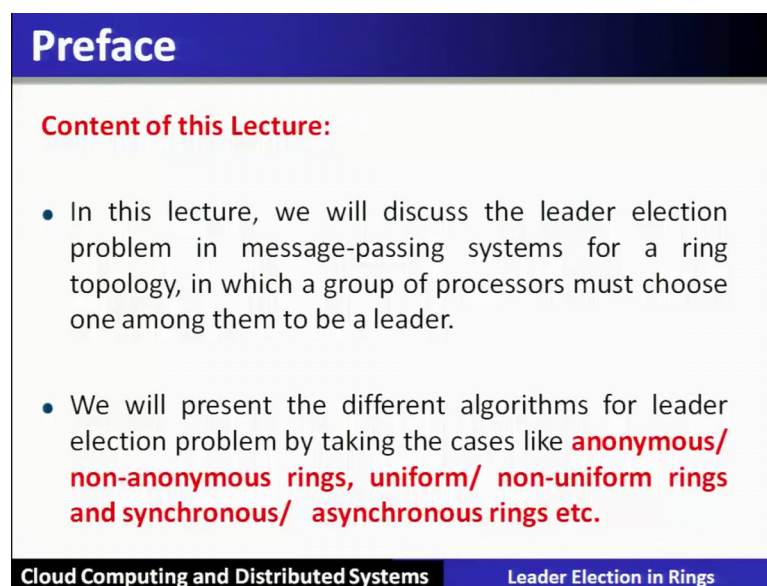


Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture - 07
Leader Election in Rings (Classical Distributed Algorithms)

Leader Election in the Rings Classical Distributed Algorithms. In this lecture we will discuss leader election problem in a message passing systems.

(Refer Slide Time: 00:24)



Preface

Content of this Lecture:

- In this lecture, we will discuss the leader election problem in message-passing systems for a ring topology, in which a group of processors must choose one among them to be a leader.
- We will present the different algorithms for leader election problem by taking the cases like **anonymous/ non-anonymous rings, uniform/ non-uniform rings and synchronous/ asynchronous rings etc.**

Cloud Computing and Distributed Systems **Leader Election in Rings**

Especially, for the ring topology in which the group of processors must choose one of them to be the leader. Different algorithms for leader election, different scenarios such as anonymous non-anonymous rings, uniform non-uniform rings, synchronous and asynchronous rings.

(Refer Slide Time: 00:49)

Leader Election (LE) Problem: Introduction

- The leader election problem has several variants.
- LE problem is for each processor to decide that either it is **the leader or non-leader**, subject to the constraint that exactly one processor decides to be the leader.
- LE problem represents a general class of **symmetry-breaking** problems.
- For example, when a deadlock is created, because of processors waiting in a cycle for each other, the deadlock can be broken by electing one of the processor as a leader and removing it from the cycle.

Cloud Computing and Distributed Systems Leader Election in Rings

Leader election problem; the leader election problem has several variants, leader election is for each process to decide either it is the leader or a non-leader; subject to the constraint that exactly one processor decides to be a leader.

So, leader election problem represents a general class of symmetry breaking problems. For example, when a deadlock is created one of the processors waiting in a cycle for each other, the deadlock can be broken by electing one of the waiting processes as a leader and removing it from the cycle; that is breaking up the deadlock.

(Refer Slide Time: 01:23)

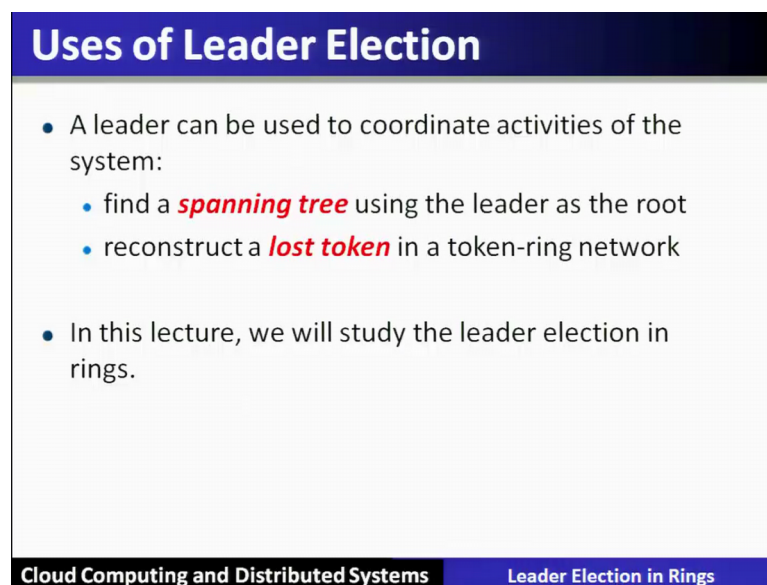
Leader Election: Definition

- Each processor has a set of **elected (won)** and **not-elected (lost)** states.
- Once an elected state is entered, processor is always in an elected state (and similarly for not-elected): i.e., irreversible decision
- In every admissible execution:
 - every processor eventually enters either an elected or a not-elected state
 - exactly one processor (the **leader**) enters an elected state

Cloud Computing and Distributed Systems Leader Election in Rings

So, the leader election problem definitions, definition each process or each processor has a set of elected and a non-elected states. Once an elected state is entered the processor always in the elected state. And similarly for non-elected, that is irreversible decisions. So, in every admissible execution every processor eventually enters either as elected or non-elected state exactly one processors; are exactly one processor that is the leader enters into the elected state.

