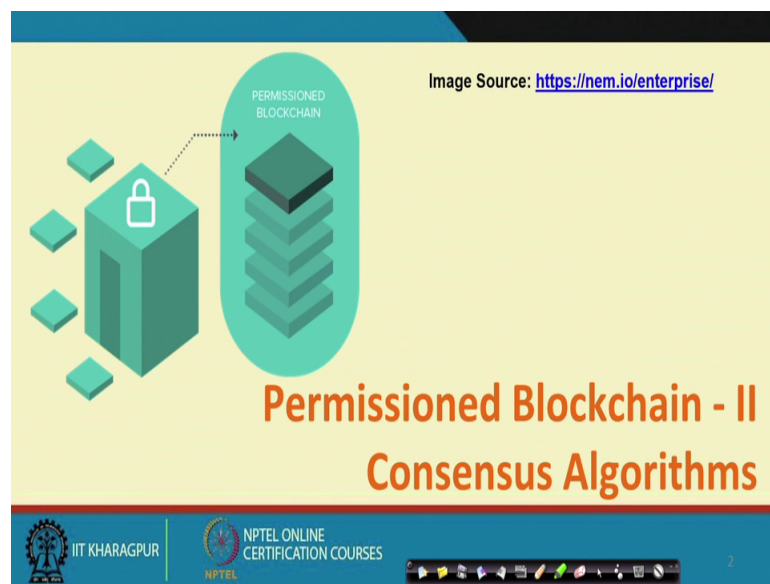


Blockchains Architecture, Design and Use Cases
Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 15
Permissioned Blockchain – II (Consensus)

So, welcome back to the course on Blockchain. So, in the last class we are looking into the permission blockchain settings.

(Refer Slide Time: 00:25)

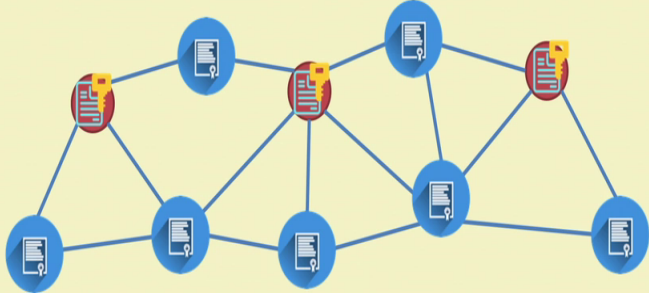


And, in the context of permission blockchain, the concept of state machine replication which will help you to achieve a consensus in a permission model.



(Refer Slide Time: 00:37)

State Machine Replication

- **Use state machine replication** – execute contract at a subset of nodes, and ensure that the same state is propagated to all the nodes



The diagram shows a network of 10 nodes connected by lines. Three nodes are highlighted with a red and yellow icon, representing the subset of nodes where a contract is executed. The other seven nodes are represented by blue icons with a document symbol, representing nodes that receive the propagated state.

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

So, in the last class we are looking into this concept of state machine replication, where we have seen that well you do not need to execute a smart contract to all the nodes, rather you can execute it on a subset of nodes. And then you can ensure that the state of the contract is getting propagated to all the nodes in the network.

And there are certain consensus mechanisms, which will ensure that well the states which have been propagated by multiple state machines, or the contract executer that they are in that on the same page, or there in that correct. So, by applying this kind of distributed state machine replication technology you can ensure consensus in a permission blockchain environment.

(Refer Slide Time: 01:26)

State Machine Replication

- **State machine**
 - A set of states (S) based on the system design
 - A set of inputs (I)
 - A set of outputs (O)
 - A transition function $S \times I \rightarrow S'$
 - An output function $S \times I \rightarrow O$
 - A start state

Image source: commons.wikimedia

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, let us look into that what is meant by a state machine replication. So, a state machine can be characterized by a set of parameters. So, the set of parameters are a set of states based on the system design. So, here in this particular example you have three states S_1 , S_2 and S_3 , then you have a set of inputs, which will tell you about how the your system will behave.

So, here are two inputs zero can be one input to this system and, one can be another input to the system. Then you have a set of outputs. So, here S_3 is the final output of the system, that that is represented by the state machine, a transition function. So, the transition function will take a state and then input, a set of input and it will produce a set as the output.

So, here from state S_1 if you take 0 as an input, it produces state S_2 as the output. So, this is a transition function, then you have output function which may produce the output of the system. So, this particular example we do not have any output, but it may happen that well.

(Refer Slide Time: 02:42)

State Machine Replication

- **State machine**
 - A set of states (S) based on the system design
 - A set of inputs (I)
 - A set of outputs (O)
 - A transition function $S \times I \rightarrow S$
 - An output function $S \times I \rightarrow O$
 - A start state ✓

Image source: commons.wikimedia

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

If you have S 1 and input as 0 you are moving to state S 2 that is your transition function and at the same time you may producing as output 0. So, that that way for certain state machines you may have a output function and, there would be a designated start state.

So, here S 1 is denoted as the start state of the system. Now, any algorithm any finite state machine, which represents an algorithm, that can be represented using a finite state machine.

(Refer Slide Time: 03:20)

Smart Contract State Machine - Crowd-Funding

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, let us look into one example, where you can represent a smart contract as a state machine. So, earlier we are talking about this kind of crowd funding platform. So, in a crowd funding platform, you have a set of fund set of people who have certain funds available, who are agree to donate the fund if certain jobs are being done.

So, there are the project proposals, which are coming from the proposal. So, the proposer propose certain jobs certain projects. And if certain funding agencies or funder, they find that particular proposal interesting, they may fund for the proposal in a collective way. So, that is a kind of crowd funding platform like a kick started with example, we have looked sometime back.

So, let us look into that how we can represent the crowd funding application, in the form of a state machine. So, here the system has a initiation state, where some users they have submitted certain proposal. Now, that proposal wants this someone like this proposal they may transfer the money. So, here say Bob has transferred the committed money to the system. So, you are reaching to a state that Bob has donated the money. Similarly if Alice has transfer the committed money, you are reaching a state that Alice has donated the money.

Now, after Bob has donated the money Alice may donated the money after that, or it may happen that Alice has first donated the money and, after that Bob has donated the money. Now, when both of them has donated the money either in the order of Bob Alice, or Alice Bob, you have the money available in the system. Now, once you have the money available in the system, you can initiate the job and say there are 2 jobs that need to be executed, but there is no specific sequence in which the jobs need to be get executed.

So, it may happen that the job 1 is first complete and, then the job 2 is complete. So, then you follow this path to make both job 1 and job 2 complete, or it may happen that the job 2 is first get finished. After that job 1 is get finished. So, in this particular order you may again have this job 1 and job 2 get complete. So, once both the job done, then your smart contract will transfer the money from the funders, which are here the Bob and Alice to the project proposal say, if you has transferred that project proposal the money will be transferred to if.

So, that way you can represent a smart contract in the form of a state machine in that, any algorithm you can represent it in the form of a finite state machine.

(Refer Slide Time: 06:07)

Distributed State Machine Replication

1. Place copies of the state machine on multiple independent servers

The diagram illustrates a distributed system with three servers connected to a central cloud. Each server has a copy of a state machine with four states (S1, S2, S3, S4) and transitions (a, b, c, d, e, f, g). The state machine is shown as a directed graph with states as nodes and transitions as edges. The servers are represented by server racks. The cloud is represented by a blue cloud icon. The diagram is part of a presentation slide from IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Now in a typical distributed architecture the distributed state machine replication mechanism works in this way. So, you have multiple servers, which works in a distributed fashion. So, the advantage of using distributed servers in place of a centralized server that we discussed earlier. So, the idea is that first you have multiple servers here. So, you place the copies of the state machine on each of this server. So, each of this server has a copy of this entire state machine, then servers get the client request.

