

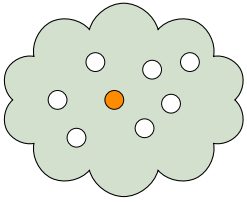
Distributed Operating Systems

9th Chapter
Leader Election

Matthias Werner
Chair Operating Systems

9.1 Introduction

- Many distributed algorithms and applications do not work completely symmetric, but require a distinguished participant
- **Examples**:
  - Initial determination of a token holder (e.g. for ABCAST ➡ Chapter 5)
  - Election of a coordinator for centralized algorithm (e.g. for mutual exclusion ➡ Chapter 8)
  - ...



- If this distinguished participant is not determined *a priori* [1], it has to be determined at runtime
- Algorithms that determine such a distinguished participant are called leader election algorithms

[1] ...what would lead to a *single point of failure*

Problem

Election

1. Each node shall know, whether it is elected or not
2. Only one node shall be elected

- Slightly more formal:
  - Each node is in one of three states: *undecided*, *elected* or *not elected*
  - Initially, each node is in state *undecided*
  - Each node transitions into another state (*elected* or *not elected*) exactly once (final state)

Problem Variants

- Depending on the variant, the solution of the election problem is **trivial** to **impossible**
- e.g.:
  - An algorithm with known identities in a fully meshed system is trivial
  - In an anonymous ring, a deterministic algorithm is impossible
- We consider several variants

Problem Variants (cont.)

- **Identity**: Do the nodes have well-distinguished identities?
  - Yes ➡ **MAX-Algorithms**
  - No ➡ **anonymous algorithms**
- **Number** of participants?
  - Known
  - Unknown
- **Determinism** of the algorithm
  - **Deterministic** algorithms
  - **Randomized** Algorithms
- Other:
  - **Synchronization** (e.g. FIFO-Channels, limited communication times,…)
  - **Topology** (e.g. irregular, fully meshed, ring, tree, torus, …)

9.2 MAX-Algorithms

- MAX-Algorithms are always simple (in contrast to anonymous algorithms)
- Basic Idea:

Each node names its ID to all other nodes, the largest ID wins.

- Hence, we focus on **efficiency**
  - Message count?
  - Time complexity?

Election algorithm for arbitrary topologies

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

R_p: {a message <j> arrived}
   IF M_p < j THEN
      M_p := j;
      SEND <M_p> TO all neighbours;
   FI


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

- Each process has a unique identity p and a local variable M_p which initially is 0
- I_p is executed by initiators
- Once R_p was executed, p may not become initiator anymore ➡ p also cannot win anymore
- ➡ the highest initiator wins

Election algorithm for arbitrary topologies

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

R_p: {a message <j> arrived}
   IF M_p < j THEN
      M_p := j;
      SEND <M_p> TO all neighbours;
   FI

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

But how?

- Each process has a unique identity p and a local variable M_p which initially is 0
- I_p is executed by initiators
- Once R_p was executed, p may not become initiator anymore ➡ p also cannot win anymore
- ➡ the highest initiator wins

Election algorithm for arbitrary topologies (cont.)

- **Problem**: Determination of termination
  - Termination is nontrivial
  - Algorithms for the determination of termination may be used (➡ later chapter)
  - Release at “suspicion” ➡ timeout

- **Complexity**
  - There is maximal one message per initiator and edge
  - Worst case: maximal number of edges and initiators
  - For n nodes: $\mathcal{O}(n^3)$
  - This does not even include the effort to detect termination

Echo Election Algorithm

- Usage of **Echo Algorithm** for election (➡ Chapter “Mutual Exclusion”)
  - Works with arbitrary, connected topologies
  - Each initiator starts an instance of the Echo Algorithm
  - Explorer and echos carry along the **identity** of the initiator
  - Weaker messages (explorer and echos) are **not** forwarded but swallowed ➡ **message cancellation**

Echo Election Algorithm (cont.)

