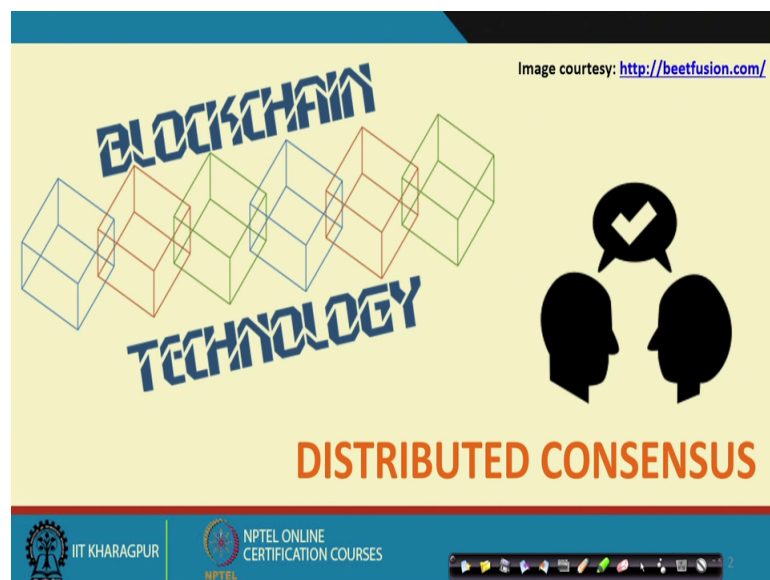


Blockchains Architecture, Design and Use Cases
Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 10
Distributed Consensus

Welcome to all of you in the course of a Blockchain. So, till now we have discussed about the permission model of blockchain. And we have looked into the details of a bitcoin network as an example of permission blockchain. And, we have seen different aspects of a bitcoin network where user can join in the network and start doing the transaction.

(Refer Slide Time: 00:44)

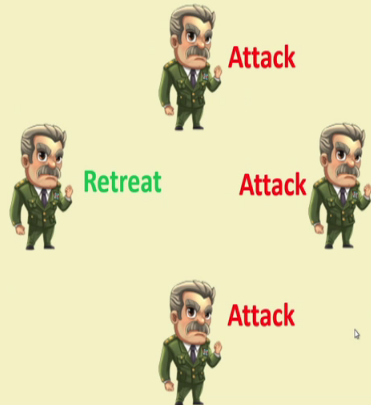


So, in this particular class a we will discuss about a one interesting property of a blockchain network or rather an set of algorithms, which help in achieving consensus in a distributed or decentralized network. So, we look into the details different methods of consensus and a how they are applicable for a general blockchain environment and with a special case of bitcoin network.




(Refer Slide Time: 01:11)

Consensus

- A procedure to reach in a common agreement in a distributed or decentralized multi-agent platform
- Important for a message passing system



The diagram shows four cartoon generals in green uniforms. Three of them are pointing towards the word 'Attack' in red text, while one is pointing towards 'Retreat' in green text. This visualizes a majority-based consensus where the group decides on 'Attack'.

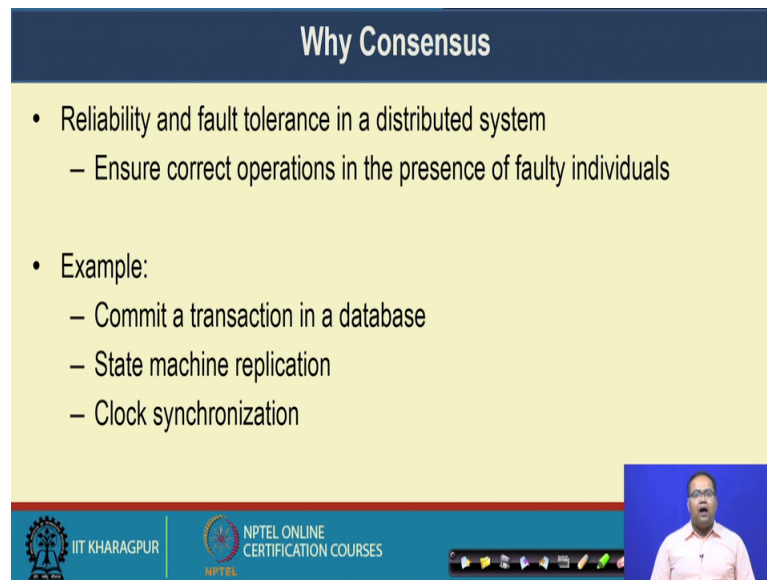
So, this concept of consensus is an interesting topic in a decentralized or in a distributed network. So, consensus means it is a procedure to reach in a common agreement in a decentralized or in a distributed platform. So, here is an example of a typical example of a consensus mechanism you can assume, that in an army there are 4 generals and the generals are taking some kind of decisions. And when the generals are taking the decisions in a decentralized or in a distributed way; that means, they have their own policies to take the decision they can either make an attack or can reflect from the attack.

