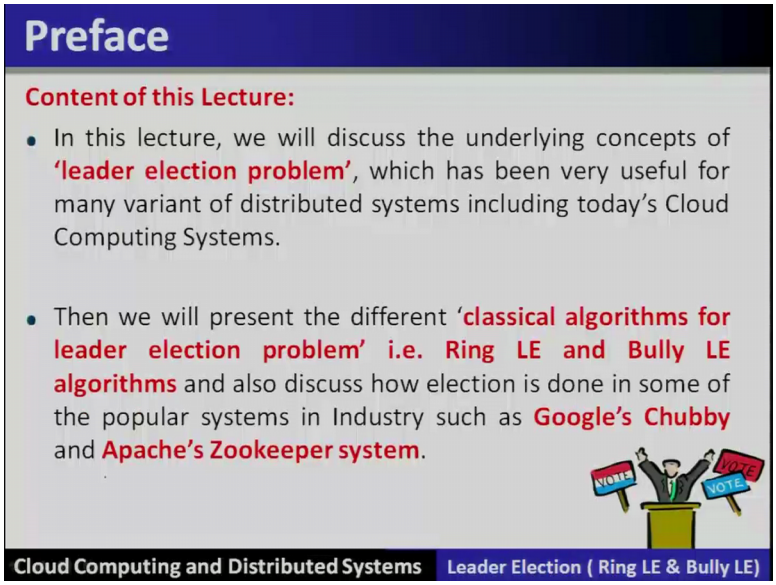
Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture - 08
Leader Election (Ring LE & Bully LE)

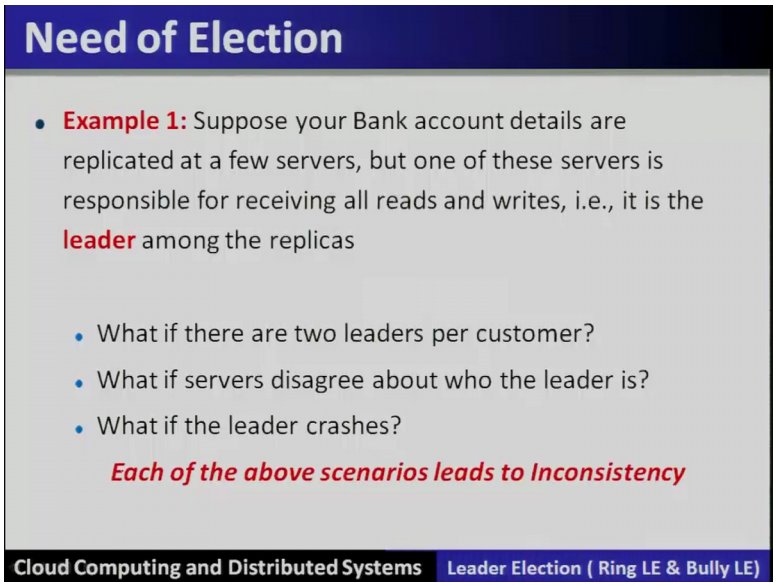
Leader election and Bully Leader Election algorithm. Preface; content of this lecture.

(Refer Slide Time: 00:21)



We will discuss the underlying concepts of leader election problem, which has been very useful for many variants of distributed systems, and in today's cloud computing systems, we will present the different classical distributed algorithm for leader election problem, namely ring leader election, Bully's leader election problem and also we will discuss how this election is done in some of the popular systems, used in the industry; such as in Google's Chubby and in Apaches Zookeeper system. So, the need of the leader election.

(Refer Slide Time: 01:09)

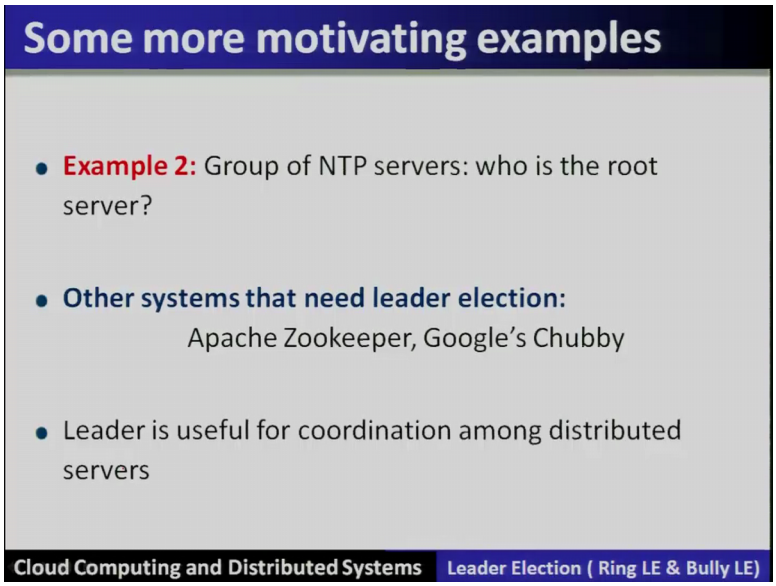


Leader election is an important problem in a distributed system. To understand the use of leader election, let us see through some examples. Suppose your bank account details are replicated in a few servers and out of these many servers one of them is responsible for receiving all the request for reading and writing operations; that is out of several replicated servers, one of them is responsible that is called leader, that is coordinating the work on behalf of all other replicas.

Now, in this particular scenario what will happen if there are two different leaders per customer, then it will be unclear or a confusion across the different replicas. that who is the leader or who is the coordinator, who is responsible to control all the operations. what if the server disagree about who the leader is that means, if there is no agreement among all the replicas about the server then the application will not be able to be coordinated, to perform the operations, as per the specification of the application.

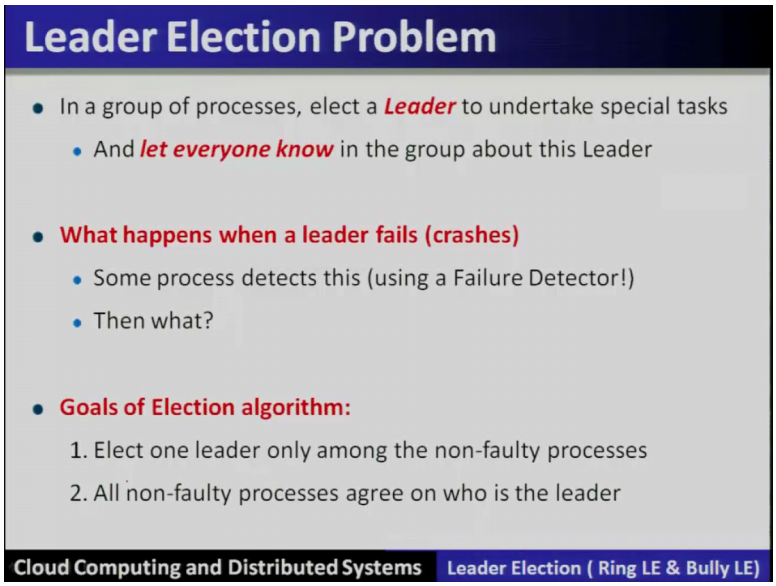
What if the leader is crashed or is down. So, these different situations will lead to inconsistency in this situation of the leader and that requires to be resolved in the leader election problem.

(Refer Slide Time: 03:15)



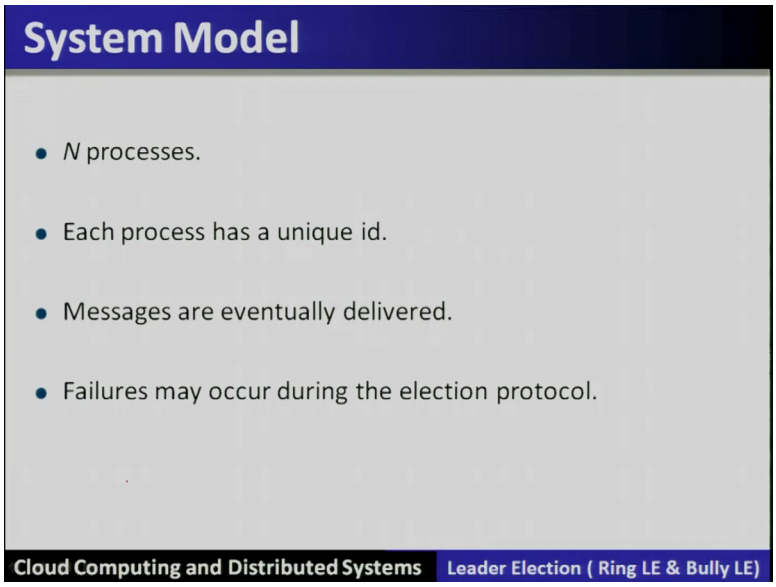
Similarly, there is another example of the group of servers in NTP that is network time protocol that we have seen among them one of them is called the root server. So, the one who is the root is the leader among all other servers. Similarly in other systems that need the leader elections are such as Apache Zookeeper and Google's Chubby system. So, the leader is useful for the coordination across the distributed servers that too we have seen in the cloud computing also comprises of several servers, so the coordination across all the servers is to be done through a server. So, that is the requirement for distributed and the cloud computing system.

