Technische Universität Berlin

Distributed Algorithms 2016/17
Election Algorithms

Odej Kao | Complex and Distributed IT Systems

With material of R. Karnapke | KBS

Overview

The Election Problem

Election Algorithms for

- arbitrary connected topologies
- unidirectional and bidirectional rings
- trees

Randomized election algorithms for

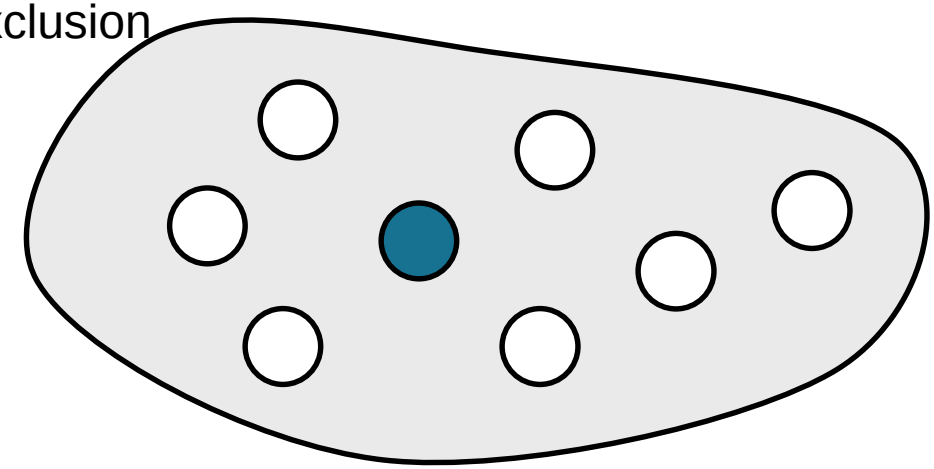
- bidirectional rings
- anonymous rings

The Election Problem

From a set of (almost) identical processes a *unique* leader shall be elected

Exemplary Applications

- Determining the monitor station of Token-Ring-LANs
- Generating a unique token
- Determining the root node of a spanning tree
- Determining the master in distributed files systems or centralized mutual exclusion



The Election Problem

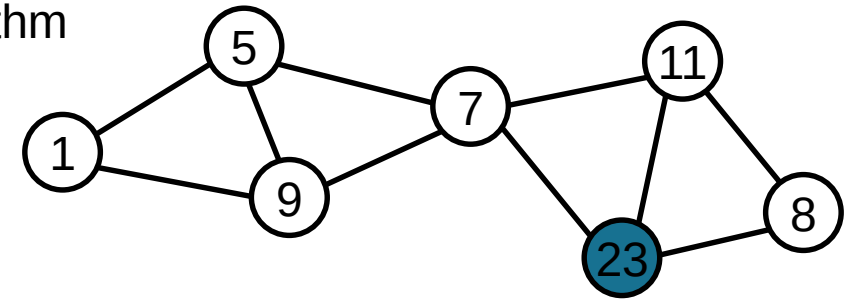
Assumption: Each node has a unique integer identity > 0

Requirements

- Each node shall know the winner
- Each node may (concurrently) initiate the algorithm

MAX-Algorithms

- Determine the largest identity in the topology or among the initiators
- Can be used as election algorithm



Election Algorithms for Arbitrary Topologies

Election Algorithms for Arbitrary Topologies

```
I_p: {M_p == 0}
   M_p := p;
   SEND <M_p> TO all neighbors;
```

I_p is executed by initiators.

```
R_p: {A message <j> has arrived}
   IF M_p < j THEN
      M_p := j;
      SEND <M_p> TO all other neighbors;
   FI
```

After R_p was executed, p cannot become initiator. Thus, p cannot win and the highest initiator wins.

```
T_p: {Termination was discovered}
   IF M_p == p THEN
       "I am the master"
   ELSE
      "M_p is the master"
   FI
```

Each process has a unique identity p and a local variable M_p, which is 0 initially.

But how?

Echo Election Algorithm

- Works with arbitrary connected topologies
- Each initiator starts an instance of the echo algorithm
- Explorer and echoes carry the identity of the initiators with them
- Weaker messages (explorer and echoes) are not passed on but swallowed

  → Message extinction
- Strongest wave prevails and terminates at the winner
  - The winner knows that it has won
  - If an initiator receives a stronger explorer, it accepts lost election

Echo Election Algorithm

- But how do the losers know of the termination and get to know the ID of the winner?
  - The winner starts the echo algorithm once again to distribute the win notification
  - Using the generated spanning tree for this purpose is also possible
- All other waves finally stagnate somewhere
  - At least the strongest initiator sends no echo for the weaker waves
  - No echo algorithm of any other initiator terminates

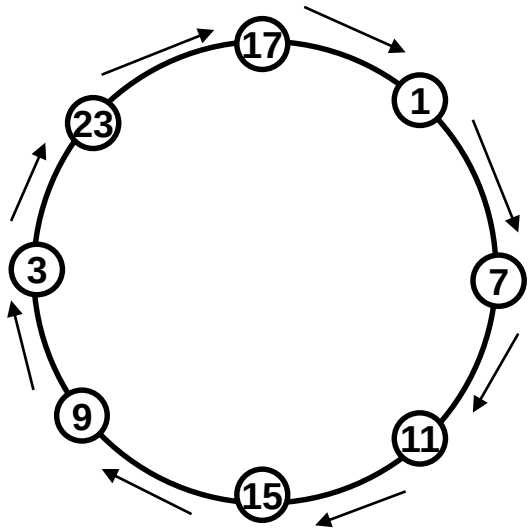
Election Algorithms for unidirectional Rings

1. Idea: Bully-Algorithm

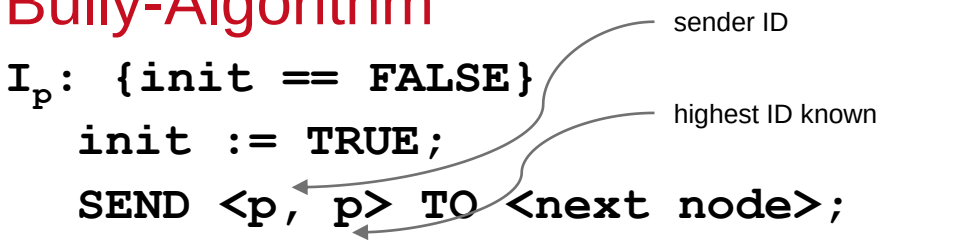
- Each process wakes up, either as initiator or at the latest when it receives a message from its neighbor node
- *Each* waking up starts a complete ring circulation
- At the end of *its* circulation, each node receives a message with its own identity and the ID of the node with the largest ID within the ring
- If both IDs are identical, the node has won the election; otherwise it has lost
- Highest of *all* nodes wins the election

⇒ n complete circulations

⇒ n^2 single messages



Bully-Algorithm




```
I_p: {init == FALSE}
   init := TRUE;
   SEND <p, p> TO <next node>;
R_p: {A message <i, j> has arrived}
   IF i != p THEN
      k := MAX(j, p);
      SEND <i, k> TO <next node>;
   ELSE
      IF p == j THEN
          "I am the master"
      ELSE
         "j is the master"
      FI
   FI
```

I_p is spontaneously executed by the concurring initiators and by the other nodes with reception of the 1st message.

For each process initially **`init == FALSE`**

Variant of Bully-Algorithm

- Also possible: Variant that determines the highest node *among the initiators*

- Changes
  - Nodes not participating in the election pass the messages unchanged
  - Only winner knows that it won

    → extra round to inform all nodes

2. Idea: *Message Extinction*

With the Bully-Algorithm, messages that cannot lead to a win are passed on as well

Idea of Chang and Roberts, 1979

- Messages are only passed on if they can lead to a win;
  all others are extinct
- Since only the winner receives its own message, the other nodes are informed of the win by an additional ring circulation

2. Idea: *Message Extinction*

Attention: Here, only an initiator can win!

