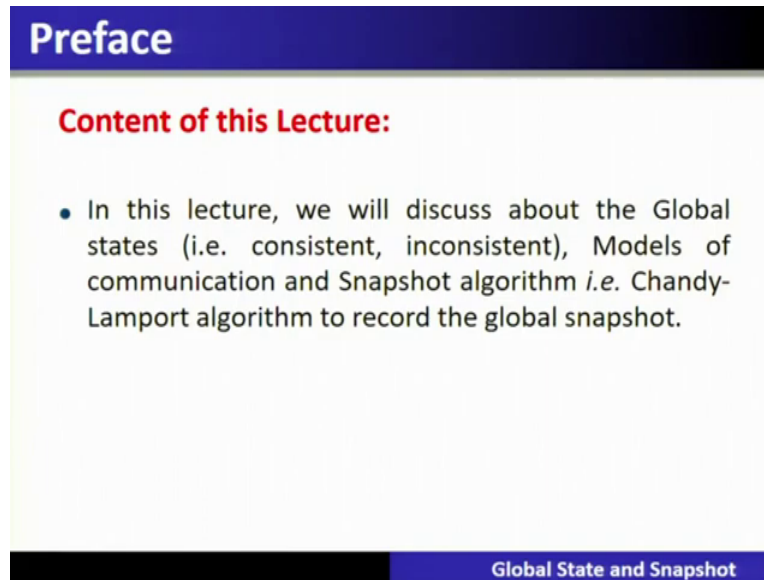


**Foundation of Cloud IoT Edge ML**  
**Professor Rajiv Misra**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Patna**  
**Lecture 15**  
**Global State and Snapshot Recording Algorithms**

(Refer Slide Time: 0:22)



**Preface**

**Content of this Lecture:**

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


Global State and Snapshot

I am Doctor Rajiv Misra from IIT, Patna. The topic of this lecture is global state and a snapshot recording algorithms. So, content of this lecture this lecture will discuss about the Global states. There are 2 types of global state, consistent and inconsistent global states. And we will also see the model of communication and the snapshot algorithm specifically Chandy Lamport's algorithm to record the global snapshot.


(Refer Slide Time: 0:44)

## Snapshots

Here's Snapshot: Collect at a place



**Distributed Snapshot**  
How do you calculate a "global snapshot" in this distributed system?  
What does a "global snapshot" even mean?



→ 105 ✓

Global State and Snapshot

So, what is a snapshot? So, you can see in the picture that all the people they collect at one place and a photographer take a snapshot. So, this is called a snapshot but if what if the people do not collect at one place and still you want to take the snapshot it is called a distributed snapshot. So, how do you calculate that global snapshot in this distributed environment.

So, take this environment the distributed system where these particular processes are running at different places around the globe and you want to take a global snapshot in this environment that is called a distributed snapshot. So, that is not so easy so we are going to see this kind of topic. So, the use case will be for example if you are having the IoT devices which are deployed in a particular city so many number of devices and often they are connected with the edge and then it is connected through the cloud.

So, if you want to monitor or debug a particular instance or the events then you sometimes require the global snapshot if they are also having the actuators. So, this kind of use case in IoT that is applying global snapshot distributed snapshot is also one of the most important distributed system problem which is applicable in the IoT system.

(Refer Slide Time: 2:17)

**In the Cloud: Global Snapshot**

- In a cloud each application or service is running on multiple servers ✓
- Servers handling concurrent events and interacting with each other ✓
- The ability to obtain a “global photograph” or “Global Snapshot” of the system is important ✓
- Some uses of having a global picture of the system
  - **Checkpointing:** can restart distributed application on failure ✓
  - **Garbage collection of objects:** objects at servers that don't have any other objects (at any servers) with pointers to them ✓
  - **Deadlock detection:** Useful in database transaction systems ✓
  - **Termination of computation:** Useful in batch computing systems ✓

Global State and Snapshot

So, let us see as far as this edge data center or cloud data center IoT devices this all becomes a distributed system and how we are going to take a global snapshot. So, in the cloud each application or service is running on the multiple servers that we have seen in the previous lectures. So, the servers handling the concurrent events and often interacting with each other this is another part of the model.

So, the ability to obtain the global snapshot of this kind of distributed system is sometimes important for the application such as the check pointing that is you can restart the distributed system on the failure that means if let us say some device is failed so how do you restart that application which is enabled with the help of IoT devices.

Similarly, the garbage collection of the objects so objects is the server that do not have any objects within pointer that sometimes that is also garbage collection requires a global snapshot recording for that. Sometimes the deadlock detection which is useful in a database transaction systems and termination of the computation that also requires the global snapshot.