(Refer Slide Time: 06:42)

Distributed State Machine Replication

2. Receive client requests, as an input to the state machine

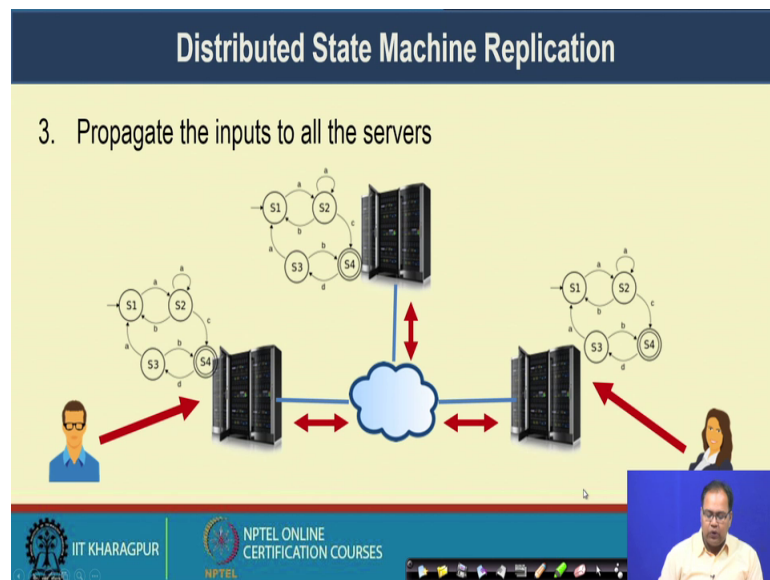
The diagram illustrates the same distributed system as in slide 1, but now with client requests. A man icon on the left sends a red arrow (request) to the left server, and a woman icon on the right sends a red arrow (request) to the right server. Blue arrows indicate the network connections between the servers and the cloud. The state machine diagrams are also present on each server. The diagram is part of a presentation slide from IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, here it is like that Bob is submitting his request and, Alice is submitting her request and the request goes to different servers. Now, when the request are going to the different servers. So, if you execute the state independently at its server the state that will be there in this server and the state, that will be there in this server they may be different. So, here you will reach to a state that Bob has transfer the money here you have reach the state that Alice has transfer the money.

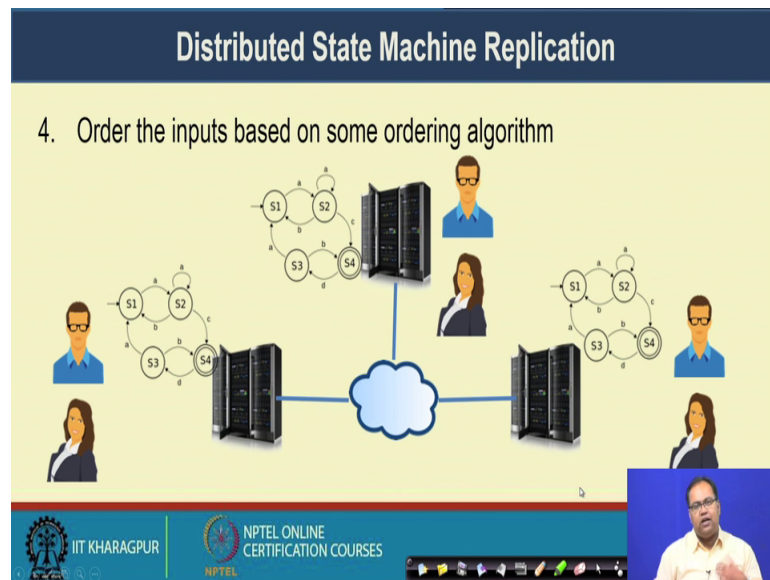
But, collectively you have to ensure that all the 3 servers which is to a state after certain time that both Bob and Alice has transfer their share of the money. So, to achieve this the state machine replication mechanism works in this principle that you propagate this inputs to all the servers.

(Refer Slide Time: 07:34)



So, all the servers has the input like that Bob has transferred the money and Alice has transfer the money.

(Refer Slide Time: 07:44)



Then you order the inputs based on certain ordering algorithm, you can have a associated timestamp with every individual transaction say, whenever Bob is making his transaction it is associated with the timestamp and, when Alice is transferring her share of money, that particular transaction is associated with another timestamp.

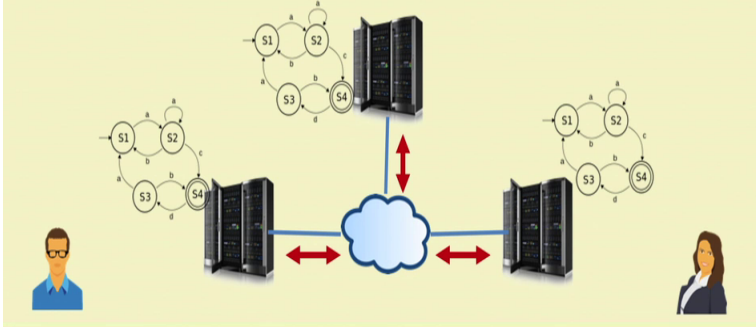
So, based on the timestamp you can have an ordering of the transactions and, then once this ordering is done, then every individual system they execute the inputs based on the orders which have been decided.

And they execute it individually at each server. Now, once this information is executed in each server in that particular ordering algorithm, then you know that well first the transaction corresponds to Bob that will get executed, then the transaction corresponds to Alice that will get executed and all the servers reaching to the same state that both Bob and Alice has transfer their share of money.



(Refer Slide Time: 08:49)

Distributed State Machine Replication

5. Sync the state machines across the servers, to avoid any failure.



The diagram illustrates a distributed system with three servers connected to a central cloud. Each server has a state machine with four states (S1, S2, S3, S4) and transitions labeled a, b, and c. Red double-headed arrows indicate synchronization between the servers and the cloud. A person icon is on the left and a woman icon is on the right.

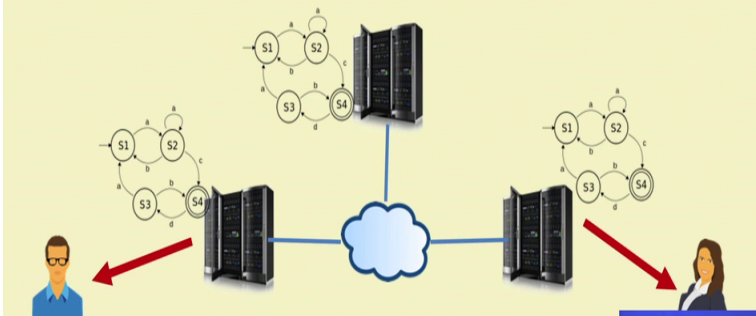
So, once this transaction is done, then see seeing the state machines across all the servers to avoid any failure, it may happened that this server during the say this server during the execution during the execution it has failed and after sometime it recovers.

So, once it recovers it should get this updated speed, that will both Bob and Alice has transfer their share of the money. So, that update is done through synchronization of the state machine. Now, here in this entire algorithm.



(Refer Slide Time: 09:22)

Distributed State Machine Replication

6. If output state is produced, inform the clients about the output



The diagram illustrates a distributed system with three servers connected to a central cloud. Each server has a state machine with four states (S1, S2, S3, S4) and transitions labeled a, b, and c. Red arrows point from the servers to the client icons (a person on the left and a woman on the right), indicating that the output state is being communicated to the clients. A person icon is also visible in the bottom right corner.

