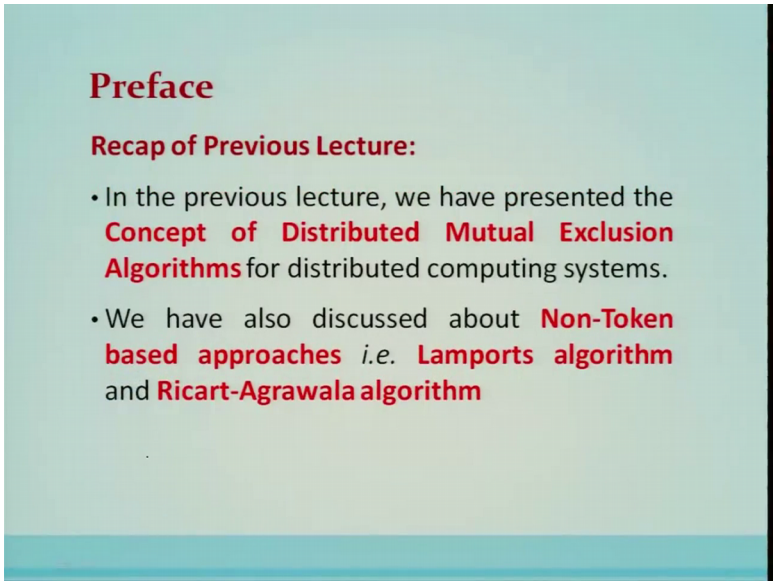
Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture – 08
Quorum Based Distributed Mutual Exclusion Algorithms

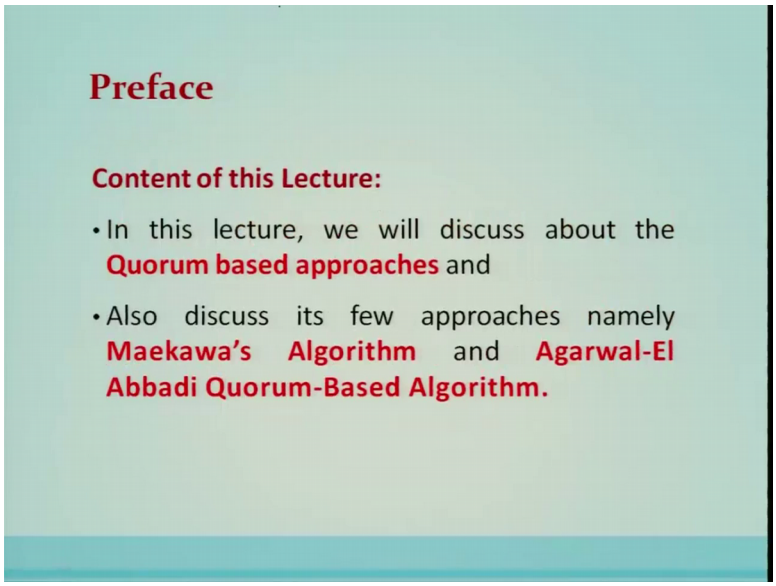
(Refer Slide Time: 00:20)



Lecture 8 Quorum based distributed mutual exclusion algorithms. Preface: Recap of previous lecture. In the previous lecture, we have presented the concept of distributed mutual exclusion algorithm for distributed computing system. We have also discussed about non token based approaches that is Lamports algorithm Ricart Agrawala algorithm. In distributed mutual exclusion we have seen that the mutual exclusion problem is fundamental to the distributed systems.

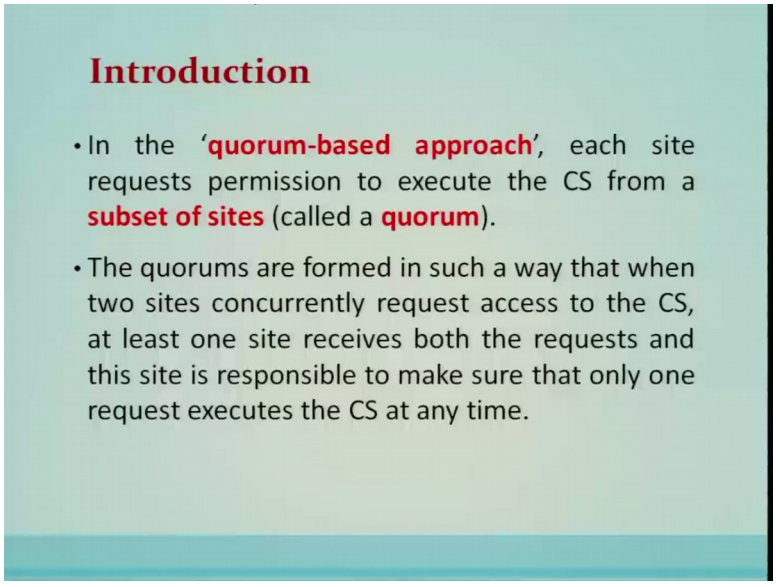
Now, this particular mutual exclusion ensures the concurrent axis of processes to the shared resources or the data is serialized. So, this particular problem states that, only one process is allowed to execute the critical section at any point of time. So, in distributed system the shared variables or the semaphores or a local variable through the kernel can be allowed because there is no common memory in the distributed system.

(Refer Slide Time: 01:21)



Content of this lecture. In this lecture we will discuss about Quorum based approaches for the distributed mutual exclusion algorithm. Also we will discuss few approaches namely Maekawas algorithm and Agarwal El Abbadi Quorum based algorithms Quorum based approaches.

(Refer Slide Time: 01:42)

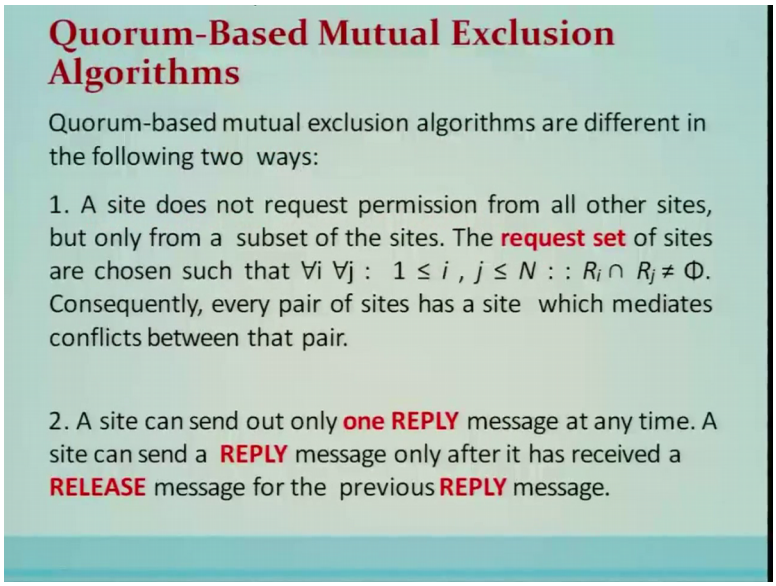


Introduction: In the Quorum based approach each site requests permission to execute the critical section from a subset of the sites that is the quorum; that means, all the sites are not required to be to be taken permission to execute any critical section. And this is the

deviation from the previous lectures where all the sites are required to be taken into account before request is being granted to execute the critical section.

So, the Quorum based. So, the quorums are formed in such a way that when 2 sites concurrently request access to the critical section at least one site it receives both the request and this site is responsible to make sure that only one request executes the critical section at any point of time.

(Refer Slide Time: 02:35)

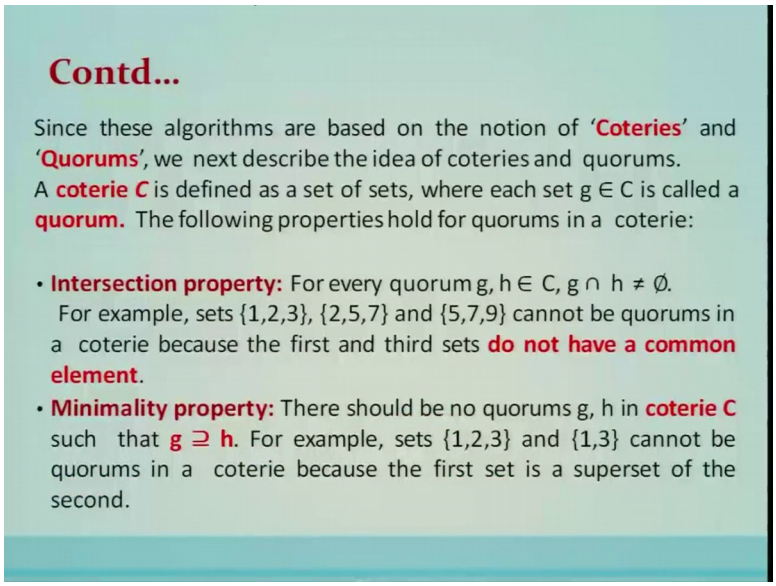


Quorum based mutual exclusion algorithms. Quorum based mutual exclusion algorithms are a different in the following 2 ways.

So, a site does not request permission from all other sites, but only from a subset of the sites. And that subset of the site is called a request set of a site. So, the request set of the sites are chosen such a way that for i and j. So, in the request set of i and request set of j for any i j between 1 to n. So, the intersection is not null, that is consequently every pair of sites has a site which mediates conflicts between that pair.

So, a site can send out only one reply message at a time. A site can send reply message only after it has received the release message from the previous reply message. Since the algorithms are based on the notion of coteries and quorums.

