Leader Election II

Distributed Algorithms

Sebastian Forster

University of Salzburg



This work is licensed under a Creative Commons Attribution 4.0 International License.

Motivation: Leader Election

Leader Election:

- Nodes in a network agree on a *leader*
- All others are called *followers*
- **Goal:** An algorithm that decides for each node whether it is the leader or a follower
- **Motivation:** The leader can take over coordination tasks

Motivation: Leader Election

Leader Election:

- Nodes in a network agree on a *leader*
- All others are called *followers*
- **Goal:** An algorithm that decides for each node whether it is the leader or a follower
- **Motivation:** The leader can take over coordination tasks

Symmetry Breaking:

- A priori, every node is suitable as leader
- The solution is not unique; there are n different solutions
- The difficulty in finding a solution is making a consistent choice among the solutions

What Happened So Far...

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms

What Happened So Far...

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms
   Today: Randomized algorithm

What Happened So Far...

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms
   Today: Randomized algorithm
2. Clockwise algorithm:
   - Deterministic synchronous/asynchronous algorithm for uniform/non-uniform rings
   - $O(n)$ rounds, $O(n^2)$ messages

What Happened So Far...

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms
   Today: Randomized algorithm
2. Clockwise algorithm:
   - Deterministic synchronous/asynchronous algorithm for uniform/non-uniform rings
   - $O(n)$ rounds, $O(n^2)$ messages
3. Waiting-time algorithm:
   - Deterministic synchronous algorithm for non-uniform rings
   - $O(n \cdot \min_v \mathrm{ID}(v))$ rounds, $O(n)$ messages

What Happened So Far...

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms
   Today: Randomized algorithm
2. Clockwise algorithm:
   - Deterministic synchronous/asynchronous algorithm for uniform/non-uniform rings
   - $O(n)$ rounds, $O(n^2)$ messages
3. Waiting-time algorithm:
   - Deterministic synchronous algorithm for non-uniform rings
   - $O(n \cdot \min_v \mathrm{ID}(v))$ rounds, $O(n)$ messages
   - Question: Is efficiency in both measures possible?

What Happened So Far...

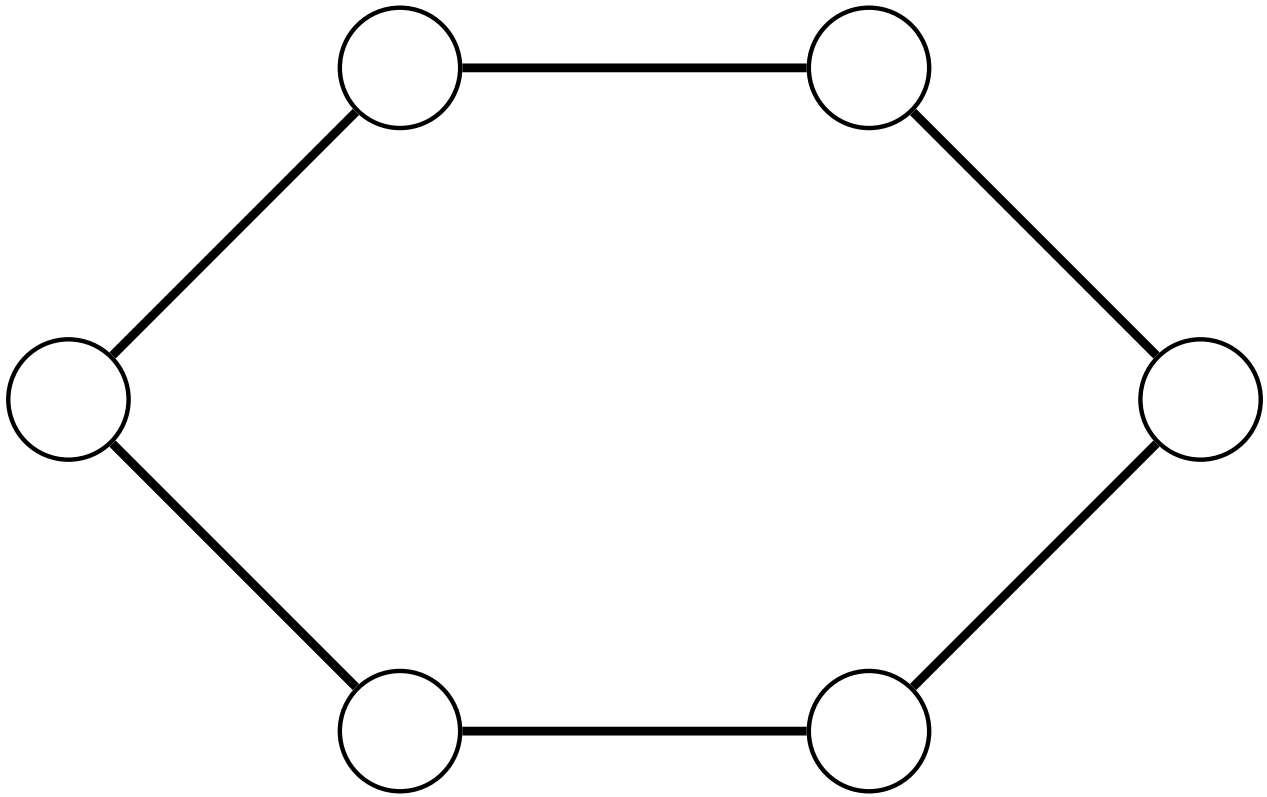
Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms
   Today: Randomized algorithm
2. Clockwise algorithm:
   - Deterministic synchronous/asynchronous algorithm for uniform/non-uniform rings
   - $O(n)$ rounds, $O(n^2)$ messages
3. Waiting-time algorithm:
   - Deterministic synchronous algorithm for non-uniform rings
   - $O(n \cdot \min_v \mathrm{ID}(v))$ rounds, $O(n)$ messages
   - Question: Is efficiency in both measures possible?

