Database management system

Prof. D. Janakiram
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 25
Basic 2-phase & 3-phase Commit protocol

In the last lecture, we have just looked at basic two phase commit protocol. Just stopped at that point where we just had the basic protocol and how it works. Actually two parts are there. One part is the participant protocol and other is the coordinator protocol. How the coordinator protocol working is, the coordinators starts with sending a prepare message and once he sends a prepare message from the initial state actually he reaches the state where he is now undecided.

So the prepare message is sent to all the participants. Once the prepare message has been received by the participants, they will reply back with the ready message to the coordinator. If the coordinator actually receives all the ready messages, then he will take the commit and he will send the commit message then the coordinator will reach the commit state and if he receives any of the participant actually says he wants to abort that means for the prepare message he gives an abort answer message, then you are going to take an actually abort command message and this state.

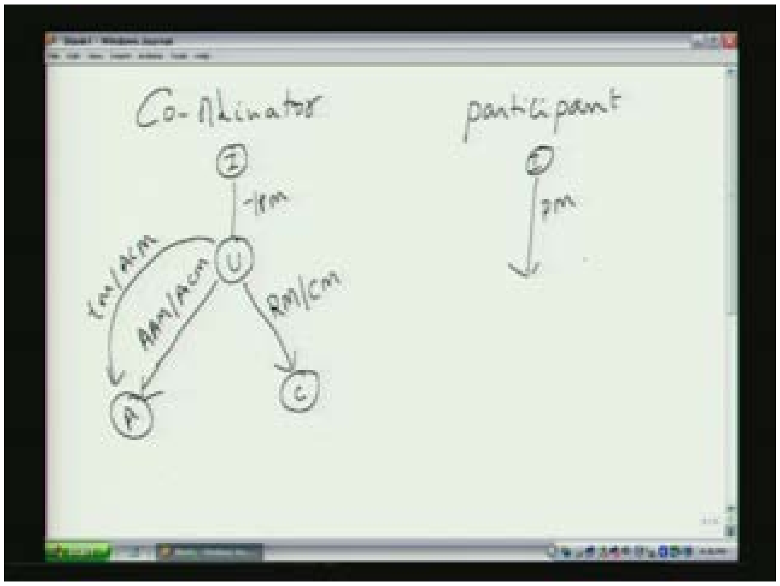
Actually the coordinator reaches the abort state, it also possible for the coordinator to reach the abort state. If he times out, that means he does not receive the message from the participant, then one of the participant failed. Since one of the participants is failed you are bound to actually taken abort message. Right this is the basic part of the coordinator. As far as the participant is concerned there is each participant site for the two phase commit is concerned, each participant start with an initial state now he is going to get a prepare message. In response to the prepare message, he can basically give a ready message.

Now he gives the ready message, he is in the basically ready state then he is in the ready state. Now wait for the coordinator’s decision. If the coordinator actually gives the abort command message, then you are going to acknowledge and reach the abort state. Even after actually you reply with the ready message, it is possible for coordinator can still take a abort decision because in that particular case, the other participants may not have replied back unless he get all the ready message, he cannot commit the transaction.

Now in the other case, all the participants replied with ready message. The coordinator is going to send the commit message in which case the participant actually commits, then sends an acknowledgement that means he has written all the values on to the database. This is the basic two phase commit protocol that we have seen in the last lecture and were we have stopped is, just explain this protocol and then said that what happens in terms of different failure scenarios, we thought that we looked at in the last class that’s why we stopped in the last class.

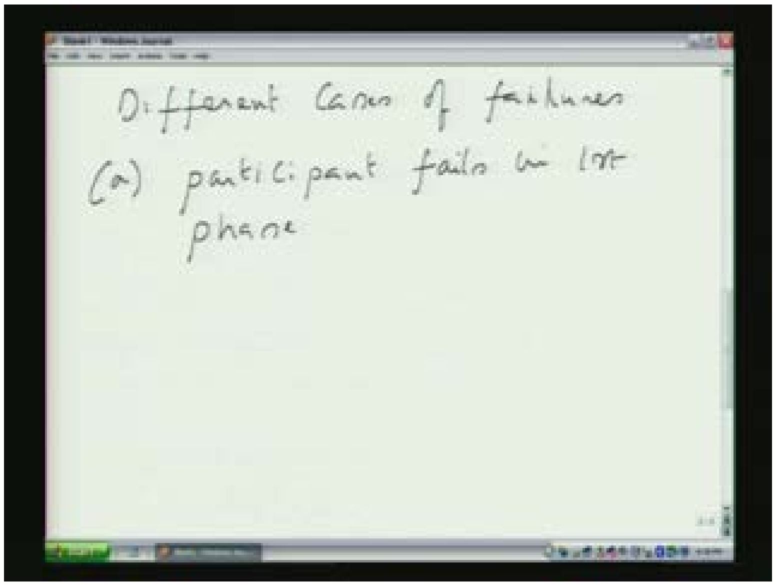
Now that several things can happen in this basic protocol. For example: coordinator can fail in the first phase or second phase participant can fail in the first phase or in second phase network could fail which means that its possible for the various nodes to get partition, some nodes in one partition and some nodes in another partition. So, all these are possible scenarios as far as this protocol is concerned. Now one of the things we looked at what are the cases were this protocol is resilient which means that which are the failure scenarios were this will still work correctly.

(Refer Slide Time 5.58)



Now let us first take the case were you know we are going to look at different cases of failures. Now the first kind of failure is a participant fails in phase 1. What happen if the participant fails in the two phase commit? We are talking about the first phase.

(Refer Slide Time 6.39)

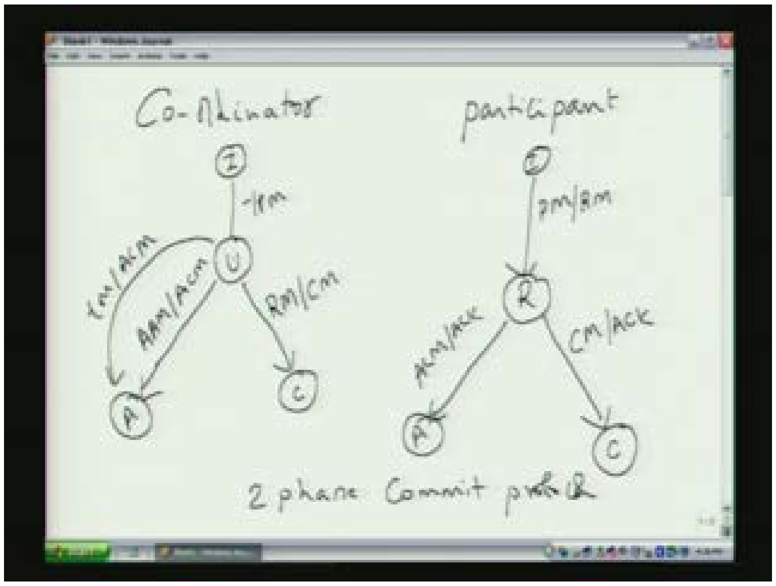


Different Cases of failures

(a) participant fails in 1st phase

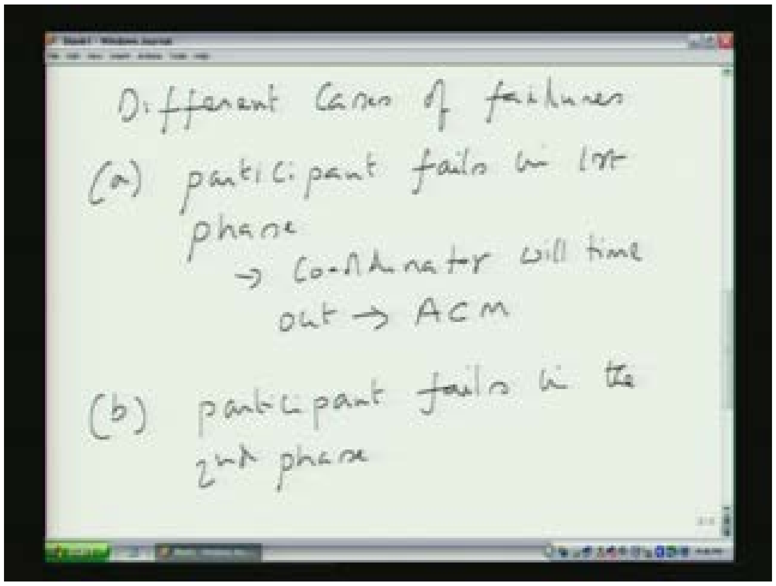
Now what happens in this case is the coordinator will time out and then once the coordinator times out is going to take the decision of aborting or sending an abort message here. For example: here the coordinator is going to timeout and he is going to in that response is going to take an abort message, abort command message in the minute actually one of the participants has not replied.

(Refer Slide Time: 7:02)



So, i think in that particular case, the coordinator will time out and then and take an abort decision. Co-coordinator will timeout. As a result he is going to take an abort command message that's what going to happen when participant fails. So obviously two phase commit is resilient for participant failures in the first phase. Now what happens if the participant fails in the second phase of the Participant fails in the second phase that means in the first phase, the participant has replied with the ready message. It means that he actually replied the ready message and now he is waiting for the coordinator's decision. Coordinator can take either an abort decision or a commit decision.

(Refer Slide Time: 08:09)



Different Cases of failures

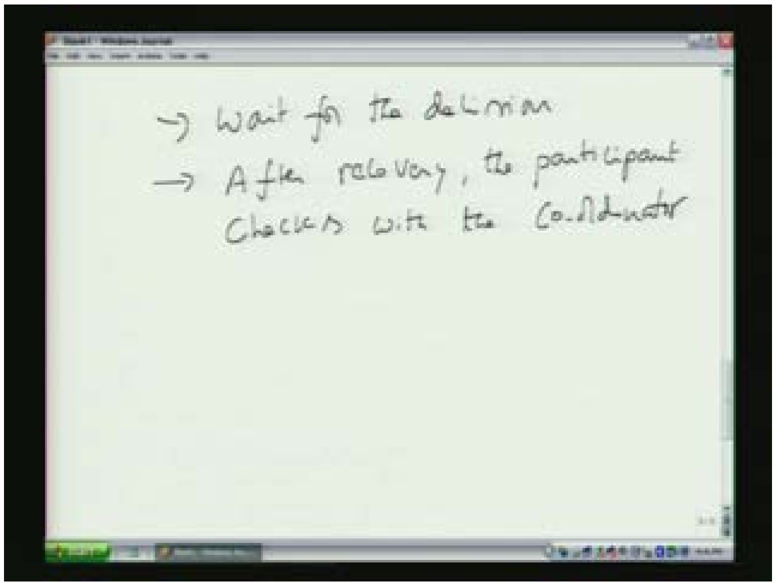
(a) participant fails in 1st phase
→ Coordinator will time out → ACM

(b) participant fails in the 2nd phase

Now, if you say that the coordinators have taken an abort decision or commit decision? After recovery you have to check with the coordinator and accordingly you have to do. As far as you have concerned, you have replied with the ready message that means you actually now all the logs and you cant release any of the locks or any of the resources till such time you know the decision of the coordinator. Right in this particular case, you are going to wait for the decision of the wait for the decision.

Now since you are waiting for the decision, since you have failed for the decision all that you have to do is after recovery that means after you recover from failure. After recovery, the participant should check with the coordinator. Participant checks with the coordinator for the decision. What happened about the transaction he has to check with the coordinator and accordingly he has to terminate the transaction that means based on what the coordinator tells now. Coordinator need not wait for the participant because it already replied with replied message.