(Refer Slide Time: 01:59)



Uses of Leader Election

- A leader can be used to coordinate activities of the system:
 - find a **spanning tree** using the leader as the root
 - reconstruct a **lost token** in a token-ring network
- In this lecture, we will study the leader election in rings.

Cloud Computing and Distributed Systems Leader Election in Rings

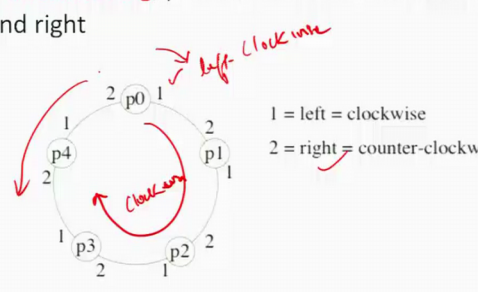
Now there are different uses of leader election algorithm. So, leader election can be used to coordinate the activities in the system. For example, to find out the spanning tree in a system requires a leader to be known which is called a root. Hence, if a root is given finding a spanning tree becomes easier.

Similarly, in a token ring system if a token is lost; that means, there is no leader so, it can elect a leader and restart the system with a token.

(Refer Slide Time: 02:42)

Definition: (1) Ring Networks

- In an **oriented ring**, processors have a consistent notion of left and right



1 = left = clockwise
2 = right = counter-clockwise

- For example, if messages are always forwarded on channel 1, they will cycle clockwise around the ring

Cloud Computing and Distributed Systems Leader Election in Rings

So, in this lecture we will study the leader election in a ring. So, let us define different terms which are used in the leader election problem; that is the ring problem. So, the first definition is about the ring networks.

So, in an oriented ring, the processors have a consistent notion of left and right. So, in this particular example, we can see here the label one called as a left side, if it is used to forward the message in this particular direction that is called a clockwise direction. Then this particular way the ring will be oriented in a clockwise manner. Similarly, if let us say that if p0 and other processes they basically use the right side, that is the label number 2, always then it will be a counter clockwise and the ring will be oriented in that manner such rings are called oriented rings.

Now another definition is about anonymous rings.

(Refer Slide Time: 04:02)

Definition: (2) Anonymous Rings

- How to model situation when **processors do not have unique identifiers?**
- **First attempt:** require each processor to have the same state machine

Cloud Computing and Distributed Systems Leader Election in Rings

So, if the processors do not have the unique ids; that means they are not given an ids, then that particular ring is called anonymous rings. So, in that situation each processor is like same running state machine there is no distinction.

(Refer Slide Time: 04:24)

Definition: (3) Uniform (Anonymous) Algorithms

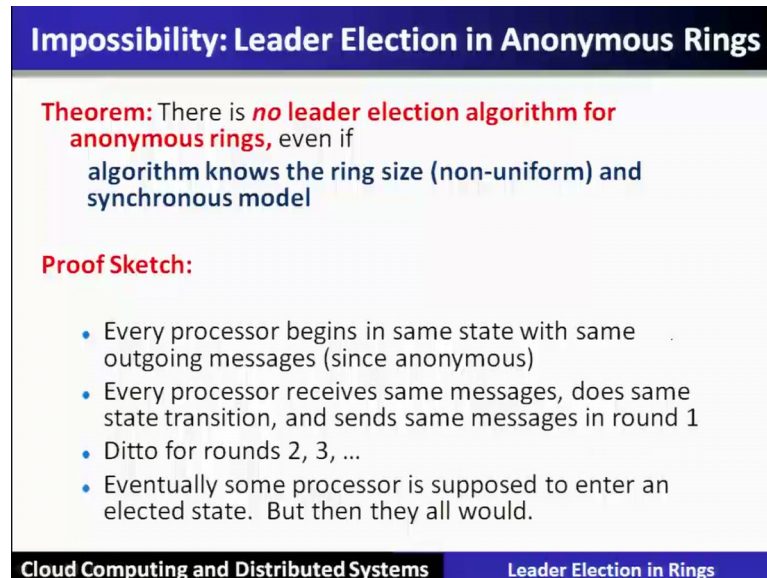
- A **uniform** algorithm **does not use the ring size** (same algorithm for each size ring) *ie n*
 - Formally, every processor in every size ring is modeled with the same state machine
- A **non-uniform** algorithm **uses the ring size** (different algorithm for each size ring) *size*
 - Formally, for each value of n , every processor in a ring of size n is modeled with the same state machine A_n .
- Note the lack of unique ids.