   Today: $O(n)$ rounds, $O(n \log n)$ messages

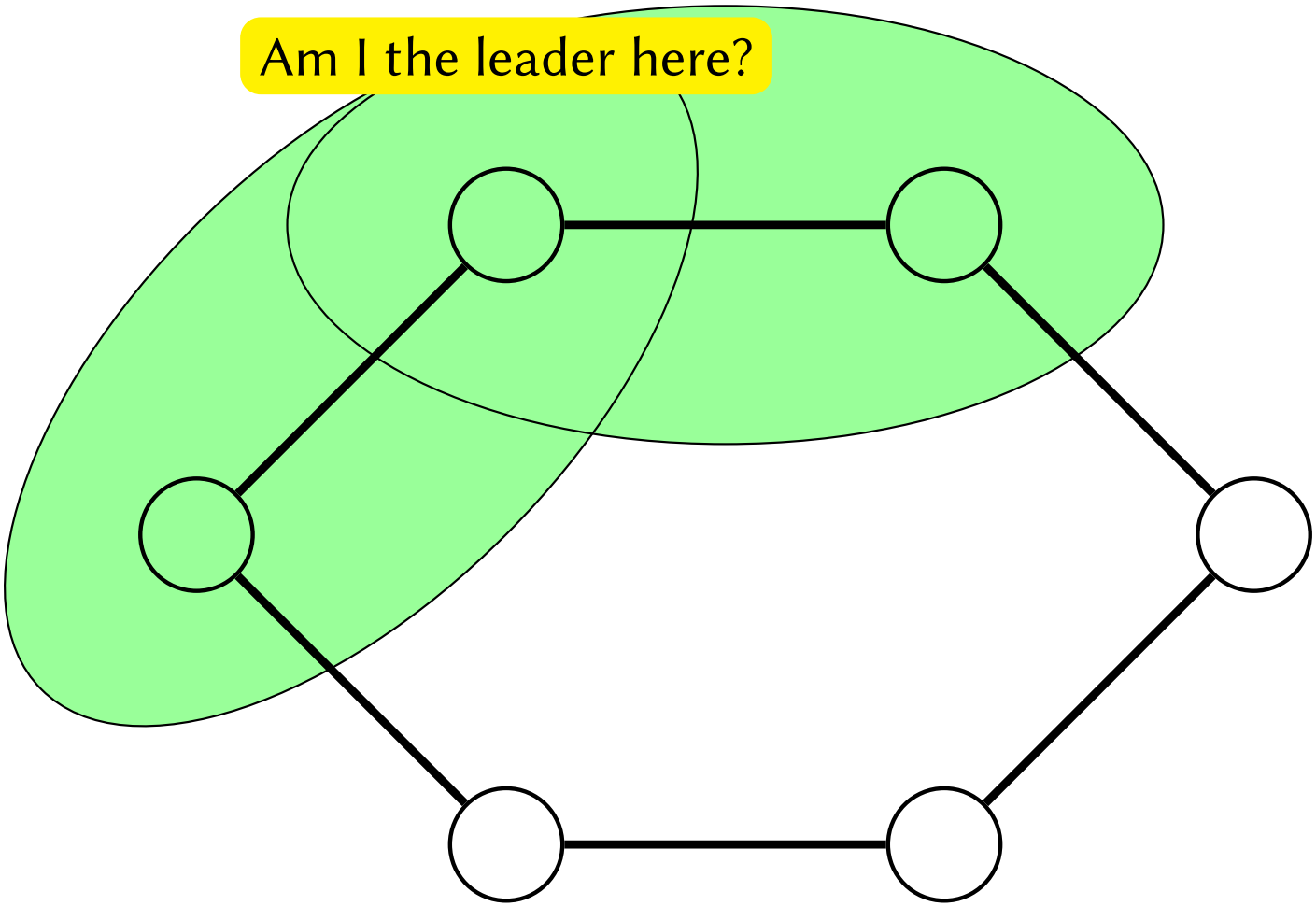
Idea: Radius Growth

Assumptions: synchronous, identifiable, non-uniform



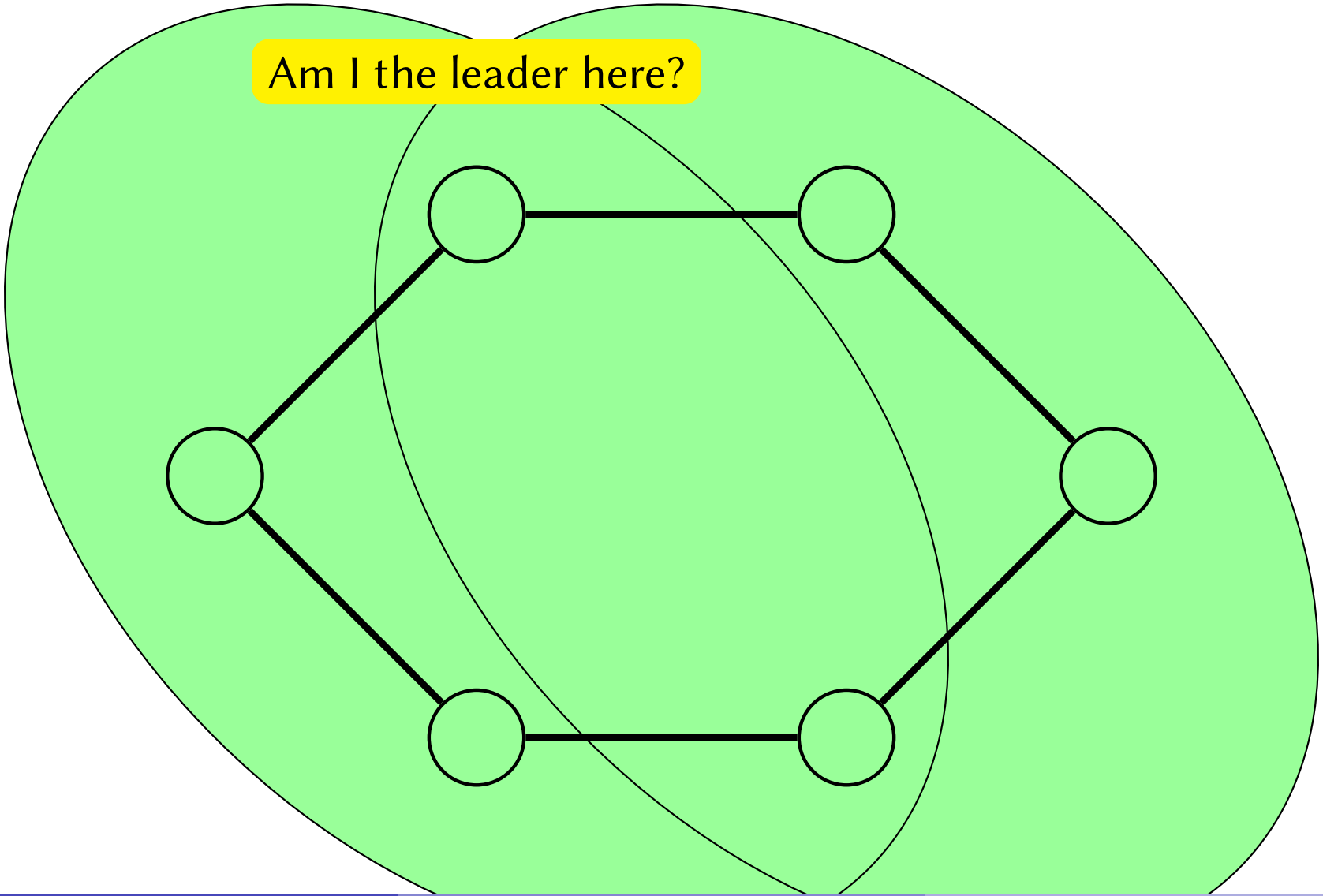
Idea: Radius Growth

Assumptions: synchronous, identifiable, non-uniform



Idea: Radius Growth

Assumptions: synchronous, identifiable, non-uniform



Radius Growth Algorithm [Hirschberg/Sinclair '80]

- Divide the rounds into $\lceil \log n \rceil$ consecutive phases
- The ith phase lasts $2^{i-1} + 1$ rounds

Radius Growth Algorithm [Hirschberg/Sinclair '80]

- Divide the rounds into $\lceil \log n \rceil$ consecutive phases
- The ith phase lasts $2^{i-1} + 1$ rounds

Algorithm for each node v:

First round of each phase:

Radius Growth Algorithm [Hirschberg/Sinclair '80]

- Divide the rounds into $\lceil \log n \rceil$ consecutive phases
- The ith phase lasts $2^{i-1} + 1$ rounds

Algorithm for each node v:

First round of each phase:

1 **if** v *is not yet a follower* **then**

2 $\quad$ v sends $\mathrm{ID}(v)$ to its clockwise and counterclockwise neighbors

Radius Growth Algorithm [Hirschberg/Sinclair '80]

- Divide the rounds into $\lceil \log n \rceil$ consecutive phases
- The ith phase lasts $2^{i-1} + 1$ rounds

Algorithm for each node v:

First round of each phase:

1 **if** v *is not yet a follower* **then**
2 $\quad$ v sends $\mathrm{ID}(v)$ to its clockwise and counterclockwise neighbors

Every other round:

1 **if** v *receives message* M *from its counterclockwise (clockwise) neighbor* **then**

Radius Growth Algorithm [Hirschberg/Sinclair '80]

- Divide the rounds into $\lceil \log n \rceil$ consecutive phases
- The ith phase lasts $2^{i-1} + 1$ rounds

Algorithm for each node v:

First round of each phase:

```
1 if v is not yet a follower then
2     v sends ID(v) to its clockwise and counterclockwise neighbors
```

Every other round:

```
1 if v receives message M from its counterclockwise (clockwise) neighbor
    then
2     if M < ID(v) then
3         v becomes a follower (if this has not already happened)
4         if not the final round of the phase then
5             v sends M to its clockwise (counterclockwise) neighbor
```

Radius Growth Algorithm [Hirschberg/Sinclair '80]

- Divide the rounds into $\lceil \log n \rceil$ consecutive phases
- The ith phase lasts $2^{i-1} + 1$ rounds

Algorithm for each node v:

First round of each phase:

```
1 if v is not yet a follower then
2     v sends ID(v) to its clockwise and counterclockwise neighbors
```

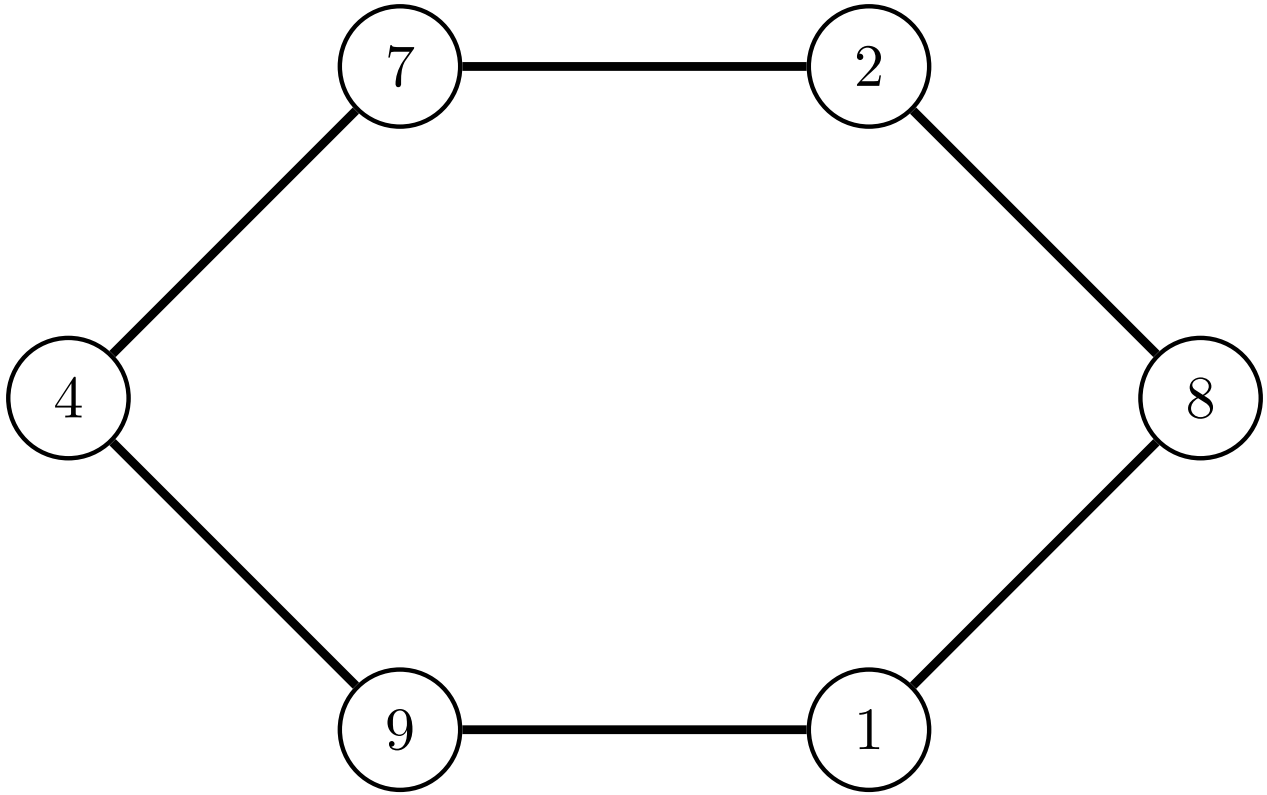
Every other round:

```
1 if v receives message M from its counterclockwise (clockwise) neighbor
    then
2     if M < ID(v) then
3         v becomes a follower (if this has not already happened)
4         if not the final round of the phase then
5             v sends M to its clockwise (counterclockwise) neighbor
```

