Motivation

Leader Election

Nodes in network agree on *exactly one* leader.
All other nodes are followers.

Reasons for electing a leader?

Reasons for *not* electing a leader?

Motivation

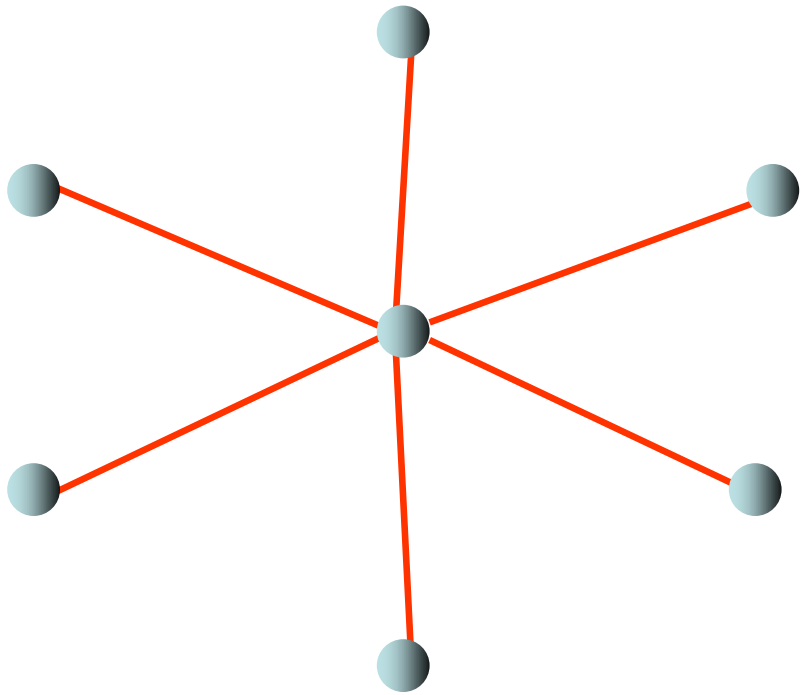
Reasons for electing a leader?

- Once elected, coordination tasks may become simpler
- For example: wireless medium access (break symmetry)

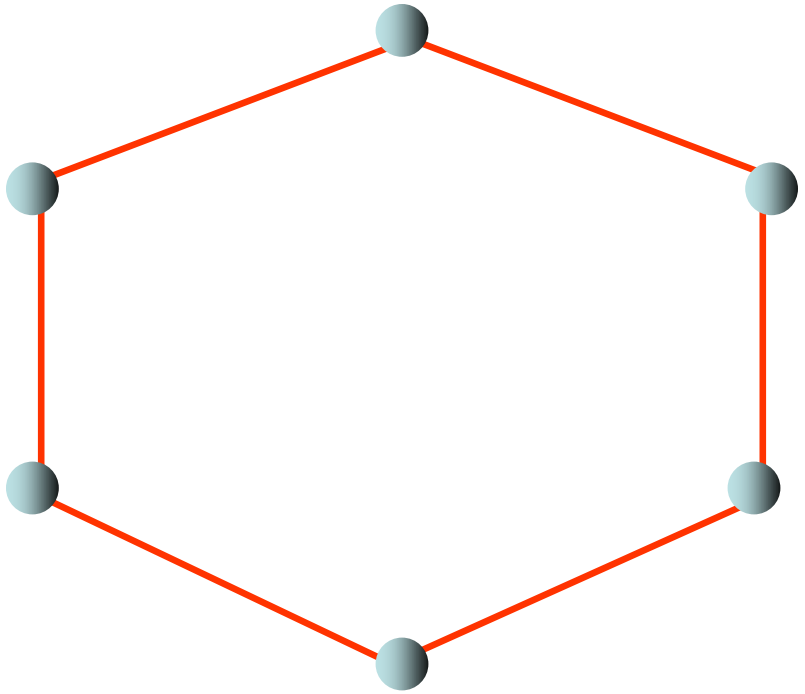
Reasons for *not* electing a leader?

- Reduced parallelism?
- Self-stabilization needed: re-election when leader „dies“
- Leader bottleneck / single point of failure?

How to elect a leader in a star?



And in a ring?



What is the basic problem?

Symmetry… and how could it be broken?

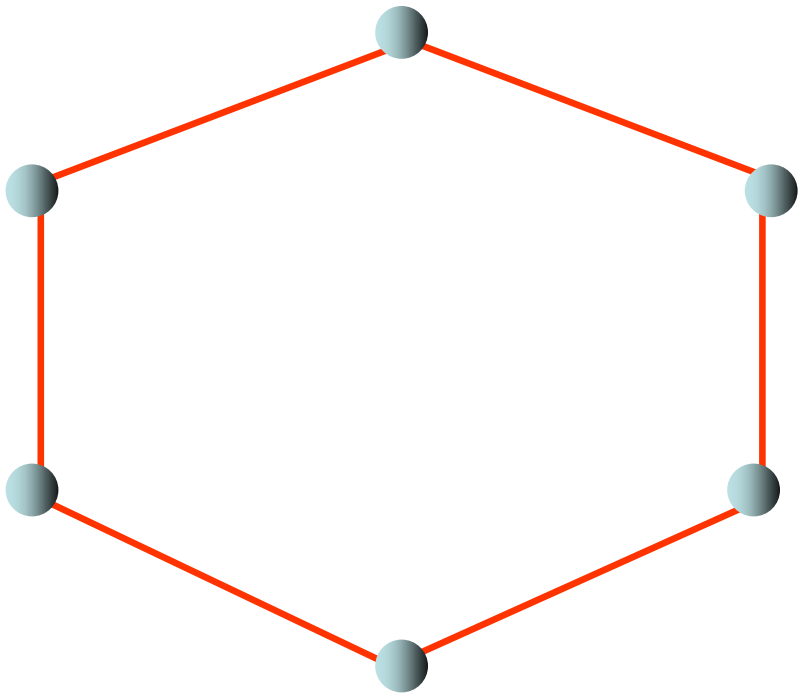
Asynchronous Ring

Uniform System

Nodes do not know n.

