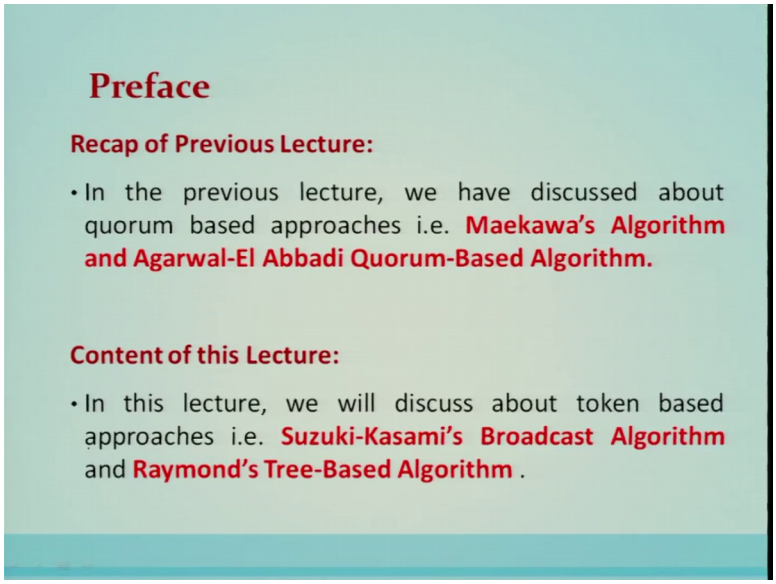
Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture - 09
Token Based Distributed Mutual Exclusion Algorithms

Lecture 9: Token Based Distributed Mutual Exclusion Algorithms.

(Refer Slide Time: 00:23)



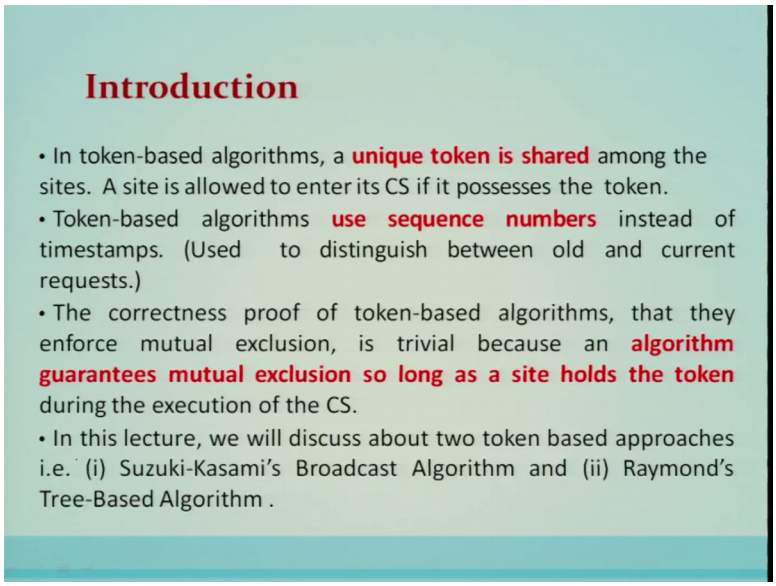
Preface recap of previous lecture, in previous lecture we have discussed about quorum based approaches that are given by Maekawas algorithm and Agarwal El Abbadi quorum based algorithms for distributed mutual exclusion. Content of this lecture: in this lecture we will discuss about the token based approaches namely Suzuki-Kasami's Broadcast Algorithm and Raymond's Tree-Based Algorithms.

(Refer Slide Time: 00:52)

So, token based algorithms uses a unique token and this unique token is shared among the sites, holding the token can allow to go into a critical section repeatedly until it sends this token to some other sites.

Numerous token based algorithms are available and they differ in the method, how the site carries out the search for the token in the network. So, token based algorithms use sequence numbers instead of timestamps, sequence number distinguishes between old and current requests. The proof of correctness in token based algorithm is trivial, but the issues which are challenging in design of token based algorithms for distributed mutual exclusions are, the freedom from starvation, freedom from deadlock and the detection of token loss and the regeneration of the last token based approaches.

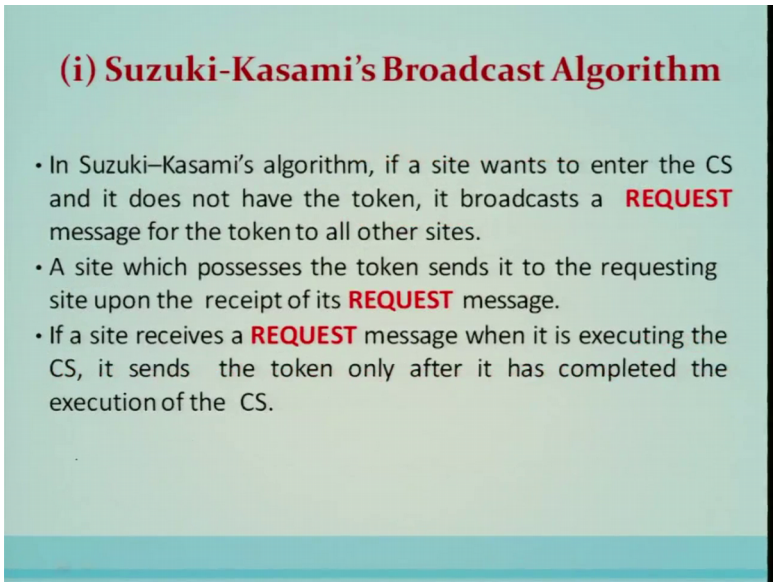
(Refer Slide Time: 02:11)



So, token based algorithms a unique token is shared among the sites and site is allowed to enter into critical section, if it possesses the token based algorithms use sequence numbers instead of timestamps. This sequence number used to distinguish between an old and the current requests for the token. The correctness proof of token based algorithm that they enforce mutual exclusion is trivial because the algorithm guarantees mutual exclusion so long as the site holds the token during the execution of the critical section.

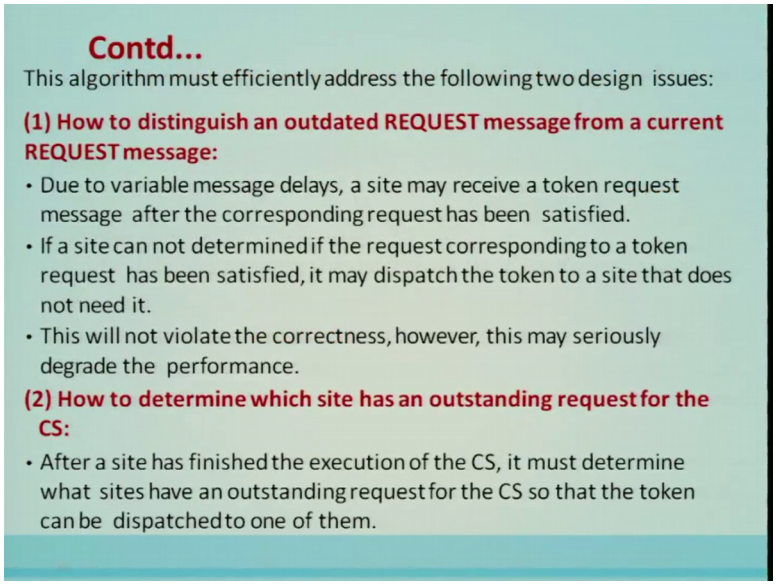
In this lecture we will discuss about two token based approaches namely Suzuki-Kazami's Broadcast Algorithm and Raymond's Tree-Based Algorithms.

(Refer Slide Time: 02:58)



Suzuki kazamis broad cast algorithm in Suzuki-Kazami's broadcast algorithm, here if a site wants to enter into a critical section it does not have the token, then it broadcasts a REQUEST message for the token to all other sites. So, a site which possesses the token sends it to the requesting sites upon the receipt of the REQUEST message. If the site receives a REQUEST message when it is executing the critical section, it sends the token only after it has completed the exhibition of the critical section.

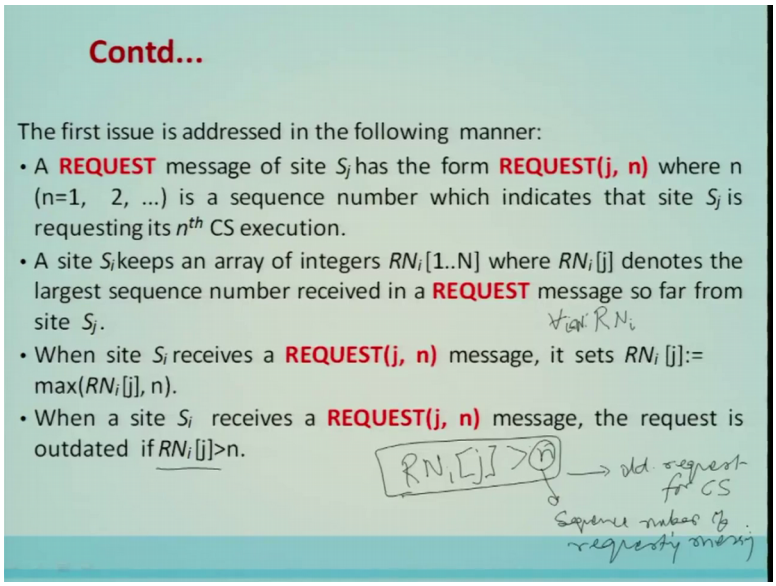
(Refer Slide Time: 03:36)



This algorithm efficiently addresses the following 2 design issues; how to distinguish an outdated requests message from a current REQUEST message, as I told you it uses the sequence numbers to basically answer this particular question.

The second issue here is how to determine which site has an outstanding REQUEST for the critical section, so after the site finished the critical the execution of a critical section, it must determine what sites have an outstanding REQUEST for the critical section.

(Refer Slide Time: 04:14)



So, that token can be dispersed to them and to do this, the algorithm uses different data structure and variables that we are going to describe now in details.

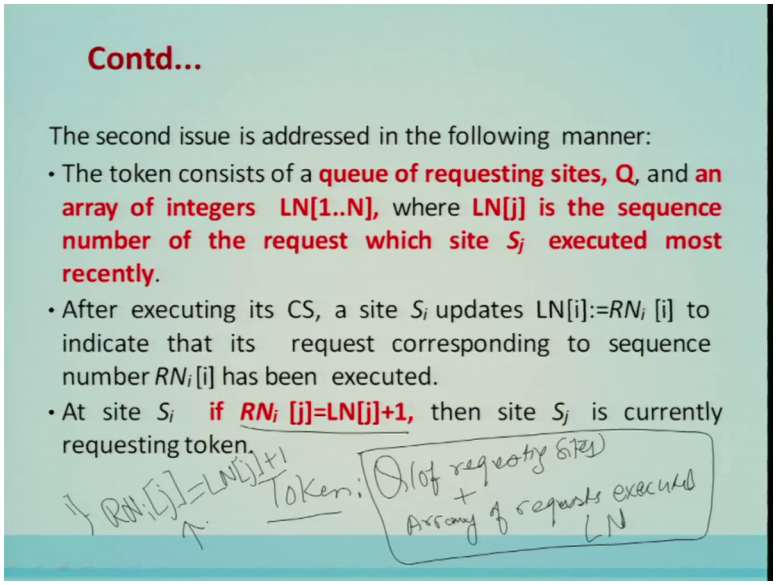
So, the first issue; that means how to distinguish an outdated REQUEST message from the current REQUEST message, is addressed in the following manner. So, a REQUEST message of a size Sj has the form REQUEST j comma n, where n is the sequence numbers which indicates that site Sj is requesting it is nth CS or a critical section execution. So, a site Si keeps an array of integer called RN i 1 2 n, where RN i of j denotes the larger sequence number received in the REQUEST message, so far from site Sj.