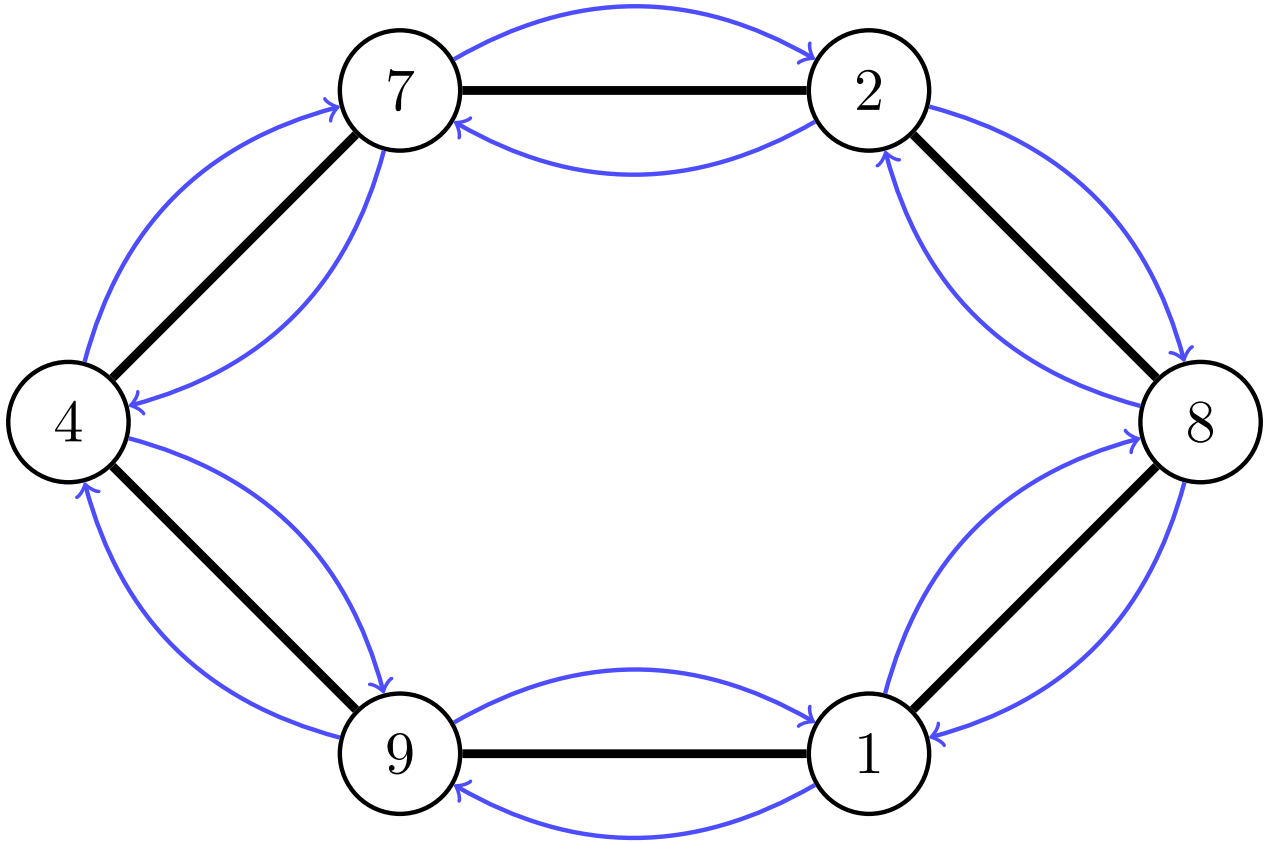
Additionally, in the final round of the final phase:

```
1 if v is not yet a follower then v becomes the leader
```