Cloud Computing and Distributed Systems Leader Election in Rings

Third definition is about uniform anonymous algorithms. So, uniform algorithm means that it does not use the information of the ring size that is n ; the number of nodes in the ring. So, formally every processor in every size ring is modeled with the same state

machine. So, the algorithm which is called a non-uniform algorithm we will use the size of the ring that is an n in the algorithm.

(Refer Slide Time: 05:03)



Impossibility: Leader Election in Anonymous Rings

Theorem: There is **no leader election algorithm for anonymous rings**, even if
algorithm knows the ring size (non-uniform) and
synchronous model

Proof Sketch:

- Every processor begins in same state with same outgoing messages (since anonymous)
- Every processor receives same messages, does same state transition, and sends same messages in round 1
- Ditto for rounds 2, 3, ...
- Eventually some processor is supposed to enter an elected state. But then they all would.

Cloud Computing and Distributed Systems Leader Election in Rings

There is an impossibility result which says that about the leader election anonymous rings theorem. There is no leader election algorithm for anonymous rings even if the algorithm knows the ring size that is for the non-uniform. And also the model is synchronous model, why; because every processor begins in the same state with the same outgoing message. Since it is anonymous, and each processor when receives the message will also be in the same state transition, and sends the message in a round one.

(Refer Slide Time: 05:48)

Leader Election in Anonymous Rings

- Proof sketch shows that either safety (never elect more than one leader) or liveness (eventually elect at least one leader) is violated.
- Since the theorem was proved for non-uniform and synchronous rings, the same result holds for weaker (less well-behaved) models:
 - uniform
 - asynchronous

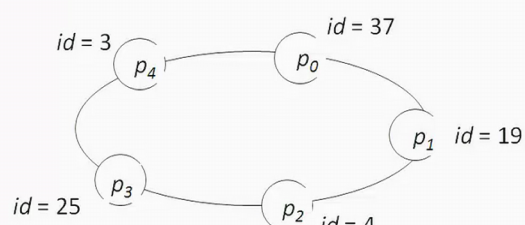
Cloud Computing and Distributed Systems Leader Election in Rings

So, there is no distinction and there is all symmetry looking up in this particular way. So, it shows that either the safety or the liveness is violated here in this case. Hence the theorem was proved for non-uniform and synchronous rings. The same result hold for a weaker model that is for the uniform and the asynchronous model. So, rings with the ids. So, we will assume that each processor has the unique id, ok. Let us see a example of a ring.

(Refer Slide Time: 06:33)

Specifying a Ring

- Start with the smallest id and list ids in clockwise order.



- Example: 3, 37, 19, 4, 25

Cloud Computing and Distributed Systems Leader Election in Rings

Here we see that every processor in the ring is having different ids for example, 3, 37, 19, 4 and 25.

(Refer Slide Time: 06:46)

Uniform (Non-anonymous) Algorithms

- **Uniform** algorithm: there is one state machine for every id, no matter what size ring
- **Non-uniform** algorithm: there is one state machine for every id and every different ring size
- These definitions are tailored for leader election in a ring.

Cloud Computing and Distributed Systems Leader Election in Rings

So, uniform algorithms means that the algorithm is not using the size of the ring. So, no matter what is the size of a ring the algorithm in the algorithm. Similarly, non-uniform rings shows that it uses the size of the ring.

(Refer Slide Time: 07:04)

**$O(n^2)$ Messages LE Algorithm:
LeLann-Chang-Roberts (LCR) algorithm**

- send value of own id to the left
- when receive an id j (from the right):
 - if $j > id$ then
 - forward j to the left (this processor has lost)
 - if $j = id$ then
 - elect self (this processor has won)
 - if $j < id$ then
 - do nothing

Cloud Computing and Distributed Systems Leader Election in Rings

Let us see the algorithm which is called a LeLann-Chang-Roberts algorithm; LCR algorithm for leader election problem. This is also called as a order n square leader

election algorithm. Here algorithm says that every processor will send its id to the left. When it receives an id from the right, and if j is greater than its id, then it will forward to the left. If less than id, then it will not do anything it will not forward it will swallow. And if id is equal to the j then it will elect itself as the leader.

(Refer Slide Time: 07:53)

Analysis of $O(n^2)$ Algorithm

Correctness: Elects processor with largest id.