Now, in a consensus algorithm these generals individually they express their opinion their individual opinion and from their individual opinion by applying some kind of choice function, which can be the majority decision in this particular case.

So, by applying such kind of choice function the environment or the system finally, decides what to do next. So, here in this particular example that 3 generals they are making a choice towards attack. So, with the majority principle the system can come to a consensus that they should make an attack collectively.

Now, this kind of consensus algorithm it is important for a message passing environment in a distributed system.

(Refer Slide Time: 02:38)



The slide is titled "Why Consensus" in a dark blue header. The main content area is light yellow and contains a bulleted list. The first bullet point is "Reliability and fault tolerance in a distributed system", with a sub-bullet "Ensure correct operations in the presence of faulty individuals". The second bullet point is "Example:", with sub-bullets "Commit a transaction in a database", "State machine replication", and "Clock synchronization". At the bottom of the slide, there is a blue footer with the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a small video inset showing a man in a pink shirt speaking.

- Reliability and fault tolerance in a distributed system
 - Ensure correct operations in the presence of faulty individuals
- Example:
 - Commit a transaction in a database
 - State machine replication
 - Clock synchronization

So, let us first look into that why do we require consensus. So, in a traditional or conventional distributed system we apply consensus to ensure reliability and fault tolerance. So, by means of reliability and fault tolerance it is like that in a decentralized environment when you have multiple individual parties and they can take their own decision, then it may happen that some nodes are, some parties are, some individuals are working as maliciously or they are working as a faulty individual.

So, in those particular cases it is important to come to a common decision or a common view point. So, having a common view point in an environment where people can behave maliciously or people can crash or work as a faulty way it is a difficult thing. So, under this kind of distributed environment our objective is to ensure reliability; that means, to ensure correct operations in the presence of faulty individuals.

So, there are multiple examples for a consensus in a distributed system if you think of a distributed database, where the transactions are going on from multiple points of sales say; from multiple ATM's, multiple banking sectors the transactions are coming in during that time consensus as an important aspect say for example, you want to transact some money from one branch to another bank branch another account in another bank branch.

So, during that time you need to come to a consensus that all the bank branches need to decide that well this transaction is a valid transaction and after deciding that this

transaction is a valid transaction they should commit the transaction, then another example of consensus is state machine replication. So, the state machine replication is an important aspect of any distributed protocol.

So, say for example, if you want to run some kind of distributed protocol over a network. Every individual nodes runs the current protocol current version of the protocol and this stood the state of the protocol in different state machines. So, so the entire execution part of the protocol can be represented as a state machine.

Now, these state machines need to be replicated into multiple nodes. So, that every individual node can reach to a command point or command output of that protocol. So, the state machine replication is an example of consensus, then another example of consensus is clock synchronization or distributed clock synchronization. Say you have multiple clocks in your network and every individual node tries to find out that, which is the most updated clock or which is the most current clock.

So, they should make a consensus among themselves and come to a current a come to a single clock and by applying this kind of clock synchronous asynchronous clock architecture across the network they can do further operations.

So, these are the typical examples of consensus in a traditional distributed system environment.

(Refer Slide Time: 05:51)

Why Consensus Can be Difficult in Certain Scenarios

- Consider a message passing system, and a general behaves maliciously

The diagram shows three nodes (G1, G2, G3) in a network. G1 is at the top, G2 in the middle, and G3 at the bottom. Red arrows labeled 'Attack' point from G1 to G2, G2 to G3, and G3 to G1. A green arrow labeled 'Retreat' points from G2 to G1. Blue circles highlight G1 and G2. A blue checkmark is next to G2. A small inset video shows a speaker.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, let us look into that while achieving consensus can be difficult in certain scenario or typically in a message passing system. So, here the example that we have considered earlier so, we are considering that example again that we have now multiple generals. Now these generals are utilizing and message passing environment to communicate their viewpoint to others.

Now, you can just think of that every individual generals they are making a telephone call to other general and then communicating they are view point to others. Now, let us look into this diagram. So, this general it is sending it is information to it is neighbouring generals this one and this one. Now whenever general is sending the information the general makes a telephone call to the other 2 general say G 1 and G 2, and a and inform his view point that they should now attack. Similarly this general say the general G 3 he again makes a phone call to G 2 and G 1 and send his decision that the soldiers should now make an attack.

