EE 382N: Distributed Systems **Fall 2017**

Lecture 6: October 14

Lecturer: Vijay Garg *Scribe: Samuel Cherinet*

1.1 Leader Election in a Ring

In distributed systems it is a common occurrence that we want to choose the leader in a network of processes so that the leader could be for example a coordinator of some sort. It is also possible that these processes can have either a unique id or non-unique id.

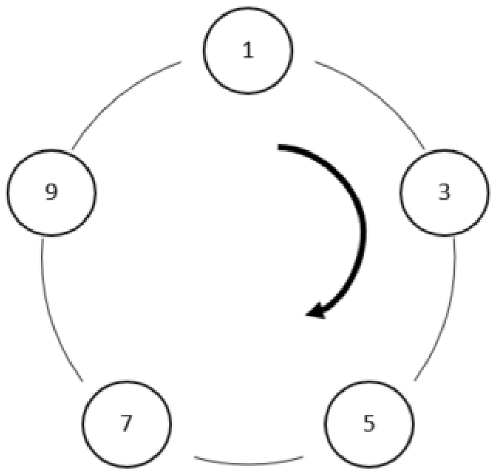


Figure 1.1: Simple ring diagram

So the question is how can we elect a leader?

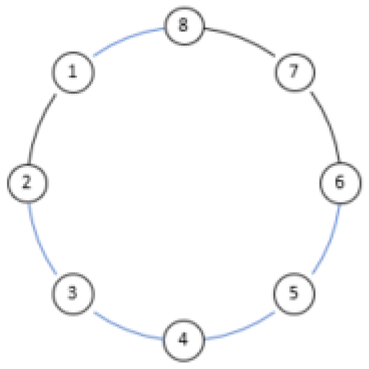
There is no deterministic algorithm if the ids are not unique. we can of course randomly pick a leader but there is no deterministic algorithm to pick one. So the assumption ,going forward , is every process has a unique id.And for the sake of simplicity the convention is the process with the biggest id is going to be the leader. the problem is that each of the process doesn't know that he has the biggest number or how many processes there there in the network for that matter.

The algorithm is started by having every process send it's id clockwise to it's left neighbor, algorithm looks much like below:

```
$P_i$::
var
  myid: integer;
  awake: boolean initally false;
  leaderid: integer initally null;
To initiate election:
  send (election,myid) to $P_{i-1}$;
  awake:=true;
Upon receiving a message(election,j):
  if(j > myid then
    send( election, j ) to $P_{i-1}$;
  else if(j = myid) then
    send( leader, myid ) to $P_{i-1}$;
  else if((j < myid) $\wedge$ $\neg$awake)then
    send( election, myid ) to $P_{i-1}$;
  awake:=true;
 Upon receiving a message (leaser,j):
  leaderid=j;;
  if (j=myid) then send(leader,j) to $P_{i-1}$;
```

Message Complexity is $O(N^2)$ Worst Case Scenario $1 + 2 + 3 + 4... + N = O(N^2)$

The figure below shows the worst and best cases when determining a ring leader using this algorithm.



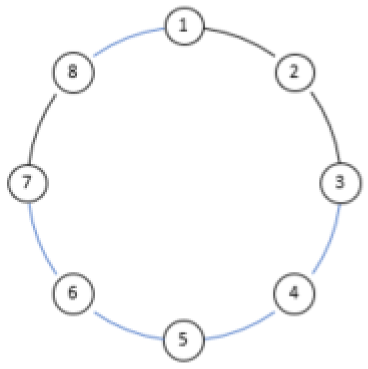


Figure 1.2: Worst and best case scenarios

1.2 Hierarchical Election

Now let's try to send message in both directions, then you will not send too many messages. Instead of going around the whole ring, we will use hierarchical election. to be a leader of the ring a process have to be a leader of it's neighborhood. only when a process wins the first round that it can move to the next one.

There is a notion of neighborhood. Let's say d is length of my neighborhood left and right

example 1 left and 1 right, I send a message to both. and with the message I keep the hope count.and the neighbor would only send it back if that guy has a bigger number than you. When you are recipient if your id is smaller than the sender's then you ignore it, since that is a waste of a message. The table below shows the number of messages that are needed as processes go through each round of electing a local leader.

Round	No. of leaders	d	No. of messages
1	n	1	4n
2	n/2	2	4n
3	n/4	4	4n
4	n/8	8	4n
..	..	..	..
i	i/(2_{i-1})	(2_{i-1})	4n

Figure 1.3: Rounds and number of messages it takes

Number of Messages = O(nlogn)

Theorem leader election requires $\Omega(nlogn) messagesinaring$

References

[GARG02] Vijay K. GARG, Elements of Distributed Computing, pp. 163–167.