We have we have certain outputs that can be produced like, it may happen that well the output is when both the share of the money has been transferred, you send a output or you notified the client that well the job is now ready to get executed. So, that particular output is send from the server to the individual users.

Now, in this entire procedure there are two glitch; the first one is that you need to maintain order in service. And, second is that in the presence of a failure you need to ensure that, all the individual severs are in the same page, everyone knows that well. Both the transactions from Bob and Alice, they got excited and the entire money has been transferred. Now, to ensure that you need to you need to have a kind of consensus algorithm in the system.

Now, why do we apply this kind of state machine replication based consensus algorithm in a permission model, in contrast to the challenge response based consensus model, which we apply for the permission less settings. So, there is certain natural reasons to use state machine based a consensus algorithms over permission blockchain.

(Refer Slide Time: 10:32)

Permissioned Blockchain and State Machine Replication

- There is a natural reason to use state machine replication based consensus over permissioned blockchains
 - The network is closed, the nodes know each other, so state replication is possible among the known nodes
 - Avoid the overhead of mining - do not need to spend anything (like power, time, bitcoin) other than message passing
 - However, consensus is still required - machines can be faulty or behave maliciously

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, the first reason is that the network is closed. So, when the network is closed every individual nodes they know each other.

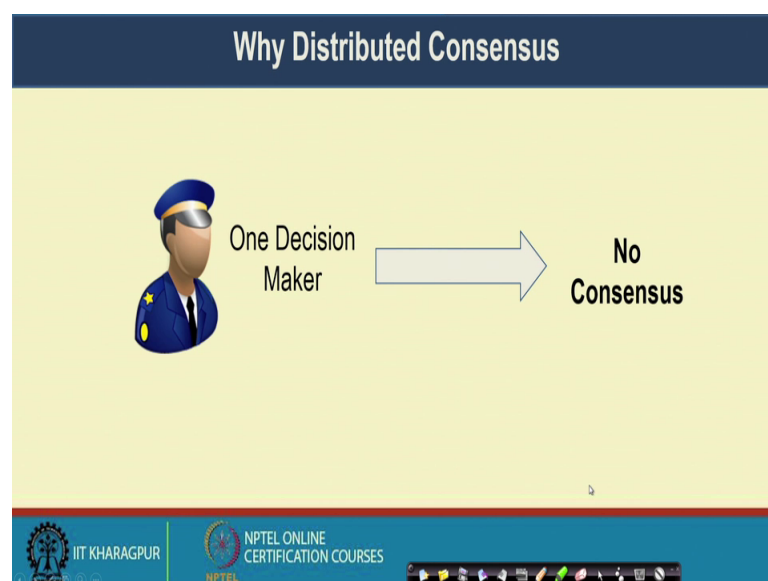
So, state the replication is possible among the node nodes. So, if you know that who are your peers you can always replicate, your state machine with the current state to your

peers, the second thing is that this particular state machine replication based scenario, it avoids overhead of mining. So, mining has a significant overhead in terms of the system power that you are using, in terms of the time that you are providing. And a amount of bitcoin that you are you may apply for this kind of a proof of state based mechanism. So, or proof of elapsed time based mechanism.

So, it is like that either in case of a challenge response base method, either you have to spent your physical assets, or you have to spend your digital assets. And it will take certain time to solve that particular challenge. So, that has significant kind of overhead in case of a permission less setting, but in case of a permission model because we know the participant each other and, in case of a state machine replication you do not have this kind of overhead, where you have to spend, your physical asset or some kind of virtual assets rather, what you have to do that you have to imply certain kind of message passing algorithm in the system. So, you can avoid the overhead of mining by applying this kind of state machine replication based method.

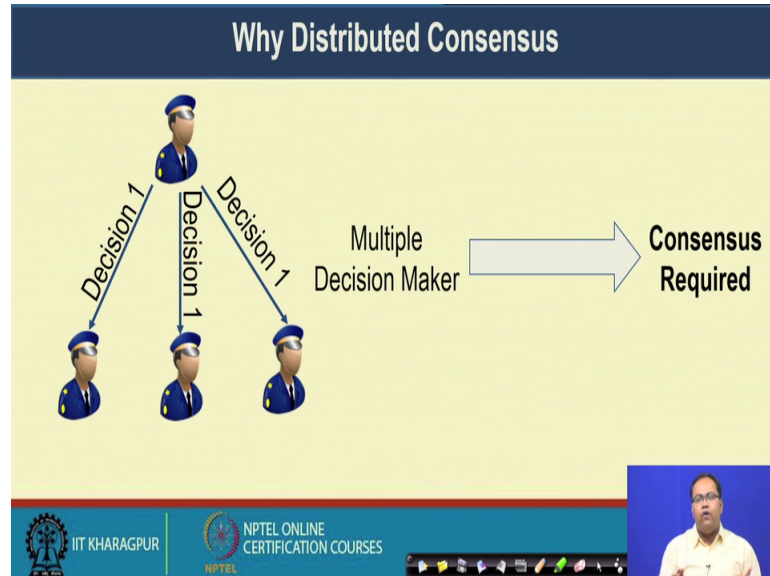
But as I have mentioned that consensus is still required on top of this state machine replication because the machines can be faulty or they can behave maliciously. So, let us quickly go to the consensus algorithms. So, earlier you have looked into this concept of distributed consensus.

(Refer Slide Time: 12:20)



So, in case of consensus algorithms well, if there is only single decision maker you do not require any consensus, whenever there are multiple decision makers.

(Refer Slide Time: 12:35)



And in a collective way they want to come to certain decision, then you require the consensus.

(Refer Slide Time: 12:43)

The slide lists the following points under "Why Distributed Consensus":

- Reaching agreement in distributed computing
- Replication of common state so that all processes have same view
- Applications:
 - Flight control system: E.g. Boeing 777 and 787
 - Fund transferring system: Bitcoin and cryptocurrencies
 - Leader election/Mutual Exclusion

A hand-drawn network diagram is shown at the bottom right, consisting of several nodes connected by arrows, with one node highlighted in red. The slide footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of the presenter.

So, the distributed consensus it helps you to reach in a agreement distributed computing. So, in case of in terms of the state machine replication concept you replicate the common states. So, that all the processes they have the same view of the state, they can

understand. So, if I just draw simple state machine that so, this particular graph can give you a simple state machine and, and say this is the final output state, this is the initial state and there are say individual inputs and, from who are here the system reaches to the state.

Now, it may happen that the system has reached to some intermediate state say the a system has reached up to this state. If the system has reached up to this state, then you need to ensure that all the processes all the users were, there in the system they have the same due date get the same view that the system has reached to this particular rate state.

Now, there are multiple applications like, one typical application of this state machine replication is the flight control system, when there are multiple flights and they want to coordinate their positions among themselves, you can apply this kind of state machine replication technique and distributed system to achieve consensus, then for fund transferring system in a distributed environment like for the crypto currencies.

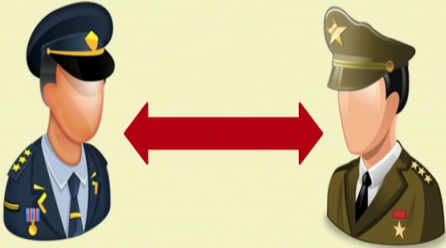
We are going to see that well in a closed environment, you can apply state machine replication based algorithm for a consensus in a open environment, we have looked into that well you can have a challenge response based consensus algorithm to achieve the consensus.