So, all the sites keeps this particular variable that is RN of all the sites, when a site Si receives a REQUEST j comma n from process j it sets it is REQUEST number variables RN i of j is equal to the maximum of RN i of j comma n. So, when a site Si receives the

REQUEST j comma n message, the REQUEST is outdated if r n of i of j is greater than n. So, this condition that means, if RN i of j denotes the critical section earlier REQUEST and if the current REQUEST that is sn is basically less than or a previous or it is n, then it will be designated as an old REQUEST for critical section.

So, this particular inequality will ensure that or will basically keep whether check that REQUEST is whether current one or it is an outdated and it uses the sequence number n of the requesting message, the second issue is addressed in the following manner.

(Refer Slide Time: 07:07)



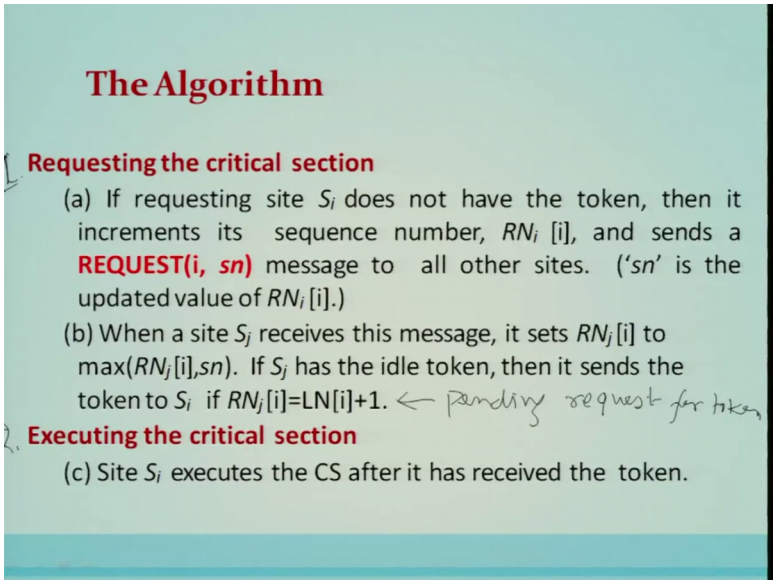
So, the second let me remind you second issue says that how to determine whether the site has an outstanding REQUEST for the critical section. So, this is called a second issue; that means, when a particular critical section is idle or a process or a site exists the critical section. Then it has to know which of this one sites are basically having the pending requests and this is called second issue and it is address in the following manner. So, the token consists of Q of requesting sites that is Q and an array of integers LN 1 to n, where LN j is the sequence number of the REQUEST which site as a has executed most recently.

So, after executing it is critical section a site Si updates LN I as RN i of i to indicate that, it is REQUEST corresponding to the sequence number RN of I has been executed. Now at a site Si if RN i of j is equal to LN j plus 1, this inequality then site Sj is currently requesting the token. So, to summarize these 2 points so first thing is the token, is

nothing but it consists of a Q of requesting sites plus it also contains an array of an array of integers that is called LNi is the sequence numbers of the requests with site as they executed most recently.

So, array of requests executed that is indicated by LN, so, these 2 information is basically makes this particular token. So, when if RN that is the REQUEST coming from j if it is equal to LN of j plus 1, then it has a pending than the site is currently requesting for a token. So, this will be the indication that a site that these are the set of sites, which are basically having a pending REQUEST for the token.

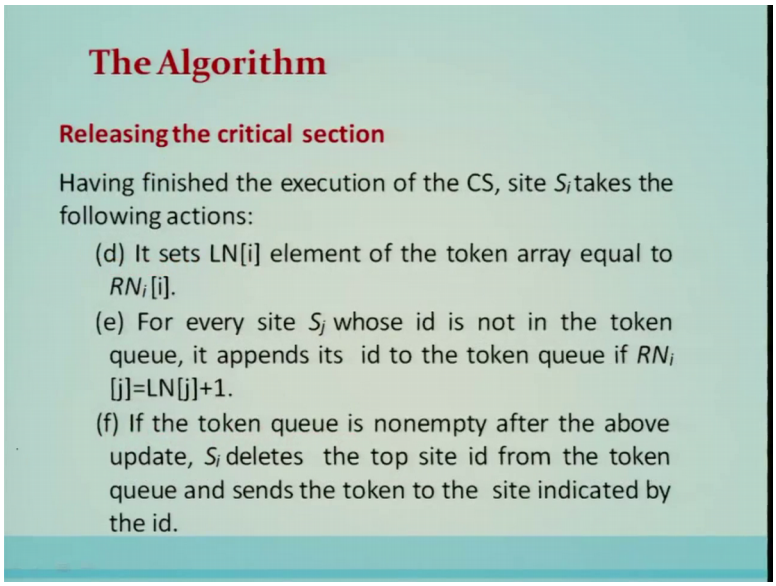
(Refer Slide Time: 10:08)



So, this comes there comes the algorithm has gives different steps, the first is step of this algorithm is about requesting for a critical section. So, requesting the critical section first step says that if requesting site Si does not have the token, then it increments it is sequence number that is RN i of i and sends the REQUEST i comma sn, there is a sequence number message to all of the sites. So, sn is updated value of RN i of i, now when a site Sj it receives this particular message it sets RN of js, i to the maximum of RN j of i and sn now if sn has the idle token, then it sends the token to Si if RN of j of i is equal to LN of i plus 1, that is it is having the pending REQUEST for the token.

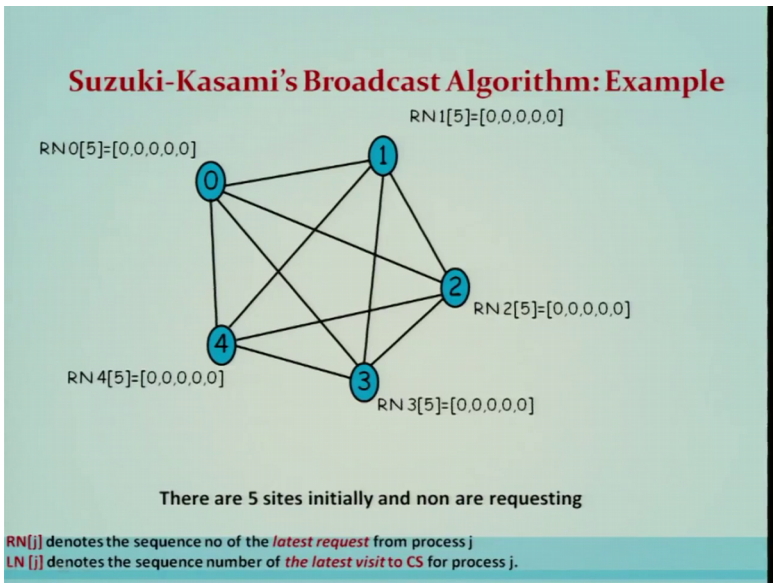
Now, the next step is of the algorithm is executing the critical section, Si execute the critical section after it has received.

(Refer Slide Time: 11:43)



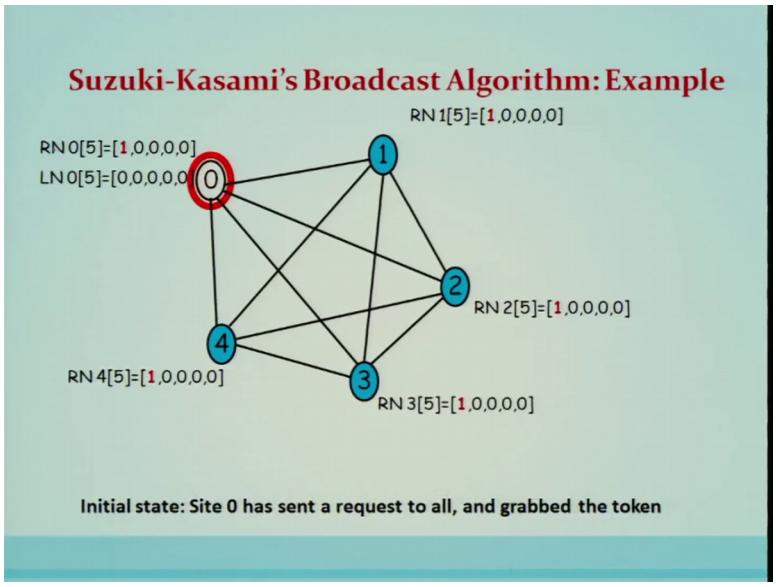
The token third step is about releasing the critical section, having finished the execution of critical section the sight Si takes the following actions. It sets LNi element of the token array equal to RNi. For every side Sj whose id is not in the token queue, it appends it is id in the token queue if RN i of j is equal to LN of j plus 1. If the token queue is nonempty after the above update, Si deletes the top side id from the token queue sends the token to the site indicated by the id. So, this we can see through this illustrative example.

(Refer Slide Time: 12:30)



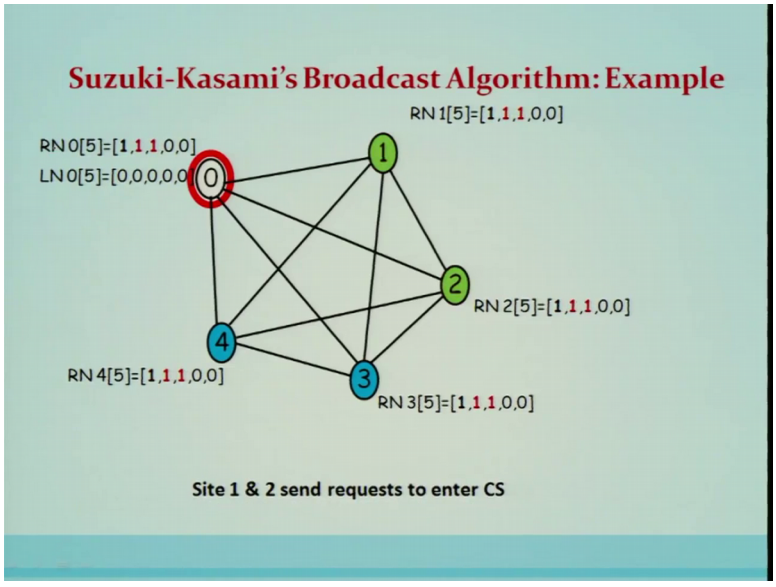
So, initially there are 5 sites and they are now requesting also RN of j denotes the sequence number of the latest REQUEST from a process j and LN j denotes the sequence number of the latest visit to the critical section for a process j.