- message containing largest id passes through every processor

Time: $O(n)$

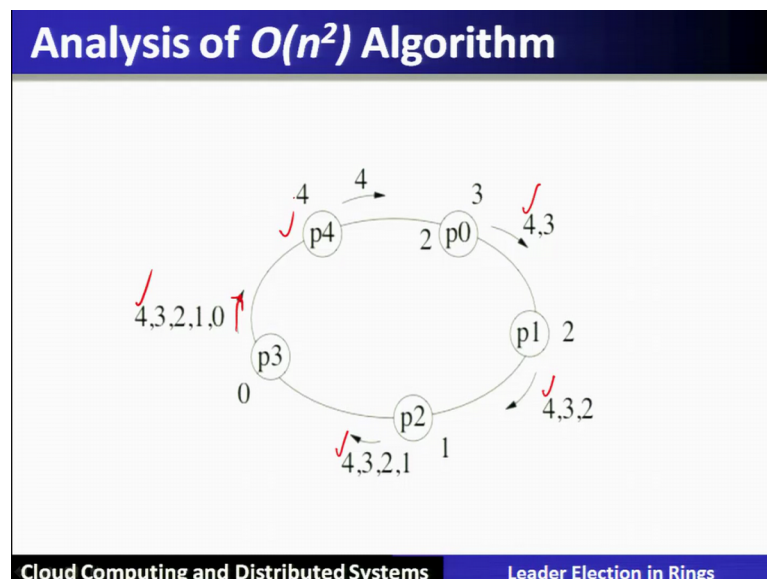
Message complexity: Depends how the ids are arranged.

- largest id travels all around the ring (n messages)
- 2nd largest id travels until reaching largest
- 3rd largest id travels until reaching largest or second largest etc.

Cloud Computing and Distributed Systems
Leader Election in Rings

We will see through an example. As far as correctness is concerned, it will always elect a processor with a largest id in the time of the order n and the message complexity is order n square, let us see through an example.

(Refer Slide Time: 08:07)



Now let us see that the processor p 3 will send it is id to the left p 4 on receiving 0 it will not forward it further. Whereas, p 4 will send it is own id, and when it reaches to p 0, it is id is greater than 3 so, it will be forwarded further. When it reaches to p 1 it will also be forwarded. And it will be forwarded, and when it reaches to this particular point because this is the highest. So, it will see its own message so, hence this will be elected as a leader.

Whereas, the other messages having lower ids will be absorbed so, let us see when this order n square algorithm will arise.

(Refer Slide Time: 09:21)

Analysis of $O(n^2)$ Algorithm

- Worst way to arrange the ids is in decreasing order:
 - 2nd largest causes $n - 1$ messages
 - 3rd largest causes $n - 2$ messages etc.
- **Total number of messages is:**
$$n + (n-1) + (n-2) + \dots + 1 = \Theta(n^2)$$


Cloud Computing and Distributed Systems

Leader Election in Rings

Let us see that the messages are being forwarded here in this particular scenario. Let us see this particular method of analyzing the worst case scenario, that is of the order n square.

(Refer Slide Time: 09:46)

Analysis of $O(n^2)$ Algorithm

- Clearly, the algorithm never sends more than $O(n^2)$ messages in any admissible execution. Moreover, there is an admissible execution in which the algorithm sends $\Theta(n^2)$ messages; Consider the ring where the identifiers of the processor are $0, \dots, n-1$ and they are ordered as in Figure 3.2. In this configuration, the message of processor with identifier i is sent exactly $i+1$ times, Thus the total number of messages, including the n termination messages, is

$$n + \sum_{i=0}^{n-1} (i+1) = \Theta(n^2)$$

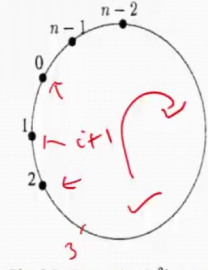


Fig. 3.2 Ring with $\Theta(n^2)$ messages.

$\Theta(n^2)$ **Clockwise Unidirectional Ring**

Cloud Computing and Distributed Systems
Leader Election in Rings

We will consider the ring where the identifiers of the processor 0 1 2 and so on up to n minus 1 and are ordered in this particular manner. Here the identifier i is sent exactly i plus 1 times. So, for example, 0 will be sent 0 plus 1 that is only one time it will be observed here. 1 that is i plus 1 times it will be forwarded. 1 will be forwarded by 0; that means, it will be forwarded 2 times and so on. So, 2 will be forwarded 3 times, 3 will be forwarded 4 times and so on. So, this is the orientation of the ring.

How many number of messages are being forwarded? Here so, we will see that for i it will be forwarded i plus 1 times. So, if we sum for the nodes from 0 to n minus 1. So, these are the total number of messages which will be forwarded in this particular worst case scenario in the ring. And when a leader is elected then the highest id will circulate it is message that is n . Now, this particular summation is of the order n square.

(Refer Slide Time: 11:49)

Can We Use Fewer Messages?

- The $O(n^2)$ algorithm is simple and works in both synchronous and asynchronous model.
- But can we solve the problem with fewer messages?

