

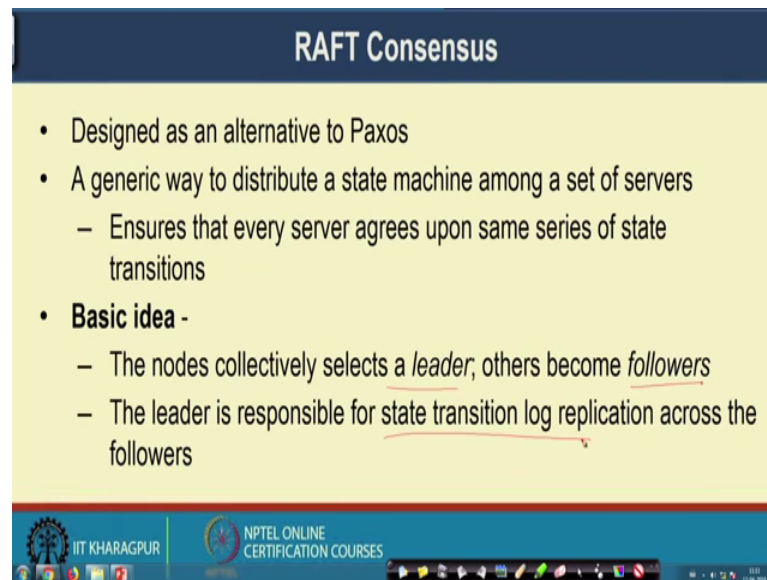
Blockchains Architecture, Design and Use Cases
Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture- 16
Permissioned Blockchain – III
(RAFT Consensus)

So, welcome back to the course on Blockchain. In the last lecture we have looked into the details of some of the consensus algorithm. So, we have looked in to a details of paxos which I have mentioned is a very complicated algorithm in theory, but very simple in concept. So, if you go for the proof theoretical proof for paxos you will look the details and different levels of optimizations which are difficult to grasp, but well the conceptually paxos is as similar as like you are proposing some value whenever you are not hearing anything and then others are accepting that value based on the majority decision.

We have not went through details of paxos, because that are not very relevant to our block chain commitment protocol, although people have started exploring paxos in more details for permission block chain environment, but there are simplified mechanisms available which nowadays people are implementing for handling a consensus in a permission model of block chain. So, today we will see another alternate of paxos which is widely used in consensus mechanism. So, that particular protocol is known as RAFT. So, we will look into the RAFT consensus.

(Refer Slide Time: 01:47)



The slide is titled "RAFT Consensus" in a dark blue header. The main content is on a light yellow background and lists the following points:

- Designed as an alternative to Paxos
- A generic way to distribute a state machine among a set of servers
 - Ensures that every server agrees upon same series of state transitions
- **Basic idea -**
 - The nodes collectively selects a *leader*, others become *followers*
 - The leader is responsible for state transition log replication across the followers

At the bottom of the slide, there is a blue footer containing the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a Windows taskbar with various application icons.

So, RAFT designed as an alternate to paxos which is a generic way to distribute a state machine among a set of servers and ensure that every server agrees upon the same series of state transition. So, the basic idea of RAFT is again simple. And interestingly the whole idea of the RAFT came from the fact that well in a distributed environment you can come to a consensus based on the paxos algorithm and whenever you can come to the consensus based on the paxos algorithm by applying the same consensus technique you can always elect a leader. And interestingly if you have a leader in the system achieving consensus becomes much easier because in that case you can then avoid the having multiple proposer proposing something all together.

So, if you just look into the way RAFT is deviating from paxos, in case of paxos you did not have any kind of leader as such. So, multiple proposer can propose the thing simultaneously and whenever multiple proposers are proposing thing simultaneously, the protocol becomes little complex that the acceptors now have to accept one of the proposals from the proposer and in that case we use that kind of highest proposal number used as a tie breaking mechanism. And we embed certain algorithm inside paxos to ensure that every proposal that is coming from different individual users or individual proposers they are in date unique.

So, it is not like that 2 proposer are proposing with the same proposal value that we ensure inside paxos. So, all this internal details makes the paxos more complicated. And

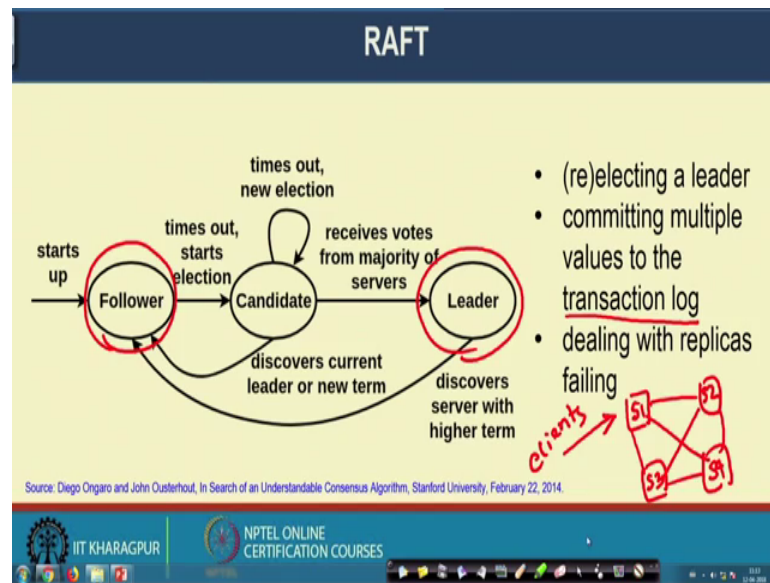
as I have mentioned earlier Lamport has to face a lot of difficulty to convince people that well paxos works in literature. So, once this paxos algorithm is well accepted, people now knew that well in a distributed environment under a synchronous assumption it is possible to design a consensus algorithm.

