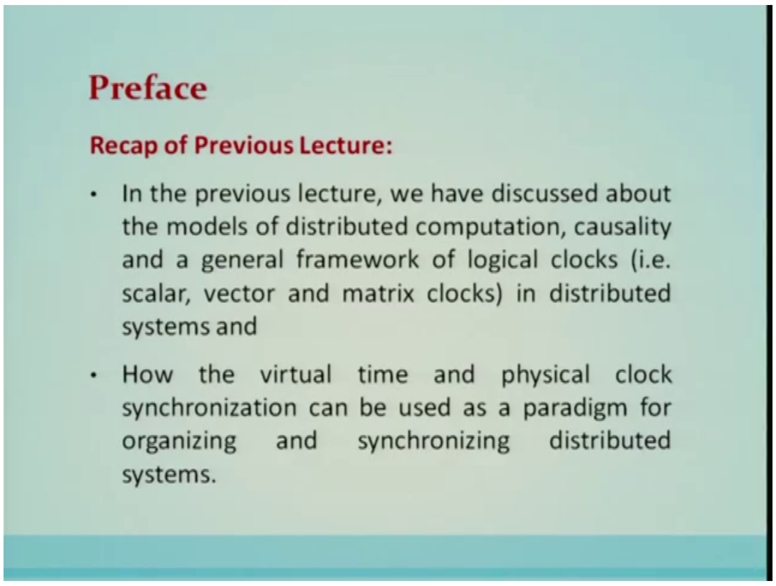
Distributed Systems

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

Lecture – 06
Global State and Snapshot Recording Algorithms

Lecture 6 Global State and Snapshot Recording Algorithms preface recap of previous lecture.

(Refer Slide Time: 00:23)



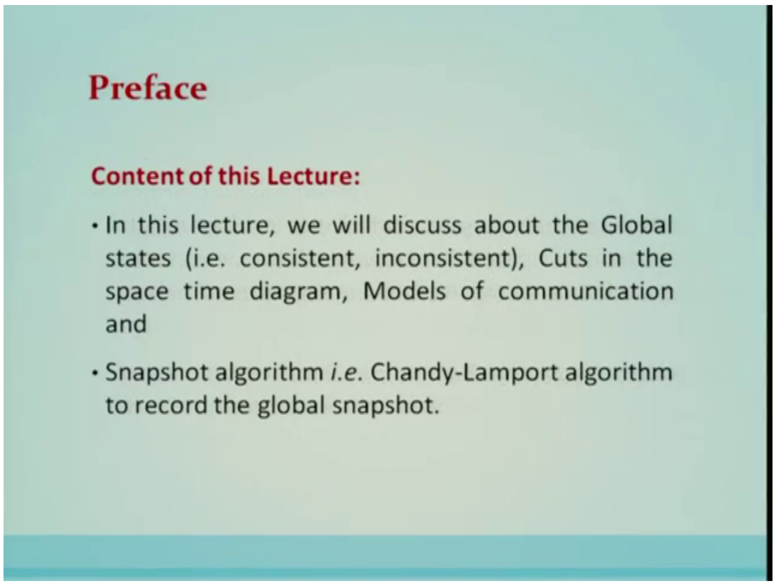
Preface

Recap of Previous Lecture:

- In the previous lecture, we have discussed about the models of distributed computation, causality and a general framework of logical clocks (i.e. scalar, vector and matrix clocks) in distributed systems and
- How the virtual time and physical clock synchronization can be used as a paradigm for organizing and synchronizing distributed systems.

In previous lecture we have discussed about the models of distributed computation causality and general framework of logical clocks that is a scalar a vector and matrix clock in distributed systems. And how the virtual time and physical clock synchronization can be used as a paradigm for organizing and synchronizing distributed systems content of this lecture.

(Refer Slide Time: 00:50)



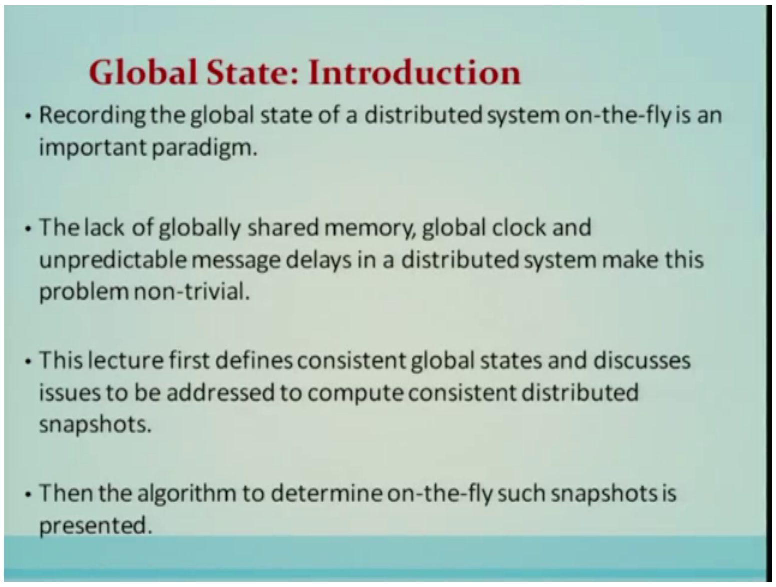
Preface

Content of this Lecture:

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

In this lecture we will discuss further details about the global states that is consistent and inconsistent states cuts in a space time diagram, Models of communication in distributed systems and snapshot algorithm that is given by the Chandy-Lamports algorithm to record the global snapshot.

(Refer Slide Time: 01:20)

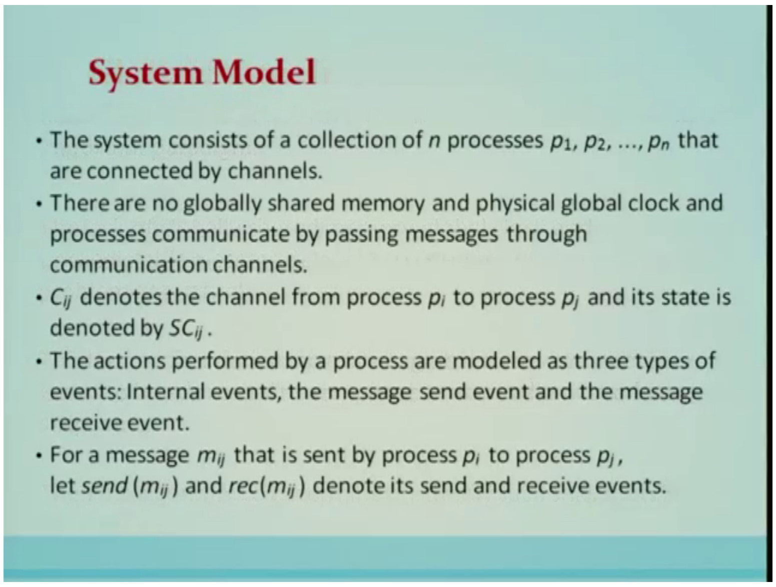


Global State: Introduction

- Recording the global state of a distributed system on-the-fly is an important paradigm.
- The lack of globally shared memory, global clock and unpredictable message delays in a distributed system make this problem non-trivial.
- This lecture first defines consistent global states and discusses issues to be addressed to compute consistent distributed snapshots.
- Then the algorithm to determine on-the-fly such snapshots is presented.

Global State Introduction recording the global state of a distributed system on-the-fly is an important paradigm. The lack of global shared memory and the lack of global clock and unpredictable message delays in an distributed system makes this problem non-trivial. This lecture first defines consistent global state and discusses issues to be discussed to be addressed to compute consistent distributed snapshot then the algorithm to determine on the fly such snapshot is presented in this part of the lecture.

(Refer Slide Time: 02:03)



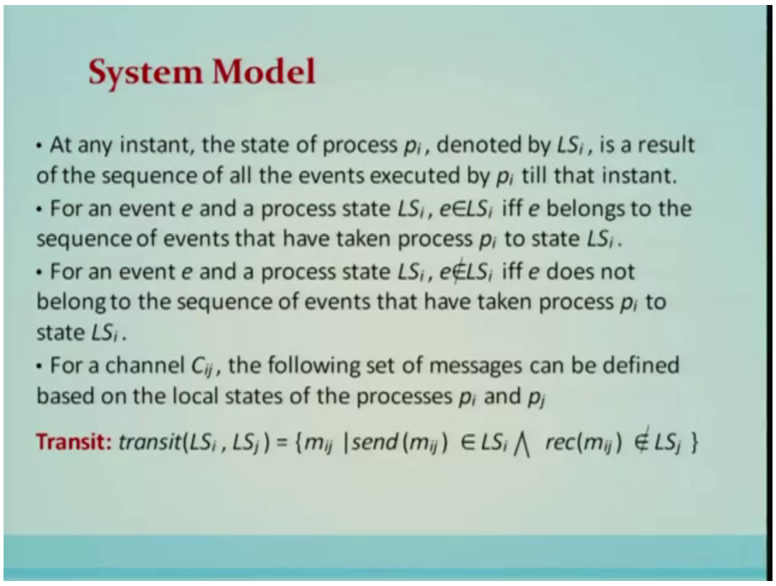
System Model

- The system consists of a collection of n processes $p_1, p_2, \ldots, p_n$ that are connected by channels.
- There are no globally shared memory and physical global clock and processes communicate by passing messages through communication channels.
- C_{ij} denotes the channel from process p_i to process p_j and its state is denoted by SC_{ij} .
- The actions performed by a process are modeled as three types of events: Internal events, the message send event and the message receive event.
- For a message m_{ij} that is sent by process p_i to process p_j , let *send* (m_{ij}) and *rec*(m_{ij}) denote its send and receive events.

System model which we are going to consider in this algorithm design the system consists of a collection of n processes p 1 to pn that are connected by a communication channel, there are no globally shared memory and there is no physical global clock and the processes communicate by passing messages through the communication channel. So, the communication channel is modeled by Cij that is the channel which connects from process i to process j and it is the state that is the channels state is represented by SCij the actions perform by processes are modeled as 3 different type of events the internal events the message send event and the message receive events.