(Refer Slide Time: 12:57)



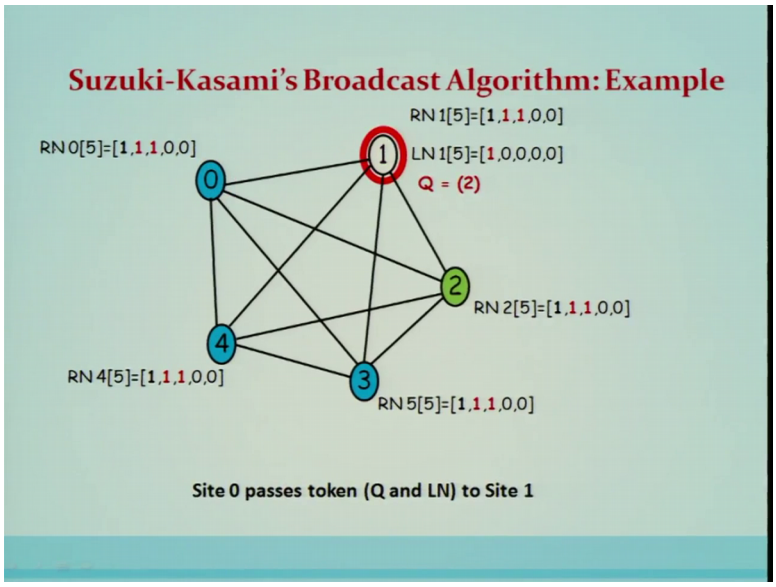
So, just see that so site 1 site over here in site 0, here is the requesting for the critical section entry. So, it is modifying when the message is flowing to basically 1 and 2.

(Refer Slide Time: 13:13)

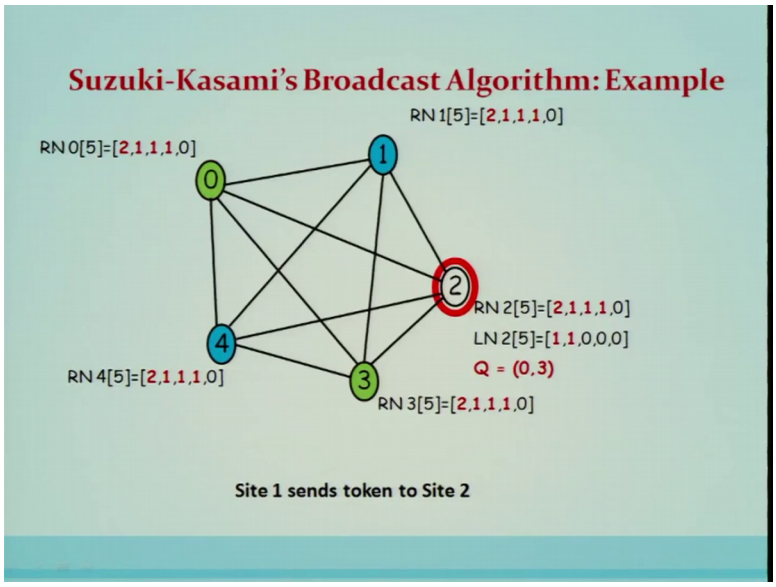


So, 1 and 2 has received this particular REQUEST and the Q will be updated as 1 and 2 they are having the pending request. So, in this way the algorithm proceeds and the token is finally being delivered.

(Refer Slide Time: 13:32)

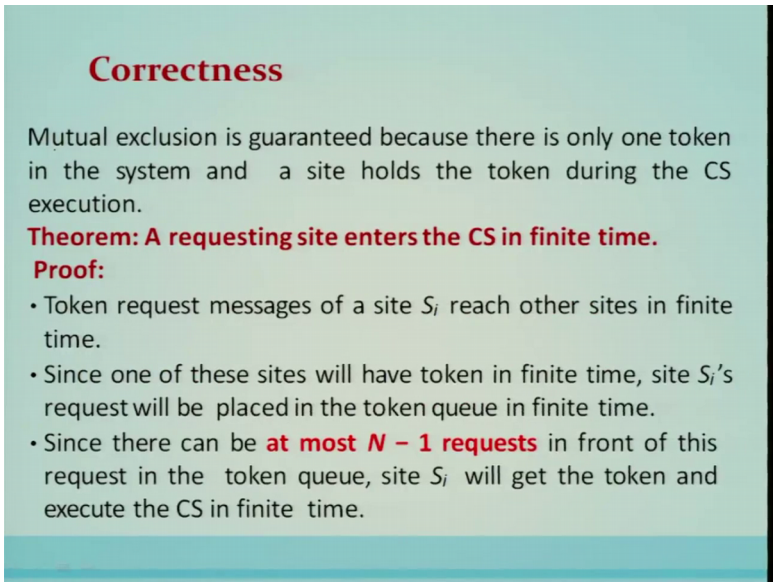


(Refer Slide Time: 13:33)



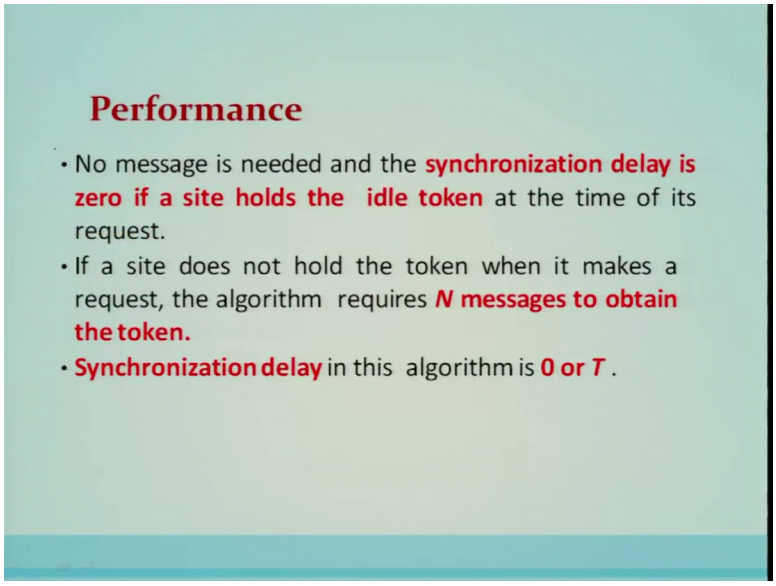
Now, the correctness of this algorithm mutual exclusion is guaranteed because there is only 1 token in the system and the site holds the token during the critical section execution.

(Refer Slide Time: 13:35)



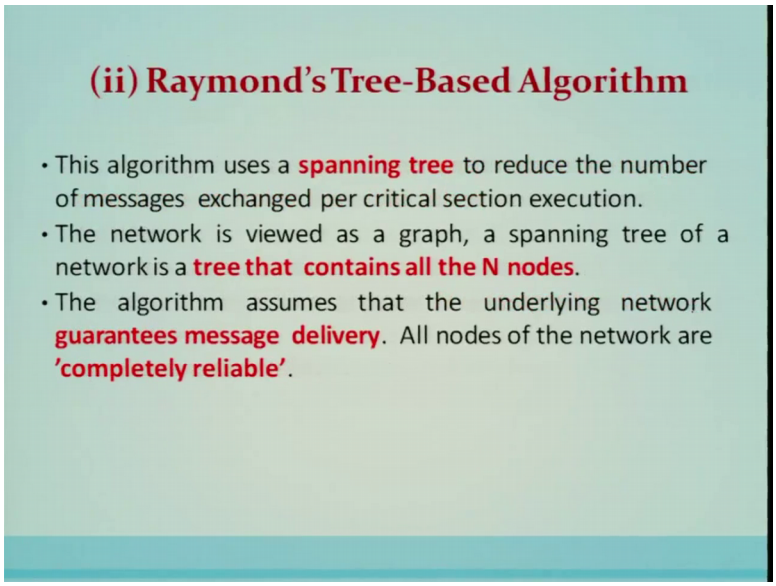
Theorem a requesting site enters the critical section in a finite time to the REQUEST, the token REQUEST message of a site Si reach the other site in a finite amount of time. Since one of these sites will have token in a finite time, site Si is the REQUEST will be placed in the token queue in a finite amount of time. Since there can be at most N minus 1 requests in front of this REQUEST in the token queue. So, site as I will get the token eventually and execute the critical section in a finite amount of time.

(Refer Slide Time: 14:28)



Performance, no messages is no message is needed and the synchronization delay is 0 if the site holds the idle token at that time at the time of it is request. So, if a site does not hold the token when it makes a REQUEST, the algorithm requires N messages to obtain the token. Hence the synchronization delay is idle in the first case or the synchronization delay is at most T, where n numbers of messages are required Raymond's tree based algorithm.

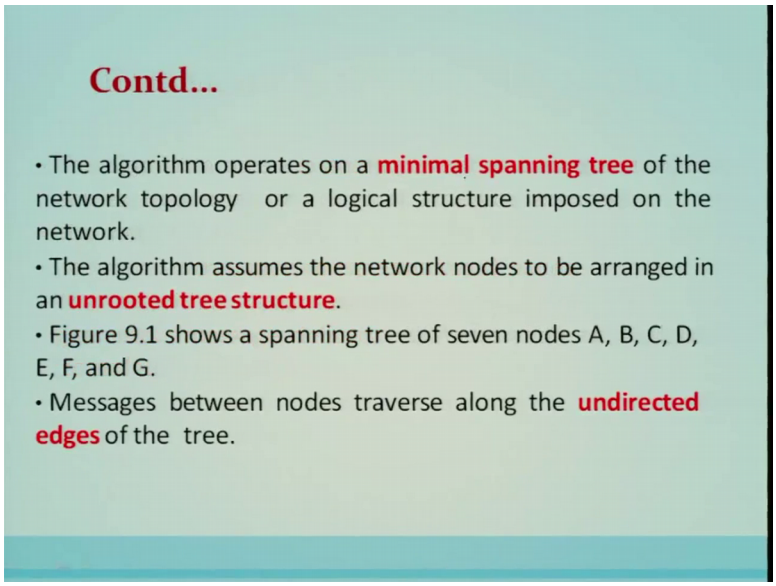
(Refer Slide Time: 15:03)



This algorithm uses a is spanning tree of the network to reduce the number of messages exchanged per critical section execution, hence this is an optimized algorithm as far as number of messages are concerned, but it uses a structure that is basically the spanning tree.

So, it assumes that the spanning tree of the network is available, now the network is viewed as a graph that is a spanning tree of the network is a tree that contains all n nodes. So, the algorithm assumes that the underlying network guarantees the message delivery, so that means, it is a reliable communication channel it assumes. So, all the nodes of the network are completely reliable. So, there is no assumption of the failures, it is assumed that the nodes are reliable the algorithm operates on a minimum spanning tree.