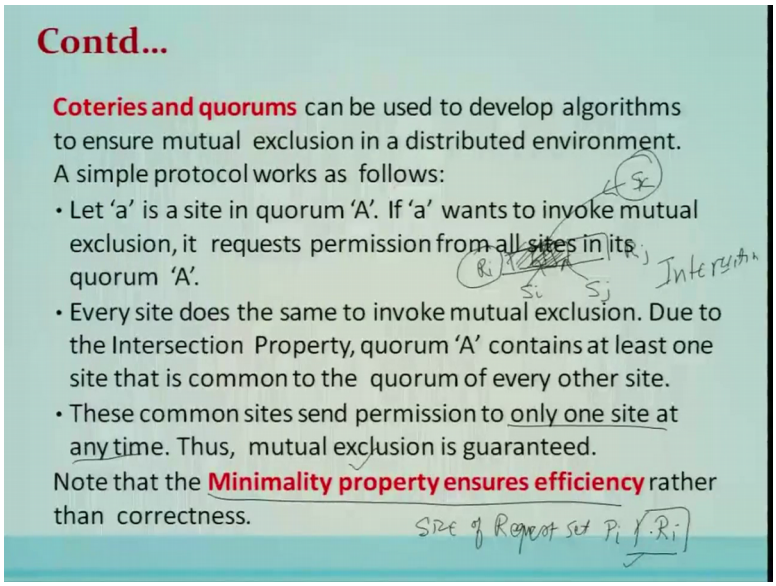
(Refer Slide Time: 03:30)



We next described the idea of coteries and quorums a coteries C is defined as a set of sets, where each set g that is in coteries C is called a Quorum the following properties hold for the Quorum in a coterie.

So, the first property is called intersection property. So, for every Quorum g edge which is in coterie. So, the intersection of these quorums g and h is not null. For example, the sets 1 2 3, 1 2 3 and 2 5 7 cannot be Quorum in a coterie, because the first and third sets do not have a common element. Minimality property there should be no quorums g and h in coterie C such that g is the superset of h, for example, the sets 1 2 3 and 1 3 cannot be cannot be the quorums in a coterie because the first set is a superset of the second one.

(Refer Slide Time: 04:59)



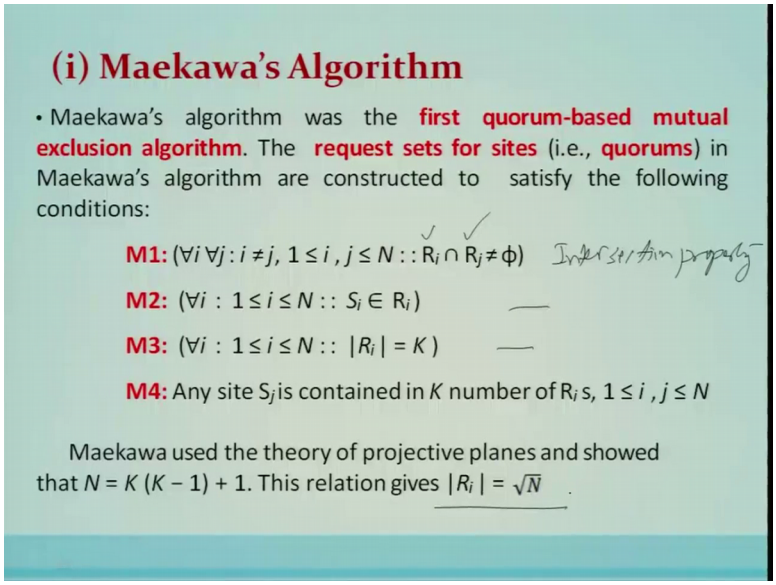
So, there are 2 properties which we have seen is required here for the Quorum to be qualified and the coterie intersection property and minimality property coteries and quorums can be used to develop algorithm to ensure mutual exclusion in a distributed environment a simple protocol works as follows. Let A is a site in a Quorum capital A and if the site a wants to invoke mutual exclusion it requests permission from all sites in it is Quorum A.

So, every site does the same to invoke the mutual exclusion due to the intersection property. Quorum A contains at least one side that is common to the Quorum of every other site. These common sites send the permission only at one site at a time. Thus mutual exclusion is guaranteed. Note that minimality property ensures efficiency rather than correctness. So, this particular aspect you can see that, if let us say site i and site j, they are requesting to go in a critical section. Their request set that is called Ri and their request set Rj.

So, during many were using intersection property there must be some site, let us say Sk which is common in their request set. And this will mediate if both they both Si and Sj want to enter into the critical section. So, this Sk will mediate, why because it will release only one reply or a 1 permission to ensure the mutual exclusion. So, intersection property will ensure here to guarantees that only one side at a time is allowed to execute into the critical section.

Now, another property which is called a minimality property, this ensures the efficiency rather than the correctness. So, for example, the size of the size of the request set of a particular process i, that is called Ri basically ensures the minimality property and basically the size is basically is such a small. So, that it will be efficiently or with the minimum number of message this particular mutual exclusion is ensured.

(Refer Slide Time: 07:22)

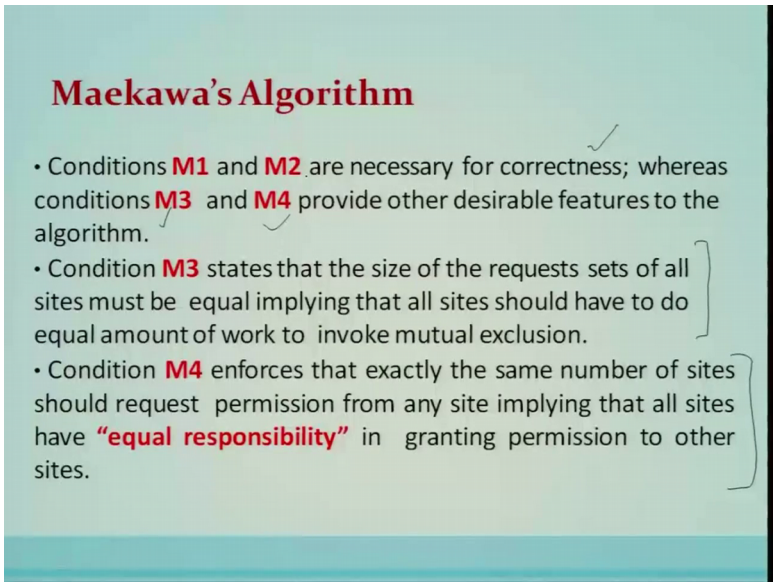


So, to have this mutual exclusion with a simple and a efficient manner, we will see the Maekawas algorithm. So, Maekawas algorithm was first Quorum based mutual exclusion algorithm that request set for the sites. That is called the Quorum in a Maekawas algorithm are constructed to satisfy the following 4 conditions. The M1 condition says that the request set of any 2 processes or any 2 sites is not null. That is the intersection property.

Now another condition says that. So, every set is a member of a request set. Now another property says that the size where request set is K. And another property enforces that any site Sj is contained in K number of Ris. So, Maekawas use the property of projective planes and showed that N is equal to K times K minus 1 plus 1 and this gives out that the quest set can be of the size root N.

So, here we can see that the size of a request that is reduced. Here in Maekawas algorithm and that is why only subset of the sets will is only required to be taken to be sent the request condition to go in the critical section.

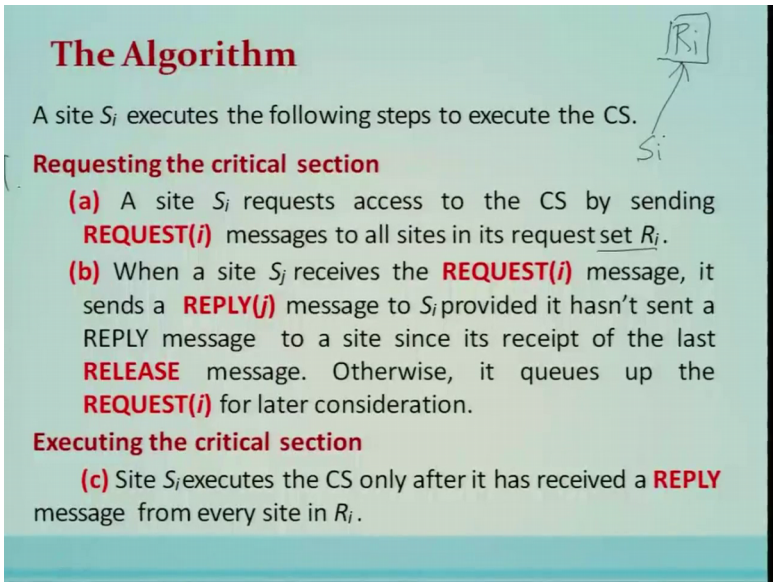
(Refer Slide Time: 09:00)



Condition M1 and M2 are necessary for the correctness whereas, M3 and M4 provide other essential features to the algorithm. Condition M3 states that the size of the request sets of all the sites must be equal implying that all the sites should have to do equal amount of work to invoke the mutual exclusion.

So, this particular M3 will ensure that particular property that all the request sets they have to do equal amount of complication and the work. Condition M4 enforces that exactly the same number of site should request permission from any site implying that all the sites have equal responsibility in granting the permission to the other sites. That also guarantees the other desirable feature of this mutual exclusion algorithm given by the Maekawas.