Then for certain other distributed applications like a distributed leader election, were all the nodes collectively need to elect one leader in the system for the kind of applications, you have to ensure certain kind of consensus that all the nodes elects same leader, or the same leader is elected at the end of the round for, this kind of agreement protocols were the nodes collectively needs to come to certain agreement, we require this kind of a distributed consensus algorithms.

(Refer Slide Time: 15:08)

Why Distributed Consensus

- So, no need of consensus in a single node process.
- **What about when there are two nodes?**
 - Network or partitioned fault, consensus cannot be reached



The slide features a dark blue header with the title 'Why Distributed Consensus' in white. Below the header, on a light yellow background, are two bullet points. The second bullet point is bolded and followed by a sub-bullet. Below the text is an illustration of two military officers in uniform, one in a blue uniform and one in a green uniform, facing each other with a large red double-headed arrow between them. At the bottom of the slide, there is a blue footer containing the IIT Kharagpur logo, the text 'IIT KHARAGPUR', the NPTEL logo, and the text 'NPTEL ONLINE CERTIFICATION COURSES'. A small navigation bar with various icons is also visible in the bottom right corner of the footer area.

Well we have looked into that we do not need a consensus in a single node process, but when there are two nodes. So, in case of two nodes, if there is kind of fault if the network is fault, or even if there is a kind of crash fault, even if the node behaves maliciously you cannot reach consensus to reach consensus, you always require more than two nodes.



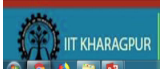
Now, this things you have look details earlier in the context of permission less blockchain, that in a distributed environment you can have primarily three types of fault, the first fault is the crash fault where a node crashes and is not able to send any message.

So, it may recovered after sometime and it may it may start sending the message again. So, this call as a fail stop behavior, then you can have the network or the partition fault, where because of a network fault that entire network gets partitioned. And the message from one partition is not able to get propagated to another partition; the third type of fault which is the difficult to handle in a distributed environment is a byzantine fault.

(Refer Slide Time: 16:18)

Faults in Distributed Consensus

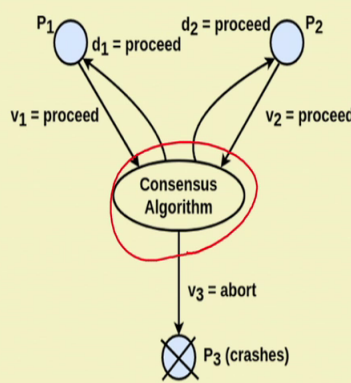
- **Crash Fault**
- **Network or Partitioned Faults**
- **Byzantine Faults**
 - malicious behaviour in nodes
 - hardware fault
 - software error



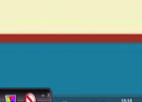


So, you have kind of malicious behaviors among the nodes, which may come due to certain kind of hardware faults or software faults. So, this byzantine fault is that for certain views, the node is behaving in one way whereas, for certain other views the node is behaving in a different way.

(Refer Slide Time: 16:43)

Consensus for three processes



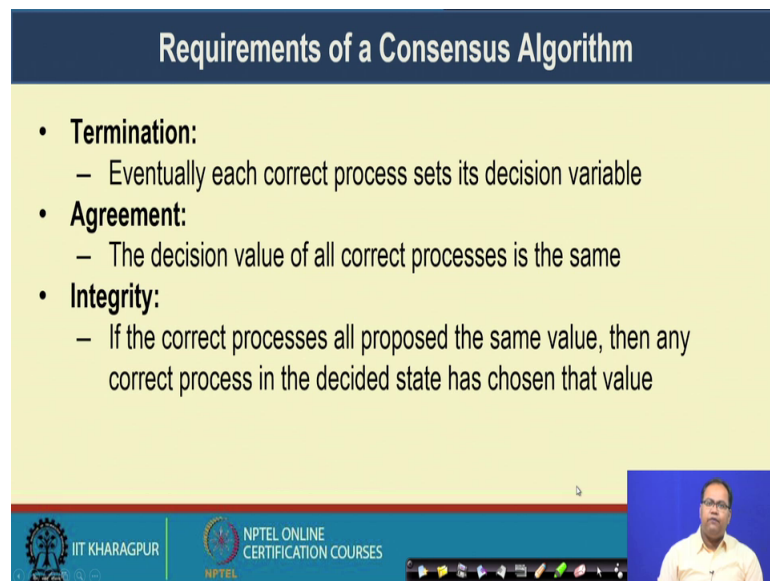
- Each process P_i ($i=1,2,\dots,N$):
 - **Undecided state**: proposed value v_i from set D
 - **Communication state**: exchange values
 - **Decided state**: set decision variable d_i



So, to reach the consensus for three process. So, every process can have in one of the three states, one is the undecided state in the undecided state the process, or the node has proposed certain value from a set of feasible value is D . So, every node has propose

some value and they are in the undecided state, then in the communication state they are exchanging. Exchanging the value among themselves. So, by exchanging the value and then by applying the come consensus algorithms, they can reach to decided state, where they set the decision that everyone in the network, they agrees under variable say D_i certain variable D_i .

(Refer Slide Time: 17:27)



The slide is titled "Requirements of a Consensus Algorithm" and lists three main requirements:

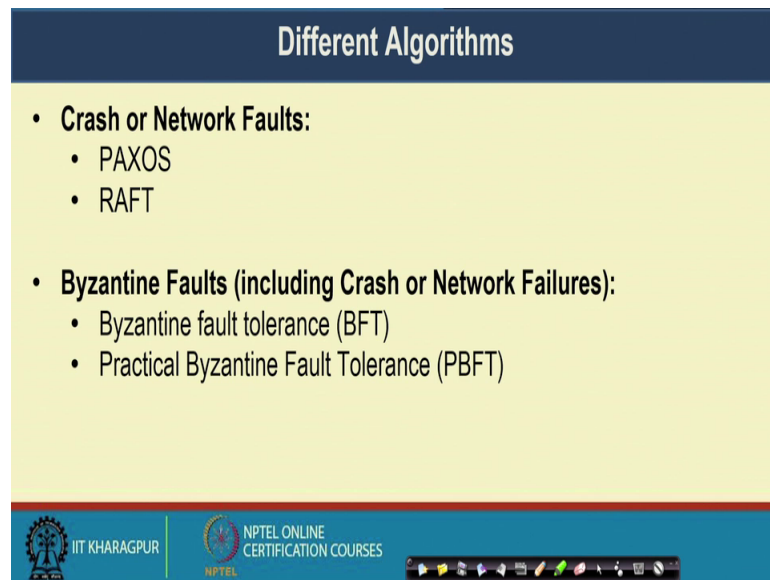
- **Termination:**
 - Eventually each correct process sets its decision variable
- **Agreement:**
 - The decision value of all correct processes is the same
- **Integrity:**
 - If the correct processes all proposed the same value, then any correct process in the decided state has chosen that value

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a speaker in the bottom right corner.

So, there are certain requirements of a consensus algorithm the details, you have looked earlier. So, you need to ensure that eventually each process sets the decision variable. So, it terminates after setting the decision variable, then the agreement property which says that the decision value for all correct process that should be same, everyone should reach to the common agreement. The third one is integrity that if the correct processes all propose the same value, then any correct process in the decided state they has to send that value.

So, if the every correct process propose one value x . So, at the decided state they should decide on the same value x .

