

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

Lecture - 07
Distributed Mutual Exclusion Algorithms and Non-Token Based Approaches

Lecture 7 distributed mutual exclusion algorithms and non-token based approach.

(Refer Slide Time: 00:18)

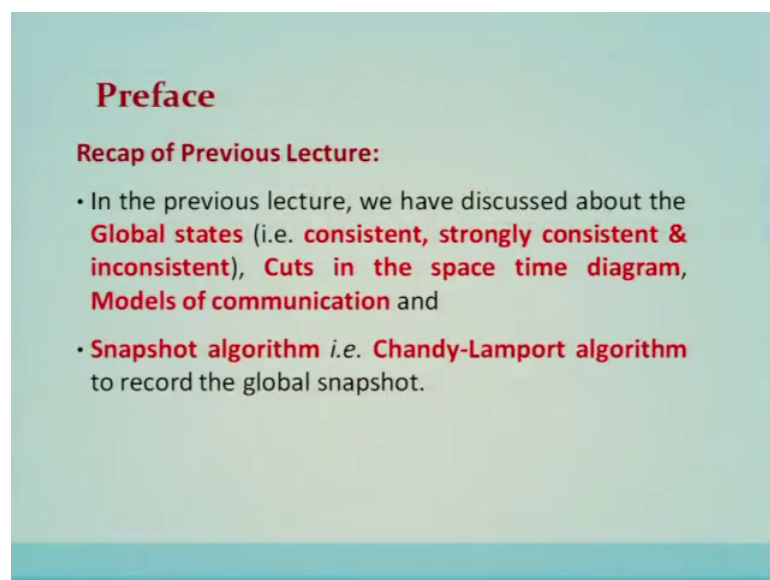


Lecture: 07

**Distributed Mutual Exclusion
Algorithms &
Non-Token Based Approaches**

 **Rajiv Misra**
Dept. of Computer Science &
Engineering
Indian Institute of Technology Patna
rajivm@iitp.ac.in

(Refer Slide Time: 00:21)



Preface

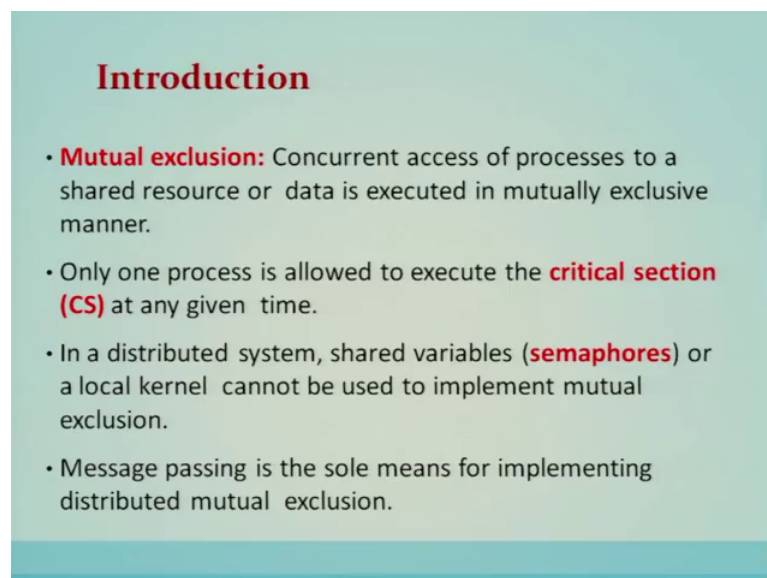
Recap of Previous Lecture:

- In the previous lecture, we have discussed about the **Global states** (i.e. **consistent, strongly consistent & inconsistent**), **Cuts in the space time diagram**, **Models of communication** and
- **Snapshot algorithm** i.e. **Chandy-Lamport algorithm** to record the global snapshot.