So, once you are convinced enough that you can design a consensus algorithm based on paxos or based on any other principle through majority voting, then you can make a simplified version of the protocol where you can think of that well, first I will elect a leader and once I have elected a leader then the task of the leader is to propose something. So, there would be a single proposer now which is the leader and all the acceptors are now the followers of the leader. So, what the leader is proposing the acceptors now we term it as the followers they can either follow the leader or they may not follow the leader.

So, it is like that whenever you want to go for a Subway or for a CCD and you want to choose something first you choose a leader among your team, and then let the leader propose whether to go for Subway or go for CCD, and then take a vote. Now if majority says that say the leader says that well I want to go for Subway how many of you are how many of you are in support of me, and now if majority just raises their hands that well we are we also agree with you. Then everyone goes to Subway otherwise if all of them declines to go to Subway you have the alternate option for CCD you go to CCD.

So, that is the basic idea behind a RAFT consensus protocol. So, the idea is that the nodes they collectively selects a leader. So, once there is a leader in the system the others becomes followers. And once you have the follower in the system now the leader is responsible for state transition log replication across the followers in terms of this replicated state machine concept.

(Refer Slide Time: 06:10)

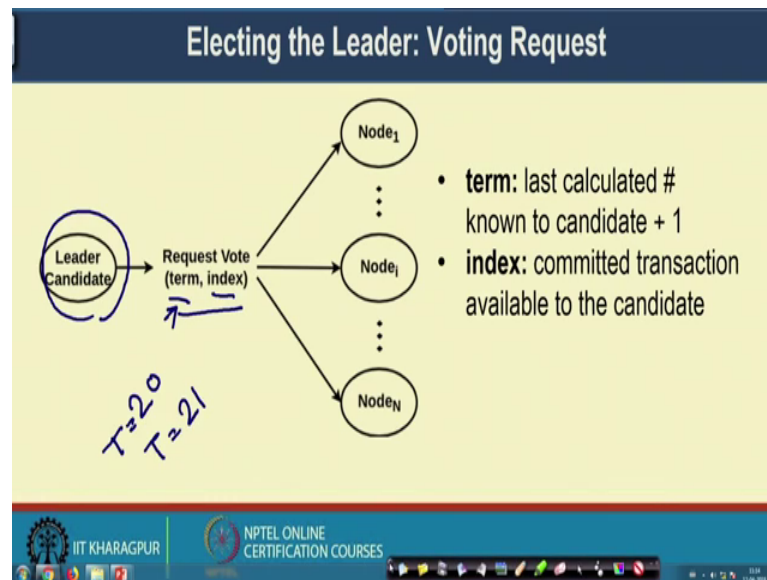


So, that is the RAFT algorithm whenever the system starts up it has a set of follower nodes. Now this follower nodes some of them just looks for whether there is already a leader or not. If it is times out; that means, there is no leader in the system you start the election. So, in the election you choose some of the candidates. So, some of your friends says that well hi I want to be a leader. Now whenever they are saying that hi I want to be a leader then you receive the votes from the majority of the server saying that who is going to be the leader.

Now, among the candidates who wins for the majority of the votes that person becomes the leader and now you have chosen one leader. So, this particular leader this leader finds out that what should be the proposed value. And then the followers can either vote for the proposal which is coming from the leader or they can go against the leader. Now here the RAFT consensus algorithm we explain traditionally in the form of transaction log. So, it is the concept from this replicated database. It is like that you have multiple replicated servers and you want to build up a consensus among these multiple replicated servers.

So, whenever some transactions are coming up from the clients then you want that this replicated server say S 1 S 2 S 3 and S 4. They collectively take some decision about this consensus and based on that they decide whether to commit that transactions or not.

(Refer Slide Time: 08:09)

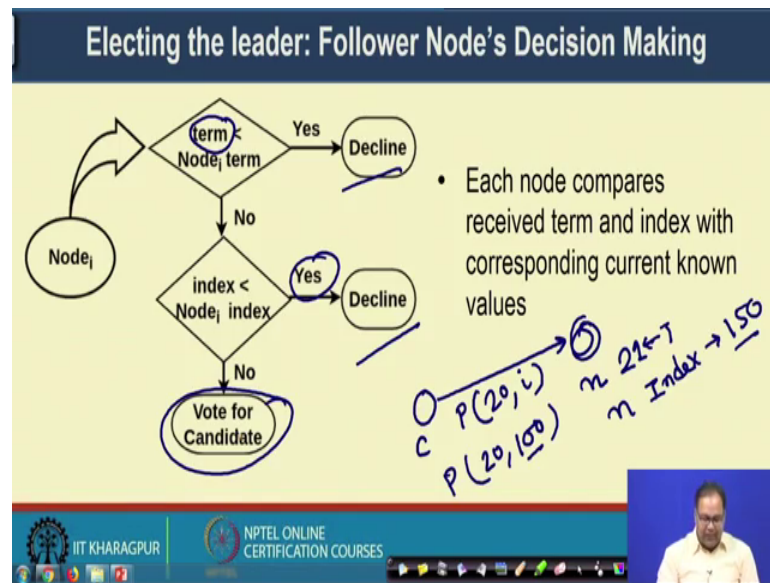


So, from here let us, let us move to the algorithm in little details. So, initially the first part of RAFT is to elect a leader. Now the question comes that how will you elect the leader. So, to elect a leader you need to be certain leader candidates. So, it is just like among the servers someone can just wait for certain amount of time to see whether some leader is already there. And if once it is times out you just think of whether you want to be leader or not if you want to be a leader you just announce say I am the candidate.

So, once these leader candidates are there then this leader candidates request for the votes. So, these votes contain 2 parameter: one we called as the term, the second one is called as the index. Now just like paxos this RAFT algorithm also runs in rounds. Now every round you need to take one decision and this term this particular parameter it denotes in which particular round you are. So, this term is calculated as the last calculated number known to the candidate. That what was the last candidate term plus 1. It is like that the earlier round was round 20 when your term was equal to 20.