(Refer Slide Time: 04:09)



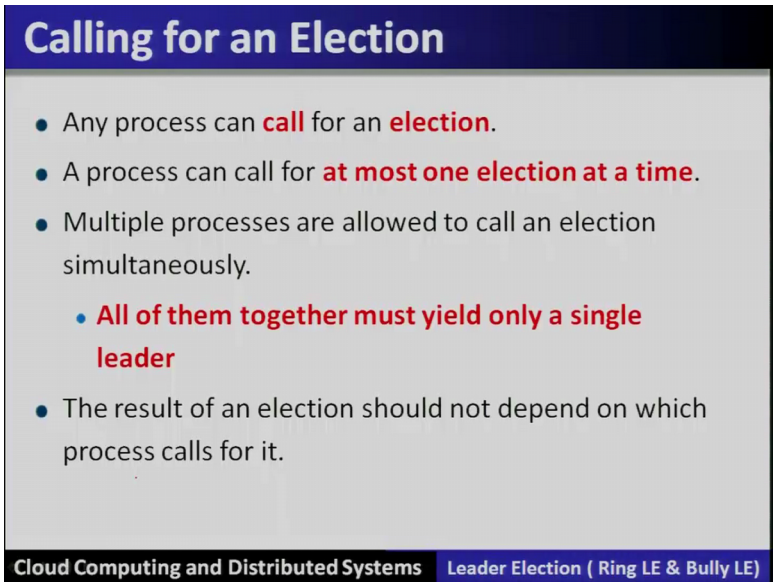
So, the leader election problem, in a group of processes elect a leader to undertake the a special task and that everyone knows in the group who the leader is. And if the leader crashes then some process will detect this failure of a leader, may be sometimes using the failure detector and then elect out of the non faulty servers, one of them as the leader. So, the goal of leader election is to elect one leader among the non faulty processes and all non faulty processes agree on who is the leader.

(Refer Slide Time: 05:03)



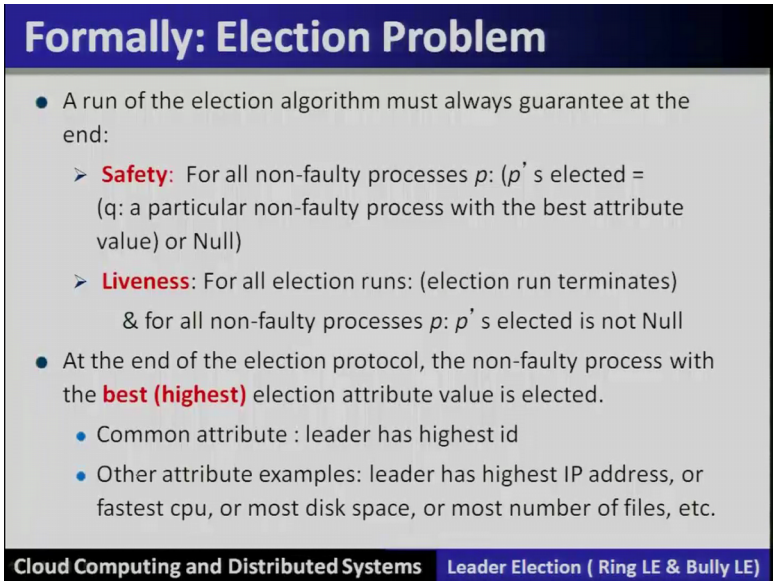
So, let us understand the system module which will be the assumption under which this particular problem will be formulated. Let us consider there are N processes. Each process has the unique id and the message are eventually delivered in a message passing system and the failures may occur during the election. Now any process can call a leader election process.

(Refer Slide Time: 05:35)



That means anyone can initiated, and a process can call at most one leader election at a time, so the multiple processes are allowed to call the leader election simultaneously, but they should lead to only one leader elected. So, the result of an election should not depend on which process calls it, and how many process are calling that leader election. At the end there should be one leader among the non faulty processes and everyone else should know who the leader is.

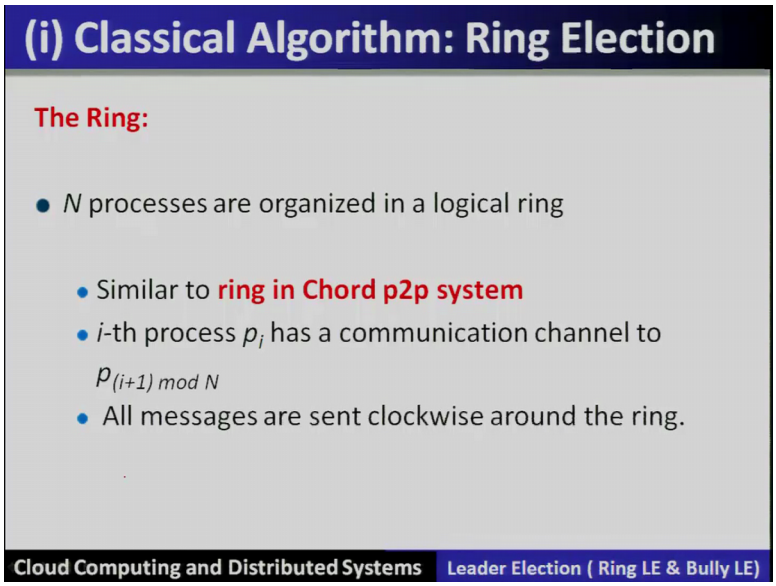
(Refer Slide Time: 06:18)



So, the leader election problem has to guarantee two properties; the first one is called safety property, the other is called liveness property. Safety property says that nothing bad happens that is for all non faulty processes p. So, p is elected one having the best attribute among all the other non faulty processes, this is called a safety condition. The second one is called liveness that eventually something good happens; that is for all election runs, that is election process terminates and for all non faulty processes, the p s elected is not null that means, the algorithm eventually terminates and where a leader elected among the non faulty processes.

So, at the end of the election protocol the non faulty processes with the highest attribute defined is elected as a leader. So, the common attribute over which the leader is elected, is for example, the non faulty processor having the highest id becomes the leader. The other attributes for the leader election which can be used is the highest IP address or the fastest CPU or the most disk space etcetera. Let us understand the classical leader election in a ring; this is called a ring election algorithm.

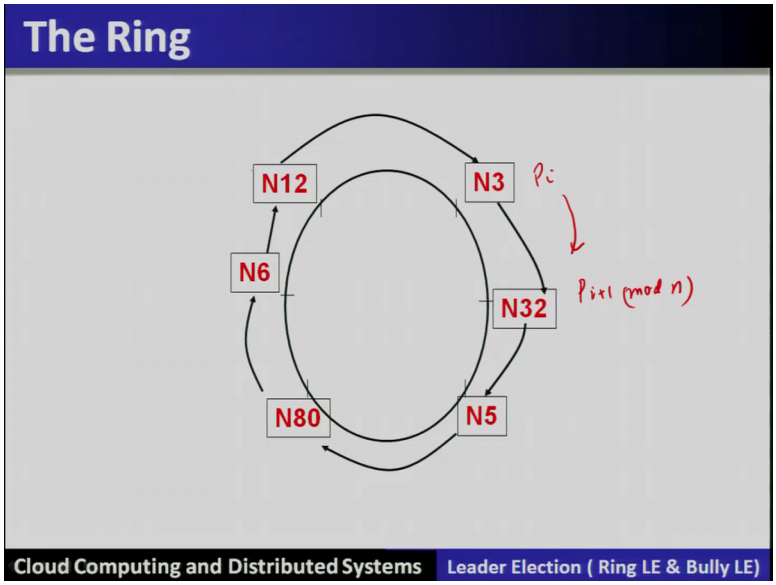
(Refer Slide Time: 07:59).



So, the ring is found out of N processes, that is the logical ring which we have seen in the chord system. Here i th process that is p i has the communication channel connection to p i plus 1 mod N. So, this way every process p i has its successor p i plus 1 mod N and therefore, defines a logical ring.

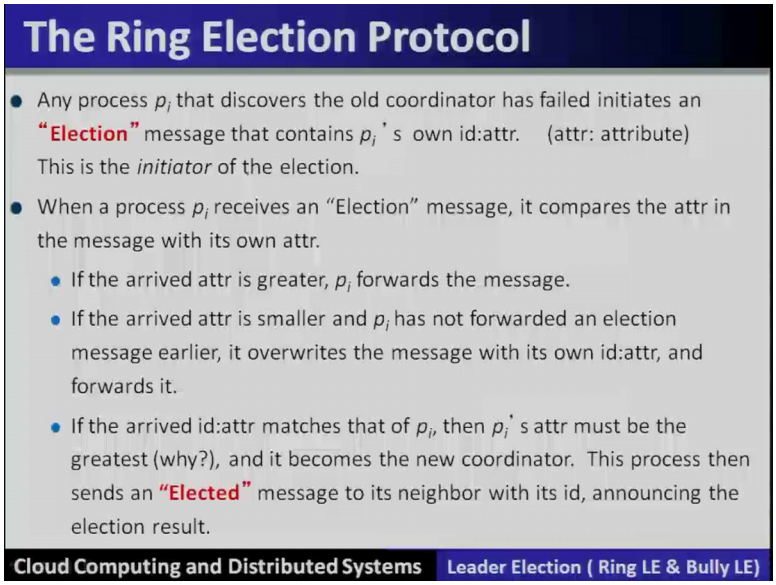
So, all the messages are sent in a clockwise around the ring. So, this is the ring which we have just defined that is p i, will basically be able to communicate with p i plus 1.