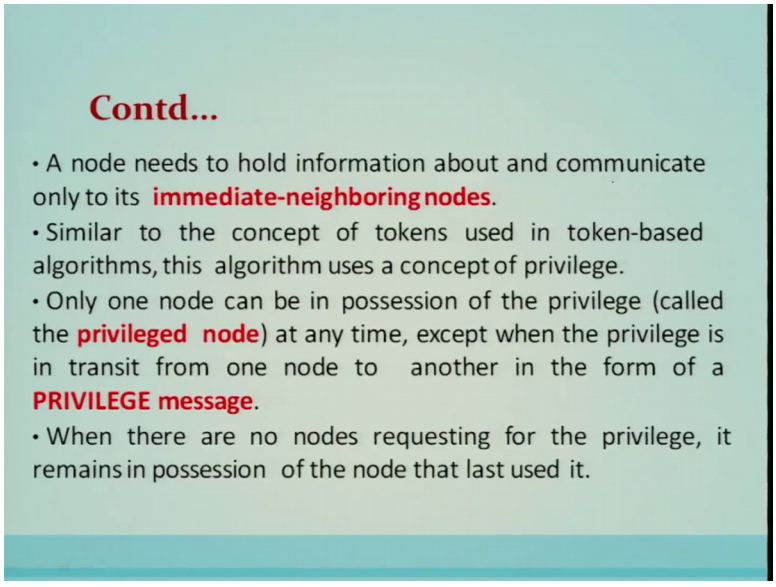
(Refer Slide Time: 15:56)



So, by spanning tree is basically the minimum spanning tree which is required in the algorithm of the topology.

The algorithm assumes the network node to be arranged in an uprooted tree structure, the next figure shows the spanning tree of 7 nodes and the message between the node traverse along the undirected edges of the tree.

(Refer Slide Time: 16:24)

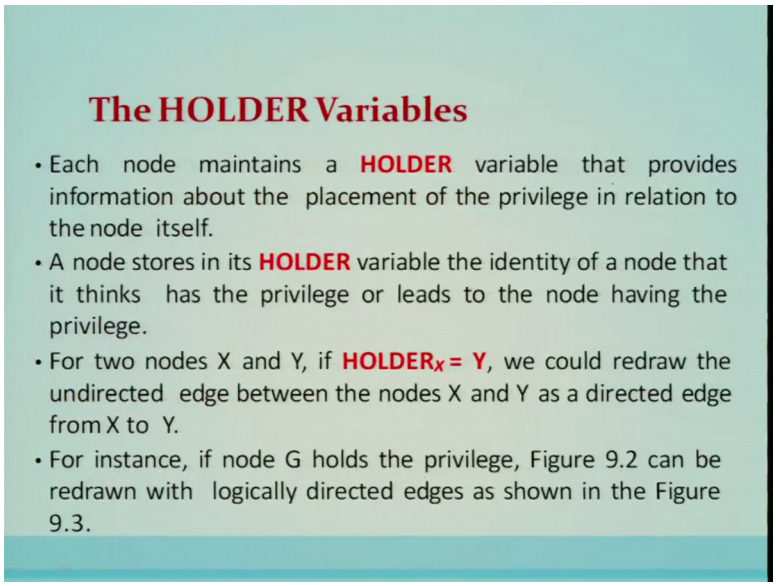


A node needs to hold the information about and communicate only to it is immediate neighbors, similar to the concept of token used in token based algorithm this algorithm

uses the concept of privilege. Only one node can be in a position of the privilege called the privileged node at any time, except when the privilege is in transit from one node to another node in the form of the privileged message.

So, any point of time at most one node holds the privileged and called the privileged node. So, when there are no nodes requesting for the privilege, it remains in the possession of the node that last used it.

(Refer Slide Time: 17:07)



Now, the variables which are used here in remains algorithm are as follows. So, the first variable is called the holder variable, now holder variable will impose a directed tree or an undirected graph or undirected tree which we have seen. So, each node maintains a holder variable that provides the information about the placement of privilege in relation to the node itself. A node is stores in it is holder variable the identity of the node that it thinks has the privilege or leads to the node having the privilege.

So, for 2 nodes x and y, if holder of x is equal to y, then we could redraw an undirected edge between x and y as the directed edge. So, holder variable will impose or will basically impose the directed structure on top of that. So, for instance if a node G holds the privilege, in the next figure we will show and the figure can be redrawn with logically directed edges as shown in figure 9.3.

(Refer Slide Time: 18:19)

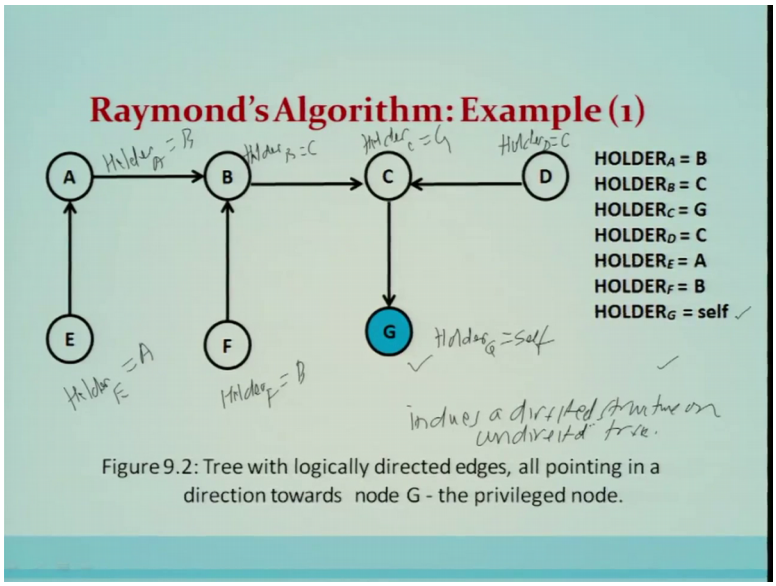
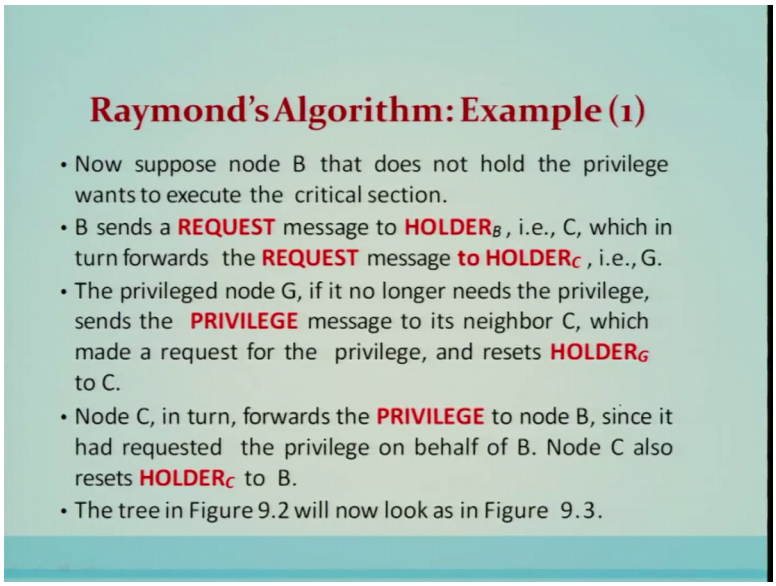


Figure 9.2: Tree with logically directed edges, all pointing in a direction towards node G - the privileged node.

So, here we can see that the previous figure which shows you the undirected graph. So, using holder variable this will basically induce directed structure of a tree and this basically directions are pointing towards the node which is currently holding the token. So, if C is a node which is the neighbor of a node which is holding the privilege token will have an edge and D can direct it is edge towards the node who in turn knows the privileged node. So, that way all the nodes will basically place as far as G is concerned which is holding the token or the privileged node will basically put self in holder of G.

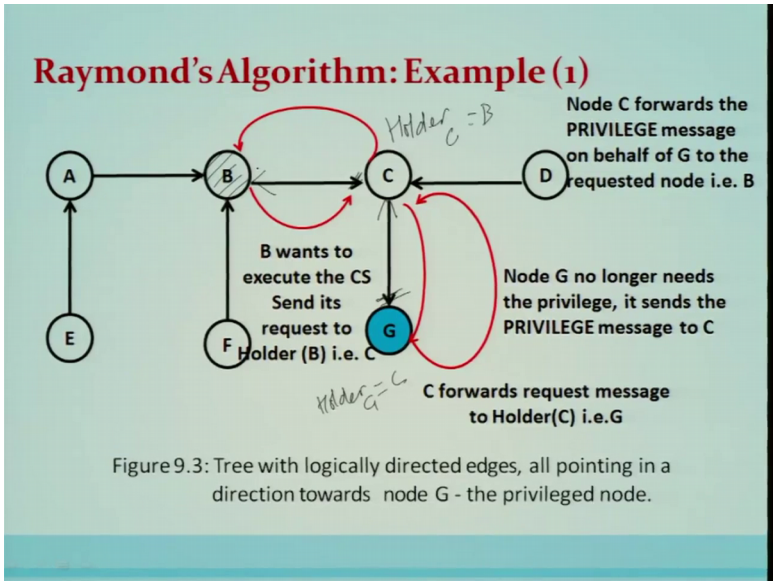
So, holder of G is nothing, but the self and the holder of D is nothing but C, holder of C is nothing but G, holder of B is nothing but C, holder of F is nothing but B, holder of E is nothing but A and holder of a is nothing but B. So, these particular holder variables will induce a directed structure on an undirected spanning tree.

(Refer Slide Time: 20:19)



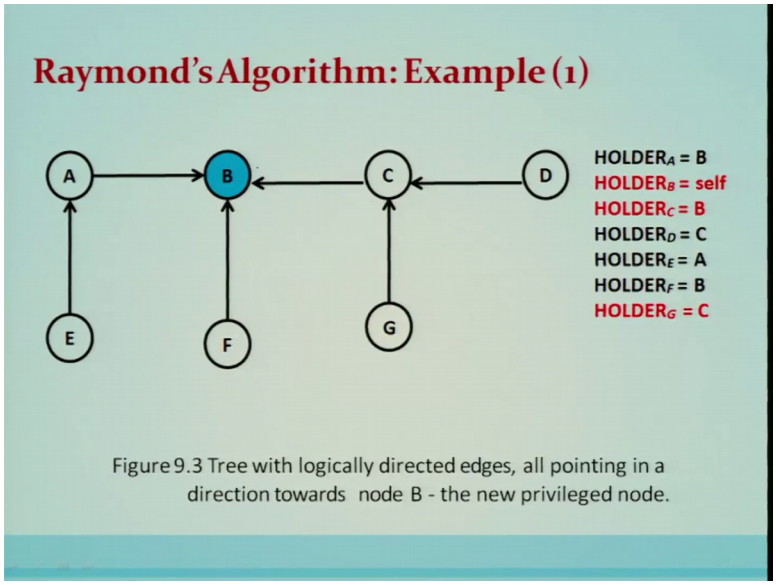
Now, suppose node B that does not hold the privilege wants to execute the critical section. So, B sends the REQUEST message to the holder of B that is C, which in turn forwards the REQUEST message to the holder C that is G. The privileged no G is no longer needs the privilege, sends the privileged message to it is neighbor C, which made a REQUEST for the privilege, on behalf of B a resets the holder variable. Holder of G is equal to C. Node C in turn forwards the privilege to node B, since it has requested the privilege on behalf of B. Node C also resets. So, basically the tree will look like in this particular manner.