So, all the processes are basically a 3 events types that is internal events message send event and the message receive events for a message mij that is sent by a process i to process j let send mij and receive mij denote the send and receive events. So, with these notations we are going to see some more details.

(Refer Slide Time: 03:22)



System Model

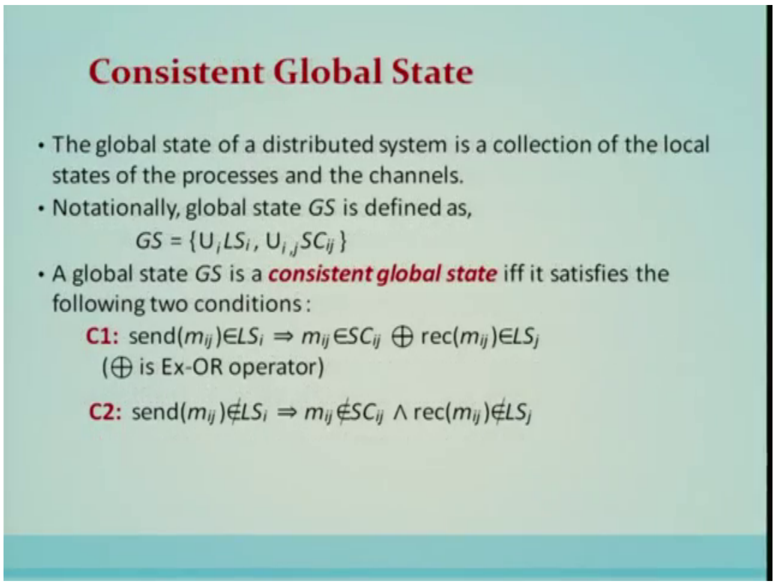
- At any instant, the state of process p_i, denoted by LS_i, is a result of the sequence of all the events executed by p_i till that instant.
- For an event e and a process state LS_i, $e \in LS_i$ iff e belongs to the sequence of events that have taken process p_i to state LS_i.
- For an event e and a process state LS_i, $e \notin LS_i$ iff e does not belong to the sequence of events that have taken process p_i to state LS_i.
- For a channel C_{ij}, the following set of messages can be defined based on the local states of the processes p_i and p_j

Transit: $transit(LS_i, LS_j) = \{m_{ij} \mid send(m_{ij}) \in LS_i \wedge rec(m_{ij}) \notin LS_j\}$

So, at any instant the state of a process i is denoted by LSi is the result of the sequence of all the events executed by pi till that time. For an event e and a process state LSi event is in LSi if and only if belongs to the sequence of events that have taken process pi to a state LSi. For an event e and a process the state LSi e is not in LSi if and only if does not belong to the sequence of event that have taken process pi to LS state.

For a channel Cij the following set of messages can be defined based on the local states of a process pi and process pj transit. So, transit between LSi and LS j that is the local state of i and local state of j is nothing, but the message mij this means; that means, the send of mij is in local state of i and receive mij is not in local state of j.

(Refer Slide Time: 04:59)



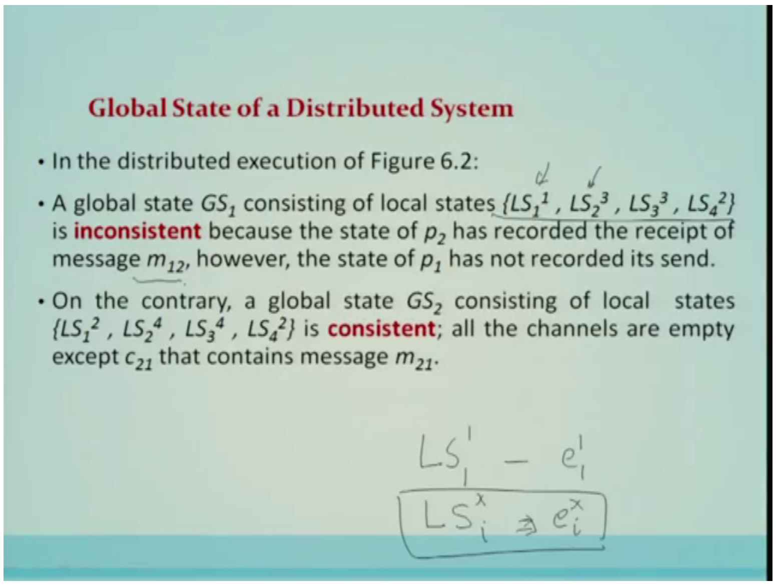
Consistent Global State

- The global state of a distributed system is a collection of the local states of the processes and the channels.
- Notationally, global state GS is defined as,
  $GS = \{\bigcup_i LS_i, \bigcup_{i,j} SC_{ij}\}$
- A global state GS is a ***consistent global state*** iff it satisfies the following two conditions:
  C1: $send(m_{ij}) \in LS_i \Rightarrow m_{ij} \in SC_{ij} \oplus rec(m_{ij}) \in LS_j$
  ($\oplus$ is Ex-OR operator)
  C2: $send(m_{ij}) \notin LS_i \Rightarrow m_{ij} \notin SC_{ij} \wedge rec(m_{ij}) \notin LS_j$

Where it is the message is in transit that is in the channel. So, consistent Global State definition; the global state of a distributed system is a collection of local state of the processes and the state of channels notationally global state GS is defined as the union of is the union of all processes local state and the union of all ij; that means, all the state of a state of channels.

So; that means, global state is nothing, but the collection of all the local states and collection of all the channel states. So, a global state GS is a consistent global state if and only if satisfies the following 2 conditions that is the condition c one and the condition C 2 condition C 1 says that send mij is an event is in LSi that is local state of process i this implies that the message mij is either in channel state of ij or the message is received mij at the process GS local state that is LSi. Similarly the condition C 2 says that if this send event send of a message mij is not in the local state of i this implies that the message mij is not in the state of a channel recorded and also the received of mij message is not in the local state.

(Refer Slide Time: 06:45)



Global State of a Distributed System

- In the distributed execution of Figure 6.2:
- A global state GS_1 consisting of local states $\{LS_1^1, LS_2^3, LS_3^3, LS_4^2\}$ is **inconsistent** because the state of p_2 has recorded the receipt of message m_{12}, however, the state of p_1 has not recorded its send.
- On the contrary, a global state GS_2 consisting of local states $\{LS_1^2, LS_2^4, LS_3^4, LS_4^2\}$ is **consistent**; all the channels are empty except c_{21} that contains message m_{21}.

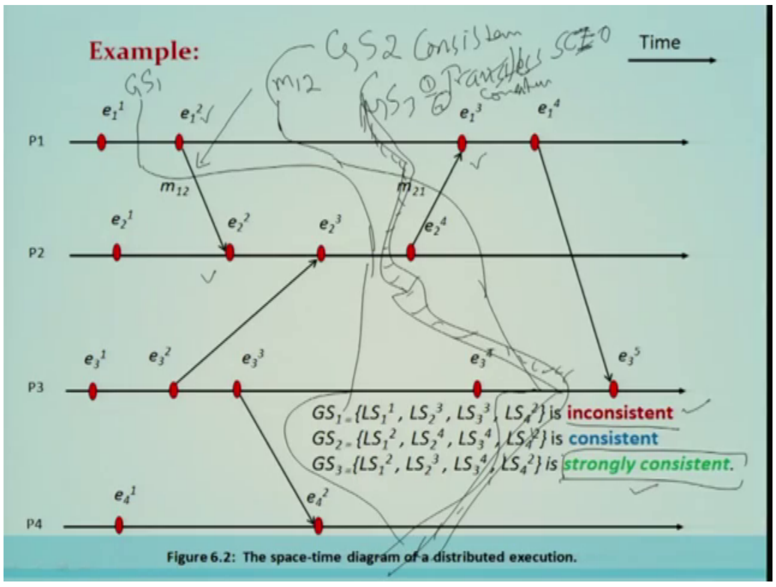
$LS_1^1 - e_1^1$

$\boxed{LS_i^x \Rightarrow e_i^x}$

So, we have seen the global state definitions. So, global state of a distributed system execution you can see in figure 6.2 which is given ahead before that we have to see that the global state GS 1 consisting of a locally states LS 1 of 1, LS 2 of 3, LS 3 of 3, LS 4 of 2. Let me explain you what does this means. So, LS 1 of 1 indicates that all the event e 1 up to one that has been covered in LS 1. So, in general we have to say that LS i of x is nothing, but set of all the events of i which has executed or progressed up to x of ed event. So, this particular notation is basically referenced over here.

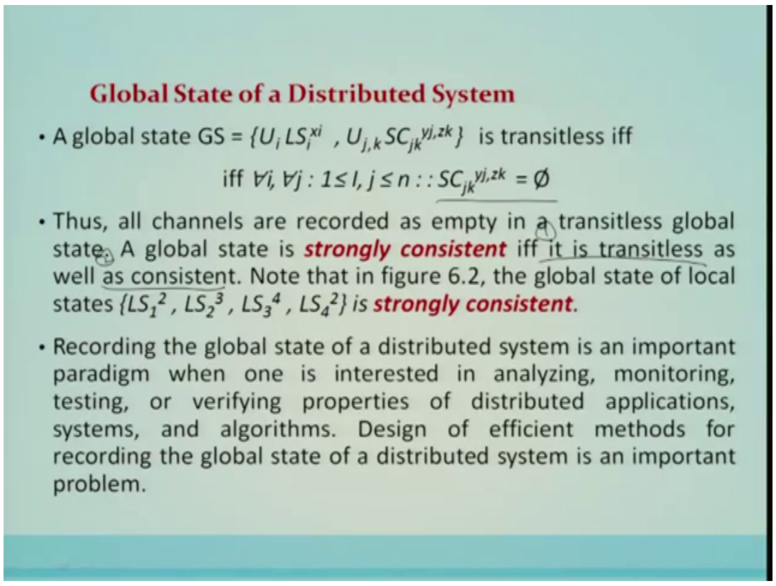
So, this set of global state GS 1 which comprises of all these set of local state is inconsistent because here the state of p 2 has recorded the received of a message m 1 2; however, the state of the p 1 has not recorded it is send of a message.