Now, you want to have a new vote to elect a new leader. So, you make the new term as 21. So, that way the terms are decided. The second parameter is the index parameter. The index parameters says about the committed transaction which is available to the candidate. So, it is just like that the index parameters sees that well up to this many of transactions that have been committed till now. Now, this request vote message. It is passed to all the nodes who are there in the network.

(Refer Slide Time: 10:03)



Now, this nodes they receive a message. Once the nodes receive a message their task is to elect a leader. So, this is the mechanism to elect a leader in RAFT consensus. So, the idea is again pretty simple just by making a comparison on top of the term and a index value. So, node i it executes the algorithm in this way. So, the node i receives the term value from one of the candidate. And so, it looks into it is own term. So, it if first find out whether it is own terms is the proposed term is less than it is own term.

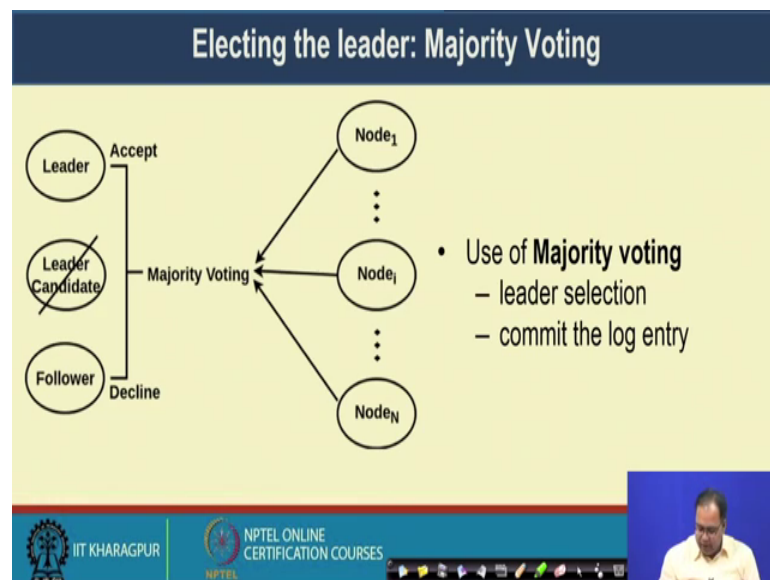
So, what it means that the node I say this is the candidate and this is a node. Now this candidate sends a proposal message with a term value. Now the candidate say proposed something with term 20 and some index parameter currently I am writing the index as i and this particular proposal message receives to the node n. Now the node n looks that what is the term that it has, and it finds out that it is already in term 21 say 21 is the term which is already there at node n.

Now, if node n finds out that 21 is the current term and some candidate is requesting to be a leader for term 20; that means, it is requesting for some earlier terms some earlier round. So, you decline this message. If it is not, like if your term is either equal to the current term it may happen in the current term because the leader may fail in the current term and it wants to choose another leader in the current term or current round itself or they may initiate a new round if that is the case then you look into the index parameter.

So, if you find out that the proposed index is again listed on than the nodes index. That a proposed index is 100 say now assume that my index value is 100 in the proposal message. If the proposed index is 100 and a node has a index value of index value of say say 150 it indicates that this node has already committed up to transaction 150 and some proposer some, I will not that are used the term proposers some leader candidate it is proposing to be a leader for some transaction some old transaction 100.

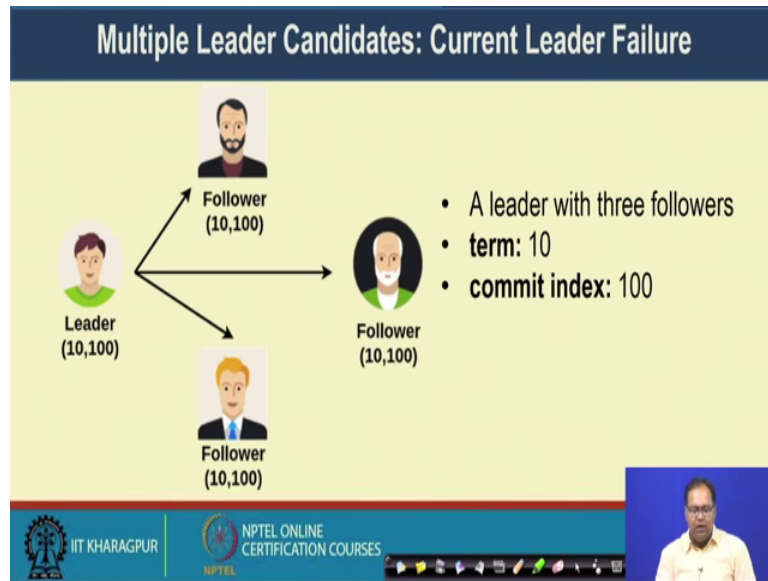
So, we should not allow that because that particular transaction has already been committed. So, if this is the case then decline again decline that particular nodes request to be a leader. Otherwise you make a vote for the candidate.