(Refer Slide Time: 9:19)



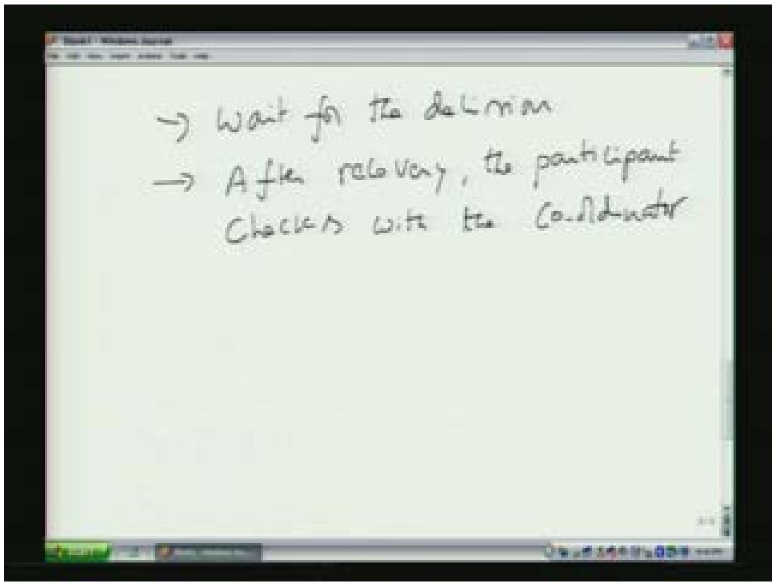
→ Wait for the decision

→ After recovery, the participant checks with the coordinator

So, in this particular case, the coordinator is feed to choose a decision depending on read message or if he get all the ready messages then he take the decision to commit the message, commit the transaction in which case the commit decision would have been taken. So, that participant has to find out what happens to this transaction he has to do accordingly. One of the things is neither other participants will be blocked because of this participant failure the coordinator will be blocked.

All the coordinator has to do is to read the decision when the participant comes back, he has to tell the participant about his decision and when he get the acknowledgement from participant he is going to write the final log on this transaction, it has been successfully completed. That's the thing that needs to be done in this particular case. Now, it is possible also for the failures to happen for the coordinator at different phases and also possible that both coordinator and participant also fail. Need not be just failures of participant alone coordinator alone. But right now we looked at what happens if the participant fails, one or more participants fail in the first phase and in the second phase.

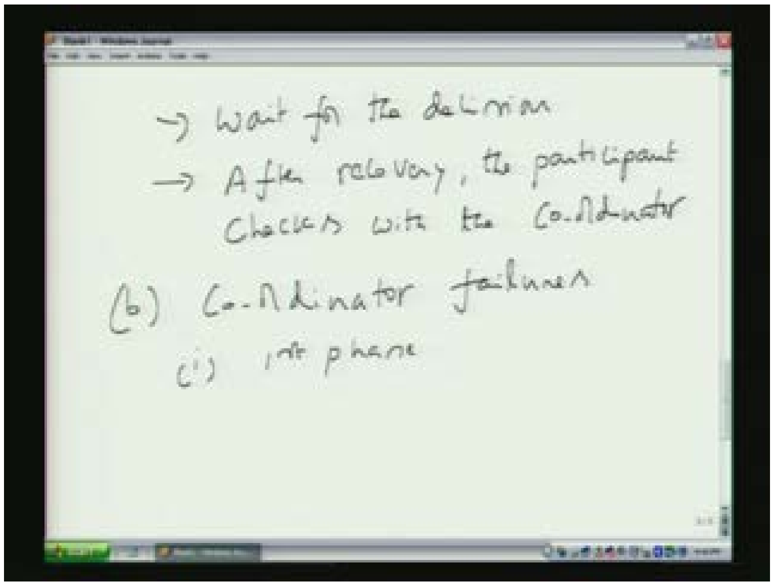
(Refer Slide Time: 10:30)



→ wait for the decision
→ After recovery, the participant
Checks with the Co-ordinator

So, if you want to look at what happens to the coordinator failure? Suddenly you have to now understand in which phase the coordinator has failed. Again we can see the case were coordinator fail in the first phase if the coordinator fails in the first phase, that means he sends the prepare message and after that has failed. He has not actually taken any from this transaction.

(Refer Slide Time: 11:36)

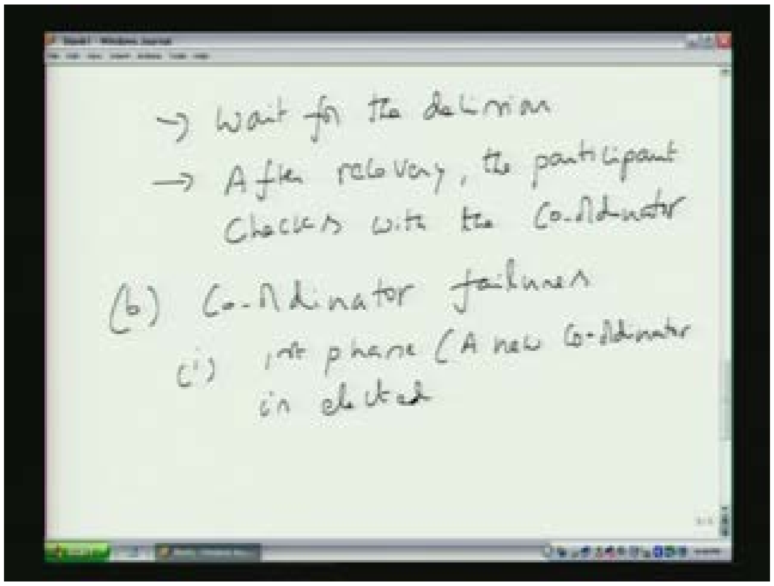


→ wait for the decision
→ After recovery, the participant
Checks with the Co-ordinator

(b) Co-ordinator failures
(i) 1st phase

Then he fails before the ready messages actually arrived at the coordinator. So the coordinator has not reached the decision. But after sending the prepare message if he fail the coordinator has failed. Now the way to recover is simple. All that the participants have to do is, they have to elect the new coordinator and then restart the protocol. That means the new coordinator will send the prepare message and correspondingly he will take the decision about the transaction by running by the two phase commit protocol. So in this particular case, a new coordinator has elected assuming that there are no failures, new coordinator is elected here and then the transaction is restarted.

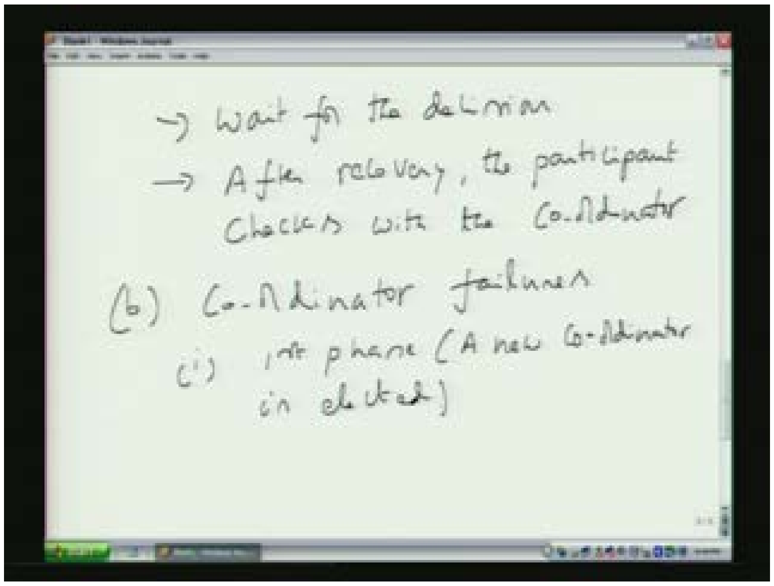
(Refer Slide Time 12.12)



→ Wait for the decision
→ After recovery, the participant checks with the Co-ordinator
(b) Co-ordinator failures
(i) 1st phase (A new Co-ordinator is elected

Two phase commit of the transaction is restarted protocol and then based on that the commit protocol will be executed based on that. One of the assumptions that is making here is that all the participants are live.

(Refer Slide Time 12.24)

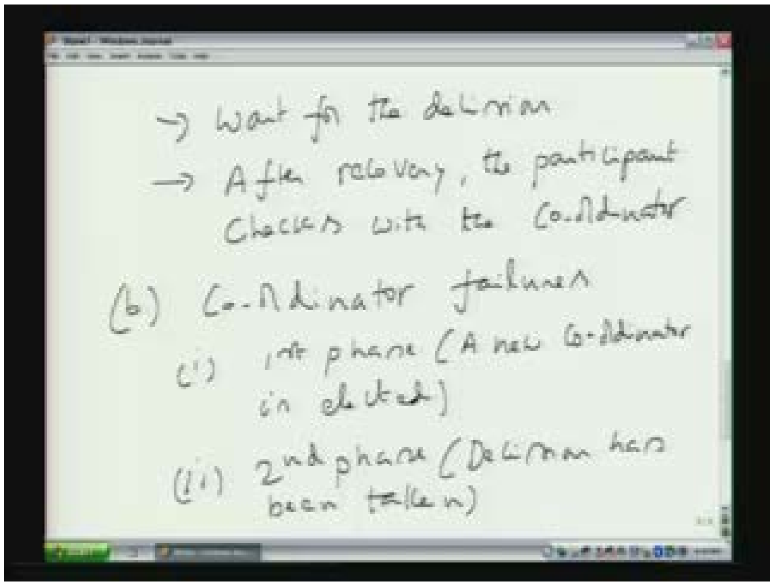


→ Wait for the decision
→ After recovery, the participant checks with the Co-ordinator
(b) Co-ordinator failures
(i) 1st phase (A new Co-ordinator is elected)

They are not dead but if they are if it is not the case also, what will happen is, the new coordinator will be elected and he will be time out when he starts the protocol. So, he will make an abort decision. So in that sense, the protocol become reentered means any number of times, any number fails failures is occurred. For example: you elected a new coordinator. Now that coordinator will fail again in the first phase again you are going to again have the election algorithm to elect a new coordinator. This is basically protocol is very simple.

Now in the other case, if the coordinator actually fails in the second phase. This is an interesting problem. Actually the fact you are saying that the coordinator actually fail in the second phase means that taken a decision. Right now, it means that decision has been taken decision of aborting or committing has been taken. Now one of the conditions that can happen here is as this decision known by another partition which we can say now there is a case were at least one participant knows the decision coordinator before he fail.