But, this particular general this general say G 1 this is a malicious general and this general make a malicious call like a he makes a call to 1 general say G 4 and ask for retreat. Whereas, whenever he is making a call to general G 3 he is saying as an attack.

Now in a complete distributed or a decentralized environment the generals may get confuse say for example, whenever these general is informing G 1 is informing G 4 to retreat G 4 can get are confused that whether or what he should do?.

So, that way in a in a decentralized or a distributed platform achieving consensus over a message passing system can be difficult, when you have this kind of malicious node or you have nodes, which starts working maliciously. So, we have a technical term for this kind of nodes we called them as the a byzantine time nodes are we call this kind of failure as byzantine time failures.

So, later on we will discuss about this kind of byzantine failures in details, but in brief that in the presence of this kind of byzantine failures the system can behave maliciously or it may be difficult to achieve a consensus in a distributed environment.

(Refer Slide Time: 08:23)

Distributed Consensus

- If there is **no failure**, it is easy and trivial to reach in a consensus
 - **Broadcast** the personal choice to all
 - Apply a **choice function**, say the maximum of all the values

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So but as we know that, if there is no failure in the system, then it may be easy and trivial to reach in a consensus. So, in a genetic algorithm can be something like this like you broadcast the personal choice to all, and then you apply your choice function say in this case if your choice is the maximum of all the receive value, then you can you can achieve consensus.

So, here are 4 nodes individual nodes make a choice of 10, 20, 30 or 40 and they informed they are individual choices to all other nodes in the network, and whenever every node receives all the choices from all the neighbours they can apply on max function to find out that what is the maximum. So, you can easily see that everyone will reach to the value 40, if they apply the maximum function of all the received values. So, whenever we are saying that achieving consensus in this particular architecture is easy and straight forward it is under certain scenarios.

So, this scenario search first of all the system need to be need to be a faultless there should not be any failure in the system. So, that every individual node can receive the message correctly and, but second requirement is that the system should behave in a synchronous way. So, what is mean by synchronous way we will look into the formal definition of a synchronous message passing system, but by definition synchronous message passing system is something, where it is expected that you will receive all the messages within some predefined time interval.

So, here in this particular architecture every individual node is expected to receive all the messages from the peers in a non failure environment, within a predefined timeout and every node can wait for that much of time and in a synchronous system, it is expected that everyone will receive the message.

So, once they receives the message they can find out the maximum of all the received values from different messages and take that as a value for the consensus.

(Refer Slide Time: 10:23)

The slide is titled "Distributed Consensus" and lists three types of faults in a distributed system:

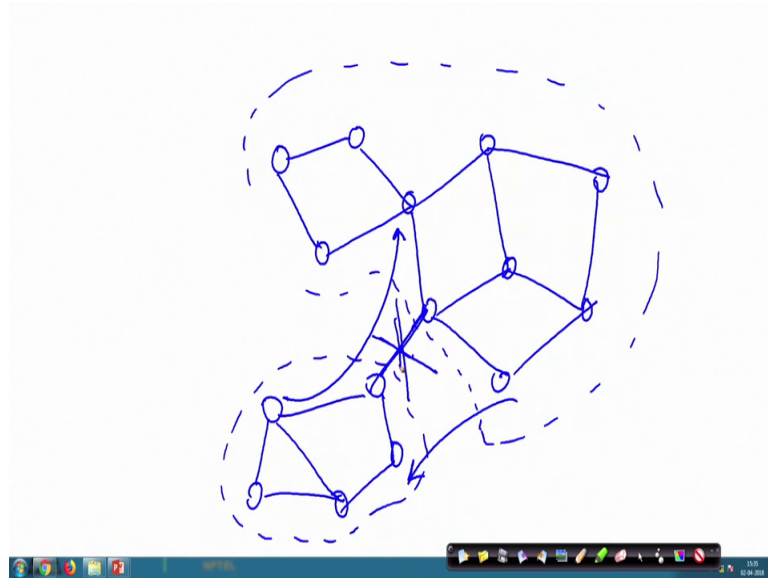
- There can be various types of faults in a distributed system.
- **Crash Fault:** A node suddenly crashes or becomes unavailable in the middle of a communication
- **Network or Partitioned Faults:** A network fault occurs (say the link failure) and the network gets partitioned
- **Byzantine Faults:** A node starts behaving maliciously

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

However, a consensus achieving consensus can be non-trivial in case of a distributed environment due to the presence of multiple type of failures. Now typically in a distributed system we consider 3 different type of failures the first one is called as the crash fault. So, a crash fault is something like a node suddenly crashes. So, the node suddenly crashes or the nodes become unavailable in the middle of a communication. So, you are not expected to receive any message from that particular node.