(Refer Slide Time: 21:01)



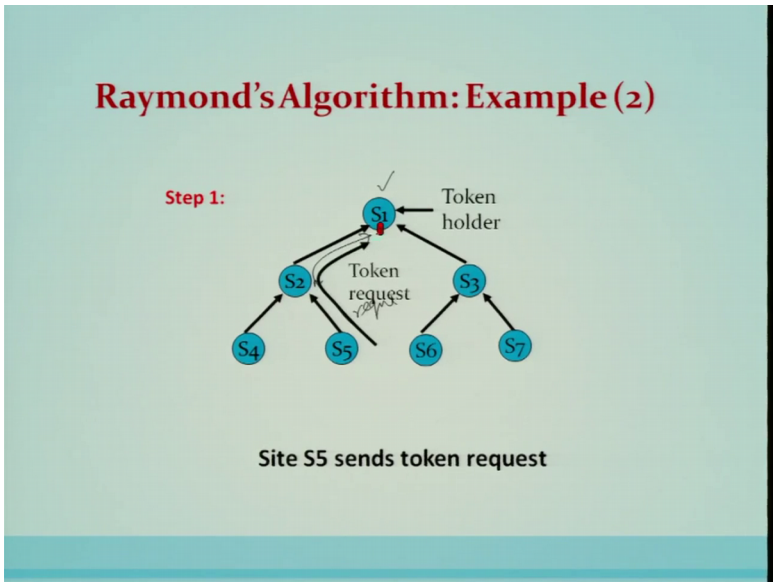
Now, here the node C forwards the privilege message on behalf of G to the requested node that is B and node G no longer needs the privilege. So, it sends the privilege to C and will change it is holder variable, holder of G will become now C and holder of C will become as B. So, these arrows will be changed in this manner and this will now contain the privileged node.

(Refer Slide Time: 21:58)



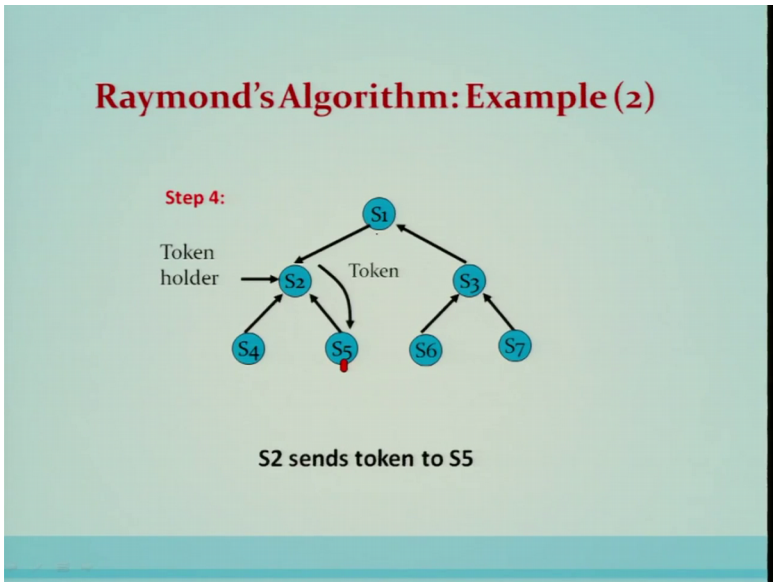
So, I have explained and the same thing is basically illustrated in this particular example.

(Refer Slide Time: 22:03)



Now, another example shows that the node which is currently holding the token is basically the token holder that is the route that is S 1, now side S 5 sends a token request. So, it will send first to S 2 and S 2 on behalf of S 5 will send a token REQUEST to S 1.

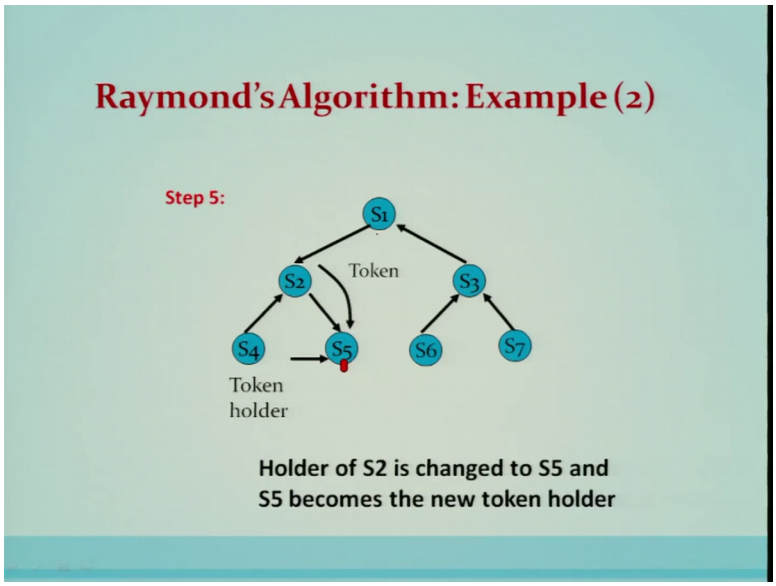
(Refer Slide Time: 22:43)



So, S 1 if it is not using the privilege at that point of time, then it will and basically that REQUEST from S 2 is at the top of it is REQUEST queue, then it will send the token to S 2.So, holder of S 1 is also need to be changed to S 2.

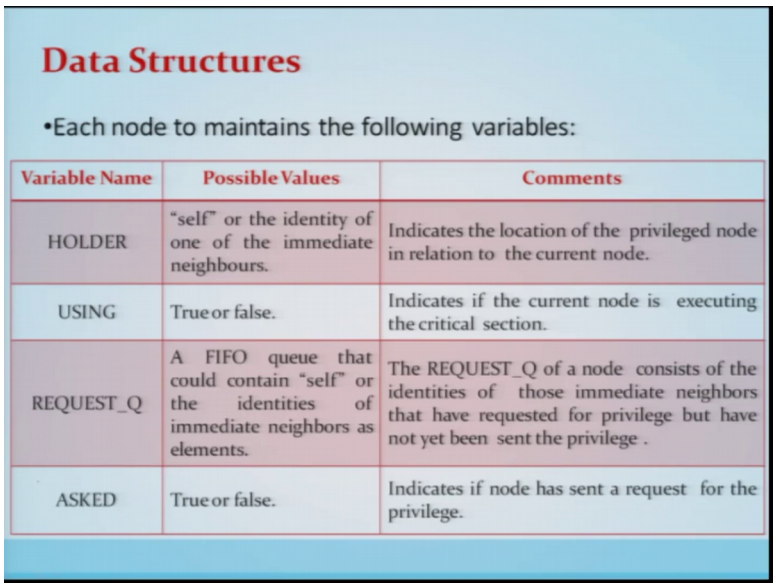
Now, S 2 will send the token to S 5 and the holder of S 2 is also changed to S 5 and S 5 becomes the new token holder.

(Refer Slide Time: 23:14)



So, the data structures used in Raymond's algorithms, at each node the following variables are used; which are summarized as follows.

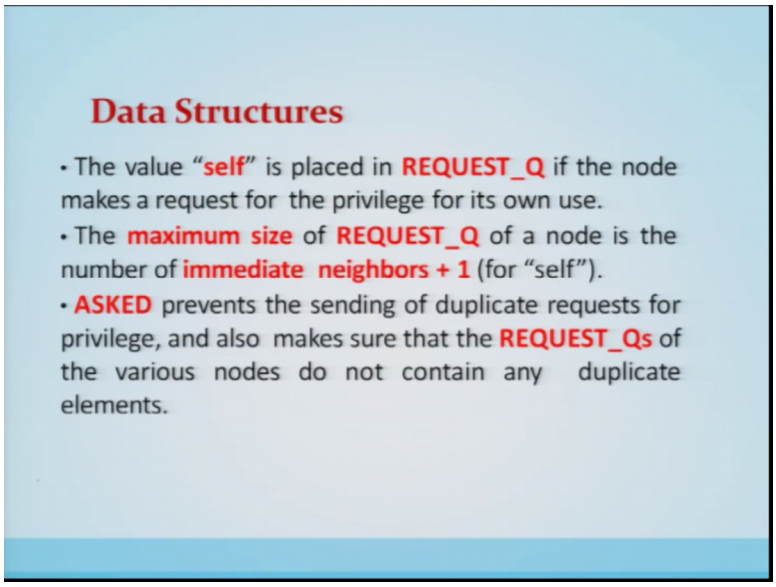
(Refer Slide Time: 23:36)



Variable name Holder will contain either the self, if the token or the privileges with the node itself or the identity of one of the immediate neighbors using variable is true or false. So, using variable indicates if the current node is executing the critical section. So, if the node currently executing into the critical section, it is using variable will be flagged as true. REQUEST Q is a FIFO queue that contains self or the identities of the immediate

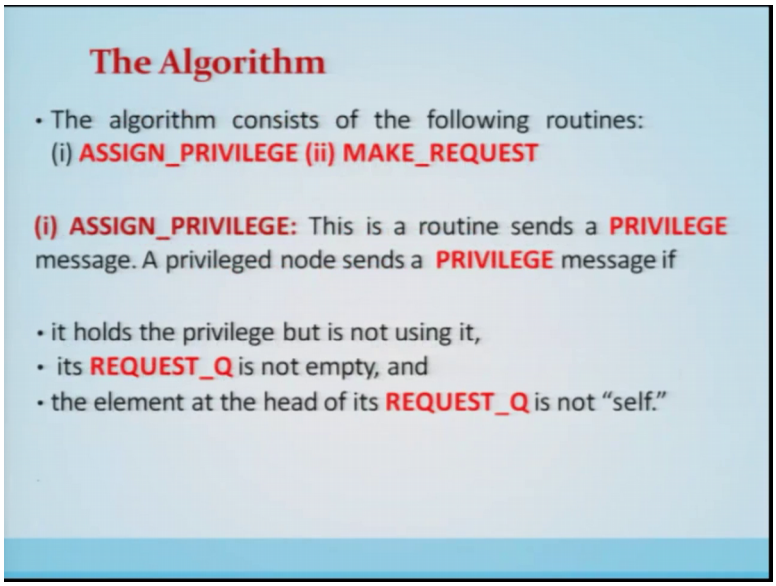
neighbors as the elements. REQUEST Q of a node consists of the identities of those immediate neighbors that have requested for privilege, but have not been sent the privilege. ASKED is true or false, indicates if the node and sent a REQUEST for the privilege.