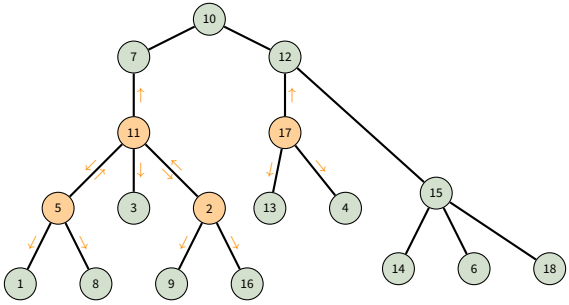
- Strongest wave prevails and **terminates at the winner**
  - Winner knows that he won
  - If an initiator receives a stronger explorer, then he knows that he lost
- How do the **losers** learn of the termination?
  - All other waves stagnate somewhere ultimately
  - At least the strongest initiator does not send an echo for the weaker waves
  - His echo algorithm does not terminate at any other initiator
  - If this is not sufficient, the winner has to inform the others in a **futher** wave
- Worst-case **complexity** is again $\mathcal{O}(n^3)$ (better, if spanning tree already exists)

Modified Echo Algorithm with Tree Topology

- Use tree topology for wave algorithm
- Tree phases
  - **Explosion phase**
    Election request is propagated to the leaves
  - **Contraction phase**
    Maximum of the so far collected identities is propagated from the leaves to the center
  - **Information phase**
    Propagation of the true maximum from the center to all other nodes in the network

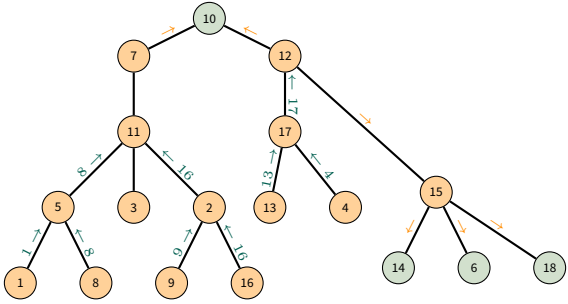
Explosion Phase

- The explosion starts at multiple nodes (initiators)
- When receiving an explosion message for the first time, the message is forwarded to all other neighbours
- The explosion wave concentrates, where explosion messages meet on an edge



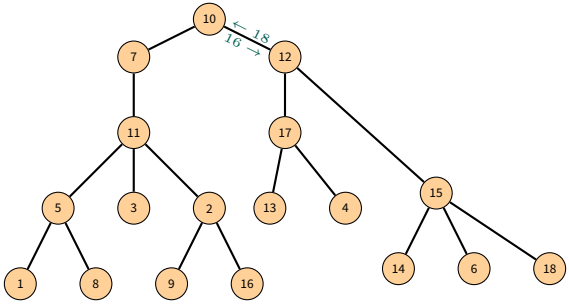
Contraction Phase

- Leaves respond to explosion message with their own identity immediately
- All other nodes send the maximum from the identities received through the other edges and their own identity over the last remaining edge



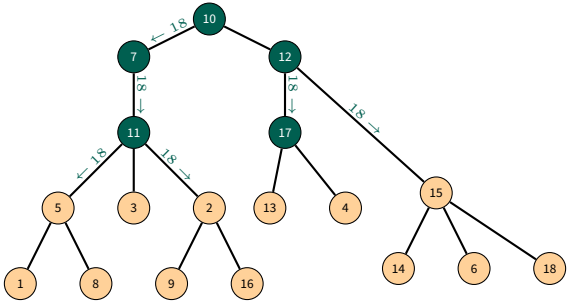
End of Contraction Phase

- On **exactly one** edge, two contraction messages with **different** maxima meet
- Both receiving nodes know the true maximum afterwards



Information Phase

- From both nodes, the maximum is flooded into the network, with the edge between these nodes omitted



Message Complexity

- Explosion Phase: $(n-1)+(k-1)=n-2+k$
  - One message over each edge
  - Exception: 2 messages over each of 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 each edge
  - Exception: no message over the central edge
- $3n+k-4$ message overall (➡ $\mathcal{O}(n)$)

9.3 Election in Anonymous Networks

- In anonymous networks, nodes do **not** have a permanent **unique** identity
- Is it still possible to determine a unique winner of the election?
  - With deterministic algorithms: no
    (as long as a known topology does not provide an implicit distinction)
  - With randomized algorithms: maybe

