Blockchains Architecture, Design and Use Cases

Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 17
Permissioned Blockchain – IV (Byzantine General Problem)

So, welcome to the course on Blockchains Architecture, Design and Use Cases. So, till the last lecture we have talked about the basic fault tolerant protocols in a distributed system. And we are continuing our discussion from this permission block chain environment which mainly follows the principles of general distributed system.

So, we have looked into different aspects of this fault tolerant nature of a distributed consensus protocol, we have looked into the raft consensus and paxos consensus protocol in little details, and during that discussion we have understood that well the raft and paxos. They work good for crash faults, but if there is some kind of byzantine behaviour in the network; where the node starts behaving maliciously, then the paxos and the raft protocol cannot handle this kind of scenarios.

So, on this ground we will look into a different class of fault tolerant protocol for distributed system under the permission or a closed environment, which we call as the byzantine fault tolerant protocol. So, we will look into this byzantine fault tolerant protocols in details.

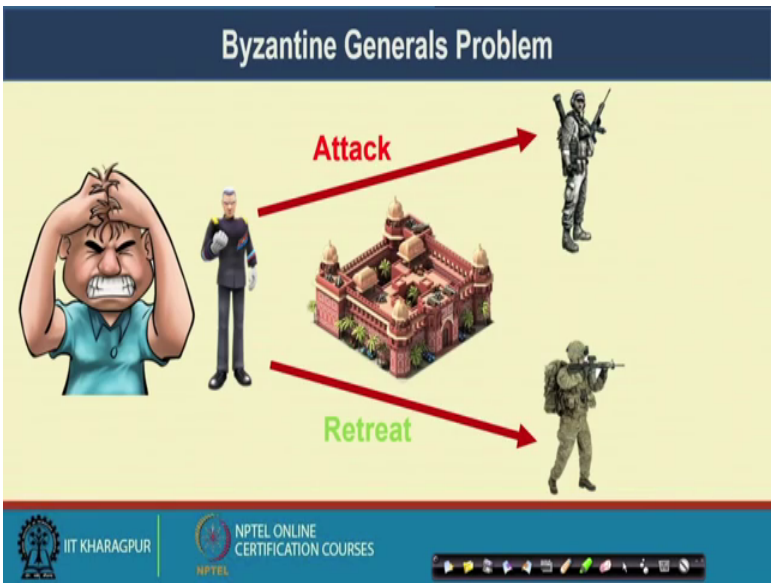
So, as you have mentioned that we are talking about this permission block chain environment which is a closed environment and every node knows each other. And under this permission model fault can occur and whenever there is a possibility of having certain kind of faults. During that time you may have broadly 3 different types of faults the crash fault where a node can fail arbitrarily due to crash that can be a hardware crash or a software crash. And in case of a crash fault it may happen that the node stop transmitting messages to all it is sphere.

Now the crash faults are kind of recoverable. So, a node can recover from the crash faults after certain duration. And after this recovery from the crash fault the node can again start behave in normally. But a second class of faults which is the byzantine fault.

So, crash fault has another notion of network fault where you can have a communication failure between 2 nodes, and if there is a kind of communication failure between 2 nodes and it gets repeated it may happen that the network may get partitioned. Now if the network get partitioned, during that time you do not have any way to send messages from one partition of node to another partition of node and you may not be recover from this correct behaviour.

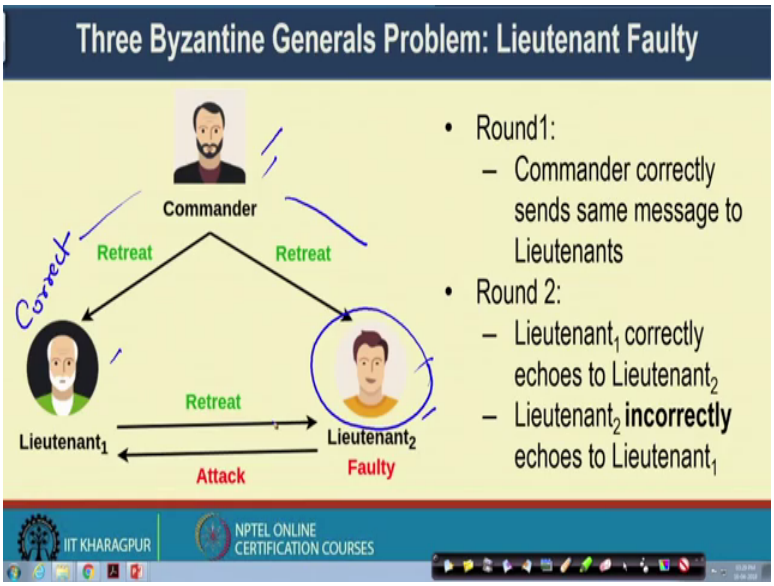
Now, if you cannot recover from the correct behaviour, during that time you have this kind of consensus protocols like the raft and paxos but if you have the fault like a byzantine fault when a network or sorry, when a node starts behaving maliciously. So, in the in the presence of this kind of byzantine fault, it may happen that, it may happen that the node sends different messages to different peers. So, here is an example of an byzantine fault. The fault came from the concept of byzantine general problem where the general sends an attack message to one troupe.

(Refer Slide Time: 03:48)



Whereas, send the retreat message to another troupe. Now if the general becomes faulty, then it becomes difficult for a system to find out what to do. So, we will look into this general class of faults in a distributed system under a closed environment, which we call as a byzantine general problem or byzantine fault tolerant problem. And we will try to develop a fault tolerant architecture where the system will be able to tolerate this kind of byzantine faults.