(Refer Slide Time: 12:56)



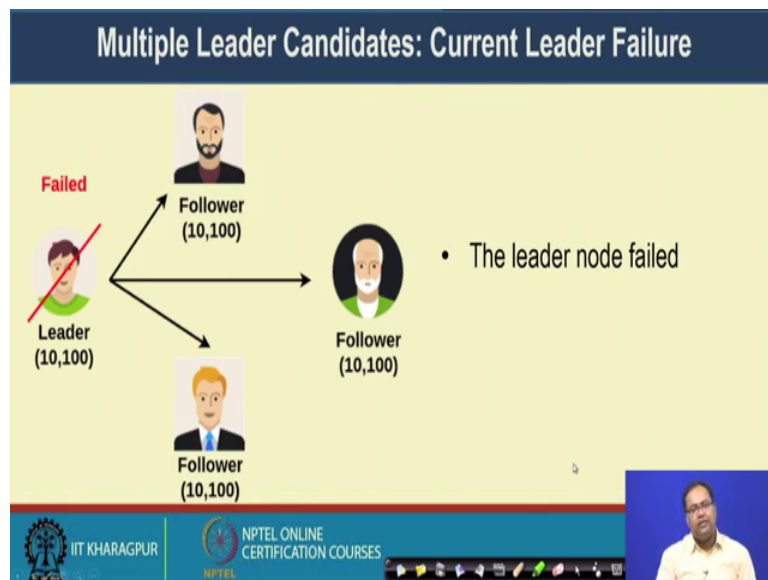
So, this is the way you select a leader in case of the RAFT consensus algorithm. Now the leader is getting a vote. So, what we do that every node sends their vote and we use the concept of majority voting for leader election. And commit the corresponding log entry. So, it is like that if certain leader candidate it receives majority of the vote from the nodes, then that particular candidate becomes the leader and other becomes the follower of that node.

(Refer Slide Time: 13:31)



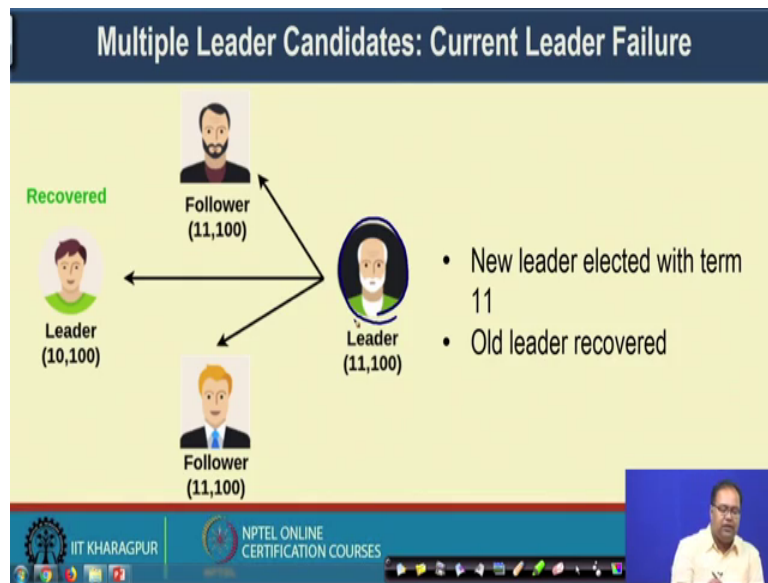
So, let us look into an example when there are multiple leader candidates at the failure of the current leader. So, we are taking a typical example where, a leader has 3 followers the current term is 10 and a commit index is 100.

(Refer Slide Time: 13:50)



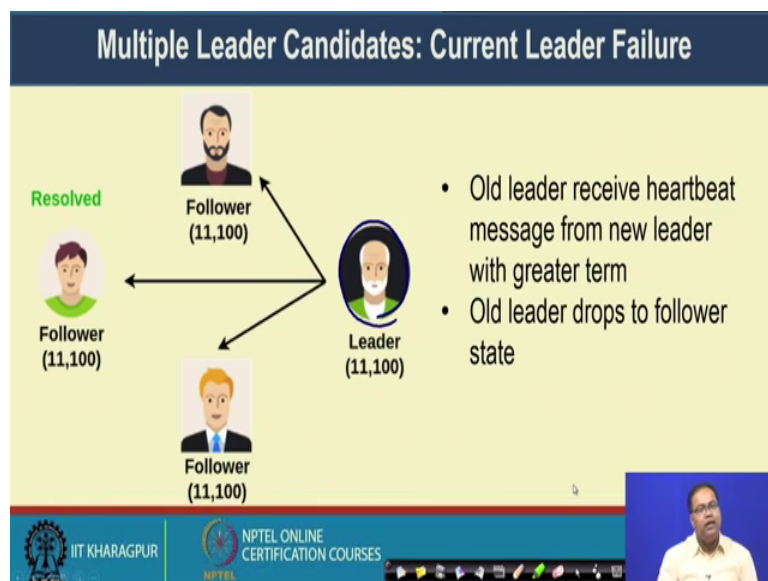
Now, say the leader node has failed. So, once the leader node has failed.

(Refer Slide Time: 13:53)



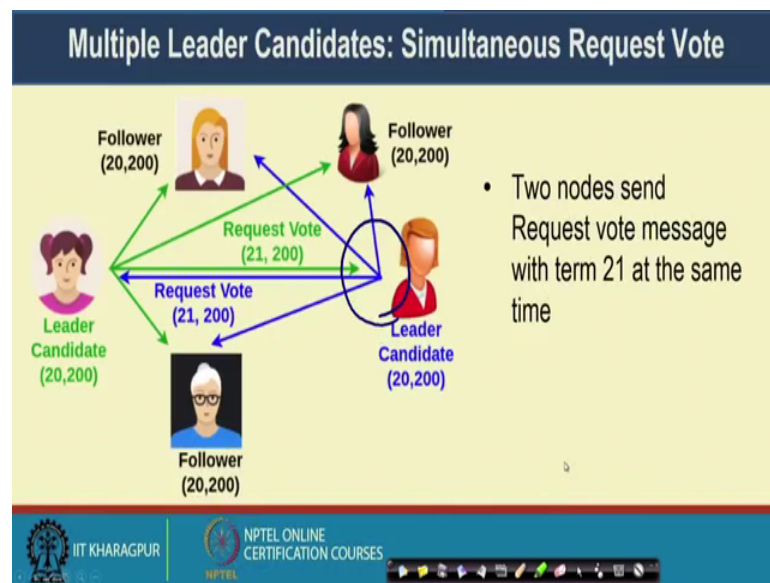
Now, you need to elect a new leader with term 11; that means, you need to move from round 10 to round 11 because your leader is going to change. At this particular time there can be like that a old leader it gets recovered. So, it is like that a new leader is elected among these 3 followers. This old person is elected as a new leader. And the new leader sends the message that well I am the leader for in term 11 the round 11 and all of us are in the commit index 100. So, we are on the same page. And this particular message reaches to the old leader.