So, this is one type of typical fault that can be a kind of hardware fault or a software fault due to which the node or the process, which is communicating with another one that particular process, fails. The second type of fault is network fault or the partition fault a network fault is something when a network link fails. So, a network failure may result in a partition in the network. So, we can we can see an example of that.

(Refer Slide Time: 11:30)



So, say assume that there are multiple nodes in the network and these nodes are interconnected to each other.

So, we have multiple nodes in the network and these individual nodes are interconnected now. So, we have few other nodes they are interconnected with each other. Now in this particular node if this node fails then the, if this link fails then the entire network get partitioned into 2 different parts. So, we have one partition here and we have a second partition here. So, the entire network get partition and you are not expected to receive any message from any node of this partition to any node of this partition or the vice versa.

So, this message communication will not happen, because you had you had a bottleneck link and that particular bottleneck link has failed. So, under this kind of network failure, because this kind of partition can happen in the network we say that it is a type of a partition fault, we used the term partition fault to denote this kind of network failure. So, this can kind of network failure can hamper reaching in the consensus. So, if a network failure makes a partition of the network the nodes in one partition they becomes unavailable to the nodes in another partition.

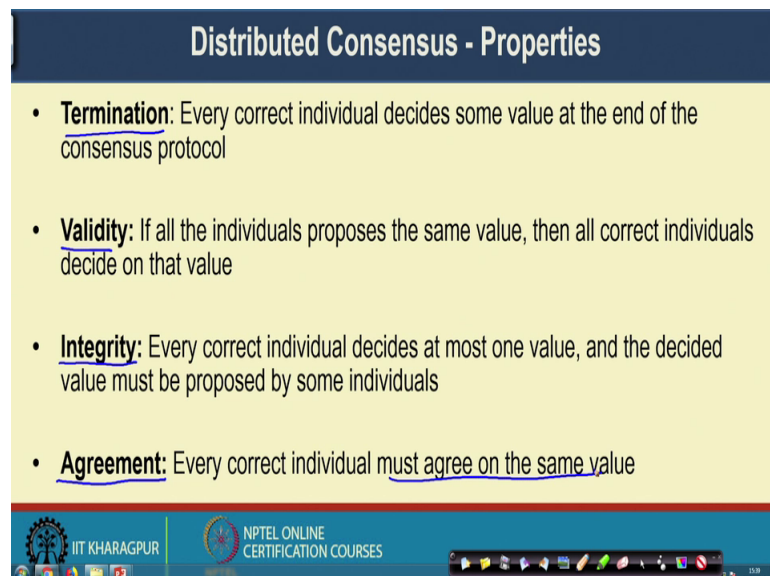
So, all the nodes cannot communicate with each other and come to a general consensus. So, these are the 2 typical type of faults which are coming from either the hardware failure or a software failure, but a third type of faults that is the byzantine faults it is the

kind of more difficult fault to handle in a distributed environment. So, byzantine fault is just like the example that I have shown you earlier here the node start behaving maliciously.

Now, whenever a node starts behaving maliciously you do not know that what would be the action for that node. The node can the node can send a positive vote or sometime the node can send the negative vote. In case of say the crash fault or in case of a network or a partition fault, it is still expected or you can find out or you can make a expectation that what is going to be the effect of a network fault or a crash fault, but the effect of a byzantine fault is difficult to guess, because it completely depends on how maliciously the node is behaving and what the node is doing. So, sometime the node can give a vote against a, the consensus or sometime the node can give a vote in for of the consensus.

So, that is why handling byzantine nodes become difficult in a in a typical distributed system, but while we are dealing with different kind of consensus protocols you have to deal with this 3 fault 3 different types of faults. Now a whenever we are talking about distributed consensus protocols we have need to satisfy certain properties of the protocols. So, the first property is called termination.

(Refer Slide Time: 14:46)



The slide is titled "Distributed Consensus - Properties" and lists four key properties of consensus protocols:

- **Termination:** Every correct individual decides some value at the end of the consensus protocol
- **Validity:** If all the individuals proposes the same value, then all correct individuals decide on that value
- **Integrity:** Every correct individual decides at most one value, and the decided value must be proposed by some individuals
- **Agreement:** Every correct individual must agree on the same value

The slide footer includes the IIT KHARAGPUR logo, NPTEL ONLINE CERTIFICATION COURSES text, and a system tray with a clock showing 12:39 on 10/10/2018.