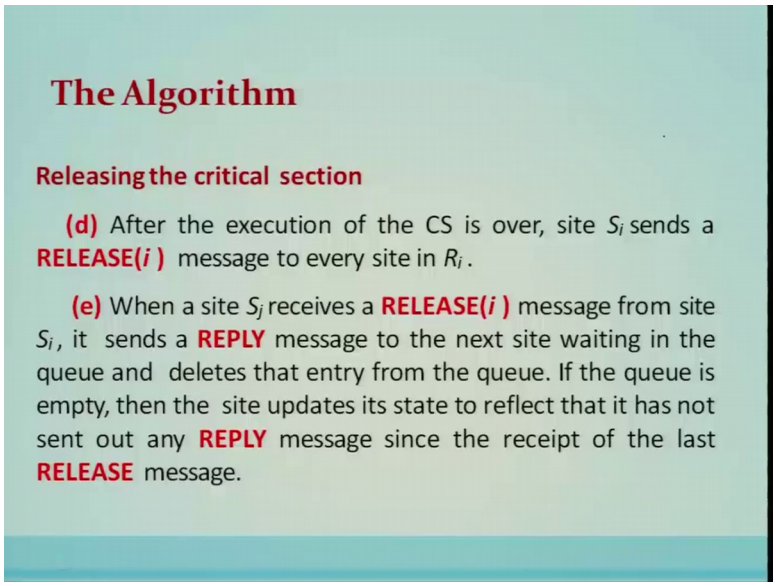
(Refer Slide Time: 09:59)



Now we have to describe the algorithm. So, a site Si executes the following steps to execute the critical section. First step is requesting the critical section. A site Si request access to the critical section by sending a request time messages to all the sites in it is request set. Now when Sj receives the request i message, it is sends a reply message to Si provided it hasn't sent a reply message to a site. Since it is a receipt of the last release message otherwise it queues up the request type for later consideration that is that will conclude the requesting the critical section that is part one.

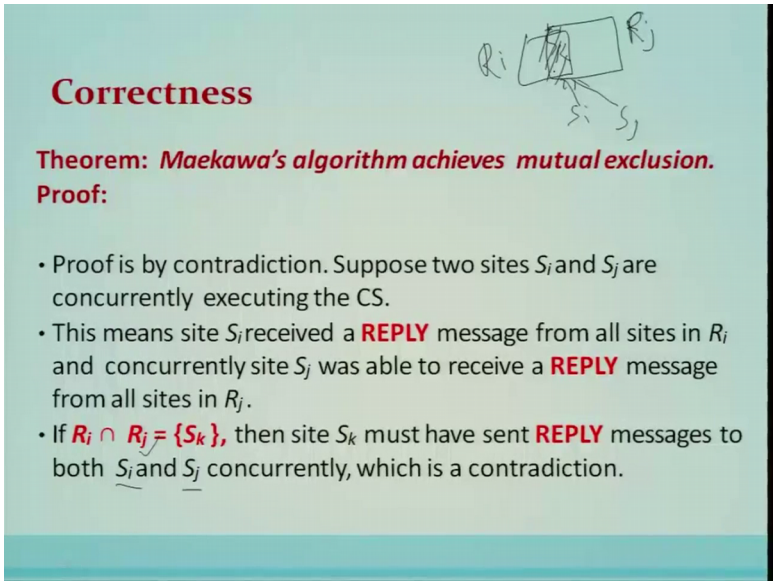
The second part of the algorithm is executing the critical section site. Si execute the critical section only after it has received the reply message from every site in Ri.

(Refer Slide Time: 11:03).



Releasing the critical section, that is after the execution of the critical section is over site i sends a release i message to every site in it is Ri. When a site Sj receives the release message from the site Si, it is sends a reply message to the next site waiting in the queue and deletes that entry from the queue, if the queue is empty then the site updates it is state to reflect that it has not sent out any reply message since the receipt of the last release message.

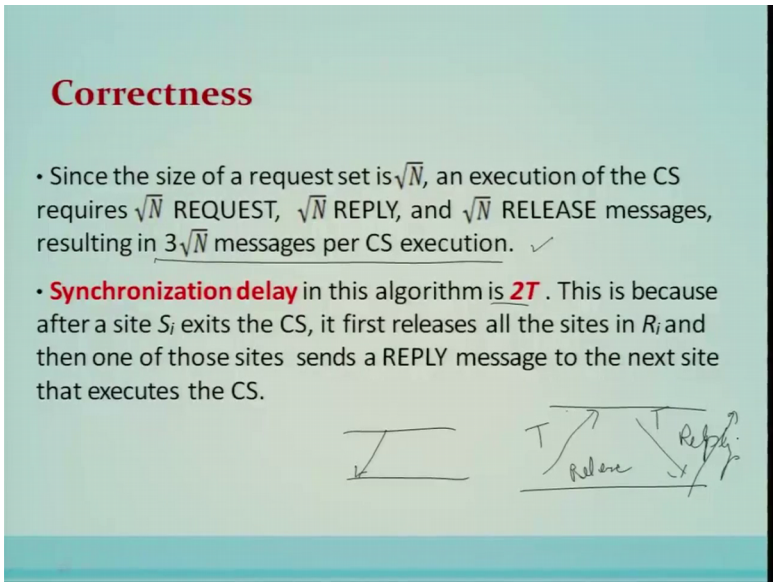
(Refer Slide Time: 11:37)



Now, the correctness Maekawas algorithm achieves mutual exclusion the proof goes by the contradiction. Suppose 2 sites i and j they concurrently execute the critical section, Si and Sj. They basically are executing the critical section. This means that site Si received the reply message from all the sites in the request set Si and concurrently Sj they also receives a reply which are from all the sites in Rj.

Now, you we know that during the using the intersection property, there is a common site Sk which is basically following this condition. That Sk must have sent a reply message to both Si and Sj concurrently because this is the common site. So, it is responsible to ensure the mutual exclusion if it has sent that reply message to both the sites; that means, it is allowing 2 particular sites to go in a critical section concurrently which is contradiction.

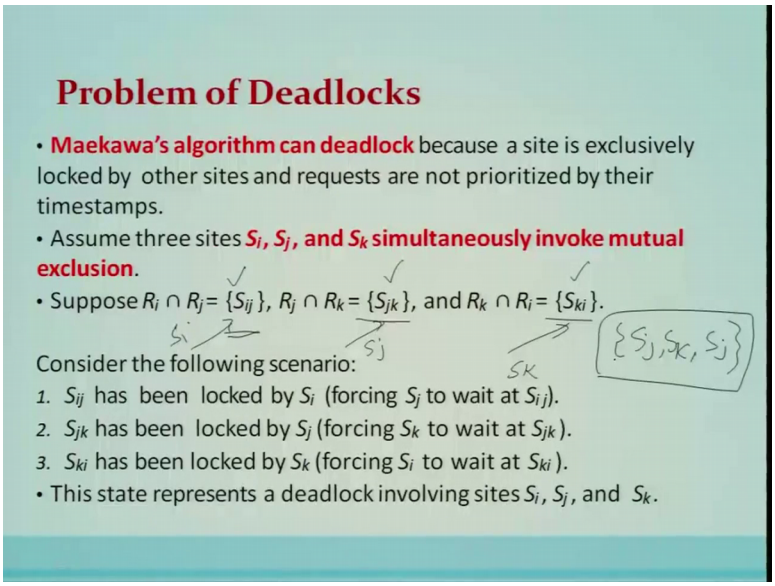
(Refer Slide Time: 12:50)



Now, the correctness since the size of the request set is root N, an execution of the critical section requires root N request message root N replying message root N release message. So, total 3 root N messages for critical section is required by the Maekawas algorithm. Now the synchronization delay of this algorithm is 2T. This is because after site Si exists the critical section it first releases all the sites in Ri and then one of those sites sends a reply message to the next site.

So, the release message and a reply message T times it will take total 2T times it will take then the next to allow.

(Refer Slide Time: 13:48)

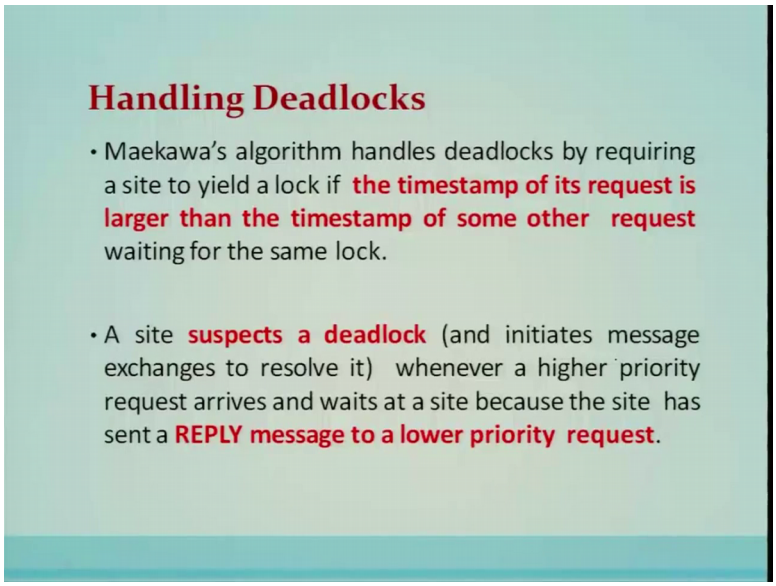


The next waiting process to go in a critical section, hence the synchronization delays 2T the Maekawas algorithm suffers from a deadlock. So, Maekawas algorithm can deadlock because a site in site is exclusively locked by the other sites and the requests are not prioritized by the timestamp assume 3 sites S1 Si Sj and Sk simultaneously in both mutual exclusion.