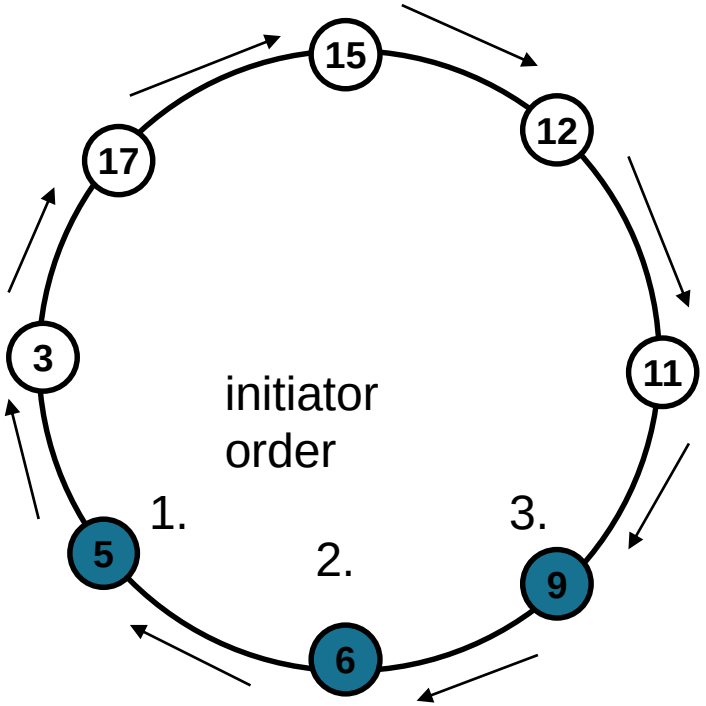
```
I_p: {M_p == 0}
    M_p := p;
    SEND <M_p> TO next node;

R_p: {A message <j> has arrived}
    IF M_p < j THEN
      M_p := j;
      SEND <M_p> TO next node;
    FI
    IF j == p THEN
       "I am the master"
      <Inform all by another ring circuit>;
    FI
```

For each process, initially $M_p == 0$

Worst-Case Message Complexity with *k* Initiators

- Occurs, if the initiators are arranged on the ring in descending order *and* initiate election in ascending order
  - *k*- largest initiator
    → *n* – (*k* – 1) messages
  - …
  - 3rd-largest initiator
    → *n* – 2 messages
  - 2nd-largest initiator
    → *n* – 1 messages
  - Largest initiator
    → *n* messages



Worst-Case Message Complexity with *k* Initiators

Message complexity with *k* initiators

$$n + (n-1) + (n-2) + \ldots + (n-(k-1))$$

$$= \frac{n(n+1)}{2} - \frac{(n-k)(n-k+1)}{2}$$

$$= \frac{(n^2 + n - n^2 + nk - n + nk - k^2 + k)}{2}$$

$$= \frac{(2nk - k^2 + k)}{2}$$

$$= nk - \frac{k(k-1)}{2}$$

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

⇒ $O(n^2)$ with ring size n and $k = n$

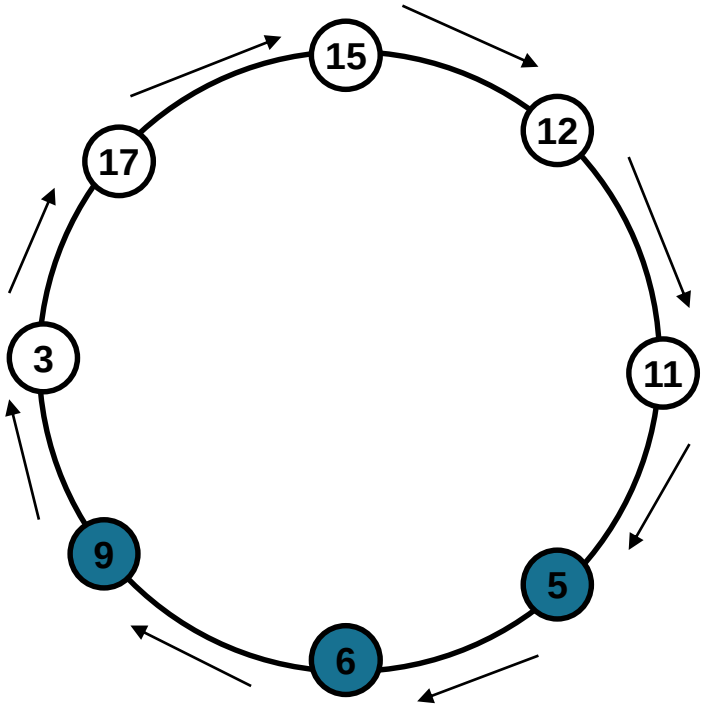
> Still n additional messages for the win notification

Best-Case Message Complexity with *k* Initiators

- Occurs, if the initiators are arranged on the ring in ascending order and initiate the election approximately simultaneously
- All but the largest initiator cause only 1 message
- Largest initiator causes *n* messages

⇒ $n + k - 1$ messages for the actual election

- Again, still *n* additional messages for the win notification

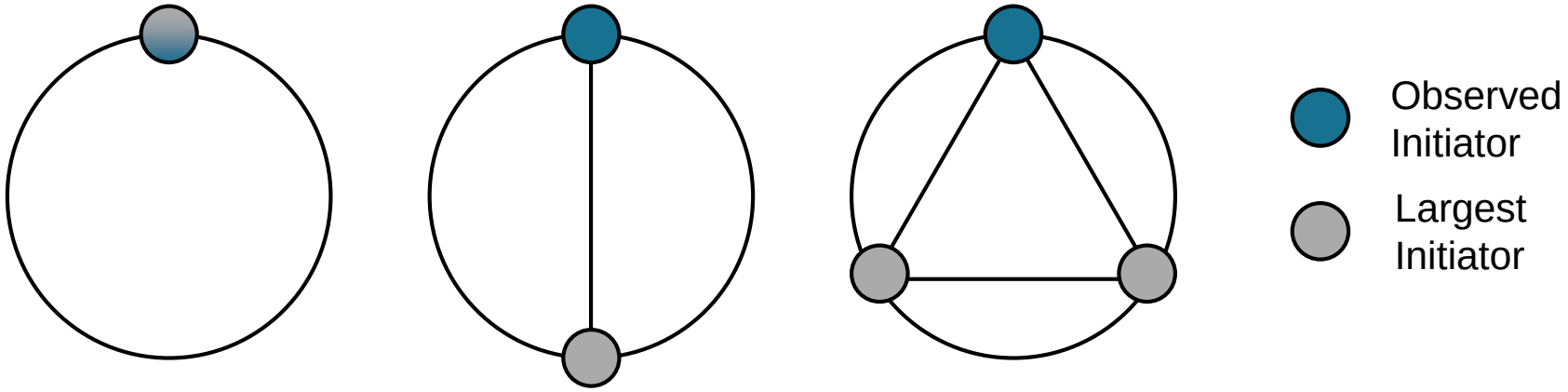


Average-Case Message Complexity

Average the message complexity over all possible permutations of the identities on the ring!

Informal argumentation

- Largest initiator → n messages (always)
- 2nd-largest initiator → $n / 2$ messages on average
- 3rd-largest initiator → $n / 3$ messages on average
- …
- i-largest initiator → n / i messages on average



Average-Case Message Complexity

- The average-case message complexity is
  $$\boldsymbol{n\, H_k \approx n \ln k} \qquad \text{with } H_k = 1 + 1/2 + \ldots + 1/k$$
  - This is optimal for unidirectional rings.
  - H_k is the k-th Harmonic Number → Harmonic series

- For very large rings, almost never more messages are required than on average (cf. Rotem et al.)

- Again, still n additional messages for the win notification

- Remark: It was assumed implicitly that no message overtakes can take place on a link

- How do overtakes influence the message complexity?