So, the termination properties says that a every correct individual decides and some value at the end of the consensus protocol so; that means, who ever be the correct or non-faulty node in the network the non-faulty node must terminate the protocol, and decide

on one value, and that value should be correct value. Then the second property is the validity the validity property says that if all the individual proposes the same value, then all correct individual decide on that value.

So, so that is that is the basic idea of these validity property which says that if all the individuals in the node they propose on the same value like the example that we have shown earlier, that if all the individual propose a value 10 then every correct nodes should come to a consensus with value 10 they should not deviate from that particular value.

So, that is the validity property of consensus protocol the third properties integrity. So, the integrity property says that every correct individuals decides at most one value. And the decided value must be proposed by some individual. So, the integrity property ensures that the consensus value should not deviate from the values which are proposed by individuals in the network. So, you should not get a value of 20 in the consensus, if none of the nodes in the network proposes a value of 20.

So, this is the integrity or the third property of a consensus protocol, the fourth property of the consensus protocol is agreement the agreement properties says that every correct individual must agree on the same value.

So; that means, and that is the most important property of a consensus, protocol that all the individuals in the network they must agree on the same value. So, whenever they agree on the same value after termination we call that the system has reached to the consensus well.

(Refer Slide Time: 16:46)

The slide is titled "Synchronous vs Asynchronous Systems" in a dark blue header. The main content is on a light yellow background and lists two types of message passing systems. The first is the "Synchronous Message Passing System", where a message must be received within a predefined time interval, offering a strong guarantee on message transmission delay. The second is the "Asynchronous Message Passing System", where there is no upper bound on the message transmission delay or the message reception time, meaning messages can be delayed for arbitrary periods.

Synchronous vs Asynchronous Systems

- **Synchronous Message Passing System:** The message must be received within a predefined time interval
 - Strong guarantee on message transmission delay
- **Asynchronous Message Passing System:** There is no upper bound on the message transmission delay or the message reception time
 - No timing constraint, message can be delayed for arbitrary period of times

The slide footer includes the IIT Kharagpur logo, the text "NPTEL ONLINE CERTIFICATION COURSES", and a Windows taskbar with various application icons.

So, while dealing with consensus with deferrer we consider 2 different type of message passing system, one type of message passing system we call as the synchronous message passing system and the second one is the asynchronous message passing system.

So, in case of synchronous message passing system so, the message must be received within a predefined time interval. So, we have a kind of strong guarantee on the message passing delay and you know apriori that what can be the maximum delay of message passing for this particular network. Now, this kind of synchronicity give you simplification in designing the protocol, in the sense that you can think of that you will wait for certain duration that duration will be the maximum lifetime or maximum bound on the message delay and if you wait for that amount of duration, it is guaranteed that you will receive all the messages within that time.

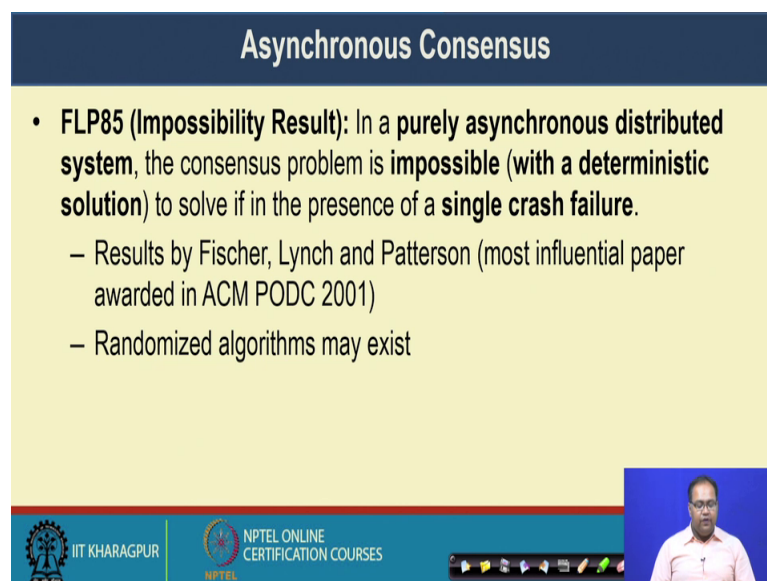
So, this is the synchronous message passing system, but in case of asynchronous message passing system on contrary we do not have any upper bound on the message transmission delay or the message reception time. So, you do not have any such constraint message can be arbitrary delayed or the message delivery time can be arbitrarily long. So, in case of a asynchronous message passing system you cannot expect that if you wait for a finite duration you will receive all the messages with some certain probability or with some guaranteed probability.