(Refer Slide Time: 18:14)



The slide is titled "Different Algorithms" and is divided into two main categories of faults:

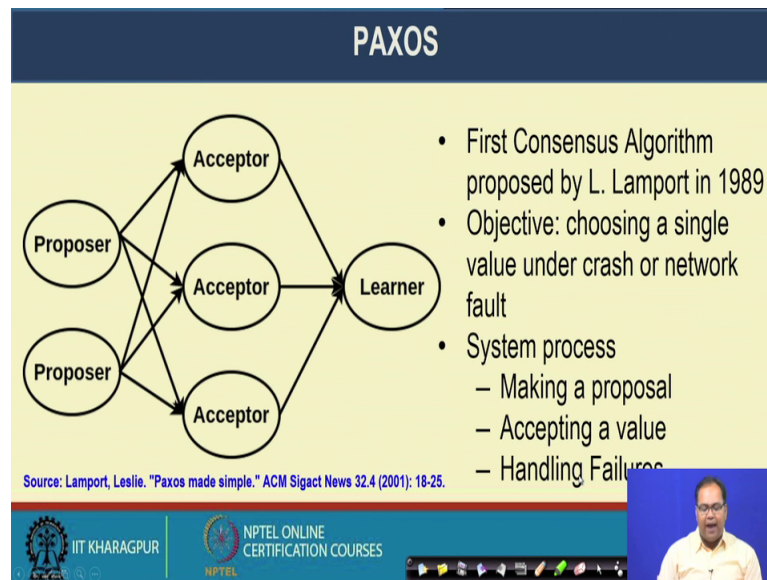
- **Crash or Network Faults:**
 - PAXOS
 - RAFT
- **Byzantine Faults (including Crash or Network Failures):**
 - Byzantine fault tolerance (BFT)
 - Practical Byzantine Fault Tolerance (PBFT)

The slide footer includes the IIT Kharagpur logo, the text "IIT KHARAGPUR", the NPTEL logo, and the text "NPTEL ONLINE CERTIFICATION COURSES".

Then we have different kind of algorithms for ensuring consensus in a typical distributed system, which we apply in permission blockchain and, these algorithms are based on this state missions application principal. So, the initial algorithms for distributed consensus or PAXOS and RAFT, which supports crash or network faults, but they cannot support byzantine fault. So, to support byzantine fault we have algorithms like byzantine fault tolerant algorithm, or practical byzantine fault tolerance PBFT algorithms.

So, this byzantine fault tolerance algorithm, they support byzantine faults in addition with the crash or network fault. So, initially we look into the PAXOS RAFT in details and then we will go to the byzantine fault tolerant algorithm.

(Refer Slide Time: 19:00)



The slide titled "PAXOS" features a diagram on the left showing two Proposer nodes on the left, three Acceptor nodes in the middle, and one Learner node on the right. Arrows indicate communication from Proposers to Acceptors and from Acceptors to the Learner. To the right of the diagram is a bulleted list: "First Consensus Algorithm proposed by L. Lamport in 1989", "Objective: choosing a single value under crash or network fault", and "System process" with sub-points: "Making a proposal", "Accepting a value", and "Handling Failures". At the bottom left, a source is cited: "Source: Lamport, Leslie. 'Paxos made simple.' ACM Sigact News 32.4 (2001): 18-25." The bottom of the slide includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a speaker.

So, let us go to the PAXOS. So, PAXOS has an interesting history, it was first dot com it was the first consensus algorithm that was proposed by Leslie Lamport in 1989 and, the objective of PAXOS over to choose a single value under crash or network fault. So, this PAXOS although it was proposed by Lamport in 1918 89, but it took around the 13 years to get the paper published and, there was a lot of discussion among the community about the correctness of this particular algorithm.

So, that if you are so, and not very confident that the algorithm proposed by Lamport is fair enough to ensure consensus in a distributed environment. And Lamport try to come up with a formal proof of this particular consensus algorithm and, because of that complicated proof people say that will it is very hard to understand the PAXOS algorithm, or it is very hard to implement the PAXOS algorithm, but although the inherent concept of PAXOS is kind of very simple.

So, we look into the PAXOS in a simplified view and, then will see that how PAXOS can be implemented in a real system to ensure consensus. So, the idea behind PAXOS is pretty simple it is like that, I am just giving you one example so, in IIT Kharagpur we have broadly two options, were the student can go say after the class say, we have a subway and we have a CCD. Now, after the class the students can collectively decide that all of them want to go for either for the subway or for the CCD. Now, the question comes that how will they select that whether, they want to go to CCD and go to subway.

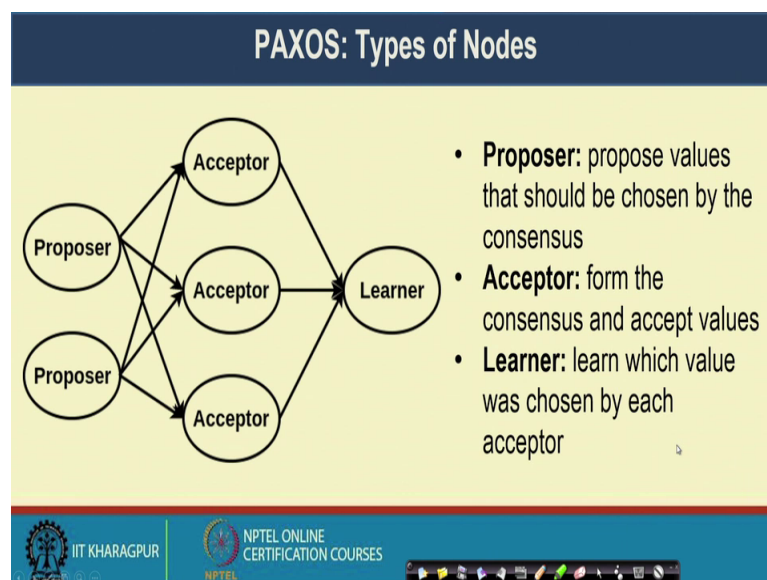
And the interesting is that all of them want to go for together otherwise, the party would not be that much good.

So, that is why the students want to make some decisions collectively that, they want to go to either subway or CCD, but there is no central leader the only way to communicate with that, they will just everyone will propose certain values and from that value proposal, they will try to come to a consensus.

So, the idea is something like that say assume that I am a member of that particular team of students. Now, I just wait for some amount of time and, I see that whether someone else's proposing a value to go for a CCD, or to go for a subway, if none of my friends are proposing any value, then I initiate a proposal say I initiate a proposal say that let us go for subway. So, now, I will see whether my friends are accepting that value or not.

So, that way the information get is propagate in the entire network and, they try to have to some certain consensus based on a majority decision that well. If the majority of the nodes proposes that, or either proposes, or agrees that us we want to go to subway, then all of them goes to for subway otherwise, if majority of the students proposed to go for CCD, then all of them go to CCD. So, that is the broad idea behind PAXOS.

(Refer Slide Time: 22:25)



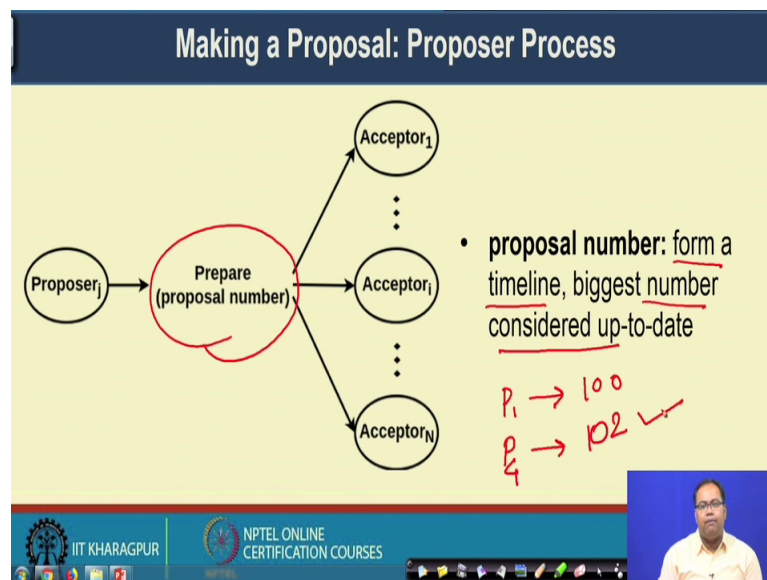
So, let us look into that algorithmically how it works. So, you have certain proposers who are proposing the values that proposed values should be chosen by the consensus

algorithm. So, here the proposers can propose a value that either to go for a subway or to go for a CCD, then you have certain acceptors they form the consensus and accept the values. So, the acceptors whenever they are hearing certain proposal from the proposer, they either accepted or rejected.