Let‘s assume:

- non-anonymous nodes with unique IDs
- asynchronous ring (asyn start and transmissions)
- uniform ring: n unknown!
- no message losses etc.



How to elect a leader now?

Chang-Roberts algorithm

Consider a directed ring.

Initially only *initiators* are active, and send a message with their id.

Let an *active* process p receive a message q:

- If $q < p$, then p *dismisses* the message.
- If $q > p$, then p becomes passive, and *passes on* the message.
- If $q = p$, then p becomes the leader.

Passive processes (including all noninitiators) pass on messages.

Worst-case message complexity: $O(N^2)$

Average-case message complexity: $O(N \log N)$

Chang-Roberts algorithm

Consider a directed ring.

Initially only *initiators* are active, and send a message with their id.

Let an *active* process p receive a message q:

- If $q < p$, then p *dismisses* the message.
- If $q > p$, then p becomes passive, and *passes on* the message.
- If $q = p$, then p becomes the leader.

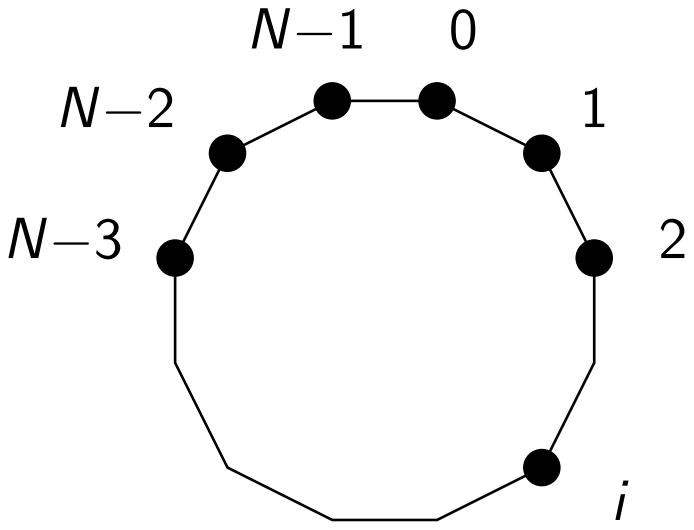
Passive processes (including all noninitiators) pass on messages.

Worst-case message complexity: $O(N^2)$

Average-case message complexity: $O(N \log N)$

Chang-Roberts algorithm - Example

All processes are initiators.



anti-clockwise: $\frac{N(N+1)}{2}$ messages

clockwise: $2N-1$ messages

Franklin's algorithm

Consider an undirected ring.

Each *active* process p repeatedly compares its own id with the id's of its nearest *active* neighbors on both sides.

If such a neighbor has a larger id, then p becomes *passive*.

Initially, initiators are active, and noninitiators are passive.

Each round, an active process p:

- sends its id to its neighbors on either side, and
- receives id's q and r:
  - if $\max\{q, r\} < p$, then p starts another round
  - if $\max\{q, r\} > p$, then p becomes passive
  - if $\max\{q, r\} = p$, then p becomes the leader

Passive processes pass on incoming messages.

Franklin's algorithm

Consider an undirected ring.

Each *active* process p repeatedly compares its own id with the id's of its nearest *active* neighbors on both sides.

If such a neighbor has a larger id, then p becomes *passive*.

Initially, initiators are active, and noninitiators are passive.

Each round, an active process p:

- sends its id to its neighbors on either side, and
- receives id's q and r:
  - if $\max\{q, r\} < p$, then p starts another round
  - if $\max\{q, r\} > p$, then p becomes passive
  - if $\max\{q, r\} = p$, then p becomes the leader

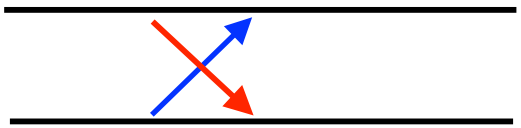
Passive processes pass on incoming messages.

Franklin's algorithm - Complexity

Worst-case message complexity: $O(N \log N)$

In each round, at least half of the active processes become passive.

So there are at most $\lfloor \log_2 N \rfloor + 1$ rounds.



Each round takes $2N$ messages.

Question: Give an example with $N = 4$ that takes three rounds.

Question: Show that for any N there is a ring that takes two rounds.

Franklin's algorithm - Complexity

Worst-case message complexity: $O(N \log N)$

In each round, at least half of the active processes become passive.

So there are at most $\lfloor \log_2 N \rfloor + 1$ rounds.

Each round takes $2N$ messages.

Question: Give an example with $N = 4$ that takes three rounds.

Question: Show that for any N there is a ring that takes two rounds.

Franklin's algorithm - Complexity

Worst-case message complexity: $O(N \log N)$

In each round, at least half of the active processes become passive.

So there are at most $\lfloor \log_2 N \rfloor + 1$ rounds.