Now, designing a distributed system in a synchronous environment is much easier, because you have that particular strong assumption on the message passing delay. And that is why if you wait for that amount of duration it is expected that you will receive all the messages, but in case of an asynchronous message passing system because you do not have that much of delay. You have to also deal with the kind of fault that may come due to the asynchronicity nature of the system.

That means, it may happen that you are waiting for certain duration and you may not receive any vote for some of the neighbours, because the message, which is coming from those nodes in the network they have observed some kind of unbounded or sometime of very long delay.

So, if you wait for certain amount you may not receive the messages from all the neighbours. So, having consensus under this kind of environment is much difficult compared to designing a consensus protocol for synchronous distributed system.

(Refer Slide Time: 19:24)



Asynchronous Consensus

- **FLP85 (Impossibility Result):** In a **purely asynchronous distributed system**, the consensus problem is **impossible (with a deterministic solution)** to solve if in the presence of a **single crash failure**.
 - Results by Fischer, Lynch and Patterson (most influential paper awarded in ACM PODC 2001)
 - Randomized algorithms may exist

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | NPTEL

Now, there is an interesting result we called this result as FLP 85 or sometime people call it as an impossibility result. So, the impossibility result states that in a purely asynchronous distributed system, the consensus problem is impossible with a deterministic solution, if there is a single even a single crash failure in the system.

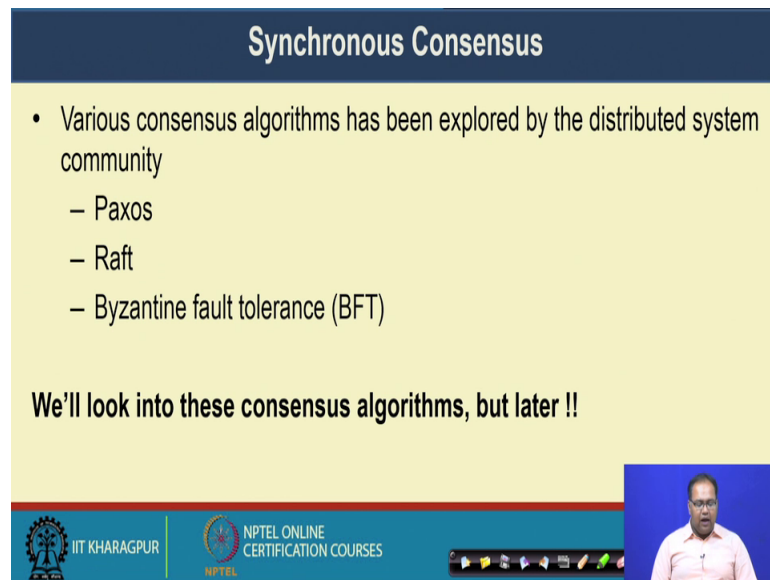
So, in case of a purely distributed environment or purely asynchronous distributed environment, if there is a single fault in the system you cannot design any kind of consensus protocol or better to say any kind of deterministic consensus protocol for that particular system. But in this particular impossibility theorem note that term deterministic consensus protocol, it is true that will not be able to design any kind of deterministic consensus protocol or we cannot have any kind of deterministic solution, but we can always designs some kind of randomize solution or some kind of probabilistic solution for purely asynchronous distributed system in the presence of failures.

But, this particular a environment gives on a bound on what type of consensus protocol you can design for a distributed environment. And this is the foundational paper foundational work for a distributed system, which was initially proposed by Fischer Lynch and Patterson in 1985 fan in one of the top Asian Conference and Distributed System ACM Policy 2001 this paper got the most influential paper award.

So, I suggest all of you to look into the formal proof for the impossibility result we are not going to that details, if you are interested you can you can look into this paper and look into the formal proof that why it is impossible to achieve a consensus in a purely asynchronous distributed system in the presence failure.

But, but ideally this gives us a idea about in for what type of system you can design a consensus protocol.

(Refer Slide Time: 21:37)



The slide features a dark blue header with the title "Synchronous Consensus" in white. Below the header, on a light yellow background, is a bulleted list of consensus algorithms. At the bottom of the slide, there is a blue footer containing logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a speaker on the right side.

Synchronous Consensus

- Various consensus algorithms has been explored by the distributed system community
 - Paxos
 - Raft
 - Byzantine fault tolerance (BFT)

We'll look into these consensus algorithms, but later !!