Suppose the request set of i and j, if we take intersection and that is a common site is Sij, similarly between j and k the common site is jk and k and i the common site is the Sk. Now consider the following situation. Now Sij has been locked by Si forcing Sj to wait at Sij. Sjk has been locked by Sj forcing Sk to wait at Sjk. Now third one, Sk I has being locked by Sk forcing Si to wait at Ski.

So, there will be set of waiting processes which are waiting for each other for this particular to unlock this state represents a deadlock of consisting of 3 sites Si Sj and Sk.

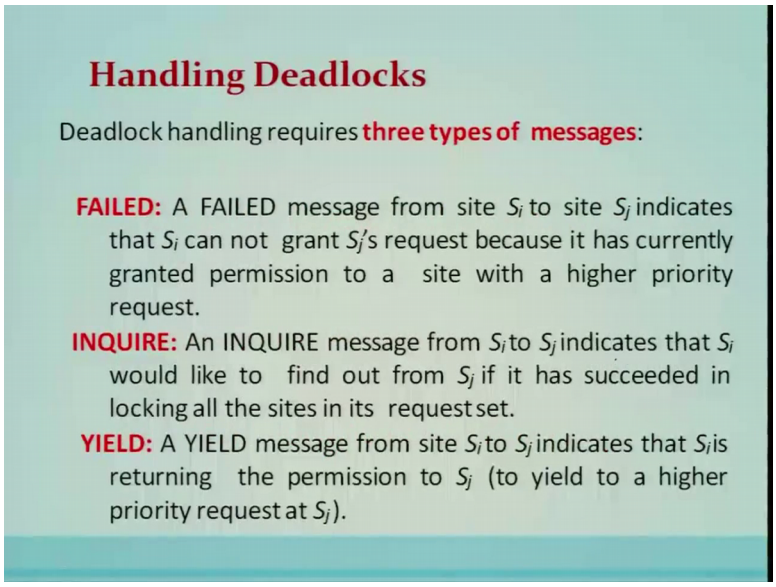
(Refer Slide Time: 15:22).



So, how to handle the deadlock in Maekawas algorithm? Maekawas algorithm handles deadlock by requiring a site to yield the lock, if the timestamp of it is request is larger than the timestamp of some other request waiting for the same lock.

The sites suspect a deadlock and initiate message exchange to resolve it. Whenever a higher priority message request arrives and waits at a site because the site has sent a reply message to a lower priority request.

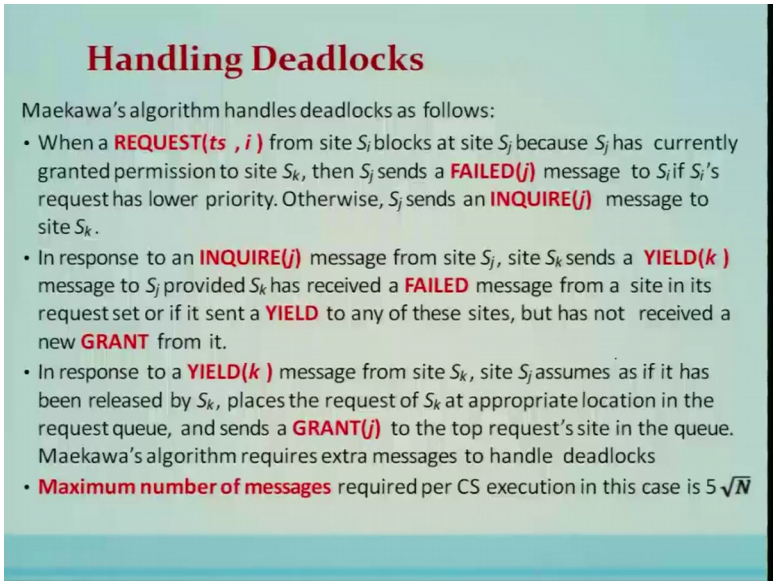
(Refer Slide Time: 15:57)



This basically is identified and used to break the deadlock now this is handled using 3 kind of messages. Failed message, failed message from site i to j indicates that Si cannot grant Sj request because it has currently granted permission to a site with a higher priority request. So, in that case this failed message will indicate that this it is not possible to grant the permission.

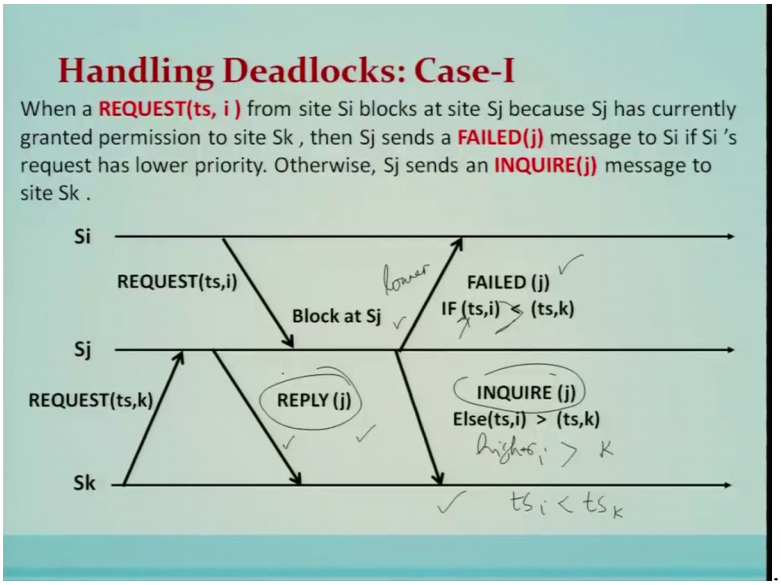
Now, another type of message for handling deadlock is called inquire. And inquire message from I to j indicates that Si would like to find out from the site Sj if it has succeeded in locking all the sites in it is request set. Third kind of message is called yield message yield message from site i to j indicates that Si is returning the permission to Sj to yield to a higher priority requested Sj.

(Refer Slide Time: 17:00)



Now handling the deadlock Maekawas algorithm and handles deadlock as follows when a request time stamp i from site i blocks at site Sj because Sj has currently granted the permission to site.

(Refer Slide Time: 17:29)

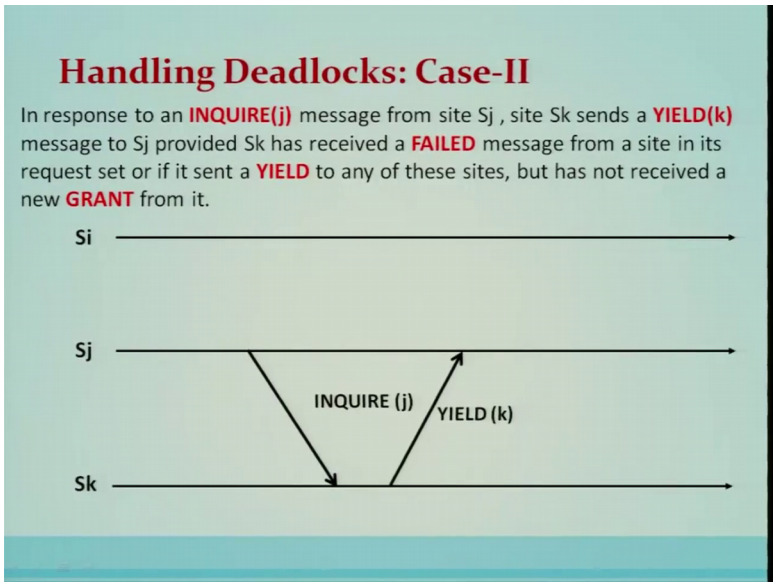


Sk then Sj sends a failed message to Si if sis requests has a lower priority otherwise Sj sends inquire message to site Si. This we can see from by illustrating diagram. So, here in this diagram we can see that when a request from i blocks at a site, Sj because Sj has currently granted permission to site Sk, then Sj send a failed message to Si if Si is request is lower priority otherwise Sj sends an inquire message to site Si.

Now, if this particular request time stamp or a time stamp based basically of request is basically higher, than time stamp of this particular request K. Then basically this is of lower priority message. So, it will basically generate a field message here in this particular case. Now if the incoming request of i is having higher priority higher priority compared to the priority of K message.

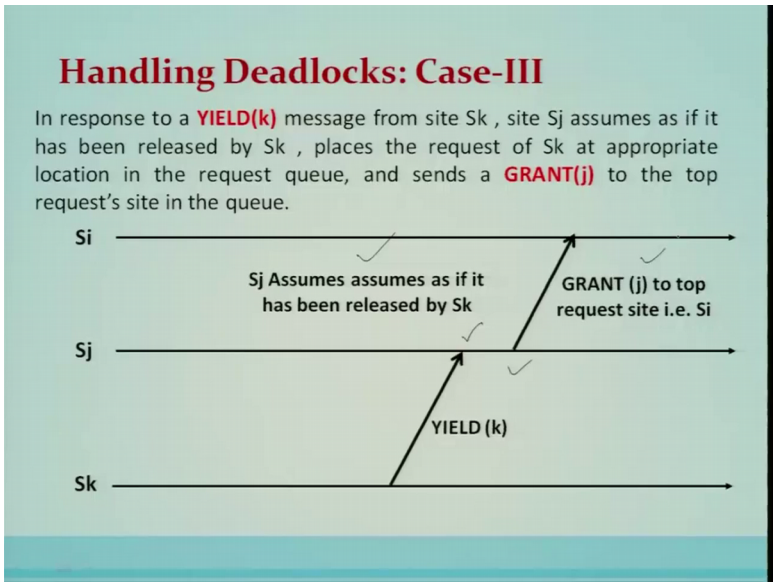
so; that means, it is time stamp of i is less than time stamp of K. Then in that case the Si will send an inquiry message to Sk to review this previously send the reply message.