Each round takes $2N$ messages.

Question: Give an example with $N = 4$ that takes three rounds.

Question: Show that for any N there is a ring that takes two rounds.

Question

How can Franklin's algorithm be adapted to a directed ring ?

Dolev-Klawe-Rodeh algorithm

Consider a directed ring.

The comparison of id's of an active process p and its nearest active neighbors q and r is performed at r.

$$\cdots \blacktriangleright s \cdots \blacktriangleright q \cdots \blacktriangleright p \cdots \blacktriangleright r \cdots \blacktriangleright t \cdots \blacktriangleright$$

- If $\max\{q, r\} < p$, then r changes its id to p, and sends out p.
- If $\max\{q, r\} > p$, then r becomes passive.
- If $\max\{q, r\} = p$, then r announces this id to all processes.

  The process that originally had the id p becomes the leader.

Worst-case message complexity: $O(N \log N)$

Dolev-Klawe-Rodeh algorithm

Consider a directed ring.

The comparison of id's of an active process p and
its nearest active neighbors q and r is performed at r.

$$\cdots \blacktriangleright s \cdots \blacktriangleright q \cdots \blacktriangleright p \cdots \blacktriangleright r \cdots \blacktriangleright t \cdots \blacktriangleright$$

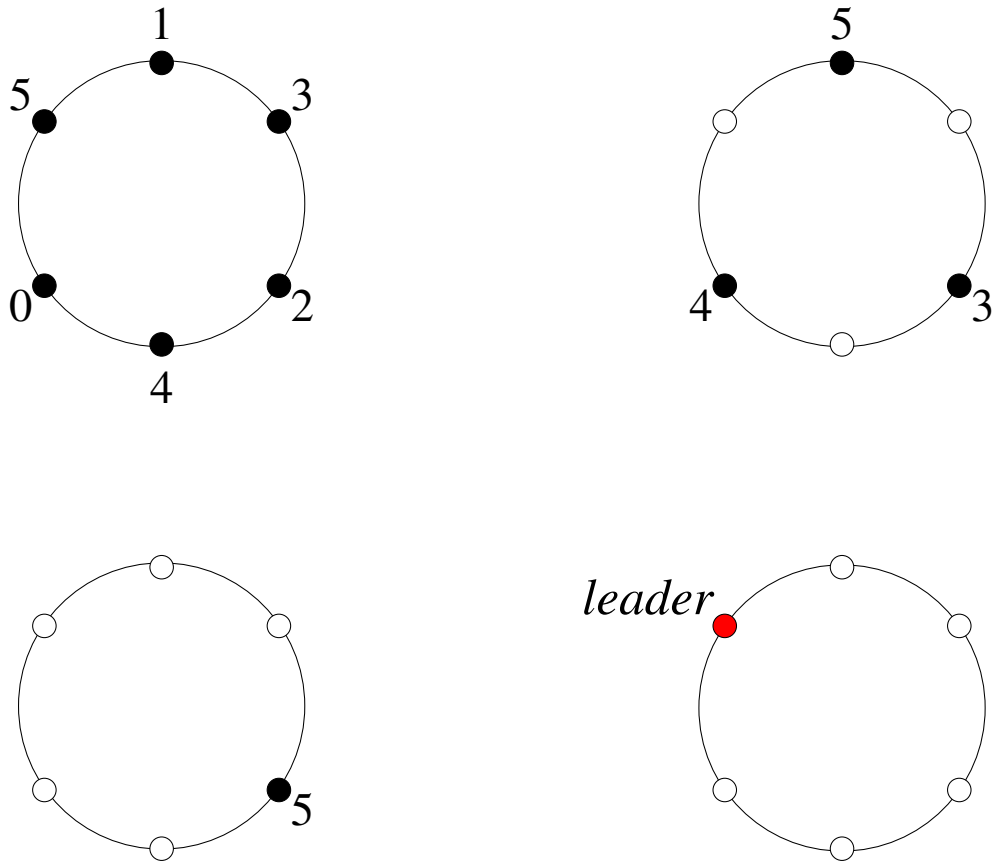
- If $\max\{q, r\} < p$, then r changes its id to p, and sends out p.
- If $\max\{q, r\} > p$, then r becomes passive.
- If $\max\{q, r\} = p$, then r announces this id to all processes.

  The process that originally had the id p becomes the leader.

Worst-case message complexity: $O(N \log N)$

Dolev-Klawe-Rodeh algorithm - Example

Consider the following clockwise oriented ring.



Dolev-Klawe-Rodeh algorithm

Consider a directed ring.

The comparison of id's of an active process p and
its nearest active neighbors q and r is performed at r.

$$\cdots \blacktriangleright s \cdots \blacktriangleright q \cdots \blacktriangleright p \cdots \blacktriangleright r \cdots \blacktriangleright t \cdots \blacktriangleright$$

- If $\max\{q, r\} < p$, then r changes its id to p, and sends out p.
- If $\max\{q, r\} > p$, then r becomes passive.
- If $\max\{q, r\} = p$, then r announces this id to all processes.

  The process that originally had the id p becomes the leader.

Worst-case message complexity: $O(N \log N)$

Average-case lower bound: $\Omega(N \log N)$

Lower Bound

Assumptions:

- Asynchronous ring: nodes wake up at arbitrary times but always when receiving a packet
- nodes have IDs, and node with max ID should become leader (strong assumption?)
- every node must know ID of leader
- uniform algorithm: n is not known
- arbitrary scheduler but links are FIFO

For our lower bound proof, we define the concept of *open schedules*:

Open Schedule

Schedule chosen by scheduler. Open if there is an open edge in the ring. Edge is *open* if no message traversing this edge has been received so far.

