Leader Election
Rings, Arbitrary Networks

Smruti R. Sarangi

Department of Computer Science
Indian Institute of Technology
New Delhi, India
1 Leader Election in Rings
   - $O(n^2)$ Algorithm
   - $O(n \log(n))$ Algorithm

2 Leader Election in Trees
1. Leader Election in Rings
   - $O(n^2)$ Algorithm
   - $O(n \log(n))$ Algorithm

2. Leader Election in Trees
We assume that we are using a ring based overlay.

We wish to choose the process with the smallest id as the leader. (NOTE: asymmetry)

Messages can only be sent to the clockwise neighbor (left) or anti-clockwise neighbor (right).
Basic $O(n^2)$ Algorithm

1 if $p$ is initiator then
   2 state ← find
   3 send $p$ to next($p$)
   4 while state $\neq$ leader do
      5 receive($q$);
      6 if $p = q$ then
         7 state ← leader
      7 end
      8 else if $q < p$ then
         9 if state = find then
            10 state ← lost
         10 end
         11 send $q$ to next($p$)
else

while true do

    receive q
    send (q) to next(p)

    if state = sleep then
        state = lost
    end

end

end
Analysis

### Message Complexity
- Assume there are $O(N)$ initiators.
- The leader’s message will be sent $N$ times.
- For other initiators, the message will be sent $N - i$ times.
- $\sum_i (N - i) = O(N^2)$. 
Message Complexity

- Assume there are $O(N)$ initiators.
- The leader’s message will be sent $N$ times.
- For other initiators, the message will be sent $N - i$ times.
- $\sum_i(N - i) = O(N^2)$.

Optimization

Global broadcast is not necessary.
Outline

1. Leader Election in Rings
   - $O(n^2)$ Algorithm
   - $O(n \log(n))$ Algorithm

2. Leader Election in Trees
Basic Idea

- We get a $O(N^2)$ complexity because each message can travel $O(N)$ hops.
- Instead of sending a message to everybody, we need to find a way to filter the set of messages (similar to Maekawa’s algorithm).
- We will consider gradually larger sizes of windows in a sequence of rounds.
- Each window will allow only one of its members to participate in the next round.
- If, we are able to filter the number of participating members by a factor of 2 in each round, we will have $O(\log(N))$ rounds.
- If in each round, we send only $N$ messages, then a total of $O(N \log(N))$ messages need to be sent.
**O(n log(n)) Time Algorithm**

1. **initialize:**
   send (probe, id, 0, 1) to left and right

2. **receive (probe, j, k, d) from left(right):**
   if \( j = id \) then
   leader ← j
   Terminate

3. end

4. if \( j < id \) and \( d < 2^k \) then
   send (probe, j, k, d+1) to right (left)

5. end

6. if \( j < id \) and \( d = 2^k \) then
   send (reply, j, k) to left (right)

7. end
Leader Election in Rings
Leader Election in Trees

\(O(n^2)\) Algorithm
\(O(n \log(n))\) Algorithm

\(O(n \log(n))\) Time Algorithm - II

1. receive \((\text{reply}, j, k)\) from left(right):
   - if \(j \neq \text{id}\) then
     2. send \((\text{reply}, j, k)\) to right(left)
   end
   else
     5. if received \((\text{reply}, j, k)\) from right(left) then
       6. send \((\text{probe}, \text{id}, k+1, 1)\) to left and right
     end
   end

Smruti R. Sarangi
Leader Election 11/17
The maximum number of winners after $k$ phases is:
- To winners can at the least be $2^k$ entries apart.
- Thus, the total number of winners after $k$ phases is $n/(2^k+1)$

The total number of messages for each initiator in phase $k$ is $4 \times 2^k$

Total number of messages in the $k^{th}$ phase is:

$$4 \times 2^k \times \frac{n}{2^{k-1} + 1}$$

Total number of messages is:

$$M = \sum_{k=1}^{\log(n)} 4 \times 2^k \times \frac{n}{2^{k-1} + 1} = O(n \log(n)) \quad (1)$$
Let us consider arbitrary networks.

Creating a ring based overlay is difficult (It amounts to constructing a Hamiltonian cycle – NP Hard).

However, creating a tree based overlay is easy.

To further optimize the process, we can choose the MST (minimum spanning tree) as the overlay.

Assumptions:
- Let the current node be termed as $p$
- Let a neighbor be termed $q$
- All the leaves (degree=1) are initiators
/* Wakeup all the nodes */

1. if p is an initiator then
2.   awake ← true
3.   foreach q ∈ neigh(p) do
4.     send wakeup to q
5.   end
6. end
7. while numWakeups < | neigh(p) | do
8.   receive( wakeup )
9.   numWakeups ← numWakeups + 1
10. if awake = false then
11.   awake ← true
12.   foreach q ∈ neigh(p) do
13.     send wakeup to q
14. end
15. end
Send Proposal to Parent

/* Collate result from the leaves and send to parent */

1 while received > 1 do
2 receive < r > from q
3 rec_p[q] ← true
4 received ← received + 1
5 min_p ← min(min_p, r)
3 end
4 send min_p to parent such that rec_p[parent] = false
/* Receive the result from the parent, and send to the leaves */

1 receive \(< r >\) from parent
2 \(\text{res} \leftarrow \min(\min_p, < r >)\)
3 \text{if } \text{res} = p \text{ then}
4 \quad \text{state} \leftarrow \text{leader}
5 \text{end}
6 \text{else}
7 \quad \text{state} \leftarrow \text{lost}
8 \text{end}
9 \text{foreach } q \in \text{neigh}(p), q \neq \text{parent} \text{ do}
10 \quad \text{send } \text{res} \text{ to } q
11 \text{end}
Analysis

Message Complexity

- On every edge, we can send at the most two wakeup messages.
- We can send a proposal and its reply.
- A tree with $N$ nodes as $(N - 1)$ edges.

Complexity

Message Complexity: $4N - 4 = O(N)$
Leader Election in Rings
Leader Election in Trees

Introduction to Distributed Algorithms by Gerard Tel, Cambridge University Press, 2000

Distributed Computing, Fundamentals, Simulations and Advanced Topics by Haggit Attiya and Jennifer Welch, Wiley 2004


Distributed Algorithms by Nancy Lynch, Morgan Kaufmann, 1996