(Refer Slide Time: 04:19)



Now let us look in to the byzantine general problem under multiple node scenario. So, here we are considering the 3 byzantine general problem; where you have 3 generals, in terms of these are kind of Lamport timestamp to denote this kind of byzantine general problem.

And in this architecture you have one commander and 2 different lieutenants. So, the commander sends the message to the lieutenants. And the lieutenant can share the messages among themselves and try to find out whether the commander is faulty or whether the lieutenant is faulty.

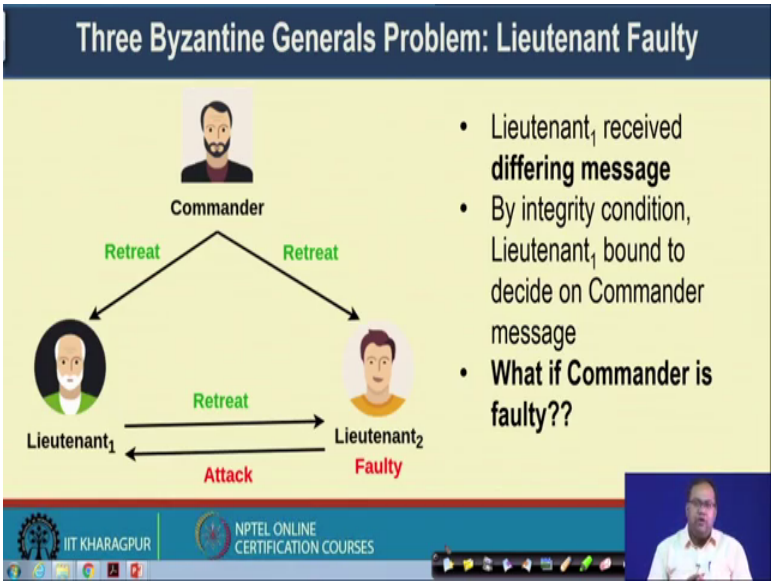
Now, let us look into the case from the perspective of 3 generals. And in this particular case we will try to design a problem, we will try to design a solution for this byzantine fault tolerant system. So, the in this architecture we assume that the lieutenant is faulty. Now if a lieutenant is faulty, then the lieutenant may send different messages from what it here. So, the commander is a correct commander.

So, the commander sends a retreat message to both the lieutenants. Now this particular lieutenant this is a faulty lieutenant. So, the faulty lieutenant does not obey the message which is send by the commander, rather the faulty lieutenants sends an attack message to the other lieutenant. Now this lieutenant the second lieutenant, this is a correct lieutenant. So, the correct lieutenant sends the retreat message, what he has heard from the commander the same message is sent to the second lieutenant. Now under this particular

scenario, when the commander is correct, but one of the lieutenant is faulty let us see that whether we can reach into a consensus under this scenario.

Now in this scenario lieutenant 1, it receives different messages. So, it receives a retreat message from. So, it receives a retreat message from the commander, and it receives an attack message from the lieutenant 2.

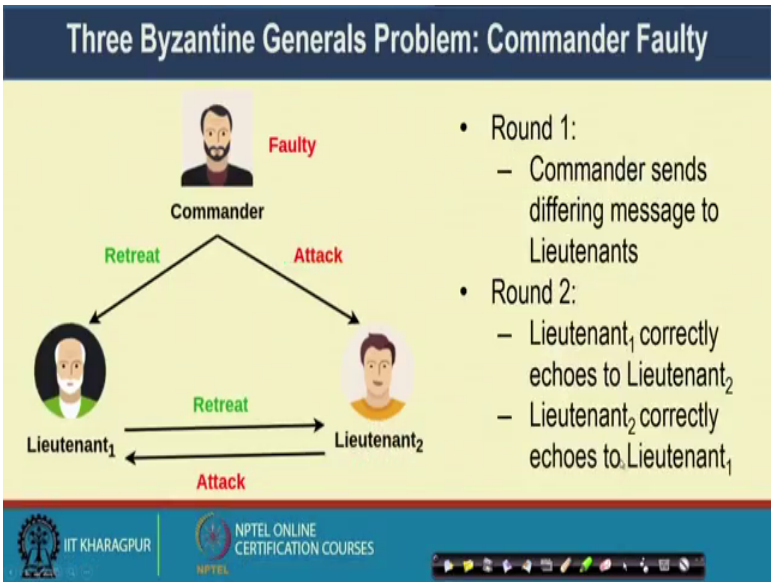
(Refer Slide Time: 06:13)



Now, in this case in general principle of a normal army scenario, the lieutenant always obey the commander. Now with this integrity condition of the army the lieutenant may obey the commander. And if the commander is non faulty, then the entire system what is correctly even if the lieutenant 2 is faulty and the lieutenant 2 sends a wrong message like this attack message to lieutenant 1.

So, we see that well if the lieutenant is faulty, but the commander is correct, then by this general principle of integrate in the army, we will be able to solve this problem that, all the lieutenant will follow the commander. Because the commander is correct, it is then we do not have any kind of consensus problem in the system.