Note: any leader election algorithm must send over each edge at some point!
Otherwise whole network could be hidden behind it.

Some Intuition...

Open Schedule

Schedule chosen by scheduler. Open if there is an open edge in the ring. Edge is *open* if no message traversing edge has been received so far.

Intuition: Open schedule = endpoints have not heard anything from nodes on this edge, protocol *cannot stop* yet as it may hide critical infos on the leader!

We want to show that there exists a *bad schedule* which requires lots of messages until a leader is elected. To achieve this, we play god and compute a bad open schedule inductively (looking into the future, where many messages will be sent!).

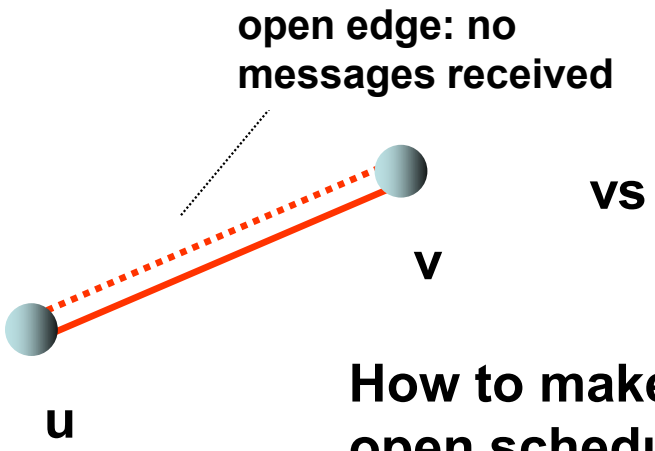
Lower Bound by Induction

Proof by induction:

Lemma: 2-node Ring

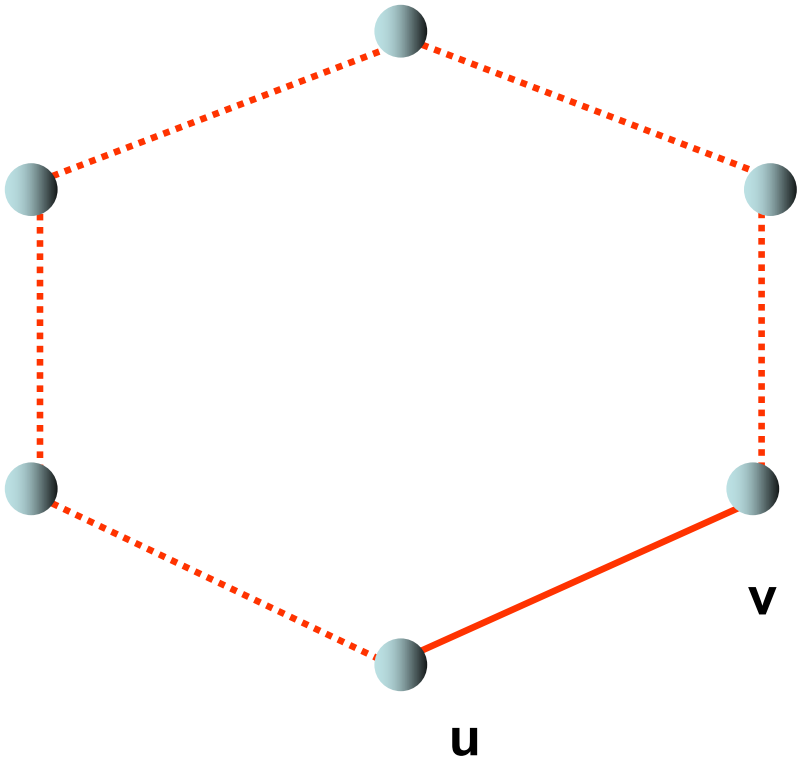
Given a ring R with two nodes, we can construct an open schedule in which at least one message is received. The nodes cannot distinguish this schedule from one on a larger ring with all other nodes being located where the open edge is.

Proof of Lemma: u and v cannot distinguish between the two scenarios!



vs

How to make an open schedule?



Proof of Lemma: Open Schedule

Lemma: 2-node Ring

Given a ring R with two nodes, we can construct an open schedule in which at least one message is received. The nodes cannot distinguish this schedule from one on a larger ring with all other nodes being where the open edge is.

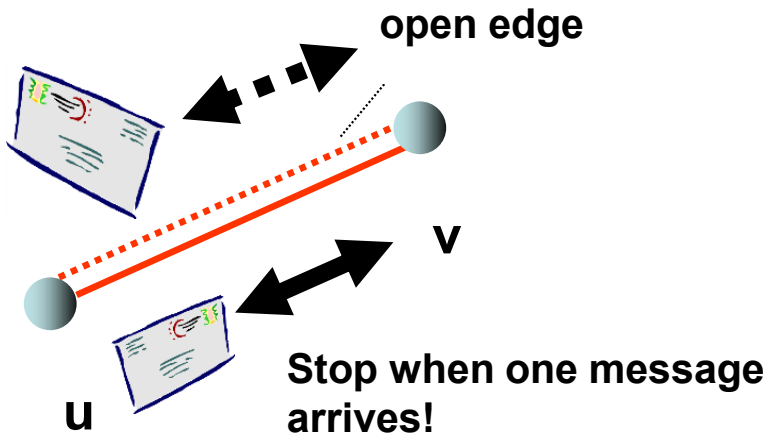
Open schedule for 2-node ring?