Impossibility of Anonymous Algorithms

- Consider fully symmetric topology:
  - Anonymous ring (ring size may be known)
  - Synchronous communication
- **Assumption**: There is a **deterministic** algorithm $\mathcal{A}$ that solves the election problem
- **Proof by Contradiction:**
  - Since all participants are identical, all participants of $\mathcal{A}$ start in the same state
  - In each round of $\mathcal{A}$ all participants execute the same steps ➡ are in the same state
  - If someone declares itself the winner, all declare themselves the winner ➡ Contradiction
- **Solution**: probabilistic algorithms

Distinction of Randomized Algorithms

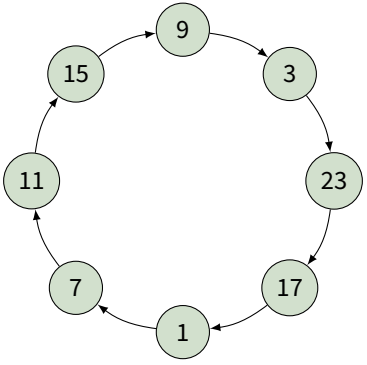
- **Las Vegas Algorithms**
  - **Always** deliver a correct result
  - However, the worst-case time complexity may be **unlimited** ➡ weakening of the **termination requirement**
- **Monte Carlo Algorithm**
  - May with limited probability deliver an **incorrect** result
  - Worst-case execution time is limited
- We consider Las Vegas Algorithms for the election problem

Las Vegas Election for Anonymous Rings

- There are even algorithms for symmetric topologies such as rings
- We consider the algorithm of ITAI and RODEH ➡ based on MAX-algorithm of CHANG and ROBERTS
- Hence, we first consider CHANG and ROBERTS
  - Topology: ring of size n
  - Each node has unique UID
  - Communication in one direction only

Excursion: Chang/Roberts

- Each node is in one of three states: $Z \in$ {Non-Participant, Participant, Winner}
  - Initially all nodes are Non-Participants

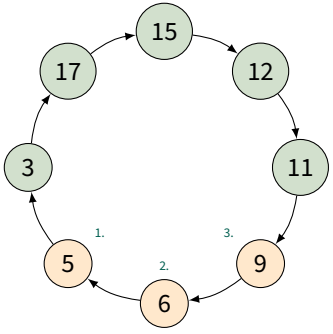


Excursion: Chang/Roberts – Algorithm

- If one (or more) nodes learns that there is no coordinator (timeout), it (or they) declares itself (or themselves) Participants and sends (or send) an election message with its (or their) UID
- After receiving an election message a node reacts as follows:
  - If the message UID is bigger than the own UID the node becomes Participant (if that is not the case) and the message is **forwarded**
  - If the message UID is smaller than the own UID and the receiving node is not Participant, the receiver becomes Participant and **forwards** the message after replacing the messge UID with its own
  - If the message UID is smaller than the own UID and the receiving node isParticipant the message is **ignored**
  - If the message UID is equal to the own UID the receiving node declares itself Winner
- A Winner sends a winner-message with its UID that is forwarded by all other nodes
  - If the Winner receives its own winner-message the algorithm terminates

Excursion: Chang/Roberts – Worst-case Message Complexity

- **Assumption**: k initiators
- A worst case is if the initiators are placed in decreasing order around the ring and the selection is initiated in decreasing order
  - k-biggest initiator
    ➡ $n - (k - 1)$ messages
  - ...
  - 3.-biggest initiator
    ➡ $n - 2$ messages
  - 2.-biggest initiator
    ➡ $n - 1$ messages
  - biggest initiator
    ➡ n messages



Excursion: Chang/Roberts – Worst-case Message Complexity (cont.)

- Message complexity for k initiators

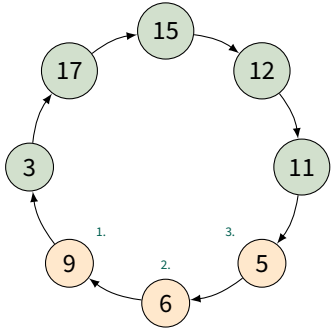
$$\begin{aligned} N &= n + (n-1) + (n-2) + \cdots + (n-(k-1)) \\ &= \frac{n(n-1)}{2} - \frac{(n-k)(n-k+1)}{2} \\ &= \frac{2nk - k^2 + k}{2} \\ &= nk - \frac{k(k-1)}{2} \end{aligned}$$

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

- ➡ $\mathcal{O}(n^2)$ for a ring of size n and $k = n$
- n additional winner-messages