(Refer Slide Time: 08:50)



So, N 3 will have its successor N 3 2 and N 3 2 is having a successor N 5 and so on. Finally, the node N 12 will have its successor N 3 and together they will form a logical ring.

(Refer Slide Time: 09:25)

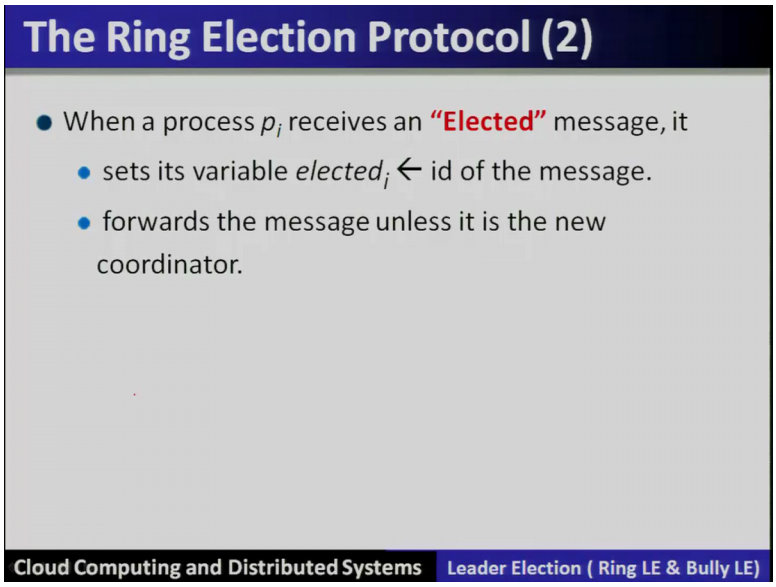


So, the ring election protocol; any process p i that discovers that the old coordinator has failed will initiate a election algorithm, using a message called election, that contains the

p i s own id that is the attribute. So, this is the initiation of the election process. So, when p i receives an election message it compares with attributes in the message with its on attribute, if the arrived message is greater then p i forwards the message. If the arrived attribute is smaller and p i has not forwarded an election message earlier, it overwrites the message with its own id and forwards it.

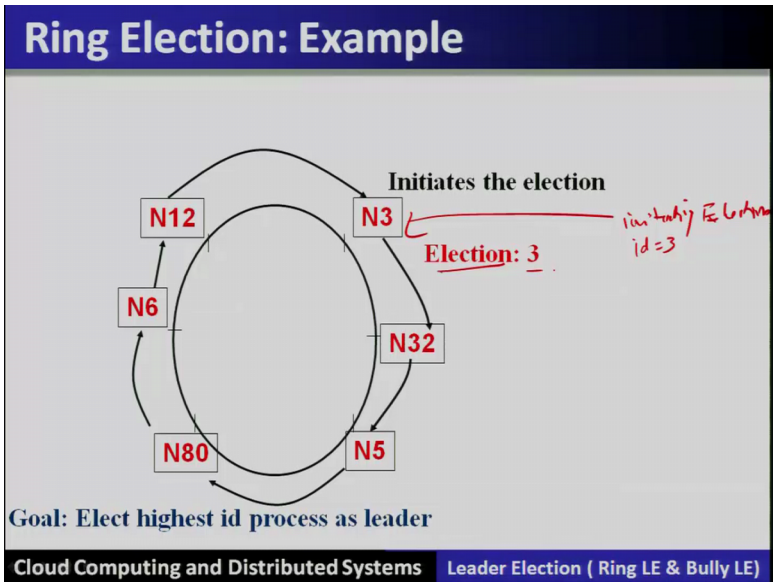
If the arrived id matches that of p i then p i s attribute must be greater, and if it is happening at all the processes and comes back then to the same one, then it is the new coordinator, this process then sends the elected message to its neighbor with its id announcing the election result. Let us see through a working example here before that.

(Refer Slide Time: 11:07)



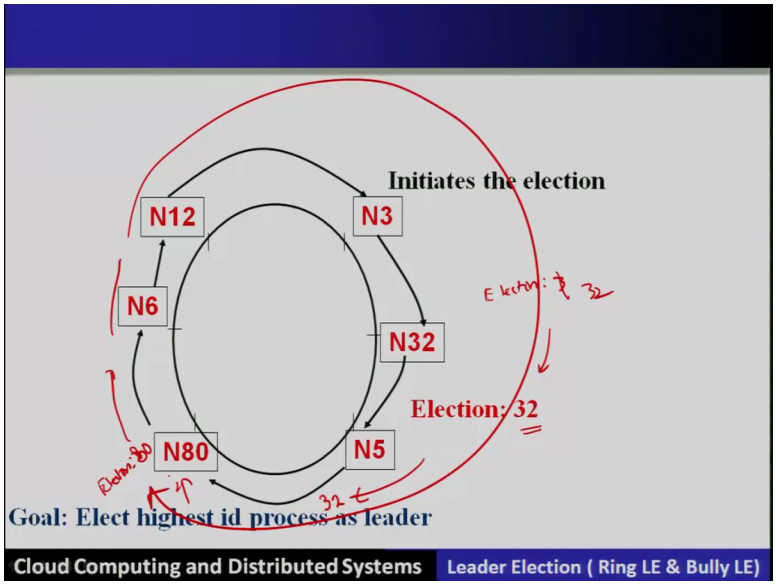
So, when a process p i receives a elected message, it sets the variable elected as its id of the message and forwards the message, unless it is the new coordinator. So, let us see how this particular leader election algorithm will work.

(Refer Slide Time: 11:30)



Let us say that N 3 is initiating the election with its id that is 3. So, the election message and with this id will go to N 32, N 32 will check the id of incoming message.

(Refer Slide Time: 12:03)

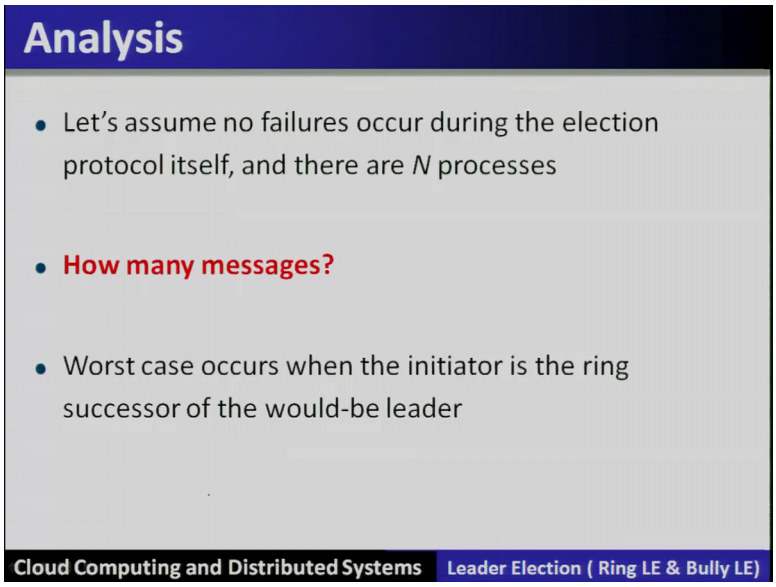


Now, N 32 is greater than 3, then it will overwrite in its message, 3 will be overwritten with 32 and it will be forwarded further and when it reaches at N 5 which is lower id, then it will forward N 32 and when it reaches at this particular N 80. So, N 80 will overwrite its id in the message and forwards that election N 80. Similarly it will forward from here and so on and it will come back again over here. So, then once that particular

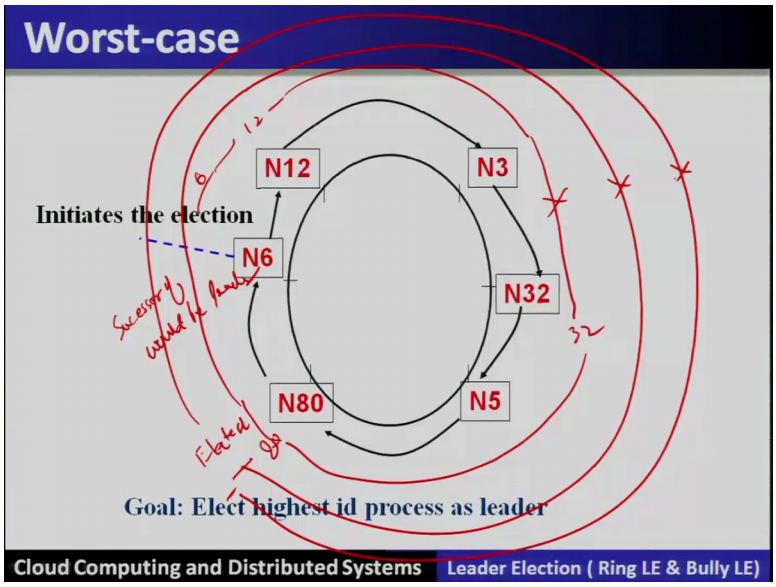
message will come back again to the same highest id process, then it will change from election to elected message and this will be circulated so that, so that all other processes knows that who is the leader which is elected.