(Refer Slide Time: 14:40)



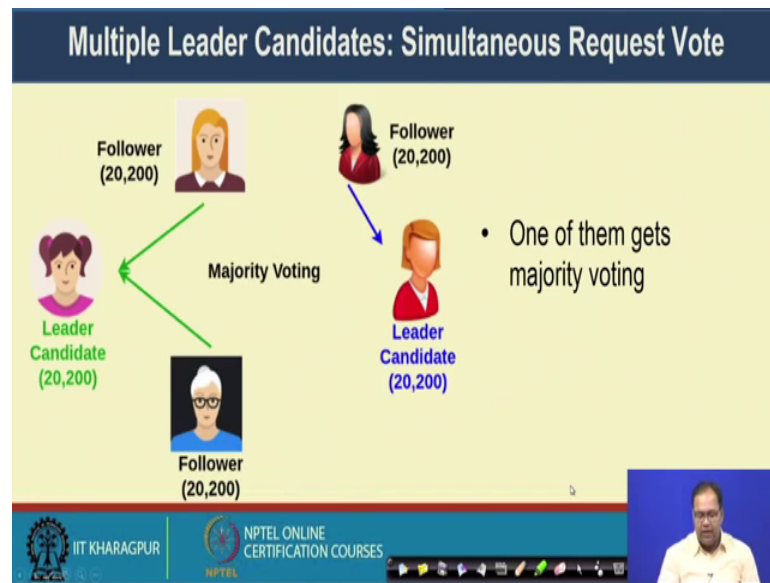
So, once the old leader receives this message the old leader finds out that well the old leader was a leader at round 10 and now there was a new leader in the system which is a term 11. So, the term 11 or the round 11 has already started in the system. So, that particular leader then falls down from a leader to a follower. So, it this poor boy it he falls down as a follower from the leader. So, that way whenever a leader fail you have one way of handling a new leader by utilizing this term parameter.

(Refer Slide Time: 15:22)



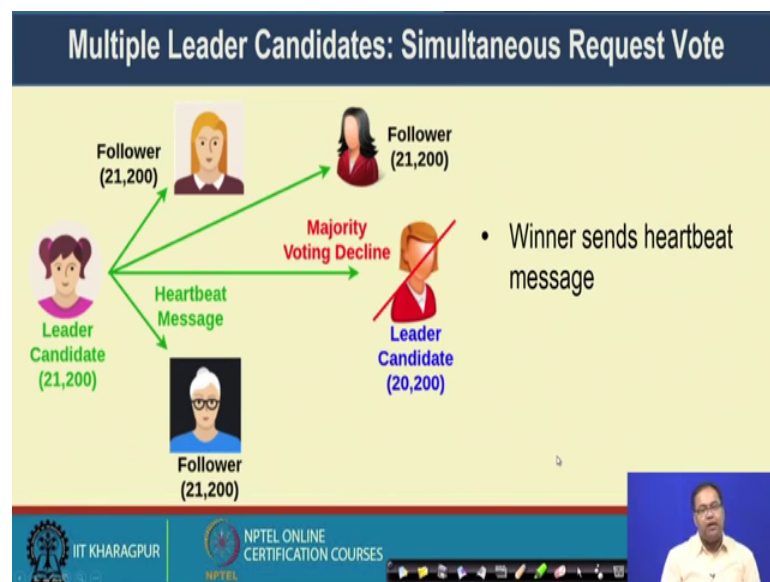
Then there can be another case when the 2 node this sends the request vote message with term 21 at the same time. So, it is like that there are 2 leader candidates. And both the leader candidates the entire system is currently say at term 20. So, at that term 20 there are 2 leader candidate this node and this node they are sending a request vote for round 21. So, if it happens in that case.

(Refer Slide Time: 15:57)



They look for the majority voting. So, say these 2 followers have voted for this girl and this follower has voted for this women candidate. So, one of them will get the majority voting.

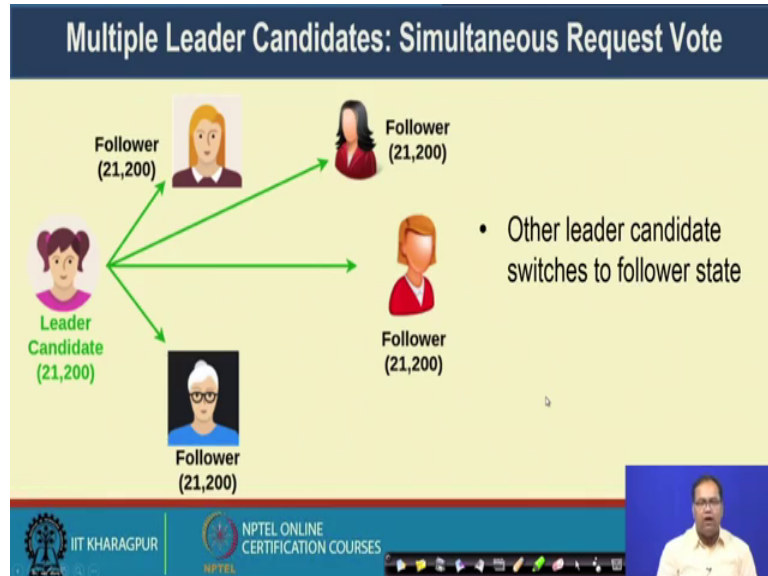
(Refer Slide Time: 16:11)