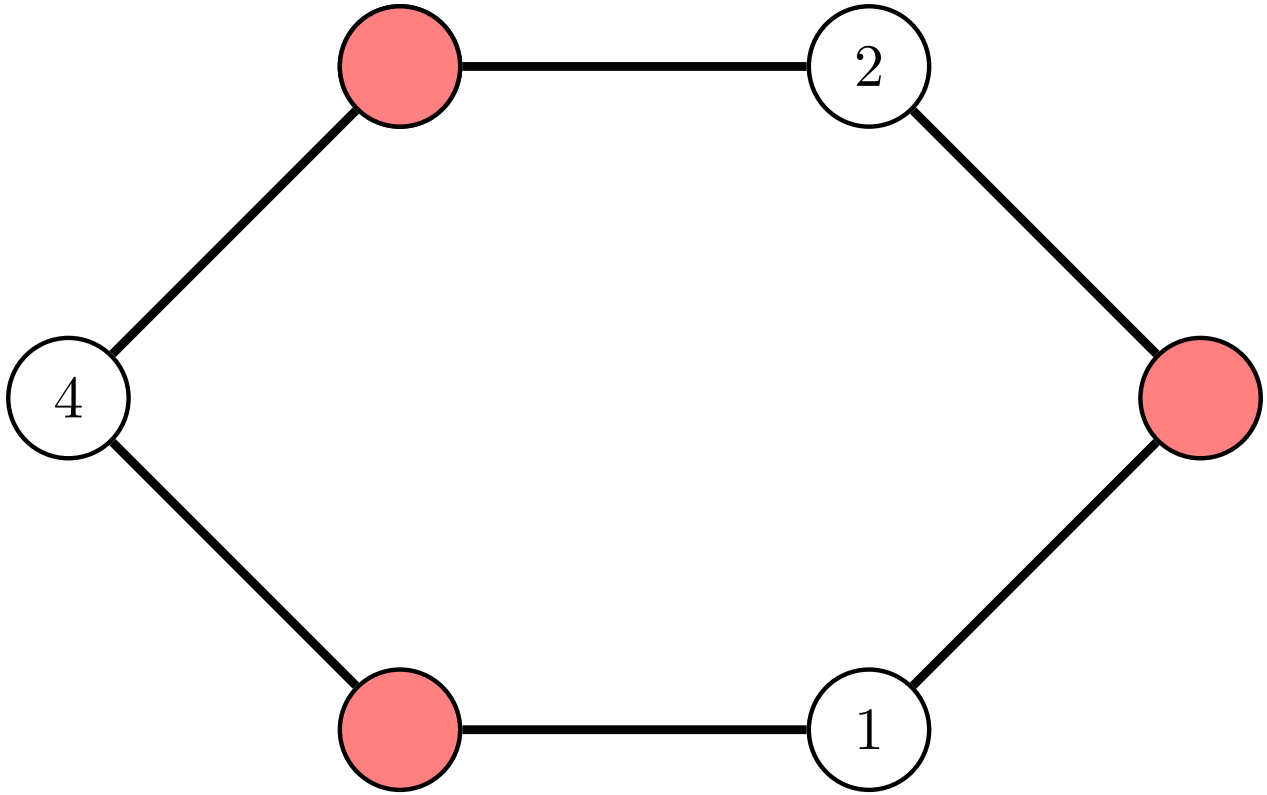
Example



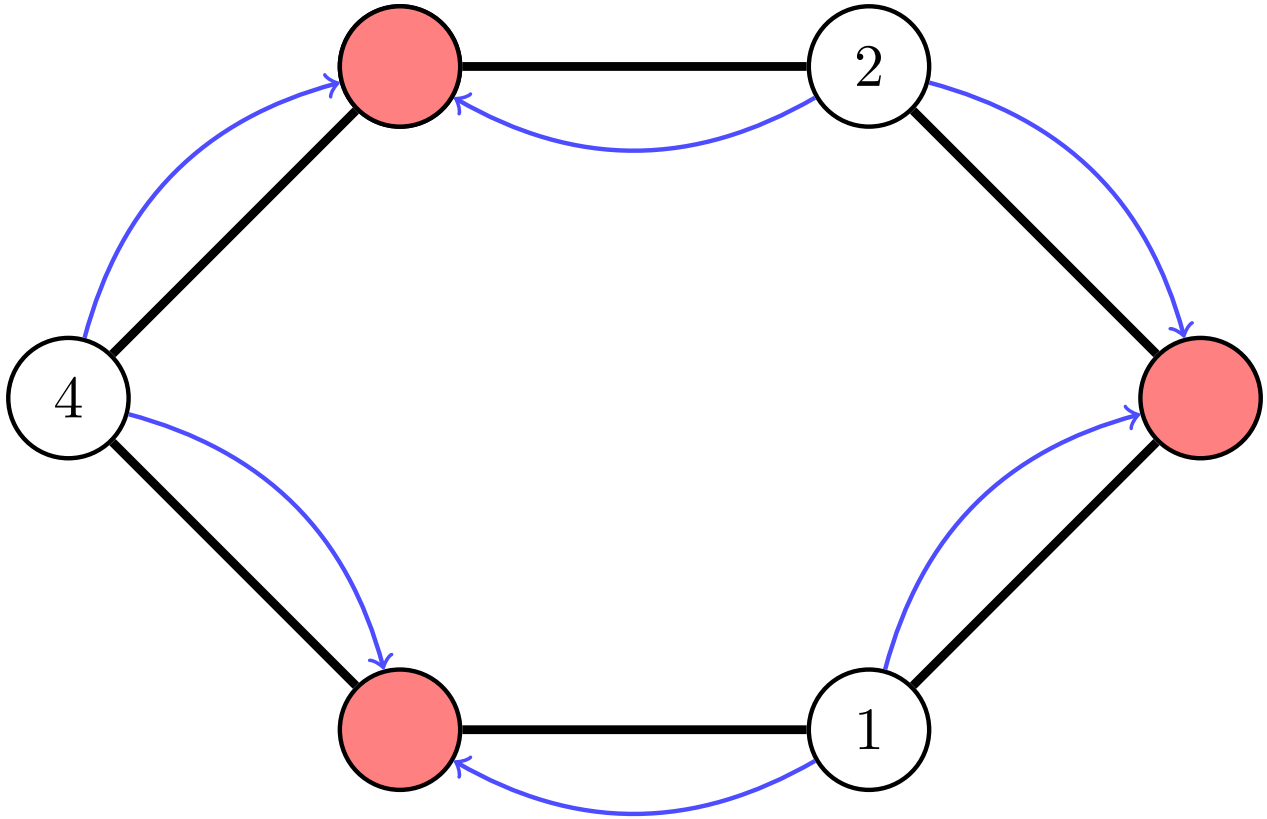
Example



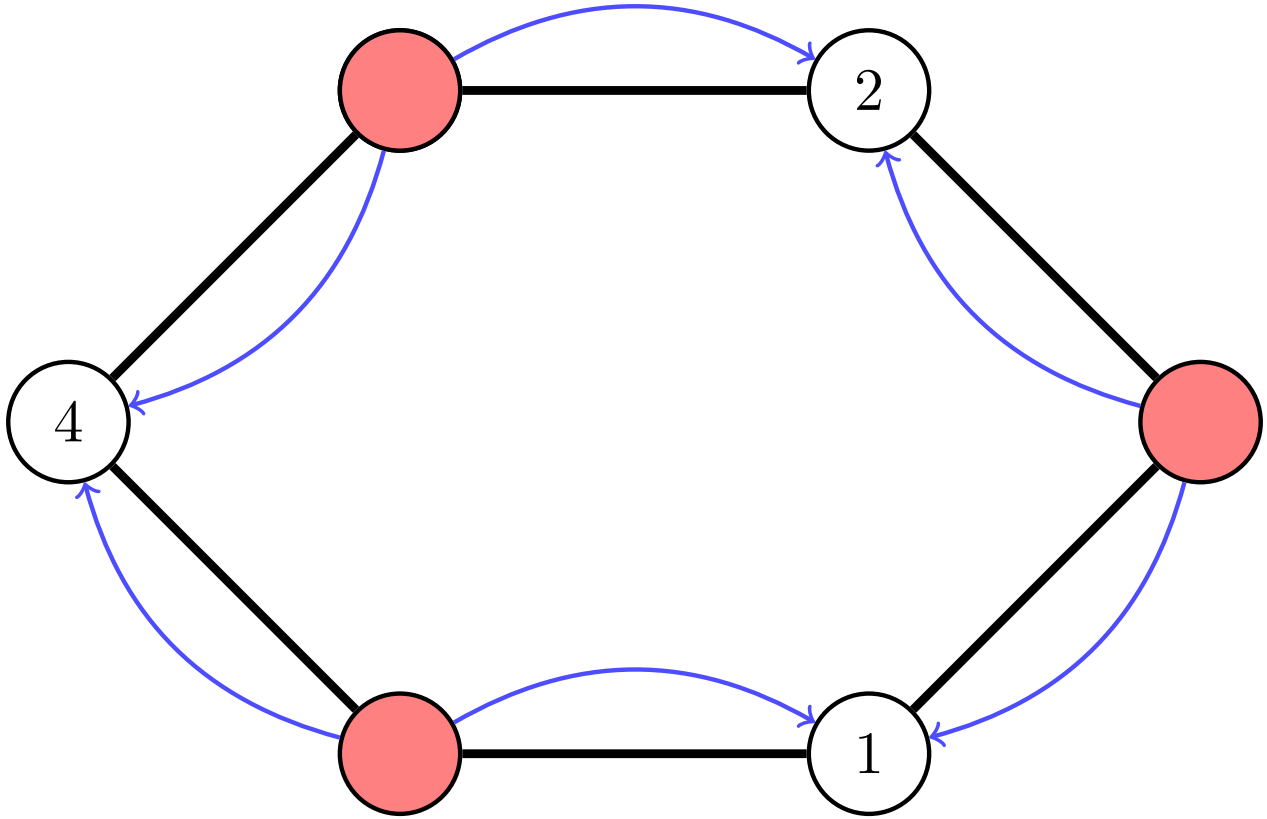
Example



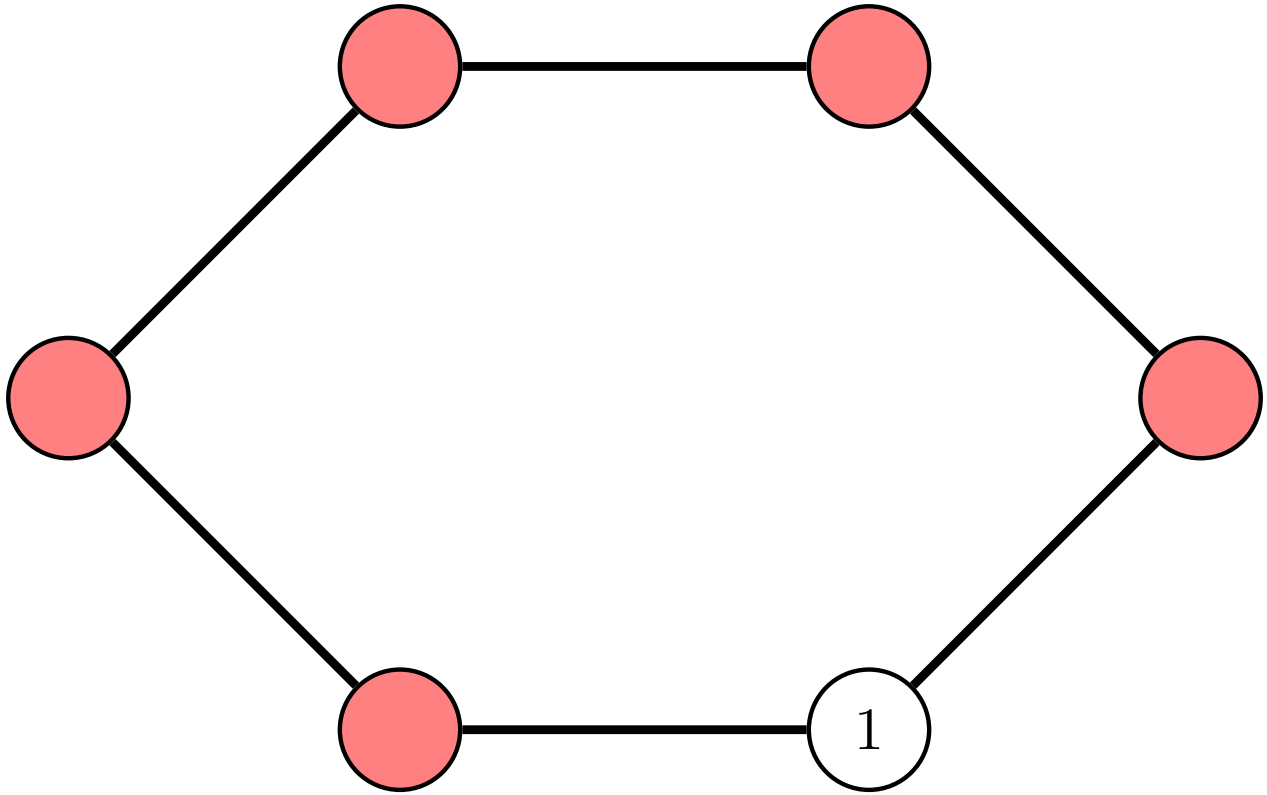
Example



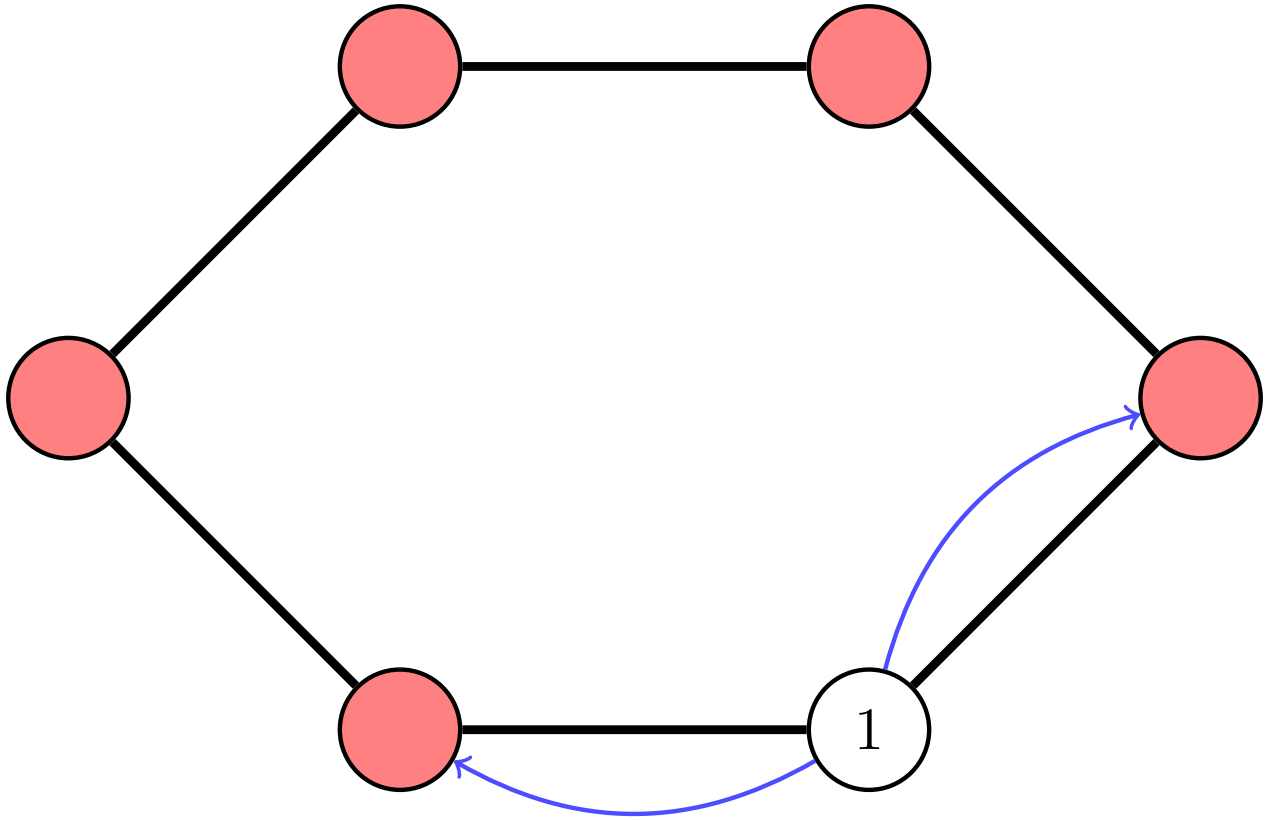
Example



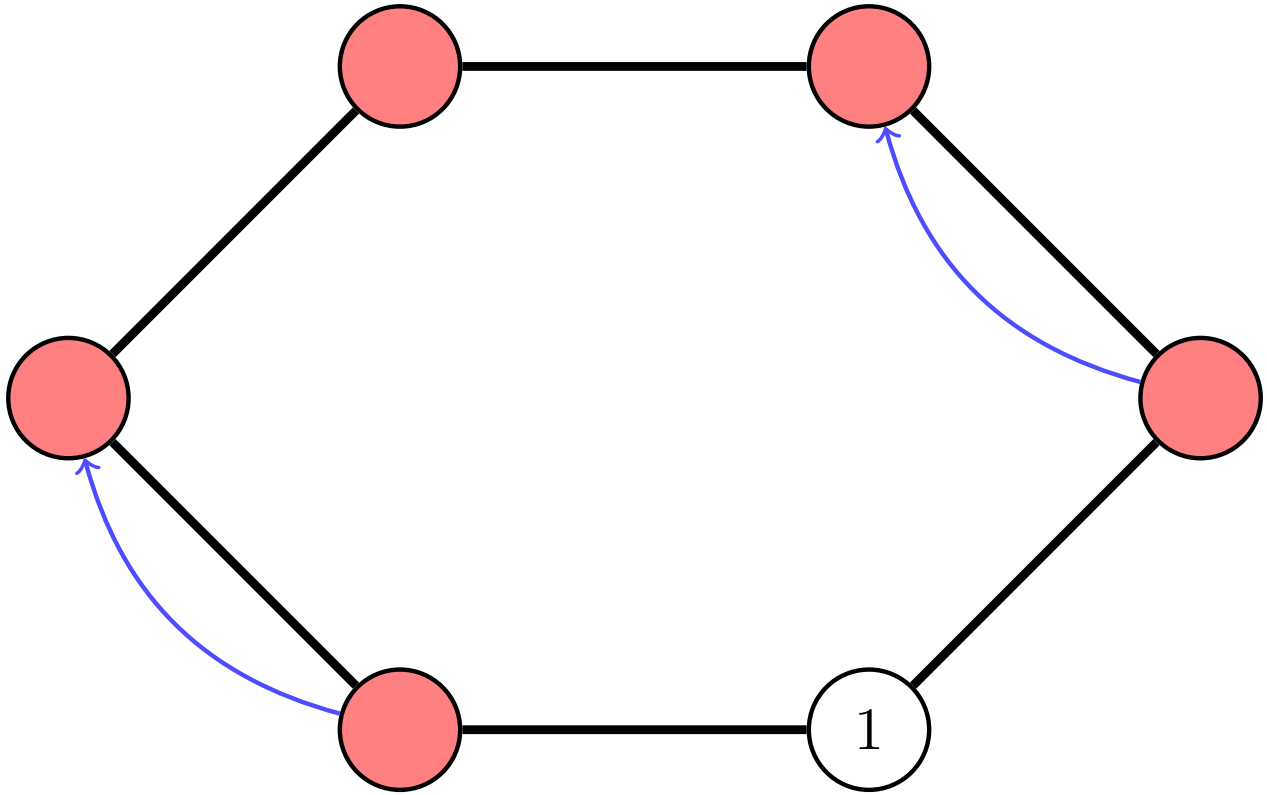