(Refer Slide Time: 3:43)

**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 and Snapshot

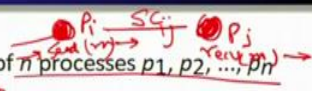
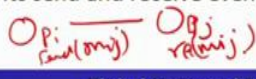
So, globally snapshot algorithm is the basis of many applications which is running in this kind of environment which is called a distributed system. So, how do you record the global state in this environment which is called a distributed system on the fly without stopping the current application which is running. So, that is most important part. So, lack of global why there is a problem what are the issues which this particular algorithm will need as a model.

So, in a distributed system there is a lack of globally shared memory we do not expect common or shared memory also there is a global there is no common clock and all the devices are running autonomously with their own clock. Similarly, the messages which are being exchanged their delays are also unpredictable. So, this makes this distributed system problem of global snapshot recording a non-trivial.

So, it is not like the people gather at one place and a photograph or takes a snapshot is quite non trivial. So, this lecture firstly finds some few preliminaries such as consistent global state and then how these particular what is the computation to be done to do this compute consistent in a distributed a snapshot recording then we will use an algorithm that is given by the Chandy Lamport algorithm to determine on the fly snapshot recording in this way.

(Refer Slide Time: 5:18)

### System Model

- The system consists of a collection of  $n$  processes  $p_1, p_2, \dots, p_n$  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. 

Global State and Snapshot

So, let us understand the system model in which this algorithm will be explained. So, the system consists of  $n$  different processes that are connected by the channel. So, here the use case may be that you have  $n$  IoT devices and these IoT devices are now connected in the form of a mesh network and this particular model is feeding into that also.

So, now, we also assume that there is no global shared memory out of these  $n$  systems and also the physical clocks and process can communicate only by the messages through the communication channel. So, if a process  $p_i$  and  $p_j$  if they want to be connected or they want to communicate using the messages then we assume a communication channel between process  $p_i$  to  $p_j$ .

And we state it as the state of the channel  $SC$  means state of the channel between  $i$  and  $j$ . So, the actions performed by these process or model as 3 different types. So, internal events means all the events which are happening inside this particular process called internal events and whenever this particular process will send a message called message send event and when the process is received then it is received of a event.

So, all the events are divided into 3 types one is called internal event which is happening in these inside these particular processes or when a process will send a message this will be the second event a third event when a process will receive a message. So, there are 3 different events to make the system model usable for this particular algorithm.

So, for a message  $m_{ij}$  that is being sent between the process  $i$  and process  $j$  called  $m_{ij}$  that is that is the message  $m_{ij}$  which will be sent by your process  $p_i$  send of  $m_{ij}$  denoted by  $send_{m_{ij}}$ . And similarly, when process  $j$  will receive this  $m_{ij}$  it is called the receiver of this particular message and the sender and receiver events will be denoting this message flow.

(Refer Slide Time: 7:54)

**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  $LS_j$ ,  $e \in LS_j$  iff  $e$  belongs to the sequence of events that have taken process  $p_i$  to state  $LS_j$ .
- For an event  $e$  and a process state  $LS_j$ ,  $e \notin LS_j$  iff  $e$  does not belong to the sequence of events that have taken process  $p_i$  to state  $LS_j$ .
- 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:**  $transit(LS_i, LS_j) = \{ m_{ij} \mid send(m_{ij}) \in LS_i \wedge rec(m_{ij}) \notin LS_j \}$

Global State and Snapshot

So, the system model another assumption is that at any instant the state of a process  $p_i$  will be designated by local state of  $i$ . So,  $LS_i$  of  $i$  means the local state at the process  $i$  is the result of the sequence of all the event executed by  $p_i$  by the process  $i$  up to that particular time instant. So, for an event  $e$  are the processes that  $LS_i$  or  $e$  is an is a part of this local state  $LS_i$  if and only if  $e$  belongs to the sequence of events that have taken place inside process  $i$  to that particular to the local state.

Similarly, for the event  $e$  and the processing state  $LS_i$  if  $e$  is not there into the local state of  $i$  that means if and only if it does not belong to the sequence of events that have taken process  $p_i$  to the state  $LS_i$  similarly for the channel between  $i$  and  $j$  the following set of messages can be defined based on the local state of a process  $i$  and  $j$ . So, the message is set to be in the transit that is between local state of  $i$  and local state of  $j$  there is a message which is there.

So, we are send of a message  $m_{ij}$  is available in the local state of  $i$  and it is not received in the local state of  $j$  then this particular message which is sent with not received is set to be in the transit.

(Refer Slide Time: 9:37)

**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,  
 $GS = \{ \cup_i LS_i \cup_{i,j} SC_{ij} \}$
- 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:**  $send(m_{ij}) \notin LS_i \Rightarrow m_{ij} \notin SC_{ij} \wedge rec(m_{ij}) \notin LS_j$

Global State and Snapshot

Now, let us see that after defining the processes and the local states let us see that what do you mean by the consistent global state. So, the global state of a distributed system is a collection of such local states of the processes and the channels. So, notably the global state  $GS$  is a union of all the local states of different processes and the union of all the channel states called state have a channel between  $i$   $j$ .