So, let us see the analysis. So, here we are assuming there are no failure during the election itself and there are N processes, so, we will count how much is the complexity in terms of messages.

(Refer Slide Time: 13:21)

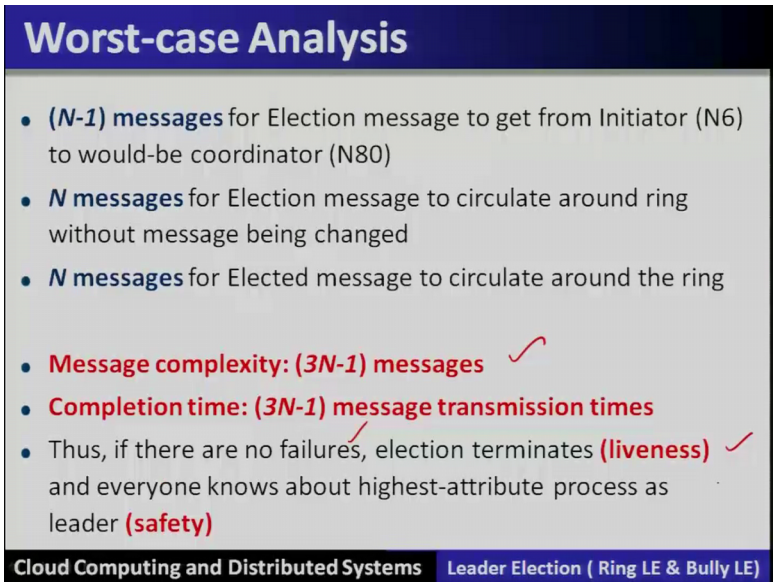


(Refer Slide Time: 13:28)



So, in the worst case situation happens, when the successor of would be, would be leader will become the initiator, the process it will go it will change at as 12 and here also 12 will continue, here it will change to 32 and it will continue, here it will change 80 and now it will continue and here then it will be changed to elected and then it will continue; likewise and here it will terminate. So, just see that how many times 1 2 3, 3 times the total messages will take around.

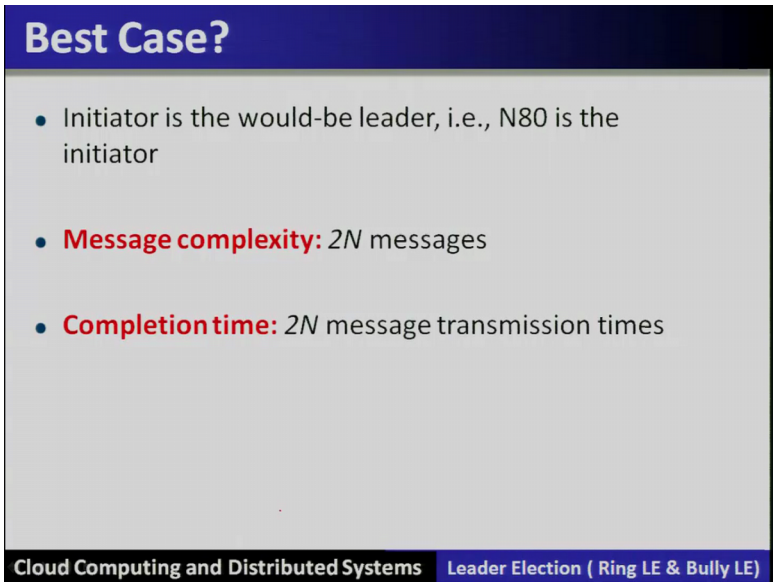
(Refer Slide Time: 14:46)



So, let us say that N minus 1 is basically the number of messages in a round. So, there are 3 N minus 1 messages, which are required in this particular algorithm in the worst case.

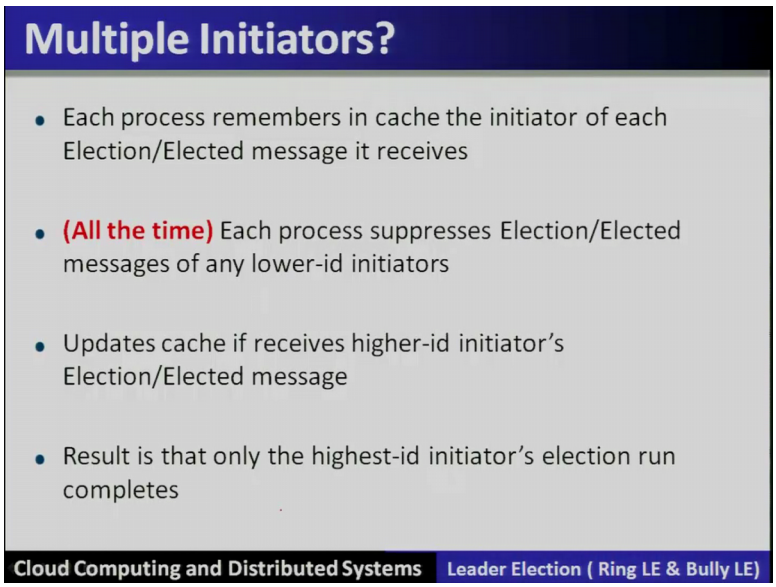
Now, here there are no failures, hence the election will terminate eventually and this ensures the liveness property and everyone will know that the highest attribute process is elected as a leader, and there is only one highest attribute process, hence the safety is ensured.

(Refer Slide Time: 15:23)



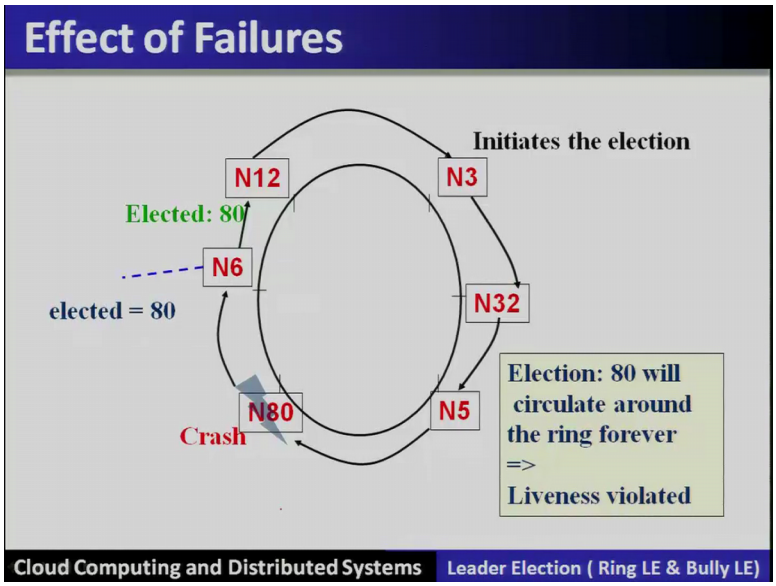
So, the best case is that when the would be leader will become an initiative in that case only 2 N message complexities will be there. How about multiple initiators? So, each process in that case will remember in the cache, the initiator of each election message it receives at all point of time the process suppresses, the election messages of any lower id initiators.

(Refer Slide Time: 16:00)



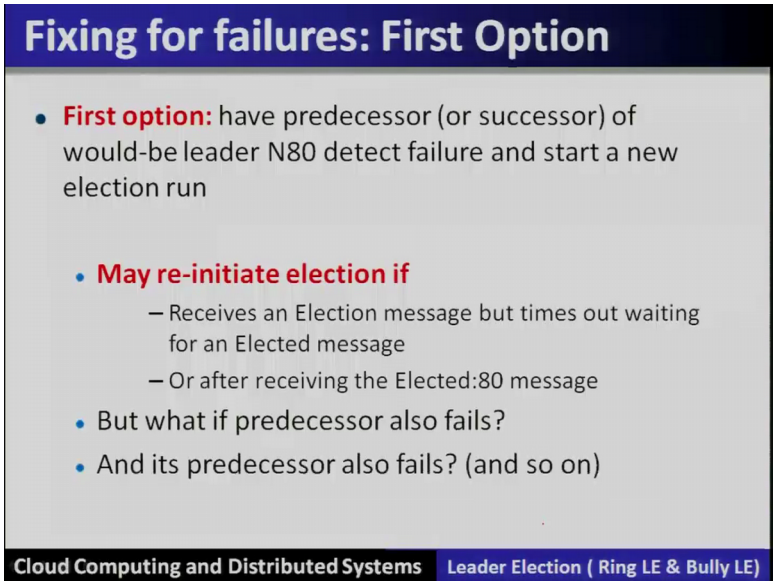
So, the result is that only highest id initiators election run will be completed in that case.