Example



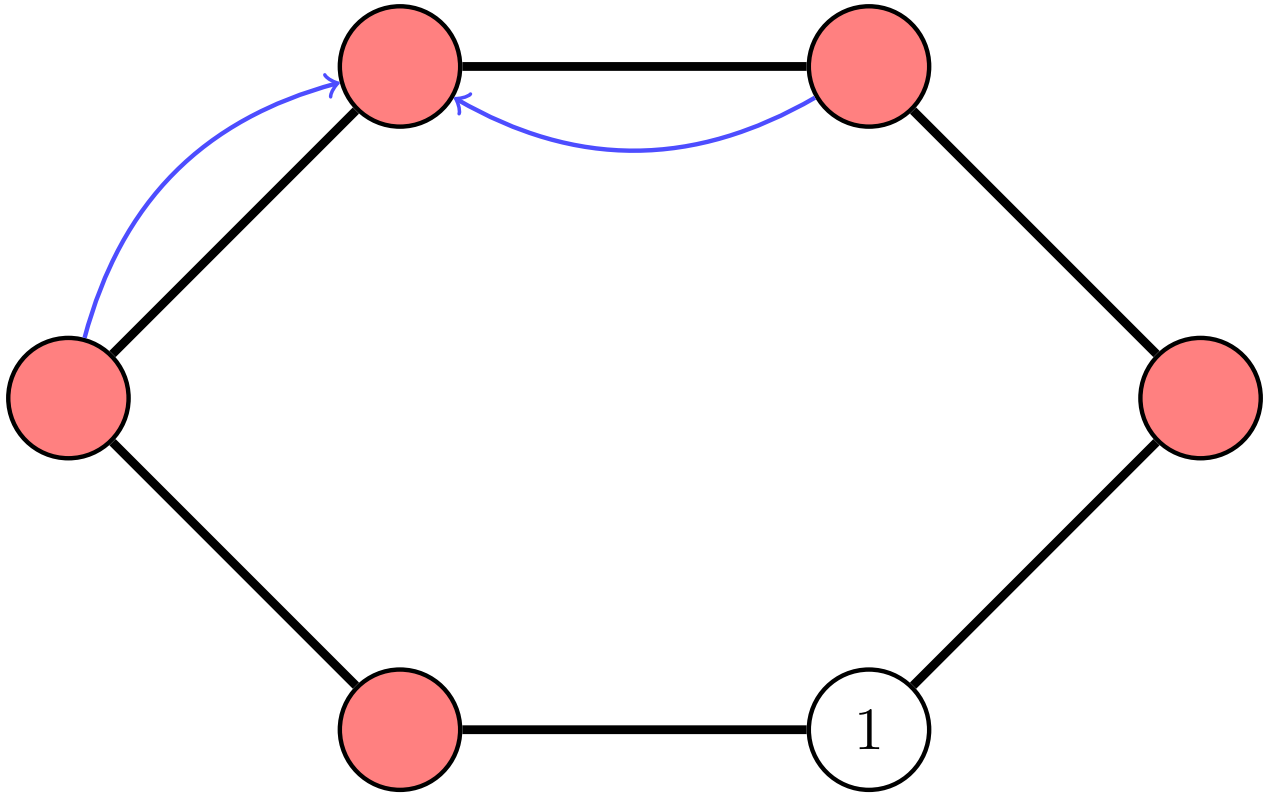
Example



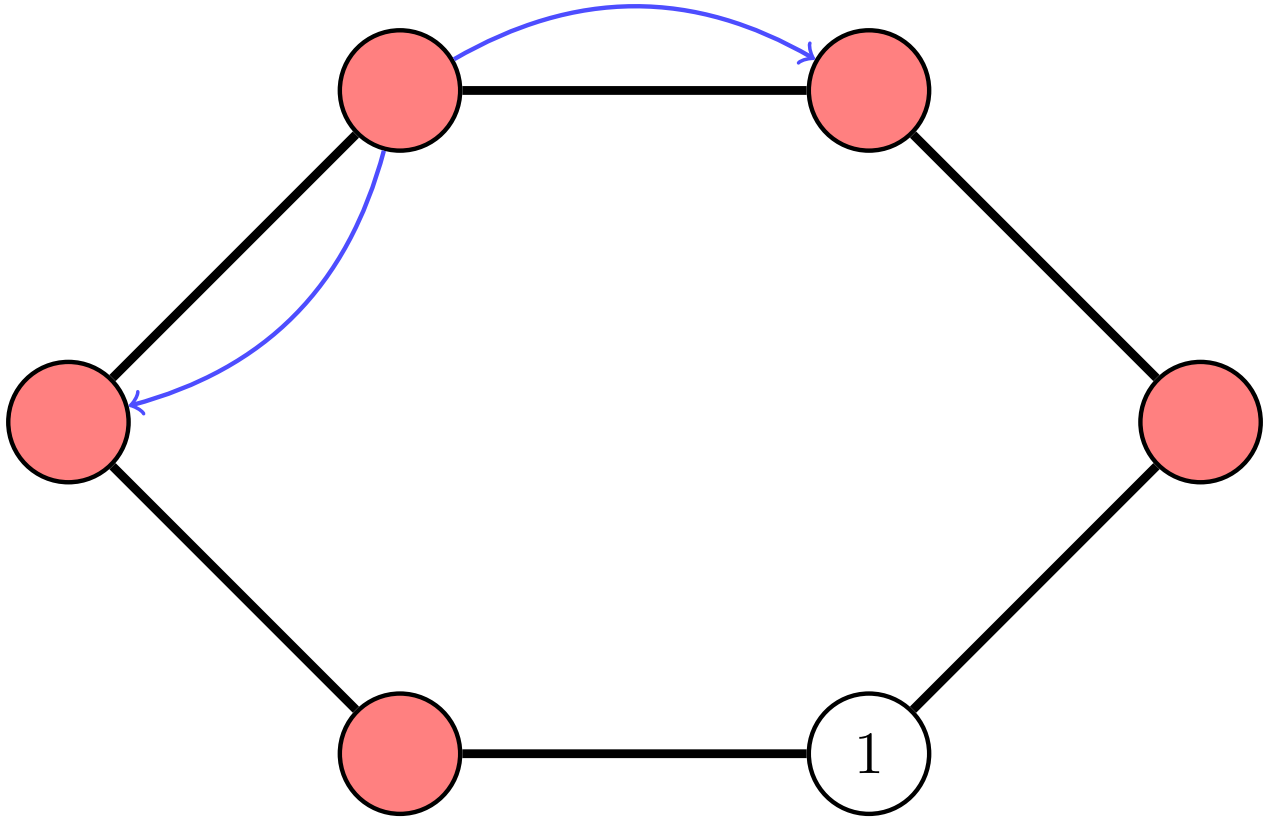
Example



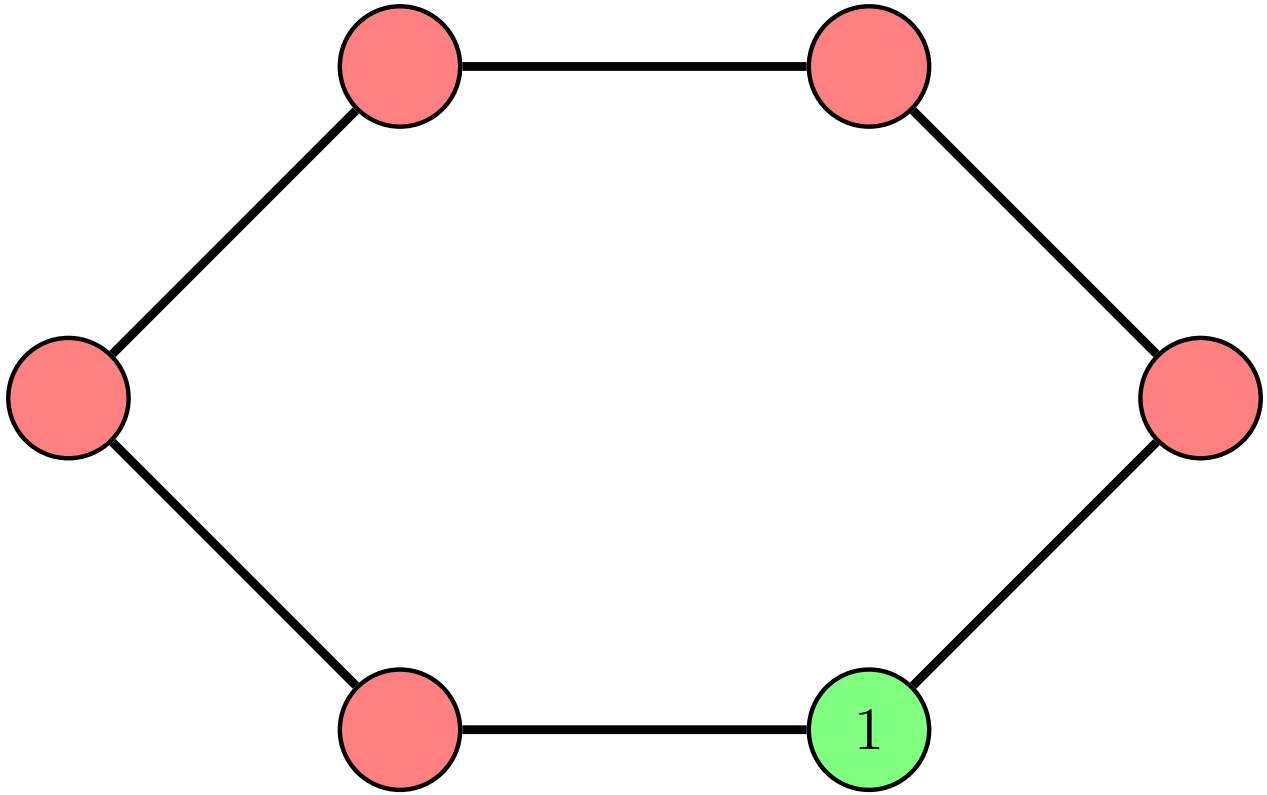
Example



Example



Example



Correctness

Theorem

The radius growth algorithm selects a unique leader.