Idea:

- Try to have messages containing smaller ids travel smaller distance in the ring

non-uniform - uniform $O(n^2)$ LE *Synch*
Asynch

Cloud Computing and Distributed Systems Leader Election in Rings

So, it requires n square number of messages in the worst situation we have shown in this analysis. Algorithm which is of the order n square algorithm is simple and works in both synchronous and asynchronous model. Another good thing about this algorithm is does not use the value of n or the size of n . Hence, it is non-uniform, hence it is uniform. So, as it is shown that it will work in both the situation either for the synchronous model or for asynchronous model, anonymous order n square leader election algorithm.

(Refer Slide Time: 12:44)

$O(n \log n)$ Messages LE Algorithm: The Hirschberg and Sinclair (HS) algorithm

- To describe the algorithm, we first define the k -neighbourhood of a processor p_i in the ring to be the set of processors that are at distance at most k from p_i in the ring (either to the left or to the right). Note that the k -neighbourhood of a processor includes exactly $2k+1$ processors.
- The algorithm operates in phases; it is convenient to start numbering the phases with 0. In the k th phase a processor tries to become a winner for that phase; to be a winner, it must have the largest id in its 2^k -neighborhood. Only processors that are winners in the k th phase continue to compete in the $(k+1)$ -st phase. Thus fewer processors proceed to higher phases, until at the end, only one processor is a winner and it is elected as the leader of the whole ring.

Diagram: A ring of processors with a central processor p_i . Its k -neighbourhood is shown as a circle of radius k around p_i , containing $2k+1$ processors. The k -neighbourhood of p_i is labeled $2k+1$ processors.

Cloud Computing and Distributed Systems Leader Election in Rings

Now, can we do with a lesser message complexity than order n^2 ? We will see an algorithm which is given by Hirschberg and Sinclair HS algorithm, it is called it is order $n \log n$ leader election algorithm. This algorithm uses the concept of k neighbourhood for any processor p_i in the ring. That is nothing but distance of at most k . It is (Refer Time: 13:09) and this is k this is k and this is one that is $2k + 1$ nodes for the processors in the k neighbourhood of a processor.

So, k neighbourhood of a processor includes exactly $2k + 1$ processors. This algorithm operates in the phases. Therefore, it is convenient to start numbering the phases starting from the phase 0. So, the k th phase of a process will try to become the winner, for that phase to be the winner it must have the largest id in its $2k + 1$ neighbourhood. So, only the processor that are the winners of k th phase will continue to participate in $k + 1$ th phase. Thus, fewer processor proceeds to a higher phase until at the end only one processor is the winner and is elected as the leader of the whole ring.

(Refer Slide Time: 14:29)

The HS Algorithm: Sending Messages

Phase 0

- In more detail, in phase 0, each processor attempts to become a phase 0 winner and sends a **<probe>** message containing its identifier to its **1-neighbourhood**, that is, to each of its two neighbors.
- If the identifier of the neighbor receiving the probe is greater than the identifier in the probe, it swallows the probe; otherwise, it sends back a **<reply>** message.
- If a processor receives a reply from both its neighbors, then the processor becomes a phase 0 winner and continues to phase 1.

Cloud Computing and Distributed Systems

Leader Election in Rings

So, let us see the phase 0, in more detail phase 0 each processor attempts to become of a phase 0 winner, and sends a probe message containing its id to its one of neighbourhood to each of the 2 neighbours so for example if this is the neighbourhood and to one neighbourhood on both the sides in phase 0 if the [FL].

[FL].

[FL], If the id identifier of the neighbour receiving the probe is greater than the identifier in the probe, it swallows the probe, otherwise it sends back the reply.

So, if it gets the replies back from both the end; that means, it has become the winner of phase 0 and it will continue to the phase 1. So, in the phase 1 the neighbourhood size will basically double. If it is let us say that it is phase k ; that means, all the winners of phase k minus 1 will send it is probe message with its id to its 2 raised power k neighbourhood one in each direction. Each message traverses 2 raised power k processor one by one and the probe is swallowed by the processor if it contains an id that is smaller than it is own id.

If the probe arrives at the last processor on the neighbourhood without being swallowed, then the last processor will send back the reply as we have seen in the first or the 0th phase.

(Refer Slide Time: 16:17)

The HS Algorithm: Sending Messages

Phase k

- In general, in phase k , a processor p_i that is a phase **$k-1$ winner** sends **<probe>** messages with its identifier to its **2^k -neighborhood** (one in each direction). Each such message traverses **2^k processors** one by one, A probe is swallowed by a processor if it contains an identifier that is smaller than its own identifier.
- If the probe arrives at the last processor on the neighbourhood without being swallowed, then that last processor sends back a **<reply>** message to p_i . If p_i receives replies from both directions, it becomes a phase k winner, and it continues to phase **$k+1$** . A processor that receives its own **<probe>** message terminates the algorithm as the leader and sends a termination message around the ring.