(Refer Slide Time: 13:53)


→ Wait for the decision

→ After recovery, the participant checks with the Co-ordinator

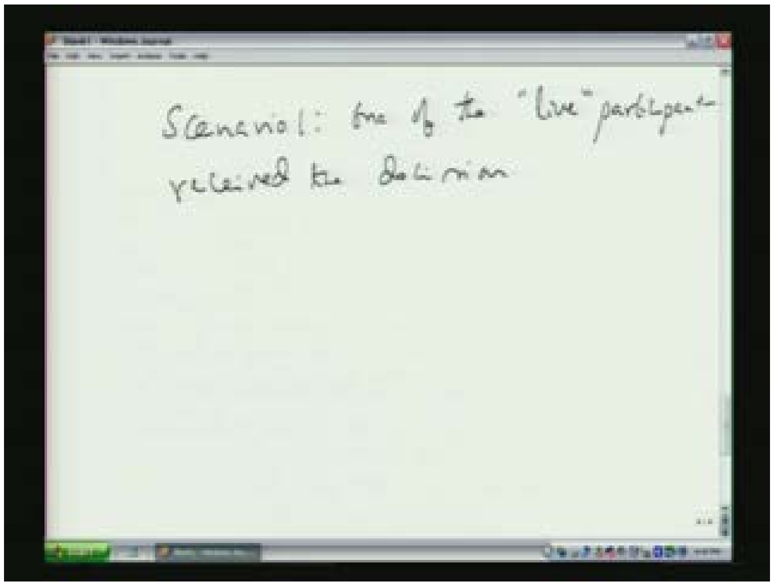
(b) Co-ordinator failures

(i) 1st phase (A new Co-ordinator is elected)

(ii) 2nd phase (Decision has been taken)

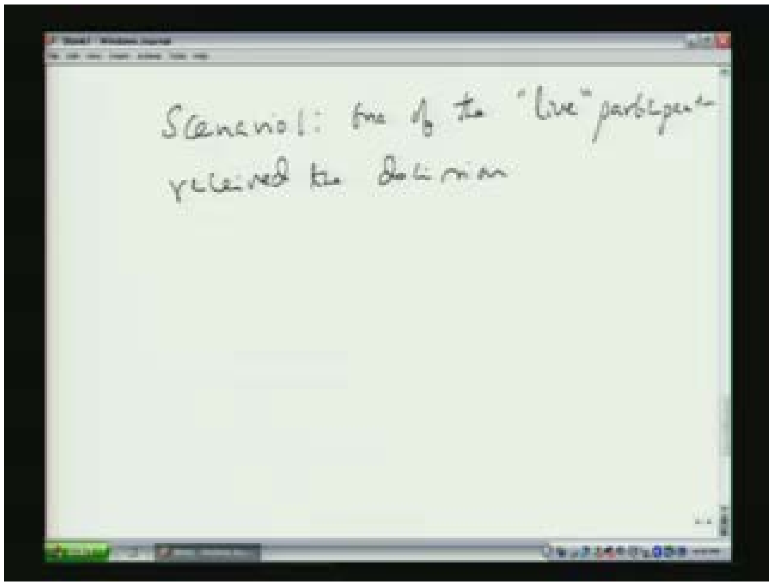
So, the scenario one is here one of the participants received. One of the live participants, which means that he is actually live and then he actually receives the coordinator of decision live participants. Received the decision which actually means that all that you need to do in this particular case is, since the decision is known the same decision can be conveyed to everybody.

(Refer Slide Time 14.27)


Scenario1: One of the "live" participant received the decision

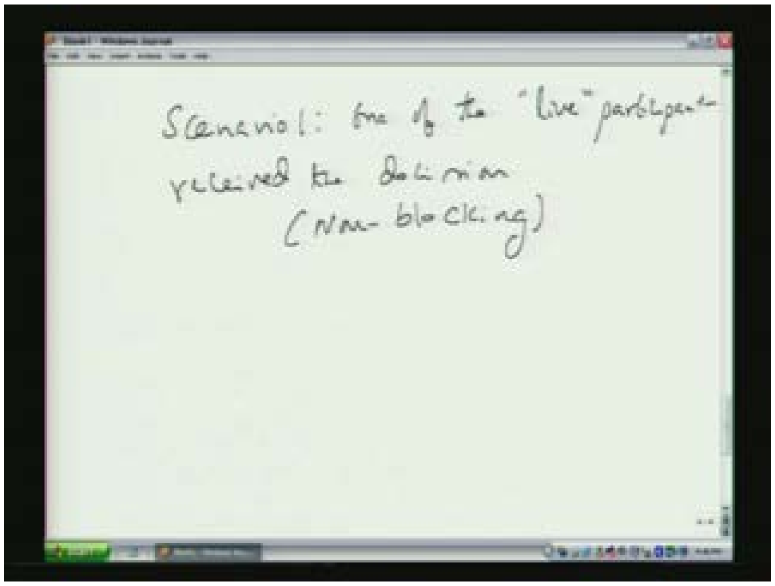
If it is a commit decision everybody will commit, if it is an abort decision everybody will abort. Right it is very simple case where the decision of the coordinator is already known the coordinator fails some participant does not receives the decision.

(Refer Slide Time 14.47)


Scenario1: One of the "live" participant received the decision

All that they have to do now is, find out if any of the participants has received the decision of the coordinator. If they have then will become simple scenario were that decision can be implemented by other participant also. In this particular case, it becomes non-blocking. That means that normal that means is nobody actually block. This is very important you are not waiting for somebody to recover back right.

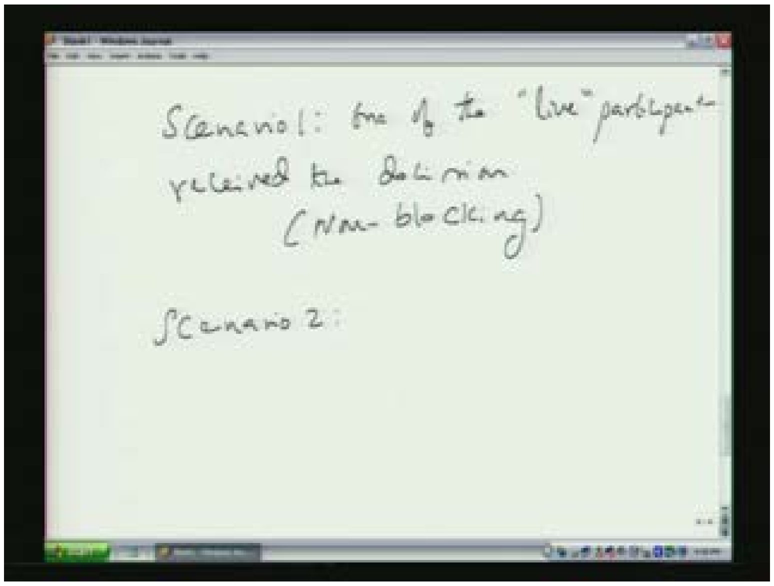
(Refer Slide Time: 15:25)



Scenario1: One of the "live" participant
received the decision
(Non-blocking)

For example: if shooting is happening, our recording person decides to go out for tea or something, and then we are all blocked. Since the time backs know that is a blocking protocol. You do not want to think blocked. You wont actually even it goes and put in automatic modes and off then we are not blocked. We can continue our job. Similar way, even one of the coordinators fails and it still wait for the participant to recover back from the failure. It is a very nice thing. It is a very simple case that is how we have to the model of the protocol we do not want to know any other participants suffer. Because of this failure suffer means have to wait now is you do not know when that recover is going to happen is you get indefinitely blocked if such a thing happen. So one of all the cases we have seen so far two phase commit protocol is non-blocked. It does not block anybody.

(Refer Slide Time: 16:44)



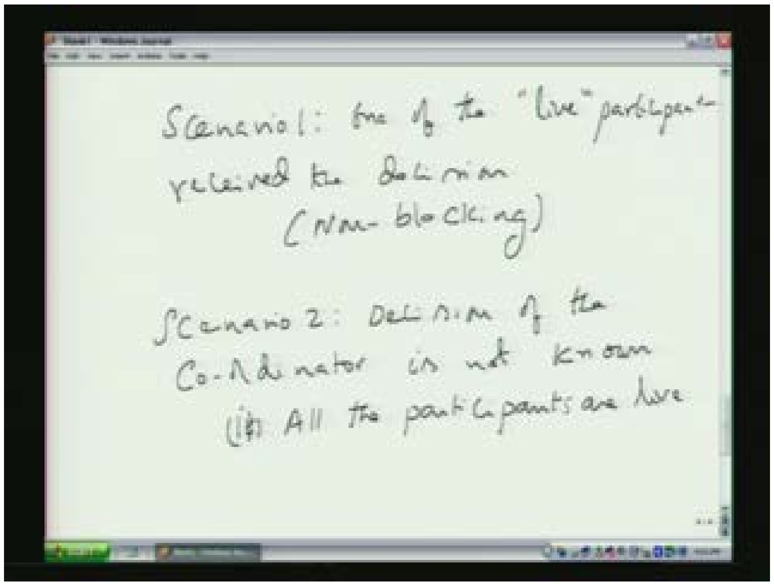
Scenario 1: One of the "live" participant received the decision
(Non-blocking)

Scenario 2:

Now there is an interesting case that is going to come now where we can see the case were the scenario 2 were the coordinator fail. None of the live participants know about the decision of the coordinator. But again there going to be two sub cases here. Now we can say decision of the coordinator is not known. In this case, it is possible that you know all the participants are live, that means there is no other participant failure. That means only coordinator has failed. The decision of the coordinator is not known. But all the case where this is only first case is not the second case.

First case, were all the participants are live that means there is no participant failure. All the participants are live. In this particular case, since all the participants are live, a simple thing they can do is they can restart the two phase commit protocol by electing a new coordinator. In this particular case, you will elect a new coordinator and restart the two phase commit protocol.

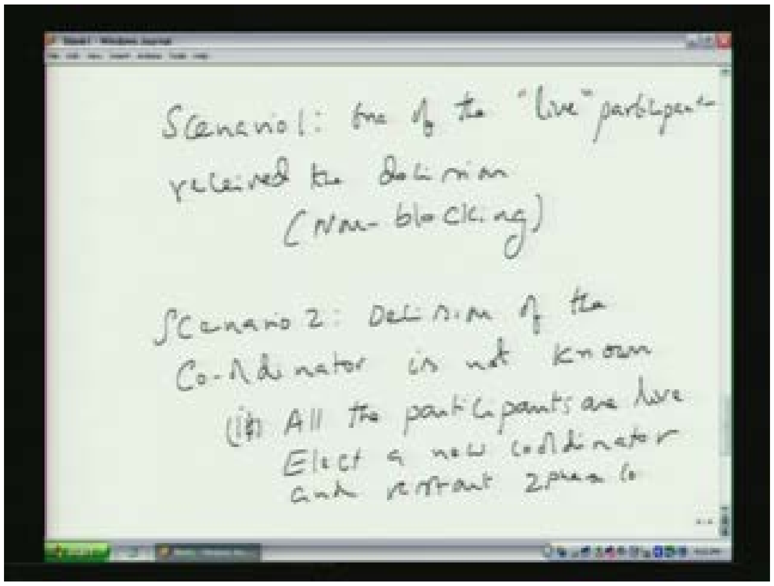
(Refer Slide Time 17.54)



Scenario 1: One of the "live" participant received the decision
(Non-blocking)

Scenario 2: Decision of the Co-ordinator is not known
(i) All the participants are live

(Refer Slide Time 18.20)

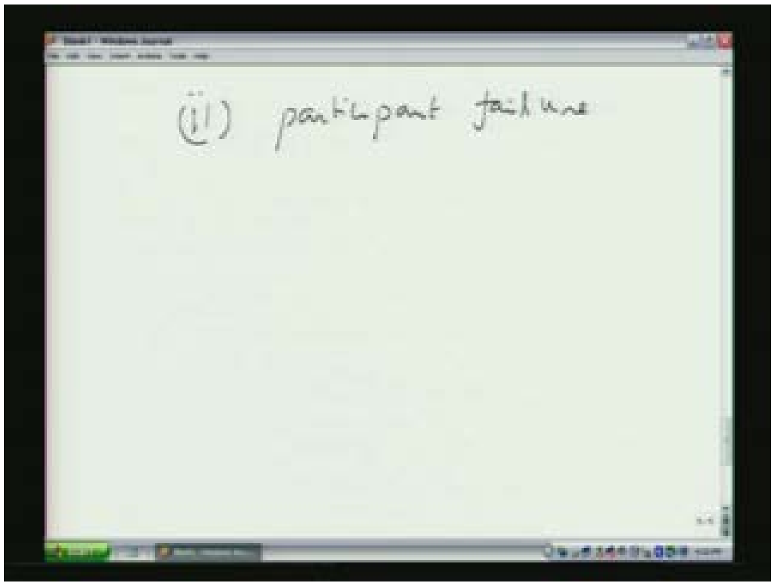


Scenario 1: One of the "live" participant received the decision
(Non-blocking)

Scenario 2: Decision of the Co-ordinator is not known
(i) All the participants are live
Elect a new coordinator and restart 2phase C

You are going to see is, the case where if you say that the two phase commit the participant also failed along with the coordinator right. So, there is not only a coordinator failure, but there is a participant failure which means that, now the rest of the people the interesting case is here.