So, the local states and the channel state together is called the global state. So, global state  $GS$  is set to be consistent global state if and only if satisfies 2 conditions. So, condition 1 says that if there is a send of event send of a message  $m$   $i$   $j$  is available in some of the local state of  $i$ . This implies that  $m$   $i$   $j$  is there either if it is there in a state of a channel that is the message is either flowing in the channel or it is received in the local state if it is sent by a particular process.

So, if a message is sent then it must be either in the channel or it is received at the receiver and the second condition says that if the message  $m$   $i$   $j$  is not in any of the local states so this means that neither it will be appearing in the channel states nor it is there in the receive of that event  $m$   $i$   $j$  so if these 2 conditions  $C$  1 and  $C$  2 are satisfied in that global state then that global state is called a consistent global state.

(Refer Slide Time: 11:25)

## 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  $m_{21}$ .

Global State and Snapshot

## Example:

The diagram shows four horizontal timelines for processes P1, P2, P3, and P4. Events are marked with red dots and labeled as follows:

- P1:**  $e_1^1$ ,  $e_1^2$  (circled in red),  $e_1^3$ ,  $e_1^4$
- P2:**  $e_2^1$ ,  $e_2^2$ ,  $e_2^3$ ,  $e_2^4$
- P3:**  $e_3^1$ ,  $e_3^2$ ,  $e_3^3$ ,  $e_3^4$ ,  $e_3^5$
- P4:**  $e_4^1$ ,  $e_4^2$

Messages are shown as arrows:

- $m_{12}$  from  $e_1^2$  to  $e_2^3$
- $m_{21}$  from  $e_2^4$  to  $e_1^3$
- Another  $m_{21}$  from  $e_2^3$  to  $e_1^4$

Handwritten red annotations include:

- A checkmark above  $e_1^2$ .
- A circle around  $e_1^2$  with the word "send" written next to it.
- A checkmark below  $e_2^3$  with the text "rxd (m12)".

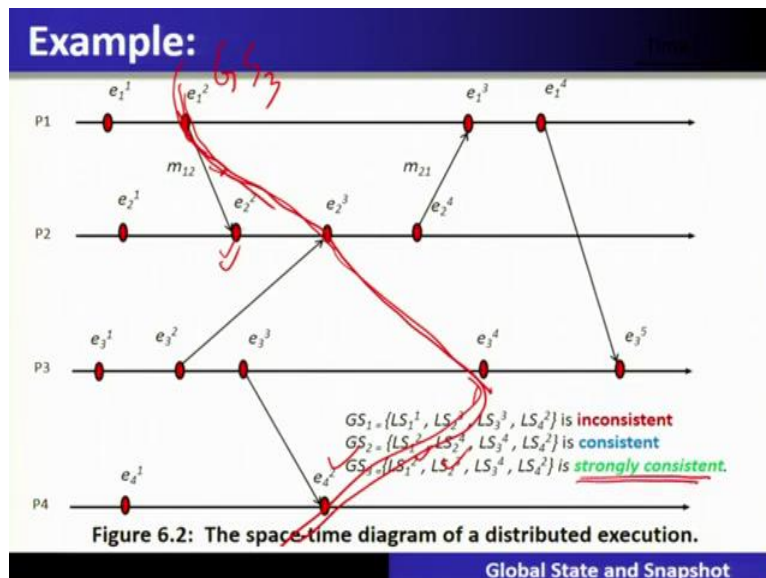
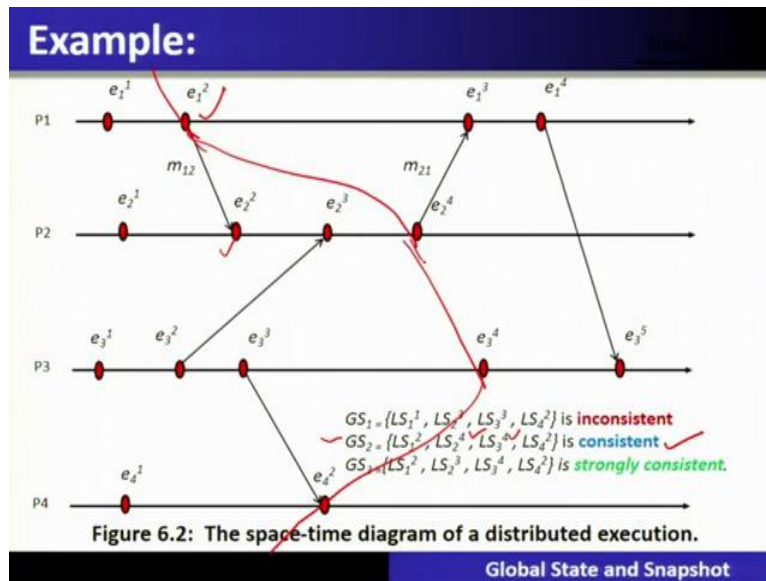
Legend for global states:

- $GS_1 = \{LS_1^1, LS_2^3, LS_3^3, LS_4^2\}$  is **inconsistent** (marked with a red checkmark and "KS")
- $GS_2 = \{LS_1^2, LS_2^4, LS_3^2, LS_4^2\}$  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.