Preface; recap of previous lecture. In previous lecture we have discussed about the global state; that is consistent, strongly consistent and inconsistent global states, cuts in a space time diagram, and model of communications. We have also seen the snapshot algorithm; that is given by Chandy-Lamport and that is called Chandy-Lamport algorithm, to record the global snapshot.

Content of this lecture: In this lecture, we will discuss about mutual exclusion algorithm for distributed computing systems; such as non token based approach, quorum based approach, and token based approach, for the distributed mutual exclusion algorithms

(Refer Slide Time: 01:12)



Introduction

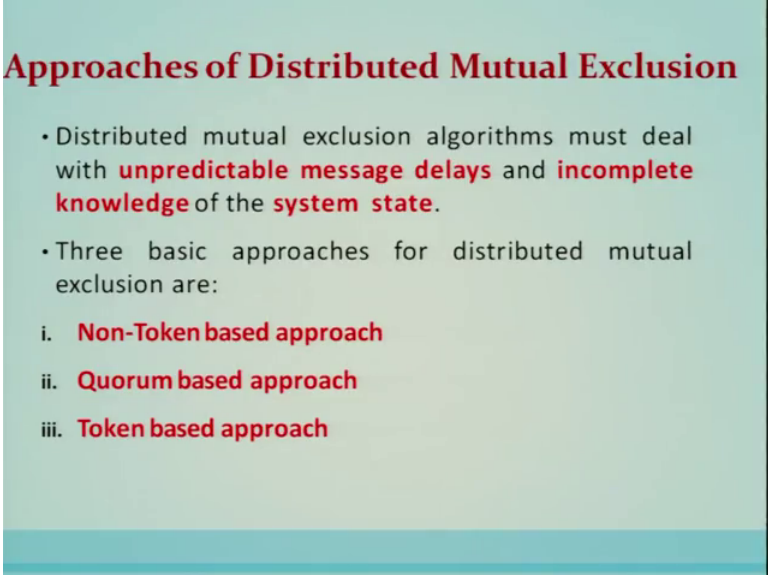
- **Mutual exclusion:** Concurrent access of processes to a shared resource or data is executed in mutually exclusive manner.
- Only one process is allowed to execute the **critical section (CS)** at any given time.
- In a distributed system, shared variables (**semaphores**) or a local kernel cannot be used to implement mutual exclusion.
- Message passing is the sole means for implementing distributed mutual exclusion.

Introduction. Mutual exclusion in a distributed system. So, concurrent accesses to the processes to a shared resource or a data, is executed in a mutually exclusive manner; that is one process at a time, is allowed to execute in a critical section. And in the constraints of a distributed system where there is no shared memory, there is no physical clock, and the messages are basically, is having unpredictable delays, but it is a finite delay.

So, in that scenario, designing this mutual exclusion, distributed mutual exclusion, is basically going to be a challenging task. So, we are going to see in the introduction, that only one process is allowed to execute the critical section at any given time. So, in a distributed shared memory, the existence of semaphores or shared variables, and a local kernel system cannot be used to implement the mutual exclusion. Hence we are going to discuss how the distributed mutual exclusions, are going to be implemented, with the

message passing only. So, message passing is the sole means for implementing distributed mutual exclusion approaches of distributed mutual exclusion.

(Refer Slide Time: 00:31)



Approaches of Distributed Mutual Exclusion

- Distributed mutual exclusion algorithms must deal with **unpredictable message delays** and **incomplete knowledge** of the **system state**.
- Three basic approaches for distributed mutual exclusion are:
 - i. **Non-Token based approach**
 - ii. **Quorum based approach**
 - iii. **Token based approach**

Distributed mutual algorithms must deal with unpredictable message delays, and incomplete knowledge of the system state. Three basic approaches for distributed mutual exclusion, we are going to cover up in this part of the lecture. They are non token based approach, quorum based approach, and token based approach.

(Refer Slide Time: 03:01)



(i) Non-token based approach:

- Two or more successive **rounds of messages** are exchanged among the sites to determine which site will enter the CS next.