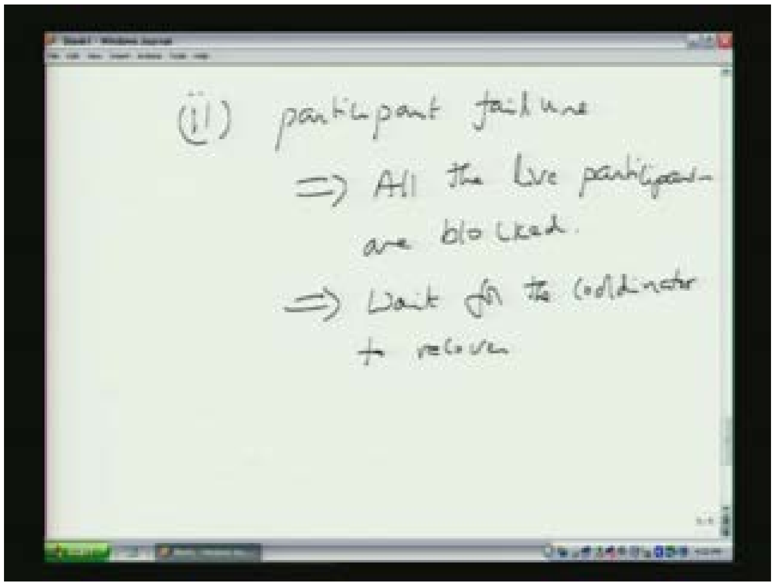
(Refer Slide Time 18.51)



Imagine that the coordinator took a decision of committing and that is only known to the participant and he actually wrote all the values of the database and then he actually fail. Now, if the remaining participant elect the leader, coordinator and then terminate the protocol, it might contradict. If the earlier coordinator already taken a decision and that is know to this partition. There is no way to recover back from this.

So basically this is the case were all the other participants in this particular case, all the live participants are blocked. All the live participants are blocked till such time the coordinator recovers. You cannot do anything except to wait for the coordinator to recover wait for coordinator to recover. Now often a coordinator failure also means a participant failure, because the transactions were it is started normally coordinator is one of the participant sides.

(Refer Slide Time 20.09)


(ii) participant failure
⇒ All the live participants are blocked.
⇒ Wait for the coordinator to recover

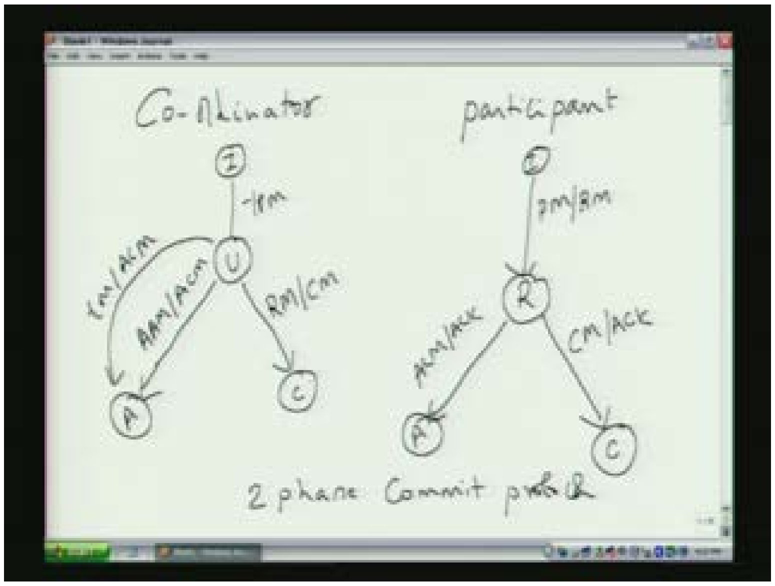
For example: if you basically look at how the transaction is started one of the sub transactions were it is getting executed. He also takes up the responsibility of being a coordinator. Normally coordinator failure will be equivalent to a participant failure and that is the scenario were two phase commit will not be able to recover back and this is an important case. Normally you do not have a coordinator failure. A coordinator failure normally means a participant failure. Only thing is, if it is fail in the first phase then you are bit lucky because in the particular case, the remaining participant can elect the coordinator can make a decision but if you actually failed.

In the second phase, phase two after actually sending the ready message, you discover that the coordinator actually fail you will be in trouble because you would not know whether the coordinator is taken a decision or not take the decision. So that is the case were the participant is forced to wait till the coordinator recovers back from the failure. Now have explained for this, what we are going to see is how this protocol gets modified to also allow protocol to recover in the case where the coordinator fails in the second phase along with the participant and that modification is what we see as the three phase commit protocol.

What the three phase commit protocol does is, intuitively to understand what the three phase commit protocol does is, when the coordinator actually fail you would not know what the state it state is, and this is very important distributed system. This is the typical problem you will face in the distributed system. You need to know the state of other node when things have actually failed or when the system crash. If you know the state of the system is easy for you to recover back.

But now if you do not know the state of the system, then you are in trouble recovering back from that failure of the particular system. Now, in this particular case when the coordinator actually fails you won't know whether it is commit decision or an abort decision. Now what do you want to decide now is or want to see is, what is the state in which it is? To get out of this ambiguity, for introducing a three phase that means the coordinator will not directly to get in to the phase of committing. If you carefully observe the problem is in terms of moving um coordinator moving directly when it gets the ready message in to the commit state. If you actually record in between a state where the coordinator says that, now I have received all the ready message.

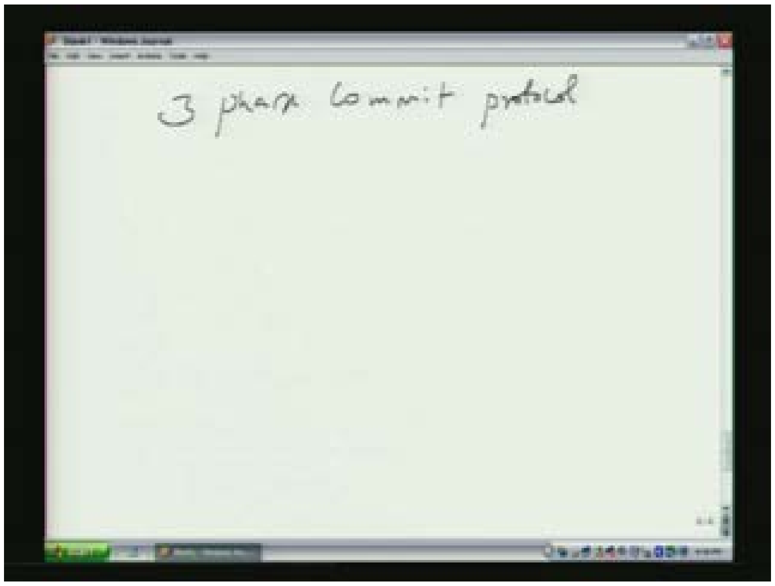
(Refer Slide Time 23.32)



Now i am prepared to commit, it is still not committed. But, then now you introduce a prepare to the commit state for the coordinator which means that now you know whether the coordinator has prepared to reach the commit state or not. If it is not reached to prepare commit state, then no harm done because he failed in the second phase but only in the third phase actual commitment would have taken place.

Right basically you need to know the state of the coordinator when he actually die or when he fails. So in that you remove that you need to introduce a third phase that’s were you basically look at the three phase commit protocol. What were we are going to do now is see how it get modified for the three phase commit protocol or how the three phase commit protocol works and also look at what kind of failure scenarios the three phase commit protocol can tolerate or it is resilient to what kind of failure.

(Refer Slide Time 24.49)


3 phase commit protocol

Now what we are going to do is, for both the participant as well as the coordinator will introduce a new state. A new state were they have to first get in to the before commit state or prepared to commit state before they actually commit. They cannot directly get in to the commit state from the ready state. Now the coordinator part is going to look at something like this will start with an initial state and now as usual send a prepare message to all the sides. Then i basically reach the undecided state here because I have still not decided on the state. Now if i basically get all the ready messages then I will actually enter what i call as a before commit state i have still not committed.

So, i basically reads a before commit state here. Now once i reached the state before commit state and i receive acknowledgement from all the participants. Then i take the commit message, that means i get an all those participants then i will issue the commit message and then i reach the commit state after receiving ok from all the participants. Now it is possible for me for various reasons to move from before commit state to an abort state. If i do not receive okay from all the sides, i can still end up in an abort state.

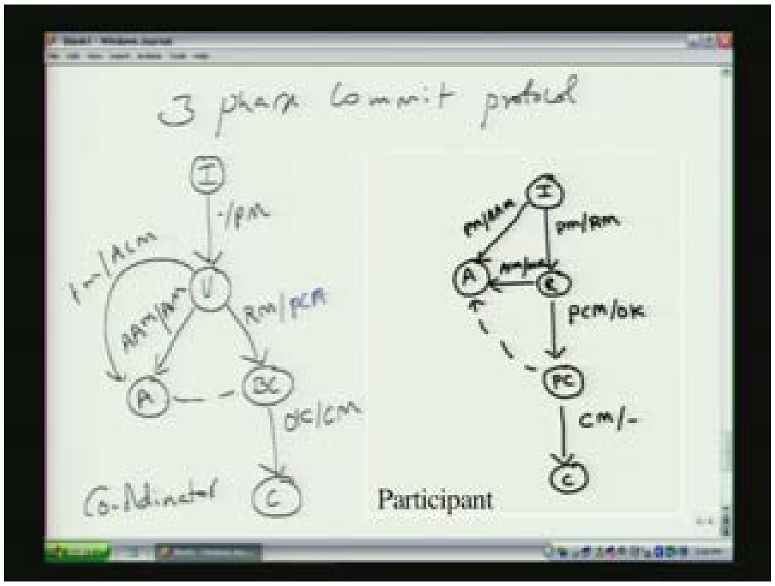
As usual i can also come to the abort state if there is an abort acknowledgement message from one of the participant. Then in response to this, as usual will take the abort command message. It also possible to a timeout message i might reach the abort message. Now this is the diagram for the coordinator. State diagram for the coordinator look something like this participant state diagram. The three phase commit would look something like this. Participant will be initial in the initial state when it receives the prepare message it will reach the ready state, if it is willing to commit the transaction in that case the participant will reach the ready state. In case, it is not willing to commit the message. Then it will answer with an abort answer message and will reach the abort state.

From the ready state also, it is possible for the participant to reach the abort state. It receives the abort command from the coordinator which is possible for the participant to actually reach this state. If there is an abort command message and then the participant reaches that state through the ready state. If the other participants are not willing to commit, the coordinator can take the decision to abort the transaction. In which case, participant will move from the debit state to abort state. If all the participants are ready to commit the transaction which means that, all of them have actually responded ready message then it is for possible for the coordinator to take a commit decision.

In which case in the three phase commit he will initially enter the state called the prepare to commit state, which means that he will prepare to commit message to the participant.
When the participant receives the prepare to commit message, he will reach prepare to commit state. Now he reaches the prepare to commit state is not still commit the transaction is waiting for the final decision coordinator.