(Refer Slide Time: 16:07)



Now, let us see another situation where we allow the failures. Now what happens after the node releases the elected 80, the node crash. Then in that case this particular message keeps on circulating forever and it will never be consumed and stopped. Hence the liveness will be violated in this case.

(Refer Slide Time: 16:50)

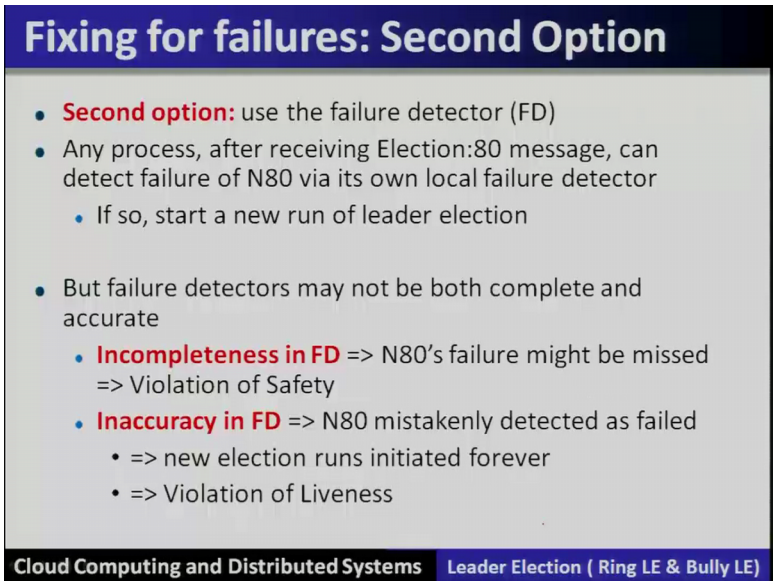


Now to fix up this particular problem the first option is that, the predecessor or the successor of the would be leader will detect the failures and start a new leader election algorithm, may re initiate the leader if the received election message, but it becomes a

timeout for looking up the elected leader or after receiving the elected leader this becomes a timeout.

Hence it will re initiate that election in that case what if that predecessors are also fails and so on. So, the second option is to use a failure detector that any process after receiving the election message can detect the failure of the highest id via its own failure detector, if so then it will start the leader election, but there are issues with the failure detectors may not be complete and accurate.

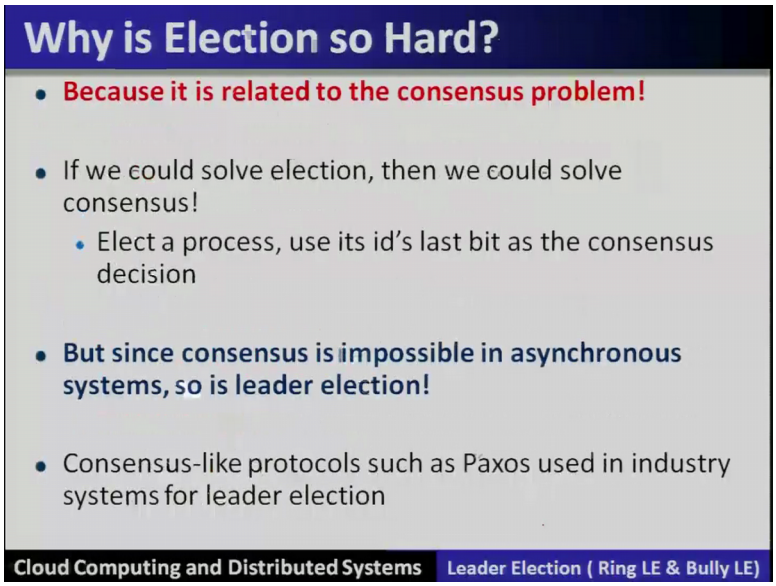
(Refer Slide Time: 17:48)



So, if it is incompleteness is there in the failure detector then it might be missed and hence it will violate the safety in that case.

Similarly, the inaccuracy is in failure detector by mistakenly detected as the failed message.

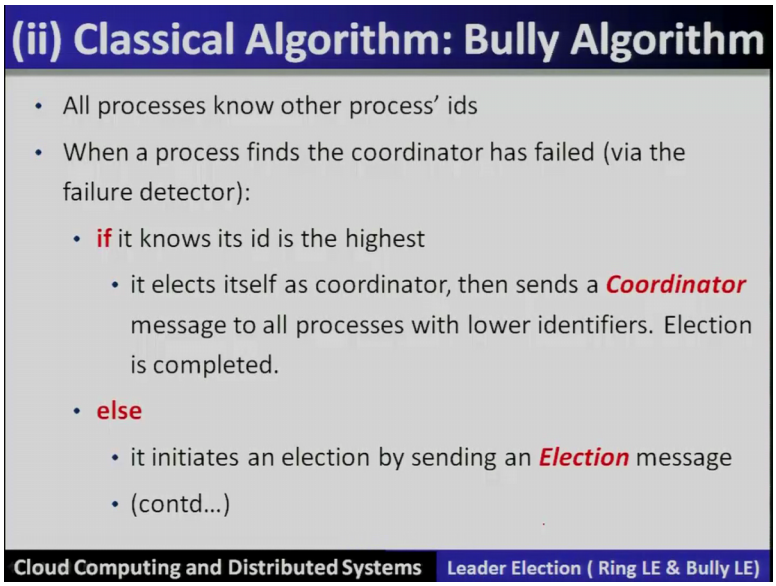
(Refer Slide Time: 18:05)



So, why the election is so hard problem? Election is hard because it is related to the consensus problem; that means, if you could solve the election process then we could also solve the consensus problem. For example, the leader if it is elected then let us say the last id, last bit in the id can be the consensus value.

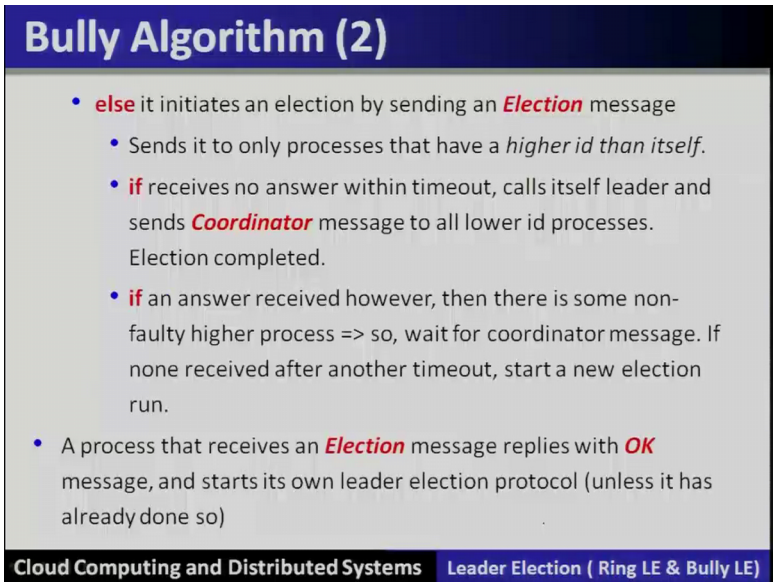
Hence if the leader election is done, then basically it will also solve the consensus problem, but you know that consensus is impossible in asynchronous system so is the leader election. So, consensus protocols such as packs of which are used in the industry system for leader election that we will see in this discussion.

(Refer Slide Time: 19:09)



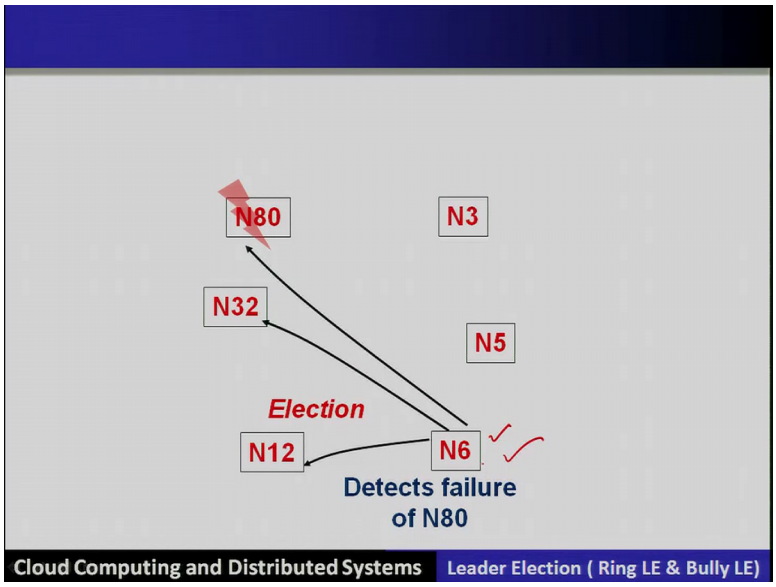
First we will see the classical algorithm that is called bully algorithm. Here all processes know other processes id, so when a process finds the coordinator has failed via the failure detector, and if it knows its id is the highest, then it elects itself as the coordinator and then sends the coordinator message to all the processes with the lower id and the election is completed, else it initiates an election by sending an election message sends it to only the processes that have the higher than itself, if receives no answer within the timeout then calls itself as the leader and sends the coordinator message to all the lower id processes and the election is completed

(Refer Slide Time: 19:43)



If an answer is received; however there is some non faulty higher id process. So, it will wait for the coordinator message, if none are received after another timeout then it will start a new election transfer. So, a process that receives an election message replies message and it starts its own leader election algorithm.