(Refer Slide Time: 19:08)



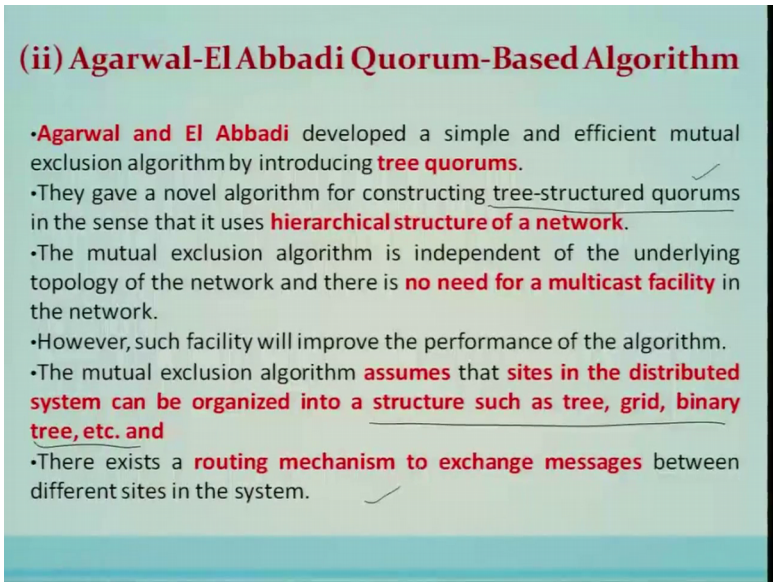
In response to the inquire message site Sj site Sj will send the yield message Sk sents the yield message to Sj. After receiving the yield message by Sj will assume that as if it has been released by Sk.

(Refer Slide Time: 19:22)



Then consequently site Sj will send a grant message to the top of the request site at Si in the queue.

(Refer Slide Time: 19:46)

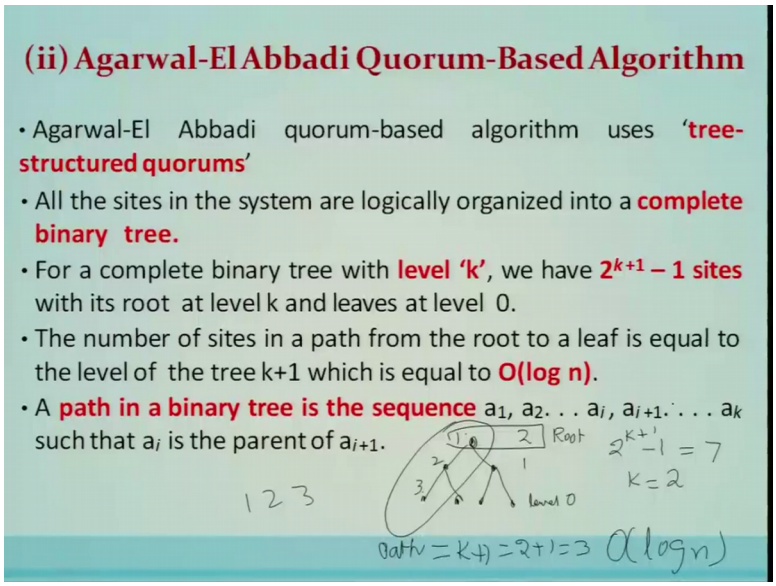


And hence the deadlock is basically resolved in using these 3 messages. Now the next algorithm which we are going to basically cover up is by Agarwal El Abbadi Quorum based algorithm. So, Agarwal and El Abbadi algorithm developed a simple and efficient mutual exclusion algorithm by introducing a tree quorums.

So, they gave a novel algorithm for constructing tree structure quorums. So, here we are going to understand what is a tree structure quorums. This concept is given here in this algorithm the mutual exclusion algorithm is independent is independent of the underlying topology of the network and there is no need for any other communication facilities like multi casting; however, such a facility will improve the performance, the mutual exclusion algorithm here given by Agarwal El Abbadi algorithm assumes that the sites in the distributed systems are organized into a structure and such as the tree structure grid or the binary tree.

Now, there exists it is also assumed that there exists a routing mechanism to exchange the messages between different sites in the system.

(Refer Slide Time: 21:13)

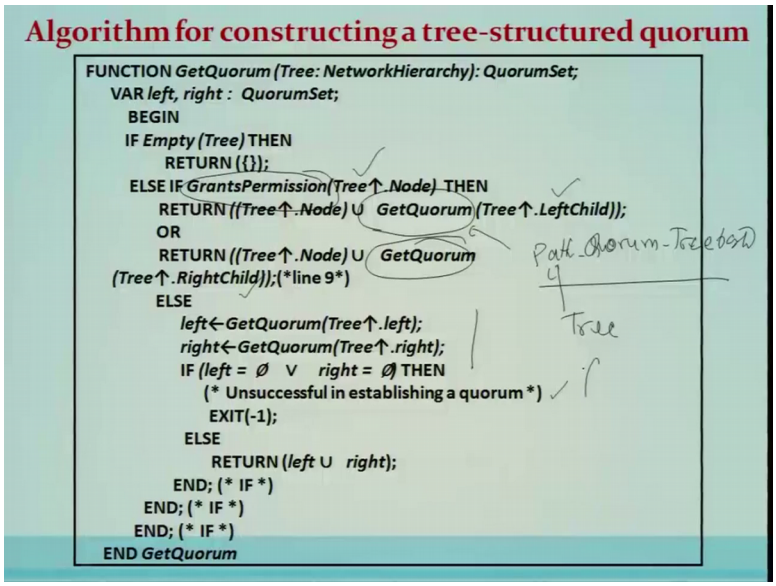


In this structured organization of the topology Agarwal El Abbadi Quorum based algorithm uses a tree structured quorums. All the sites in the system are logically organized into a complete binary tree. For a complete binary tree with the level K, we have 2 K plus 1 minus 1 sites where it is root at the top level and the leaf at the level 0. We can understand by this example. So, this is a complete binary tree now it has. So, here the leafs are at the level 0. This is level 1 and this is level 2.

Now here we have how many sites? 2 raise power K plus 1 minus 1 sites. So, here the total number of level is 2, 2 plus 1 that is 3, 3 minus 1 is there is total 7. 7 different nodes are participating in this particular case. And the root is at the level K. The number of sites in a path from root to the leaf is equal to the level of the tree K plus 1.

So, here this means that this the number of sites in a path. From root to the leaf is equal to the total level that is K plus 1 and K over here is 2 plus 1 that is 3. So, 1 2 3, 1 2 3 different sites will be there in a particular path. And that is nothing, but equal to the order log n. So, a path in a binary tree like here 1 2 3 is a sequence that is a 1 a 2 ai, ai plus 1 such that ai is a parent of ai, ai plus 1.

(Refer Slide Time: 23:33)

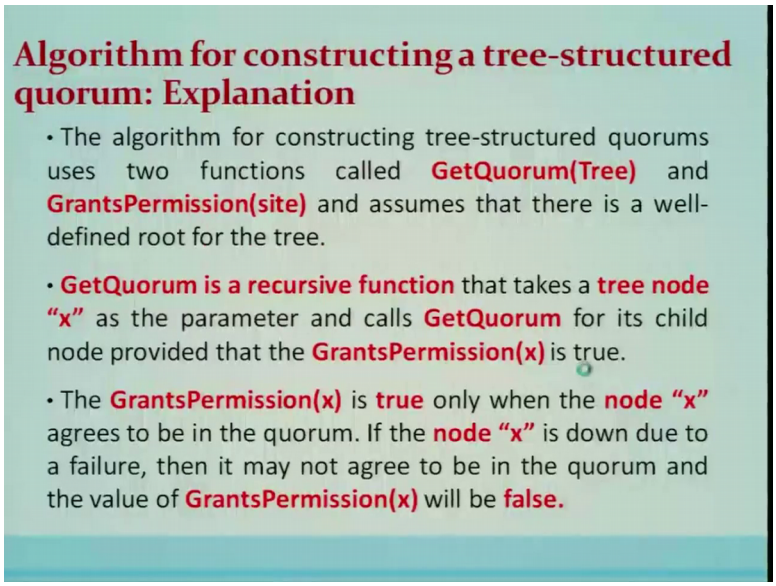


Algorithm for constructing the tree structured quorum. So, we have seen the Quorum is nothing, but a path in a tree structured quorum. So, how using algorithm this is going to be constructed in this Agarwal El Abbadi algorithm that is done, through an algorithm which is given over here. So, in this particular algorithm it uses 2 functions. The first one is called grant permission and the second function is called get Quorum function.

So, get Quorum function will give you a path and path is nothing, but the quorum, tree based quorum. And this is basically you see there is a recursive process; that means, in the tree it will start from the root it will go to the left child to it is left child and so on recursively it will try to find out the tree. Or it will go on the right side also. So, if the path is on the right side of the tree. So, it will in a recursive manner it will traverse the tree and will fetch you the path.