Cloud Computing and Distributed Systems

Leader Election in Rings

So, this is the complete algorithm for the processor i and this algorithm will be for all the processor that is from 0 to n minus 1.

(Refer Slide Time: 16:28)

Algorithm 5 Asynchronous leader election: code for processor $p_i, 0 \leq i < n$.

Initially, $asleep = true$

```

1: upon receiving no message:
2:   if  $asleep$  then
3:      $asleep := false$ 
4:     send  $\langle probe, id, 0, 1 \rangle$  to left and right

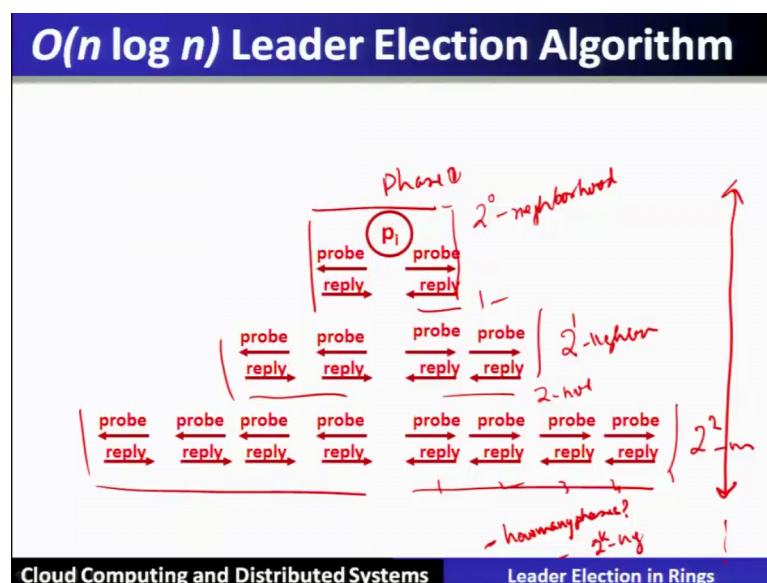
5: upon receiving  $\langle probe, j, k, d \rangle$  from left (resp., right):
6:   if  $j = id$  then terminate as the leader
7:   if  $j > id$  and  $d < 2^k$  then                // forward the message
8:     send  $\langle probe, j, k, d + 1 \rangle$  to right (resp., left) // increment hop counter
9:   if  $j > id$  and  $d \geq 2^k$  then                // reply to the message
10:    send  $\langle reply, j, k \rangle$  to left (resp., right)
                                           // if  $j < id$ , message is swallowed

11: upon receiving  $\langle reply, j, k \rangle$  from left (resp., right):
12:   if  $j \neq id$  then send  $\langle reply, j, k \rangle$  to right (resp., left) // forward the reply
13:   else // reply is for own probe
14:     if already received  $\langle reply, j, k \rangle$  from right (resp., left) then
15:       send  $\langle probe, id, k + 1, 1 \rangle$  // phase  $k$  winner
  
```

So, here we will see that the algorithm, it will send this particular probe to the left and right, and it will do the election it will initiate multiple elections in the k neighbourhood; that means, the entire ring is divided into k neighbourhood and this elections will parallelly run for the k th neighbourhood.

So, we see that that the size of the neighbourhood will double in each phase. So, if a probe reaches a node with a largest larger id the probe will stop.

(Refer Slide Time: 17:18)



If the probe reaches the end of the neighborhood, then the reply will be sent back to the initiator. So, this can be seen through this particular diagram that this is phase 0; that means this particular one half neighbourhood will perform the elections. And whose over will be the winner then the size will be doubled that is 2 raised power one neighbourhood. So, that means, 2 this was earlier 1 node neighbourhood, now it is having 2 nodes in the neighbourhood.

And whosever and if it is able to win for 2 raised power 1 neighbourhood, then the neighbourhood size will double, that is 2 raised power 2 neighbourhood; that means, 1 2 3 4 different so, the neighbourhood size will keeps on growing in this particular manner. Now we will count what is the depth; that means how long how many different phases will be there, how many phases will be there and in each phase, you know that it will be 2 raised power k neighbourhood. So, in each phase how many such elections will happen? And these particular information is required to compute the message complexity.

(Refer Slide Time: 19:10)

Analysis of $O(n \log n)$ Leader Election Algorithm

Correctness:

- Similar to $O(n^2)$ algorithm.

Message Complexity:

- Each message belongs to a particular phase and is initiated by a particular processor
- Probe distance in phase k is 2^k
- Number of messages initiated by a processor in phase k is at most $4 \cdot 2^k$ (probes and replies in both directions)

Cloud Computing and Distributed Systems
Leader Election in Rings

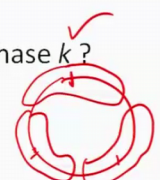
So, the message complexity, that means, here the probe distance is in a phase k is 2 raised power k . So, the number of messages initiated by a process in a phase k is at most 4 times 2 raised power k ; that means if this is the k phase it is neighbourhood. So, the message will go one times, the reply will come back 2 times, then message will go on the

right side third time and it will come back to the 4 times. So, 4 time 2 raised power k in every phase 4 times these probes will be sent in counting the message complexity.