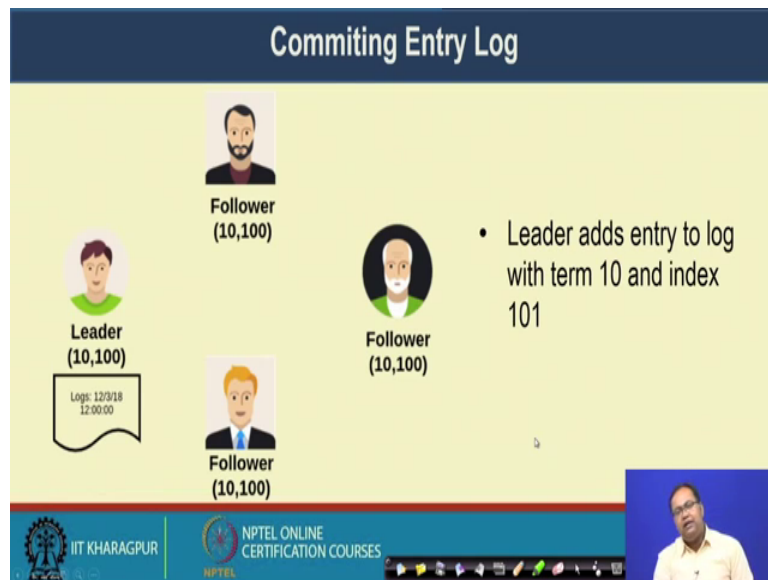
And that node which will get the majority voting it sends a special message called a heartbeat message. So, this heartbeat message through this heartbeat message it ensures, that well I have received the majority of the voting whenever it sends that well, I have received the majority of the voting then the another leader candidate it looks the

heartbeat message from the winner and that leader candidate falls back to a follower from the leader and around 21 or the term 21 gets started.

(Refer Slide Time: 16:36)



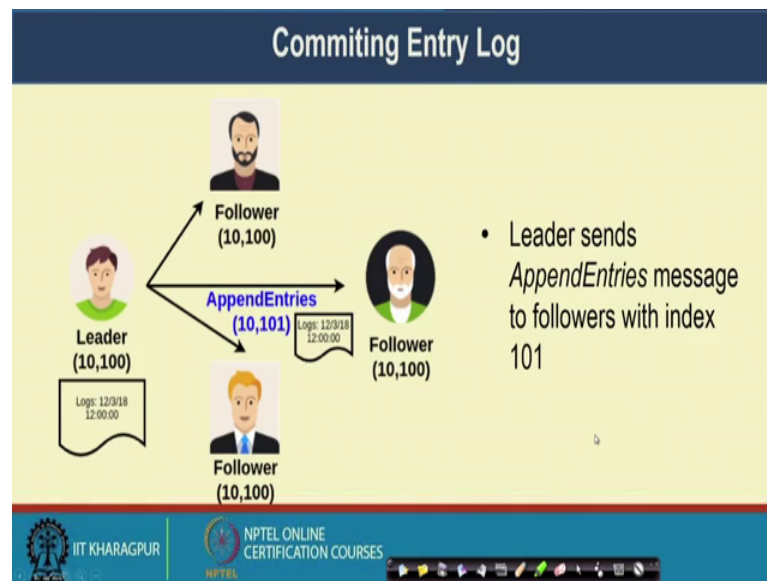
(Refer Slide Time: 16:44)



Now, the second part once the leader election is done the second part deals with that index term that how will you commit a index log, how will you commit a index or a transaction it certain index at the index log ensuring that that particular transaction has been committed as agreed upon by all the followers. So, here once the leader election is done the leader has the task to propose for a new transaction.

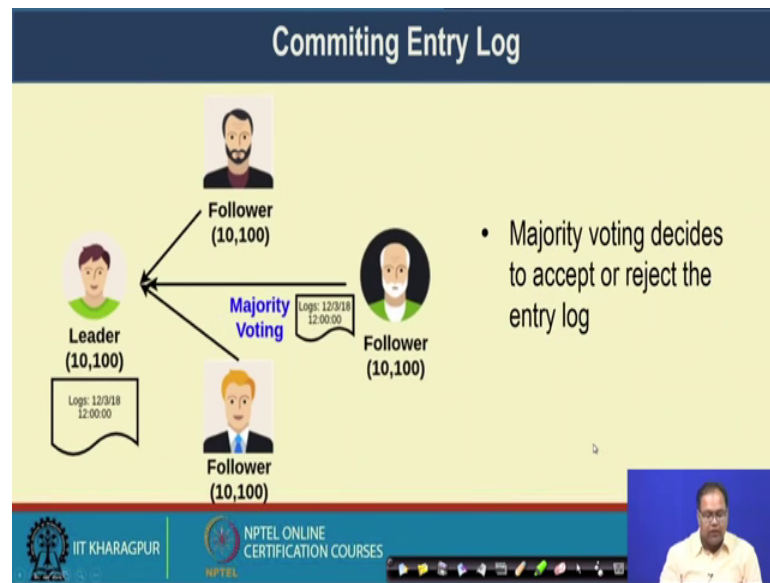
So, what the leader does that everyone is now at round ten; that means, term 10 and the last committed index the transaction index was index 100 now the leader it wants to propose a new transaction. So, it acts an entry log with term 10 the current term where everyone is and the new transaction index has index 101. So, this particular entry is done in the index log of the leader.