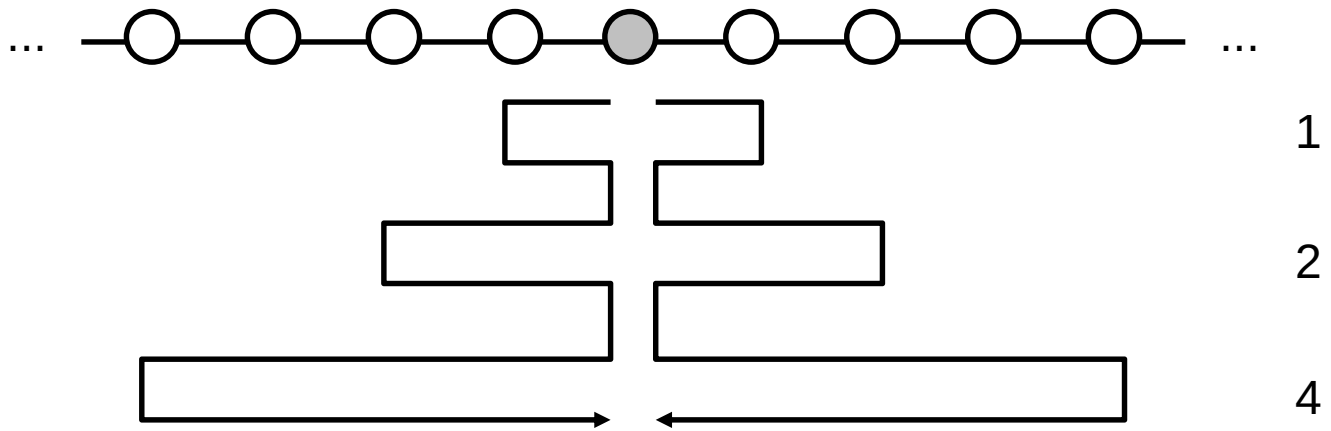
Influence of Message Overtaking

- A message can only be overtaken by higher messages because a message is only passed on by a node if it is higher than the former highest message sent
- Through the overtake, the lower message is extinct potentially *earlier*; in this case messages are saved
- No messages are saved if the receiving node is a higher initiator as in this case, the message would have been extinct anyway

Election Algorithms for Bidirectional Rings

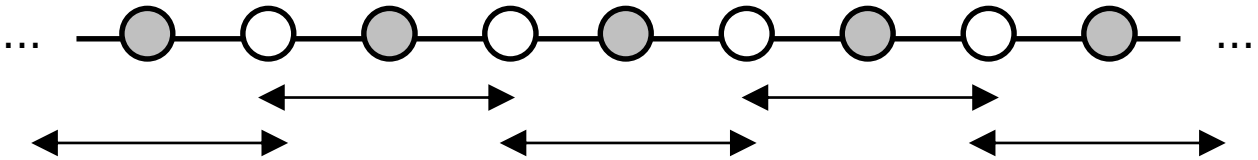
Hirschberg-Sinclair-Election Algorithm

- Each node tries to conquer successive areas of size 2^i (i = 1, 2, 3, ...) on a bidirectional ring until the message arrives again at the node
- Larger node ID encountered on the way appeals veto
- Initiator is informed of the veto by the message travelling back and changes from *active* to *passive*



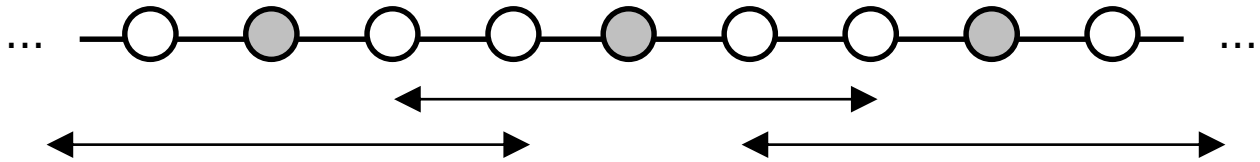
Assessment Worst-Case Message Complexity

1 passive between active processes after phase **1**



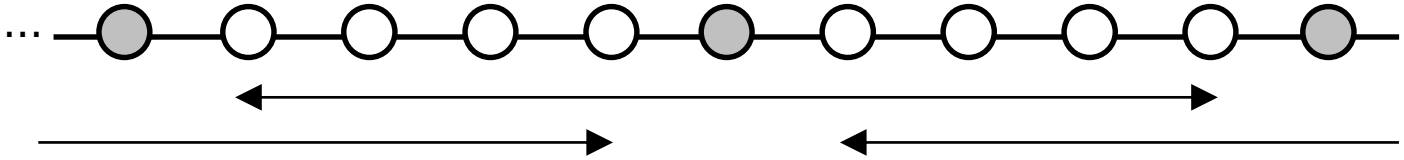
max. $\frac{n}{2}$ survive

2 passive between active processes after phase **2**



max. $\frac{n}{3}$ survive

4 passive between active processes after phase **3**



max. $\frac{n}{5}$ survive

⇒ 2^{i-1} passive between active processes after phase ***i*** *max.* $\frac{n}{(1+2^{i-1})}$ survive

Assessment Worst-Case message Complexity

n processes can initiate chains with the length 1

n / 2 processes can initiate chains with the length 2

n / 3 processes can initiate chains with the length 4

n / 5 processes can initiate chains with the length 8

$\frac{n}{(1+2^{i-1})}$ processes can initiate chains with the length 2i

Each chain with the length 2i generates at most $4 \cdot 2i$ messages

Assessment Worst-Case message Complexity

- In phase i, there are max. $\frac{4 .2^{i}. n}{(1+2^{\mathrm{i}-1})} = \frac{2^{i-1}\, 8n}{(1+2^{\mathrm{i}-1})} < \boldsymbol{8n}$ messages

- There are maximal 1+ ⌈log2 n⌉ phases
- Upper bound is thus 8n + 8n ⌈log2 n⌉) messages
- Worst-case message complexity is O(n log n)
- Again n messages extra for win notification

- Worst-case unit time complexity is 4n – 2 for n = 2k (Best-Case) and 6n – 6 for n = 2k + 1 (Worst-case)

Considering the Phase with the Extinction

- The initiators do not have to proceed the phases synchronously
- An initiator which is already in a high phase can still be stopped by a new larger initiator
- To avoid that, pairs (*phase, node identity*) are used and ordered lexicographically rather than by node identities only
  - Thus, an initiator in a higher phase always prevails against an initiator in a lower phase; only if the phase is the same, the node identity is used as a tie breaker
  - The algorithm is no longer a MAX-Algorithm because it does not necessarily determine the node (or initiator) with the highest ID as the winner

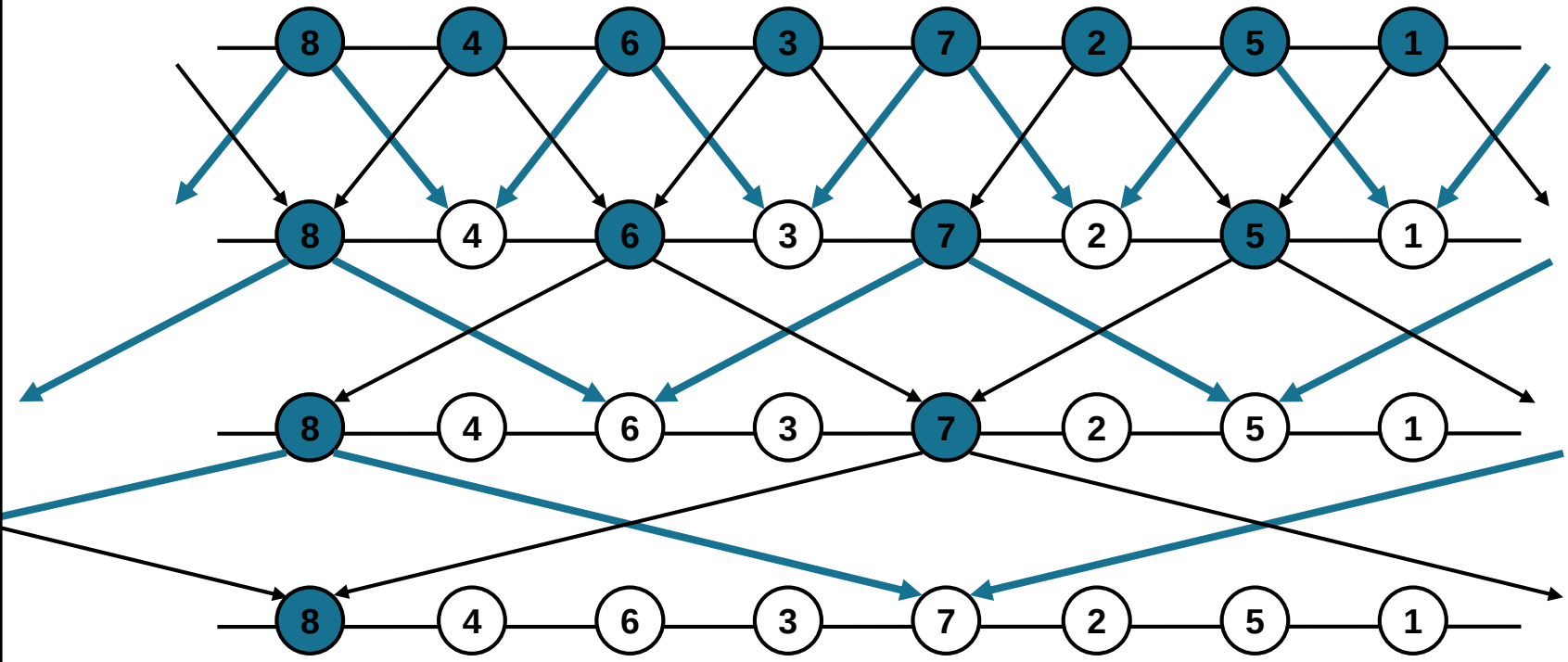
⇒ Not every election algorithm has to be a MAX-Algorithm