Correctness

Theorem

The radius growth algorithm selects a unique leader.

Proof:

- Node z with the smallest ID can never receive a message with a smaller ID, so it never becomes a follower and therefore becomes the leader in the final round of the final phase.

Correctness

Theorem

The radius growth algorithm selects a unique leader.

Proof:

- Node z with the smallest ID can never receive a message with a smaller ID, so it never becomes a follower and therefore becomes the leader in the final round of the final phase.
- In the final phase ℓ (where $\ell = \lceil \log n \rceil$), the message containing the ID of z reaches $2^{\ell-1}$ nodes in each direction.

Correctness

Theorem

The radius growth algorithm selects a unique leader.

Proof:

- Node z with the smallest ID can never receive a message with a smaller ID, so it never becomes a follower and therefore becomes the leader in the final round of the final phase.
- In the final phase ℓ (where $\ell = \lceil \log n \rceil$), the message containing the ID of z reaches $2^{\ell-1}$ nodes in each direction.
- Since $2^{\ell-1} = 2^{\lceil \log n \rceil - 1} \geq 2^{\log n - 1} = \frac{2^{\log n}}{2} = \frac{n}{2}$, the message containing the ID of z reaches all other nodes, which become followers by then at the latest because they have IDs larger than z.

Correctness

Theorem

The radius growth algorithm selects a unique leader.

Proof:

- Node z with the smallest ID can never receive a message with a smaller ID, so it never becomes a follower and therefore becomes the leader in the final round of the final phase.
- In the final phase ℓ (where $\ell = \lceil \log n \rceil$), the message containing the ID of z reaches $2^{\ell-1}$ nodes in each direction.
- Since $2^{\ell-1} = 2^{\lceil \log n \rceil - 1} \geq 2^{\log n - 1} = \frac{2^{\log n}}{2} = \frac{n}{2}$, the message containing the ID of z reaches all other nodes, which become followers by then at the latest because they have IDs larger than z.
- Thus: a valid partition into one leader and followers.

Running Time

Theorem

The radius growth algorithm requires $O(n)$ rounds.

Running Time

Theorem

The radius growth algorithm requires $O(n)$ rounds.

Proof:

- Phase i requires $2^{i-1} + 1 \leq 2^i$ rounds.

Running Time

Theorem

The radius growth algorithm requires $O(n)$ rounds.

Proof:

- Phase i requires $2^{i-1} + 1 \leq 2^i$ rounds.
- Total number of rounds:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i$$

Running Time

Theorem

The radius growth algorithm requires $O(n)$ rounds.

Proof:

- Phase i requires $2^{i-1} + 1 \leq 2^i$ rounds.
- Total number of rounds:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1}$$

Geometric series: $\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ for $r \neq 1$ and $n \geq 0$

Running Time

Theorem

The radius growth algorithm requires $O(n)$ rounds.

Proof:

- Phase i requires $2^{i-1} + 1 \leq 2^i$ rounds.
- Total number of rounds:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2}$$

Geometric series: $\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ for $r \neq 1$ and $n \geq 0$

Running Time

Theorem

The radius growth algorithm requires $O(n)$ rounds.

Proof:

- Phase i requires $2^{i-1} + 1 \leq 2^i$ rounds.
- Total number of rounds:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2} \leq 4 \cdot 2^{\log n}$$

Geometric series: $\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ for $r \neq 1$ and $n \geq 0$

Running Time

Theorem

The radius growth algorithm requires $O(n)$ rounds.

Proof:

- Phase i requires $2^{i-1} + 1 \leq 2^i$ rounds.
- Total number of rounds:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2} \leq 4 \cdot 2^{\log n} = 4n$$

Geometric series: $\sum_{k=0}^{n} r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ for $r \neq 1$ and $n \geq 0$

Message Complexity I

We call nodes that are not yet followers **active**.

Message Complexity I

We call nodes that are not yet followers **active**.

Lemma

For every active node v, at the end of phase i, all other nodes at distance at most 2^{i-1} are inactive.

Message Complexity I

We call nodes that are not yet followers **active**.

Lemma

For every active node v, at the end of phase i, all other nodes at distance at most 2^{i-1} are inactive.

Proof:

- Suppose there is a node w at distance at most 2^{i-1} that is active at the end of phase i.

Message Complexity I

We call nodes that are not yet followers **active**.

Lemma

For every active node v, at the end of phase i, all other nodes at distance at most 2^{i-1} are inactive.

Proof:

- Suppose there is a node w at distance at most 2^{i-1} that is active at the end of phase i.
- Then w was also active at the beginning of phase i.

Message Complexity I

We call nodes that are not yet followers **active**.

Lemma

For every active node v, at the end of phase i, all other nodes at distance at most 2^{i-1} are inactive.

Proof:

- Suppose there is a node w at distance at most 2^{i-1} that is active at the end of phase i.
- Then w was also active at the beginning of phase i.
- If $\mathrm{ID}(v) > \mathrm{ID}(w)$: During phase i, v receives a message containing $\mathrm{ID}(w)$ (because w was active at the beginning of the phase) and therefore becomes inactive: a contradiction to the assumption that v is active at the end of phase i.

Message Complexity I

We call nodes that are not yet followers **active**.

Lemma

For every active node v, at the end of phase i, all other nodes at distance at most 2^{i-1} are inactive.

Proof:

- Suppose there is a node w at distance at most 2^{i-1} that is active at the end of phase i.
- Then w was also active at the beginning of phase i.
- If $\mathrm{ID}(v) > \mathrm{ID}(w)$: During phase i, v receives a message containing $\mathrm{ID}(w)$ (because w was active at the beginning of the phase) and therefore becomes inactive: a contradiction to the assumption that v is active at the end of phase i.
- If $\mathrm{ID}(v) < \mathrm{ID}(w)$: During phase i, w receives a message containing $\mathrm{ID}(v)$ and therefore becomes inactive: a contradiction to the assumption that w is active at the end of phase i.

Message Complexity II

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

Message Complexity II

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

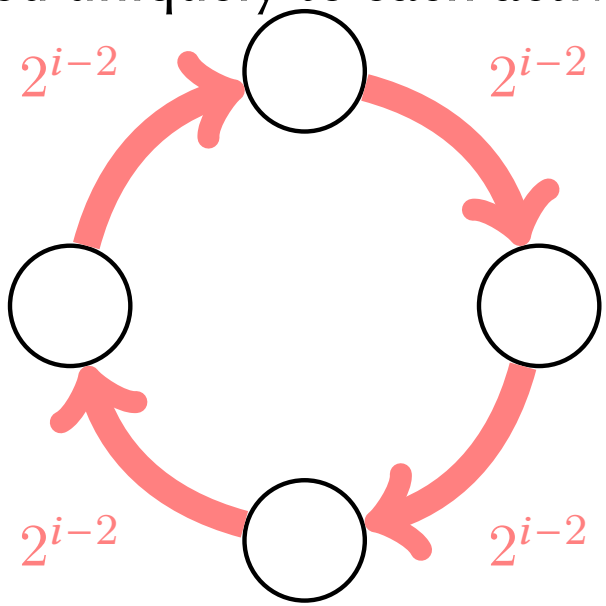
- By the previous lemma, the next 2^{i-2} inactive nodes in the clockwise direction can be assigned uniquely to each active node.

Message Complexity II

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

- By the previous lemma, the next 2^{i-2} inactive nodes in the clockwise direction can be assigned uniquely to each active node.

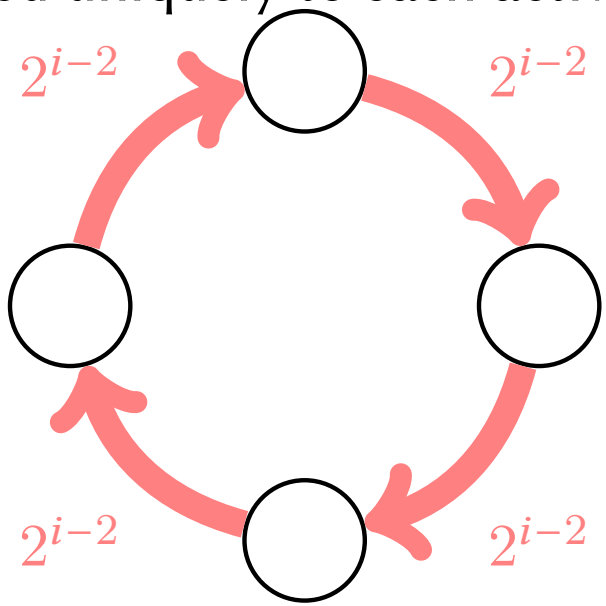


Message Complexity II

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

- By the previous lemma, the next 2^{i-2} inactive nodes in the clockwise direction can be assigned uniquely to each active node.



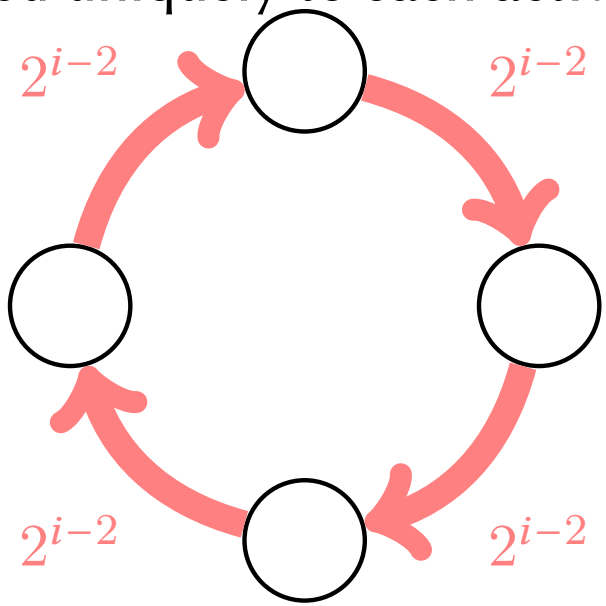
- Let a be the number of active nodes.