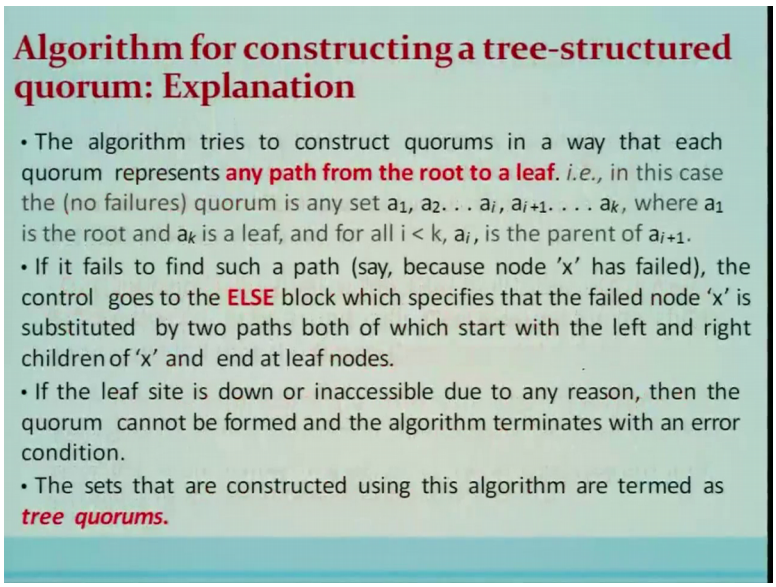
Now, the nodes when while it is traversing and collecting the nodes in a path. So, every node has to give a permission true. So, that is why grant permission is required. Now if the permission is not granted as a true; that means, there is some failure or some other region. Then that particular node is basically considered as a failed node. And then again this particular algorithm will be collecting up the path in that situation that is when some of the nodes are failed; that means, their grant permission is false. And if the leaf node is basically failed then basically it will be unsuccessful. So, that is we are going to see in this particular explanation.

(Refer Slide Time: 25:36)



So, the algorithm for constructing the tree based tree structured Quorum uses 2 functions called get Quorum tree and grant permission that is of the site. And assumes that there is a well defined route for the tree get Quorum is a recursive function. There is I told you that takes the tree node x as a parameter and calls get Quorum for it is child provided that grounds grant permission access to that is what we have just explained.

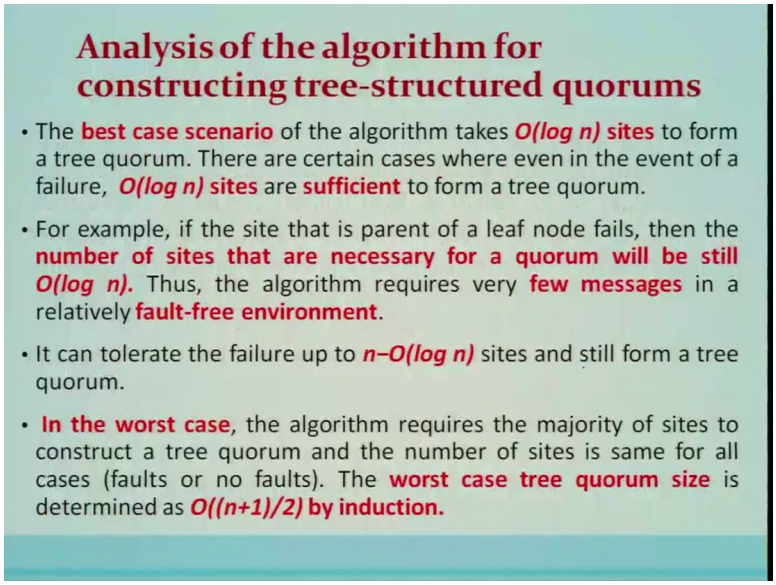
(Refer Slide Time: 26:15)



Now the algorithm tries to construct the quorums in a way that each Quorum represents any path from root to the leaf. That is in this case no failure Quorum is any set a 1 a 2

and so on up to ak, where a 1 is the root and ak is basically the leaf. And between any 2 nodes ai is the parent of ai plus 1 if it fails to find such a path the control goes to the else block of that algorithm which is specifies that the failed node x is substituted by 2 paths both of which starts at the left and right node that we will explain. The sets that are constructed using this algorithm are called tree quorums.

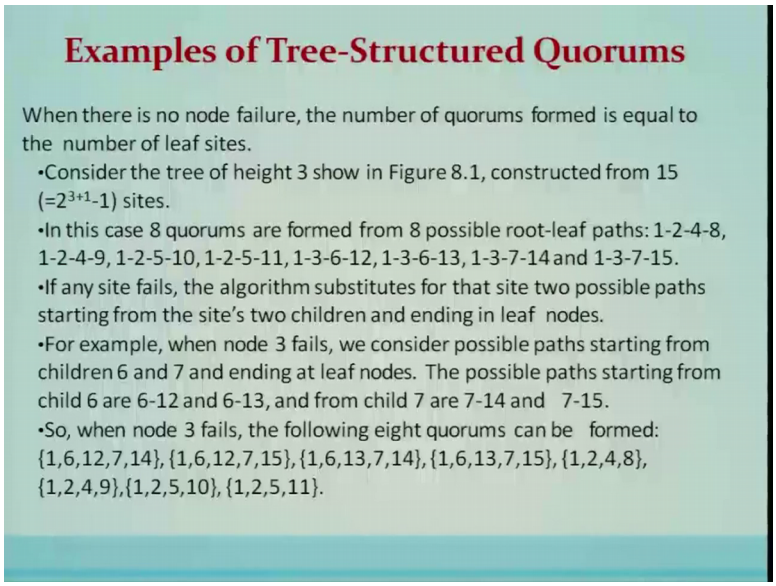
(Refer Slide Time: 27:05)



Now, the analysis of this algorithm the best case scenario the algorithm of the algorithm takes order of log n sites to form the tree quorums. So, that particular size was also there in the Maekawas algorithm that we have seen to form. So, there are certain cases we are even in the event of failures order log n sites are sufficient to form the tree quorums. Now if the site that is a parent of a leaf node fails then the number of sites that are necessary for the Quorum is still O log n thus the algorithm requires a few messages relatively fault free environment it can tolerate the failure up to up to N minus order log n sites and still form a tree.

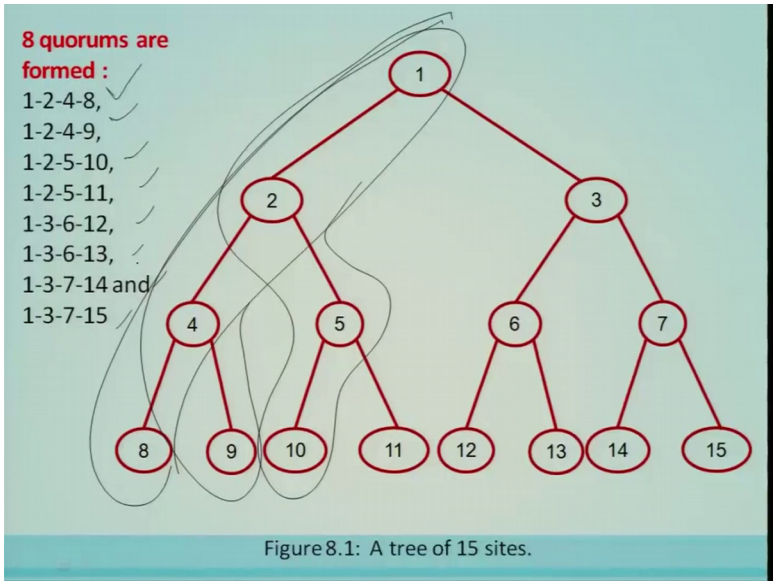
So, in the worst case the algorithm requires the majority of the sides to construct the tree quorums and the number of sites is the same for all the cases faults and no faults. The worst case tree Quorum size is determined as order of N plus 1 by 2 by induction.

(Refer Slide Time: 28:06)



Now, we are going to see the example of a tree based quorum. So, here this is a complete binary tree consisting of 15 sites.

(Refer Slide Time: 28:12)



Now, here if we see any path this path is nothing, but a tree quorum. So, the first tree Quorum will be 1 2 4 8 the second tree Quorum will be this one and third Quorum will be like this and so on. So, all these cases is listed and there are 8 different ways. We can find out a path from root to the leaf and they become a tree Quorum in this example.