Exkurs: Chang/Roberts – Best-Case Message complexity

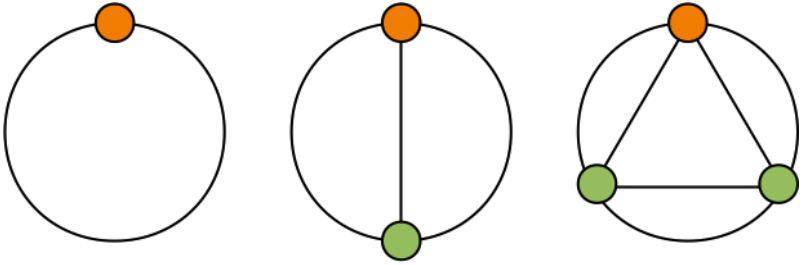
- Best case is if the initiators are in ascending order placed around the ring
- all but the biggest initiator contribute just one message
- the biggest initiator contributs n messages ➡ $n + k + 1$ messages for selection
- Again n additional winner-messages



Excursion: Chang/Roberts: Average Complexity

- Message complexity for all $n!$ permutations of numbers around the ring
- Informal perspective
- biggest initiator ➡ n messages (always)
- 2.-biggest initiator ➡ $\frac{n}{2}$ messages on average
- 3.-biggest initiator ➡ $\frac{n}{3}$ messages on average
- ...

➡ i.-biggest initiator ➡ $\frac{n}{i}$ messges on average



● current initiator ● biggest initiator

Excursion: Chang/Roberts: Average Complexity (cont.)

- Therefore the average-case message complexity is

$$nH_k \approx n \ln k$$

  with $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k}$
  - For unidirectional rings this is optimal
  - H_k is the kth **harmonic number** (➡ **harmonic series**)

- For huge rings there is nearly all the time no more than the average number of messages needed (Rotem et al.)
- Again: n additional winner-messages

Itai and Rodeh

Back to Itai and Rodeh

- Assumption:
  - Anonymous ring: no nodes has a (permanent) identity
  - Size of the ring n has to be known
  - Each node has a random number generator ➡ randomization
- **Differences** to Chang/Roberts:
  - Algorithm works in rounds each of which correlates with a selection in Chang/Roberts
  - Message contains ID, hopcount h and a flag f

Itai and Rodeh (cont.)

- (Each) initiator chooses a **random** ID and sends a message <ID,$f = 1$,$h = 0$>
- Each node increments the hopcount $h \leftarrow h + 1$ if it forwards a message
- If a node receives a message with its own identity there are two cases
  - If $h \neq n$ then there is at least one other node with the same identity
    - f is set to 0 and the message is forwarded
  - If $h = n$ then the node is the winner
    - If $f = 1$ then the node is the only winner and it can send its winner-message
    - If $f = 0$ then there is more than one winner

Itai and Rodeh (cont.)

- Selections are done till a unique winner is identified
  - Just the **winner** of the current selection take part in the next round
    ➡ each winner chooses a **new** random identity and sends it around the ring
  - Eliminated nodes are **passiv** and just forward all messages (still incrementing the hopcount)
  - Messsages from previous selections are **swallowed**

Itai and Rodeh – Discussion

- There is no guarantee that the algorithm terminates
- But: the **probability** of **not** terminating tends to 0
- Limit[2] of the expectation E of the number of selections with a number $c = n$ of possible identities is

$$E \leq e\frac{n}{n-1}$$

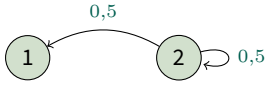
(e is Euler's number)

[2] This is an estimation. The limit is $E \leq 2{,}441715 \cdot n$

Itai and Rodeh – Discussion (cont.)

Derivation of the expectation for $n = c = 2$

- Application of Markow-Chains



$$
\begin{aligned}
E &= \sum_{i=1}^{\infty} i \cdot \frac{1}{2^i} \\
&= 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2^2} + 3 \cdot \frac{1}{2^3} + \cdots \\
&= 2
\end{aligned}
$$

- Derivation is also for general n possible.

Itai and Rodeh – Discussion (cont.)