(Refer Slide Time: 17:56)



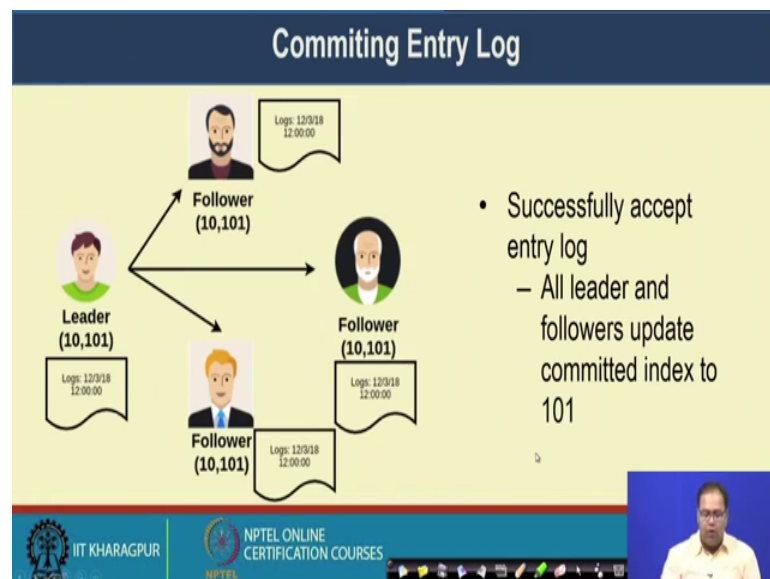
Then the leader sends a message called append entries to all the followers. So, this append entries is actually containing the proposal which is coming from the leader. So, in that particular proposal the leader says that well now we are at index 10; that means, we are at round 10. And I am proposing for the transaction 101 whether you want to commit it or not. So, once this particular transaction log is broadcasted to all the followers.

(Refer Slide Time: 18:34)



So, the followers collectively the followers collectively vote either for the transactions or against the transaction. So, if it happens that the leader receives the votes a for this transaction 100 0 1 transaction 1 0 1 and if majority says that well we are fine with committing this particular log then it decides whether to accept or reject that particular new proposed 100 log, log 101.

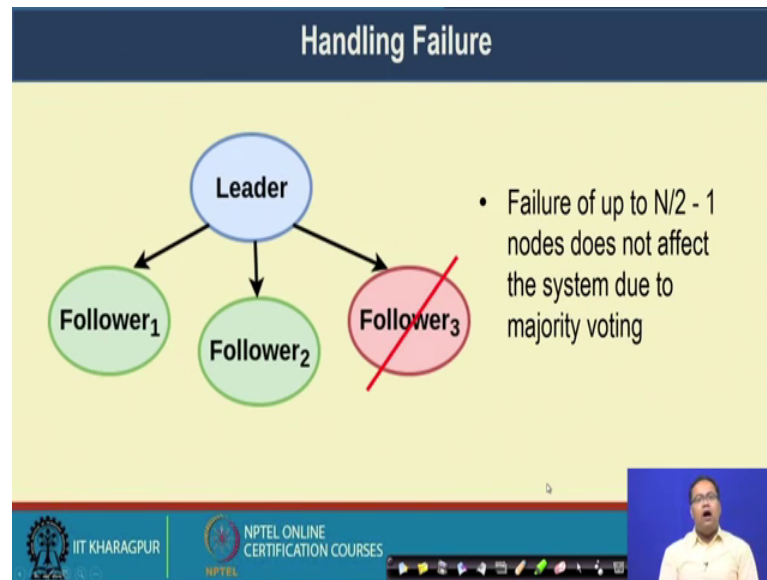
(Refer Slide Time: 19:05)



So, once this is done then this log 101 gets accepted by everyone it is put to the entry log of all the all the individual followers and then what we need to do that in the current

round that particular logs get enter and the leader sends a accept message based on the majority voting to all the individual followers. So, the followers they update the committed index to 101.

(Refer Slide Time: 19:45)



Now, the thing is that there can be kind of failures in case of RAFT consensus. And as we have mentioned earlier that RAFT and paxos they are good for handling crash fault or network fault. So, it may happen that a follower has crashed. So, the system can tolerate up to the failures of n by 2 minus, now one number of nodes. It does not affect the system because we rely on the majority voting because in that case till majority of the followers they are non-faulty and they can send a vote and the leader can take the majority decision whether to accept or reject a particular a particular transactions or not now.

So, so the entire protocol if you look into from the point of view of paxos and RAFT, and if you make a kind of comparative analysis between the 2, you will see that well the idea of paxos was difficult to prove because the idea of paxos was that the individual nodes they are proposing for certain values and acceptor needs to accept those values and because there are no leaders in the system.

You wait for certain amount of time to see whether someone is proposing a value, if none of them are proposing a value then you propose a value. So, whenever you are proposing a value during that time some of the nodes can accept that proposal and some of the

nodes cannot, if you are getting a majority voting then you know that your proposal is being accepted and then you send that accept message to all the nodes

Now because there can be multiple proposer from the system, the it becomes difficult to theoretically proved that how repeated execution of paxos which we called as a multi paxos. And if you look into the Wikipedia page you will see that they define paxos as a set of protocols rather than a single protocol. So, that set of protocols collectively decides that well the system can go for a consensus. Now when lamport has designed this paxos kind of algorithm that concept of leader was still not there because to elect a leader you again lead need something like consensus algorithm, where everyone will agree on the same leader.

But the entire system becomes stream line, once you have elected a leader. So, that way RAFT has improved the concept from paxos, that well, rather than going for repeated paxos which we call as the multi paxos we will go for a leader election mechanism. So, in that particular leader election mechanism we will apply certain consensus algorithm based on the majority voting. So, this majority voting so, there can also be multiple candidates multiple leader candidates similar to paxos proposers.

So, in RAFT you can first apply this kind of paxos type of algorithm to elect a leader, but once you have elected a leader based on the majority voting date then you can always ensure that well the next series of transactions can be committed just by that leader. So, in comparison to multi paxos and RAFT in case of multi paxos for every individual transactions you need to run the paxos algorithm repeatedly to come to a consensus, but in case of RAFT you just need to execute that complicated paxos algorithm just for once to elect a leaders.