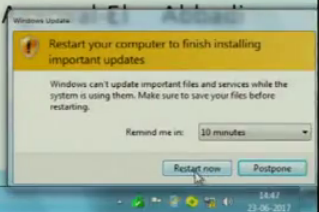
e.g. Lamport Algorithm, RicartAgarwala Algorithm etc.

Non token based approach consists of; two or more successive rounds of message exchanges, among the sites to determine which site is allowed to enter into the critical section next. For example, we are going to cover up two algorithms, they are known as Lamports algorithm and recartagarwala algorithm for non token based approaches

(Refer Slide Time: 03:26)

(ii) Quorum based approach:

- Each site requests permission to execute the CS from a subset of sites (called a **quorum**).
- Any two quorums contain a **common site**.
- This common site is responsible to make sure that only one request executes the CS at any time.
- e.g. Maekawa's Algorithm, Quorum-Based Algorithm etc.



Quorum based approach: Each site request permission to execute the critical section from only a subset of sites, not all the sites, and that subset of site is called a quorum. So, any two quorums contains a common site. So, this common site is responsible to make sure that only one request executes that only one request executes the critical section at any time. For example, the algorithms like Meekawa algorithm, Agarwal el Abbadi algorithm. They are all quorum based algorithm which we are going to cover up.

Then next is the token based approach for designing distributed algorithms.

(Refer Slide Time: 04:05)

(iii) Token-based approach:

- A unique token (also known as the **PRIVILEGE message**) is shared among the sites.
- A site is allowed to enter its CS if it possesses the token.
- Mutual exclusion is ensured because the **token is unique**.
- e.g. Suzuki-Kasami's Broadcast Algorithm, Raymond's Tree-Based Algorithm etc.

So, here unique token, is also known as the privileged message is shared among the sites. A site is allowed to enter into the critical section, if it possesses that particular token. So, mutual exclusion ensured, because there is only one token, and that is unique

(Refer Slide Time: 04:34)

Preliminaries: System Model

- The system consists of N sites, S_1, S_2, \dots, S_N .
- We assume that a single process is running on each site. The process at site S_i is denoted by p_i .
- A site can be in one of the following three states: requesting the CS, executing the CS, or neither requesting nor executing the CS (i.e., idle).
- In the 'requesting the CS' state, the site is blocked and can not make further requests for the CS. In the 'idle' state, the site is executing outside the CS.
- In token-based algorithms, a site can also be in a state where a site holding the token is executing outside the CS (called the **idle token** state).
- At any instant, a site may have several pending requests for CS. A site queues up these requests and serves them one at a time.

So, the token based approaches are Suzuki Kasami broadcast algorithm, Raymonds tree based algorithms now preliminaries for the system model we are going to, now cover up now. So, the system consists of n different sites S_1 to S_n . we assume that a single process is running on each side. So, the process at site i is denoted by p_i . So, either we

can. So, instead of saying sites, we will be now calling it as a process running on that particular site

The site can be in one of the following three states, requesting for a critical section, executing the critical section, or neither executing nor requesting for the critical section, they are idle state. So, in requesting the critical section state, the site is blocked, and cannot make for the request for the critical section. In idle state, the site is executing outside the critical section. In a token based algorithm, the site can also be in the state where the site holding the token is executing outside the critical section, and this is called an idle token. So, at any instant, a site may have several pending requests for critical section. A site queues up these requests and serves them one at a time

(Refer Slide Time: 05:47)

Requirements of Mutual Exclusion Algorithms

1. Safety Property: At any instant, **only one process** can execute the critical section.

2. Liveness Property: This property states the absence of **deadlock and starvation**. Two or more sites should not endlessly wait for messages which will never arrive.