(Refer Slide Time: 08:22)


Example:
Time
GS2 Consistent SC≠0
GS1
m12
GS3 ① Transitless ② Consistent
P1
P2
P3
P4
e_1^1
e_1^2
e_1^3
e_1^4
m_{12}
m_{21}
e_2^1
e_2^2
e_2^3
e_2^4
e_3^1
e_3^2
e_3^3
e_3^4
e_3^5
e_4^1
e_4^2
$GS_1 = \{LS_1^1, LS_2^3, LS_3^3, LS_4^2\}$ is inconsistent
$GS_2 = \{LS_1^2, LS_2^4, LS_3^4, LS_4^2\}$ is consistent
$GS_3 = \{LS_1^2, LS_2^3, LS_3^4, LS_4^2\}$ is strongly consistent.
Figure 6.2: The space-time diagram of a distributed execution.

So, let us see this in the diagram. So, in the red GS 1 is shown as inconsistent. So, LS 1 of 1 will start over here LS 2 of 3; that means will go like this and LS 3 of 3 will come over here and LS 2 of 4 will basically come over here. So, you can see that this particular message m 1 2 is causing this particular global state is inconsistent why because in this global state 1 GS 1 it is called the received of a message is recorded, but send of a message is not recorded. So, it is an inconsistent global state example.

(Refer Slide Time: 09:23)



Global State of a Distributed System

- A global state GS = $\{U_i LS_i^{xi}, U_{j,k} SC_{jk}^{yj,zk}\}$ is transitless iff

  iff $\forall i, \forall j : 1 \le i, j \le n :: SC_{jk}^{yj,zk} = \emptyset$

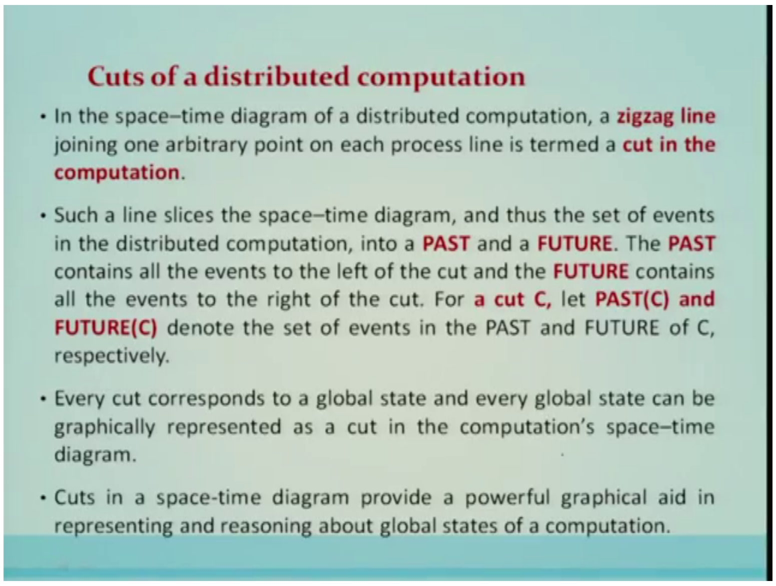
- Thus, all channels are recorded as empty in a transitless global state. A global state is ***strongly consistent*** iff it is transitless as well as consistent. Note that in figure 6.2, the global state of local states $\{LS_1^2, LS_2^3, LS_3^4, LS_4^2\}$ is ***strongly consistent***.
- Recording the global state of a distributed system is an important paradigm when one is interested in analyzing, monitoring, testing, or verifying properties of distributed applications, systems, and algorithms. Design of efficient methods for recording the global state of a distributed system is an important problem.

On contrary the global state GS 2 consisting of local states is consistent all the channels are empty here except C 2 1 that contains m 2 1 that we will see once we come to that figure. Now global state of a distributed system a global state GS is nothing, but a union of all local states and union of all general states this is transitless if and only if the state of the channel all the state of channels are empty; that means, no message is in transit. So, that global state is called transitless thus all channels are recorded as empty in the transitless global state a global state is strongly consistent if it is transitless as well as the consistent which definition we have seen.

Note that in figure 6.2 the global state of a local states this is strongly consistent let us see in this example. So, GS 2 comprises of LS 1 2, LS 1 2 is here and LS 2 4, LS 2 4 will come over here LS 3 4 will come over here and LS 2 4 will again come over here. So, this is GS 2 global state 2 this is a consistent global state why because there is a message which is crossing this particular a global state; that means, the message who send is recorded, but received is not recorded. So, this is an example or this is under the definition of a consistent global state strongly consistent global state LS 1, LS 1 of 2 will be here LS 2 of 3 will be here, then LS 3 of 4 will be here, then LS 2 of 4 will be here.

So, this particular example does not see any message which is crossing this particular line or a global state. Hence all the channels when this global state GS 3 is recorded they are transitless; that means, the state of channels they are all empty and it is a consistent also why because there is no message which is crossing hence it is called strongly consistent global state. So, recording the global state of a distributed system is an important paradigm when one is interest in analyzing monitoring testing verifying properties of a distributed applications and algorithms the design of efficient methods for a recording the global state of a distributed system is an important problem.

(Refer Slide Time: 12:57)



Cuts of a distributed computation

- In the space–time diagram of a distributed computation, a **zigzag line** joining one arbitrary point on each process line is termed a **cut in the computation**.
- Such a line slices the space–time diagram, and thus the set of events in the distributed computation, into a **PAST** and a **FUTURE**. The **PAST** contains all the events to the left of the cut and the **FUTURE** contains all the events to the right of the cut. For **a cut C,** let **PAST(C) and FUTURE(C)** denote the set of events in the PAST and FUTURE of C, respectively.
- Every cut corresponds to a global state and every global state can be graphically represented as a cut in the computation's space–time diagram.
- Cuts in a space-time diagram provide a powerful graphical aid in representing and reasoning about global states of a computation.

Now, cuts of a distributed system in a space time diagram of a distributed computation zigzag line joining an arbitrary point on each process line is termed as a cut in a distributed system. Such a line slices by space time diagram and thus the set of event in a distributed system is basically partitioned into the PAST events and into the FUTURE events, past event contains all the events to the left of the cut and future event contains all the event to the right of the cut for a cut C let past C and future C denote the set of events in the past and future of C respectively.

So, every cut corresponds to a global state and every global state can be graphically represented as a cut in a computations space time diagram cut in a space time diagram provides a powerful graphical aid in representing and reasoning about the global state of a computation.

(Refer Slide Time: 13:59)

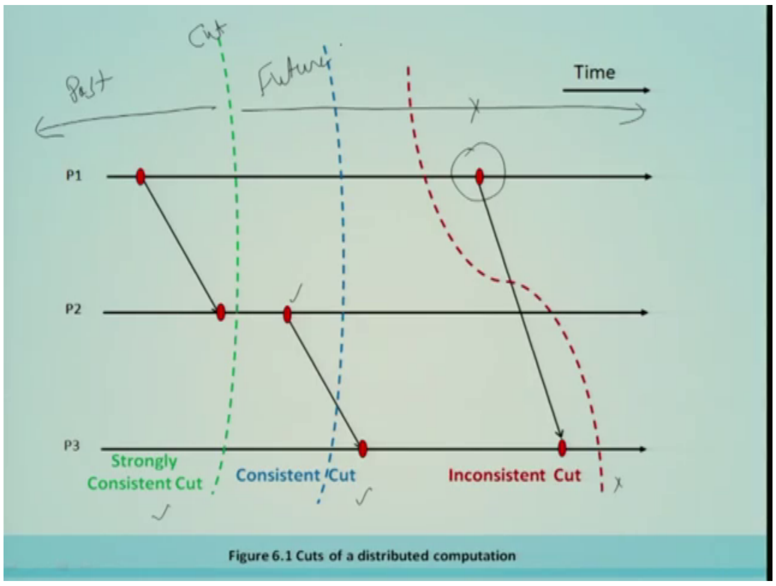


Figure 6.1 Cuts of a distributed computation

So, this is an example of the cut in a distributed system. So, this cut basically is a strongly consistent cut why because there is no message crossing it and all the channels are transitless it is a consistent and a transitless is called strongly consistent. Consistent cut where the message whose send is recorded within the cut, but the received is not record is called consistent cut inconsistent inconsistent cut says that the future event is crossing into the past event and this is not correct as far as the distributed system is concerned that is why this cut is called inconsistent cut.

So, whenever there is a cut on the left side will become the past event and on the right side becomes the future event. So, a space time diagram the cut divides into 2 parts the set of events which are happening in the past and set of events which are happening in the future and this is an important to the reasoning of a distributed system.

(Refer Slide Time: 15:15)

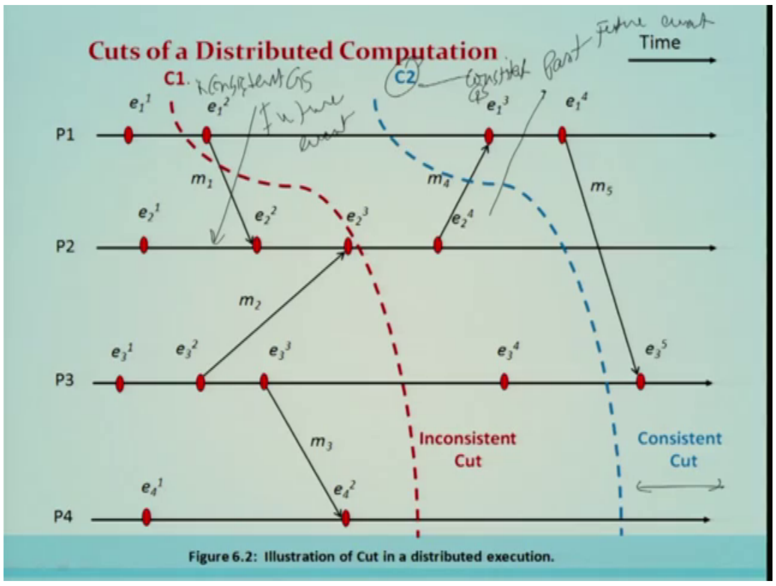
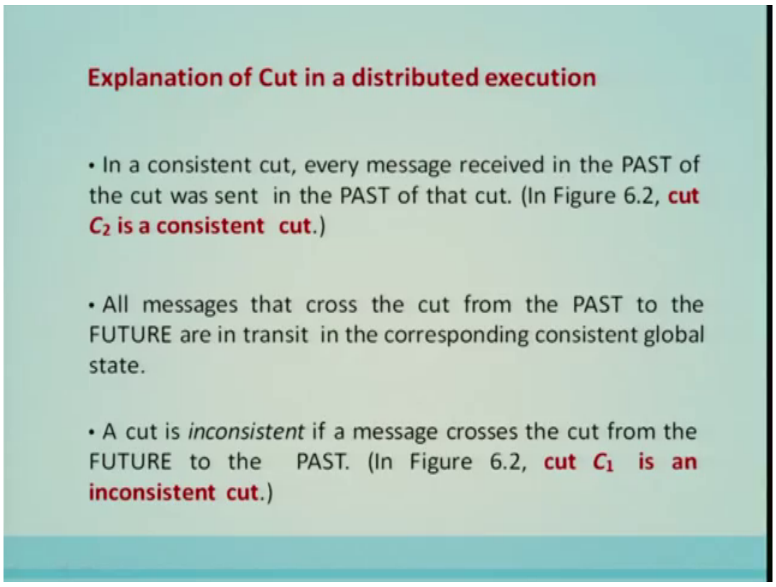


Figure 6.2: Illustration of Cut in a distributed execution.