So, once you have elected a leader the entire things becomes similar. So, that way this particular RAFT consensus algorithm, it has improved this concept of leader election. And it has improved and it has made the consensus easy to understand as well as easy to prove theoretically. So, in this particular talk because this is a kind of general topic to discuss among the audience I am not going to all this theoretical proof, but I will always encourage you to look into this paxos papers.

So, the paxos papers as I have mentioned that it has nice history like the initial paper by lamport that got rejected by the reviewers and ultimately from 1989 to 19 I think 1998

that paper got published the initial version of the paxos paper. Then lamport made another paper public in 2001 which tells about that the title of the paper is paxos made easy sorry just paxos made simple.

So, that particular paper also does not meet paxos too simple rather it tries to describe the entire thing in a more simpler way than the earlier problem that lamport termed as a parliament problem. So, from compared to that paxos algorithm, where you can have multiple proposers and you have to elect one proposer out of that this RAFT consensus algorithm it makes a simple way for round wise processing of committing the sequence of transactions.

So, you elect a leader and then once the leader is there then you then the leader will propose the new transaction. If majority of the followers accept that particular transaction, then the transaction is committed and a increase made in to the law. And that way once the leader is elected then you cannot have multiple proposers. And your life becomes simple. So, that was the RAFT and a paxos consensus algorithms, but as I have mentioned earlier that for both RAFT and paxos there they can tolerate up to n by 2 minus 1 number of crash faults.

(Refer Slide Time: 25:51)

The slide is titled "Byzantine Generals Problem" and contains the following content:

- Paxos and Raft can tolerate up to $\frac{N}{2} - 1$ number of crash faults
- What if the nodes behave maliciously?

The diagram shows a central node (a man) sending messages (represented by a sandwich and a mug) to two other nodes (a woman and another man). The woman node is positioned above the man node, and the man node is positioned below the woman node. The central node is positioned to the left of the woman node and above the man node.

The slide footer includes the IIT KHARAGPUR logo and the NPTEL ONLINE CERTIFICATION COURSES logo.

So, your life is good as long as you have only crash faults, but what if the nodes behave maliciously like your friend. Now to some of your friends you are proposing to go for a sub and for some nodes you are proposing to go for a CCD. So, if that is the case. So,

that is a kind of malicious behaviour say partly you are proposing for sub and partly you are proposing for this CCD for this kind of applications your paxos and RAFT will not be able to tolerate this kind of faults. So, this particular class of faults we call it as a byzantine fault.

(Refer Slide Time: 26:42)



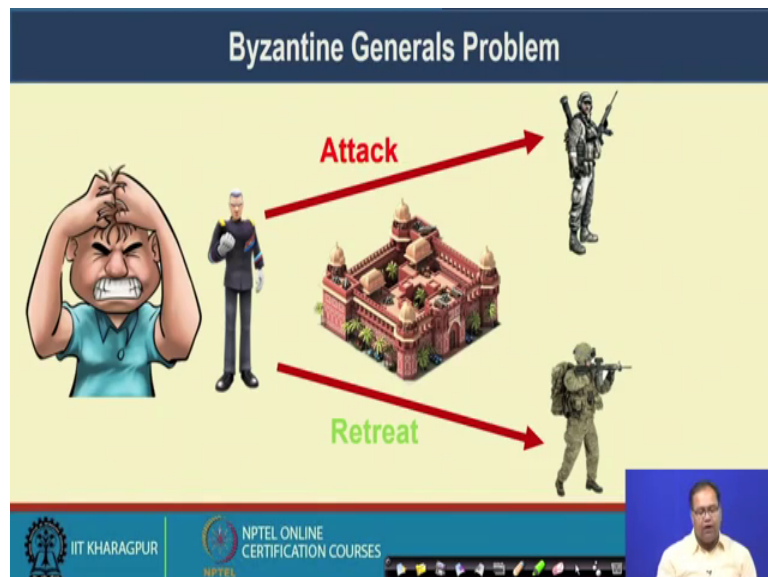
So, this byzantine fault came from a interesting problem called a byzantine general problem. So, this byzantine general problem, you have certain forts that group of army wants to attack. Now based on the scenario the general can either make a order to all the armies to make a attack from different sides of the fort or what you can do that well.

(Refer Slide Time: 27:10)



You can ask them to retreat. Now the good thing is that well we are happy if the general is good, the general is always saying the same thing to both the say lieutenants we call as the lieutenants or the soldiers both the true troupe of the soldiers that either to go for a go for an attack or go for an retreat.

(Refer Slide Time: 27:33)



But at unfortunately our general is not so good. The general can behave maliciously the general can sometime say attack to one soldier and say retreat to another soldier. So, it may happen that the general has taken some bribe from the general of this fort and

started behaving maliciously. So, this particular problem we call it as a byzantine general problem, where a particular node can behave maliciously and your RAFT and paxos algorithm which relies on that faulty nodes will not send any vote. So, from the non-faulty nodes I will always receive a correct vote that particular assumption does not hold any time. It may happen that faulty node which we call as a byzantine faulty, that faulty node it is selectively sending votes to some of the nodes.

So, it is like that some of the followers they say, some of the followers are getting a information that well I am for this particular transactions. And to some of the other followers it is saying that well I am against this transaction. So, it may happen that say just think about the RAFT algorithm. In the RAFT algorithm try to map it with this byzantine general problem and now assume that rather than the leader to have a crash fault the leader is behaving in a byzantine way. Now if the leader behaves in a byzantine way it is like that the leader is proposing transaction to a selective number of nodes selective number of followers among the all followers.

So, that way if the leader is behaving byzantine way the RAFT consensus will not be able to recover from that system. So, in that case we need to have a strict class of consensus algorithms that can tolerate the byzantine fault and that consensus algorithm we call it as the byzantine fault and consensus algorithm. So, in the next class we will look into this class of consensus algorithm, which can tolerate byzantine faults in details.

So, thank you for today.