Global State and Snapshot





So, a distributed system execution let us trace back the global state is consist of the local state LS 1 and so on is inconsistent because p 2 has recorded the receipt of a message m 1 2 however the state p 1 has not recorded it is state as sent let us see here in this example this global estate is called inconsistent let us see where is that LS 1 of 1 then 4 2 3 and then 3 3 and then 2 4.

So, this particular state you will see that if you see or hear that the message is received, but it is sent is not recorded therefore it is called inconsistent global state. So, this is the send of a message is not there but receiver of a message receiver of m 1 2 is there in the local state or in a global state of 1 this is the first global state.

Now, let us see the second example. The second example says that GS 2 is consistent let us trace what is GS 2 so local state of 1 2 this then 2 4 then 3 4 and then 4 2. So, this is called consistent why because you can see that the send of event is already there because receive is there into the so all the messages either they are who are there in the send they are either be received or is in transit therefore it is called consistent state.

Now, let us see the third one, which is called the strongly consistent, strongly consistent GS 3 stands for 1 2 then 2 3 then 3 4 and then 2 4. So, this particular snapshot this is GS 3 here you can see that there is no message in transit all messages which are sent there have been received. So, this kind of global state is called a strongly consistent global state.

(Refer Slide Time: 14:26)

**Global State of a Distributed System**

- A global state  $GS = \{U_i, LS_i^{x_i}, U_{j,k}, SC_{jk}^{y_j, z_k}\}$  is transitless iff  $\forall i, \forall j : 1 \leq i, j \leq n :: SC_{jk}^{y_j, z_k} = \emptyset$
- 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.

Global State and Snapshot

So, recording the global state of a distributed system without stopping is a very important paradigm and is often interested in the analyzing monitoring testing and verifying the properties of distributed applications or the systems and through the algorithm. So, design our efficient method for recording the global state of a distributed system becomes very important.

(Refer Slide Time: 14:54)

**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.

Global State and Snapshot

Now, let us see what are the issues are there in recording the global state autonomously with the help of algorithms. So, there are 2 issues need to be addressed. One is called issue number 1 says that how to distinguish between the messages to be recorded the snapshot from those not to be recorded the snapshot.

That means any message that is sent by your process before recording it is a snapshot must be recorded in a global snapshot that is has to be following this condition C 1 and any message that is sent by your process after recording a snapshot must not be recorded in the global snapshot that is it has to follow the condition C 2.

So, therefore, the condition to says that how to determine the instant when a process takes it a snapshot. So, a process  $p_j$  must record a snapshot before processing a message  $m_{ij}$  that was sent by a process  $p_i$  after recording a snapshot see these 2 issues required to be resolved.

(Refer Slide Time: 16:01)

### Example of Money Transfer

- Let  $S_1$  and  $S_2$  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  $S_1$  to site  $S_2$  and from site  $S_2$  to site  $S_1$  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  $S_1$  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  $S_2$ . Account  $A = \$550$ , Account  $B = \$200$ ,  $C_{12} = \$50$ ,  $C_{21} = \$0$ .

Global State and Snapshot

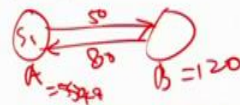
So, let us see through an example of a money transfer that is an online banking system. Let us see that you have 2 sites  $S_1$  and  $S_2$ . And these sites maintain the bank account  $A$  and  $B$  respectively. So, it maintains the bank account  $A$  and this maintains bank account  $B$  now site  $A$  site refers to a process in this example. So, there is a communication channel from site  $S_1$  to  $S_2$  and this communication channel is called  $C_{12}$  and  $C_{21}$  respectively.

Now, consider the following sequence of action which is also illustrated in the next slide figure at time  $t_0$  initially the amount 600  $A$  was having 600 and account  $B$  was having 200 and these channel values were 0 initially at time  $t_0$ . Now, when the time is increased to  $t_1$   $S_1$  initiates a transfer of 50 dollar from account  $A$  to account  $B$  so when the time is increased so account  $A$  will be now debited 50 dollars.

So, it will become 550 and since this particular money is debited it must be in the channel if it is not delivered so that will be the channel and therefore it will be debited from account  $A$  and account  $B$  is not altered and that 50 dollar will appear in that channel state.