Message Complexity II

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

- By the previous lemma, the next 2^{i-2} inactive nodes in the clockwise direction can be assigned uniquely to each active node.



- Let a be the number of active nodes.
- Then $a \cdot 2^{i-2} \leq n$ and therefore $a \leq n/2^{i-2}$.

Message Complexity III

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

Message Complexity III

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

Theorem

The radius growth algorithm sends at most $O(n \log n)$ messages.

Message Complexity III

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

Theorem

The radius growth algorithm sends at most $O(n \log n)$ messages.

Proof:

- In each phase, every active node initiates two messages, which are forwarded once in every round of the phase.

Message Complexity III

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

Theorem

The radius growth algorithm sends at most $O(n \log n)$ messages.

Proof:

- In each phase, every active node initiates two messages, which are forwarded once in every round of the phase.
- Number of messages in phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$

Message Complexity III

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

Theorem

The radius growth algorithm sends at most $O(n \log n)$ messages.

Proof:

- In each phase, every active node initiates two messages, which are forwarded once in every round of the phase.
- Number of messages in phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$
- Number of messages in phase 1: $2n$

Message Complexity III

Lemma

For $i \geq 2$, the number of active nodes at the beginning of phase i is at most $n/2^{i-2}$.

Theorem

The radius growth algorithm sends at most $O(n \log n)$ messages.

Proof:

- In each phase, every active node initiates two messages, which are forwarded once in every round of the phase.
- Number of messages in phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$
- Number of messages in phase 1: $2n$
- There are $\lceil \log n \rceil$ phases, and therefore $O(n \log n)$ messages in total.

Summary

Radius Growth:

- Selects the node with the smallest ID as leader
- Running time: $O(n)$
- Message complexity: $O(n \log n)$
- Can also be formulated as an asynchronous algorithm for uniform rings

Randomized Algorithm [Attiya/Welch '98]

Assumptions: synchronous, anonymous, non-uniform

Randomized Algorithm [Attiya/Welch '98]

Assumptions: synchronous, anonymous, non-uniform

Algorithm:

1. Each node chooses an ID uniformly at random from 0 to $2n-1$
2. Nodes execute the clockwise algorithm with the chosen IDs

Randomized Algorithm [Attiya/Welch '98]

Assumptions: synchronous, anonymous, non-uniform

Algorithm:

1. Each node chooses an ID uniformly at random from 0 to $2n-1$
2. Nodes execute the clockwise algorithm with the chosen IDs
   Problem:
   - IDs may be assigned to multiple nodes
   - The algorithm might produce an incorrect result
   - The clockwise algorithm makes all nodes with the smallest assigned ID leaders

Randomized Algorithm [Attiya/Welch '98]

Assumptions: synchronous, anonymous, non-uniform

Algorithm:

1. Each node chooses an ID uniformly at random from 0 to $2n-1$
2. Nodes execute the clockwise algorithm with the chosen IDs
   Problem:
   - IDs may be assigned to multiple nodes
   - The algorithm might produce an incorrect result
   - The clockwise algorithm makes all nodes with the smallest assigned ID leaders
3. Nodes verify whether the result is correct

Randomized Algorithm [Attiya/Welch '98]

Assumptions: synchronous, anonymous, non-uniform

Algorithm:

1. Each node chooses an ID uniformly at random from 0 to $2n-1$
2. Nodes execute the clockwise algorithm with the chosen IDs
   Problem:
   - IDs may be assigned to multiple nodes
   - The algorithm might produce an incorrect result
   - The clockwise algorithm makes all nodes with the smallest assigned ID leaders
3. Nodes verify whether the result is correct

   The validity of a leader/follower assignment can be checked in $O(n)$ rounds using $O(n)$ messages

Randomized Algorithm [Attiya/Welch '98]

Assumptions: synchronous, anonymous, non-uniform

Algorithm:

1. Each node chooses an ID uniformly at random from 0 to $2n-1$
2. Nodes execute the clockwise algorithm with the chosen IDs
   Problem:
   - IDs may be assigned to multiple nodes
   - The algorithm might produce an incorrect result
   - The clockwise algorithm makes all nodes with the smallest assigned ID leaders
3. Nodes verify whether the result is correct
   The validity of a leader/follower assignment can be checked in $O(n)$ rounds using $O(n)$ messages
4. If the result is incorrect, repeat from step 1

Success Probability of One Iteration I

Observation

An iteration of the algorithm is successful if and only if the smallest assigned ID occurs exactly once.

Success Probability of One Iteration I

Observation

An iteration of the algorithm is successful if and only if the smallest assigned ID occurs exactly once.

Observation

If exactly one node receives ID 0, then the iteration is guaranteed to succeed.

Success Probability of One Iteration I

Observation

An iteration of the algorithm is successful if and only if the smallest assigned ID occurs exactly once.

Observation

If exactly one node receives ID 0, then the iteration is guaranteed to succeed.

- Random variables X_v (for all $v \in V$):

$$X_v = \begin{cases} 1 & \text{if node } v \text{ receives ID } 0 \\ 0 & \text{otherwise} \end{cases}$$

Success Probability of One Iteration I

Observation

An iteration of the algorithm is successful if and only if the smallest assigned ID occurs exactly once.

Observation

If exactly one node receives ID 0, then the iteration is guaranteed to succeed.

- Random variables X_v (for all $v \in V$):

$$X_v = \begin{cases} 1 & \text{if node } v \text{ receives ID } 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\Pr[X_v = 1] = \frac{1}{2n}$, because the ID is chosen uniformly from $0 \ldots 2n-1$

Success Probability of One Iteration I

Observation

An iteration of the algorithm is successful if and only if the smallest assigned ID occurs exactly once.

Observation

If exactly one node receives ID 0, then the iteration is guaranteed to succeed.

- Random variables X_v (for all $v \in V$):

$$X_v = \begin{cases} 1 & \text{if node } v \text{ receives ID } 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\Pr[X_v = 1] = \frac{1}{2n}$, because the ID is chosen uniformly from $0 \ldots 2n-1$
- Each X_v has a Bernoulli distribution with success probability $p := \frac{1}{2n}$

Success Probability of One Iteration I

Observation

An iteration of the algorithm is successful if and only if the smallest assigned ID occurs exactly once.

Observation

If exactly one node receives ID 0, then the iteration is guaranteed to succeed.

- Random variables X_v (for all $v \in V$):

$$X_v = \begin{cases} 1 & \text{if node } v \text{ receives ID } 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\Pr[X_v = 1] = \frac{1}{2n}$, because the ID is chosen uniformly from $0 \ldots 2n-1$
- Each X_v has a Bernoulli distribution with success probability $p := \frac{1}{2n}$
- Random variable $X := \sum_{v \in V} X_v$ counting the number of nodes with ID 0

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Thus: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ for all $0 \leq k \leq n$

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Thus: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ for all $0 \leq k \leq n$

In particular: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Thus: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ for all $0 \leq k \leq n$

In particular: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1}$$

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Thus: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ for all $0 \leq k \leq n$

In particular: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1}$$

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Thus: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ for all $0 \leq k \leq n$

In particular: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$

Bernoulli inequality: $(1+x)^n \geq 1 + xn$ for $x \geq -1$ and $n \geq 0$

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Thus: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ for all $0 \leq k \leq n$

In particular: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$
$$\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right)$$

Bernoulli inequality: $(1+x)^n \geq 1 + xn$ for $x \geq -1$ and $n \geq 0$

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Thus: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ for all $0 \leq k \leq n$

In particular: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$
$$\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right) = \frac{1}{4}$$

Bernoulli inequality: $(1+x)^n \geq 1 + xn$ for $x \geq -1$ and $n \geq 0$

Success Probability of One Iteration II

Observation

The X_v's are mutually independent, and therefore $X := \sum_{v \in V} X_v$ is binomially distributed (with parameters n and p).

Thus: $\Pr[X = k] = \binom{n}{k} p^k (1-p)^{n-k}$ for all $0 \leq k \leq n$

In particular: $\Pr[X = 1] = n \cdot p(1-p)^{n-1}$

$$\Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$
$$\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right) = \frac{1}{4}$$

Bernoulli inequality: $(1+x)^n \geq 1 + xn$ for $x \geq -1$ and $n \geq 0$

Thus: Every iteration is successful with probability $q \geq \frac{1}{4}$

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k \geq 1} A_k$: the algorithm eventually terminates

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k \geq 1} A_k$: the algorithm eventually terminates
- Since the events A_k are disjoint:

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k\geq 1} A_k$: the algorithm eventually terminates
- Since the events A_k are disjoint:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q$$

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k\geq 1} A_k$: the algorithm eventually terminates
- Since the events A_k are disjoint:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k\geq 1} A_k$: the algorithm eventually terminates
- Since the events A_k are disjoint:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k}$$

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k\geq 1} A_k$: the algorithm eventually terminates
- Since the events A_k are disjoint:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k} = q \cdot \frac{1}{1-(1-q)}$$

Geometric series: $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ for $-1 < r < 1$

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k\geq 1} A_k$: the algorithm eventually terminates
- Since the events A_k are disjoint:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k} = q \cdot \frac{1}{1-(1-q)} = q \cdot \frac{1}{q}$$

Geometric series: $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ for $-1 < r < 1$

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k\geq 1} A_k$: the algorithm eventually terminates
- Since the events A_k are disjoint:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k} = q \cdot \frac{1}{1-(1-q)} = q \cdot \frac{1}{q} = 1$$

Geometric series: $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ for $-1 < r < 1$

Correctness

Theorem

The randomized leader election algorithm terminates with probability 1.

Proof:

- Event A_k: the algorithm terminates after exactly k iterations
- Event $\bigcup_{k\geq 1} A_k$: the algorithm eventually terminates
- Since the events A_k are disjoint:

$$\Pr\left[\bigcup_{k\geq 1} A_k\right] = \sum_{k\geq 1} \Pr[A_k] = \sum_{k=1}^{\infty} (1-q)^{k-1} q = q \cdot \sum_{k=1}^{\infty} (1-q)^{k-1}$$

$$= q \cdot \sum_{k=0}^{\infty} (1-q)^{k} = q \cdot \frac{1}{1-(1-q)} = q \cdot \frac{1}{q} = 1$$

Geometric series: $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$ for $-1 < r < 1$

Caution: "Probability 1" ≠ "always"—the complementary event is possible!