In any leader election algorithm, the two nodes must learn about each other! We stop execution when first message is received (on whatever link).

We can do this because it's an asynchronous world (no simultaneous arrivals, delay accordingly)...

So other edge is open:

Nodes don't know, is it an edge, or is it more?

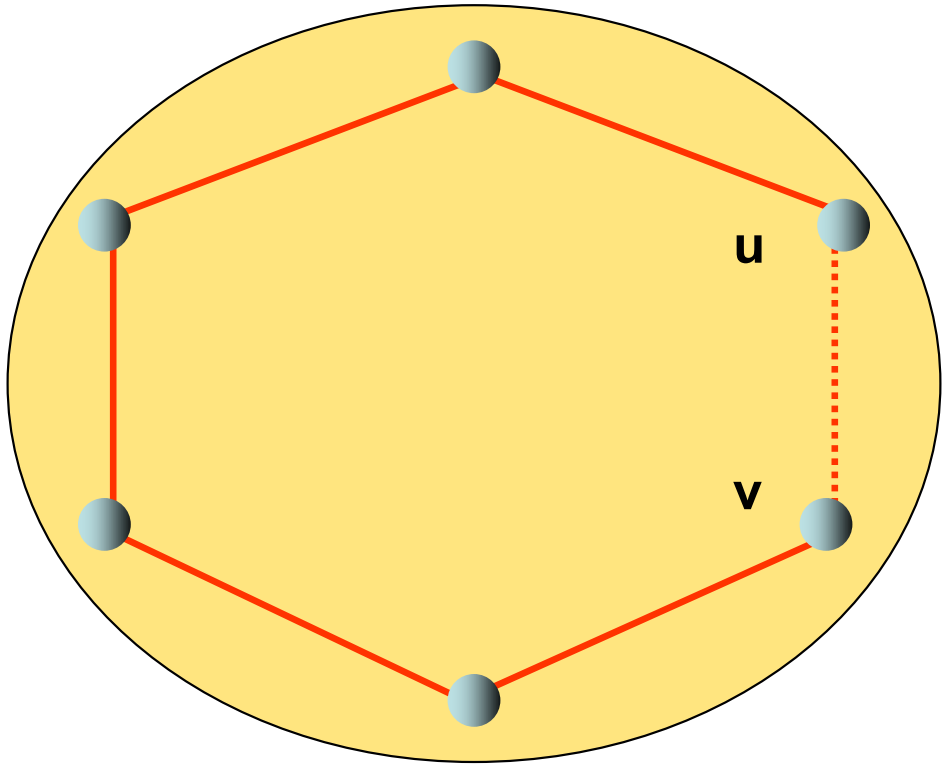


Open Schedules for Larger Rings?

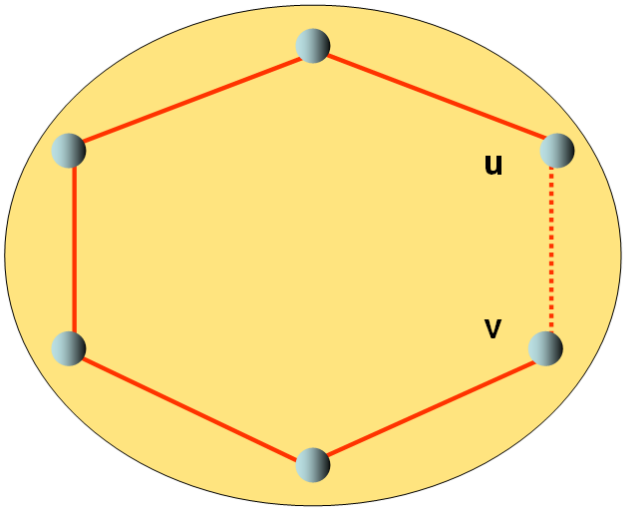
n-node Ring

By gluing together (at the two open edges) two rings of size n/2 for which we have open schedules, an open schedule can be constructed on a ring of size n. Let M(n/2) denote the number of messages used in each of these schedules by some algorithm ALG. Then, in the entire ring 2M(n/2)+n/4 messages have to be exchanged to solve leader election.

Proof? Open schedule?