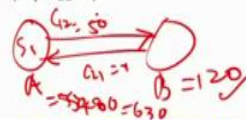
(Refer Slide Time: 18:09)

- Time  $t_2$  : Site S2 initiates a transfer of \$80 from Account B to Account A.
- Account B is decremented by \$80 to \$120 and a request for \$80 credit to Account A is sent on Channel  $C_{21}$  to site S1. Account A=\$550, Account B=\$120,  $C_{12}$ =\$50,  $C_{21}$ =\$80.
- Time  $t_3$ : Site S1 receives the message for a \$80 credit to Account A and updates Account A.  
Account A=\$630, Account B=\$120,  $C_{12}$ =\$50,  $C_{21}$ =\$0.
- Time  $t_4$ : Site S2 receives the message for a \$50 credit to Account B and updates Account B.  
Account A=\$630, Account B=\$170,  $C_{12}$ =\$0,  $C_{21}$ =\$0.



Global State and Snapshot

- Time  $t_2$  : Site S2 initiates a transfer of \$80 from Account B to Account A.
- Account B is decremented by \$80 to \$120 and a request for \$80 credit to Account A is sent on Channel  $C_{21}$  to site S1. Account A=\$550, Account B=\$120,  $C_{12}$ =\$50,  $C_{21}$ =\$80.
- Time  $t_3$ : Site S1 receives the message for a \$80 credit to Account A and updates Account A.  
Account A=\$630, Account B=\$120,  $C_{12}$ =\$50,  $C_{21}$ =\$0.
- Time  $t_4$ : Site S2 receives the message for a \$50 credit to Account B and updates Account B.  
Account A=\$630, Account B=\$170,  $C_{12}$ =\$0,  $C_{21}$ =\$0.



Global State and Snapshot

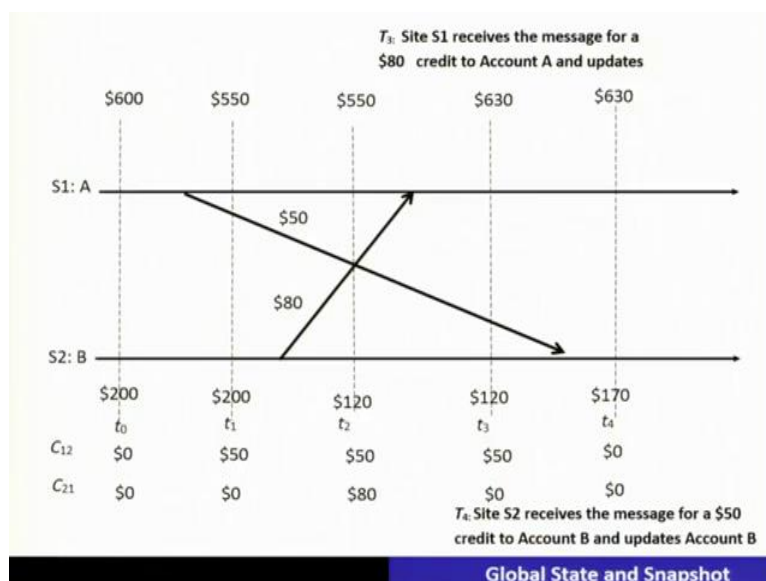
- Time  $t_2$  : Site S2 initiates a transfer of \$80 from Account B to Account A.
- Account B is decremented by \$80 to \$120 and a request for \$80 credit to Account A is sent on Channel  $C_{21}$  to site S1. Account A=\$550, Account B=\$120,  $C_{12}$ =\$50,  $C_{21}$ =\$80.
- Time  $t_3$ : Site S1 receives the message for a \$80 credit to Account A and updates Account A. Account A=\$630, Account B=\$120,  $C_{12}$ =\$50,  $C_{21}$ =\$0.
- Time  $t_4$ : Site S2 receives the message for a \$50 credit to Account B and updates Account B. Account A=\$630, Account B=\$170,  $C_{12}$ =\$0,  $C_{21}$ =\$0.

Global State and Snapshot

At time  $t_2$  S2 initiates a transfer of 80 dollar from account B simultaneously to A. So, it is now sending 80 dollar. So, his account will be debited by 80 dollar and it will become from 200 it will become 120 and there it was 550 and 50 dollar was in the transit now at time  $t_3$  as one receives the message for 80 dollar credit.

So, it will be now adding up the 80 dollars and update the account as 630 that is what is shown over here whereas the account for B is 120 and you can see that  $C_{21}$  is equal to 0 and  $C_{12}$  is equal to 50 dollars at time  $t_4$  site 2 receives this 50 credit and thereby these particular values in the channel becomes 0 and this will become 170 so A will be having 630 so this is the correct way of the snapshot.

(Refer Slide Time: 19:43)



And that is what is being explained through this particular figure.

(Refer Slide Time: 19:50)

The slide, titled "Global State and Snapshot", contains the following text:

- 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.