When the coordinator receives for all prepared to commit messages, then he will actually issue the commit message. It is a commit instruction in which case the participant will reach the commit state. This additional state is needed because is possible that if the coordinator fails know after one coordinator and one participant fails after reaching the prepare commit state is still possible for the participant to elect a new coordinator in which case it possible that decision to abort has been taken by the new coordinator. In which case after coming back after recovery from failure, it is possible for the participant to move from prepare to commit abort state. Now this is the participant state diagram. We looked at the actual failure scenarios here.

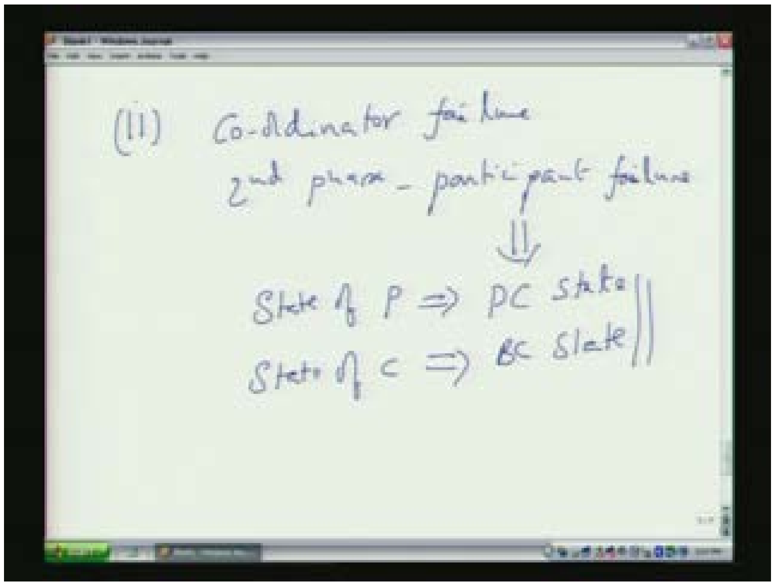
(Refer Slide Time 31.06)



Now the case where the failures can occur as usual participant failing in the first phase second phase are similar. Similarly the coordinator failing in the first phase is also similar. Only the case were we need to look at the second case where the coordinator actually fails that is what we recorded there that means there is a coordinator failure in the second phase and there is also one participant failure or also we have a participant failure. This is the case were we have a problem earlier. Now let us examine this case and see what happens in this particular case in terms of how the recovery takes place in this particular case as the coordinator fails in the second phase along with the participant.

Now in this particular case, there is going to be third phase obviously, because the second phase is not the last phase. Now if the coordinator nobody receives the decision from the coordinator obviously at most, the coordinator is likely to be in the second phase which means that the participant here could have been in the prepared to commit state prepared to commit state it has not yet actually committed. So at most, this state can be prepared to commit, it cannot be commit state. The coordinator would also been this is for participant. This is the state of the participant can be this. Now the state of the coordinator can be also before commit state. Now if these two are in this state, the rest of the participants can still now elect a new coordinator

(Refer Slide Time 33.20)


(II) Co-ordinator failure

2nd phase – participant failure

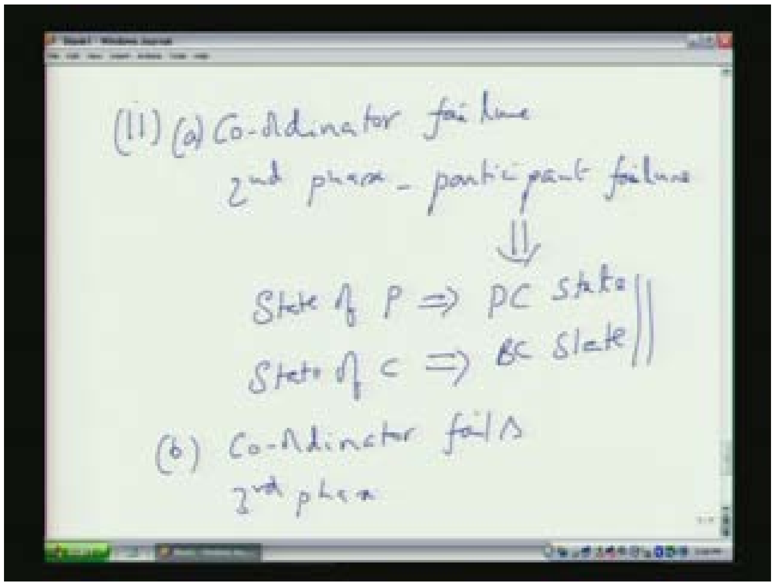
$\Downarrow$

State of P $\Rightarrow$ PC state ||

State of C $\Rightarrow$ BC state ||

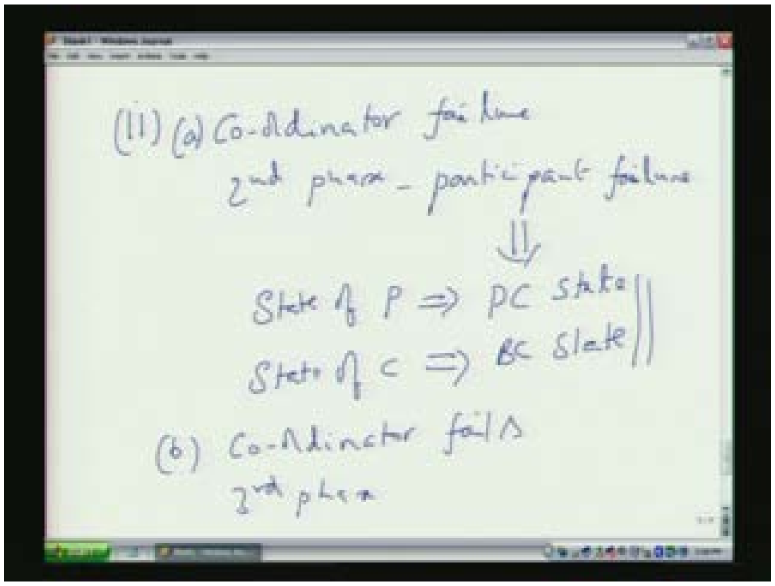
and still go ahead with the protocol, because they have not still committed yet. They have not committed the transaction. Now, in a simple case were the coordinator actually passes the second phase, now you imagine this is going to be a sub case in this particular case where the coordinator has passed in to the third phase means it fail in the third phase not in the second phase. Now this becomes very simple case. Coordinator fails in third phase.

(Refer Slide Time 34.06)


(II) (a) Co-ordinator failure
2nd phase - participant failure
$\Downarrow$
State of P $\Rightarrow$ PC state
State of C $\Rightarrow$ BC State
(b) Co-ordinator fails
2nd phase

This is the simple case because the coordinator actually taken a decision to commit and this is actually conveyed to all the participants. Otherwise, you would not have got him in the third phase. Because unless everybody replied for the prepared to the commit state, he will not reach to the second phase in the commit message which means even one of the participant have received it. Then only, it is cleared that actually moved in to the third phase. Otherwise he is actually not moved in to the third phase. So in this particular case is easy for you just commit the message because you know the decision of the coordinator that it has been actually committed.

(Refer Slide Time 34.58)

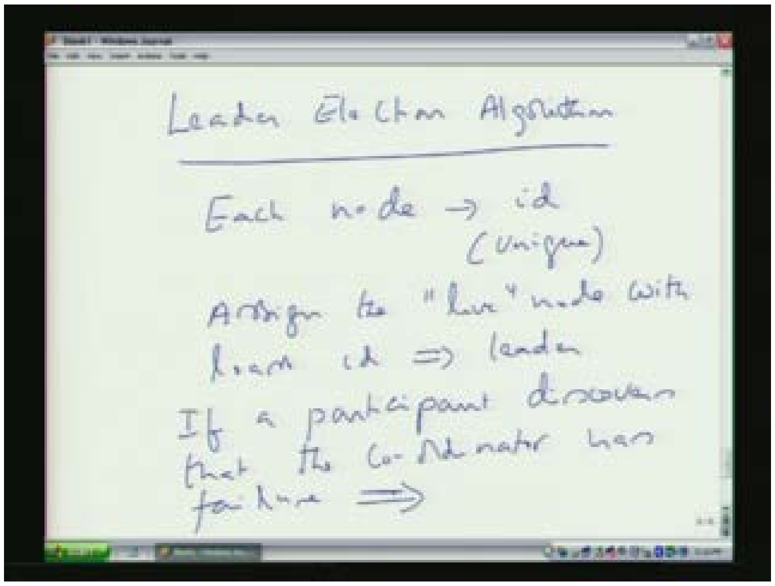

(II) (a) Co-ordinator failure
2nd phase - participant failure
$\Downarrow$
State of P $\Rightarrow$ PC state
State of C $\Rightarrow$ BC State
(b) Co-ordinator fails
2nd phase

So this ambiguity of not knowing the coordinators state when it actually failed is taken off in the case were the state of the coordinator is known by introducing an extra phase was the participant and the coordinator directly do not commit. But they actually reach before state before commit or before prepared to commit state before actually committing and that is how this ambiguity is actually resolved. That is how the failure is handled by introducing an extra phase for commitment. Now one of the interesting thing is, how the election is going to happen because every time, actually seeing that if the coordinator is failing these equivalent to actually some kind of a leader election algorithm Because you are actually electing a leader whenever there is a coordinator failure, there doing a leader election.

In a simpler way, the leader election can be done by a signing in terms of each participant for each node an id and these ids could be in an ascending order. Obviously every node is going to get the unique id here that means there is a unique id node is going to get and you can actually assign at any given point of time, simple way assign the highest of the you can use the node id or highest node id. So basically you can assign the live node with least id as the leader. However, this is not simple because, now you actually find out you are you should take as the leader. Because, somebody has to dictate that the coordinator has actually failed that means what really is going to happen is, if a participant discovers it is going to be only the participant who is going to discover that there is the coordinator failure.

Because he is waiting for some decision from the coordinator that is how the participant discovers that there is the coordinator failure. So what happen is, if participant discovers it can be anybody. A participant discovers that the coordinator has failed. Now what he does is, he has to do an extra. If he basically discovers that, there was the failure of the coordinator, then he has the option of finding out or he has to do this process of finding out, who is the next highest or least node that is still live.

(Refer Slide Time 38.04)



Leader Election Algorithm

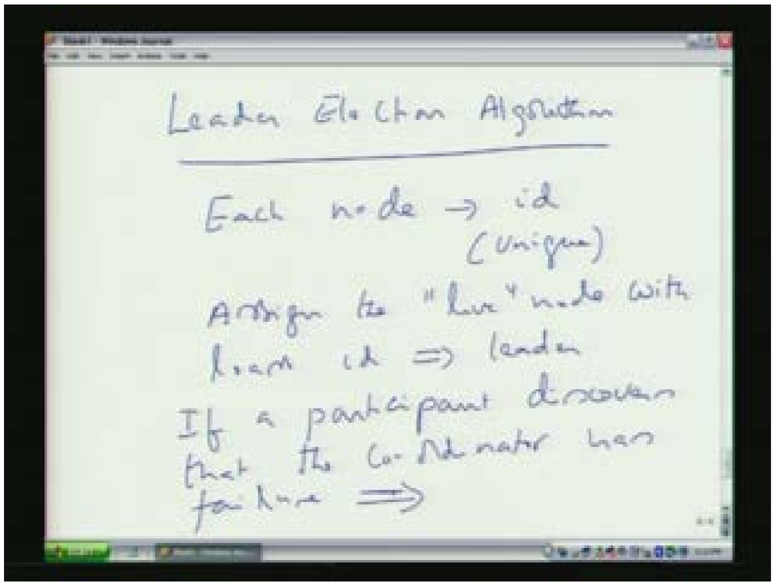
Each node $\rightarrow$ id
(unique)

Assign the "live" node with
least id $\Rightarrow$ leader

If a participant discovers
that the co-ordinator has
failure $\Longrightarrow$

So, he starts now looking for the possible highest least node, live node and tries to know make that node as the coordinator. But then, that node also discovers that the coordinator would have fail participant. So basically all that need is, it has to see whether it has a least node, live node and if it so basically it now elects himself as the coordinator starts running the recovery protocol.