(Refer Slide Time: 24:32)



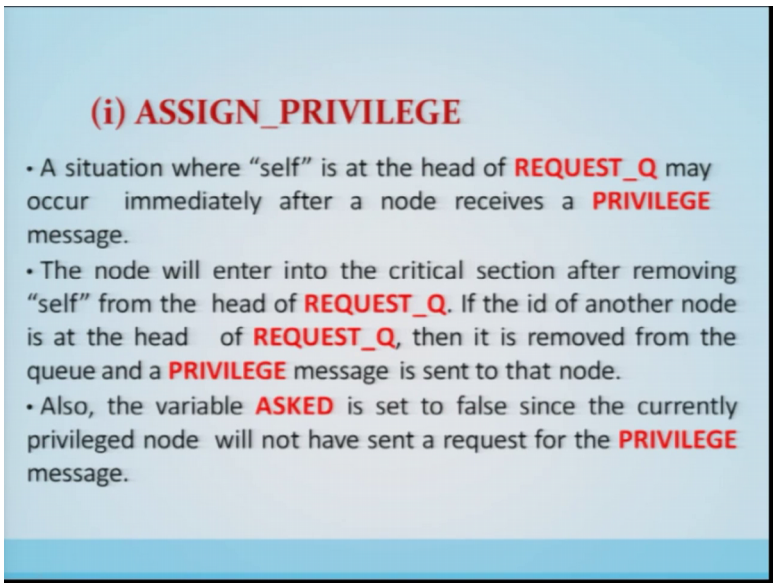
So, the data structures the value self is placed in the REQUEST Q, if the node makes a REQUEST for the privilege for it is own use. The maximum size of the REQUEST Q of a node is the number of immediate neighbors plus 1 for self. ASKED prevents the sending of duplicate requests for the privilege and also makes sure that the REQUEST Qs of the various nodes do not contain any duplicate elements.

(Refer Slide Time: 25:04)



So the Algorithm, the algorithm consists of the following routines. So, first routine is assign privilege, make REQUEST. Assign privilege this routine sends a privilege message, a privilege node sends a privilege message; if it holds the privilege but is not using it, it is REQUEST Q is not empty and the elements at the head of it is REQUEST Q is not the self.

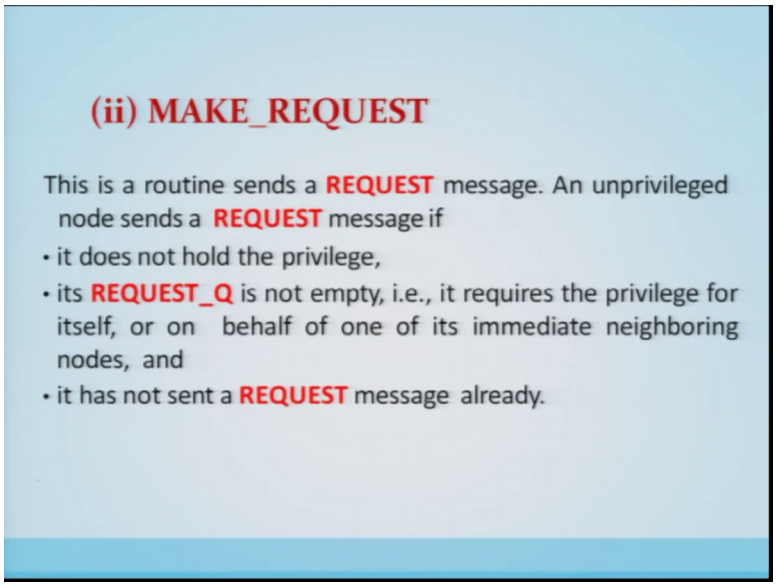
(Refer Slide Time: 25:35)



Assign privilege the situation where self is at the head of the REQUEST Q may occur immediately after the node receives a privilege message. So, the node will enter into the

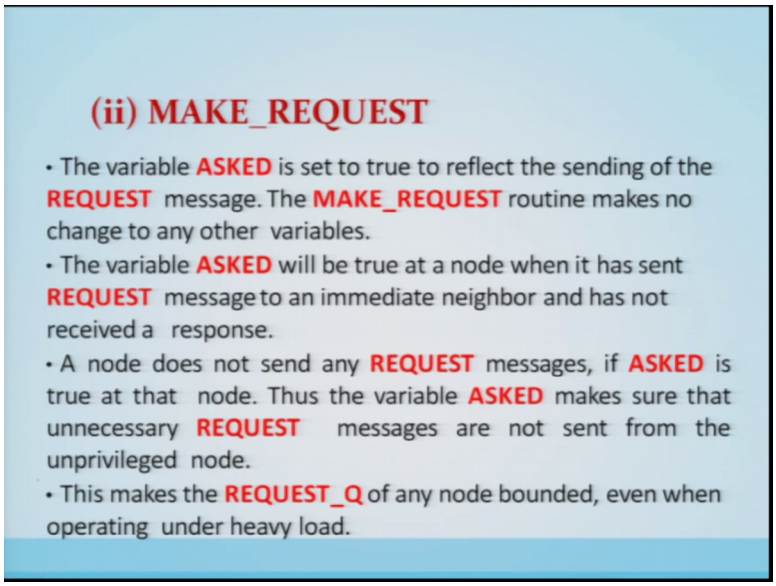
critical section after removing self from the head of the REQUEST Q. If the id of another node is at the head of REQUEST Q, then it is removed from the queue and a privilege message is sent to that node. Also the variable ASKED is set to false, since the currently privileged node will not have sent a REQUEST for the privilege message.

(Refer Slide Time: 26:15)



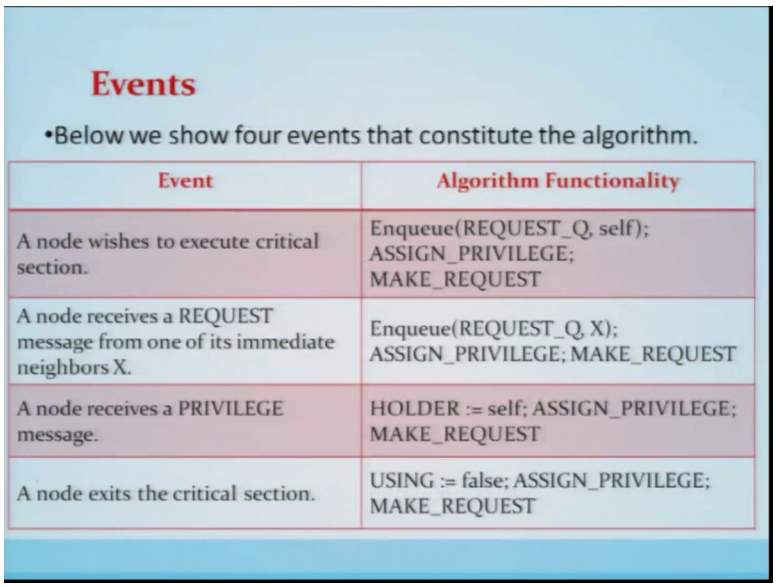
Make REQUEST this is a routine which sends the REQUEST message. An unprivileged node sends a REQUEST message if; it does not hold the privilege, it is REQUEST Q is not empty, that is required the privilege for itself or on behalf of one of it is immediate neighbors nodes and it has not send the REQUEST message already. Make REQUEST the variable ASKED is set to true to reflect the sending of the REQUEST message, the make requests routine makes no change to any other variable.

(Refer Slide Time: 26:38)



The variable ASKED will be true at the node when it has sent the REQUEST message to an immediate neighbor and has not received the response. A node does not send any REQUEST message, if ASKED is true at that node. Thus the variable ASKED makes sure that unnecessary REQUEST message are not sent from the unprivileged node. This makes a REQUEST Q of any node bounded, even when operating under heavy load.

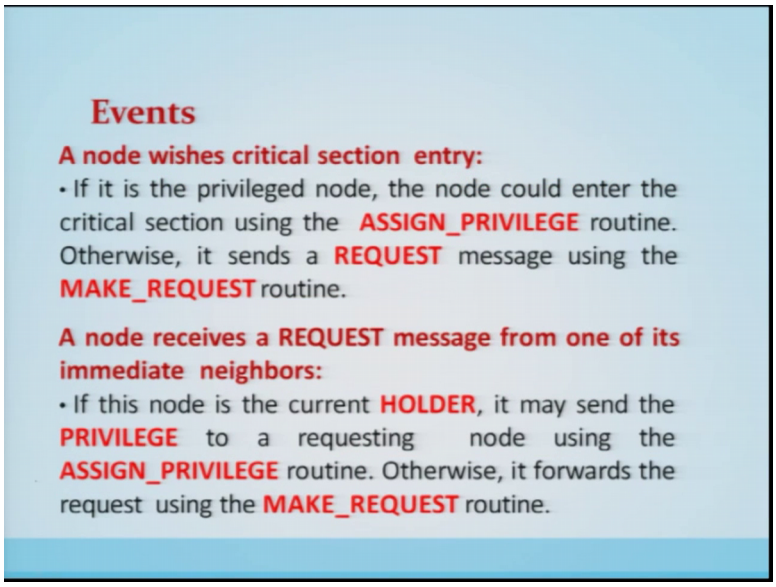
(Refer Slide Time: 27:16)



Below we mentioned 4 events that constitute the algorithm. So, the first event since that are not wishes to execute the critical section. So, this particular aspect basically is

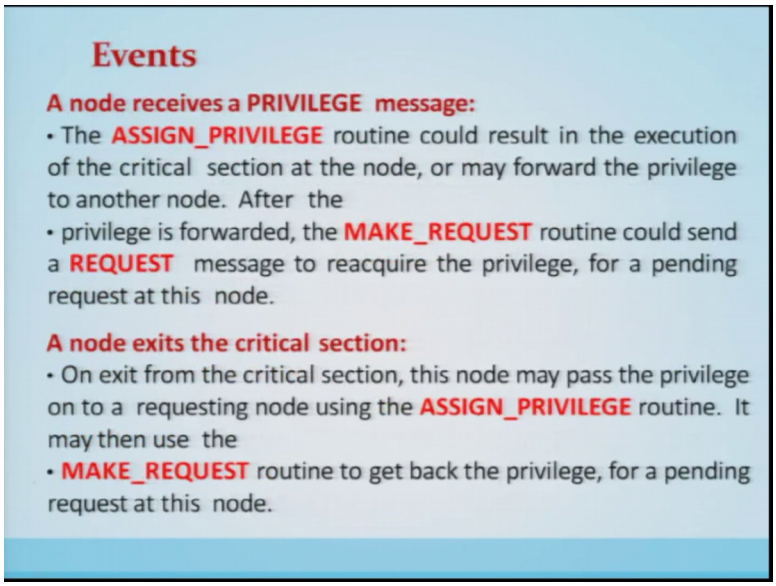
handled by the algorithmically. Algorithmic functionality such as Enqueue REQUEST Q and self assign privilege and make requests. A node it receives a REQUEST message from one of it is immediate neighbors x and Enqueue requests Q and x. A node receives a privileged message, then in that case holder will become to self and assign the privilege and make requests. A node exists the critical section, using will become false and assign privilege and make request.