Global State and Snapshot

Now, suppose if you are recording this global state through the algorithm and what do you find is that record A 1 is recorded at time  $t_0$  to show the amount as 600 dollars and local state of B and channel is C 1 and C 2 is recorded to show all these respectively. So, the recorded state shows 850 in the system. So, 50 dollars will appear extra and this particular region is due to the inconsistent global state recorded.

Therefore, this example we will illustrate in the next slide shows that the recording consistent global snapshot is not a trivial task. So, recording is to be coordinated through an algorithm. So, recording activities of individual computers must be coordinated properly.

(Refer Slide Time: 20:45)

### 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  $rec(m_{ij}) \rightarrow rec(m_{kj})$ "

Global State and Snapshot

So, to do this algorithm let us understand some of the models of communication. So, we will use the FIFO model where each channel will act as first in first out message queue and the message ordering is also preserved by this particular channel. So, there is another model of communication called non FIFO channel non FIFO model where the channel acts like the set in which the sender acts the messages and receiver removes the messages in any random order. So, we are not following this model at this moment.

For this for the sake of simplicity we are using this FIFO model. Now, the system also supports casual delivery of the message which satisfied the following property that if there are 2 messages that is this is  $i$  this  $m_{ij}$  is being sent this is another  $m_{kj}$  is being send. So, this process is  $j$  this is  $i$  and so on. So, if the send of  $m_{ij}$  is happened before send of  $m_{kj}$  then the receive in the same process  $j$  should happen before the receive of  $m_{kj}$ . So, this ordering is called causal delivery.



(Refer Slide Time: 22:06)

### Snapshot algorithm for FIFO channels

**Chandy-Lampert algorithm:**

- The **Chandy-Lampert** 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.
- A process must record its snapshot no later than when it receives a marker on any of its incoming channels.

Global State and Snapshot

Now, let us go ahead with the snapshot recording algorithm with a model called FIFO channel model. So, this particular algorithm is given by a famous person which works which is implemented in a cloud and distributed system. So, that is called Chandy Lampert algorithm uses the control messages which is which is called a marker whose role in the FIFO system is to separate the messages in the channel.

So, after the state has recorded a snapshot, it sends the marker along outlaw outgoing channels before sending out more messages. So, marker separates the message in the channel into those included in the snapshot from those not to be recorded in the snapshot.


(Refer Slide Time: 22:55)

### Chandy-Lampert 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.

Global State and Snapshot

## 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.

Global State and Snapshot

So, the algorithm is initiated by any process executing the marker sending rule by which it records its local state and sends the marker on each outgoing channel. So, take this particular example. If let us say this is the process who is sending the marker sending rule it will first record its local state and then it will send the marker on all the outgoing channels. This is the call marker sending rule.

So, the process which executes the marker receiving rule on receiving this particular marker message  $m$  when it is received when a process receives the marker sending rule  $m$  if the process has not 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. So, the algorithm terminates.

So, again, I am repeating so on receiving this marker message which is special if the process is not recorded local state it will record the state of the channel on which the market is received as empty and execute the market sending rule to record its local state. So, this is important to understand there are 2 rules in the Chandy Lamport algorithm one is marker sending rules the other is marker receiving rule will explain in more detail about the example.

So algorithm terminates after each process has received a marker on all of its incoming channels. So, all the local snapshot get disseminated to all other processes and all the processes disseminate the global state.

(Refer Slide Time: 24:48)

## 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  $C$  ✓

Global State and Snapshot

So now let us see these 2 rules. In the Chandy Lamport algorithm marker sending rule for a process. It says that process  $i$  records its state and for each outgoing channel  $C$  onwards the marker has not been sent,  $i$  send some marker along  $C$  before it sends further messages. So, this is called marker sending rule.

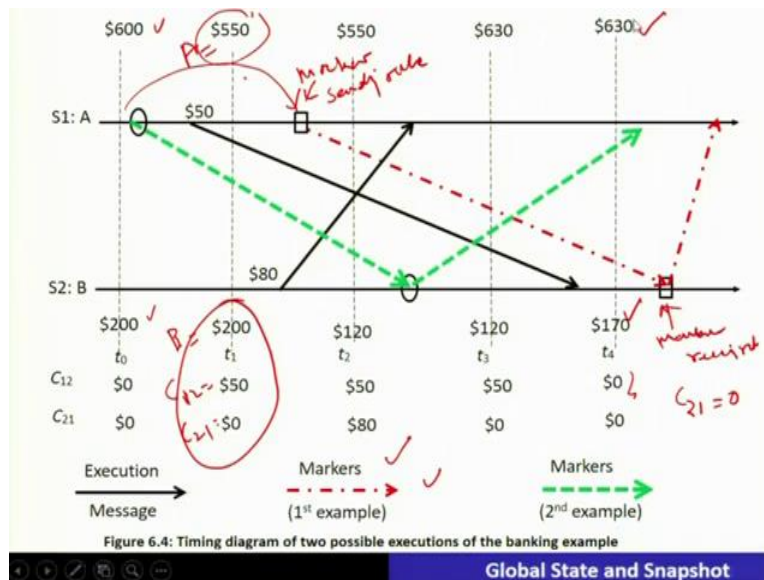
Marker receiving rule that is when a process  $j$  receives the marker along the channel  $C$  if the  $j$  has not recorded its state then it will record the state of a channel as empty and follows the marker sending rule that means now it will initiate this tool if  $j$  has recorded its state, then it will record the state of the channel as the set of messages received along see after  $j$  state was recorded and before  $j$  received the marker along  $C$ .