(Refer Slide Time: 07:20)

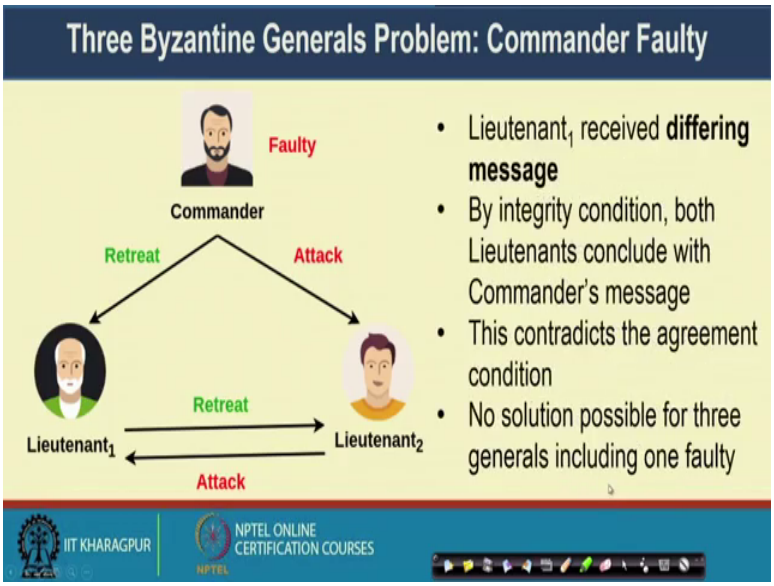


But now let us look into the second condition, that the commander is faulty. So, this is a kind of a non-trivial case. This is a kind of difficult case which to we need to handle in case of a byzantine general problem with 3 generals.

So, in this case, the commander it sends the retreat message to one lieutenant. And it sends a different message the attack to the second lieutenant. Now in this particular case, we assume that the commander is faulty. So, the commander is faulty, but both the lieutenants are correct.

Now if both the lieutenants are correct and the commander is faulty, and we try to concept to the integrity principle that we discussed earlier, then this lieutenant 2 it has heard and attack message attack instruction from the commander, and it sends back or echoes this attack message to the other lieutenant. So, in this particular case, we see that well by the message pressing principle the entire system will not be able to work.

(Refer Slide Time: 08:23)



Because this lieutenant it will have so, say here this lieutenant 1, it will have different messages from different persons different generals. So, it will get a retreat message from the commander where we yes it will get an attack message from the lieutenant 2.

So, because it has different messages. So, it will with the integrity condition if we try to do a attack, sorry if it try to do a retreat, then it will follow that principle, but whereas, this lieutenant 2 it got the attack message, if it attacks then obviously, the entire army will get defeated.

So, what we see here that, by this message passing principle if we try to obey to the commander, and if we do not have any way to find out that whether to go for the majority voting or whether to go for our the instruction that the commander has send, then the entire system is in a dilemma that which particular instruction to follow.

So, even under the byzantine general problem, we can solve this particular problem of byzantine failure with the principle of majority principle or the majority voting. That we have send in the case of a paxos or in the case of raft. So, if you remember the raft consensus protocol, which we have discussed in the context of a crash failure that the entire system works on the principles of a leader.

So, in case of the raft consensus we had a concept of the leader. And what the leader says all the other followers will follow what the a leader instructs. But in this particular case,

when the system is a kind of byzantine system and the leader itself so, here the commander if it sends different messages to different lieutenants, then coming up to a consensus on this principle become difficult.

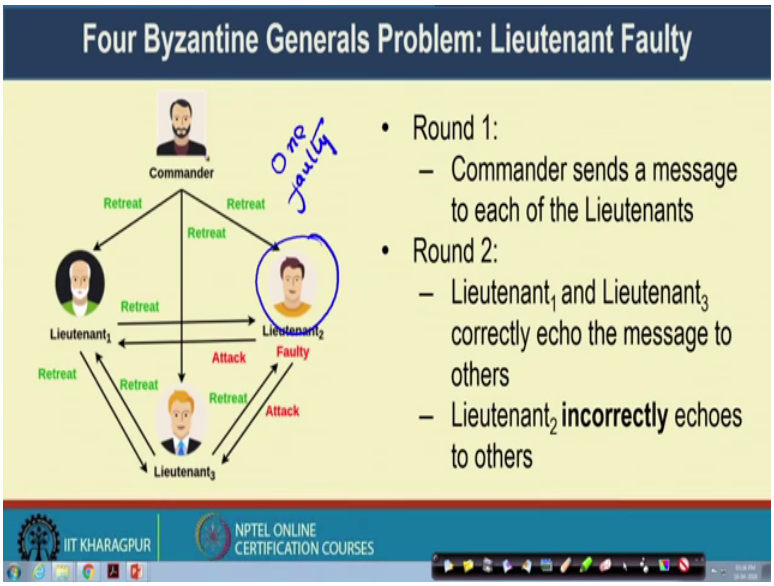
So, under this particular scenario what may happen? So, the protocol abstraction that we are looking into with the help of an example, that the lieutenants or the followers in the raft consensus terminology so, the lieutenants can start sharing the message among themselves.

And they can look into this majority principle construct like well, if the majority persons or the majority of the lieutenants votes against the commander, then we will assume that the commander is faulty. Otherwise we will we will consider that the lieutenant is faulty if we are not getting majority voting rather some individual lieutenant, who are differing from the message which is send by the commander.

So, if we try to apply this majority voting principle what we see that with 3 byzantine generals with one commander and 2 lieutenants the problem still remains unsolvable. Because the majority principle does not work, you get equal voting for attack and retreat. Or a lieutenant gets equal voting for attack and retreat, and a lieutenant cannot decide what to do. And if the lieutenant at that time tries to follow what the commander has asked he to do, then the lieutenant can take a decision which differs from the decision taken by the other lieutenant.

So, we will not be able to solve the byzantine general problem with 3 general's where we have one commander and 2 different lieutenants. So, that is one learning that well we will not be able to solve the byzantine general problem with 3 different commander a 3 different generals with one commander and 2 lieutenants, but let us look into another use case.