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Well, now whenever talk about synchronous consensus they are exist various time of type of consensus algorithms, that people have design in a traditional distributed system environment; like the Paxos raft byzantine fault tolerance BFT type of a consensus algorithms people have tried to implement many of this type of consensus algorithm for asynchronous system as well by apply probabilistic nature inside it.

So, we look into all those different type of consensus algorithm later, but, but ideally this gives us an algorithm that what type of consensus you can design for a distributed algorithm.

(Refer Slide Time: 22:16)

Correctness of a Distributed Consensus Protocol

- **Safety:** Correct individuals must not agree on an incorrect value
 - Nothing bad happens
- **Liveness (or Liveness):** Every correct value must be accepted eventually
 - Something good eventually happens

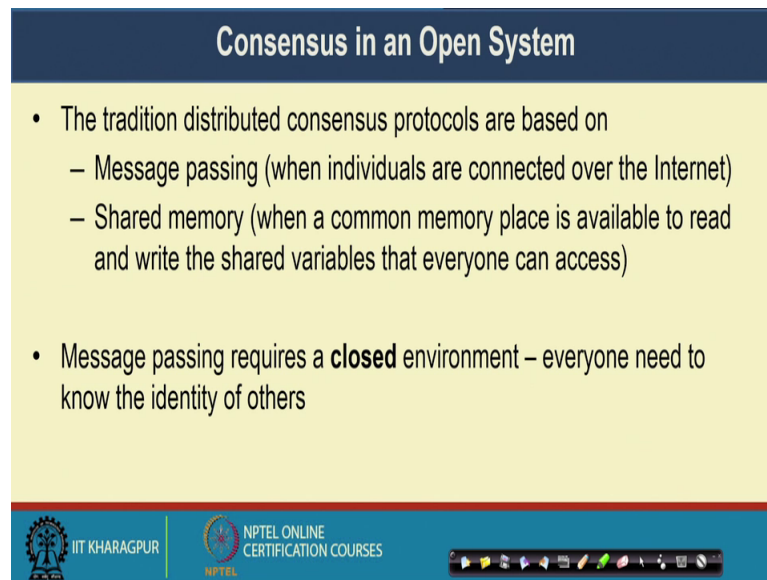
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, the correctness of a distributed consensus algorithm, that can be characterized by these 2 properties safety and liveness. So, the safety properties say that the correct individual must not agree on an incorrect value. So; that means, nothing bad has happened in the system. So, so, the safety property ensures that, you will you will never converts to an incorrect value or the correct individuals in the network they will never converts to an incorrect value. And the liveness property or in some book are difference manual they like it they like it as liveness property.

So, the liveness property states that every correct value must be accepted eventually so; that means, something good will eventually happen. So, if you have proposing some good values. So, that good value will be committed eventually although there can be some time lag or there can be some delay in reaching into the consensus, but after the consensus protocol terminates you will you will expected to get a get a consensus value out of that.

So, these are the 2 correctness properties for a distributed consensus that we need to ensure whenever we design a distributed consensus algorithm.

(Refer Slide Time: 23:30)



Consensus in an Open System

- The traditional distributed consensus protocols are based on
 - Message passing (when individuals are connected over the Internet)
 - Shared memory (when a common memory place is available to read and write the shared variables that everyone can access)
- Message passing requires a **closed** environment – everyone needs to know the identity of others

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us look into the consensus algorithms from the perspective of a blockchain environment. So, the type of blockchain environment like Bitcoin that we have discussed till now, is a kind of open environment. Open environment in the sense like any node can join the network any time, but a type of distributed systems that we are considering till now, like synchronous system synchronous distributed system and the type of consensus algorithms, that have been explored for a distributed system the traditional distributed system those kind of consensus algorithms realize on a message passing environment.