(Refer Slide Time 39.00)



Leader Election Algorithm

Each node $\rightarrow$ id
(unique)

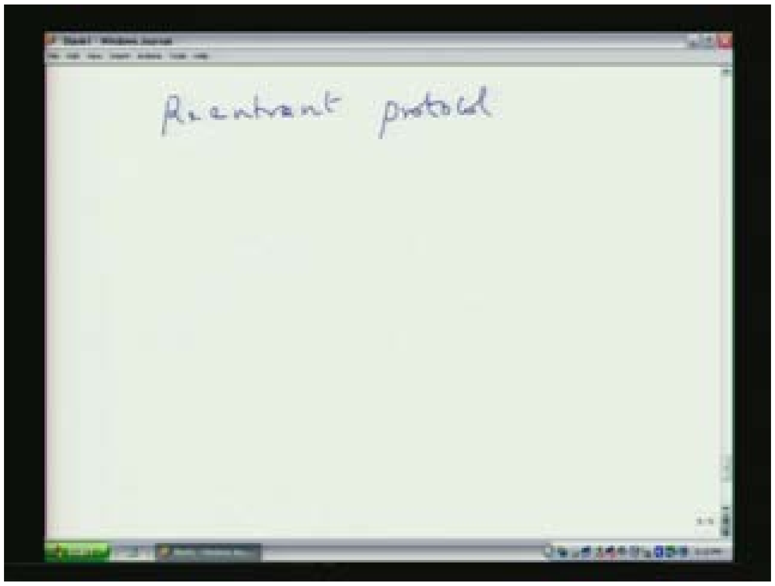
Assign the "live" node with
least id $\Rightarrow$ leader

If a participant discovers
that the co-ordinator has
failure $\Longrightarrow$

So in this particular way, any number of coordinator failures are tolerated by this protocol because all that protocol has to do in this particular case is, make a participant whenever it discovers that there is a coordinator failure, figure out the next highest or least node that is supposed to take over and its turn to take over. It will basically become the coordinator tries to run the protocol. Now, there are two ways. This can be further achieved in terms of how this protocol can be run by the coordinators. One of the things the properties has to ensure that protocol is a reentrancy of a protocol that means the protocol can be run any number of times that means a coordinator fails then a new coordinator takes over.

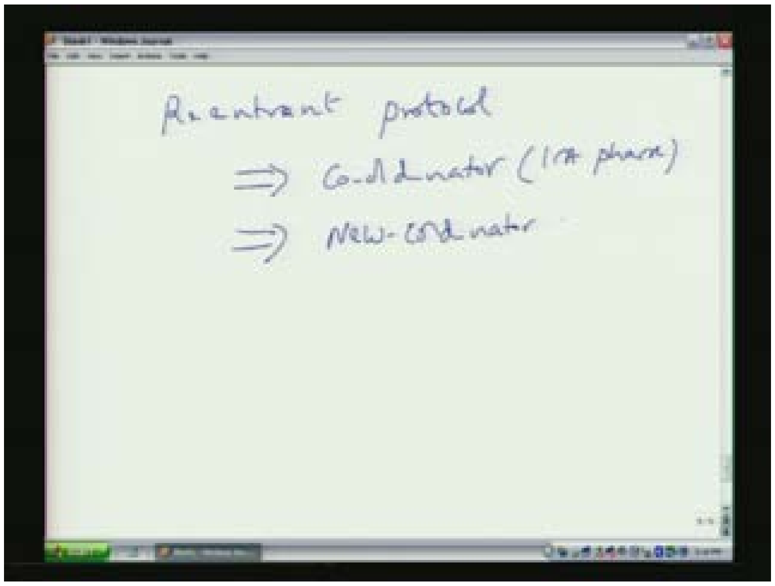
Now, while running the algorithm protocol that coordinator could also have failed which means that another coordinator has to be elected. It has to take over this should become reentrance. That means any number of such failures should be possible when you are actually recovering from the failures in which case basically the protocol is called reentrant protocol. That means it is tolerant to multiple failures of the coordinator any number of times. So, it becomes a reentrant protocol. Now there are two ways the reenter protocol can be configured. One is a proactive way of actually making it move forward the other is basically you know a pessimistic way of reentrance.

(Refer Slide Time 40.35)


Reentrant protocol

This simple way what the coordinator can take as the decision is, if the coordinator is in a state of before commit state, he will basically whatever the state in he basically enforces that on the other people. Let us understand this point a little carefully the coordinator has failed in the first phase which means that node decision has been taken by the coordinator. Now a new coordinator is in place. How this new coordinator is to start the protocol. Now a new coordinator realizes that obviously the new coordinator is not likely to be any other state other than the first phase. He has not received any of the coordinator. So he is going to run the entire protocol staring from the first phase.

(Refer Slide Time 41.45)

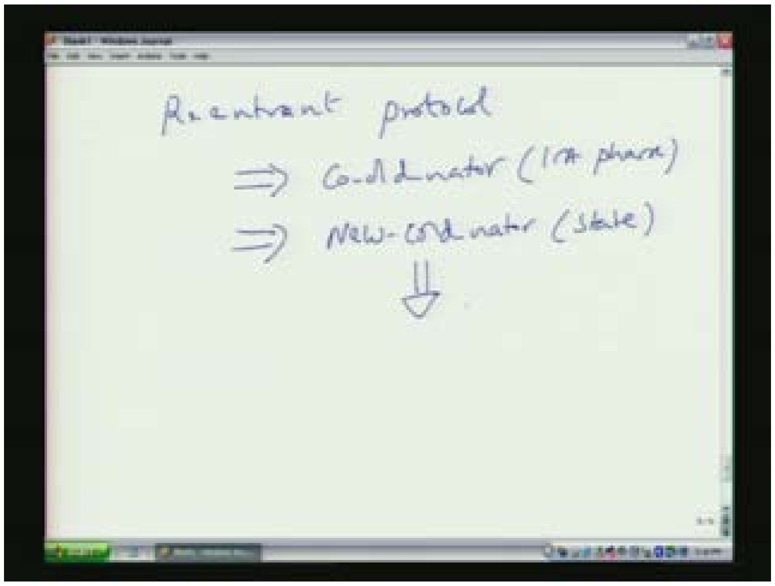

Reentrant protocol

$\Rightarrow$ Co-ordinator (1st phase)

$\Rightarrow$ New-Co-ordinator

Now he actually imitates the prepare message to the other participants receives the ready message then get in to the second phase then goes in to the third phase things like that. Now the new coordinator can be further improve the protocol is, whatever he is state he is in he can actually find out if there is any other participant who is in a before commit state which actually means that now he can actually need not restart the protocol, but then he can knows that actually the decision of earlier coordinator is actually to commit the message.

(Refer Slide Time 42.22)


Reentrant protocol

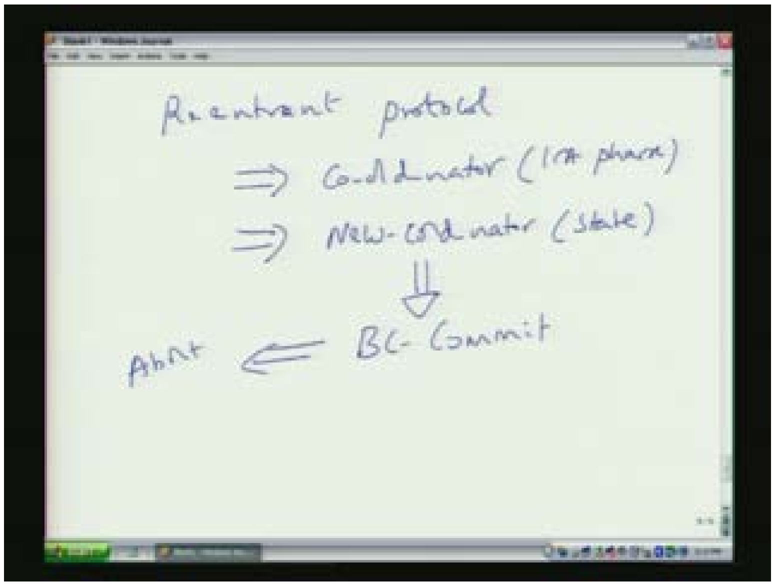
$\Rightarrow$ Co-ordinator (1st phase)

$\Rightarrow$ New-Co-ordinator (state)

$\Downarrow$

Now you could actually when there is a before commit state, it is possible for the protocol to terminate as an abort because the new coordinator now ignore this decision but then just conducts the sends a message to the all the participants. Now some of them not going to reply back to him then he can actually enter the abort decision is quite possible for the new coordinator to find out if one of the participant has reach the commit state. In which case, actually it can force the protocol to a commit state.

(Refer Slide Time: 42:51)



Reentrant protocol

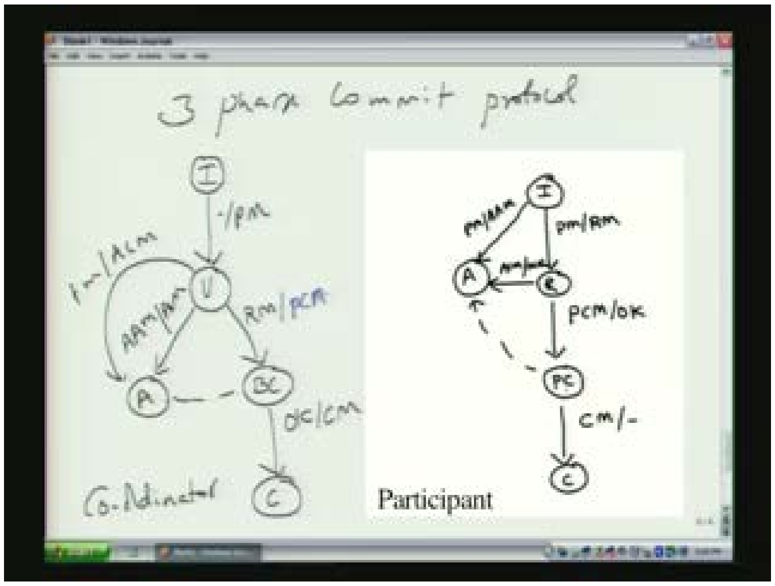
$\Rightarrow$ Co-ordinator (1st phase)

$\Rightarrow$ New-cordinator (state)

$$\begin{tikzcd} \text{New-cordinator (state)} \arrow[d, Rightarrow] \\ \text{Abort} \quad \text{BC-Commit} \arrow[l, Rightarrow] \end{tikzcd}$$

To just explain this a little more probably in a using our diagram, you can see here the coordinator actually the new coordinator who has elected.