Peterson-Election Algorithm

- In the beginning, all processes are active, by and by all processes but one become passive
- The algorithm proceeds in phases
- Course of a phase
  - Each active process communicates its ID to the next active process in both directions
  - Processes receiving a higher ID become passive and only pass messages on
  - Only active processes participate in the next phase
- A node wins if it receives its own ID

- Additionally: A circulation for the win notification of all nodes with *n* messages, again ignored in the following calculations

Worst-Case Message Complexity

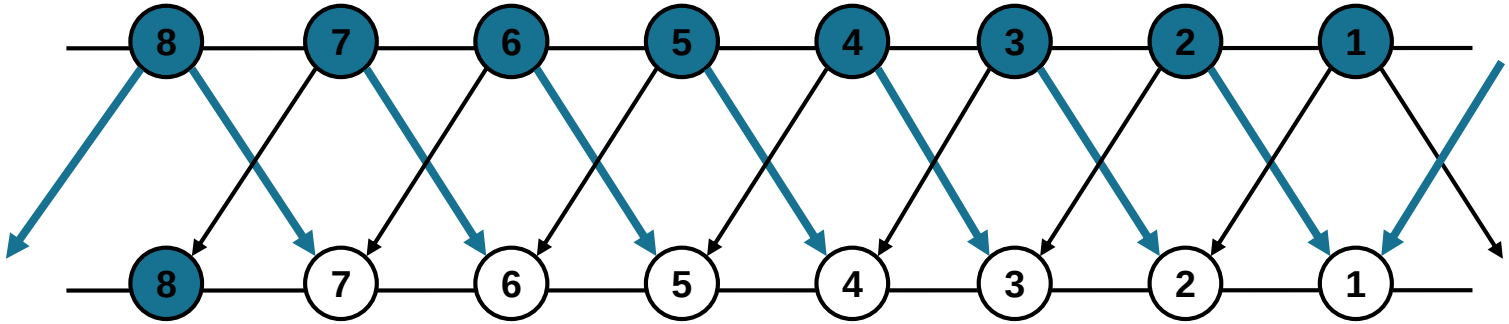


Worst-Case Message Complexity

- In the beginning, all processes are active
- If a node survives, both of its active neighbors do not survive
- In each phase the number of active processes is at least halved → maximal $\lfloor ld\ n \rfloor + 1$ Phases
- At most $2n$ messages per phase because each node sends at most 2 messages per phase
- Worst-Case is $2n\ (\lfloor ld\ n \rfloor + 1)$

Best-Case Message Complexity

- The best arrangement sorts the nodes
- Each node except for the largest is extinct by one of its neighbors in the first phase
- Termination after at most two phases with 2*n* messages each
  → At most *4n* messages



Best-Case Message Complexity

- The best-case occurs if the messages reach the nodes in a certain order
- The message sent from the largest to the smallest node circulates the ring completely if it reaches each node after the message of its other neighbor (the node is already passive then) → n messages
- All other messages go only to their respective neighbor → $2n - 1$ messages
- Then, the algorithm terminates after only one phase

⇒ Best-Case message complexity is $3n - 1$ messages

Average-Case Message Complexity

ID	1. not higher	2. not higher	both not higher
1	0	0	0
2	1/(*n*-1)	0	0
3	2/(*n*-1)	1/(*n*-2)	2/((*n*-1)(*n*-2)
4	3/(*n*-1)	2/(*n*-2)	6/((*n*-1)(*n*-2)
…	…	…	…
n-2	(*n*-3)/(*n*-1)	(*n*-4)/(*n*-2)	(*n*-4)(*n*-3)/((*n*-1)(*n*-2)
n-1	(*n*-2)/(*n*-1)	(*n*-3)/(*n*-2)	(*n*-3)(*n*-2)/((*n*-1)(*n*-2)
n	(*n*-1)/(*n*-1)	(*n*-2)/(*n*-2)	1

To derive the expected value for the survival of a node the sum of those values is divided by $n(n-1)(n-2)$

$$\sum_{i=1}^{n}(i-1)(i-2)=\sum_{i=1}^{n}(i^2-3i+2)=\frac{n(n+1)(2n+1)}{6}-3\frac{n(n+1)}{2}+2n$$

$$=\frac{n(n+1)(2n+1)}{6}-\frac{9n(n+1)}{6}+\frac{12n}{6}=\frac{n(n+1)(n-4)+6n}{3}=\frac{n(n^2-3n+2)}{3}=\frac{n(n-1)(n-2)}{3}$$

⇒ A node survives a phase with the probability 1/3

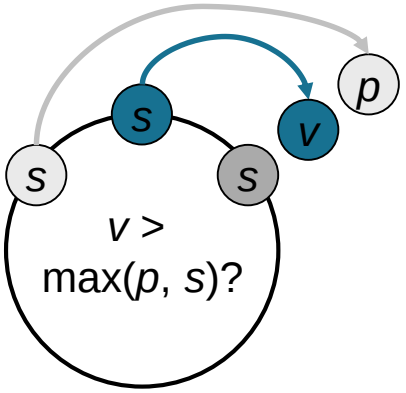
Average-Case Message Complexity

- An active node survives the next phase with probability 1/3
- Thus, $\lfloor log_3\ n \rfloor + 1$ phases on average

$\Rightarrow$ Average-Case is $2n\ (\lfloor log_3\ n \rfloor + 1)$

Peterson-Election Alg. – Unidirectional Variant

- With the bidirectional variant, an active node compares its value with the value of the next active predecessor and the value of the next active successor
  - If it has the largest ID, it remains active
  - Otherwise, it becomes passive
- But: On unidirectional rings messages can be only sent forward
- Solution
  - The IDs of the active predecessor and that of the node are transmitted to the active successor and stored in the variables *v* and *p*
  - The comparison of the values with the own ID *s* is carried out by the successor
  - If $v > \max(p, s)$ applies, it remains active and takes
  - part in the next phase with the ID of its predecessor (*v*)

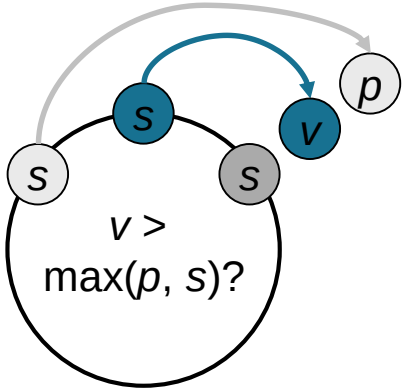


Peterson-Election Alg. – Unidirectional Variant

```
s := id;
do forever begin
   send(s);
   receive(v);
   if v == id then goto leader;
   send(v);
   receive(p);
   if p == id then goto leader;
   if v > max(p, s) then s := v else goto relay;
end

relay:
do forever begin
   receive(s);
   if s == id then goto leader;
   send (s);
end
```