(Refer Slide Time: 12:13)



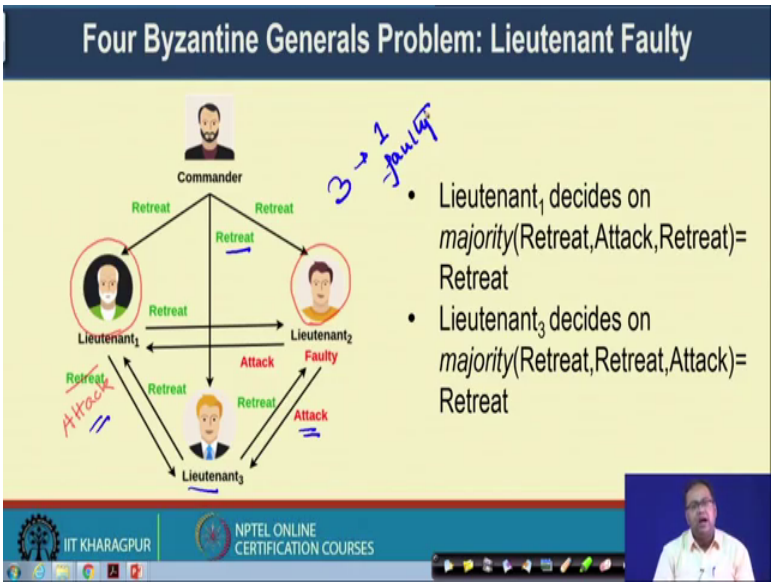
If we have 4 different byzantine generals, with this 4 byzantine general, we have 3 lieutenants. And every individual lieutenant to with the message passing principle that we have discussed earlier they talk with each.

So, here in this case as well. The commander it sends the message to all the lieutenants. And the lieutenants they share the message what the commander has shared with them with each other. But as we have discussed earlier, that the lieutenant can be faulty, and in this case we assume that one lieutenant is faulty. So, we have one faulty lieutenants, one faulty lieutenant. Now among this 3 lieutenants, if one is faulty let us see what happens for the 2 different cases. So, the first case we are going to consider is the lieutenant is faulty, and the case we look into when the commander is faulty.

Now when the lieutenant is faulty. So, the lieutenants sends message is which differ from the messages which is send by the commander. Now the commander here sends the retreat message to all the lieutenants. And this lieutenant 1 and lieutenant 3, they correctly echoes the message to other lieutenants. So, this lieutenant 1 it informs retreat to lieutenant 2 and retreat to lieutenant 3, but lieutenant 2 differs.

So, because lieutenant 2 is a faulty node here the faulty general here so, the lieutenant 2 incorrectly echoes the message to other. So, the lieutenant 2 sends an attack to lieutenant 1. And similarly it sends an attack to lieutenant 3. Whereas, the commander has send to retreat from the (Refer Time: 14:01).

(Refer Slide Time: 14:10)



Now, let us see what happens. So, here in this case, here in this case if we go for the majority voting principle. So, we are going with the majority voting principle. So, with the majority voting principle, we see that for this lieutenant's, it receives 2 retreat message one retreat message from the commander second retreat message from lieutenant 3.

So, the majority is lieutenants so, this lieutenant 1 is able to correctly decode what the commander has actually instructed for. Now in the second case, let us see what happens for lieutenant 3 which is a correct lieutenant. So, the lieutenant 3 receives retreat message. It receives this retreat message from the commander. It receives another retreat message from lieutenant 1, but it receives an attack message from lieutenant 2.

Now again if we go for the majority voting principle with the majority voting principle, it receives 2 retreat messages, and a one attack messages. So, with the majority voting principle it decides to retreat from the where. But this lieutenant is faulty; we do not consider the behaviour of the faulty lieutenants. So, in this case we observe that well our objective of a fault tolerant algorithm, if you remember, was the integrity principle like that all the correct lieutenants, or all the correct nodes in a fault tolerant system, they will follow the majority votes.

So, here we observe that even if lieutenant 2 is a byzantine node. Then also lieutenant 1 and lieutenant 3 is able to correctly decode the message from the majority waiting voting.

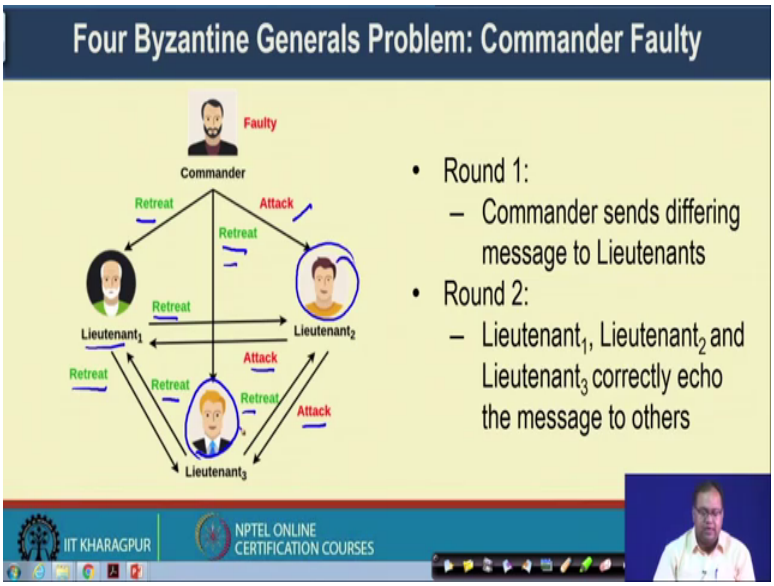
Now, if we if we further consider the majority the malicious behaviour or the byzantine behaviour of lieutenant 2. So, instant of sending an attack if the lieutenant 2 sends an retreat here. So, it is like that the lieutenant 2 is making a malicious behaviour, where it is sending the correct message to one node. So, it is sending attack to one node and it is sending this faults message to one node and the correct message to another node, even in this case we can find out that well this particular lieutenant it gets all the retreat message so, it remain retreat whereas, this with the majority voting principle lieutenant 1 can correctly decode the message.