(Refer Slide Time: 25:52)

### 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 .

**Global State and Snapshot**



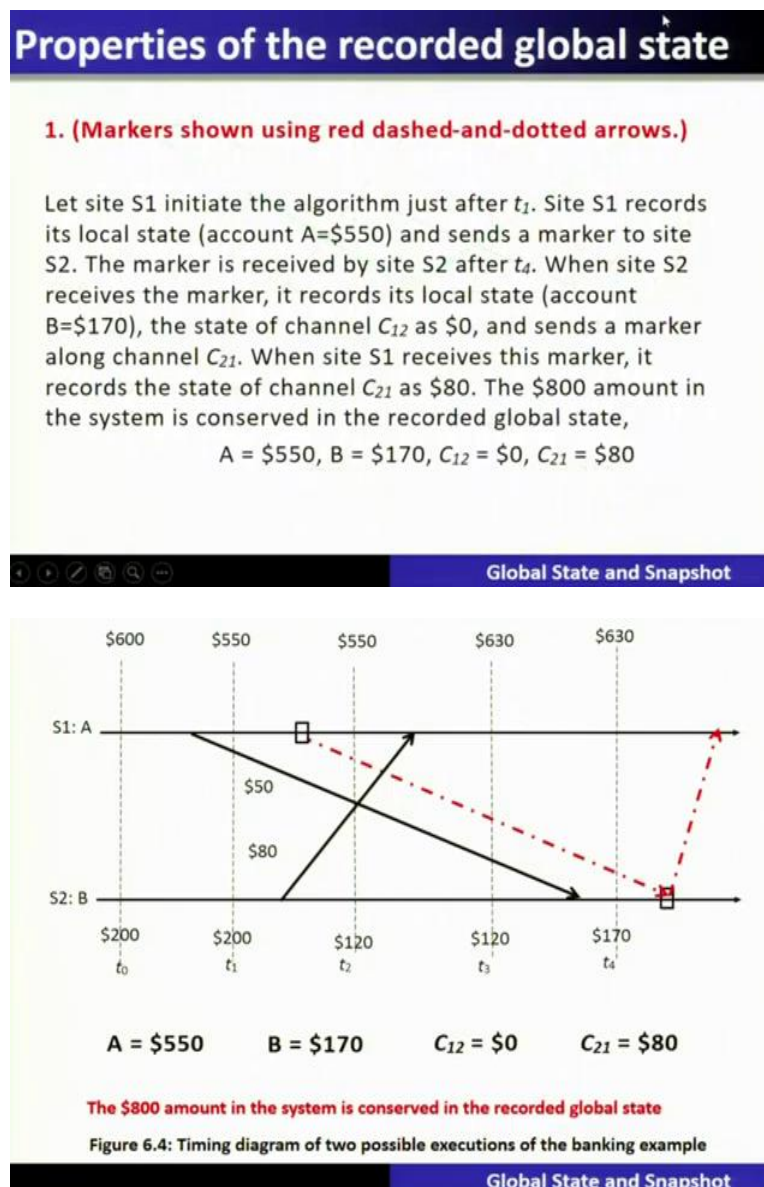
So, these are 2 simple rules and let us take this by an example. So, there are 2 examples, let us trace the first example which is shown by the red one. So, the same running example we have taken for the bank account site S 1 and S 2 and their initial values are 600 and 200. So, at this point the algorithm will be initiated. So, marker sending rule will be applied over here.

So, instead so site A since it has not recorded so it will record its state and send this marker on the all the outgoing channel that is to B which is shown as the red. So, when it records its state so it is a state will become like this that is 200 50 and 0 so it will record its state as the value of A is 550 and B is 200 and I stayed of this particular channel C A B you will C 1 2 has got 250 and C 2 1 is equal to 0.