Geometric Distribution

Random variables Y_i (for $i \geq 1$):

$$Y_i = \begin{cases} 1 & \text{if the } i\text{th iteration of the algorithm is successful} \\ 0 & \text{otherwise} \end{cases}$$

Geometric Distribution

Random variables Y_i (for $i \geq 1$):

$$Y_i = \begin{cases} 1 & \text{if the } i\text{th iteration of the algorithm is successful} \\ 0 & \text{otherwise} \end{cases}$$

Random variable $Z = \min\{i \geq 1 \mid Y_i = 1\}$ representing the number of iterations of the algorithm

Geometric Distribution

Random variables Y_i (for $i \geq 1$):

$$Y_i = \begin{cases} 1 & \text{if the } i\text{th iteration of the algorithm is successful} \\ 0 & \text{otherwise} \end{cases}$$

Random variable $Z = \min\{i \geq 1 \mid Y_i = 1\}$ representing the number of iterations of the algorithm

Observation

The Y_i's are mutually independent, and therefore Z is geometrically distributed (with parameter q).

Geometric Distribution

Random variables Y_i (for $i \geq 1$):

$$Y_i = \begin{cases} 1 & \text{if the } i\text{th iteration of the algorithm is successful} \\ 0 & \text{otherwise} \end{cases}$$

Random variable $Z = \min\{i \geq 1 \mid Y_i = 1\}$ representing the number of iterations of the algorithm

Observation

The Y_i's are mutually independent, and therefore Z is geometrically distributed (with parameter q).

Thus: $\Pr[Z = k] = q(1-q)^{k-1}$ for all $k \geq 1$

Running Time Analysis I

Theorem

The expected number of iterations of the randomized leader election algorithm is $O(1)$.

Running Time Analysis I

Theorem

The expected number of iterations of the randomized leader election algorithm is $O(1)$.

Proof:

- Success probability $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in every iteration

Running Time Analysis I

Theorem

The expected number of iterations of the randomized leader election algorithm is $O(1)$.

Proof:

- Success probability $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in every iteration
- For a geometrically distributed random variable Z, the expected number of iterations is $\frac{1}{q} \leq 4 = O(1)$

Running Time Analysis I

Theorem

The expected number of iterations of the randomized leader election algorithm is $O(1)$.

Proof:

- Success probability $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in every iteration
- For a geometrically distributed random variable Z, the expected number of iterations is $\frac{1}{q} \leq 4 = O(1)$

Question: An expectation allows for large deviations; after how many iterations can we be "reasonably sure" that the algorithm has terminated?

Running Time Analysis I

Theorem

The expected number of iterations of the randomized leader election algorithm is $O(1)$.

Proof:

- Success probability $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in every iteration
- For a geometrically distributed random variable Z, the expected number of iterations is $\frac{1}{q} \leq 4 = O(1)$

Question: An expectation allows for large deviations; after how many iterations can we be "reasonably sure" that the algorithm has terminated?

Definition

An event occurs **with high probability** if it occurs with probability at least $1 - \frac{1}{n^c}$, for an arbitrarily specified constant $c \geq 1$.

Running Time Analysis I

Theorem

The expected number of iterations of the randomized leader election algorithm is $O(1)$.

Proof:

- Success probability $\Pr[Y_i = 1] = q \geq \frac{1}{4}$ in every iteration
- For a geometrically distributed random variable Z, the expected number of iterations is $\frac{1}{q} \leq 4 = O(1)$

Question: An expectation allows for large deviations; after how many iterations can we be "reasonably sure" that the algorithm has terminated?

Definition

An event occurs **with high probability** if it occurs with probability at least $1 - \frac{1}{n^c}$, for an arbitrarily specified constant $c \geq 1$. (In particular: $\lim_{n\to\infty} 1 - \frac{1}{n^c} = 1$)

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1 - q)^k$$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1 - q)^k \leq \left(1 - \frac{1}{4}\right)^k$$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k$$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k$$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

$$= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c}$$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

$$= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}}$$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

$$= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}} = \frac{1}{n^c}$$

Running Time Analysis II

Theorem

For every $c \geq 1$, with probability at least $1 - \frac{1}{n^c}$, the randomized leader election algorithm requires $O(c \log n)$ iterations.

Proof:

- We show: $\Pr[Z \leq k] \geq 1 - \frac{1}{n^c}$ for $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Probability of the complementary event (the first k attempts are unsuccessful):

$$\Pr[Z > k] = (1-q)^k \leq \left(1 - \frac{1}{4}\right)^k = \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$
$$= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}} = \frac{1}{n^c}$$

- **Thus:** $\Pr[Z \leq k] = 1 - \Pr[Z > k] \geq 1 - \frac{1}{n^c}$

Summary

Randomized algorithm:

- Chooses IDs at random
- Makes the node with the smallest ID the leader
- Repeats until leader election succeeds
- Running Time: $O(n)$ rounds in expectation
- Message complexity: $O(n^2)$ in expectation Under a naive analysis using the clockwise algorithm
- Can also be formulated as an asynchronous algorithm

General Principle

We have implicitly shown:

General Principle

We have implicitly shown:

Theorem (Monte Carlo $\rightarrow$ Las Vegas)

- *Let $\mathcal{A}$ be a randomized algorithm for a problem $\mathcal{P}$ that computes a correct solution for every input with probability $q > 0$.*

General Principle

We have implicitly shown:

Theorem (Monte Carlo → Las Vegas)

- *Let $\mathcal{A}$ be a randomized algorithm for a problem $\mathcal{P}$ that computes a correct solution for every input with probability $q > 0$.*
- *Let $\mathcal{V}$ be a verifier that can determine, for every output of $\mathcal{A}$, whether the solution is correct.*

General Principle

We have implicitly shown:

Theorem (Monte Carlo → Las Vegas)

- *Let $\mathcal{A}$ be a randomized algorithm for a problem $\mathcal{P}$ that computes a correct solution for every input with probability $q > 0$.*
- *Let $\mathcal{V}$ be a verifier that can determine, for every output of $\mathcal{A}$, whether the solution is correct.*
- *Then there is an algorithm $\mathcal{B}$ that computes a correct solution with probability 1 and calls $\mathcal{A}$ and $\mathcal{V}$ $O(1/q)$ times in expectation (or $O(c \log_{1-1/q} n)$ times with probability $1 - \frac{1}{n^c}$).*

General Principle

We have implicitly shown:

Theorem (Monte Carlo → Las Vegas)

- *Let $\mathcal{A}$ be a randomized algorithm for a problem $\mathcal{P}$ that computes a correct solution for every input with probability $q > 0$.*
- *Let $\mathcal{V}$ be a verifier that can determine, for every output of $\mathcal{A}$, whether the solution is correct.*
- *Then there is an algorithm $\mathcal{B}$ that computes a correct solution with probability 1 and calls $\mathcal{A}$ and $\mathcal{V}$ $O(1/q)$ times in expectation (or $O(c \log_{1-1/q} n)$ times with probability $1 - \frac{1}{n^c}$).*

Definition

A randomized algorithm $\mathcal{B}$ whose output is always correct is called a **Las Vegas** algorithm.

General Principle

We have implicitly shown:

Theorem (Monte Carlo → Las Vegas)

- *Let $\mathcal{A}$ be a randomized algorithm for a problem $\mathcal{P}$ that computes a correct solution for every input with probability $q > 0$.*
- *Let $\mathcal{V}$ be a verifier that can determine, for every output of $\mathcal{A}$, whether the solution is correct.*
- *Then there is an algorithm $\mathcal{B}$ that computes a correct solution with probability 1 and calls $\mathcal{A}$ and $\mathcal{V}$ $O(1/q)$ times in expectation (or $O(c \log_{1-1/q} n)$ times with probability $1 - \frac{1}{n^c}$).*

Definition

A randomized algorithm $\mathcal{B}$ whose output is always correct is called a **Las Vegas** algorithm. A randomized algorithm $\mathcal{A}$ whose output is correct with some probability is called a **Monte Carlo** algorithm.

Summary

Leader Election in a Ring illustrates fundamental principles and techniques of distributed algorithms:

Summary

Leader Election in a Ring illustrates fundamental principles and techniques of distributed algorithms:

- Variety of models: synchronous/asynchronous, anonymous/identifiable, uniform/non-uniform

Summary

Leader Election in a Ring illustrates fundamental principles and techniques of distributed algorithms:

- Variety of models: synchronous/asynchronous, anonymous/identifiable, uniform/non-uniform
- Deterministic vs. randomized algorithms

Summary

Leader Election in a Ring illustrates fundamental principles and techniques of distributed algorithms:

- Variety of models: synchronous/asynchronous, anonymous/identifiable, uniform/non-uniform
- Deterministic vs. randomized algorithms
- Complexity measures: #rounds, #messages

Summary

Leader Election in a Ring illustrates fundamental principles and techniques of distributed algorithms:

- Variety of models: synchronous/asynchronous, anonymous/identifiable, uniform/non-uniform
- Deterministic vs. randomized algorithms
- Complexity measures: #rounds, #messages
- Upper and lower bounds (or impossibility results)

Summary

Leader Election in a Ring illustrates fundamental principles and techniques of distributed algorithms:

- Variety of models: synchronous/asynchronous, anonymous/identifiable, uniform/non-uniform
- Deterministic vs. randomized algorithms
- Complexity measures: #rounds, #messages
- Upper and lower bounds (or impossibility results)
- Two ways to break symmetry:
  - Unique IDs
  - Randomization

References

The content of this lecture is partly based on lectures by Robert Elsässer and Stefan Schmid.

Literature:

- Hagit Attiya, Jennifer Welch (2004) *Distributed Computing*, Chapters 3 and 14, Wiley.
- Daniel S. Hirschberg, James B. Sinclair. "Decentralized Extrema-Finding in Circular Configurations of Processors." *Communications of the ACM* 23(11): 627–628 (1980)
- Alon Itai, Michael Rodeh. "Symmetry breaking in distributed networks." *Information and Computation* 88(1): 60–87 (1990)
- Nancy A. Lynch (1996) *Distributed Algorithms*, Kapitel 3, Morgan Kaufmann.