So, with the lieutenant faulty, or better to say with one lieutenant faulty, we can correctly decode the message, in case of this byzantine general problem. But instead of one lieutenant to be faulty, if we take a different case where 2 lieutenants are faulty. Say, here we assume that this particular lieutenant is faulty, and at the same time lieutenant 1 is also faulty. If it happens, then lieutenant 1 can also start behaving maliciously. And rather than the correct retreat message if it sends the attack message at this stage.

So, if it sends an attack message at this stage, then we see that well for lieutenant 3 lieutenant 3 receives one retreat message from the commander, but 2 attack message one attack message from the faulty lieutenant 2. And another attack message from the faulty lieutenant 1. So, it will not be able to take the correct decision based on the majority voting.

So, here we what we observe that well, out of the 3 lieutenants if one lieutenant is faulty, if one lieutenant is faulty, then we will be able to correctly decode the message, well.

(Refer Slide Time: 18:04)

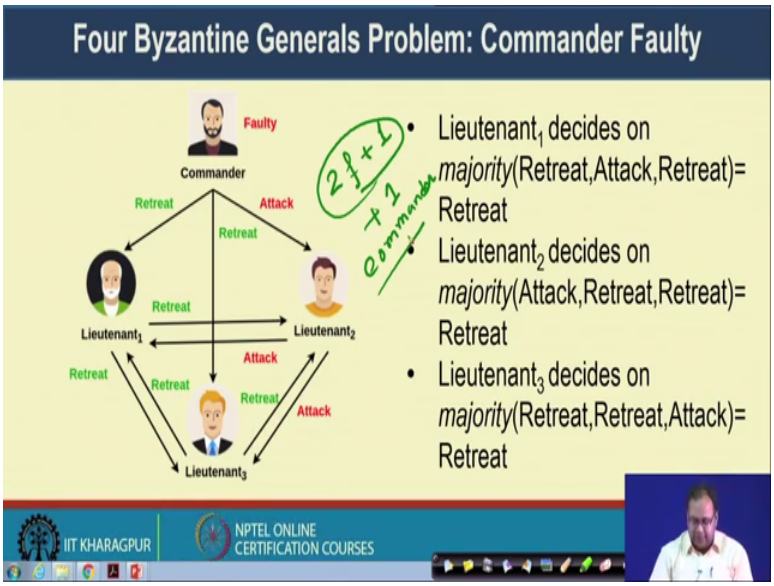


Now, let us see the second case of this problem with 4 byzantine generals, when the commander is faulty. So, the commander is faulty means, the commander is starting behaving maliciously. Now in this case, in this malicious behaviour of the commander, now say the commander sends the retreat message to one lieutenant, the retreat message to the second lieutenant, but an attack message to this lieutenant to the third lieutenant.

So, if the commander starts behaving maliciously, but the lieutenants are behaving correctly let us see what happens. So, in this case the lieutenants all the lieutenants are correct, because all the lieutenant has correct, this lieutenant 1 it has heard a retreat message from the commander so, it correctly echoes the retreat message to other 2 lieutenants. Now lieutenant 2 it is again a correct lieutenant, but it has heard an attack message.

So, these lieutenant 2 send an attack message to the other 2 nodes. Whereas, lieutenant 3 it is a correct lieutenant as we have seen earlier. So, the lieutenant 3 as has heard and retreat message. So, it correctly broadcast the retreat message or let us say multicast a retreat message to other 2 lieutenants.

(Refer Slide Time: 19:25)



Now, let us see whether the majority voting works in this case or not. So, we first look into the case of lieutenant 1. So, lieutenant 1 has received one retreat vote. Then 2 retreat votes, and one attack votes. So, it has received 2 retreat votes and one attack vote. Because it has received 2 retreat votes and one attack votes, it can currently decide that the decision is retreat.

Now for lieutenant 2; lieutenant 2 it has received one attack vote from the commander, but it has received one retreat vote from lieutenant 1 and another retreat vote from lieutenant 3. So, it has lieutenant 2 it has received one attack vote, from the commander one retreat vote from the lieutenant 1 another retreat vote from the lieutenant 3. So, it can decide based on the majority principle that the retreat is the correct decision.

And if retreat is the correct decision, it can correctly finds out that the commander he was indeed faulty. Now let us look into the case for the third lieutenant. For the third lieutenant we have for the for the third lieutenant we have this retreat message which has been received from the commander, then one retreat message which has not this one this, retreat message which has been received from lieutenant 1. And one attack message which has been received from lieutenant 2.

So, it has received retreat message from commander, retreat message from lieutenant lieutenant 1, but attack message from lieutenant 2. So, again from the majority voting principle, it can decides that the correct decision is retreat. So, again in this case we can

observe that well, if you have 3 different generals who are correct or better to say 3 different lieutenants who are correct, and you have one commander who is behaving maliciously, again you need to ensure that you need to ensure or you can ensure that the majority voting principle will give you the correct results.