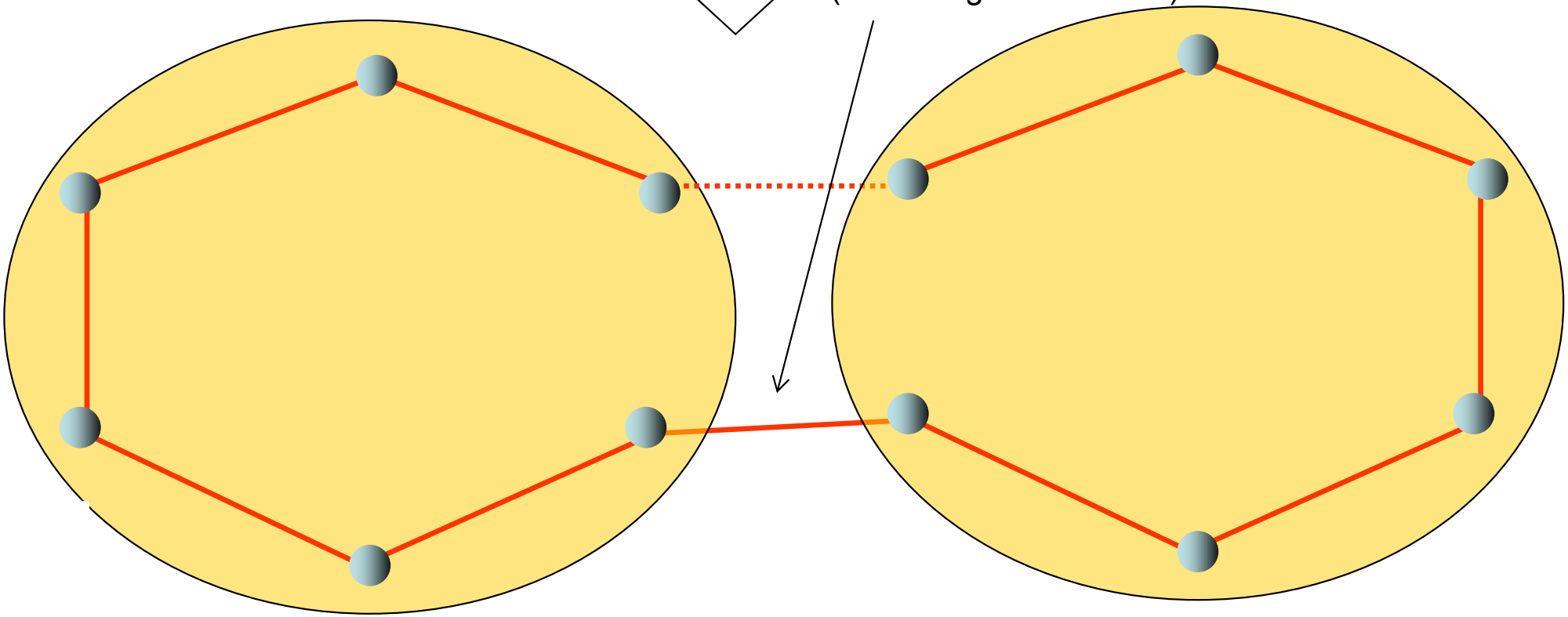


Assume ALG needs
M(n/2) messages here:



Idea: glue together at open edge;
before closing an edge
rings cannot distinguish whether n/2-
or n node-ring!

When I close one of the edges
at least n/4 message receptions are
triggered! And schedule still open.
(Other edge unaffected.)

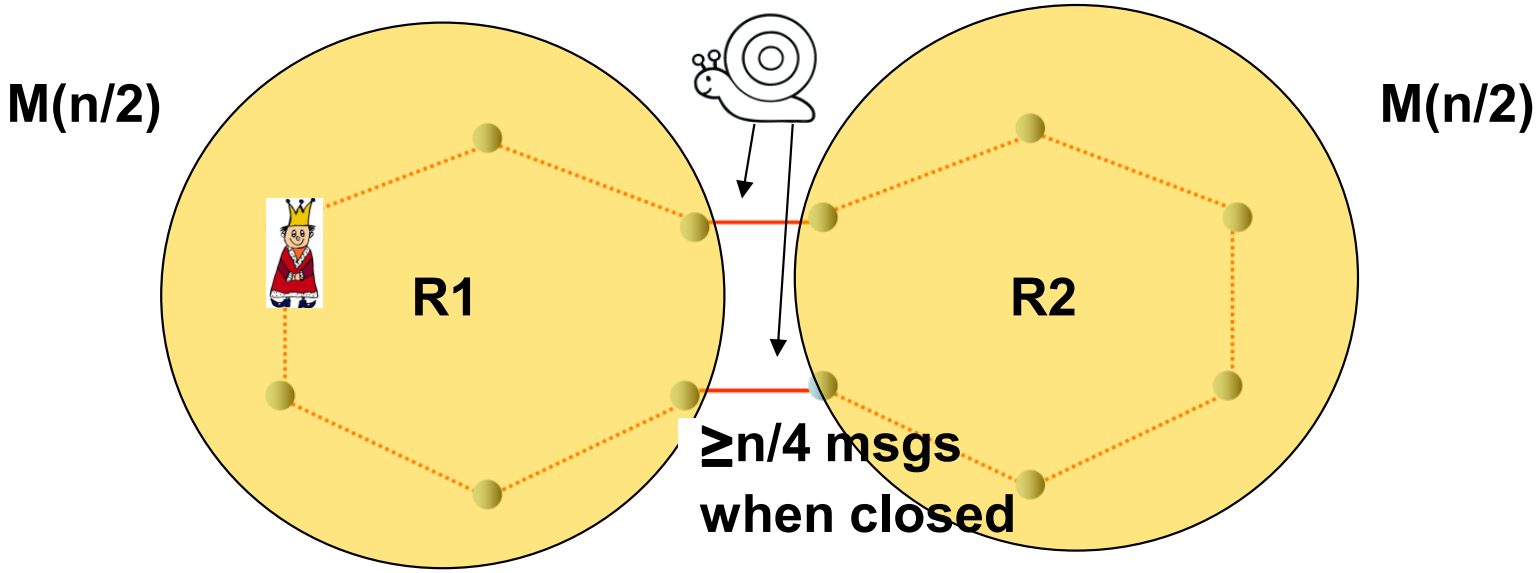


Proof of Lemma: By Induction

- Consider the ring of size n and divide it in two „subrings“ R1 and R2. As long as no message comes from outside, nodes cannot distinguish these two rings from two rings of size n/2. (Just delay messages accordingly: all other messages of algorithm are sent.)