(Refer Slide Time: 28:05)



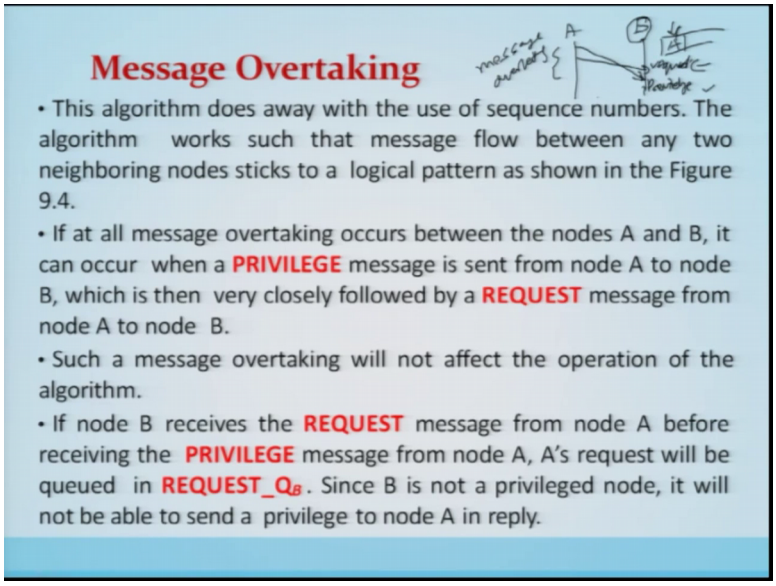
Events a node wishes the critical section entry; if it is the privileged node and the node could enter the critical section using assign privilege routine. Otherwise, it sends the REQUEST message using MAKE REQUEST routine. A node receives a REQUEST message from one of it is immediate neighbors if the node is the current holder it may send the privilege to the requesting node using the assign privilege routine. Otherwise it forwards the REQUEST using make requests. Routine the assign amount receives the privilege message.

(Refer Slide Time: 28:31)



The assign privilege routine could result in the execution of the critical section at the node or may forward the privilege to another node. After the privilege is forwarded, the make REQUEST routine could send a REQUEST message to reacquire the privilege, for the pending REQUEST at this node. A node exists the critical section; on exit the critical section this node may pass the privilege on to a requesting node using the ASSIGN PRIVILEGE for pending request.

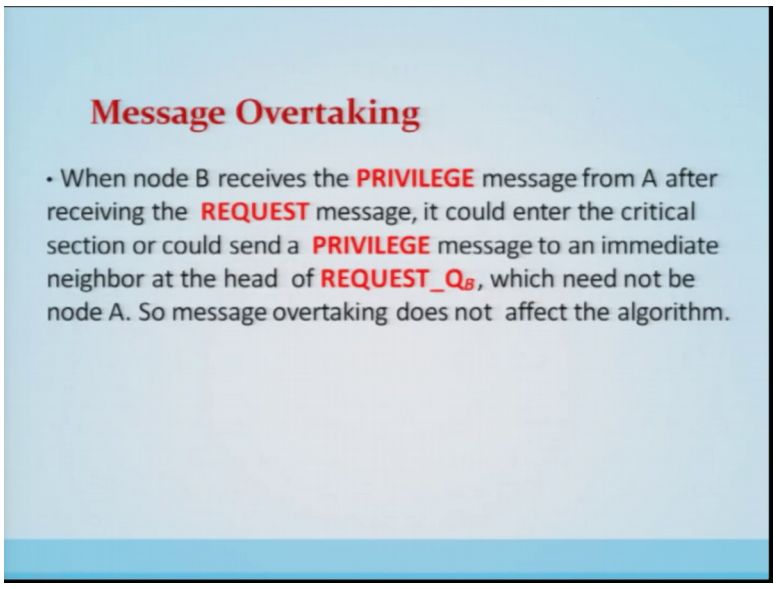
(Refer Slide Time: 29:09)



So, this particular picture will represent or illustrates the message over taking, this algorithm does away with the use of sequence numbers. The algorithm works such that the message flow between any 2 neighboring node is sticks to a logical pattern. So, if at all the message over taking occurs between node A and B, it can it can occur when privilege message is sent from node A to node B, which is then very closely followed by the REQUEST message from node. To such a REQUEST such a message overtaking will not affect the operation of the algorithm. You can see through this particular example so if node A and B. So, if node A is sending the it can node A is sent to node B to REQUEST and such that the previous recent REQUEST is basically arriving late and so basically here this. So, it has sent the privilege and then immediately a wants that privilege to be used so, it will send the request.

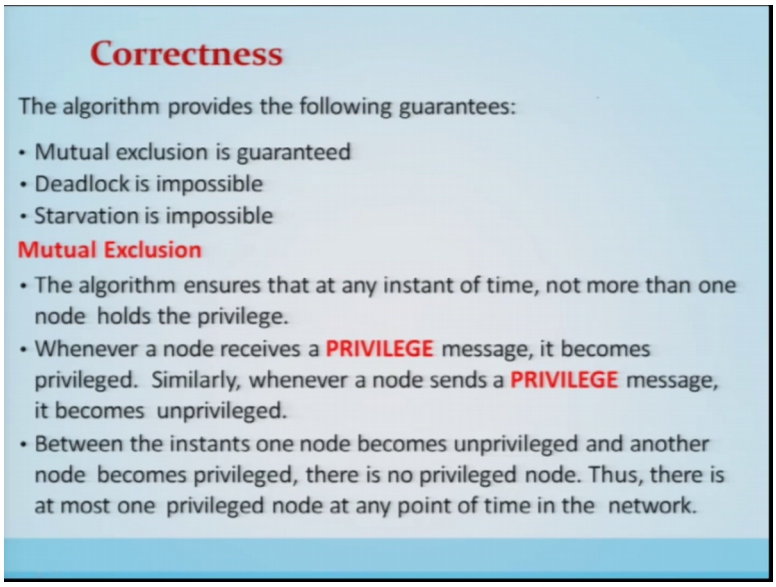
So, at the side B the REQUEST will arrive first, although it is not having the privilege. So, it will be put in the queue of requests Q of B. So, once the privilege arrives at B. So, B will become a privilege node so it becomes a privilege node, then basically it will check the REQUEST Q and the A will be on the head of the queue it will be Q and it will send the reply, it will send the privilege back again 2 A. So, even this kind of example where this is called a message over taking is not going to affect the working of this algorithm.

(Refer Slide Time: 31:18)



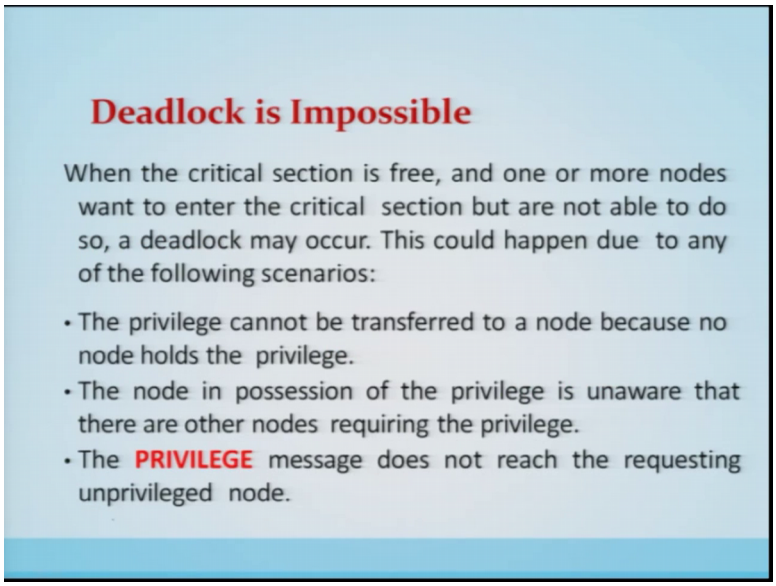
So, when a node B receives the privileged message from A after receiving the REQUEST message, it could enter the critical section or it could send the privilege message to an immediate neighbor at the head of the REQUEST Q which need not be node A. So, the message overtaking does not affect the algorithm that I way explain.

(Refer Slide Time: 31:34)



Now, correctness the algorithm provides the following guarantees; mutual exclusion is guaranteed, then deadlock is impossible, starvation is impossible. Mutual exclusion, ensures that at any at any instant of time, not more than 1 node holds the privilege message, hence the mutual exclusion is granted. So, whenever a node receives a privileged message, it becomes privileged. Similarly, whenever a node sends a privilege message, it becomes unprivileged. Between the instants one node becomes unprivileged and another node becomes privileged, there is no privileged node. Thus, there is at most one privileged node at any point of time in the network deadlock is impossible.

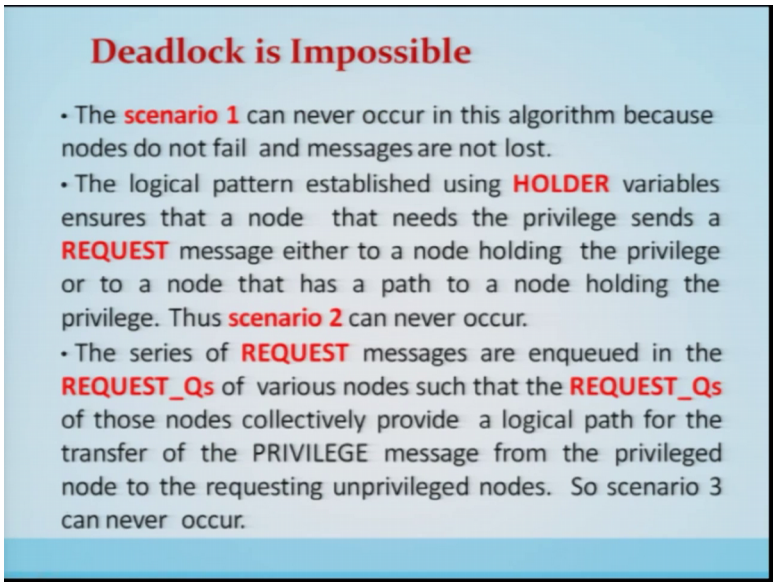
(Refer Slide Time: 32:22)



So, when the critical section is free, one or more nodes want to enter the critical section, but are not able to do so, a deadlock may occur. This could happen due to the any of the following scenarios.

A privilege cannot be transferred to a node because no node holds the privilege. A node is in possession of privileges unaware that there is other node requiring the privilege. The privileged message does not reach the requesting unprivileged node. So, all 3 conditions which basically possibly will get to a deadlock is not possible here in the algorithm, what is taken care by the algorithm; hence there is deadlock is impossible.