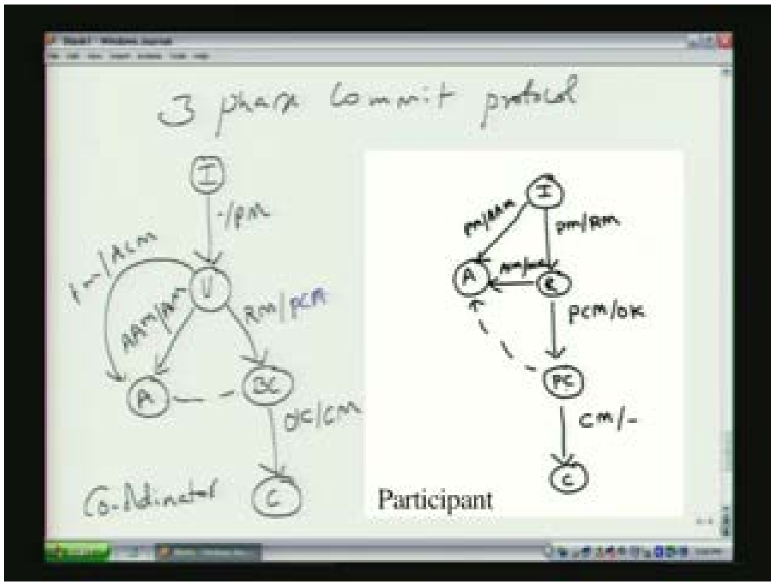
(Refer Slide Time 43.30)



Now if you basically dint receive the prepare to commit message which means that he will be in a before state he is still not received this message which could have made the whole protocol terminated the abort decision. Now, if you actually checks with the other live participant and at least one of them has actually receive this PCM state, but is not the new coordinator. This some other has been elected as the coordinator now. Then, it still can take a commit decision because everybody would have reached a state.

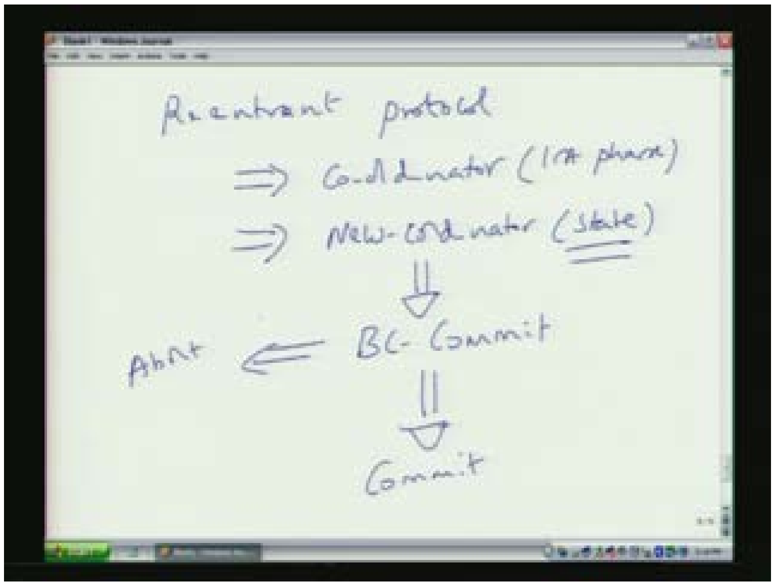
So it is possible for unless all of them replied with a ready message. This state would have not reached. Right, now the new coordinator can take a decision to commit a transaction and then make it know. Basically, all that required for the elected coordinator in this case is not only check its state, but then checks the state of all live participant based on that restart the protocol at an appropriate point.

(Refer Slide Time 44.29)



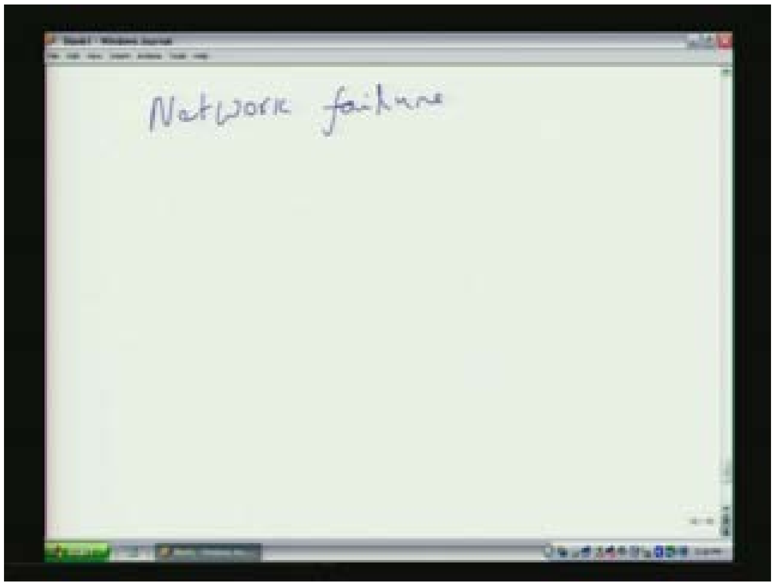
It is done like that the reentrant protocol terminates in a forward direction in terms of committing wherever there is a possibility of a protocol to commit it will commit rather than always sending out of the abort state, depending on the state of the new elected coordinator.

(Refer Slide Time 44.47)



So these two approaches are possible in terms of recovery from the failures as far as the coordinator is concerned. I think what we are going to do in the next two minutes is, not only look at this scenarios of failures but then also introduces now, the concept of what happens with the network failure, because you are now talking about only node failures so far, we are not talking about network failures as part of the recovery process. Now in terms of network failures as far as the protocol is concerned, the two ways of actually interpreting the network failure:

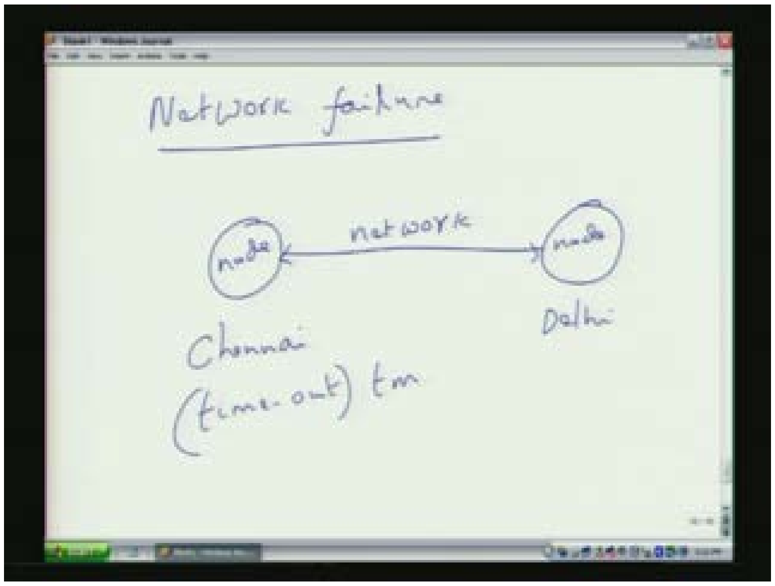
(Refer Slide Time 44.51)



Network failure actually means that the nodes are not reachable. So let us take for example a node here. This node could be a Chennai node and this other node could be Delhi node. Now all that you are looking at in terms of this system is, both of them are connected by the timings of some, let us say back bone network that could be several network. It involved connectivity with been these two nodes. All that you have is basically a network, which is actually providing this connectivity.

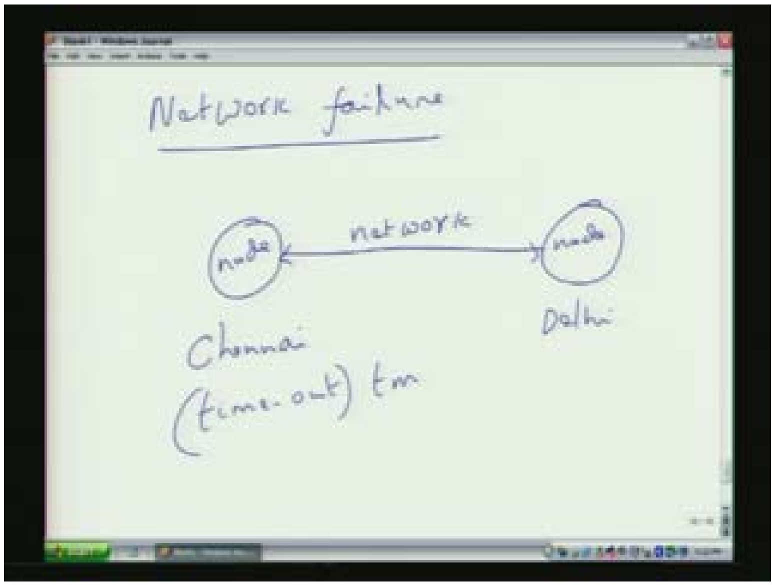
Now when the network actually fails, in a simple case this actually surfaces as the time out on the other side that is how actually you are going to discover. You are waiting for a reply and this scenario actually come as the timeout in our diagram. As we saw earlier and that timeout could be possibility.

(Refer Slide Time: 47:00)



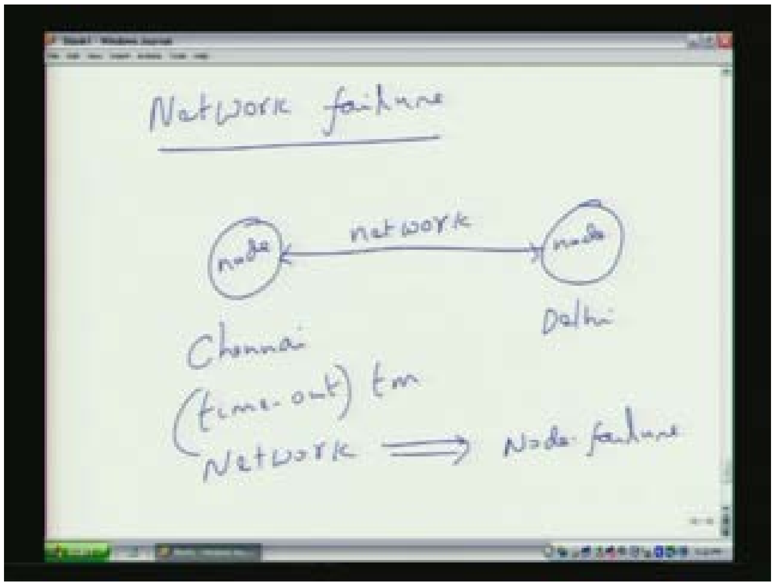
Because the network has actually failed not the node has fail. In either case it wont be able to detect the difference between a network failure and a node failure because the node as far as for example for this Chennai node is concerned it does not receive the reply in time from the Delhi node it could be network failure it need not be a node failure. The node can be still be kicking doing some work there, but when it actually sends a message, let us say ready message.

(Refer Slide Time: 47:24)



Chennai node does not receive the ready message. This is equivalent to actually a node failure that means a network failure in some sense actually translates to a node failure. The simple case this translates to a node failure.

(Refer Slide Time 48.00)

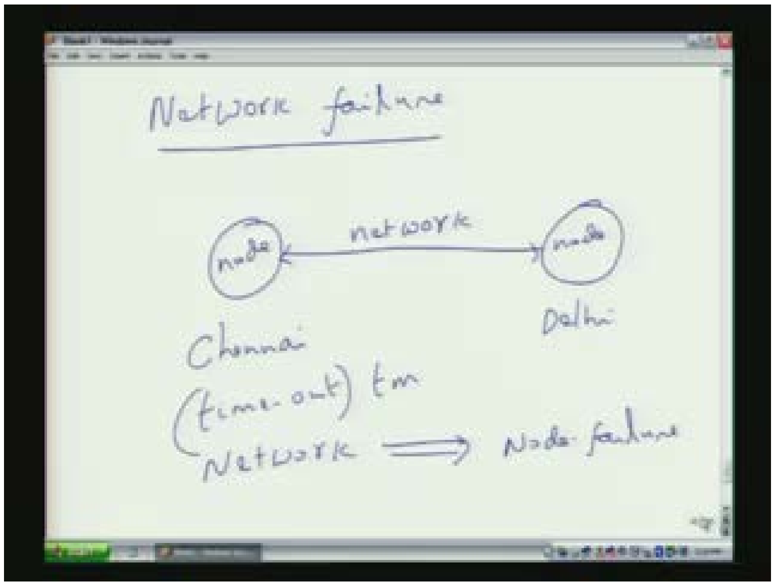


But then, it is probably not appropriate. Just consider a network failure is just a node failure. In our case it still works, because when the timeout actually reads out the other side all that you are assuming now is, the decision of the node is to abort. So nothing no harm is done in this particular case but could have done positive thing is ends up in the negative side because you could have still commit the node has still responded with a ready message.