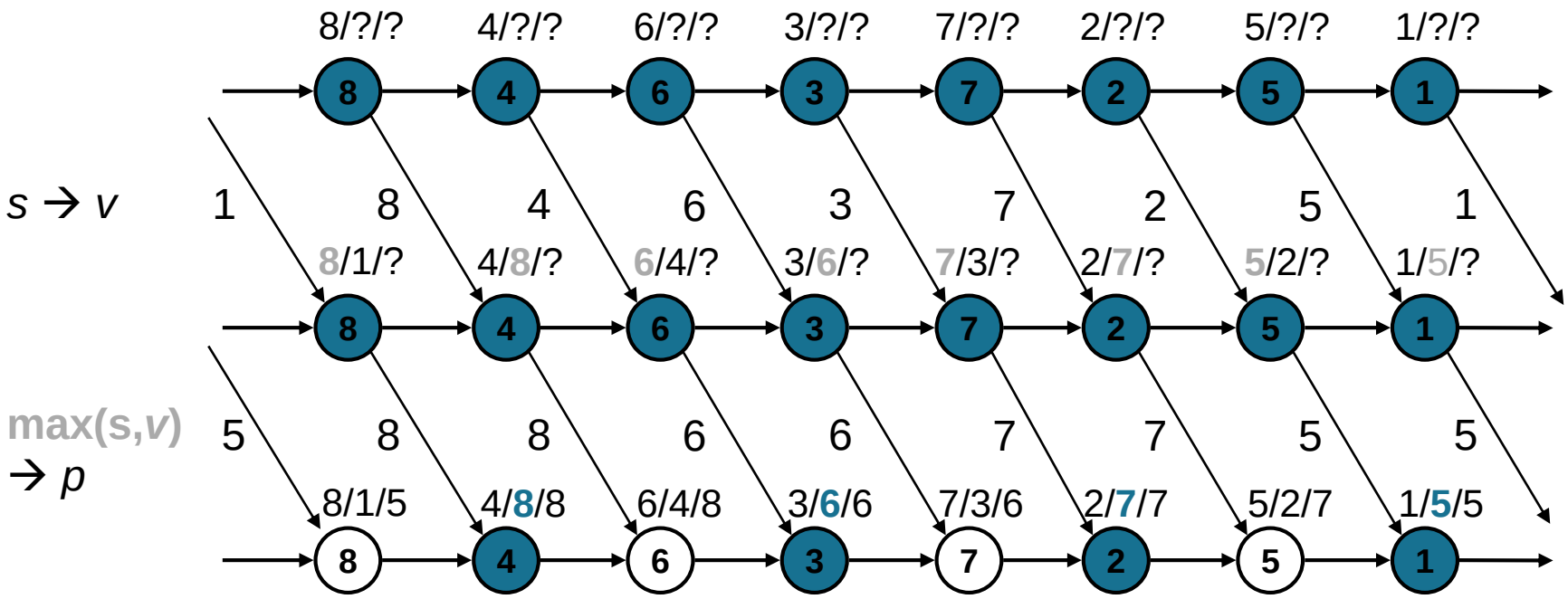
A node receives:
The *s* of its predecessor in *v*
The *s* of its prepredecessor in *p*



Messages must not overtake each other or have to be marked to assure a correct assignment

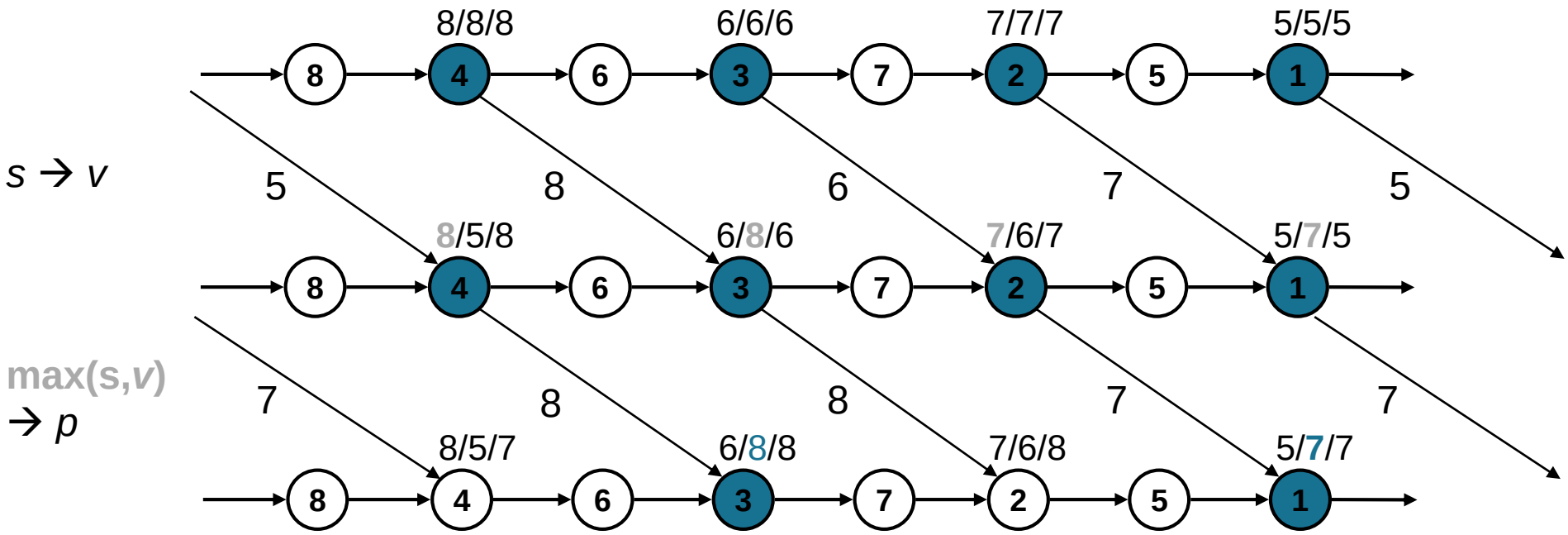
To also assure correct termination for $n = 2$, in line 6 the maximum of *s* and *v* has to be sent and in line 9 „>“ has to be substituted by „≥“ .

Example for Unidirectional Variant



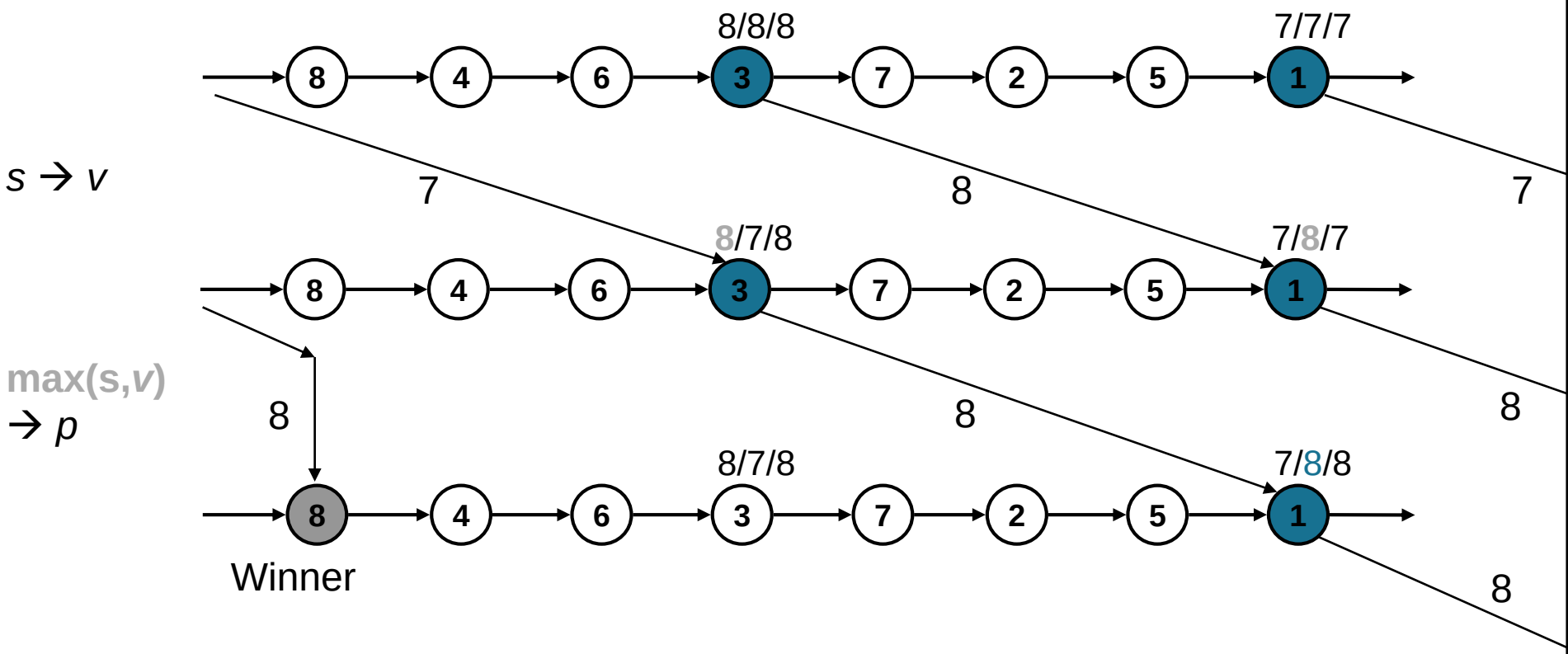
Phase 1 *s* / *v* / *p* ***v*** ≥ max(***p***, ***s***) → ***s*** := ***v***