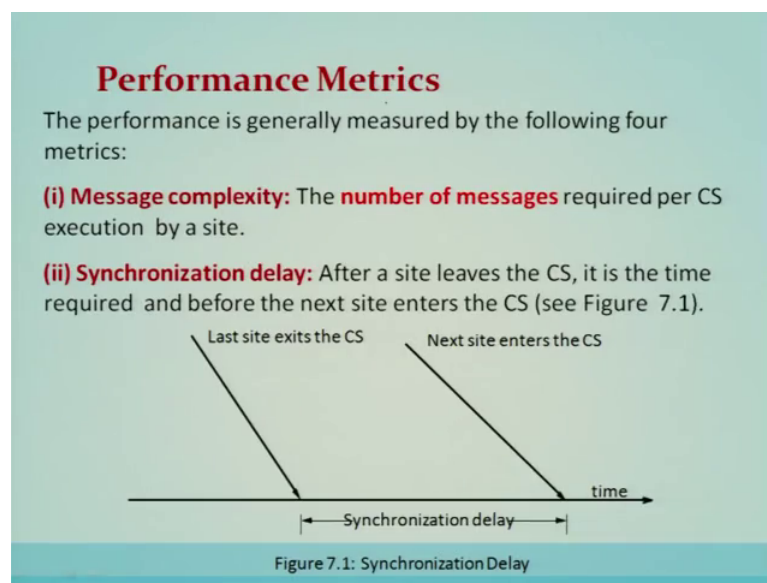
3. Fairness: Each process gets a fair chance to execute the CS. Fairness property generally means the CS execution requests are **executed in the order of their arrival** (time is determined by a **logical clock**) in the system.

Requirements of mutual exclusion algorithm: The primary requirement is the safety property at any instant, only one process can execute the critical section, and this is the most important property, essential property it is.

The next requirement of a mutual exclusion algorithm is, the liveness property. This says that this property states that, absence of deadlock and starvation, it should be maintained; that means, two or more site should not endlessly wait for the message, which will never arrive, and this is an important property, to ensure the mutual exclusion.

Third property is called a fairness. So, each process gets a fair chance to execute the critical section. Fairness property generally means, the critical section execution, requests are executed in the order of their arrival in the system, and this order of arrival is determined by the system of logical clocks, and this is also an important property for the different distributed mutual exclusion algorithms. So, this the essential, the first one is the essential property, the other properties are important properties

(Refer Slide Time: 07:25)



Now, we are going to see the performance metrics, and these performance metrics are used to compare the algorithms distributed mutual exclusion algorithms. So, the first property is, basically performance metric is called message complexity. So, message complexity is the number of messages, required per critical section execution by the site.

Second is the synchronization delay. So, after a site leaves the critical section, it is the time required, and before the next site enters the critical section. So, here we can see that, after a site exits the critical section and a new site when it enters, that particular delay is called a synchronization delay.

So, when a site exists the critical section, there are certain rounds of message exchanges, which will prepare the system. So, that the next request or next process can be allowed to go into critical section, and that is why synchronization delay and some of the distributed algorithms are basically seen.

(Refer Slide Time: 08:29)

Performance Metrics

(iii) **Response time:** The time interval a request waits for its CS execution to be over after its request messages have been sent out (see below Figure 7.2)

(iv) **System throughput:** The rate at which the system executes requests for the CS.

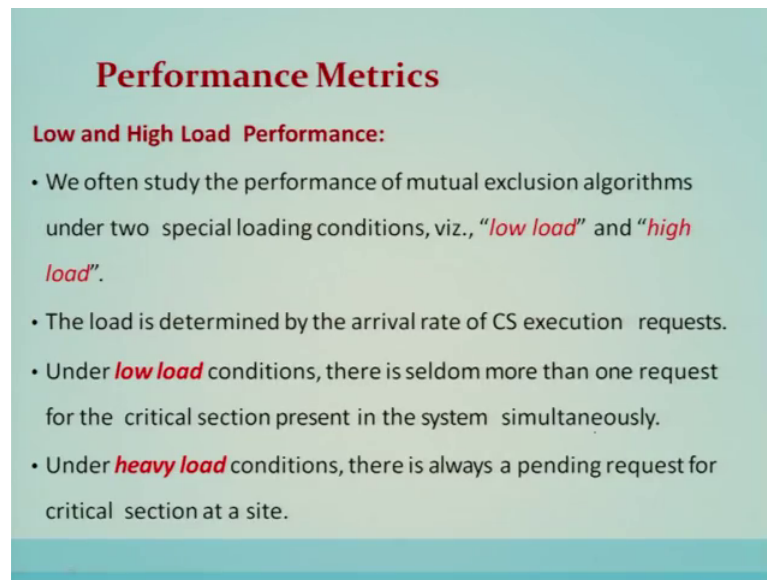
$$\text{system throughput} = 1 / (SD + E)$$