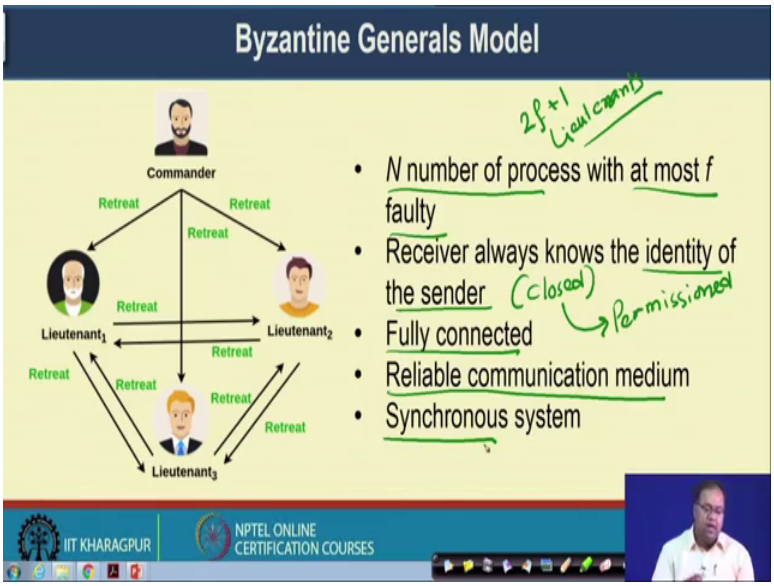
Now, if we look into a general principle, in that general principle, if we if we look into the normal byzantine general problem with f number of faulty nodes. Then we can say that well if a lieutenant is faulty in that case, we need to ensure that there are 2 f plus 1 number of correct, 2 f plus 1 number of lieutenants total lieutenants in the system. So, if we have 2 f plus 1 lieutenants plus 1 commander, then we will be again able to correctly apply this majority voting principle to find out the byzantine nodes in the system.

Now again in this entire system, if we see that well if the commander sends message or this attack message to more than one lieutenant. So, if it sends attack message to this lieutenant and rather than a retreat if it sends to a attack message to this lieutenant as well, then whatever majority decision or the majority message that the commander is sending, the entire system will come to the will take that particular message in the consensus.

So, if the commander is sending 2 attack message to 2 lieutenants and one retreat message, then in the consensus the entire system will leads to the attack message by this majority voting principle. So, our learning or our take while from this discussion is that, if you have this 2 f plus 1 number of lieutenants, and one commander, and out of them if the lieutenant is faulty, then the system will come to a consensus with the value that the commander has proposed.

And if the commander is faulty, in that case the system will come to other consensus with the value that the commander has proposed to majority of the lieutenants. So, that way, we can some to the consensus with 4 byzantine generals. And let us try to look into a general description of this particular algorithm.

(Refer Slide Time: 24:17)



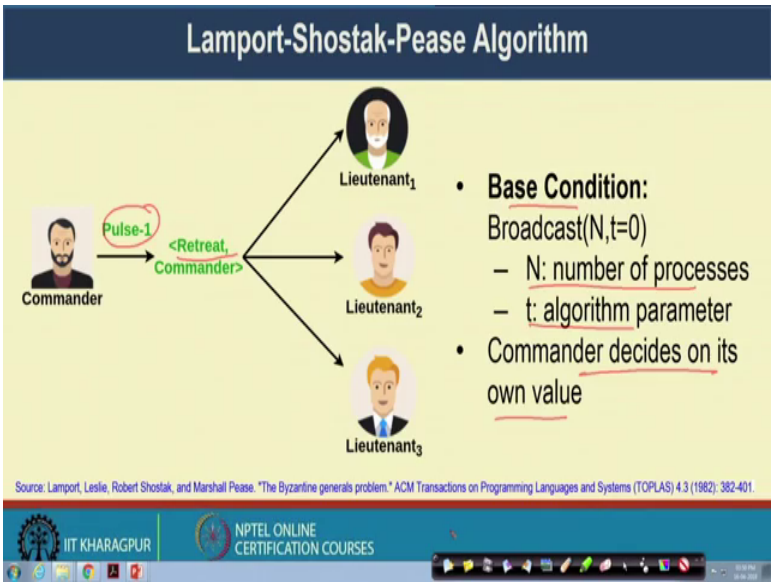
So, the algorithm works in this way, we call this entire model as byzantine general model. So, in the byzantine general model we assume that there are N number of process out of which at most f number of process can be faulty. Now when we say f number of process are faulty, we need to ensure that in the system, we have 2 f plus 1 number of lieutenants 2 f plus 1 number of lieutenants in the system.

And the receiver it always knows the identity of the sender. So, that is our closed model in the context of block chain technology, this model help us to design an algorithm for the permission block chain environment, permission block chain environment so, that system is fully connected, the communication is a reliable communication medium and the system is synchronous.

So, system is synchronous means, every node will be able to receive all the messages within some predefined time out duration. So, remember the impossibility theorem in this context that we have discussed earlier, that in an fully asynchronous system, even if a single node is faulty the system will not be able to reach a consensus within a predefined timeout interval.

So, we do not look into the asynchronous nature of the system at this moment. So, we are trying to develop an algorithm which is a synchronous in nature so, this synchronous algorithm this particular algorithm that I have just discussed right now with an example.