So, these examples are self explanatory now after so much of discussions. So, this is an inconsistent cut why because a feature event is crossing to the past event.

Similarly, this event this is a consistent cut why because only the past event is crossing the future event. Hence this is a consistent global state defined by this particular cut this is inconsistent global state which is defined by this cut C 1. Explanation of the cut in a distributed system in a consistent cut every message received in the past of a of the cut was sent in the past of that cut that I have already explained.

(Refer Slide Time: 16:12)


Explanation of Cut in a distributed execution

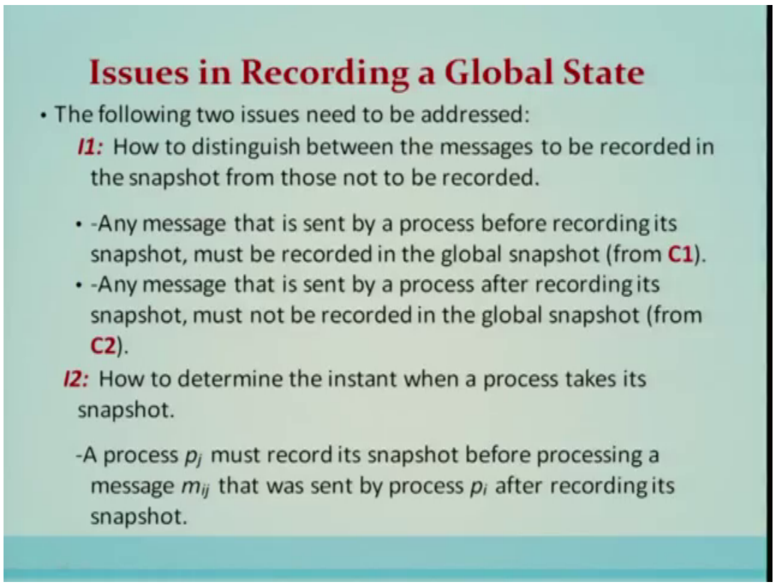
- In a consistent cut, every message received in the PAST of the cut was sent in the PAST of that cut. (In Figure 6.2, **cut C_2 is a consistent cut**.)

- All messages that cross the cut from the PAST to the FUTURE are in transit in the corresponding consistent global state.

- A cut is *inconsistent* if a message crosses the cut from the FUTURE to the PAST. (In Figure 6.2, **cut C_1 is an inconsistent cut**.)

So, all the messages that cross the cut from past to the future are in transit in the corresponding consistent global state a cut is inconsistent if a message crossing the cut from future to the past.

(Refer Slide Time: 16:43)



Issues in Recording a Global State

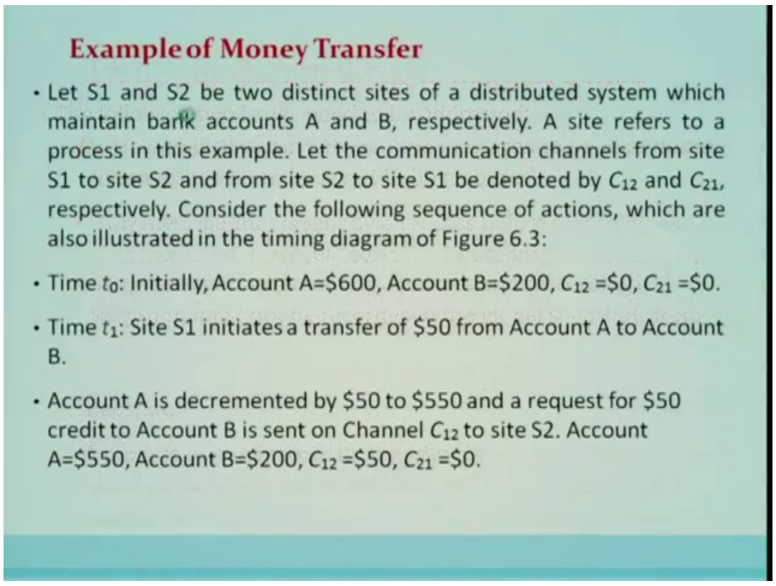
- The following two issues need to be addressed:
  - ***I1:*** How to distinguish between the messages to be recorded in the snapshot from those not to be recorded.
    - -Any message that is sent by a process before recording its snapshot, must be recorded in the global snapshot (from **C1**).
    - -Any message that is sent by a process after recording its snapshot, must not be recorded in the global snapshot (from **C2**).
  - ***I2:*** How to determine the instant when a process takes its snapshot.

    -A process p_j must record its snapshot before processing a message m_{ij} that was sent by process p_i after recording its snapshot.

Issues in a global state in a recording of a global state the following 2 issues need to be addressed issue number one that is I 1 how to distinguish between the messages to be recorded in a snapshot from those not to be recorded. So, any message that is sent by a process before recording it is snapshot must be recorded in a global snapshot and this is the condition number one any message that is send by a process after recording it is snapshot must not be recorded in a global snapshot and this is the condition number 2 of the issue number 1.

Now, issue number 2 says that how to determine the instant when a process takes it snapshot a process pj must record it is snapshot before processing a message mij that was sent by a process i after recording it is snapshot.

(Refer Slide Time: 17:51)



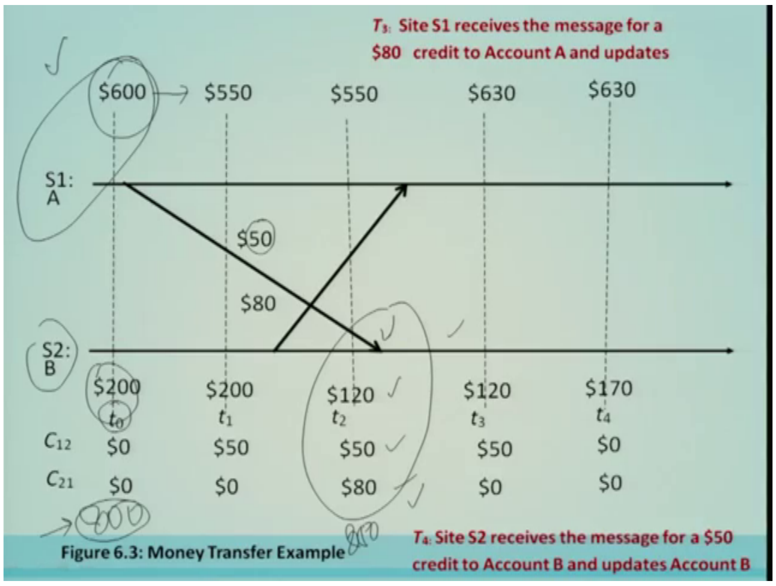
Example of Money Transfer

- Let S1 and S2 be two distinct sites of a distributed system which maintain bank accounts A and B, respectively. A site refers to a process in this example. Let the communication channels from site S1 to site S2 and from site S2 to site S1 be denoted by C_{12} and C_{21}, respectively. Consider the following sequence of actions, which are also illustrated in the timing diagram of Figure 6.3:
- Time t_0: Initially, Account A=\$600, Account B=\$200, C_{12} =\$0, C_{21} =\$0.
- Time t_1: Site S1 initiates a transfer of \$50 from Account A to Account B.
- Account A is decremented by \$50 to \$550 and a request for \$50 credit to Account B is sent on Channel C_{12} to site S2. Account A=\$550, Account B=\$200, C_{12} =\$50, C_{21} =\$0.

So, now we are going to see an example of a money transfer and this will give the motivation why the global state recording is not trivial. Let S 1 and S 2 be the 2 distinct sites of a distributed system which maintains a bank account AB respectively a site refers to a process in this example let the communication channel from site 1 to site 2 from site 2 to site 1 denoted by C 1 2 and C 2 1 respectively.

Consider the following sequence of actions which are also illustrated in the figure 6.3 time t 0 initially the account of a was 600 and account of B was 200 C 1 to channel was 0 empty and C 2 1 is also empty, in time t 1 S 1 initiates the transfer of 50 from account a to account B, account a is decremented by 50 to 550 and the request for 50 credit to the account B is sent on channel C 1 2 to C to site S 2 account a 55 and account B 200 C 1 2 50 and C 2 1 is 0.