where SD is the synchronization delay and E is the average critical section execution time.

Third performance matter is called basically the response time. The time interval a request waits for its critical section execution to be over after its requests message have been sent out, and that is called response time. So, here when the request message has been send out, till the site exists the critical section, the entire duration of the time is called the response time.

Now, fourth time is called, fourth metrics is called system throughput. The rate at which the system executes the request per critical section is called system throughput. System throughput can be calculated using a formula; that is $1 / (SD + E)$ means the synchronization delay plus the average execution time, where SD synchronization delay, and E is the average critical section execution time, and that basically will compute the throughput of the critical section execution. So, these are four performance matrices we have seen, and we are going to use it to compare different distributed mutual exclusion algorithms.

(Refer Slide Time: 09:44)



Performance Metrics

Low and High Load Performance:

- We often study the performance of mutual exclusion algorithms under two special loading conditions, viz., "*low load*" and "*high load*".
- The load is determined by the arrival rate of CS execution requests.
- Under *low load* conditions, there is seldom more than one request for the critical section present in the system simultaneously.
- Under *heavy load* conditions, there is always a pending request for critical section at a site.

Now, another performance is about high load and low load situation. So, we often study performance of mutual exclusion algorithm under two special loading conditions; that is the low load and the high load. The load is determined by the arrival rate of critical section requests. Under the low load condition there is seldom more than one request for the critical section, present in the system simultaneously. Under heavy conditions, there is always a pending request for a critical section at a particular site.

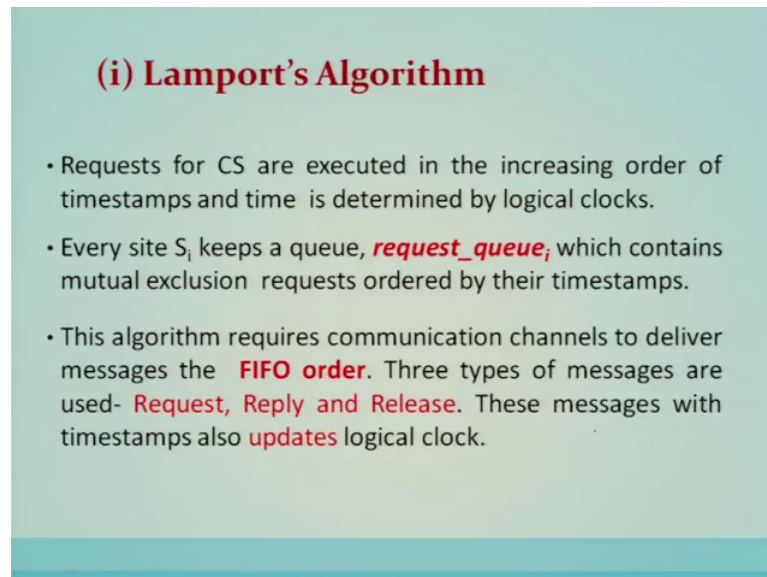
(Refer Slide Time: 10:25)



(i) Non-Token Based Approaches

So, non token based approaches. Lamports algorithm. So, Lamports algorithm is first in this particular class of non token based algorithms. So, in Lamports algorithm of distributed mutual exclusion is using the system of logical clock, which is designed by the Leslie Lamport.

