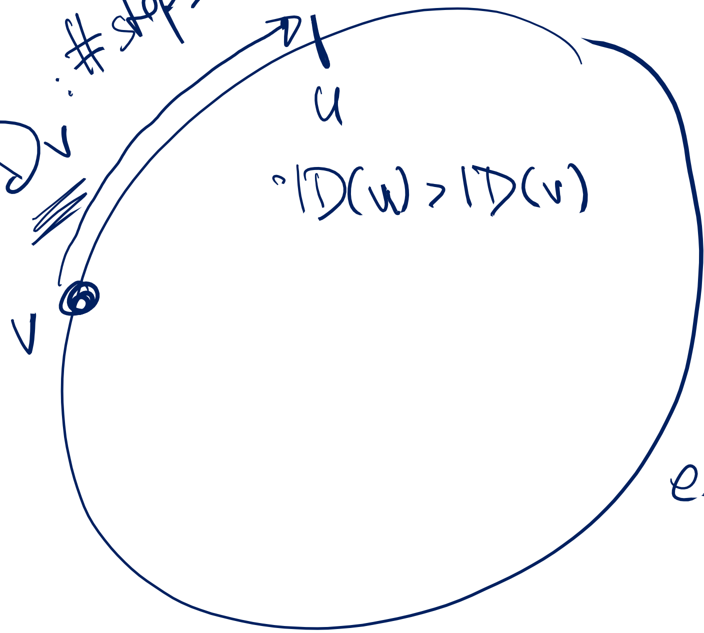
How can we improve the message complexity?

- do random delays ?
- choose the ID at random
- choose some random candidates

Randomized Clockwise Leader Election

Theorem: With random IDs, the clockwise leader election algorithm has an expected message complexity of $O(n \log n)$.

D_v : # steps of ID(v) before it is blocked



msg. complexity M

$$M = \sum_{v \in V} D_v$$

$$\mathbb{E}[M] = \mathbb{E}\Big[\sum_{v} D_v\Big]$$

lin. of expectation

$$= \sum_{v \in V} \mathbb{E}[D_v]$$

$$= n \cdot \mathbb{E}[D_v]$$

Randomized Clockwise Leader Election

Theorem: With random IDs, the clockwise leader election algorithm has an expected message complexity of $O(n \log n)$.

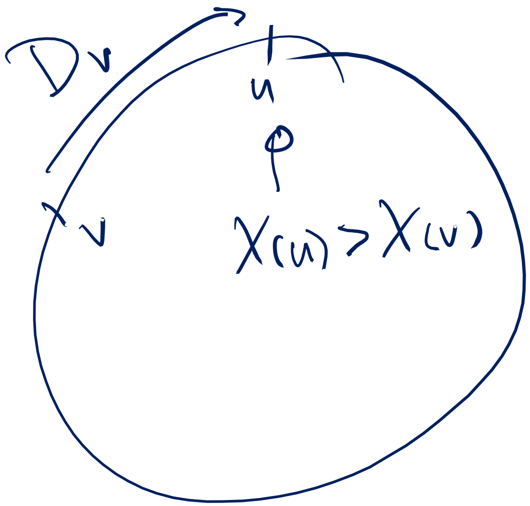
What is $\mathbb{E}[D_v]$?

$X(v)$: random numbers (rand. ID)

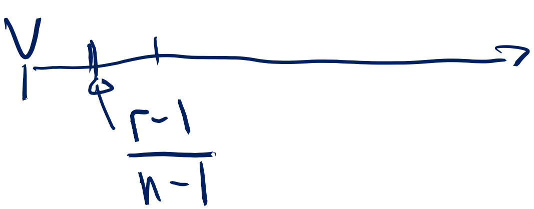
R_v: rank of the rand # of v

($R_v = 1$: v has the largest #)

($R_v = n$: v has the smallest #)



$$\mathbb{E}[D_v] = \sum_{r=1}^{n} \underbrace{\mathbb{E}[D_v \mid R_v = r]}_{\substack{= n \ (\text{if } R_v = 1) \\ \leq \frac{n-1}{r-1} \ (\text{if } R_v > 1)}} \cdot \underbrace{\mathbb{P}(R_v = r)}_{= \frac{1}{n}}$$