(Refer Slide Time: 19:15)



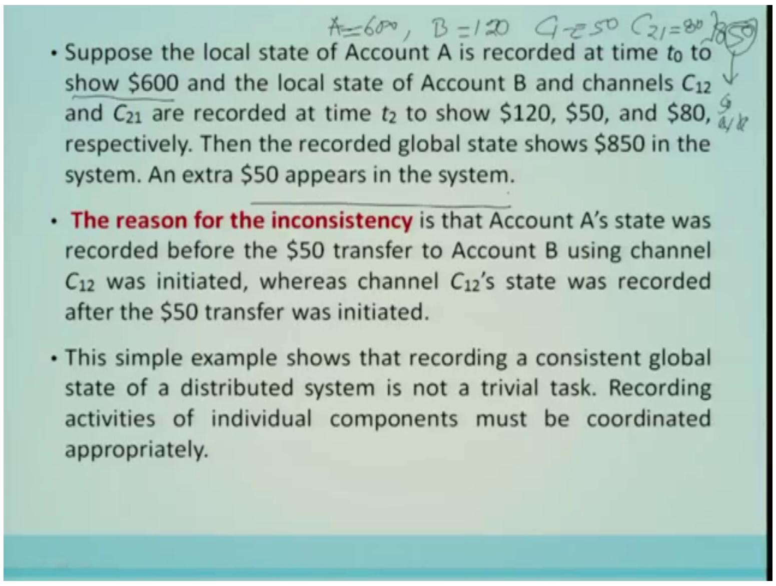
T_3: Site S1 receives the message for a $80 credit to Account A and updates
$600
$550
$550
$630
$630
S1: A
$50
$80
S2: B
$200
$200
$120
$120
$170
t_0
t_1
t_2
t_3
t_4
C_{12} $0 $50 $50 $50 $0
C_{21} $0 $0 $80 $0 $0
T_4: Site S2 receives the message for a $50 credit to Account B and updates Account B

Figure 6.3: Money Transfer Example

So, let us see this complete scenario of transaction between site S 1 and S 2 in this particular figure 6.3 that is money transfer example.

So, here you can see that at time t 0 site S 1 having a account of a is having 600 at that time after time t 0 it has sent a request to transfer 50 from a to B, just after sending this request the amount of a will be decremented afterwards sending this request and this request will reach here after t 2 time at this point instant. Now at this point t 2 time after t 2 time the scenario will be like this at the S 2 now you can see that site before time t 3 site S ones transfer of the message was received and it was being updated.

(Refer Slide Time: 20:48)



$A=600,\ B=120\ C_{12}=50\ C_{21}=80\ \}\ 850$

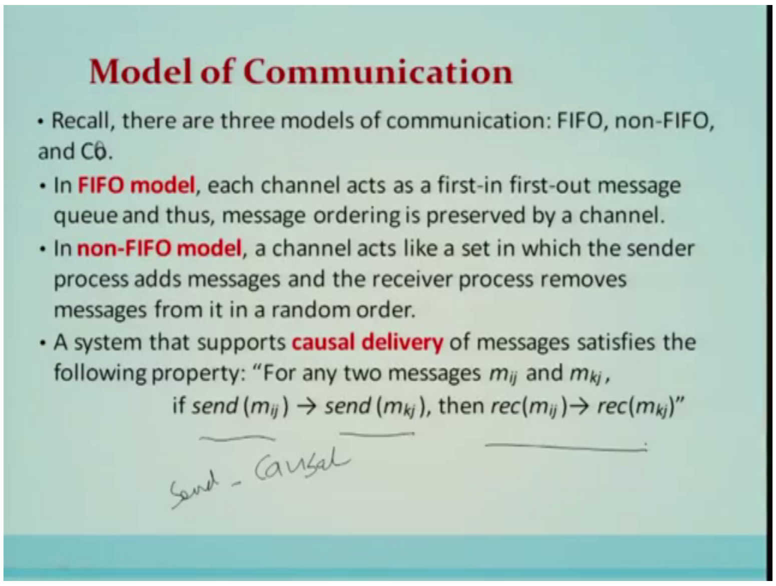
- Suppose the local state of Account A is recorded at time t_0 to show \$600 and the local state of Account B and channels C_{12} and C_{21} are recorded at time t_2 to show \$120, \$50, and \$80, respectively. Then the recorded global state shows \$850 in the system. An extra \$50 appears in the system.
- **The reason for the inconsistency** is that Account A's state was recorded before the \$50 transfer to Account B using channel C_{12} was initiated, whereas channel C_{12}'s state was recorded after the \$50 transfer was initiated.
- This simple example shows that recording a consistent global state of a distributed system is not a trivial task. Recording activities of individual components must be coordinated appropriately.

Now, as for s a time t 2 site S 2 receives the message to transfer 50 to credit this particular account. So, now, we can see here if let us say that the local state of a is recorded a time t 0 and it shows that a is 600 and the then the local state of B is recorded and that shows that, B is having 120 and the channel state is C 1 2 is 50 and C 2 1 is 80. So, this if you sum all the money this will become 6 how much 850; that means, 50 extra.

So, 50 extra will appear in the system. So, you can see here the initially both are having a plus B is equal to 800 to begin with, but this particular message to transfer 50 from A to B is in the transit.

So, the global state this is recorded and then at time t 2 this is recorded this is recorded in the sense 120 is basically the amount of B and 50 and 80. So, if you sum them it will become 850; that means, 50 will be coming extra. So, this kind of recording without any coordination is inconsistent the reason of inconsistency is that account as state was recorded before 50 transfer to b using channel C 1 2 was initiated whereas, channel C 1 twos state was recorded after 50 dollar was initiated this simple example shows that recording of consistent global state of a distributed system is not trivial recording activities of individual components must be coordinated appropriately; that means, it requires a particular coordination of time and the model of communication.

(Refer Slide Time: 23:15)



Model of Communication

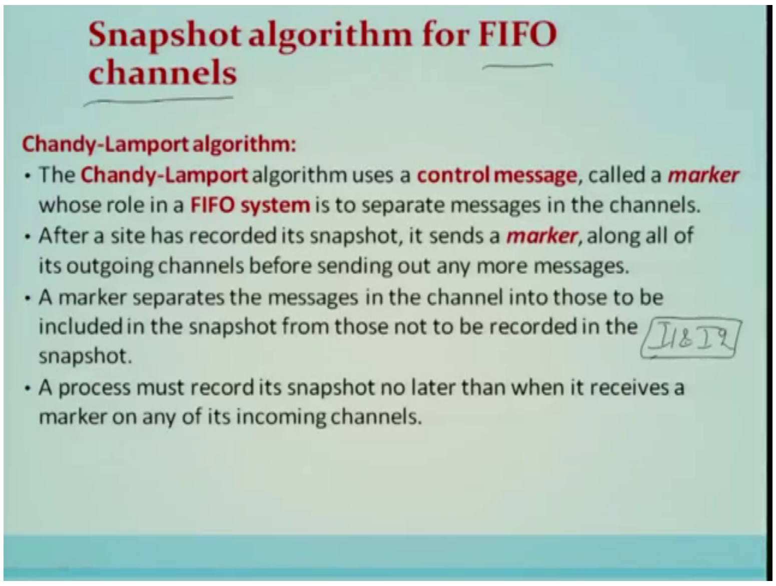
- Recall, there are three models of communication: FIFO, non-FIFO, and CO.
- In **FIFO model**, each channel acts as a first-in first-out message queue and thus, message ordering is preserved by a channel.
- In **non-FIFO model**, a channel acts like a set in which the sender process adds messages and the receiver process removes messages from it in a random order.
- A system that supports **causal delivery** of messages satisfies the following property: "For any two messages m_{ij} and m_{kj}, if $send(m_{ij}) \rightarrow send(m_{kj})$, then $rec(m_{ij}) \rightarrow rec(m_{kj})$"

Send - Causal

So, model of communication is very much going to affect the design of the global snapshot recording algorithm. So, that is why we are again stressing upon or we are looking back again the same model of communication. Recall that there are 3 different model of communication in a distributed system which is assumed the first one is called FIFO the other one is called non-FIFO and casual order co in a FIFO model each channel acts as first in first out message queue and thus the message ordering is preserved by the channel in non-FIFO model a channel acts like a set in which the sender process adds a message and receiver process removes message from it in a random order.

The casual order model the system supports casual delivery of the message satisfies the following property for any 2 messages mij and mkj which are going to the same destination it send of mij and it precedes send of mkj; that means, both the sends both the message they have the causal relation in the send message then this particular relation will be respected when they are being delivered that is received of mij is also going to proceed before received of mkj then it is called casual order model of communication.

(Refer Slide Time: 24:58)



Snapshot algorithm for FIFO channels

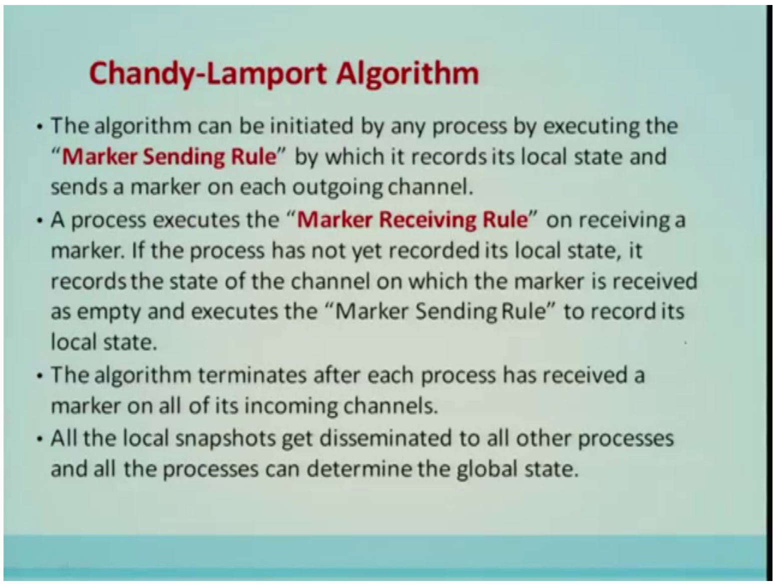
Chandy-Lamport algorithm:

- The **Chandy-Lamport** algorithm uses a **control message**, called a ***marker*** whose role in a **FIFO system** is to separate messages in the channels.
- After a site has recorded its snapshot, it sends a ***marker***, along all of its outgoing channels before sending out any more messages.
- A marker separates the messages in the channel into those to be included in the snapshot from those not to be recorded in the snapshot. $\boxed{I1 \& I2}$
- A process must record its snapshot no later than when it receives a marker on any of its incoming channels.