(Refer Slide Time: 10:43)



(i) Lamport's Algorithm

- Requests for CS are executed in the increasing order of timestamps and time is determined by logical clocks.
- Every site S_i keeps a queue, *request_queue*, which contains mutual exclusion requests ordered by their timestamps.
- This algorithm requires communication channels to deliver messages the **FIFO order**. Three types of messages are used- **Request, Reply and Release**. These messages with timestamps also **updates** logical clock.

So, this particular algorithm is a demonstration of use of the Lamports clock, in designing the distributed mutual exclusion algorithm. So, Lamports algorithm request for critical sections are executed in the increasing order of the timestamp, and the time is determined by the system of logical clock. And this will ensure that this is a fair algorithm. So; that means, this algorithm ensures the fairness property. Fairness property; that means, the order in which the requests are arriving and they are being served in that particular order and that is ensured by the using, the use of logical clocks, that we will see here in this algorithm.

Now, every site S_i keeps a queue; that is called a request queue of the process i , which contains the mutual exclusion request ordered by their timestamps. This algorithm requires the communication channels, to deliver the message in FIFO order. There are three type of messages, used in this algorithm, they are called request message, reply message, and release message. These messages with timestamps also update the logical clock, that we will see in the algorithm.

So, the algorithm; Lamports algorithm for non token based approach, for designing the distributed mutual exclusion. The first action is, requesting the critical section. So, when a site S_i want to enter the critical section, it broadcasts a request message, and request message has the parameter, the timestamp of and the process id that is i . So, the request message contains the timestamp request message of process i , and this message will be broadcast to all the other sites and places the request on its request queue also.

(Refer Slide Time: 12:50)

The Algorithm

Requesting the critical section:

- When a site S_i wants to enter the CS, it broadcasts a **REQUEST**(ts_i, i) message to all other sites and places the request on **request_queue_i**. ((ts_i, i) denotes the timestamp of the request.)
- When S_j receives the **REQUEST**(ts_i, i) message from site S_i , S_j places site S_i 's request on **request_queue_j** and it returns a **timestamped REPLY** message to S_i .

Executing the critical section: Site S_i enters the CS when the following two conditions hold:

- L1:** S_i has received a **message** with timestamp larger than (ts_i, i) from all other sites.
- L2:** S_i 's request is at the top of **request_queue_i**.

Now when a particular site S_j receives this request message from S_i , this S_j places the site as high requests, on his request queue that is request queue of j , and also returns a time stamped reply message back to S_i . So, these two actions we have seen; that is to request for the critical section execution by a particular process i .

Now, second part of this algorithm is about executing the critical section. The site S_i enters the critical section, when the following two conditions hold. The first condition is says that S_i has received a message with the timestamp, larger than t_{S_i} from all other sites. So; that means, S_i is or site i is, message is basically having the lowest timestamp. So, it is having the highest priority, and that is why it is executing inside critical section.

Another condition is L 2 condition, says that site S_i request is at the top of the request queue. Also if you see the request queue, the i is request will be at the head of the queue. These two conditions are basically ensuring that a particular process i , is executing the critical section.

Now, third part of the algorithm says that; once a particular process finishes the critical section execution, then it has to release the critical section execution. So, releasing the critical section, a site S_i upon executing the critical section, removes its request from the top of its request queue, and broadcast a timestamp release message to all other sites. So, it will send the release message. So, and also it will remove i from the queue, and the release message will be broadcasted. So, when a site S_j receives the release message from S_i , it removes S_i requests from its request queue. So, when a site removes, a request from its request queue, its own request may come at the top of the queue enabling it to enter into the critical section, if it is interested for, or if it is requesting to go in a critical section.

Now, correctness. So, theorem Lamports algorithm achieves mutual exclusion, that proof goes like by contradiction. Suppose the two sites S_i and S_j , they are executing inside the critical section concurrently at the same instance of a time. So, for this to happen, these two conditions must hold L 1 and L 2 at both the sites. This implies that at some instant in time let us say t both S_i and S_j have their own requests in the top of the request queue, and the condition L 1 holds at them.