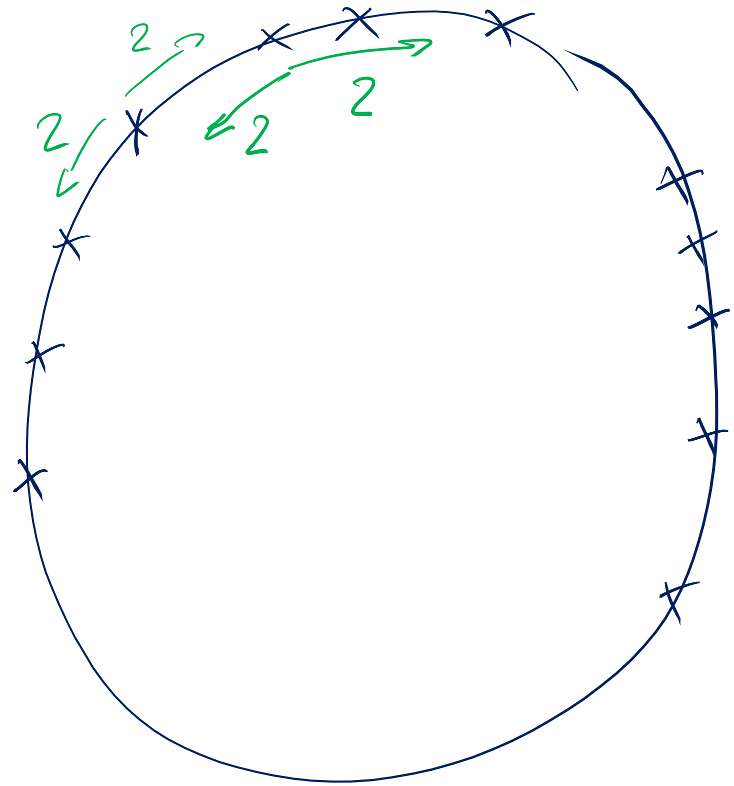
$$\leq 1 + \sum_{r=2}^{n} \frac{n-1}{n} \cdot \frac{1}{r-1} \leq 1 + \sum_{i=1}^{n-1} \frac{1}{i} = 1 + H_{n-1} \leq 2 + \ln n$$

A Deterministic Message-Efficient Algorithm?

- Try to make sure that most IDs are not sent very far

- at the start, all nodes are active
- exchange IDs with neighbors
- become inactive if some neighbor has a larger ID
- no 2 neighboring active nodes

$\longrightarrow \leq \frac{n}{2}$ active nodes



Radius Growth Algorithm

Basic idea: $i = 0, 1, 2, \ldots$

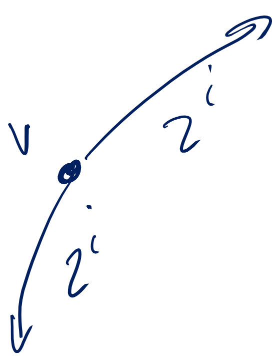
- The algorithm consists of phases, initially all nodes (IDs) are active
- After phase $i \geq \cancel{1}\ 0$ distance between any two active nodes is $> 2^i$

Algorithm:

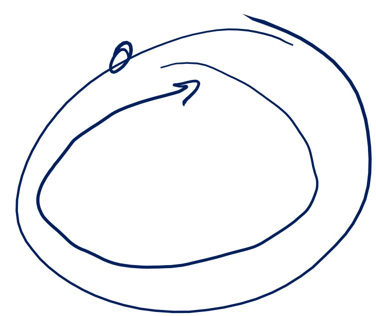
phase i

active nodes send their ID to dist 2^i in both directions, and ~~wait for~~ echo

v stays active



at some point $2^i > n$



Radius Growth Algorithm: Analysis

Theorem: The radius growth algorithm solves uniform, asynchronous leader election in time $O(n)$ with message complexity $O(n \log n)$.

time compl.