(Refer Slide Time: 20:27)

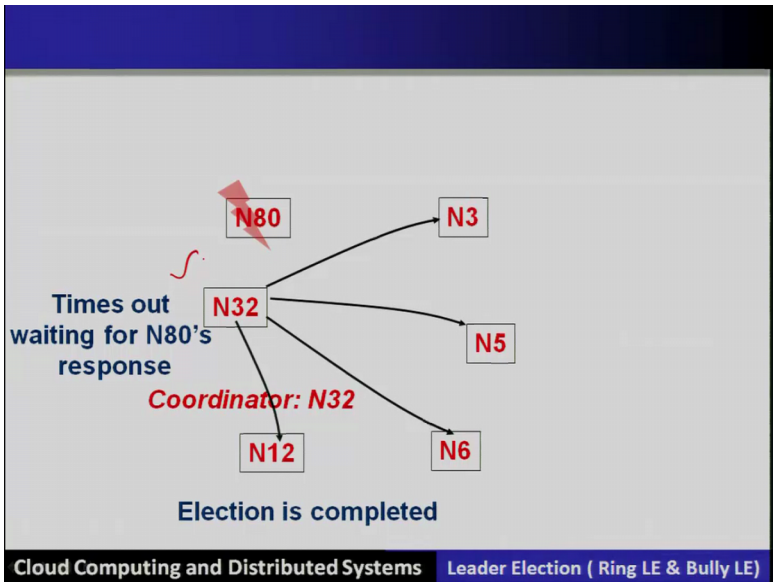


So, here let us see through an example. Now let us say that the highest id N 80 is failed and it is being detected by a process having the id N 6. So, it detects through a timeout failure detector at N 80 is down. So, it sends the message to the nodes who is having the higher id then its own id. For example, here N 6 the nodes which is have higher id than N 6 is N 12, N 32, N 80.

N 3 and N 5 they are having ids lower than N 6, so it will not be sent. So, having received these particular message, so N 12 is having higher id, so it will send a ok in the sense he will take care of since it is having the higher id, so he will initiate the leader election and also N 32 is having higher id then N 6 they will also send the message for N 6 and N 6 will now be waiting for the outcome.

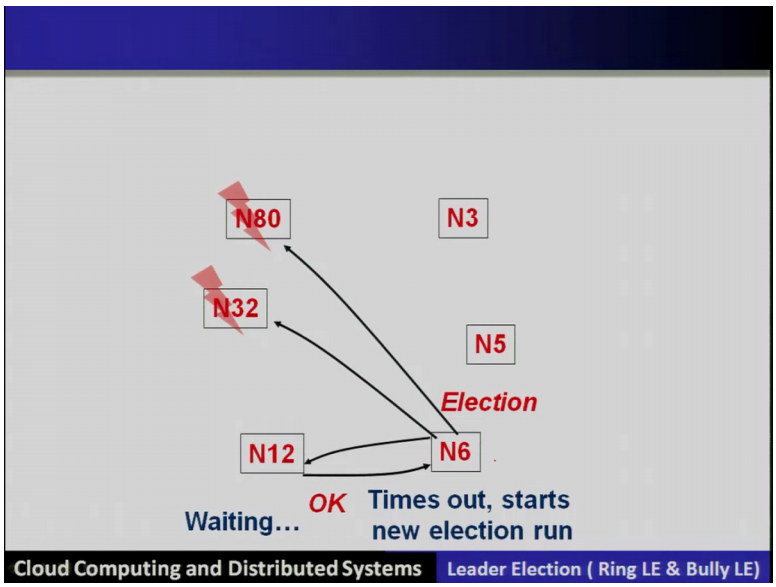
Then N 12 will send the election to its higher id that is N 32 and N 80 the same way and N 32 will send the ok, so N 12 will also be waiting. Since N 80 is faulty, so N 32 will not receive the, from N 80. So, N 32 will timeout and will assume as the coordinator. So, it will now send the coordinator message to all other nodes in the system and hence the election is complete

(Refer Slide Time: 22:11)



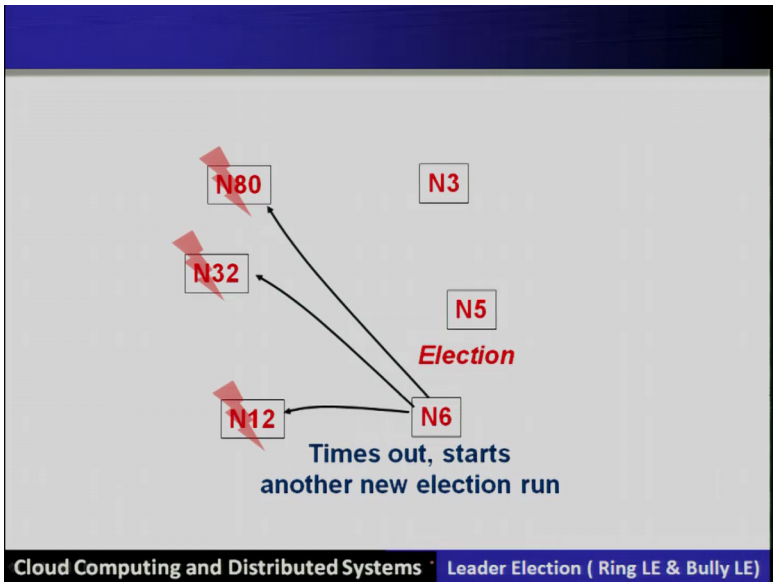
Now if N 32 also is failed, then these two waiting process they will timeout and N 6 will timeout before N 12.

(Refer Slide Time: 22:27)



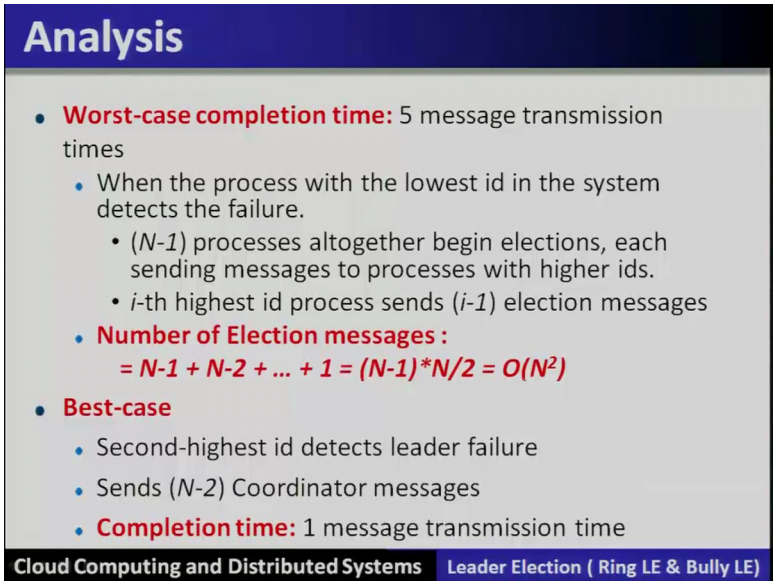
So, again it will run the election and send this messages, election message to the nodes of having higher id than N 6. Only N 12 is alive, so it will send the message. If N 12 is also crashed then it will timeout and it start another leader election where in N 6 will be elected then.

(Refer Slide Time: 22:55).



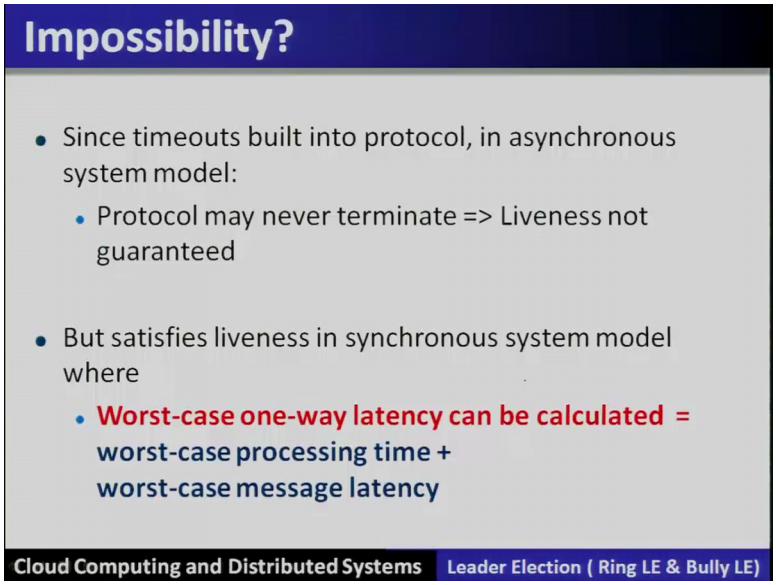
So, here irrespective of how many failures are there the algorithm will terminate with that we have seen in the Bully's algorithm. So, if the failures stop, eventually a leader will be elected. So, how do you set this timeouts will based on the worst case time to complete the election. So, the 5 message transmission times here, if there are no failure during the run that we have already seen. So, the analysis of this algorithm, if we see how many number of election messages are send.