Example for Unidirectional Variant



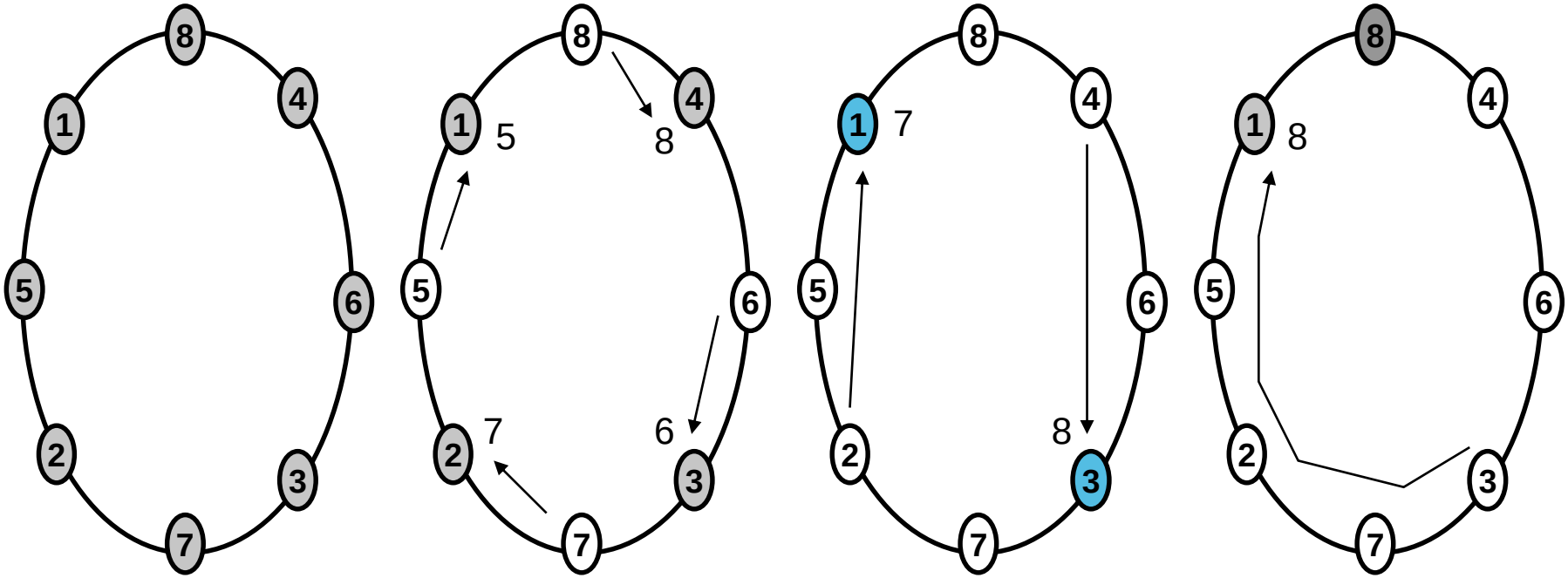
Phase 2 *s* / *v* / *p* ***v*** **≥ max(*p*, *s*) → *s* := *v***

Example for Unidirectional Variant



Phase 3 *s* / *v* / *p* ***v*** **≥ max(*p*, *s*) → *s* := *v***

Example for Unidirectional Variant



Election Algorithms for Trees

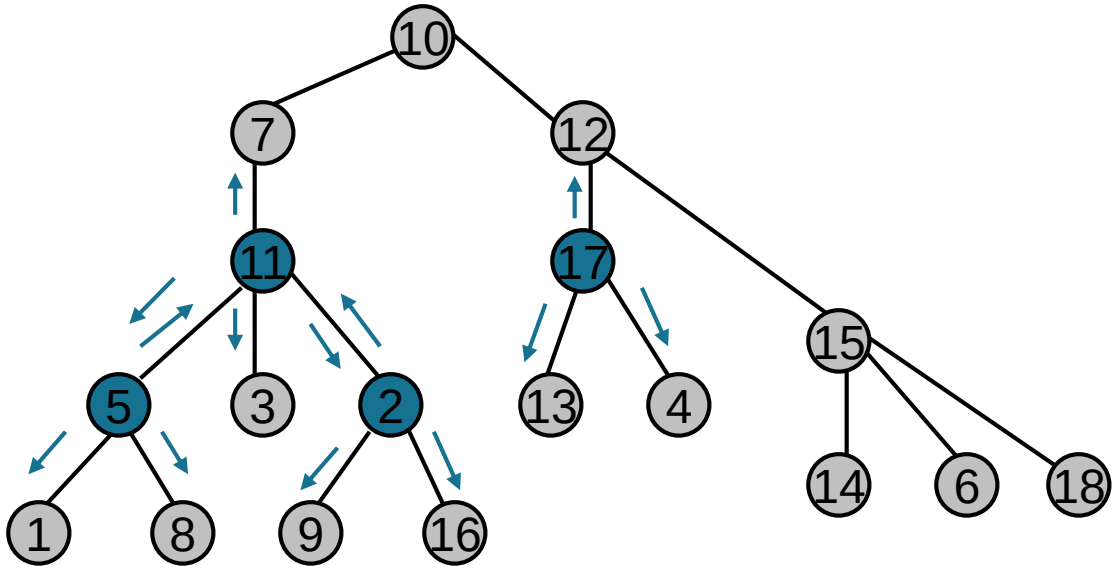
Election Algorithms on Trees

Three phases

1. **Explosion phase**
   - Election request is propagated to the leafs
2. **Contraction phase**
   - From the leafs, the maximum of the already collected identities is propagated to the center
3. **Information phase**
   - Distribution of the real maximum from the center to all nodes in the network