- So nodes exchange 2*M(n/2) messages (induction hypothesis) in the subrings before learning anything about the other subring. Wlog assume R1 has max ID. So each node in R2 must learn that ID, which requires at least n/2 message receptions.

- So there must be an edge connecting the two rings that „produces“ (= triggers, but not necessarily transmits!) at least n/4 messages.
  Schedule/close this edge and leave other open... => open schedule for larger ring! And enough messages! ☺



How to Construct an Open Schedule?

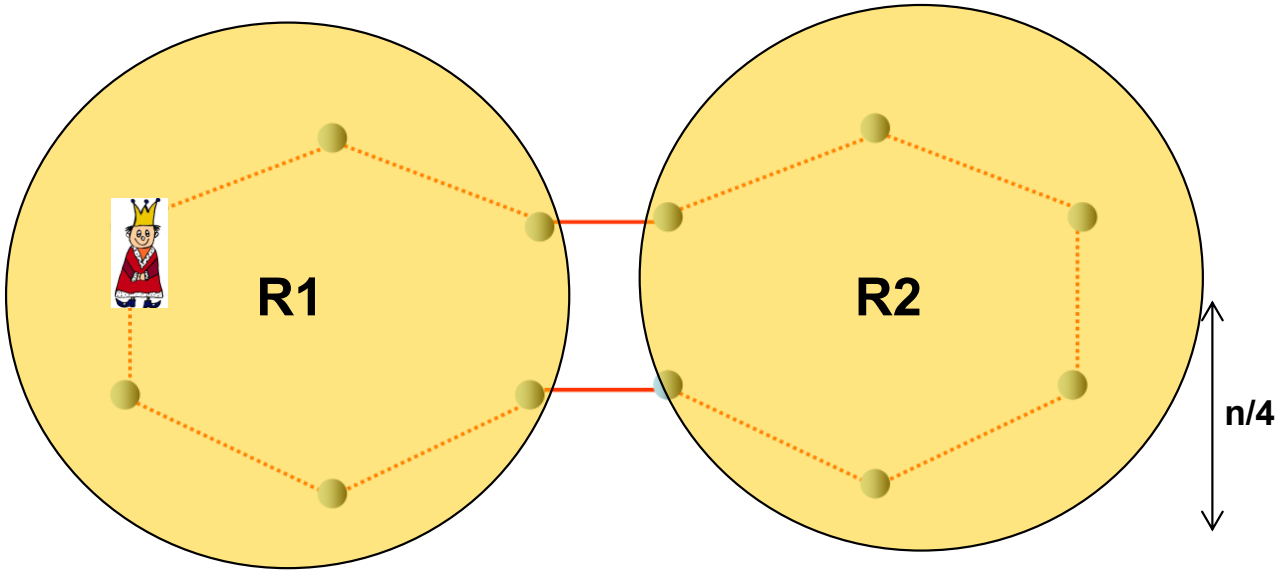
Take-Away

Just let asynchronous algorithm run and stop before last edge is closed (i.e., before message arrives).

Why ≥ n/4 messages triggered by border edge even if schedule is made open?

1. **Maybe this is whole ring: so much information must be transferred eventually!**

2. **Fact independent of schedule: learning about events / timing of other edges requires n/4 messages at least as well!**



Open Schedules for Larger Rings?

Theorem

Any algo needs at least Ω(n log n) messages.

Proof by induction: Claim follows from maths...

$$\begin{aligned} M(n) &= 2 \cdot M\left(\frac{n}{2}\right) + \frac{n}{4} \\ &\geq 2 \cdot \left(\frac{n}{8}\left(\log \frac{n}{2} + 1\right)\right) + \frac{n}{4} \\ &= \frac{n}{4}\log n + \frac{n}{4} = \frac{n}{4}\left(\log n + 1\right) \end{aligned}$$

So we are optimal.