(Refer Slide Time: 28:52)

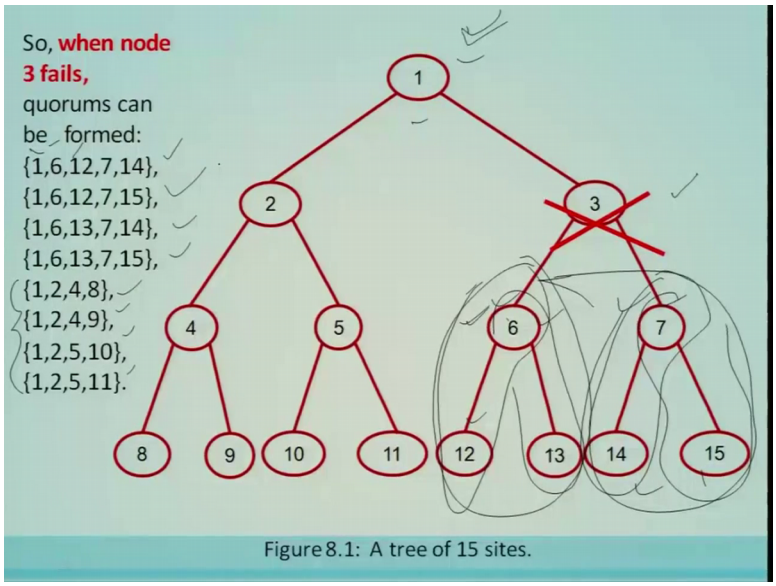


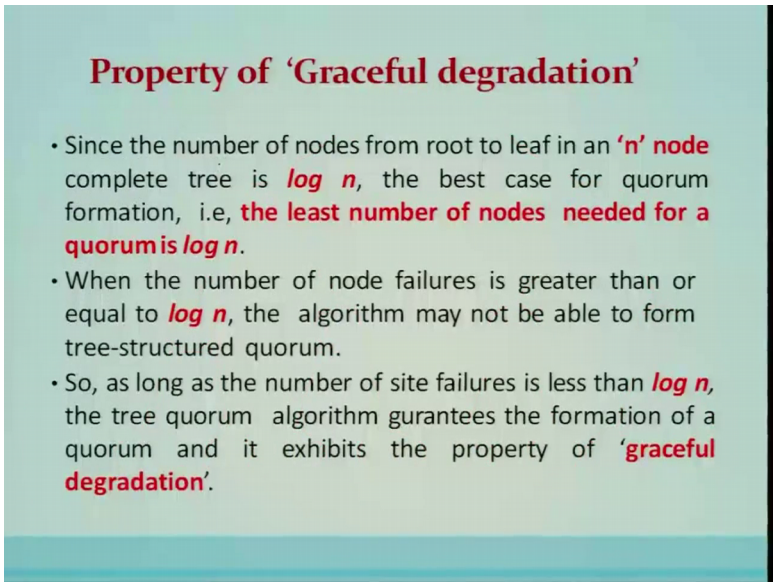
Figure 8.1: A tree of 15 sites.

Now, when a particular node is failed so; that means, failures are also introduced here in while constructing the quorums.

Now, if the node 3 is failed. So, it will be a partition into 2 different sub trees. So, these quorums where the structure is not partition or correct they are going to be the same. So, 4 different quorums are same as previously we have seen. Now these on the right side of a tree when node s is failed we are going to construct in this form. One followed by the root of the I split that is 6.

So, the root of the main tree, the root of that particular sub tree, which is partitioned and 12 1 6 12 1 6 12 then it will take the root of the other partition 7 14. So, both will be taken up then it will take 1 6 12 then 7 and 15 then 1 6 then 13 and then 7 14 1 6 13 and 7 15. So, all these cases are listed over here. So, when 3 node fails then also the quorums are being formed. 8 quorums are formed in this particular case.

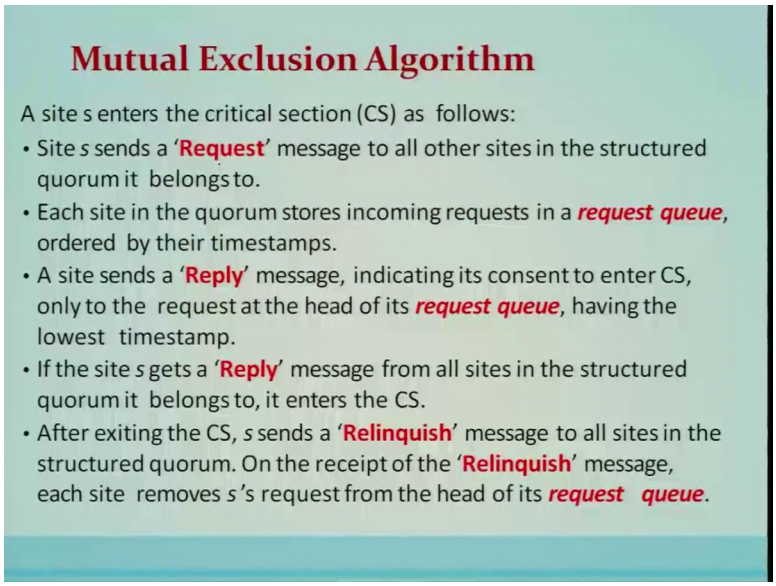
(Refer Slide Time: 30:26)



Now, property of graceful degradation; Graceful degradation in the sense means that if some nodes are failed yet you can form the quorums. Since the number of nodes from a root to the leaf in an N node complete tree is log n the best case for the Quorum formation that is the least number of nodes needed for the Quorum is log n. So, when the number of nodes failures is greater than equal to log n the algorithm may not be able to form tree structure quorum.

So, as long as the number of sites failures is less than log n the tree Quorum algorithm guarantees forming of the Quorum and it exhibits the property which is called graceful degradation mutual exclusion algorithm.

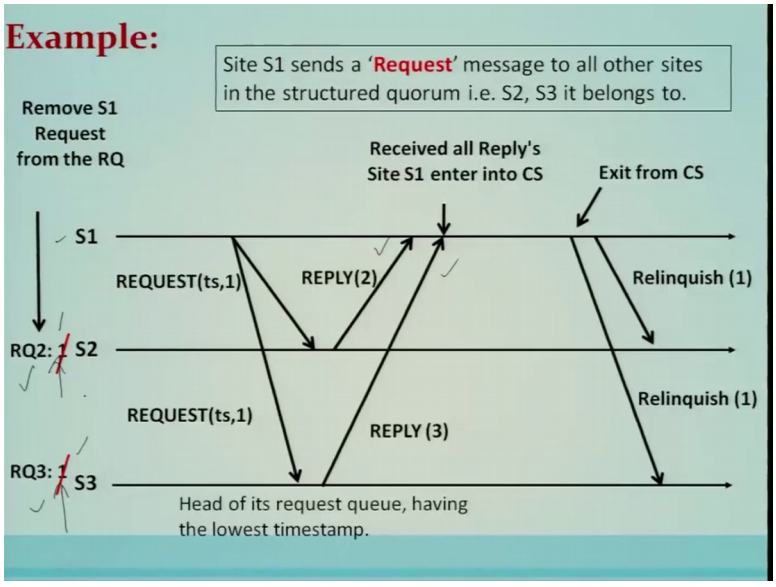
(Refer Slide Time: 31:15)



We are now going to explain the algorithm site s enters the critical section as follows. Site s sends a request message to all other sites in the structured Quorum it belongs to each site in the Quorum stores incoming requests in the request queue ordered by their time stamps a site sends a reply message indicating, it is consent to the critical section only to the request at the head of it is request queue having the lowest time stamp.

If the site s gets ready reply message from all the sites in the structure Quorum, it belongs to it enters into a critical section.

(Refer Slide Time: 32:00)

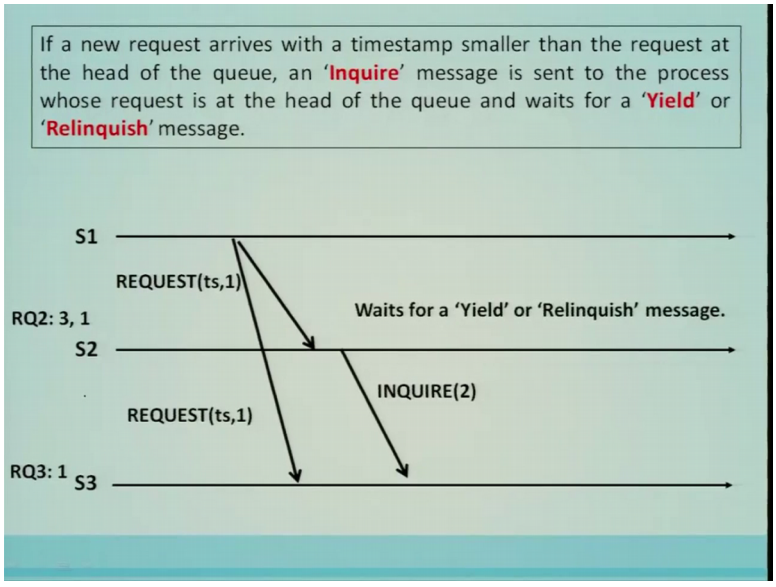


So, this we can understand using this example which is illustrated here. So, here we can see that there are 3 different sites S1 S2 and S3. We assume that S2 and S3 they are the part of the tree quorums of site S1.

Now the next step is site S1 sends a request message to the it is tree Quorum sites that is S2 and S3. Then these particular requests when it comes they will be recorded in it is basically the request queues of the site 2 and site 3. And which indicates that site 1 is basically is in the head of the queue; Head office requests queue having the lowest timestamp.

So, it will give the reply it will send the reply back of these particular requests which are at the head of the queue. Site S1 when it gets when it receives the replies from all it is sites in the tree quorum. So, now, s I S1 will enter into a critical section because it has taken up it has being granted the permission to enter into critical section, when S1 exists from a critical section it will send the relinquish message to it is tree quorums. That is to S2 and S3. Now S2 and S3 after getting the relinquish message it will remove it from it is from it is request queue.