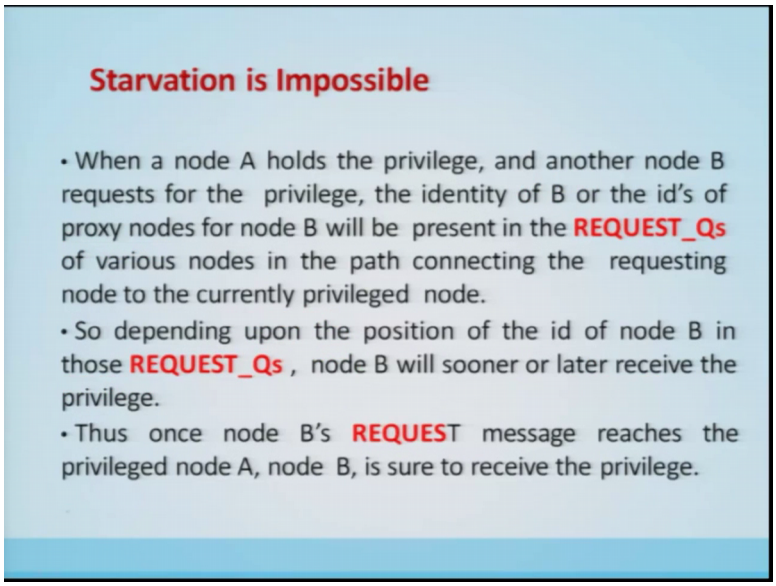
(Refer Slide Time: 33:04)



So, these are basically the scenarios in which none of these 3 conditions will arise, which will basically the reason to become the deadlock in the algorithm.

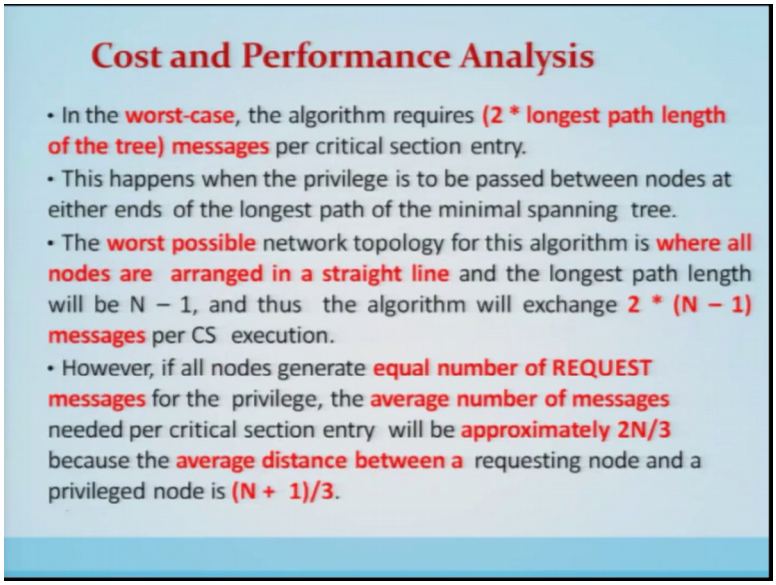
So, scenario 1 says that scenario 1 can never occur in this algorithm because nodes do not fail in the message are not lost. This is the junction of the algorithm; the logical pattern established using the holder variable ensures that, a node that needs the privilege sends a REQUEST message either to a node holding the privilege or to a node that has a path which leads to the token. The scenario to also cannot happen that is node in the possession of the privilege is unaware that, there are other nodes requiring the privilege. The series of requests messages are enquired in the REQUEST Q of various nodes, such as REQUEST Q of those nodes collectively provide the logical path for the transfer of privileged message, from the privilege node to the requesting unprivileged nodes. So, scenario 3 can never occur, hence the starvation is impossible in this algorithm.

(Refer Slide Time: 34:09)



Third, another aspect is called starvation. Starvation is also impossible when a node A holds the privilege and another node B requests for the privilege, the identity of node B or the id's of the proxy nodes of node B will be present in the REQUEST Q of various nodes in the path connecting the requesting node to the currently privileged node. So, depending upon the position of id of the node B in those REQUEST Q, node B will sooner or later receive the privilege. Thus once the node B's REQUEST message reaches the privileged node A node B, is sure to receive the privilege. So, hence the starvation is also impossible.

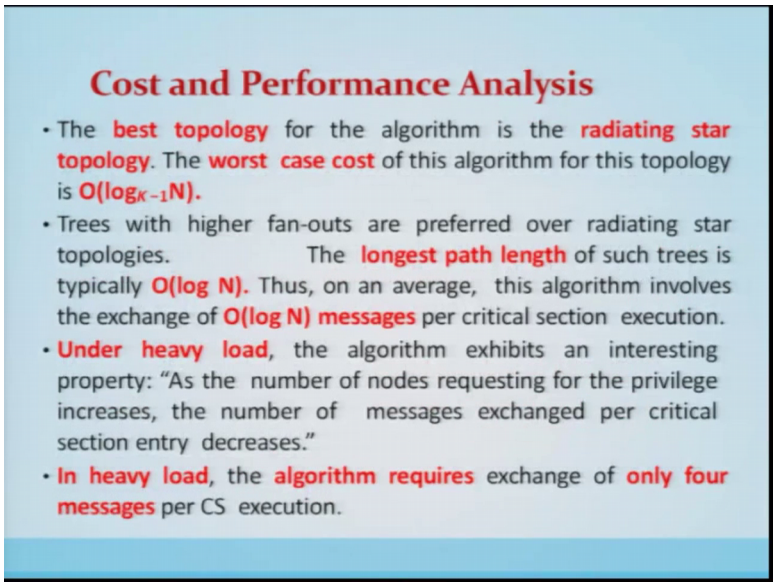
(Refer Slide Time: 34:51)



Now, cost and performance analysis of the algorithm, in the worst case the algorithm requires 2 times longest path, length of the tree that number of messages per critical section entry. This happens when the privilege is to be passed between the nodes at either ends of the longest path of a minimum spanning tree. The worst possible that network topology of this algorithm is where all the nodes are arranged in a linear fashion and the longest path will be n minus 1 and thus the algorithm will exchange 2 times N minus 1 message per critical section.

However, if all the node generate equal number of REQUEST messages for the privilege, the average number of message messages needed per critical section entry will approximately in 2 times N by 3 because the average distance between requesting node and a privileged node is N plus 1 by 3.

(Refer Slide Time: 35:44)



The best topology for the algorithm is the radiating a star topology. The worst case cost of this algorithm for this topology is O log to the base k minus 1 N. So, trees with higher fan outs are preferred over radiating star topologies. The longest path length of such trees is typically of O log N. Thus on an average, this algorithm involves the exchange of O log n messages per critical section execution.

Under the heavy load, the algorithm exhibits an interesting property; as the number of nodes requesting that the privileges increases, the number of message exchange per

critical section entry decreases. Hence in the heavy load, the algorithm requires a message exchange of only 4 messages per get those section entry.

(Refer Slide Time: 36:32)

Comparison of Distributed Mutual Exclusion Algorithms

Non-Token	Response Time	Synchronization Delay	Messages (Low Load)	Messages (High Load)	Concept
Lamport	2T+E	T	3(N-1)	3(N-1)	Give Priority with Time Stamp
Ricart-Agrawala	2T+E	T	2(N-1)	2(N-1)	Uses Implicit Release Message strategy.
Quorum	**Response Time**	**Synchronization Delay**	**Messages (Low Load)**	**Messages (High Load)**	**Concept**
Maekawa	2T+E	2T	3* √N	5* √N	Uses Theory of projective planes for quorums.
Agarwal-El Abbadi	2T+E	2T	log N	log N	Uses Tree structured quorums.
Token	**Response Time**	**Synchronization Delay**	**Messages (Low Load)**	**Messages (High Load)**	**Concept**
Suzuki-Kasami	2T+E	T	N	N	Token as a privilege message.
Raymond Tree	T(log N)+E	T(log N) / 2	log N	4	Minimal Spanning Tree and unrooted tree structure

Now, this particular comparison of this different distributed mutual exclusion algorithm, we have seen the approaches which are called Lamport and Ricart agrawala algorithm. They are non token based algorithms and here the synchronization delays they was T and the message is required in lamport was 3 times n minus 1, Ricart agrawala was basically able to complete this in 2 n minus 1. Now in Lamports algorithm basically the priority or the fairness is achieved through the time stands and Ricart agarwala algorithm uses implicit release message strategy.

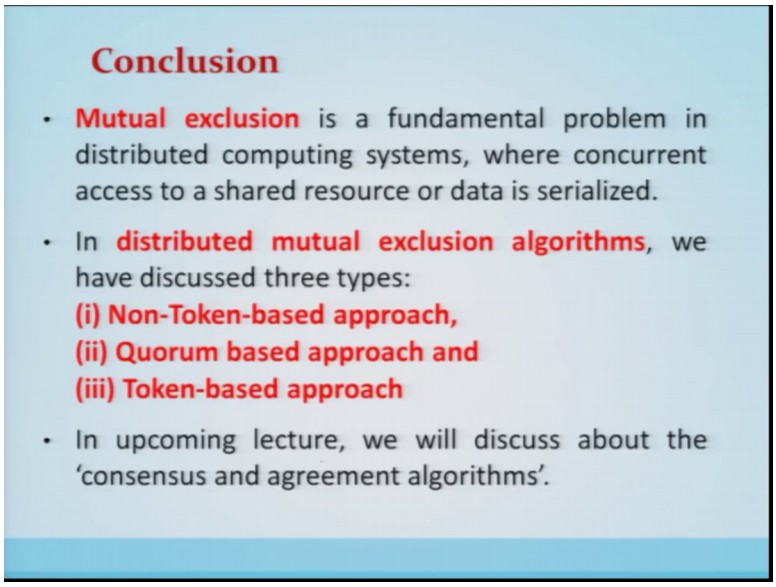
Now, the another class of algorithm which we have seen is based on quorum based algorithms, here the number of messages are to be reduced why because the permissions are not taking from everyone, but from a quorum or a subset of those sets. There are 2 algorithms we have seen Maekawa algorithm and Agarwala El Abbadi algorithm. Both algorithms they take the spawns time 2 times t, but the messages were quite reduced here in this case. So, messages in the low load 3 times root n and in agarwala algorithm log N.

Now, another class of algorithm we have seen for mutual exclusion is called token based algorithms, in token based algorithm today we have seen 2 different algorithm Suzuki kasami and Raymond tree algorithm. These algorithms require the synchronization delay Sd and this t times log N by 2, different messages now Suzuki kasami algorithm and

Raymond tree algorithm are based on the token. So, the node which is holding a token basically will be. So, that case there is only single token. So, mutual exclusion guarantee is trivial, but all other aspects we have seen for the working of the algorithm, that it is they should be deadlock free starvation free and the token loss is to be taken care of and regeneration of token also is to be taken care of so and how the token are to be searched it depends upon increment tree, that is in the tree structure finding token is quite easy. So, this is Suzuki kasami algorithms say broadcast algorithm.

So, the messages are sent to world. So, all these algorithms are there depends upon different applications, within which application which algorithm is going to be useful.

(Refer Slide Time: 39:10)



So, based on these parameters or performance parameters these algorithms are selected and used for different applications. Now conclusion mutual exclusion is a fundamental problem in distributed computing system, where concurrent access to the shared resource or a data is serialized. In distributed mutual exclusion algorithms we have discussed 3 types of algorithms non token based, quorum based and token based approach.

In upcoming lectures we will discuss about consensus and agreement algorithms.

Thank you.