So, this will be the marker sending rule will be applied. Now, once the marker is received by as to marker receiving rule when marker will receive this particular condition. Now, what it will do it will say that if it has recorded it is state before that it has not recorded then in that case it will record it is channeling state as 0. S

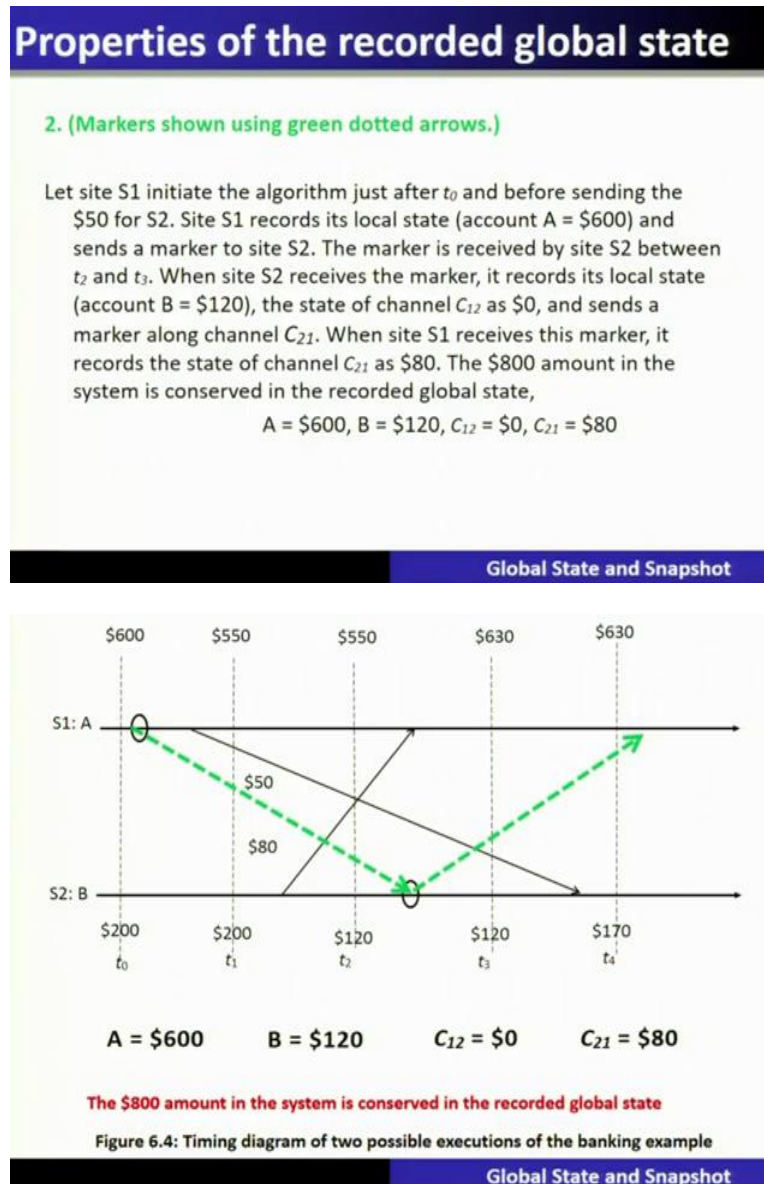
o, it will record it is state of a channel C 2 1 and 0 and then it will apply the marker sending rule. So, marker sending rule will say that it will then record it is state. So, when you say that it will record it state that means it will record the state as 170 and 630 and so on. So, just see that this happens.

(Refer Slide Time: 28:39)



So, let us go ahead by looking up here in this particular case so 550 and B will be 170 and C<sub>21</sub> will be 80. So, the amount 800 is conserved in this particular snapshot algorithm.

(Refer Slide Time: 29:09)



Let us see the same algorithm running on another example which is shown over here. So, if A is the site 1 will now run the marker sending rule marker sending rule will record it is a state that is 600. Now when the marker will be received by S 2. And since S 2 has not yet recorded anything then it will record your state that is 120 and the channel state of 1 2 will become 0. But the channel is state of 2 1 will become 80 in this case.

(Refer Slide Time: 29:57)

**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.

Global State and Snapshot

So, what we have seen is the properties of the recorded global state in both these possible runs of the algorithm the recorded global state never occurred in the execution this happens because the process can change its state asynchronously before markers are sent and received by other sites. But the system could have pass through the recorded global state in some equal and executions.

So, the recorded global state is valid in an equal and execution if some stable properties holds that in the system before a snapshot algorithm begins it holds the recorded snapshot therefore the recorded global snapshot is useful in detecting the stable conditions.

(Refer Slide Time: 30:43)

**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.

Global State and Snapshot

So, let us conclude this lecture. So, recording the global state in a distributed system is a important paradigm in the design of various applications built around the distributed system concepts and the design of efficient method for recording the global state is a very important issue. In this particular lecture we have shown the algorithm working on 2 different examples.

So, this lecture discusses the formal definition of a global state in a distributed system and issues related to that global state capture and then we have discussed the Chandy Lamport algorithm to record the global state of a distributed system. Thank you.