Say it may happen that if I am willing to go for a subway, but some proposal is proposal is coming to meet to go to CCD I will simply rejected. And some other friend of mine if he or she is proposing to go for subway I will agree to that. So, that way will try to look into that whether the majority of the node is either proposing or accepting, these going for subway, or CCD to which one and accordingly.

This learner module will find out that which value has been chosen by each of these acceptor and, then they will collectively accept that particular value. So, that is the broad idea behind the PAXOS. So, there are this three type of nodes the proposer the acceptors and the learner, everyone is a learner in the network which learns the, what is the majority decision.

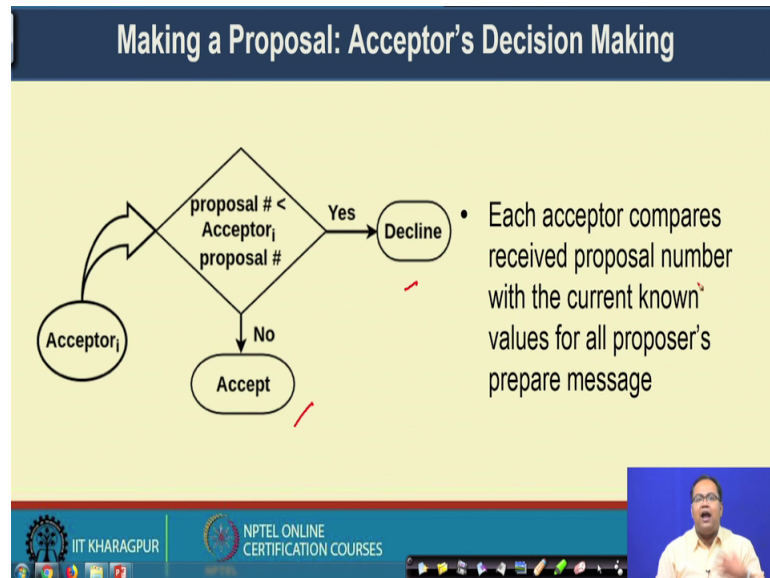
(Refer Slide Time: 23:50)



So, let us look into the process in a little detail. So, the proposer initially it prepare with a proposal number. So, this proposal number need to be good enough. So, that your proposal gets accepted.

So, the proposal propose make a prepared that proposal number and send it to the acceptor. So, this proposal number it forms a timeline and the biggest number it is considered up to date. So, it is like that one proposal is coming from P 1 with proposal number say 100, another proposal is coming from say P 4, with proposal number, 102 then we will accept the proposal which is coming from P 4.

(Refer Slide Time: 23:40)



Then the acceptor makes it in this way the may algorithm that, I have mentioned that if the proposal number is less than the acceptor is proposal number. So, if you if you receive the lower proposal number if you are if you are if your current proposal number is lower than the proposal number that you have, then you accept it otherwise you decline it.

So, each acceptor is compare the received proposal number, with the current known values for all the proposals prepare message. And if you are getting higher number, then you accept it otherwise you decline it.

(Refer Slide Time: 25:28)

Making a Proposal: Acceptor's Message

```
graph LR; A((Acceptor_i)) -- "Prepare response (accept/decline, proposal #, accepted values)" --> P((Proposer_j))
```

- **accept/decline:** whether prepare accepted or not
- **proposal number:** biggest number the acceptor has seen
- **accepted values:** already accepted values from other proposer

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, then the acceptor based on that it prepared the response, in the response the acceptor can either accept, decline a message based on the proposal algorithm. So, as I have mentioned that the biggest number are the acceptor that has it has seen till now, it put that number in the response message and, it include the acceptance values that has been accepted from the proposal. So, this accepted values is inform to the proposer.

(Refer Slide Time: 26:00)

Accepting a Value: Proposer's Decision Making

```
graph TD; P((Proposer_j)) --> D1{majority of acceptors reject proposal?}; D1 -- Yes --> U([Update with latest proposal #]); D1 -- No --> D2{majority of acceptors having accepted values?}; D2 -- Yes --> C([Proposer value cannot be chosen]); D2 -- No --> S([Send accept message]);
```

- Proposer receive a response from **majority** of acceptors before proceeding

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

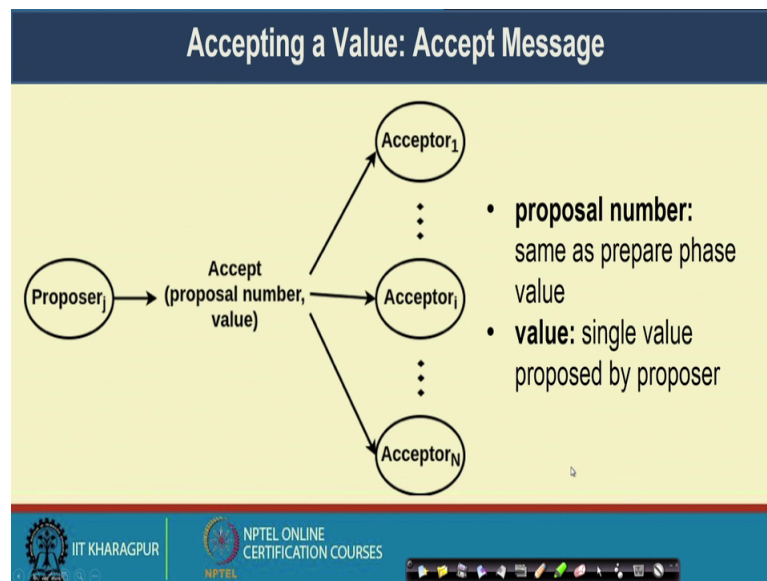
Now, we take a port based on a majority decision. So, will see that will the proposal looks that whether majority of the acceptors, they reject the proposal, if they have

rejected the proposal, then you updated with the latest proposal number, if they have not rejected it, then you see that whether the majority of the acceptors are accepted the value, if the majority of the acceptor has accepted value. Then you cannot choose that particular proposal, if it is not then you send the accept message.

So, the idea is that whenever the majority of the acceptors, there they are sending some accepted values, if they have accepted your value, then; that means, the value that you have shared that is coming to be a consensus.