Now, we are going to see a snapshot algorithm which basically uses the FIFO channel. So, let me tell you that this is an assumption. So, this same algorithm for other 2 models that is non-FIFO and casual order will not work. So, it will assume that the model is FIFO. So, non-FIFO it will not work. So, Chandy-Lamport algorithm Chandy-Lamports algorithm uses a control message which is called a marker whose role in FIFO system is to separate the messages in the channel. So, after a site has recorded it is snapshot it sends a marker along all of it is outgoing channels before sending out any more message, a marker separates the message in the communication channel into those to be included in their snapshot from those not to be recorded in the snapshot as far as if you see the issue I 1 and issue I 2 which we have seen earlier about global snapshot.

So, the process must record it is a snapshot no later when it receives marker on it is communication channel and this is the issue number 2.

(Refer Slide Time: 26:21)



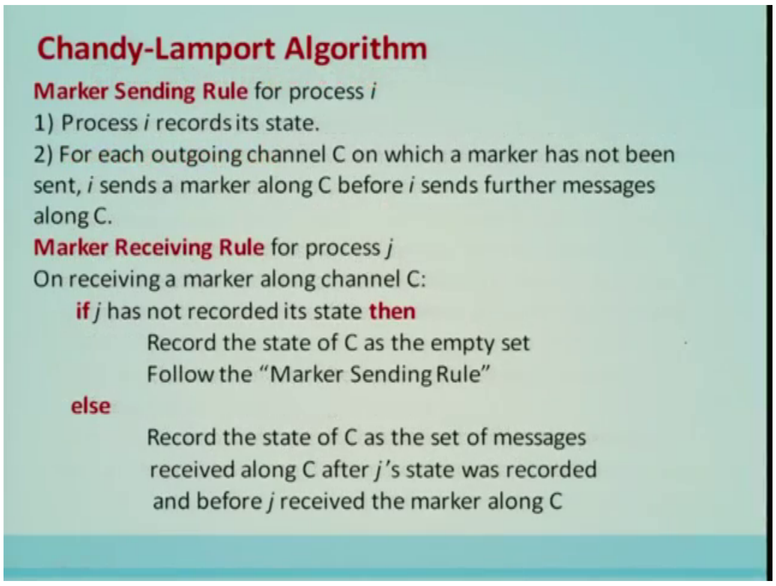
Chandy-Lamport Algorithm

- The algorithm can be initiated by any process by executing the "**Marker Sending Rule**" by which it records its local state and sends a marker on each outgoing channel.
- A process executes the "**Marker Receiving Rule**" on receiving a marker. If the process has not yet recorded its local state, it records the state of the channel on which the marker is received as empty and executes the "Marker Sending Rule" to record its local state.
- The algorithm terminates after each process has received a marker on all of its incoming channels.
- All the local snapshots get disseminated to all other processes and all the processes can determine the global state.

So, the algorithm can be initiated by any process by executing the marker sending rule by which it records it is local state and sends a marker to on each outgoing channel. A process executes marker receiving rule on receiving the marker if a process has not yet recorded it is local state it records the state on the channel on which the marker is received as empty and executes a marker sending rule to record it is local state.

The algorithm terminates after each process has received a marker on all it is incoming channel. So, all the local snapshot gets disseminated to all other processes and all the processes can determine the local state.

(Refer Slide Time: 27:10)



Chandy-Lamport Algorithm

Marker Sending Rule for process *i*

1) Process *i* records its state.

2) For each outgoing channel C on which a marker has not been sent, *i* sends a marker along C before *i* sends further messages along C.

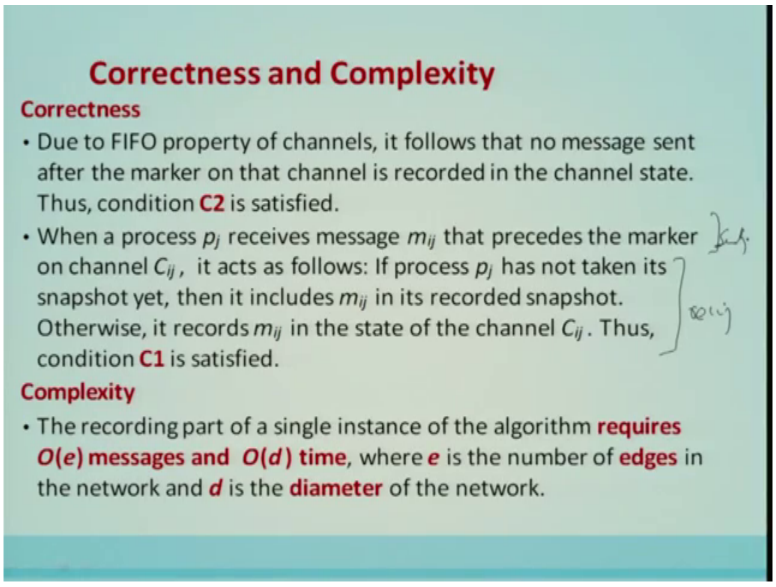
Marker Receiving Rule for process *j*

```
On receiving a marker along channel C:
    if j has not recorded its state then
            Record the state of C as the empty set
            Follow the "Marker Sending Rule"
    else
            Record the state of C as the set of messages
            received along C after j's state was recorded
            and before j received the marker along C
```

Let us see the algorithm in more detail the marker sending rule for a process i has 2 steps process i records it is state for each outgoing channel C on which a marker has not been sent i sends the marker along C before i sends further messages along C marker receiving rule for a process j on receiving a marker along channel C if j has not recorded it is state then record by state of C as empty follow the marker sending rule else record the state of C as a set of messages received along C after j's state was recorded and before j received the marker along the C.

So, the Chandy-Lamports algorithm has 2 different rules marker sending rule for a process i and marker receiving rule for process j, marker sending rule will initiate the process, i to record it is state and it will also send the marker on each outgoing channel. Similarly marker receiving rule for a process j has 2 conditions on receiving the marker along the channel C and if j has not recorded then it will record the state of a channel and also will record it is state of a process according to the marker sending rule, if the if process j has already recorded then it will record the state of a channel C as the set of messages received along C after j's state was recorded and before j receive the marker along C.

(Refer Slide Time: 28:56)



Correctness and Complexity

Correctness

- Due to FIFO property of channels, it follows that no message sent after the marker on that channel is recorded in the channel state. Thus, condition **C2** is satisfied.
- When a process p_j receives message m_{ij} that precedes the marker on channel C_{ij}, it acts as follows: If process p_j has not taken its snapshot yet, then it includes m_{ij} in its recorded snapshot. Otherwise, it records m_{ij} in the state of the channel C_{ij}. Thus, condition **C1** is satisfied. [illegible] [illegible]

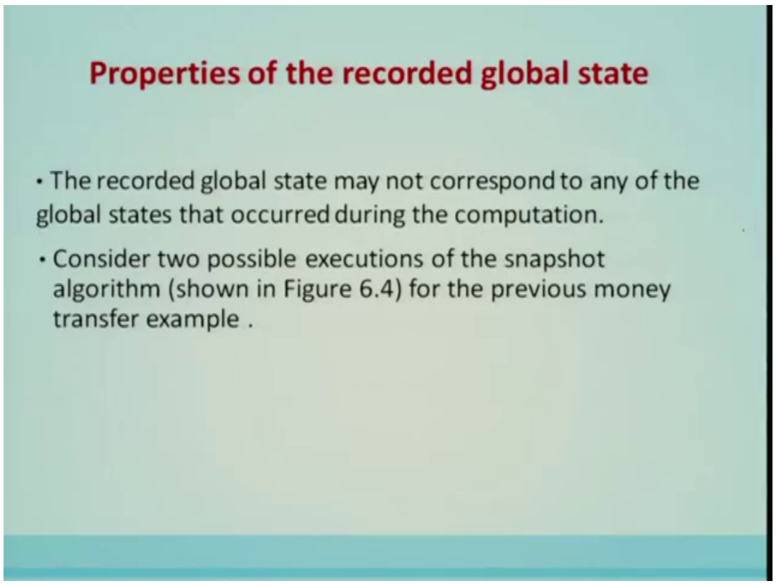
Complexity

- The recording part of a single instance of the algorithm **requires $O(e)$ messages and $O(d)$ time**, where e is the number of **edges** in the network and d is the **diameter** of the network.

So, first we have to go and see the correctness and complexity of this algorithm and then we will explain you through an example. So, correctness due to the FIFO property of the channel this is very important that it assumes a FIFO property of a communication channel it follows that no message sent after the marker on that channel is recorded in the channel state thus condition C 2 is satisfied. When a process pj receives the message mij that precedes the marker on the channel Cij it acts as follows if the process j has not taken it is snapshot yet then it includes mij in it is recorded snapshot otherwise it records mij in the state of a channel Cij thus condition C 1 satisfies.

So, this is according to the marker receiving rule and this is basically the marker sending rule. So, this shows the correctness of the snapshot recording and which is a consistent global snapshot or global state which is recorded by this algorithm, the complexity the recording part of a single instance of a algorithm requires of the order e messages where e is the number of edges and the time complexities of the order d where d is the diameter of the network.

(Refer Slide Time: 30:33)



Properties of the recorded global state

- The recorded global state may not correspond to any of the global states that occurred during the computation.
- Consider two possible executions of the snapshot algorithm (shown in Figure 6.4) for the previous money transfer example .

Now, let us see the properties of the recorded global state the recorded global state may not correspond to any of the global state that occurred during computations. So, consider 2 possible execution of a snapshot algorithm which is shown in the next figure for the previous money transfer.