But since it does not reach the other node and it decided that when it did not have any information that best thing it can do is abort the message or abort the transaction that's what actually assumes then abort. So basically a network failure and node failure in a normal case both appear to be treated in this same way right and that is how this protocol both two phase commit and three phase commit are resilient to network failures in terms of translating them to the participant failure.

But one has to be careful when the network actually fails, but then that failure is not just to one failure of one node but it can result in not able to reach to multiple nodes. So typically this scenario is a very simple case where it has been translated to a timeout case but then assume that a network failure actually results in multiple nodes not being reachable.

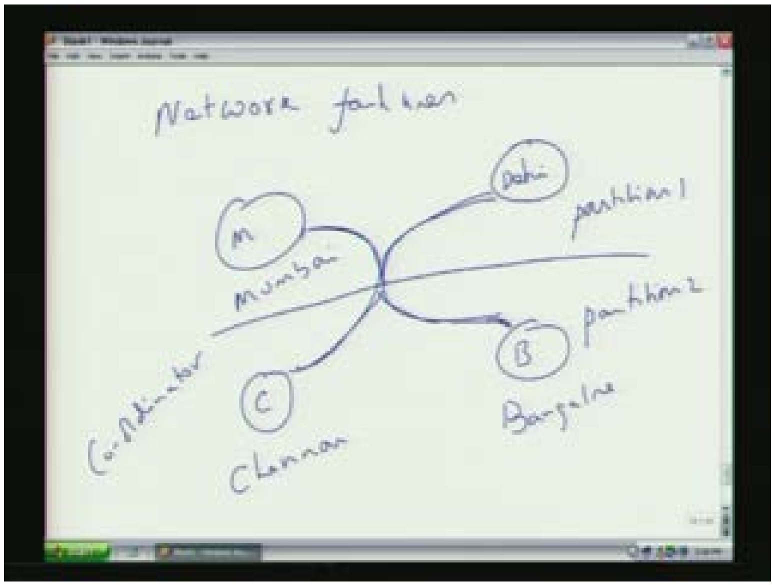
(Refer Slide Time 49.43)



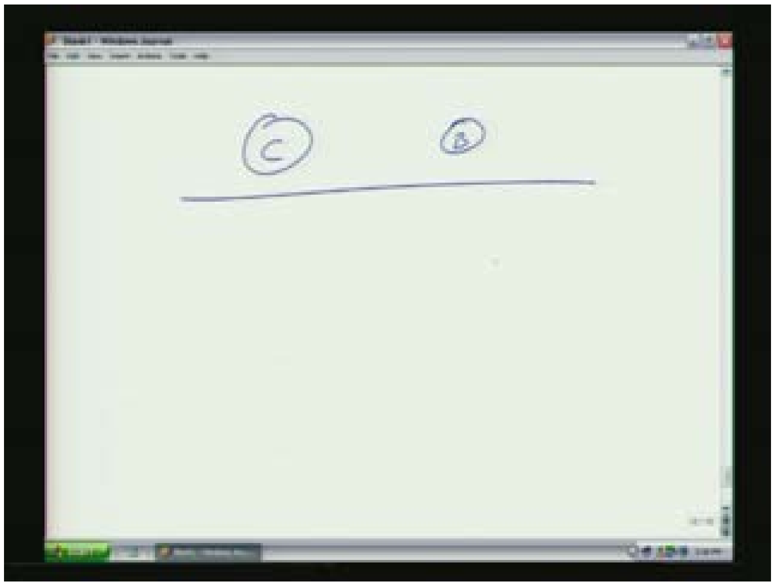
Now, let us imagine we have node in Chennai. Let us say, this is Chennai node and then we have actually Bangalore node and then we also have Mumbai node here and then we have one of Delhi. Now it is possible that a network actually connecting them. Let us assume that complicated network here which actually connects all the nodes with one another. But then, if there is actually some kind of hub that connects the southern nodes together and the northern nodes together, it is quite possible that this hub actually fail which means that these nodes are can still each other in this direction and these nodes can reach each other in this direction.

Now assume that Chennai is the coordinator. Now as far as Chennai is concerned, the whole problem translates into failure of both Mumbai and Delhi, because it still reachable. Bangalore is still reachable and let us say, it replied with whatever required messages. So, this basically results in partitioning of the whole set of nodes. Now you have actually two partitions. There is one partition, one here which consisting of Mumbai and Delhi and there is another partition two consisting of Chennai and Bangalore.

(Refer Slide Time 51.40)

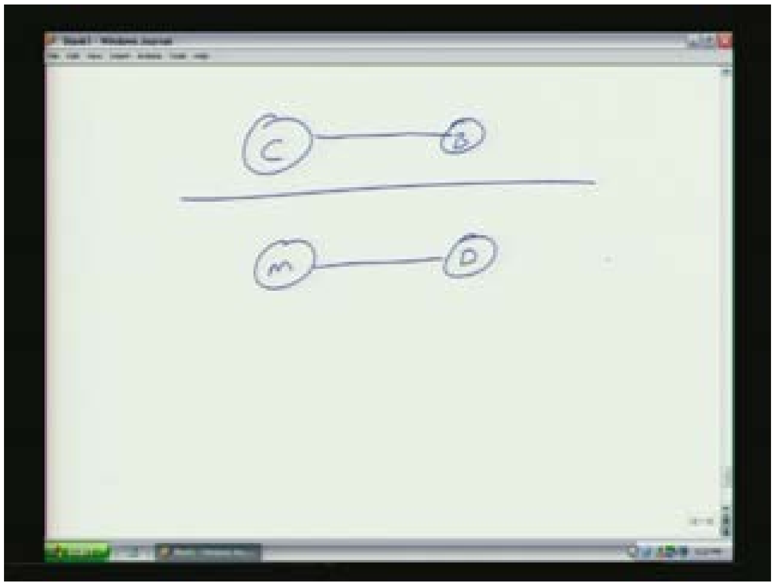


(Refer Slide Time 51.56)



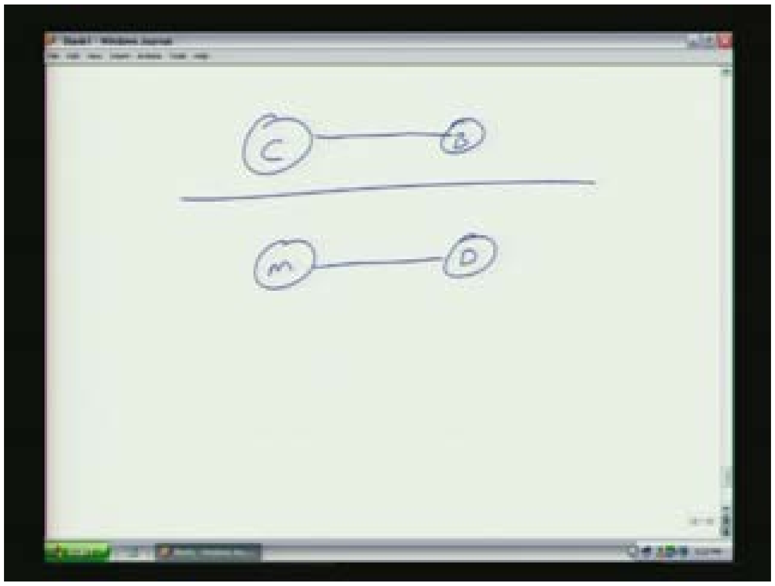
Now it if you carefully looked at this scenario and try to understand what could have happen in this, it is possible that Chennai and Bangalore could start detecting that there is participant failure which means that two participants have failed and two phase commit is resilient to participant failure. So what basically they do is depending on which phase you are in, you might end up actually making the transaction. So basically these two might right to run this protocol saying that there are two participants failure.

(Refer Slide Time 52.29)



Now as far as Mumbai and Delhi are concerned, for them it translates to one coordinator failure one participant failure. They also try recovering back from this problem by saying that now they will have a new coordinator depending on what phase they are in and how they should be handling this problem. They might actually try recovering back from this failure. Now in this case, it's possible that the decision of two partitions need not be consistent and this is happening mainly because the nodes are delivering it is a network failure. They are seeing them as failures of the participant nodes.

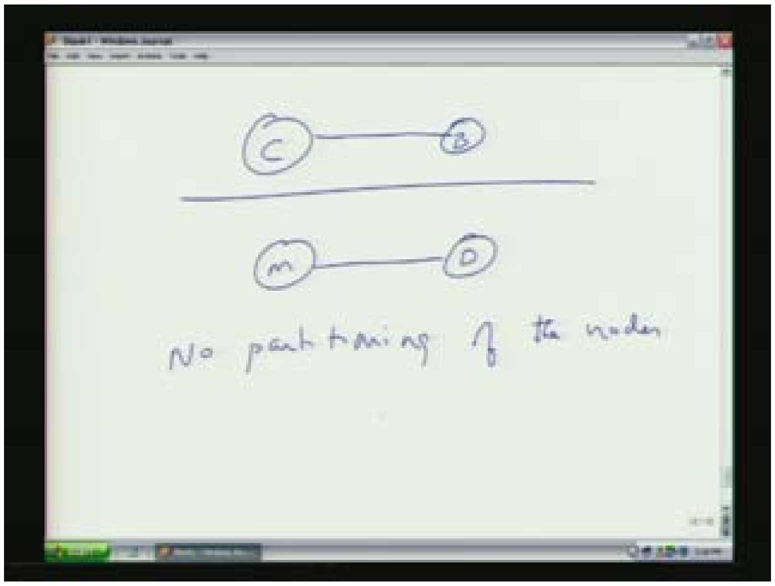
(Refer Slide Time 52.58)



The participating nodes are still live and they have working. So for each one of them scenario is different unless this is discovered that is network failure not a node failure you would not be actually making an distinction network failure or the node failure and this really results in a complicated situation were even if you use a three phase commit protocol were the network actually partitions the nodes into multiple partitions, you wont be able to recover back from the problem.

In other words, you are making an assumption here is there is no partitioning that is happening this is the case no partitioning of the node and this is an assumption with your actually working on the two phase commit protocol. If the network partition you have the problem of recovering back from the failure because all the case which discuss which we looked at are actually taking them participant failure and not really participant failures and it is a network failure basically will have a difficulty.

(Refer Slide Time 54.18)



An interesting assignment could be what are those cases were the network partitioning network partitioning can lead to problem which is the case in which the protocols the states of the coordinator and the participants in which case the partitioning could really be a problem. The other thing is to discover if you are the majority partition. Probably if you are the majority partition, probably you can still go out do something which means that you have to discover that the system has actually partition and whether you are the majority partition and based on that probably can recover out of the failures. It is still a very open ended problem in terms of how one can recover from network partitioning. It is all interest in see how one can make a distinction between network partitions and node failure.

For example: one way is to actually, if you just do a thing you only know node whether the node is live or not, that’s still remain the interesting problem in terms of seeing how one can make a distinction between a node failure and a network failure in a distributed system. Right now, what we have seen is the basic commit protocol in the next class what we are going to see is, how this commit protocol can be integrated with concurrency protocol. For example: how the two phase locking can be integrated with two phase commit protocol that’s going to continue with the next class.