Explosion Phase

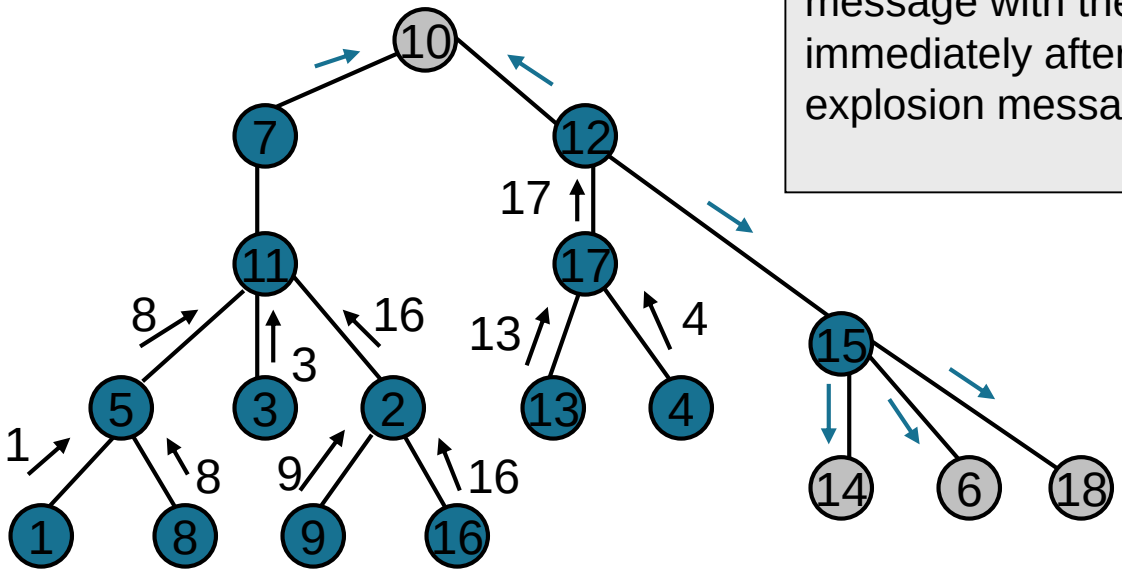
- The explosion starts at several nodes (initiators)
- With the first receipt of an explosion message, the message is passed on to all other neighbors
- The explosion waves unite, where explosion messages meet on an edge



Contraction Phase

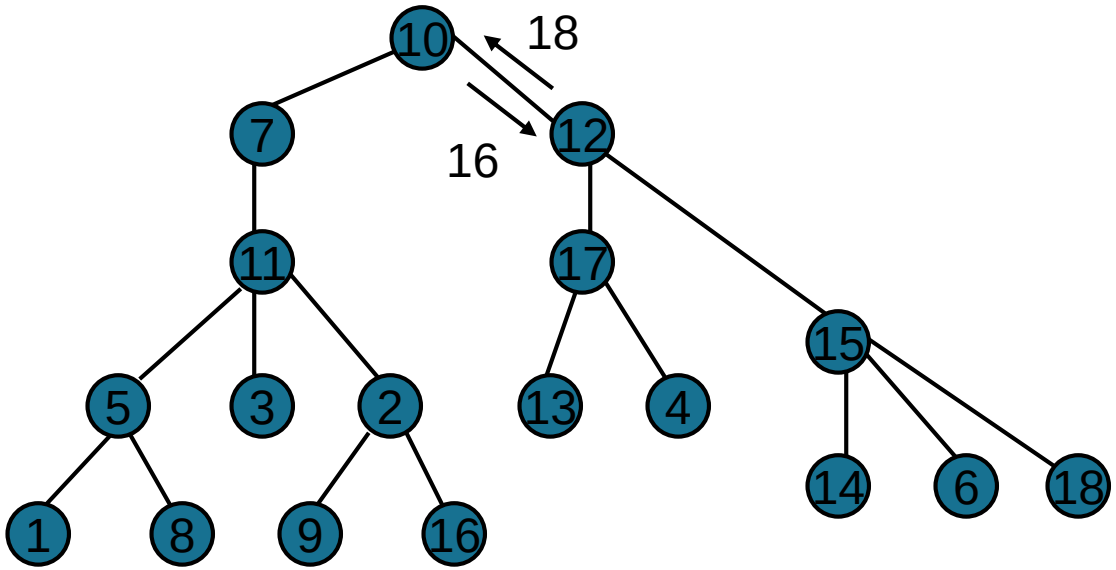
- Leafs answer an explosion message immediately with their own identity
- All other nodes send the maximum of their own identity and the identities received over the other edges over the last remaining edge

Leafs that are initiators also send their contraction message with their ID immediately after the explosion message .



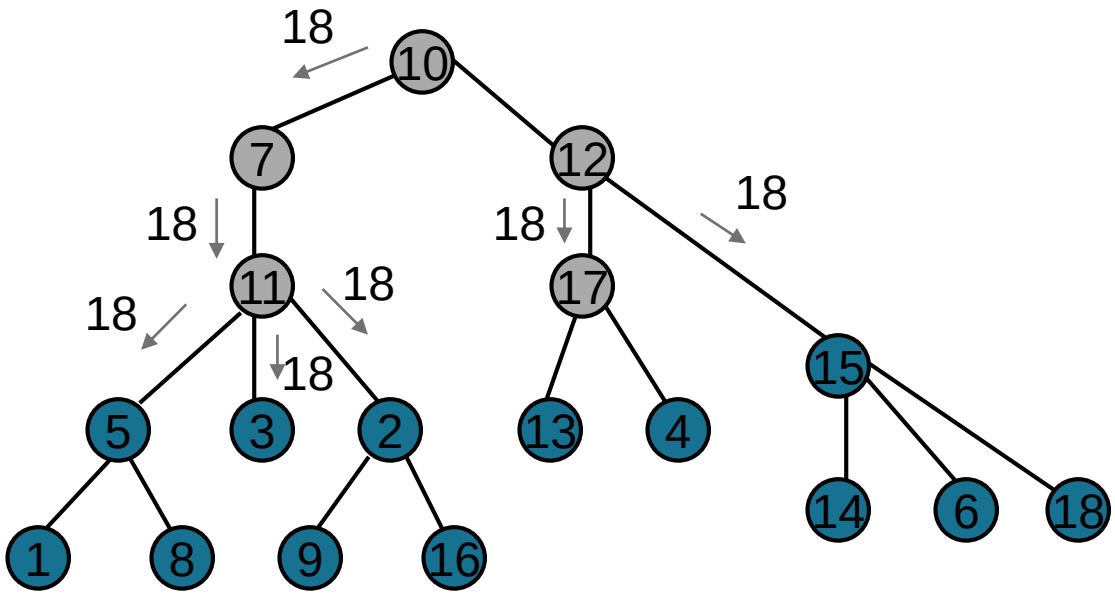
End of the Contraction Phase

On *exactly* one edge two contraction messages meet with *different* maxima
Both received nodes know the real maximum afterwards



Information Phase

From both nodes the maximum is flooded into the network; the edge between them is omitted



Message Complexity with k Initiators

- Explosion Phase: $(n - 1) + (k - 1) = n - 2 + k$
  - One message over each edge
  - Exception: 2 messages over the $k - 1$ meeting edges
- Contraction Phase: $(n - 1) + 1 = n$
  - One message over each edge
  - Exception: Two messages over the central edge
- Information Phase: $(n - 1) - 1 = n - 2$
  - One message over every edge
  - Exception: No message over the central edge
- Altogether $3n + k - 4$ messages

⇒ Election on trees is much more efficient than on rings!

Randomized Election Algorithms

Randomized Algorithms

- Influence their course *deliberately* with random numbers
- Are, thus, non-deterministic algorithms
- Can be determined or not determined
- Are often more simple than deterministic algorithms that solve the same problem
- Through randomized algorithms some problems can be solved more efficiently (or at all)

Important Categories of Randomized Algorithms

- **Las Vegas-Algorithms**

  Weakening of the termination

  – Always provide a correct result, but the worst-case time complexity is unlimited

  – The limit of the termination probability is 1 if the run time approximates infinity

- **Monte Carlo-Algorithms**

  Weakening of the partial correctness

  – Worst-case run time is limited but they sometimes provide a wrong result

Randomized Election in Bidirectional Rings

```
I_p: {M_p == 0}
   M_p := p;
   Elect arbitrarily with probability of 0.5 each one of
   the directions;
   SEND <M_p> TO next node in elected direction;


R_p: {A message <j> has arrived}
   IF M_p < j THEN
      M_p := j;
      SEND <M_p> TO next node in elected direction;
   FI
   IF j == p THEN
      "I am the master"
       <Inform all by additional ring circuit>;
   FI
```

This slide shows the randomized variant of the Chang/Roberts-Algorithm for bidirectional rings.