(Refer Slide Time: 23:32).



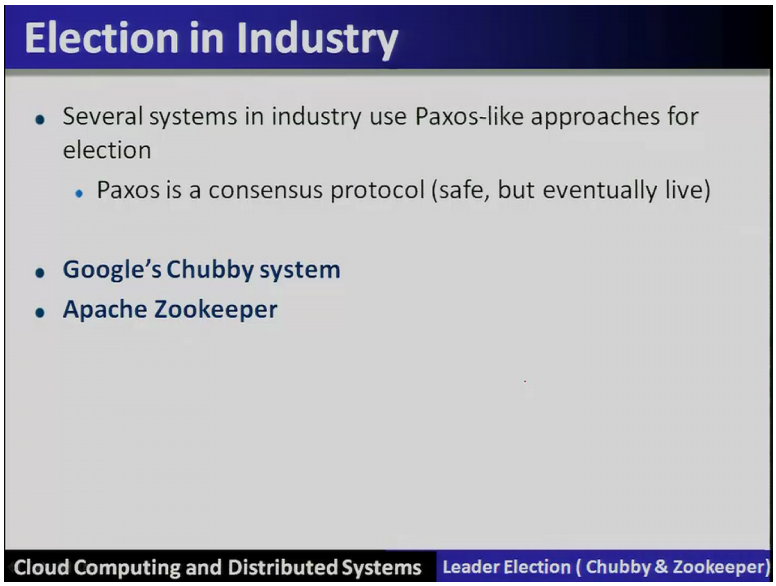
So, using 5 different message transmission time, the total number of messages which are send is N minus 1 plus N minus 2 and so on 1. So, if we some up it will be of the order N square, the best case will occur when the second highest id detects the leader failure that we have seen earlier. So, it will send N minus 2 coordinator messages and the completion time is only one message transmission here in this case.

(Refer Slide Time: 24:07)



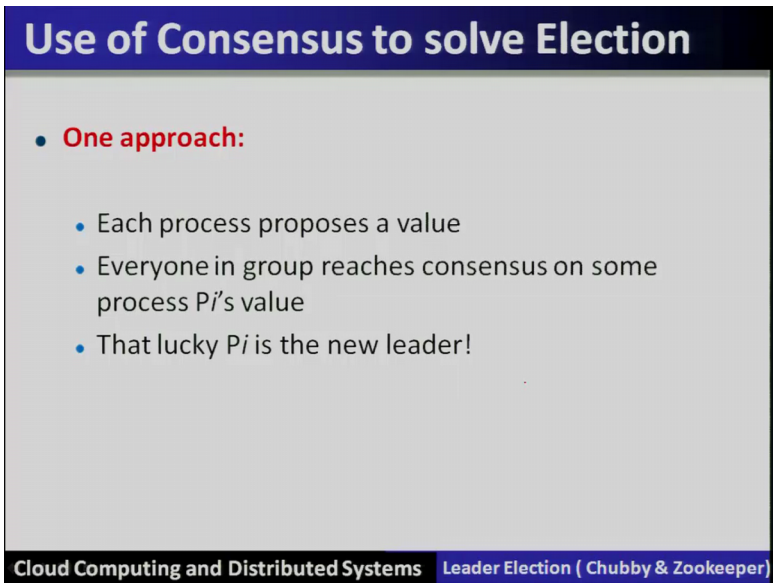
Let us see the impossibility since the timeouts built into the protocol in the asynchronous system model. So, the protocols may never terminate and the liveness is not guaranteed here in this case, but it satisfies the liveness in the synchronous model, they are the worst case one way latency can be calculated. We will see now the leader election in the industry system such as Google's Chubby and Apache Zookeeper.

(Refer Slide Time: 24:35).



So, this leader election in this industry system is different than the approach, which we have seen using the process, sending a proposed value and reaching agreement.

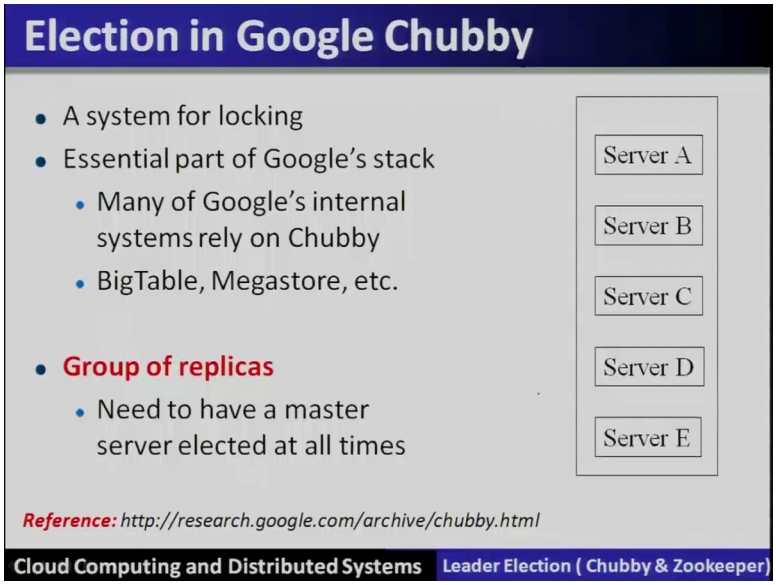
(Refer Slide Time: 24:49)



Instead of that industry uses the systems like Paxos for the leader election. We have seen earlier that Paxos is a consensus protocol which is now being used in the industry for the leader election. The Paxos is safe, but it is not guarantying the liveness property, it is guarantying the eventual live. So; that means, there are some scenarios where in the algorithm will never terminate, but it says that eventually it is live that is most of the

cases everything goes fine, then basically the algorithm terminates. Let us see how the Paxos is used in the industry system; like Google Chubby and Apache Zookeeper for leader election.

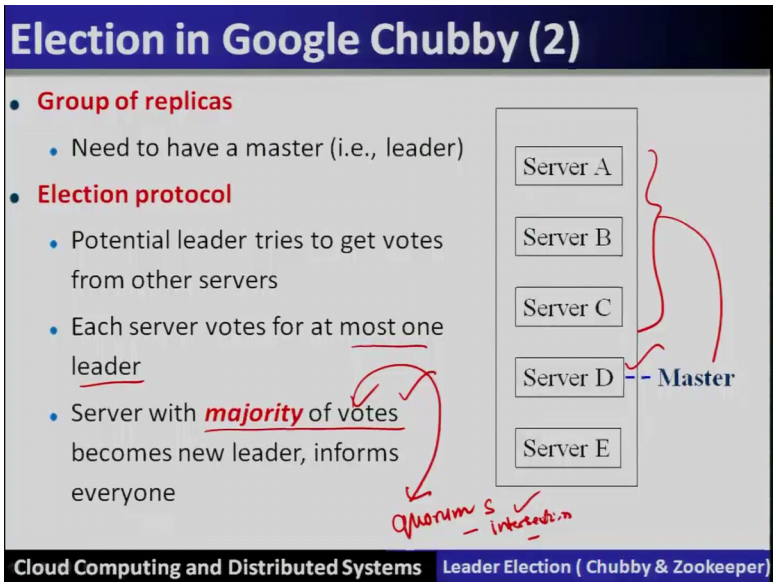
(Refer Slide Time: 25:48)



So, Google Chubby is a distributed locking system and is also a essential part of a Google's stack, and many of the Google's internal systems rely on this particular Google Chubby; such as BigTable and megastore.

In this particular system, there is a group of replicas. Here in this example we have shown server A B C D E, there are 5 different servers, they are all replicas and there is a need to elect a leader at all points of time among them. So, this particular example shows that here, let us say the server D is elected as a leader called master and this particular leader is elected using a election protocol.