- Knowledge of the size of the ring n is a strong requirement
- **Weakening** Size of the ring is *roughly* known : $\exists k, k \leq n < 2k$
  - Messages with bigger of equal ID are forwarded till $h = 2k - 1$
    - at least one round is guaranteed
  - The flag f is deleted if a message with the own ID and a hopcount of $h < k$ is received
    - If there is more than one winner, **at least one** node knows about it and tells the others

Alorithm for arbitrary graphs

- The following algorithms is from Dolev et al [DIM97] zurück
- It solves the selection-problem for arbitrary (connected) topologies
- The algorithm is dynamic and **self-stabilizing**:
  - It **terminates never** and after a failure (breakdown of a node or an edge) it will at some timepoint return to a correct state.
- **Idea:** Construction of an directed tree (*in-tree*) whose root is the current node
- **Assumption:**
  - A node can locally identify its neighbors via the communication channels
  - A node has access to variable of its neighbors (RPC); if not the communication channel is seen as broken (non-existent)

Alorithm for arbitrary graphs (cont.)

- Each node is either the **root** of a tree or it has a **parent node**
  - There is a directed edge from child to parent in the tree.
- Ech node has the following variables that are accessible by its neighbors:
  tid: the ID of the tree to which it belongs
  d: distance of the node to the root (for the root: $d = 0$)
  c: a color (value from $[0, \ldots, 7]$)
- Each node does the following steps in an endless loop
  - If a tree with bigger tid is found the node joins this tree ➡ subalgorithm **tree fusion**
  - If an error is identified the node declares itself to the new root ➡ subalgorithm **error detection**
  - If other trees with the same tid are found the node increases the tid of its own tree by a random number ➡ subalgorithm **recognition of competition**

Alorithm for arbitrary graphs (cont.)

- **tree fusion / error detection**
  - request information from all neighbors i and compute $tid_{max} = \max_i(tid_i)$ and $d_{min} = \min_{tid_i = tid_{max}}(d_i)$
  - If tid_{max} is smaller than the own tid or if they are equal but d_{min} is bigger than the own distance d, the node declares itself root
  - If that is not the case the node with $tid_i = tid_{max}$ is the (possibly new) parent whose tid is inherited; the distance is set to $d = d_i + 1$

- **Assumption:**
  - If a node was the root before, the number of trees is reduced
  - By checking the distances loops and disappearing edges are found
  - After enough runs without an error there is a forest whose trees each have the same tid

Alorithm for arbitrary graphs (cont.)

- **recognition of competition**
  - A root chooses a random color and initiates a recoloring of the tree (wave-procedure)
  - As soon as all elements of a tree finished recoloring the root initializes the next recoloring
  - If a node finds a neighbor with the same tid or a color that is neither the current color nor the tree color from before, this node notifies a competition the root
  - In this case the root increases the tid by a random value and propagates this new tree-ID in its tree
- **Remark**
  - A recoloring starts after the recoloring before ends. Hence a color that is different from the current color and the color before indicates a foreign tree.
  - The color is chosen at random and hence the probability of two roots choosing the same color decreases.
  - A competing tree can initiate an increase in its tid. Here again a random value is chosen and hence the probability of equal IDs decreases further.

Discussion

- The algorithm is described informally, for a formal perspective and proofs on correctness please refer to [DIM97]
- If the final tree has a depth of D and each of the n nodes maximally N neighbors than in case of no errors the number of rounds till exactly one root is selected is $\mathcal{O}(N \cdot D \cdot \log n)$
- Termination of the algorithm cannot be detected if the number of nodes is unknown

Other approaches for breaking symmetry

- If the topology is known and not symmetric, this can be used for breaking symmetry for anonymous nodes
  - Example: in a grid one can look for a node in a certain corner
- In practice anonymous systems are common in networks of sensor nodes
- In this case the sensorinformation can be used to break symmetry
  - GPS-coordinates
  - Realtive position of other nodes

Literature

[Mat89] Mattern, F.: *Verteilte Basisalgorithmen*. Springer, 1989 , Kapitel 2

[ChRo79] Chang, E. and R. Roberts: “An improved algorithm for decentralized extrema-finding in circular configurations of processes”. *Communications of the ACM*, 22(5)1979, 281–283

[DIM97] Dolev, S., A. Israeli and S. Moran: “Uniform Dynamic Self-Stabilizing Leader Election”. *IEEE Transactions on Parallel and Distributed Systems*, 8(4)1997, 424–440