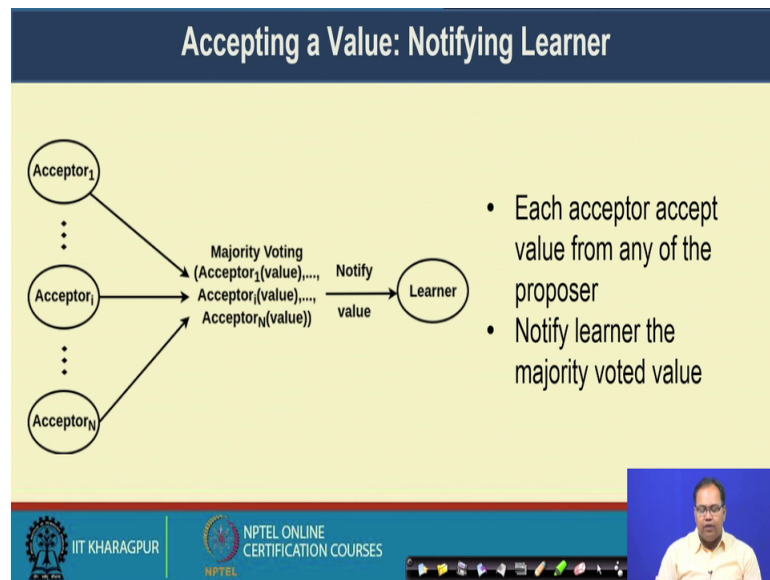
So, it is just like that you have proposed for a pees is your proposed for subway and, once you have proposed for subway and majority of your friends are accepted for the subway and from that what you can all of you can go for subway.

(Refer Slide Time: 27:23)



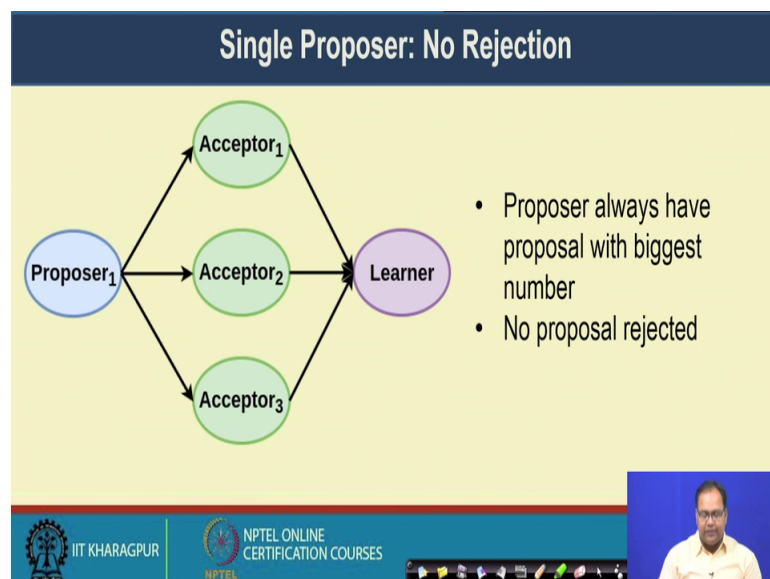
So, the final stage is this accept message. So, the proposer sends the accept message to all the acceptors in the accept message, you include the proposal number. A similar to the prepare phase that the proposal knows that my proposal has been accepted and, the value is the single value that is proposed by the proposal.

(Refer Slide Time: 27:52)



Now, whenever the acceptor it accept values from the proposer, it informs the learner about this majority voted value. So, everyone learns that what is the majority voting in the environment.

(Refer Slide Time: 28:14)



Well if you have a single proposal in the system, then the system is very simple, a with the single proposal every acceptor will accept the proposal because, that is going to be the biggest and, so the proposal does not get it rejected if the acceptors are correct.

(Refer Slide Time: 28:36)

Handling Failure: Acceptor Failure

- **Acceptor fails during prepare**
 - No issues, other acceptor can hear the proposal and vote

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, you have to ensure that if you need to ensure that majority of the acceptors are non-faulty. Now, let us see that whenever certain acceptor fails. So, this acceptor can fail during the prepare phase, if the acceptor fails during the prepare phase, then you do not have any issue that here are other acceptors who can hear the proposal and, they can vote either for the proposal or against the proposal.

(Refer Slide Time: 29:00)

Handling Failure: Acceptor Failure

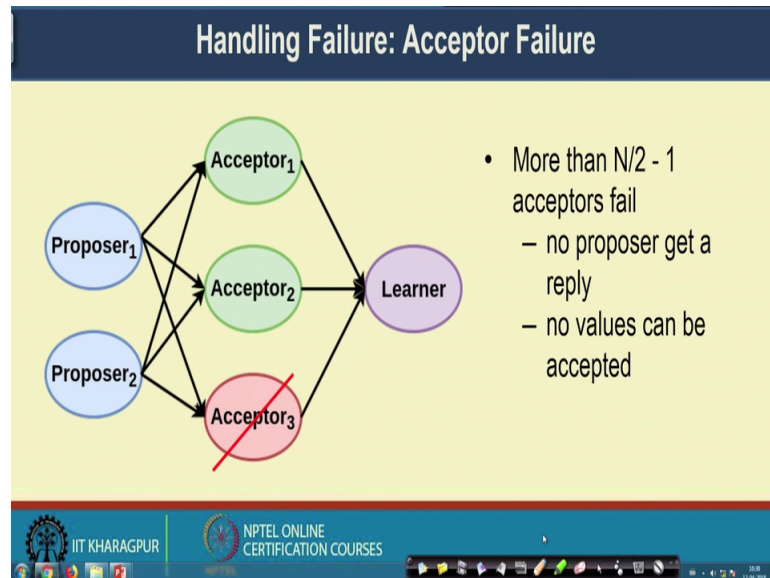
- **Acceptor fails during accept**
 - Again, no issues, other acceptor can vote for the proposal

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, the acceptor may fail during the accept stage. Again if the acceptor fails during the accept stage you do not have any issue, because other acceptors are their who have

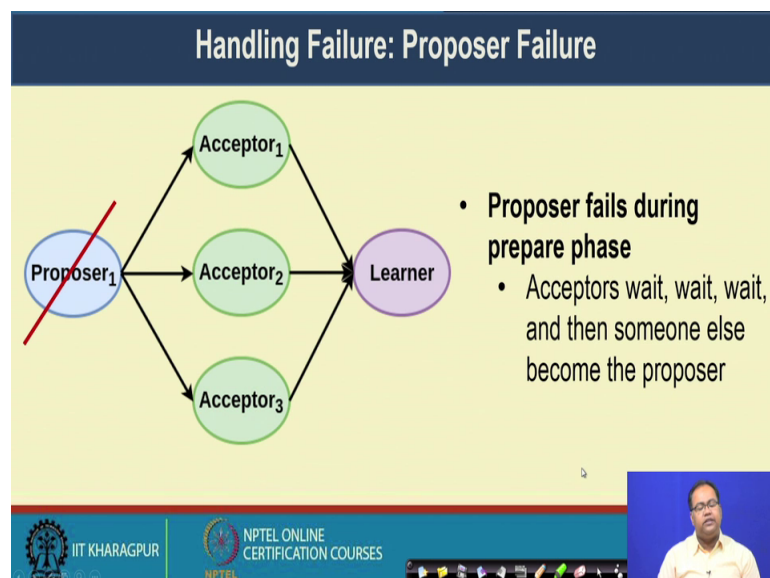
already voted for the proposal. Now, the only thing that you have to ensure that list number of N by 2 number of a acceptor, they can fail simultaneously because you are going for the majority voting principle. So, whenever you are going for the majority voting principle in a synchronous environment, you need to ensure that will majority of the nodes are correct.

(Refer Slide Time: 29:43)



So, if more than N by 2 number of acceptor fail, then no proposer they get a reply and, you cannot come to a consensus algorithm.