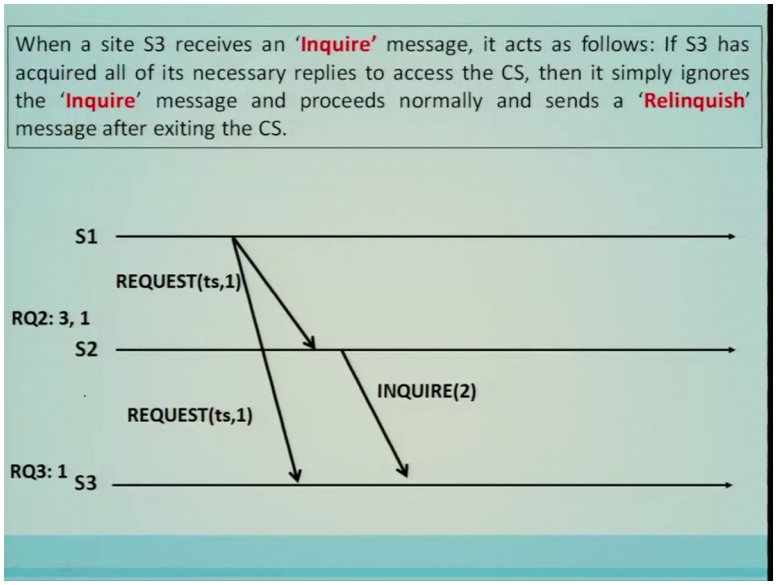
(Refer Slide Time: 33:47)



Now, if a new request arrives with a time stamp is smaller than the request at the head of the queue and inquire message will be will be sent to the process, whose request is at the head of the queue and wait for a yield or relinquish message. So, this is going to be the dealing with the deadlocks just like we have seen the Maekawas algorithm. So, inquire

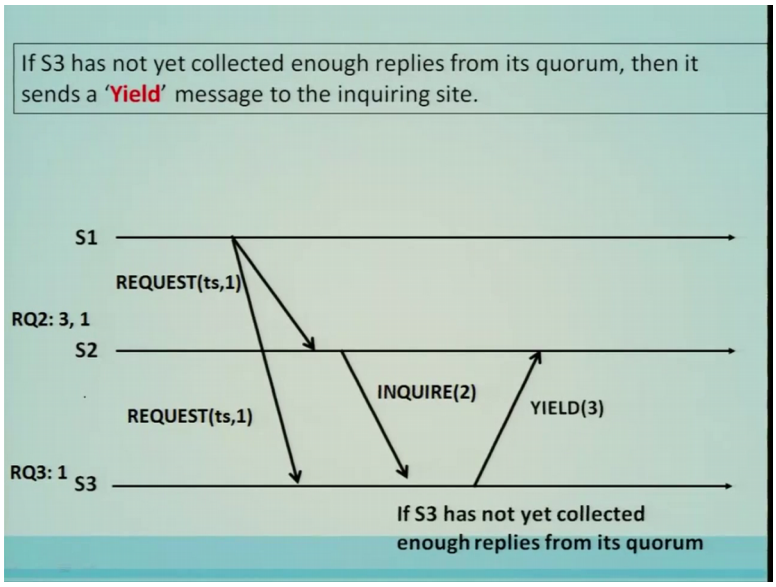
message is sent to the process, whose request is the head of the queue and waits for the yield or the relinquish message.

(Refer Slide Time: 34:22)



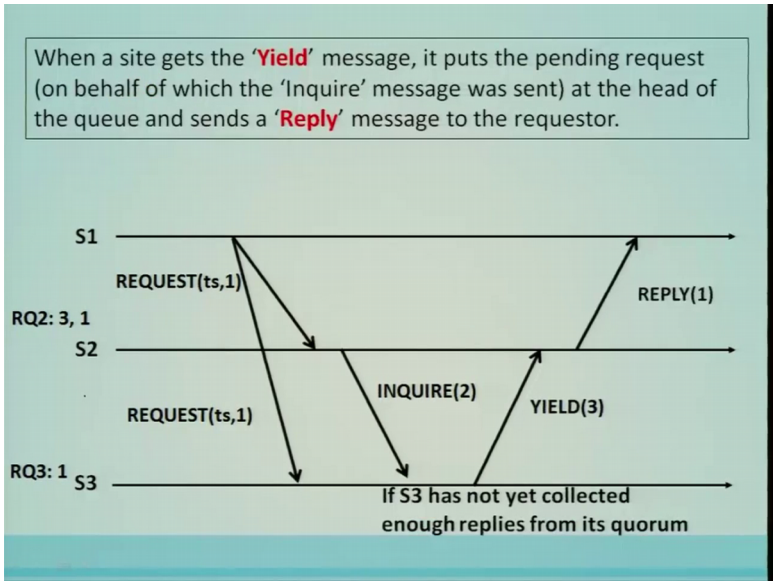
Now, when site S3 receives the inquire message it acts as follows. If S3 has acquired all of it is necessary utilize to access the critical section, then it is simply ignores the inquire message and proceeds normally and sends relinquish message after it exists the critical section.

(Refer Slide Time: 34:39)



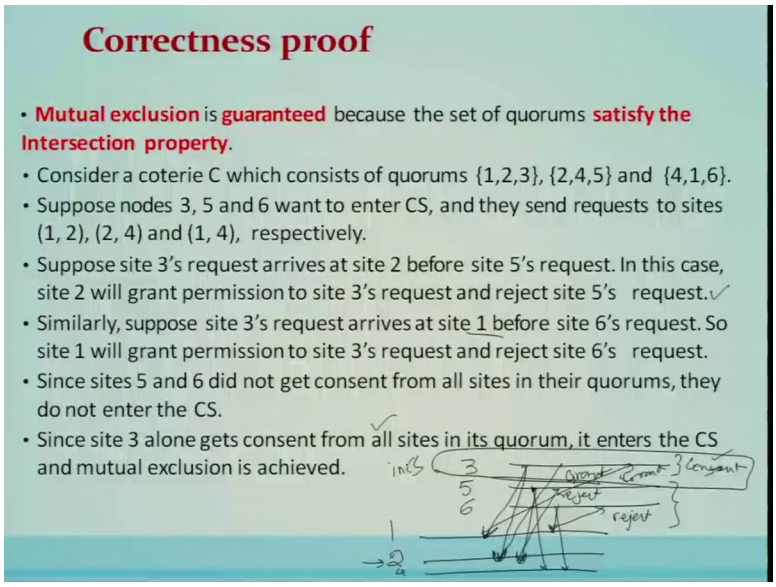
Now, if S3 has not yet collected enough replies from the from it is Quorum then it is sends a yield message to the inquiring site.

(Refer Slide Time: 34:49)



When a when a site gets a yield message it puts the pending request on behalf of which the inquire message was sent at the head of the queue like here. Head of the like head of the queue and sends the reply message to the requester this particular reply message was sent now.

(Refer Slide Time: 35:12)



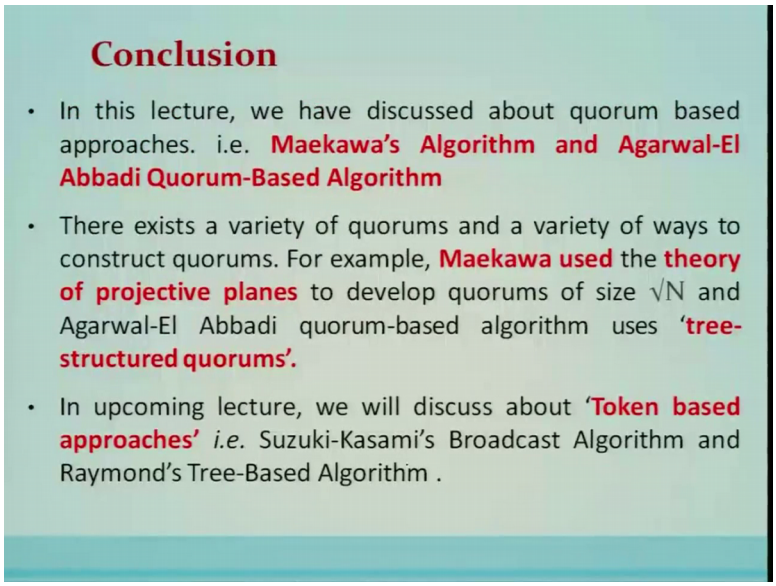
So, after getting all the replies site S1 will enter into critical section, now correctness proof of this algorithm. Mutual exclusion is guaranteed because the set of quorums satisfy the intersection property.

Consider a coterie C is consist of the quorums 1 2 3. Consider a coterie C which consists of the quorums 1 2 3 2 4 5 4 1 6. Suppose not 3 5 6 1 2 enter in critical section they send the request to the sites 1 2. They send the request to the site. So, 3 will send the request to the site 1, and it will also send the request to the site 2. And site 5 will send to 2 and 4 site 5 will send to 2 and it will send to 4. And site 6 will send to 1 and 4.

Now, suppose site threes requests arrives at site 2. So, this arrives before the request of site 5. So, site 2 will grant the permission to site s threes request. So, site 2 will grant the permission and so site 2 will grant the permission. And it will reject to the site 5 requests. Similarly, site 3's request arrives site one before site 6. Here you see that this arrives before site 6. So, it will send a grant and this will reject to the site 6.

So, here we can see that 5 and 6 did not get the consent from all the sites in the quorum. So, only site 3 get the consent only site 3 get the consent from all the quorums and others do not get the Quorum to enter. So, site 3 alone gets the Quorum from all the sites and it enters into a critical section hence the mutual exclusion is achieved.

(Refer Slide Time: 38:12)



Conclusion: In this lecture we have discussed about Quorum based approaches for solving the distributed mutual exclusion. That is by given by Maekawas algorithm and Agarwal El Abbadi Quorum based algorithms. There exists a variety of quorums and a variety of ways to construct quorums. For example, Maekawas used the theorem the theory of projective planes to develop quorums of size root N, and Agarwal El Abbadi algorithm uses the tree structured quorums. In upcoming lectures, we will discuss about token based approaches given by Suzuki Kasamis broadcast algorithm and Raymond's tree based algorithm.

Thank you.