Can we do better? ☺

Breaking the Lower Bound ☺

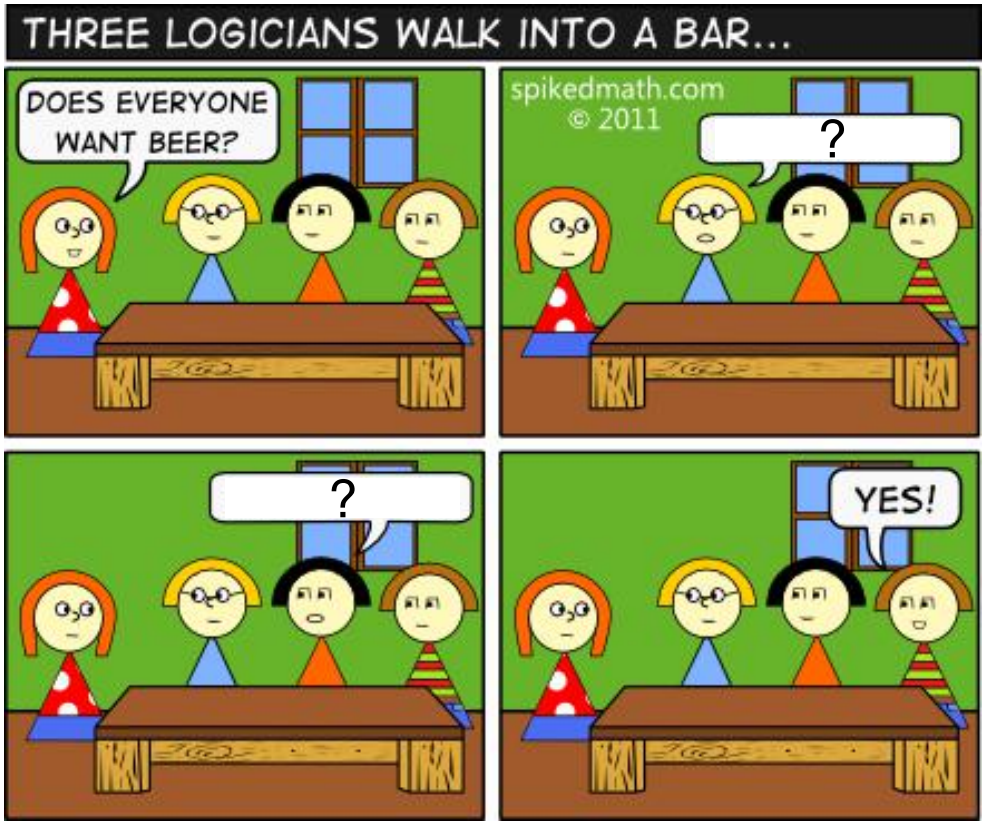
Take-Away

In synchronous systems, not receiving a message is also information!

Breaking the Lower Bound ☺

Take-Away

In synchronous systems, not receiving a message is also information!



Breaking the Lower Bound ☺

Take-Away

In synchronous systems, not receiving a message is also information!

Idea for message complexity n? E.g., find minimum ID in environment where nodes have unique but arbitrary integer IDs (but n known)...

Sync Leader Election

each node v does the following:

- Divide time into phases of n steps (leaves time for lower-ID nodes to broadcast...)
- If phase = v and did not get a message:
  - v becomes leader
  - v sends „I am leader!“ to everybody!

Breaks message lower bound but we may wait long!
Runtime O(n*minID)? What is the time – message tradeoff?

Question

How can the echo algorithm be used to get an election algorithm for any undirected network ?

Echo algorithm with extinction

Each *initiator* starts a wave, tagged with its id.

Non-initiators join the first wave that hits them.

At any time, each process takes part in at most one wave.

Suppose a process p in wave q is hit by a wave r:

- if $q < r$, then p changes to wave r
  (it abandons all earlier messages);
- if $q > r$, then p continues with wave q
  (it dismisses the incoming message);
- if $q = r$, then the incoming message is treated according to the echo algorithm of wave q.

If wave p executes a decide event (at p), p becomes the leader.

Worst-case message complexity: $O(N \cdot E)$

Echo algorithm with extinction

Each *initiator* starts a wave, tagged with its id.

Non-initiators join the first wave that hits them.

At any time, each process takes part in at most one wave.

Suppose a process p in wave q is hit by a wave r:

- if $q < r$, then p changes to wave r
  (it abandons all earlier messages);
- if $q > r$, then p continues with wave q
  (it dismisses the incoming message);
- if $q = r$, then the incoming message is treated according to the echo algorithm of wave q.

If wave p executes a decide event (at p), p becomes the leader.

Worst-case message complexity: $O(N \cdot E)$

Echo algorithm with extinction

Each *initiator* starts a wave, tagged with its id.

Non-initiators join the first wave that hits them.

At any time, each process takes part in at most one wave.

Suppose a process p in wave q is hit by a wave r:

- if $q < r$, then p changes to wave r
  (it abandons all earlier messages);
- if $q > r$, then p continues with wave q
  (it dismisses the incoming message);
- if $q = r$, then the incoming message is treated according to the echo algorithm of wave q.

If wave p executes a decide event (at p), p becomes the leader.

Worst-case message complexity: $O(N \cdot E)$

Echo algorithm with extinction

Each *initiator* starts a wave, tagged with its id.

Non-initiators join the first wave that hits them.

At any time, each process takes part in at most one wave.

Suppose a process p in wave q is hit by a wave r:

- if $q < r$, then p changes to wave r
  (it abandons all earlier messages);
- if $q > r$, then p continues with wave q
  (it dismisses the incoming message);
- if $q = r$, then the incoming message is treated according to the echo algorithm of wave q.

If wave p executes a decide event (at p), p becomes the leader.

Worst-case message complexity: $O(N \cdot E)$

Exercise:

design a leader election algorithm for
a dynamically growing network

(when a new node joins the network, it announces itself to
an arbitrary existing node, via a message tagged with its ID)

Exercise:

Consider an arbitrary network with indirect message passing
(via routing tables) and a leader election algorithm based on max ID.

Imagine a behaviour for a single “bizantine” node that enables him
to get elected as leader even when having low ID.

Question

How can the Gallager-Humblet-Spira algorithm be turned into an election algorithm for undirected networks ?

Back to election

By two extra messages at the very end,
the core node with the largest id becomes the leader.

So Gallager-Humblet-Spira induces an election algorithm for general undirected networks.

(We must impose an order on channels of equal weight.)

Lower bounds for the average-case message complexity
of election algorithms based on comparison of id's:

Rings: $\Omega(N \log N)$

General networks: $\Omega(E + N \log N)$

Back to election

By two extra messages at the very end,
the core node with the largest id becomes the leader.

So Gallager-Humblet-Spira induces an election algorithm for general undirected networks.

(We must impose an order on channels of equal weight.)

Lower bounds for the average-case message complexity of election algorithms based on comparison of id's:

Rings: $\Omega(N \log N)$

General networks: $\Omega(E + N \log N)$