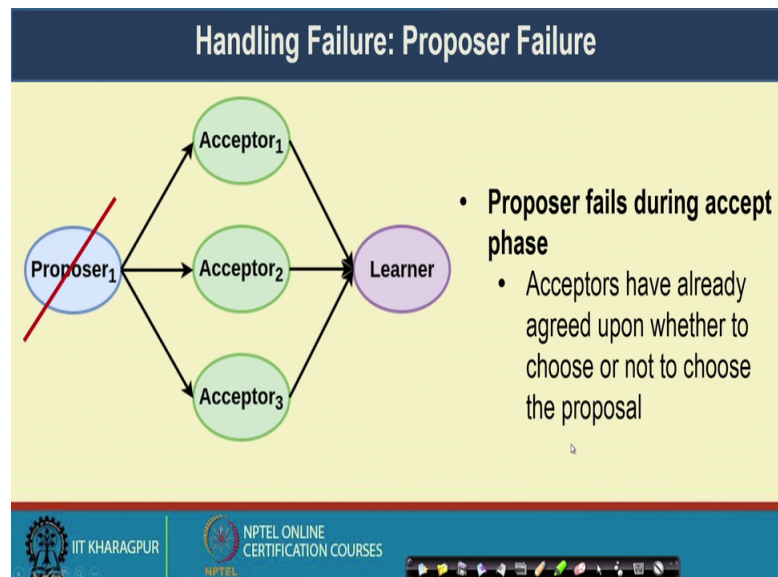
(Refer Slide Time: 30:00)



Now let us look into the scenario, when a proposer fails. Now, the proposal can again fail during the prepare phase or during the say accept phase. So, if the proposal fails during the prepare phase it is just like that no one is proposing for any value, none of your friends are proposing for idea to go for a subway or to go for a CCD.

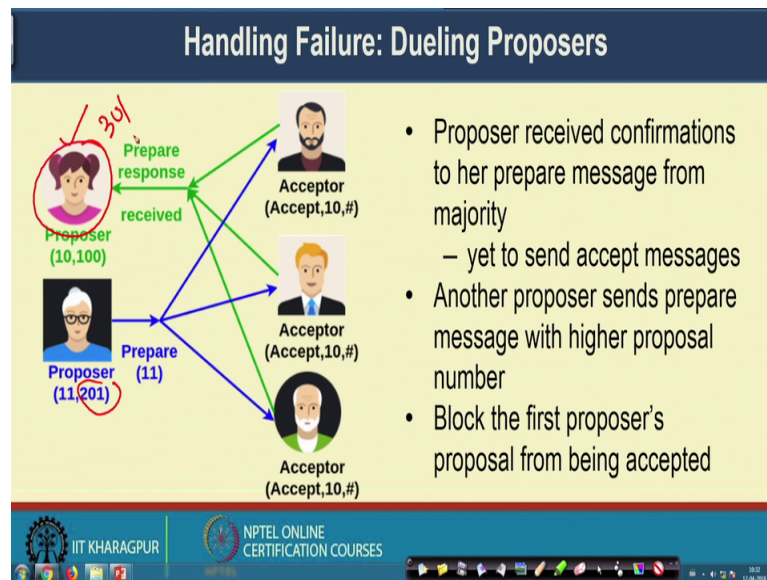
So, you wait wait wait for getting a value from your friend and, if you are not getting any value. So, you become a proposal. So, the same idea is like that the acceptor wait for certain times and, if they do not here for any proposal one of the acceptor, it becomes the proposal and propose a new value.

(Refer Slide Time: 30:43)



Now, the proposal may fail during the accept phase, but if the proposal fails during the accept phase the acceptor, they have already agreed upon whether choose or not to choose the proposal based on the majority port. So, they have shared the majority port among themselves and from, they can find out that whether the proposal has been accepted or not.

(Refer Slide Time: 30:07)

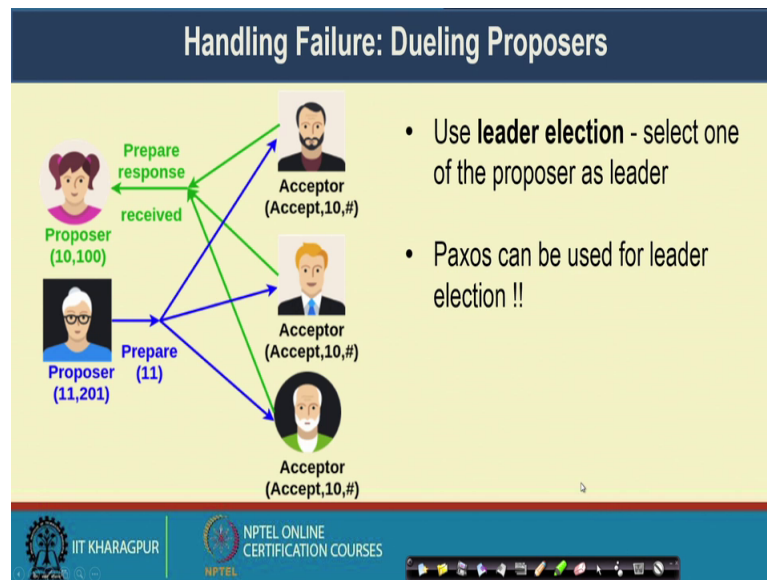


Now, there can be an interesting attack here, which we call as the dwelling proposal say do there are two proposals here, it is just like that the proposal one, it has send a it has send certain proposal two all the acceptor and, then proposer two it sends the another proposal with the higher proposal number.

Now, as soon that proposal one she is an attacker and, what she does that the moment the he says that there is a proposal with a higher proposal number; she sends another proposal based on more higher proposal numbers. So, it is like that whenever, she is here that will proposer to have send a proposal with proposal number 2 0 1 she sends, another proposal with 3 0 1.

So, that way by applying the by having there can be a kind of dwelling among the individual proposal and, you may not be reached to a consensus. Now, to break that I hear what we do that, we use the certain other certain other numbers are certain other identities which will help me to break that well. So, it is just like that whenever you are getting dwelling proposal, you block the proposer who are having lower height. So, you you may break the eyes based on the idea of the proposal.

(Refer Slide Time: 32:20)



So, to do this that which particular proposer you are going to block, you need to execute certain algorithm. So, you can have a leader election algorithm.

So, this leader election algorithm can select one of the proposer as a leader. Now, you can see that, because PAXOS is a consensus algorithm PAXOS itself can be used as a leader election algorithm. So, the few a PAXOS that I have mention due it is a kind of very simplified view, the overall idea is that the proposer is proposing certain views to all the acceptors and, acceptors they collectively look into that proposal.

And, if they agree with the proposal they send the accept message and by receiving they that particular accept message, if the proposer finds out that he is the message has been accepted then he sends back the accept to all other acceptors. And from they are to the majority voting, they come to a census protocol.

Now, this simple view of the PAXOS easier to understand it is just like making one selection, but if you look into the real system in a real system, you may go for a sequence of selection sequence of choices rather than having a single choice. So, that kind of system we call it as a multi PAXOS system. So, this multi PAXOS system it is something like this you make a sequence of choices by applying repeated PAXOS protocol. Now, applying repeated PAXOS is something complicated because, you have all these messages need to be exchange among each other again and again.

And which increases the complexity of the PAXOS protocol. So, that is why all the PAXOS gives a nice theoretical idea about why the system can reach to a consensus, or how are distributed system can leads to a consensus, indeed this PAXOS protocol was the first that has proved that will, you can have certain consensus algorithm in a distributed environment.

So, by applied this kind the proposal and acceptors accepting certain values. So, you can you can do certain level of optimizations on top of this repeated PAXOS, which is basically done in case of multi PAXOS to come to our consensus. So, will not go to the multipaxos in details rather we will go for a simplified more simplified consensus algorithm, which is widely adopted as an alternate of PAXOS which we call. So, in the next class we will look into the RAFT consensus mechanism in details are so.

Thank you all for attending we will look rafted details.