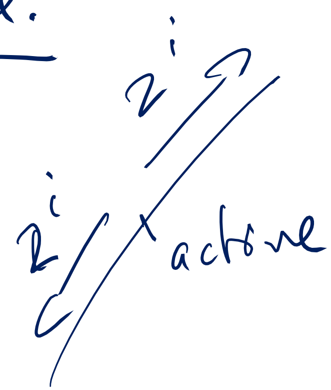
phase i: $O(2^i)$

largest i: $\hat{i} = \lceil \log_2 n \rceil$

total time: $O\left(\sum_{i=0}^{\hat{i}} 2^i\right) = O(2^{\hat{i}}) = O(n)$

msg. compl.

phase i



$\underbrace{\#\text{active}}_{< \frac{n}{2^{i-1}}} \cdot O(2^i) = O(n)$

#phases $O(\log n)$

Message Complexity Lower Bound

Recall: The asynchronous execution / schedule of a message passing algorithm is defined by the sequence of send and receive events

Remarks:

- We will assume that no two events happen at the same time
  - Such events can be ordered arbitrarily
- An execution of an asynchronous algorithm is determined by the algorithm and by an “adversarial” scheduler that decides about message delays, etc.
  - When proving a lower bound, we take the role of the scheduler
- We assume FIFO order for messages on the same edge
  - Only makes a lower bound stronger (and can always be enforced)

Message Complexity Lower Bound

Assumptions: For simplicity, we make the following assumptions:

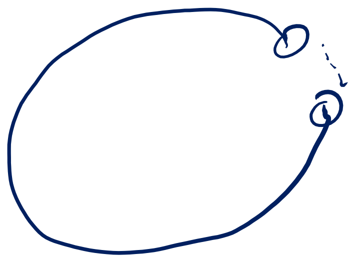
1. Asynchronous ring, where nodes may wake up at arbitrary times (but at the latest when receiving the first message)
   - For convenience, we will assume that $n = 2^k$
2. Uniform algorithms where the maximum ID node is elected as the leader
   - Assumption can be dropped with a more careful analysis
3. Explicit leader election (every node needs to learn the max. ID)
   - Can be enforced with additional $O(n)$ messages
     (at the end, the leader can send its ID around the ring)
4. For the proof, we have to play the adversary and specify in which order the messages are delivered…

Open Schedule

Open Edge: Given a (partial) schedule, an edge $\{u, v\}$ is called open if no message has been received over this edge.

– Some messages might have been sent but not received over the edge

Open Schedule: A schedule for a ring is open if there is an open edge.

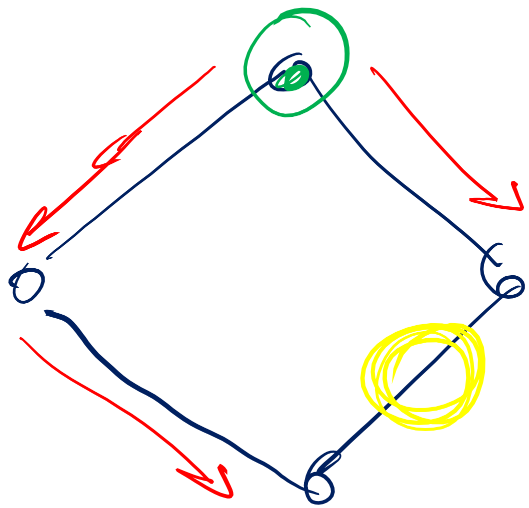


Open schedule message complexity:

- $\boldsymbol{M(n)}$: Given a ring of size n, for every asynchronous uniform leader election algorithm (and every possible assignment of IDs), there is an execution that produces an open schedule in which at least $M(n)$ messages have been received.
  – We will show that $M(n) = \Omega(n \cdot \log n)$ (by induction on n).

Open Schedule: Base Case

Lemma: Consider a cycle with $n = 4$ nodes. We can create an open schedule in which at least 3 messages are received.