So, without loss of generality, assume that S_i timestamp is smaller than smaller than, the timestamp of the request j . So, from condition L 1 and FIFO property of the communication channel, it is clear that at instant t , the request of S_i must be present in the request queue of j . So, this implies that S_j 's request is at the top of its own request queue, when a smaller timestamp requests S_i 's request is present in the request queue of j , and this is a contradiction. So, hence it is proved that Lamports algorithm achieves the mutual exclusion.

Another algorithm, another theorem is about Lamports algorithm, is about fairness. So, Lamports algorithm is fair. So, I have told you in the beginning that, this lampard algorithm is fair algorithm. The proof goes like this, the proof is by contradiction. Suppose a site S_i 's request has a smaller timestamp than the request of another site S_j and S_j is able to execute the critical section before S_i . So, for easy to execute the critical section, it has to satisfy the conditions L 1 and L 2, this implies that at some instant in time t S_j has its own request at the top of its queue, and it has also received the message with the timestamp, larger than the timestamp of its request from all other sites, but request queue at a site is ordered by the timestamp, and according to the assumption S_i

has the lower timestamp. So, S_i 's request must be placed ahead of S_j 's request in the request queue, and this is a contradiction. So, this basically proves that the Lamports algorithm is fair.

Lamports algorithm examples. Let us have three sites S_1 , S_2 and S_3 participating in the distributed mutual exclusion. Now site S_1 and S_2 are requesting for the critical section. So, S_1 will send these arrows, and S_2 also will send or broadcast this message to all other sites.

Now, S_2 , S_1 after receiving. So, these sites are, now S_2 after receiving the message from S_1 compares with its own timestamp, or a request timestamp, and a particular process. So, $1 < 2$. So, basically it is found that the incoming request, is having lower timestamp, so it will send a reply, back to S_1 and similarly S_3 is also neither requesting, nor it is executing critical section, it will also send the reply. So, S_1 will receive the replies from all other sites. So, now, S_1 will enter into the critical section. Now N site S_1 exists the critical section then it has to send the release message. So, release message will be sent to all other sites, and after receiving the release message, site S_2 will basically, will found itself on the top of the queue

Now, after receiving the reply message also from S_1 , the S_3 can go into the critical section. So, if you see the performance of Lamports algorithm for each critical section execution, Lamports algorithm requires; $N - 1$ request messages, $N - 1$ reply messages, and $N - 1$ release messages. So, the Lamports algorithm required $3N - 1$ messages per critical section invocation.

The synchronization delay of this algorithm is t . So, that you can see that once a particular process S_1 exists the critical section, then it requires this message exchange; that is the release message to flow, and that will take only t time till S_2 can go into the critical section again. So, the synchronization delay in this particular algorithm is given by t .

Optimization of this algorithm in Lamports algorithm reply messages can be omitted in certain conditions. for example, if site S_j receives a request message from S_i , after it has sent its own request message with a timestamp higher than the timestamp of site S_i 's request then site S_j , need not send the reply message to the site S_i why, because and S_j 's timestamp is lower. So, it is having higher priority. So, this is because when S_i 's S_i

receives the site S_j 's request with the timestamp higher than its own, it can conclude that site S_j does not have any smaller timestamp requests, which is still pending. With this optimization Lamports algorithm require that, between $3N - 1$ to $2N - 1$ messages per critical section invocation.

Now, the next algorithm which we are going to discuss, in the non token based approach is called the Recart Agarwala algorithm. The Recart Agarwala algorithm assumes the communication channel are FIFO also. In this algorithm same assumption is there, the algorithm uses only two types of messages; request and reply. So, this particular algorithm is not going to use the release message. The process sends a request message to all other processes, to request their permission to enter the critical section.

A process sends a reply message to a process, to give its permission to that particular process. So, processes used Lamport style logical clocks to assign the timestamp, to the critical section request, and timestamps are used to decide the priority of the request. Each process p_i maintains data structure, which is called a request default array RDI, the size of which is same as the number of processes, or the sites in the system.