Randomized Election in Bidirectional Rings

- Average-case message complexity for $k = n$ is

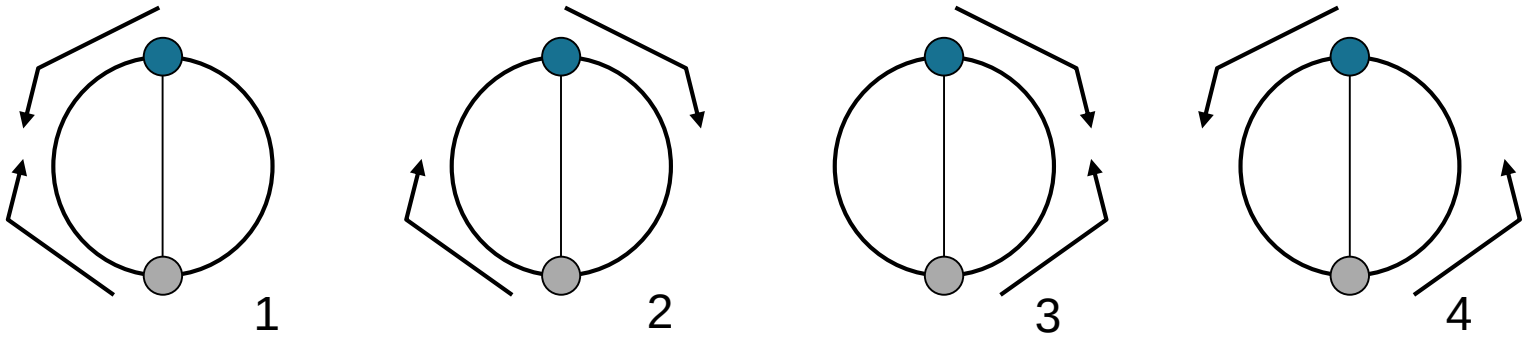
  $0.5\sqrt{2}\, n \ln n \approx 0.71\, n \ln n$ (Lavault, 1990)

- That is about 30% better than the deterministic algorithm for unidirectional rings by Chang and Roberts!

- Why?

Randomized Election in Bidirectional Rings

- We consider the simple case of two initiators assuming equal message delays
- In half of the cases (Figure 1 and 3 below), a higher message approaches a lower message
- In this case, only half as much messages are needed on average (*n*/4 instead of *n*/2)
- Therefore, on average 3/8 *n* messages instead of 1/2 *n*.
  → 25% saving

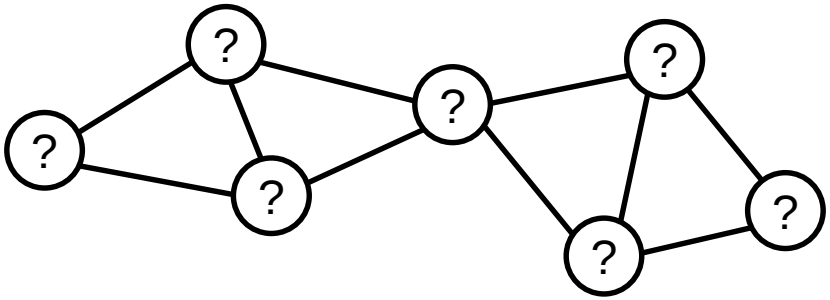


Randomized Election in Bidirectional Rings

- For large *n*, the fact that the highest message has to make a whole round can be ignored
- For more than two initiators the consideration can be applied to all pairs formed from the largest and one of the other initiators
- This explains 25% savings.
- The remaining 5% result from the fact that if more than two initiators are involved, messages of a “medium” initiator can also erase message of a “smaller” initiator

Election in Anonymous Networks

- In anonymous networks, nodes do not have permanent unique identities
- Is it possible to determine a unique winner of the election, then?
- If so, under which conditions?



Las Vegas-Election for Anonymous Rings

- Assumption: Ring size n is known
- Algorithm of Itai a. Rodeh is based on the algorithm of Chang and Roberts
- Each node is initiator and arbitrarily elects a temporary identity from the numbers 1, 2, …, c with $c \geq 2$
- Message extinction as usual but messages contain
  - a hop-count h which is 1 initially and is incremented by each relaying node
  - a flag f which has the value 1 initially when sending
  - a variable with the number of the corresponding election round

Las Vegas-Election for Anonymous Rings

- If a node receives a message with its own identity two cases are distinguished
  - If *h* != *n*, there is at least one other node with the same identity. Thus, *f* is set 0 and the message is relayed
  - If *h* = *n,* it won the election
    - If *f* = 1, it is the only winner and can send the win message
    - If *f* = 0, there are several winners

Las Vegas-Election for Anonymous Rings

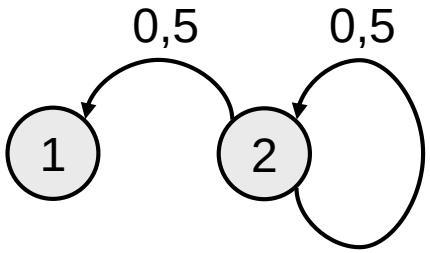
- Elections are carried out iteratively until a unique winner is determined
  - Only the winners of the recent election participate actively in the next election. Therefore, each winner elects a new, arbitrary identity and sends it around the ring
  - Eliminated nodes are passive and only relay messages (with incremented Hop-Count)
  - Messages from smaller elections are swallowed

- Expected value E for the number of elections for $c = n$ is bounded by
  $\mathcal{E}\,(n / (n - 1))$ ($\mathcal{E}$ is Euler's number)

- A generalization of the algorithm on general networks is (with application of the echo-election algorithm) is possible

Derivation of the Expectation Value for $n = c = 2$

Application of a Markov-Chain



State (Number in Circle) represents number of Leaders

$$E = \sum_{i=1}^{\infty} i \cdot 2^{-i}$$

$$= 1 \cdot 2^{-1} + 2 \cdot 2^{-2} + 3 \cdot 2^{-3} + \ldots = 2$$

Derivation is also possible for a general n

Election in Rings with Unknown Size

- Through observation of repetitions or differences of the identities of the visited nodes, the ring size can be estimated by several ring circulations
  - With large rings and large *c* it is probable that one estimates correctly; but what happens if one estimates incorrectly?

Election in Rings with Unknown Size

- Through observation of repetitions or differences of the identities of the visited nodes, the ring size can be estimated by several ring circulations
  - With large rings and large *c* it is probable that one estimates correctly; but what happens if one estimates incorrectly?
    - Then, there are several winners and this is not detected
- What amount of minimal (structural) information is necessary to break the symmetry deterministically or randomized?

Election in Rings with Unknown Size

- Through observation of repetitions or differences of the identities of the visited nodes, the ring size can be estimated by several ring circulations
  - With large rings and large *c* it is probable that one estimates correctly; but what happens if one estimates incorrectly?
    - Then, there are several winners and this is not detected
- What amount of minimal (structural) information is necessary to break the symmetry deterministically or randomized?
  - The unknown ring size is a prime number
  - The ring size lies between *N* and *2N-1*
  - ...
- Neither a deterministic nor a Las Vegas-Algorithm for the election in anonymous rings with unknown size exists!

Literature

1. E. Chang and R. Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes. Communications of the ACM (CACM), 22(5):281--283, 1979.
2. D. S. Hirschberg and J. B. Sinclair. Decentralized extrema-finding in circular configurations of processors. Communications of the ACM (CACM), 23(11):627--628, 1980.
3. Gary L. Peterson. An O(n log n) unidirectional algorithm for the circular extrema problem. ACM Transactions on Programming Languages and Systems (TOPLAS), 4(4):758--762, 1982.
4. C. Lavault. Average number of messages for distributed leader finding in rings of processors. Information Processing Letters, 30:167--176, 1989.
5. A. Itai and M. Rodeh. Symmetry breaking in distributed networks,. In Proceedings of the 22nd IEEE Symposium on Foundations of Computer Science, pages 150--158. IEEE Press, 1981.
6. Friedemann Mattern. Verteilte Basisalgorithmen. Springer-Verlag, 1989. Kapitel 2: Untersuchung von Election-Algorithmen