(Refer Slide Time: 30:52)

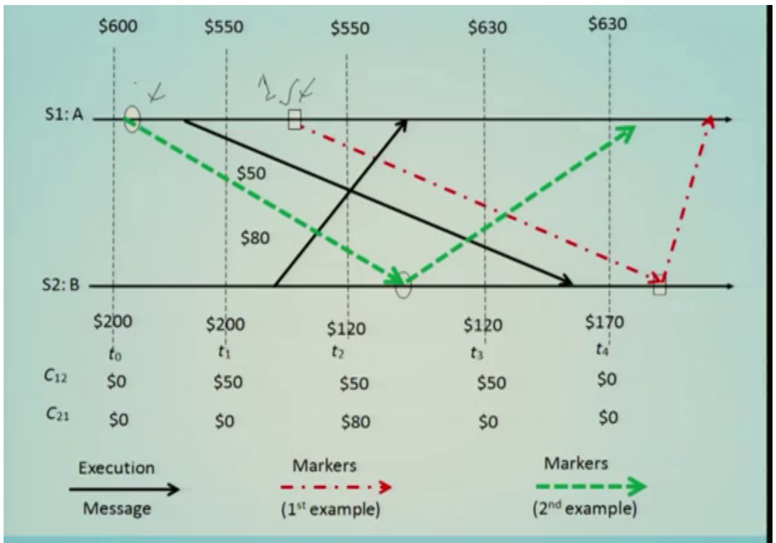
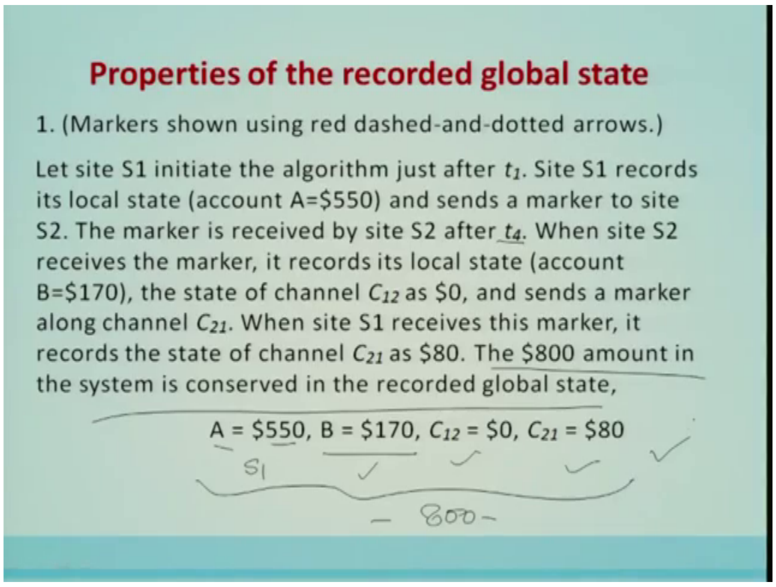


Figure 6.4: Timing diagram of two possible executions of the banking example

So, here in this particular example we will see 2 different cases of the marker or the snapshot initiation which is shown by the circle in the first one and the square is another initiation of a the same global snapshot recording algorithm which is shown by the red dots the first snapshot.

(Refer Slide Time: 31:32)



Properties of the recorded global state

1. (Markers shown using red dashed-and-dotted arrows.)

Let site S1 initiate the algorithm just after t_1. Site S1 records its local state (account A=\$550) and sends a marker to site S2. The marker is received by site S2 after t_4. When site S2 receives the marker, it records its local state (account B=\$170), the state of channel C_{12} as \$0, and sends a marker along channel C_{21}. When site S1 receives this marker, it records the state of channel C_{21} as \$80. The \$800 amount in the system is conserved in the recorded global state,

$$\underbrace{\underset{S_1}{A = \$550}, \underset{\checkmark}{B = \$170}, \underset{\checkmark}{C_{12} = \$0}, \underset{\checkmark}{C_{21} = \$80}}_{-\ 800\ -} \checkmark$$

Shows the green dots we will see that in both the cases it will record a consistent global state in that same previous example.

So, first we will see this example this particular example we will see first and this will see next. So, site S 1 initiates the algorithm just after t 1 site S 1 records it is state as 550 over here this is site S 1 records and sends a marker to the site S 2 the marker is received by the site S 2 after t 4 when site S 2 receives the marker it records it is local state that is because it is receiving the marker for the first time. So, in the marker receiving rule the first condition will be satisfy. So, it will record it is state as 170 and the state of a channel C 1 2 will be empty in that case.

And then it will send the marker sending rule along the channel C 1 now when state S 1 receives this marker which is sent by site S 2 it records by state of a channel C 2 1 as 80 why because it has already recorded it is state. So, it will go for the second option in the marker receiving rule. So, only the state of a channel is recorded and which is shown as 80. So, if you sum them it will be 800 and that is the money which the system has began. So, hence the 800 amount in the system is conserved in a recorded global state by this particular algorithm.

(Refer Slide Time: 33:29)

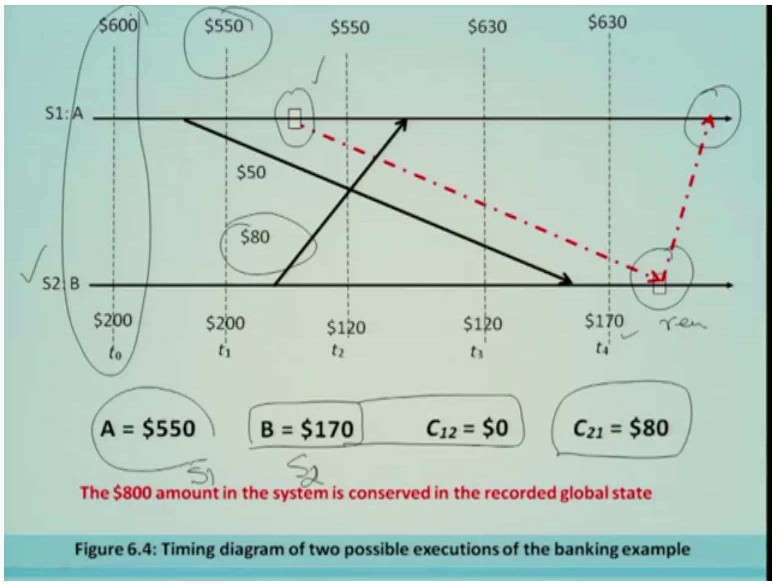
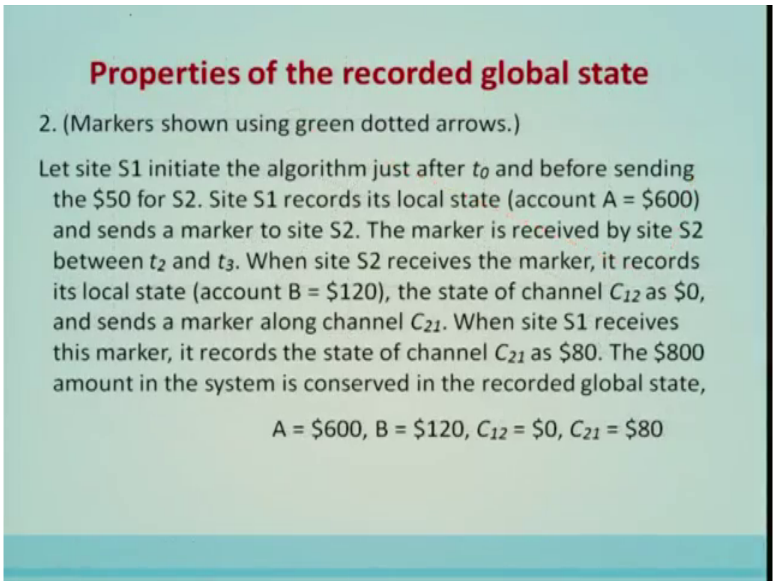


Figure 6.4: Timing diagram of two possible executions of the banking example

In second example and this is shown over here that I have explained. So, here the global state will be recorded that is a will be recording 550 over here by site S 1 before sending the marker sending rule and when the market is received. So, it will initiate site S 2 will initiate the marker receiving rule. So, it is recording for the first time. So, it will be recorded by state of B as 170 after time t 4. Now then it will also record it is state of channel C 1 2 as 0 and then it will send the marker to the site S 1, site S 1 on receiving this particular marker will initiate the marker receiving rule the second condition because it has already recorded in the snapshot or it has already recorded in state.

So, now it will only record the state of this particular channel and that is nothing, but C 2 1 is basically having the 80 dollar this particular portion will be captured over here. So, if you sum them it will become 800 that is nothing, but the start condition of the system.

(Refer Slide Time: 35:05)



Properties of the recorded global state

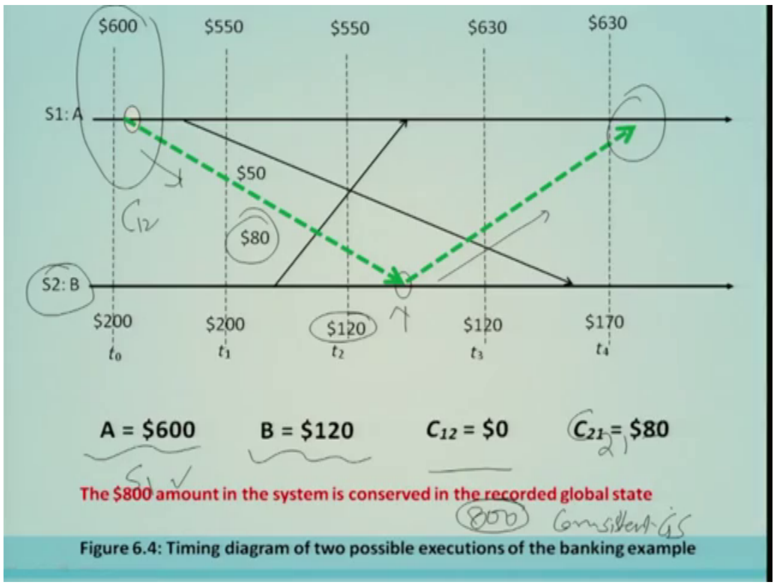
2. (Markers shown using green dotted arrows.)

Let site S1 initiate the algorithm just after t_0 and before sending the \$50 for S2. Site S1 records its local state (account A = \$600) and sends a marker to site S2. The marker is received by site S2 between t_2 and t_3. When site S2 receives the marker, it records its local state (account B = \$120), the state of channel C_{12} as \$0, and sends a marker along channel C_{21}. When site S1 receives this marker, it records the state of channel C_{21} as \$80. The \$800 amount in the system is conserved in the recorded global state,

A = \$600, B = \$120, C_{12} = \$0, C_{21} = \$80

Hence the system basically records correctly the consistent global state another example we are going to see when the markers are shown in the green dotted arrows.