(Refer Slide Time: 26:00)

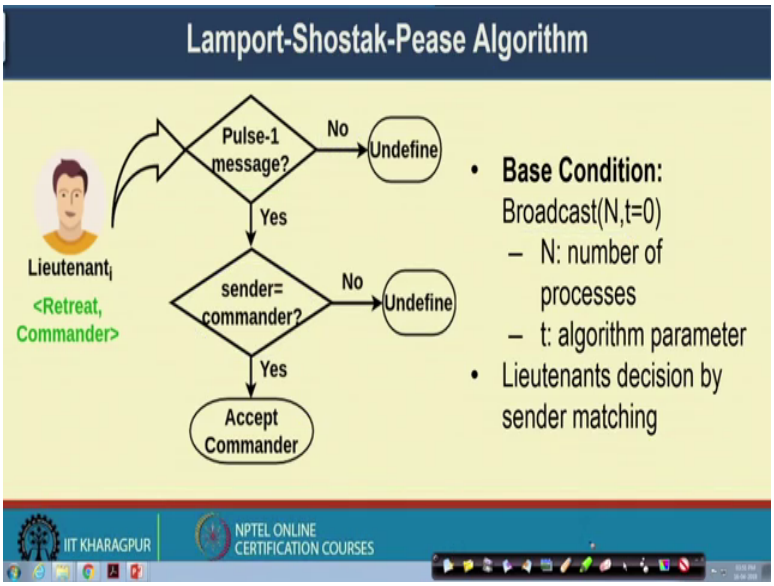


So, this algorithm is known as Lamport-Shostak-Pease algorithm, which was one of the first algorithms for having this kind of byzantine general problem. That was published in this ACM transaction on programming language and system, the title of the paper was the byzantine general problem. So, in this particular case the commander sends the message at pulse one. So, the pulse one is the initial pulse when the commander sends the message to all the lieutenants. So, we are first considering a base condition.

So, this base condition we have 2 parameters here. N is the number of processes that are there in the system. And t is the a algorithm parameter. That denotes the individual rounds. So, t basically indicates t equal to 0 indicates that you are in pulse 0 when the commander is sending a message to all the lieutenants. So, the commander it decides on it is own value, whether to go for a retreat or to go for an attack.

And at pulse 0 it broadcast this particular message. So, here it is the retreat message it is an example to all the lieutenant. So, here we consider n equal to 3, because we have 3 different process among which we are trying to reach to the consensus. So, we send the 3 messages to 3 different lieutenants. Now that is the base condition for the commander.

(Refer Slide Time: 27:31)

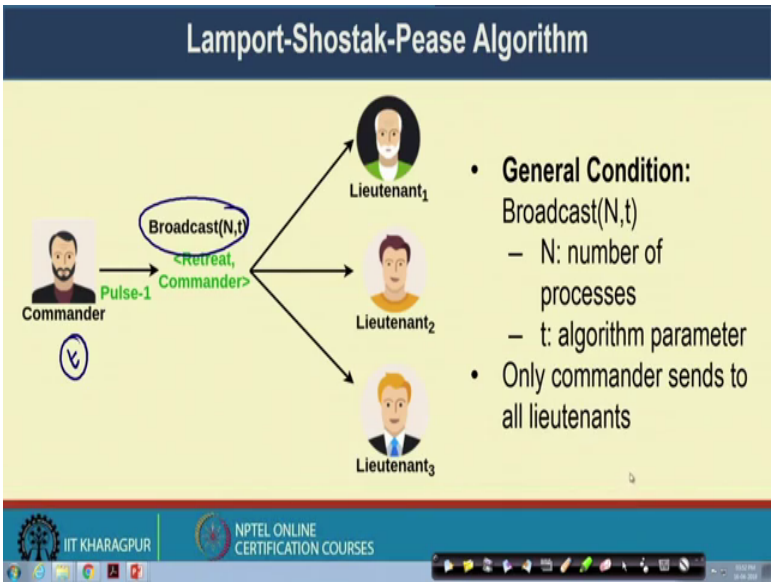


Now, let us look into the base condition for a lieutenant. So, the lieutenant it receives the message from the commander. Whenever it is receives the messaging from the commander, it first does is to check whether it is a pulse one message or not.

So, it is a pulse one message means that the message is coming from the commander. If it is know; that means, it is not a pulse one message, then you cannot take any decision, because you do not know that from where the message is coming from. If it is a pulse one message; that means, the first message in the system, and the message is coming from the commander, then you accept what the commander is saying.

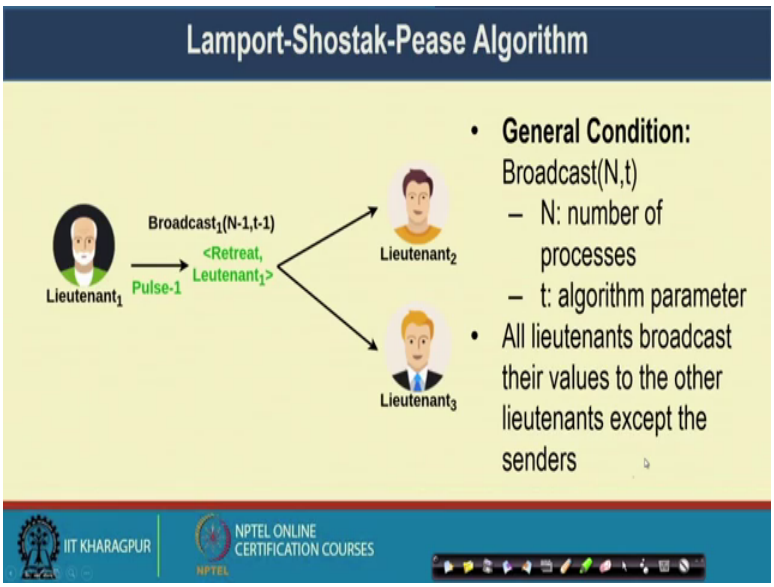
Otherwise the system goes to done defined state. So, what the commander is saying ah? You then broadcast that particular message to all other processes in the system. So, that is the initiation now at every individual round the system progress in round as we have mentioned.