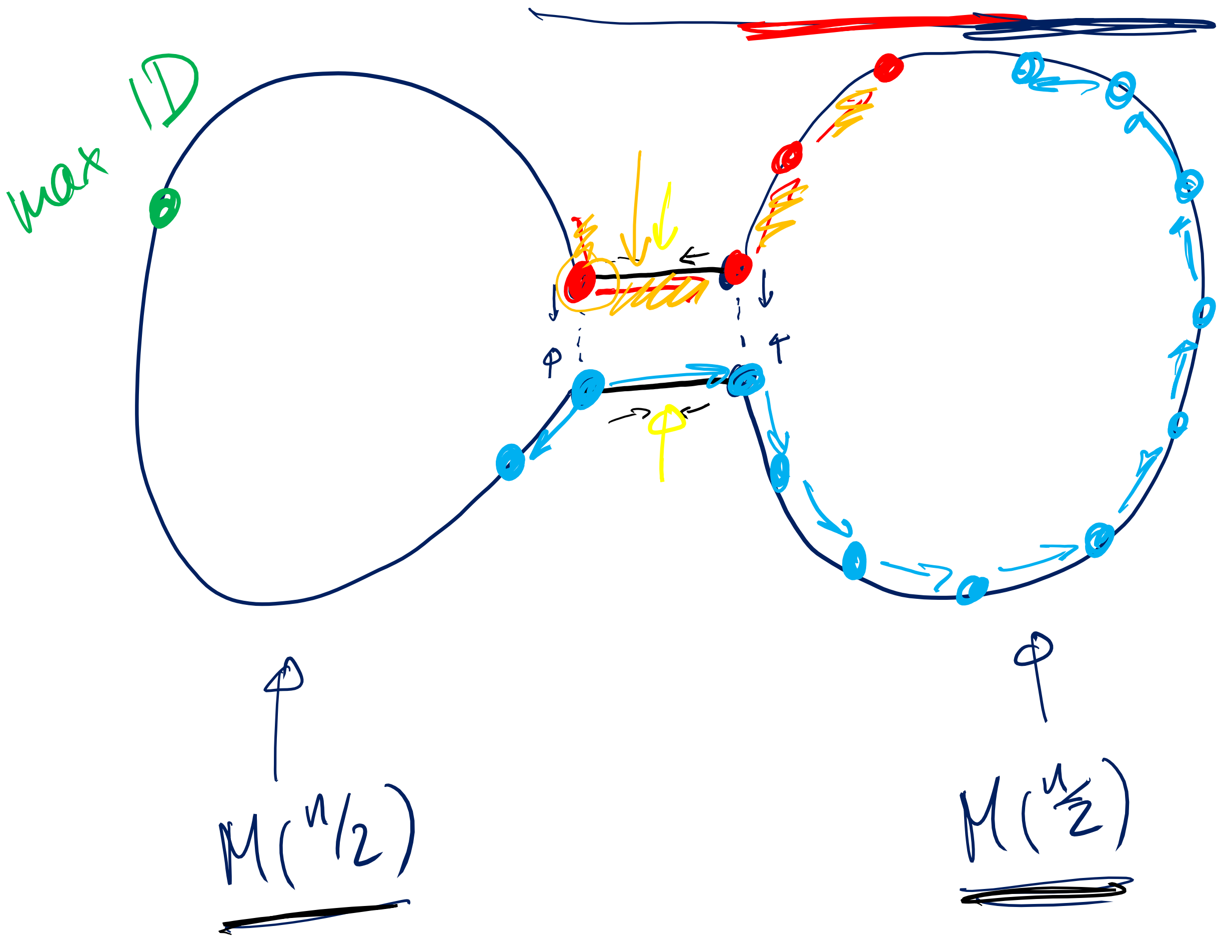


$M(4) \geq 3$

Open Schedule: Induction Step

Lemma: For $n = 2^k$ and integer $k \geq 3$, we have

$$M(n) \geq 2 \cdot M(n/2) + n/4.$$



observation

#red + #blue $\geq \frac{n}{2}$

↳ $\geq \frac{n}{4}$ red msg.

or $\geq \frac{n}{4}$ blue msg.

Open Schedule: Induction Step

Lemma: For $n = 2^k$ and integer $k \geq 3$, we have

$$M(n) \geq 2 \cdot M\left({}^{n}\!/_{2}\right) + {}^{n}\!/_{4}.$$

Message Complexity Lower Bound

Theorem: Any uniform leader election algorithm in uniform rings of size n ($n = 2^k$ for $k \geq 2$) has message complexity at least

$$M(n) \geq {}^{n}/_{4} \cdot (\log n + 1) = \Omega(n \log n).$$

Leader Election in Synchronous Rings

- Can we improve the message complexity for synchronous rings?
  - Assume that the algorithm is non-uniform (n is known)
  - Assume IDs are positive integers from $\{1, \dots, N\}$

Synchronous Leader Election Algorithm

- Algorithm consists of phases $i = 1,2, \dots$ of length n
- Every node v does the following

if phase $i = \mathrm{ID}(v)$ and v has not yet received a message **then**

    v becomes the leader

    v sends message "v is leader" arounds the ring

Leader Election in Synchronous Rings

Synchronous Leader Election Algorithm

- Algorithm consists of phases $i = 1,2, \ldots$ of length n
- Every node v does the following

if phase $i = \mathrm{ID}(v)$ and v has not yet received a message **then**
    v becomes the leader
    v sends message "v is leader" around the ring