(Refer Slide Time: 35:18)



A = \$600   B = \$120 C_{12} = \$0 C_{21} = \$80

The \$800 amount in the system is conserved in the recorded global state

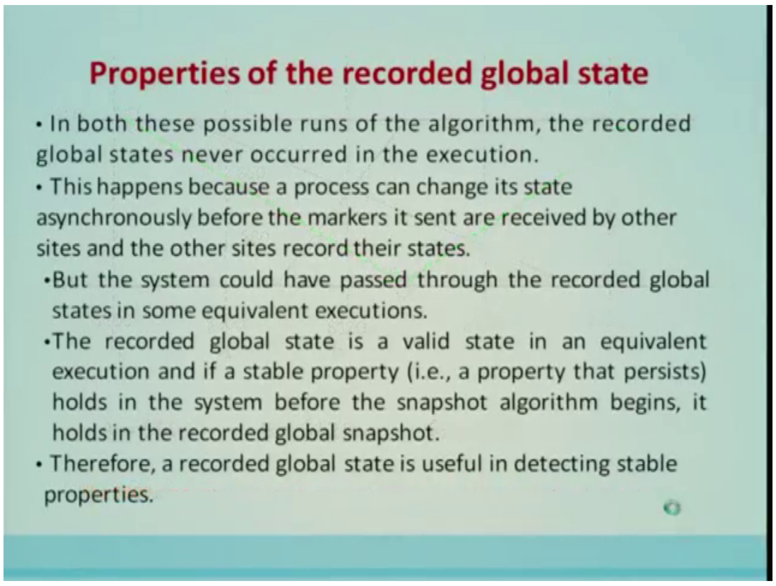
800 Consistent GS

Figure 6.4: Timing diagram of two possible executions of the banking example

In this example I will explain you the working of a Chandy-Lamports algorithm. Now here you can see that before sending the message the channel **state** is recorded at S 1. So, marker sending rule will record it is channel will is records it is state and then it will send the marker sending rule marker will be sent on the outgoing channel that is C 1 2.

Now, when the marker is received by state 2 then it will record it is state S 2 has not yet recorded it is state. So, S 2 is going to record it is state as 120. So, 120 is the current value of S 2 of B and it will also record it is channel as C 1 2 as empty and then it will send the marker again back to 1. The marker will be received by site S 1 at this end since it has already recorded it is state. So, it will only record the channel state of 2 1 what it will record in the channel state. So, it will record the channel state as this amount that is 80 now if you sum them it will become 800 and this is a consistent global state which is recorded by this particular algorithm and the 2 examples have clearly shown this particular thing.

(Refer Slide Time: 37:09)



Properties of the recorded global state

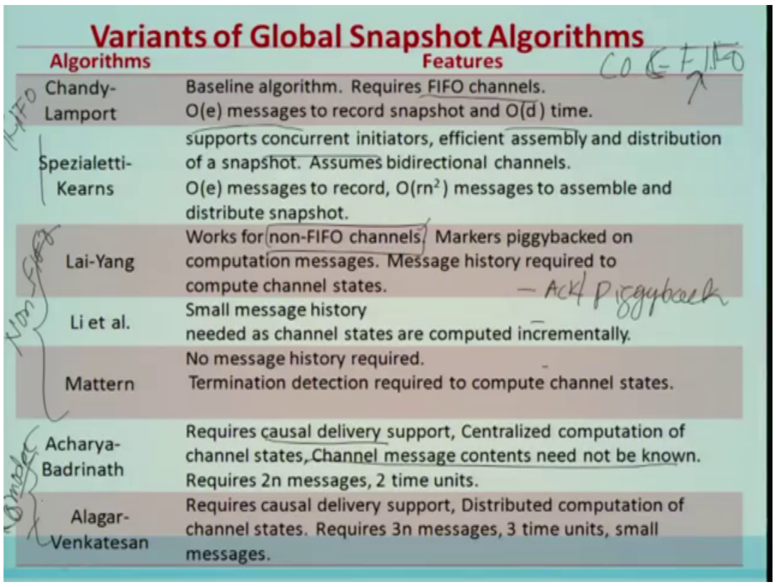
- In both these possible runs of the algorithm, the recorded global states never occurred in the execution.
- This happens because a process can change its state asynchronously before the markers it sent are received by other sites and the other sites record their states.
- But the system could have passed through the recorded global states in some equivalent executions.
- The recorded global state is a valid state in an equivalent execution and if a stable property (i.e., a property that persists) holds in the system before the snapshot algorithm begins, it holds in the recorded global snapshot.
- Therefore, a recorded global state is useful in detecting stable properties.

Now, properties of the recorded global state in both these possible runs of the algorithm the recorded global state never occurred in the execution that is in the distributed execution. This happens because the process can change it is state asynchronously before markers it sent are received by the other sites and the other sites record their states in the sense due to the asynchrony of the sender and the receiver processes.

So, the snapshot which is recorded over here may not coincide with the distributed execution which is taking place, but the system could have pass through the recorded global state in some equivalent executions the recorded global state is valid state in an equivalent execution and if the stable property that is a property that persist holds in the system before snapshot algorithm begins it holds in the recorded global snapshot.

Therefore, the recorded global state is useful in detecting the stable properties of a distributed system.

(Refer Slide Time: 38:15)



Variants of Global Snapshot Algorithms

CO ⊆ FIFO

Algorithms	Features
Chandy-Lamport	Baseline algorithm. Requires FIFO channels. O(e) messages to record snapshot and O(d) time.
Spezialetti-Kearns	supports concurrent initiators, efficient assembly and distribution of a snapshot. Assumes bidirectional channels. O(e) messages to record, O(rn^2) messages to assemble and distribute snapshot.
Lai-Yang	Works for non-FIFO channels. Markers piggybacked on computation messages. Message history required to compute channel states. — Ack Piggyback
Li et al.	Small message history needed as channel states are computed incrementally.
Mattern	No message history required. Termination detection required to compute channel states.
Acharya-Badrinath	Requires causal delivery support, Centralized computation of channel states, Channel message contents need not be known. Requires 2n messages, 2 time units.
Alagar-Venkatesan	Requires causal delivery support, Distributed computation of channel states. Requires 3n messages, 3 time units, small messages.

FIFO

Non FIFO

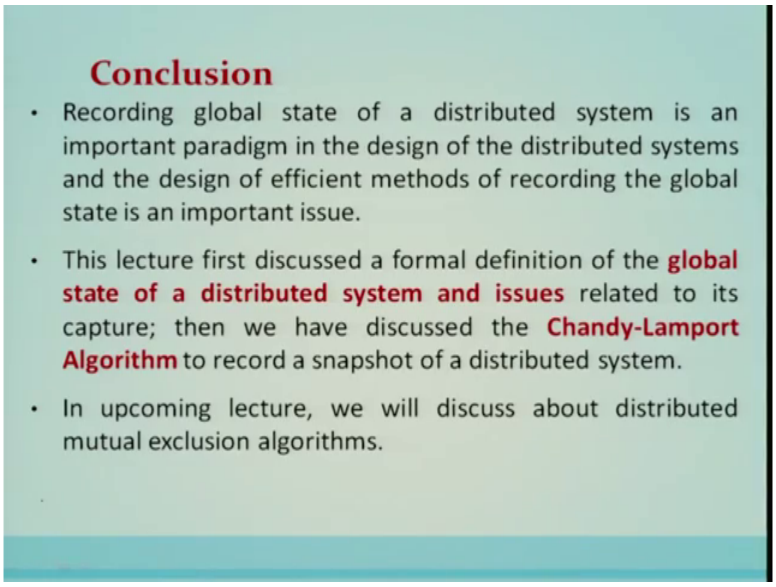
Now, we are going to see the variants of a global snapshot algorithms which are available we have seen these global snapshot algorithms as I told you that they are very sensitive to the communication channel model and the global snapshot algorithm given by Chandy-Lamport basically uses FIFO model of the channel. So, here it is also already written that the Chandy-Lamport algorithm requires the channel model to be the FIFO model. So, Chandy-Lamport algorithm uses the order e messages to record the snapshot and uses order t time. Now Spezialetti and Kearns they have improved this algorithm which supports the concurrent initiation. So, that is already following the FIFO model.

Now, then there are algorithms which uses non FIFO model and to use the non FIFO model it is not trivial it is very difficult to design the snapshot algorithm in a non FIFO model. So, Lai-Yang Li et al and Mattern there are 3 different algorithms are basically discussed in the literature. Now these algorithms basically require the concept which will either delay the message sending of the further messages till the messages which is sent by the particular process is received by the receiver. So, this delaying can be basically acknowledged using the acknowledged message or using the piggyback message can be used to synchronize or to know whether the messages which are sent is being delivered at the other end or not because it is a FIFO model cannot assume that the messages which are sent are being delivered.

So, basically there are various implementation various algorithms uses various techniques some are using the acknowledgments. So, delay and once the acknowledgement comes or the acknowledgement is being Piggybacked to deal with the asynchrony of the communication channels. Then we will see some algorithms which uses the causal order that is co model causal order is a basically proper subset of FIFO model. So, if you assume a causal order it will be even simpler than the algorithms which we have seen in the Chandy-Lamports; that means, the algorithms need not have to send the markers on all the outgoing channels. So, require the casual delivery support.

So, channel message contents need not to be known and hence this particular model if it is assumed then the global snapshot algorithm will become quite even simpler or easy. So, these are different variants if the variations are in the form of the communication model. So, we have already covered that; that means, how the Chandy-Lamport algorithm will be basically or improvements of these algorithms which we have seen.

(Refer Slide Time: 42:05)



Conclusion

- Recording global state of a distributed system is an important paradigm in the design of the distributed systems and the design of efficient methods of recording the global state is an important issue.
- This lecture first discussed a formal definition of the **global state of a distributed system and issues** related to its capture; then we have discussed the **Chandy-Lamport Algorithm** to record a snapshot of a distributed system.
- In upcoming lecture, we will discuss about distributed mutual exclusion algorithms.

Conclusion recording global state of a distributed system is an important paradigm in the design of distributed system and the design of efficient methods of recording global state is an important issue. This lecture first discussed the formal definition of a global state of a distributed system and then basically we have discussed the Chandy-Lamport algorithm to record the snapshot of a distributed system in upcoming lectures we will discuss about the distributed mutual exclusion algorithms.

Thank you.