(Refer Slide Time: 28:31)



So, at every individual round commander t so, we can say that this is a commander the commander n commander sorry, you can you can just think of it as a tth commander, or commander t who is sending this message. So, this commander t it broadcast again the message that it has received from the commander. So, this is the commander. So, the commander it sends this message to all the lieutenants in the tth round.

(Refer Slide Time: 29:18)



And similarly for the lieutenants, every individual lieutenant's whatever message it receives from the commander; it broadcast this message for to all other lieutenants,

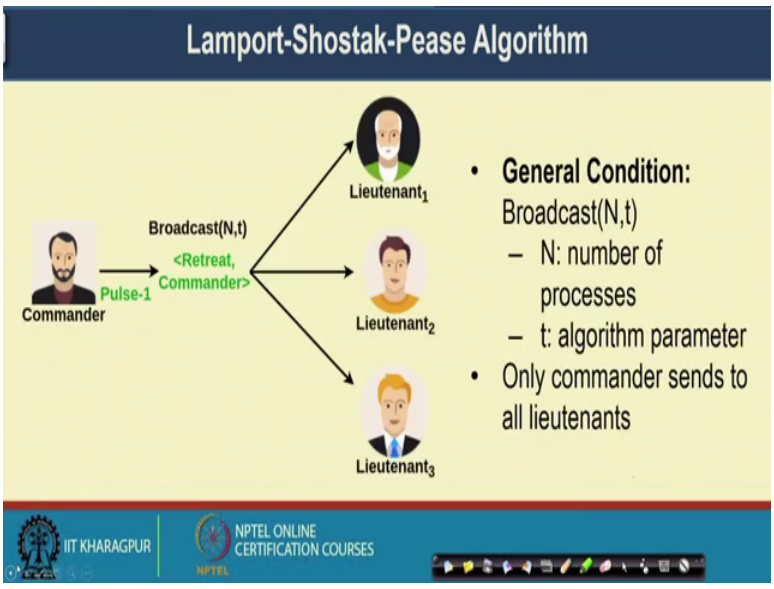
except the sender of that particular message. So, it does not broadcast the message to the commander, rather it broadcast is message to all other lieutenants in the system.

Now, with this broadcasting of messages at every individual round what may happen if you have N number of lieutenants in the system? Then after nth round you are getting the messages from all the individual lieutenants. And as we say that the system is synchronous.

So, you are known to get the messages from all other lieutenants who are there in the system after nth round. So, once you receive all this messages, then you can apply the majority voting principle. So, every individual lieutenant they apply the majority voting principle.

And by applying the majority voting principle they decides based on the majority voting, what should be their decision whether the decision would be to go for what the commander is saying or to go for what the commander is saying to the majority of the lieutenant. So, the example that we have seen for the 4 general problem. So, here in a general condition the algorithm as I have mentioned that the algorithm works in this principle.

(Refer Slide Time: 30:51)



Like, the commander at every individual round. The commander is sending the messages broadcasting the messages to every individual lieutenants. And once this lieutenant get

the message, it again broadcast that message to other lieutenant's except the sender. So, here the sender is the commander. So, every lieutenant gets the messages from all other lieutenants along with the commander and they apply the majority voting principle, what the majority voting says it takes that particular decision.

So, this is a algorithm for achieving consensus which was developed primarily by Lamport for a synchronous environment, when you know that you are going to receive the messages within some predefine timeout interval. And if you are able to receive the messages within a predefine timeout interval, then you can apply the majority voting principle to decide what should be your decision.

So, if the majority voting principles says that you go for attack then you should go for attack if the majority voting. Says, that you should go for retreat then you can go for retreat. And in this case if a lieutenant is faulty, until you have sufficient number of lieutenant in the system, like if there are f number of lieutenants which you are assuming to be faulty, and you have 2 f plus 1 number of lieutenant in the system you can simply assume that with the majority voting principle you will accept only the message which the majority of the correct lieutenant says.

Now, if the commander is faulty, in that case, the scenario is little tricky, that the commander is sending one vote to one group of lieutenants and another vote to another group of lieutenants. And if you have 2 f plus 1 number of lieutenant in the system so, you can just think of in this way that if you have 2 f plus 1 number of lieutenant in the system, you already always have some odd number of lieutenants.

And when the odd number of lieutenant's the commander has no way of saying some majority decision say f or f number of lieutenants it will say one decision and for remaining f lieutenants it may say another decision. So, what the commander is saying in this case what the commander is saying to the majority of the nodes, you will be able to decides on that majority voting on that on that majority value. So, in this case we 2 f plus 1 number of lieutenants and one commander. We can achieve consensus in a synchronous system when, you have either the commander is faulty or one of these 2 f plus 1 lieutenants are faulty.

Sorry, not one of 2 f plus 1 lieutenants are faulty, the f number of lieutenants are faulty. So, if f number of lieutenants are faulty with 2 f plus 1 number of lieutenants, you will be

able to achieve consensus. Otherwise if the commander is faulty, but all the lieutenants are correct then also you will be able to achieve consensus under this Lamports algorithm.

So, this is an consensus algorithm which is for a asynchronous environment, but as we understand that our practical networking systems environments are asynchronous which does not behave like a asynchronous environment. And in that asynchronous network, it may happen that you will not be able to receive message within a predefine timeout.

So, in the next class, we will look into another class of consensus problem ah; which can work or which we say that it provides safety principle or safety guarantee a for asynchronous network but there is certain a condition on the liveness principle. We call this algorithm as the practical byzantine fault tolerant algorithm or PBFT algorithm which is widely used in a block chain environment. So, in the next class we will look into the PBFT algorithm in details.

So, thank you for today's class.