(Refer Slide Time: 19:55)

Analysis of $O(n \log n)$ Leader Election Algorithm

- How many processors initiate probes in phase k ?
- For $k = 0$, every processor does
- For $k > 0$, every processor that is a "winner" in phase $k - 1$ does
 - "winner" means has largest id in its 2^{k-1} neighborhood



$k-1$ - phase

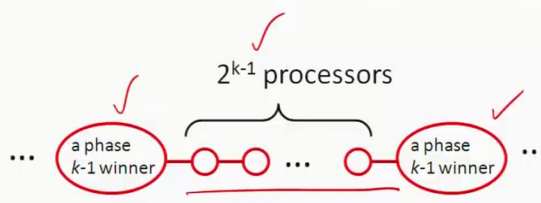
Cloud Computing and Distributed Systems
Leader Election in Rings

Now, the question is how many such processor will initiate the probe in the phase k . Meaning to say that if this is the k hop so, how many such packing of k hops will be initiated at a particular phase k . Now for k is equal to 0 every processor will be participating in this particular election process when k is greater than 0; that means, the winner of phase k minus 1 only participate in k th phase.

(Refer Slide Time: 20:58)

Analysis of $O(n \log n)$ Leader Election Algorithm

- Maximum number of phase $k - 1$ winners occurs when they are packed as densely as possible:



- Total number of phase $k - 1$ winners is at most $n/(2^{k-1} + 1)$



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

Cloud Computing and Distributed Systems
Leader Election in Rings

So, the winner of k minus 1th phase means that the largest id in 2 raised power k minus 1 neighbourhood which is being seen. So, the maximum number of phase k minus 1 winner occurs when they are packed as densely as possible.

So, let us see that here it is phase k minus 1 winner and another phase k minus 1 winner. So, how many number of processors which are in between 2 extreme phase k minus 1 winner is nothing but 2 raised power k minus 1 winner. We can see through an example that if this is phase 0 so, it will have the one neighbourhood. So, this particular 0, this particular another node will also have this kind of neighbourhood. So, the number of nodes involved is basically 2 raised power k minus 1. So, the total number of phase k minus 1 winner will be n divided by 2 raised power k minus 1 plus 1.

That means this is the packing, this is the packing of so many k minus 1. So, n divided by 2 raised power k minus 1 plus 1, this will be total number of such k minus 1 winners who are participating in k th round. Now the question is how many phases are there. Now you know that every phase the number of winners is cut in a half that is from n raised power 2 k minus 1 plus 1.

(Refer Slide Time: 22:40)

Analysis of $O(n \log n)$ Leader Election Algorithm

- How many phases are there? ✓
- At each phase the number of (phase) winners is cut approx. in half
 - from $n/(2^{k-1} + 1)$ to $n/(2^k + 1)$ - - - -
- So after approx. $\log_2 n$ phases, only one winner is left.
 - more precisely, **max phase is $\lceil \log(n-1) \rceil + 1$**

$$\frac{n}{2^{k+1}} = 1$$

$\log_2 \frac{n}{2^{k+1}} = \log_2 1 \Rightarrow \log_2 n - (k+1) = 0$
 $k = \log_2 n - 1$
 $k = \lceil \log_2 n \rceil - 1$

Cloud Computing and Distributed Systems
Leader Election in Rings

It will now go to n divided by 2^k plus 1 in the k th phase. So, we will see that we will continue in this manner so that finally, there will be only one such neighbourhood remains that is the entire ring. And if we compute then it becomes it comes out to be $\log$ of n minus 1 plus 1 total number of phases.

So, again we can see n divided by 2^{k-1} plus 1. Let us say this becomes 1 if we take the log of n , then it will become $k-1$. So, this k is equal to $\log n$ plus 1. So, this is how this particular formula is being obtained.

(Refer Slide Time: 24:14)

Analysis of $O(n \log n)$ Leader Election Algorithm

- Total number of messages is sum, over all phases, of number of winners at that phase times number of messages originated by that winner:

$$\leq 4n + n + \sum_{k=1}^{\lceil \log(n-1) \rceil + 1} 4 \cdot 2^k \cdot n / (2^{k-1} + 1)$$

phase 0 msgs

termination msgs

$$< 8n(\log n + 2) + 5n$$

$$= O(n \log n)$$

msgs for phases 1 to $\lceil \log(n-1) \rceil + 1$

Cloud Computing and Distributed Systems Leader Election in Rings

Now plugging up all the values we will find out the total number of messages in all the phases, that comes out to be this is the phase 0 messages, 4 times n and this is the termination message. And as far as this formula is concerned, this formula says that how many different messages are there for the phases 1 to $\log n$ minus 1.