(Refer Slide Time: 26:33)

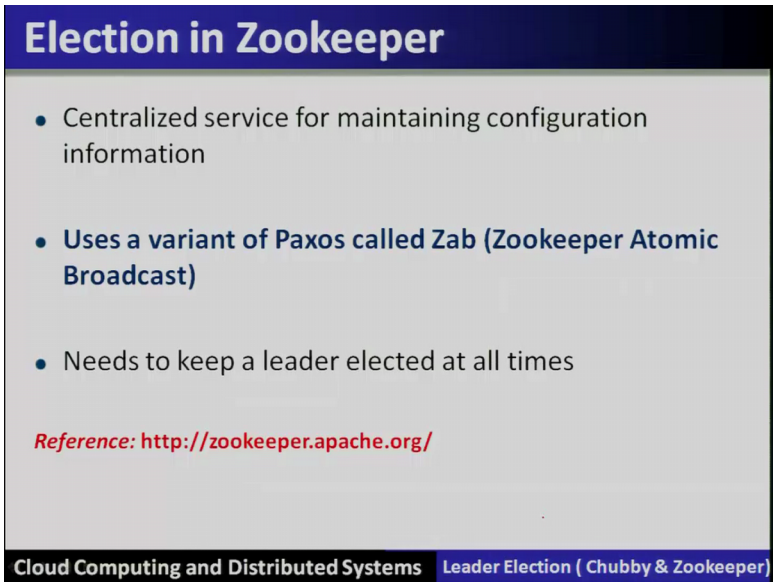


So, the potential leader tries to get the vote from the other servers. So, each server will vote at most one leader. So, the server with the majority of the vote, now become the leader and he informs about this decision. So, this particular election is based on that majority of the servers vote whose ever gets.

For example here there are 5 different servers, so majority becomes 3. So, if 3 or more any server gets the votes, then basically that particular server will be elected as the leader. Now why this majority not all? So, majority is basically equivalent to the quorums. So, in every quorum are the intersection of the quorums, there is exactly one common node between the quorums and you know that every server has to vote at most one leader; therefore, this particular election using the majority of the votes that is following the quorum principal.

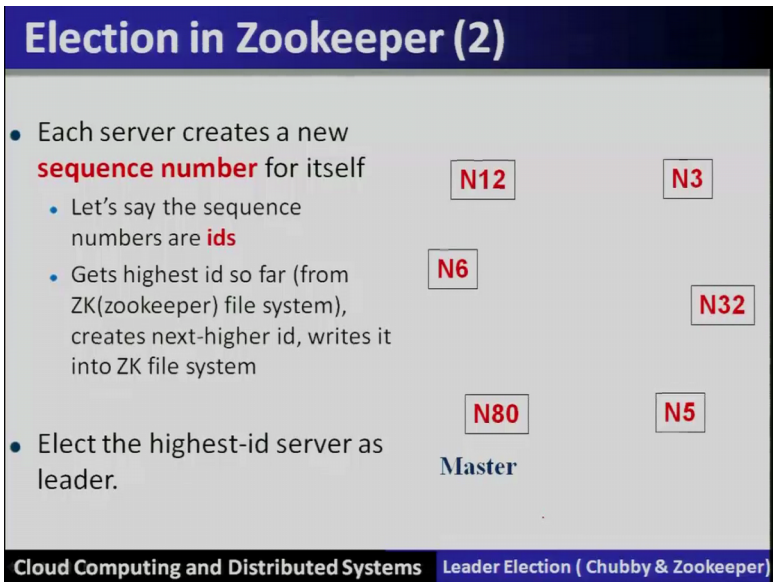
Hence it ensure the safety that at most; this particular leader will elect only one coordinator or a one master.

(Refer Slide Time: 28:18).



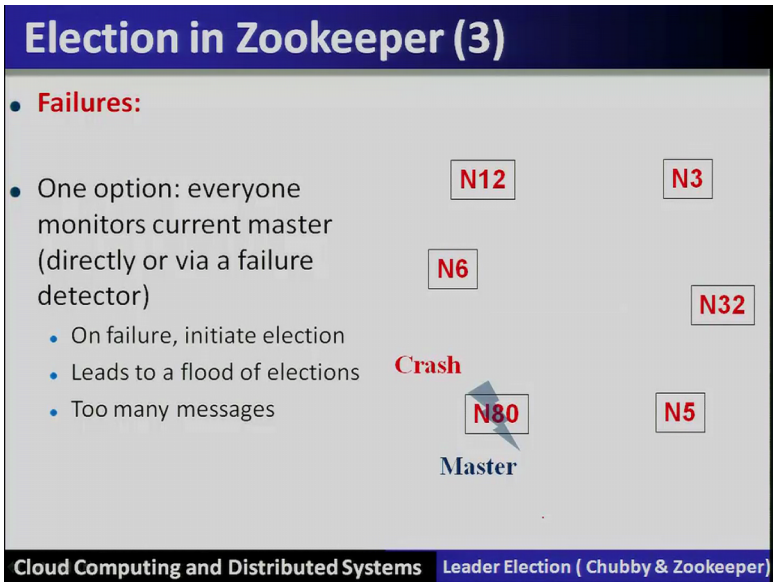
So, election in a zookeeper, zookeeper is a centralized service for maintaining the configuration information. So, zookeeper uses a variant of Paxos which is called Zab, zookeeper atomic broadcast that need to keep the leader elected at all points of time.

(Refer Slide Time: 28:37)



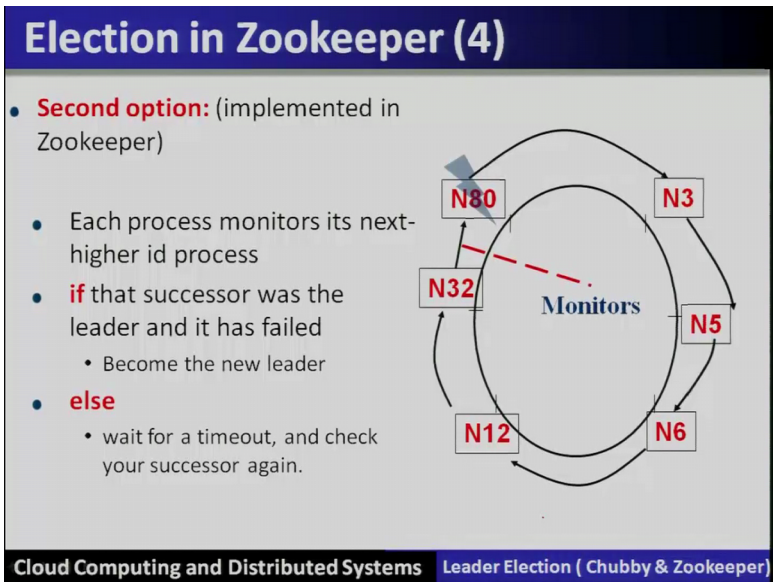
Let us see how the election in a zookeeper is done. Here each server will create a new sequence number for itself. Let us say the sequence numbers are the ids, it gets the highest id so far seen and that is written into a zookeeper file system. So, the highest id server will be elected as a leader in this particular situation regarding failures.

(Refer Slide Time: 29:10)



One option that everyone monitors the current master and on failure it will initiate again the election, and it may lead to the flood of elections and too many messages will be there.

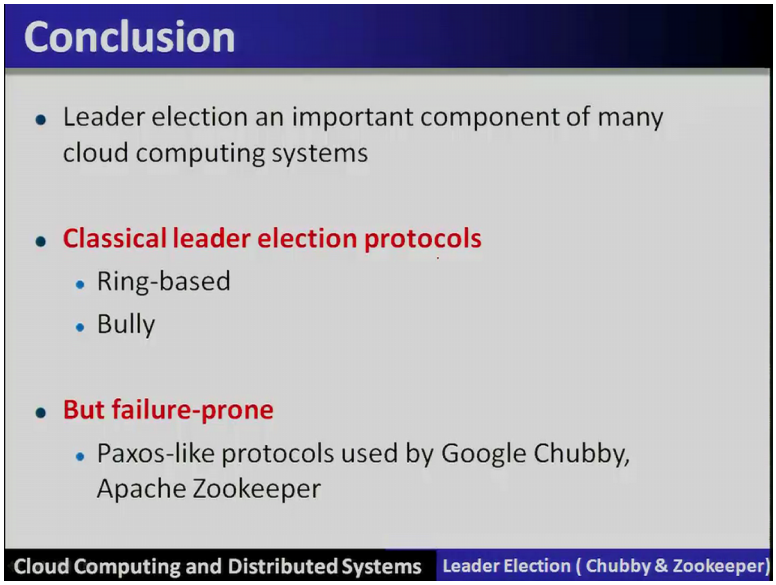
(Refer Slide Time: 29:29)



So, there is second option which says that each process monitors its next higher id process, so if the successor who was the leader is failed and it will become a new leader. For example, the current leader was let us say N 80 if it failed, the successor N 32 successor will identify will monitor it and it will become the leader if phase successor is

not responding, else it will wait for a timeout and check the other successor and if that is also not responding then it will become the leader.

(Refer Slide Time: 30:10)



So, conclusion leader election is an important component in many cloud computing applications and systems, primarily it is used for the coordination. So, we have seen the classical leader election algorithm that is ring based leader election and bully leader election. We have also seen the industry uses the protocol Paxos like protocols, which are used in Google Chubby and Apache Zookeeper for leader election, which also works in the failure situations.

Thank you.