So, initially the RDI'S for all INJ, they are equal to, they are initialized to zero. So, whenever p_i differs the request by p_j , it sets RDI of j is equal to 1, and after it has send a reply back to p_j , it resets again RDI of j is equal to 0. Let us see the use of this default replies in this Recart Agarwala algorithm.

Description of the algorithm. So, the first step of the algorithm is requesting the critical section. So, when a site S_i want to enter the critical section, it broadcasts timestamp request message to all other sites. now when S_j receives a request message from site S_i , it sends a reply to the site S_i , if the site S_j is neither requesting nor executing critical section, or if the site S_j is requesting and S_i 's request timestamp is smaller than S_j 's own timestamp, then also it will send a reply; otherwise the reply is deferred, and S_j will set the RD parameter of i is equal to 1.

Now, second step of this algorithm is executing the critical section site S_i enters the critical section, after it has received the reply message, from every site it sent a request message. When a site S_i exits the critical section, it sends all the default reply messages. So, for all j 's, if RDI of j is equal to 1, then it sends our reply message to S_j and sets RD parameter to 0. Note that when site receives a message it updates its clock using the

timestamps, which are basically piggy bagged in the messages . So, when S_i takes up our request for critical section for processing, it updates its local clock and assigns the timestamp to the request.

Now, correctness: Recart Agarwala algorithm achieves mutual exclusion. So, proof is by contradiction. Suppose two sites S_i and S_j are executing in the into the critical section concurrently. S_i 's requests has higher priority than the request of S_j . clearly S_i 's S_i receives S_j 's request, after it has made its own request. thus S_j can concurrently execute the critical section with S_i , only if S_i 's, S_i returns a reply to S_j ,in response to S_j 's request before S_i exits the critical section; however, this is impossible, as per the algorithm is concerned, because S_j 's request as the lower priority. So, it will differ it. So, that is why both the assumption, that S_i and S_j both are executing critical section, is contradicted. Therefore, Recart Agrwala algorithm achieves mutual exclusion.

Recart Agarwala algorithm example. Assume that site S_1 and S_2 are requesting for the critical section, and they will send the message to all other processes.

Now, you can see that when S_1 will receive this particular message, from S_2 and the timestamp of S_2 , is basically higher than the timestamp of S_1 . So, S_1 is having higher priority. So, it will differ it, and put in the data structure RD of i the value of j will become 1.

Now, after receiving the replies from all other sites, these are the reply messages, the site S_1 will go into the critical section, when site S_i S_1 exists the critical section, then it will send the replies to the differed messages. So, basically the request which was deferred, it will send the reply to that particular message.

Now, S_2 which is also interested to go in a critical section. Now at this point of time it has received the replies from all other message, and it will enter into the critical section. Performance of this algorithm, for each critical section execution Recart Agarwala algorithm requires only two types of messages; that is request and reply, and N minus 1 number of request messages, N minus 1 number of reply messages. So, total number of messages required per critical section execution in Recart Agarwala algorithm is $2N$ minus 1 messages only

Synchronization delay is the same, as you can see in this particular picture that, once site S_1 has come out of a critical section that will take t time to send the default messages, then only the next process, which has requested to go in a critical section, can basically enter into critical section. So, the synchronization delay s_t is basically t that we have expressed here in this algorithm

So, the conclusion; Mutual exclusion is fundamental problem in a distributed computing system, where concurrent accesses to the shared resource or a data, is serialized. Mutual exclusion in a distributed system requires not only one process be allowed to execute the critical section, at any given time, and also ensures the fairness at the same point of time. So, in this lecture, we have discussed about the concepts of distributed mutual exclusion, and we have also seen the non token based approaches; like Lamport algorithm and Ricart Agarwala algorithm, to basically achieve distributed mutual exclusion. So, in the upcoming lecture, we will discuss about the other schemes, which are quorum based schemes, and token based approaches.

Thank you.