So, this is the summation of different phases, and every phase will require 4 times 2^k raised power k times n raised power n divided by 2^{k-1} plus 1. So, if you compute it comes out to be $8n \log n$ plus 5 plus 2 plus $5n$. That comes out to be of the order $n \log n$. So, we have seen that this particular algorithm will work both for synchronous and asynchronous case. So, the question is can we reduce the number of messages even further than order $n \log n$.

So, we will we have seen that not in asynchronous model you can do this better than order $n \log n$; that for that we can show that this is the lower bound for the leader election problem that is of the order $n \log n$.

(Refer Slide Time: 25:38)

Lower bound for LE algorithm

But, can we do better than $O(n \log n)$?

Theorem: Any leader election algorithm for asynchronous rings whose size is not known a priori has $\Omega(n \log n)$ message complexity (holds also for unidirectional rings).

- Both LCR and HS are comparison-based algorithms, i.e. they use the identifiers only for comparisons ($<$; $>$; $=$).
- In synchronous networks, $O(n)$ message complexity can be achieved if general arithmetic operations are permitted (non-comparison based) and if time complexity is unbounded.

Cloud Computing and Distributed Systems Leader Election in Rings

So, the theorem says that any leader election algorithm for asynchronous rings whose size is not known a priori has the lower bound of $n \log n$ message complexity, holds also for the unidirectional rings. Both LCR and HS are comparison based algorithm; that is, they use identifiers only for the comparison. In synchronous algorithms order of n message complexity can be achieved if general arithmetic operations are permitted, non-comparison based and if the time complexity is unbounded.

(Refer Slide Time: 26:29)

Overview of LE in Rings with Ids

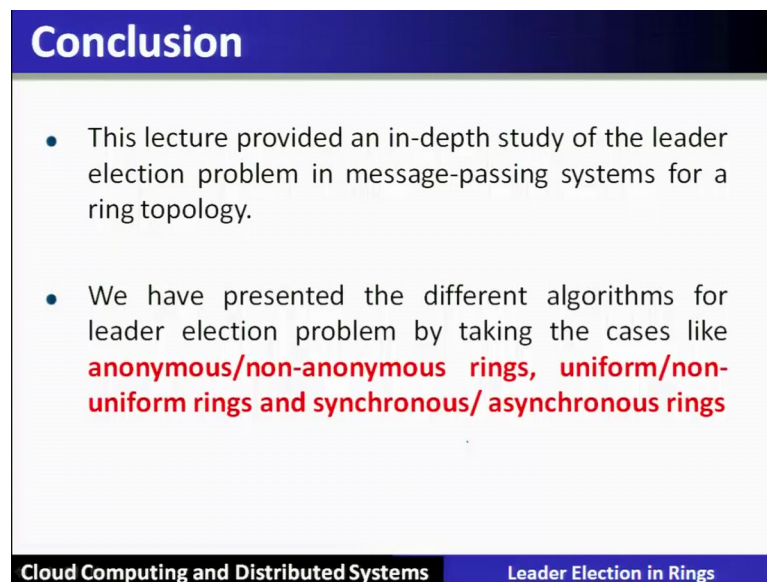
- There exist algorithms when nodes have unique ids.
- We have evaluated them according to their **message complexity**.
- **Asynchronous ring:**
 - $\Theta(n \log n)$ messages
- **Synchronous ring:**
 - $\Theta(n)$ messages under certain conditions
 - otherwise $\Theta(n \log n)$ messages
- All bounds are asymptotically tight.

Cloud Computing and Distributed Systems Leader Election in Rings

So, overview of leader election in rings with the ids, there exist algorithm when the nodes have unique ids. We have evaluated them according to their message complexity, and we found out that in the case of asynchronous ring the message complexity of the algorithm the best known algorithm gives $n \log n$ messages. Similarly, for synchronous rings we have seen that of the order n messages under certain conditions. Otherwise, the complexity is the same that is $n \log n$ messages in the comparison based algorithms.

So, all these bounds are asymptotically tight conclusion. This lecture provided an in depth study of the leader election problem in the message passing system for the ring topology.

(Refer Slide Time: 27:23)



Conclusion

- This lecture provided an in-depth study of the leader election problem in message-passing systems for a ring topology.
- We have presented the different algorithms for leader election problem by taking the cases like **anonymous/non-anonymous rings, uniform/non-uniform rings and synchronous/asynchronous rings**

Cloud Computing and Distributed Systems Leader Election in Rings

We are also presented different algorithm for the leader election by taking different cases. Such as anonymous oblique non-anonymous rings, uniform oblique non-uniform rings, synchronous oblique asynchronous rings.

Thank you.