Leader Election

Rings, Arbitrary Networks

Smruti R. Sarangi

Department of Computer Science
Indian Institute of Technology
New Delhi, India

Outline

Outline

Leader Election in Rings

- 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
       send p to next(p)
       while state ≠ leader do
3          receive(q);
           if p = q then
4              state ← leader
           end
5          else if q < p then
6              if state = find then
7                  state ← lost
8              end
9              send q to next(p)
10
```

Basic $O(n^2)$ Algorithm - II

```
1  else
2      while true do
3          receive q
           send (q) to next(p)
           if state = sleep then
4              state = lost
           end
5          end
6      end
7  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.
- $\sum_i (N - i) = O(N^2)$.

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.
- $\sum_i (N - i) = O(N^2)$.

Optimization

Global broadcast is not necessary

Outline

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

   receive (probe,j,k,d)  from left(right):
   if j = id then
2      leader ← j
       Terminate
3  end
4  if j < id and d < 2^k then
5      send  (probe, j, k, d+1)  to right (left)
6  end
7  if j < id and d = 2^k then
8      send  (reply, j, k)  to left (right)
9  end
```

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

```
1 receive (reply,j,k) from left(right):
  if j ≠ id then
2 |   send (reply,j,k) to right(left)
3 end
4 else
5 |   if received (reply,j,k) from right(left) then
6 |   |   send (probe,id,k+1,1) to left and right
7 |   end
8 end
```

Analysis

- 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 4×2^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:

$$M = \sum_{k=1}^{log(n)} 4 \times 2^k \times \frac{n}{2^{k-1}+1} = O(n\ log(n)) \qquad (1)$$

Leader Election in Trees

- 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

Initialization

```
   /* Wakeup all the nodes                                    */
1  if p is an initiator then
2  |   awake ← true
   |   foreach q ∈ neigh(p) do
3  |   |   send wakeup to q
4  |   end
5  end
6  while numWakeups <| neigh(p) | do
7  |   receive( wakeup )
   |   numWakeups ← numWakeups + 1
   |   if awake = false then
8  |   |   awake ← true
   |   |   foreach q ∈ neigh(p) do
9  |   |   |   send wakeup to q
10 |   |   end
11 |   end
```

Send Proposal to Parent

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

Decide the Leader

```
   /* Receive the result from the parent, and
      send to the leaves                                   */
1  receive < r > from parent
   res ← min( min_p , < r > )
   if res = p then
2  |   state ← leader
   |
3  end
4  else
5  |   state ← lost
   |
6  end
7  foreach q ∈ neigh(p), q ≠ parent do
8  |   send res to q
9  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 = $O(N)$

 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

 Advanced Concepts in Operating Systems by Mukesh Singhal and Niranjan Shivaratri, McGrawHill, 1994

 Distributed Algorithms by Nancy Lynch, Morgan Kaufmann, 1996