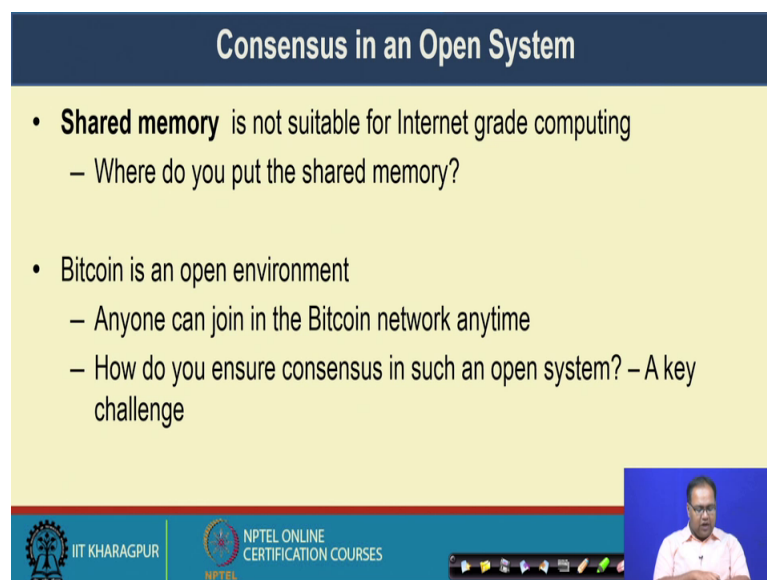
So, a message passing environment means the every individual transfer some kind of messages to others and then for waiting for some finite amount of time they receive all the information from the peers, and after receiving all the information, from the peers whatever information they have received till now they process that information, and based on that information they reach to a or they design a consensus parameter to come into the consensus.

But, because this kind of traditional distributed consensus algorithm like Raft Paxos or Byzantine fault tolerant algorithm, they rely on this kind of message passing environment, they are designed particularly for a closed environment. So, in a message passing environment you need to know that which node you want to transmit a message.

Now, the moment you have this constant like the moment you need to know that which node you want to send a message you need to know the identity of all those nodes.

So, that way this kind of distributed message passing algorithms for having a consensus was mostly design for a closed environment , but for blockchain type of environment in bitcoin applications you are mostly talking about open environment or earlier you have used the term as a permission less environment, where any node can join in the network any time. Now, under this kind of environment having a consensus protocol based on the message passing argument is difficult, because you do not know that who are the nodes in your periphery to whom you want to send a message.

(Refer Slide Time: 25:48)



The slide is titled "Consensus in an Open System" and contains the following text:

- **Shared memory** is not suitable for Internet grade computing
 - Where do you put the shared memory?
- Bitcoin is an open environment
 - Anyone can join in the Bitcoin network anytime
 - How do you ensure consensus in such an open system? – A key challenge

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.

Now, in case of a consensus in open system we have to broad type of algorithms the shared memory is one architect that people have explore, but shared memory architecture it is not suitable for internet grid computing, because you need to put a memory which should be readable and writable by every individual nodes in the network. In general the shared memory algorithms we apply whenever we try to reach consensus among multiple distributed process inside a system.

But, we apply the message passing in the other case, but as I we have looked into that message passing is not feasible in a open environment. Because in an open environment like bitcoin anyone can join the bitcoin network any time and under this kind of open

environment the challenges that how will you ensure consensus under this particular scenario.

(Refer Slide Time: 26:45)

Why Do We Require Consensus in Bitcoin Network

- Bitcoin is a peer-to-peer network
- Alice broadcast a transaction in this peer-to-peer network
- **All the nodes in this network need to agree on the correctness of this transaction**

The diagram illustrates a transaction broadcast. On the left, a woman representing Alice is shown. A red arrow points from her to a network of nodes on the right. The transaction details are shown in two boxes: a blue box labeled "Signature of Alice" and a pink box labeled "Pay to P_{pub}^{Bob} : H()".

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, let us look into that how do we ensure consensus and even before going to that, let us look into that why do we require consensus in a bitcoin network, whenever we are utilizing blockchain as the backbone of a bitcoin. So, as we have looked into the bitcoin is a peer to peer network and in this peer to peer network say let us take an example, that Alice broadcast her transactions a Alice made a transaction to bob and she broadcast this transaction in this peer to peer network.

Now, remember that this kind of broadcast is different from traditional message passing system, for a traditional message passing system you need to know that which are the peer nodes to whom you need to send a message, but this kind of broadcasting it works like a flooding. It works like a flooding in the sense like you send the message to the subset of the people whom you know, then those nodes will further send a message to another subset of the people, those people will again send it to another subset of the people this way by flooding the message in the network the message will get propagated.

So, that way it is different architecture of this flooding environment is different from a traditional message passing environment, where you need to know that who are the neighbours to whom you need to send a message? Now, flooding is most costly more

costly costlier compared to a normal message passing environment. So, that is why we restrict the flooding to certain applications.

Now, in case of a bitcoin peer to peer network whenever Alice is proposing a transaction Alice broadcast the transaction in the peer to peer network. Now here we required the consensus, because all the nodes in this network need to agree on the correctness of these transactions. How will you ensure that the transaction is actually coming from Alice and not from an attacker? The attacker may also impersonate Alice and can send an transaction in the network in the name of Alice. So, the network needs to find out come to a consensus that this particular transaction is coming from Alice and not from an attacker.

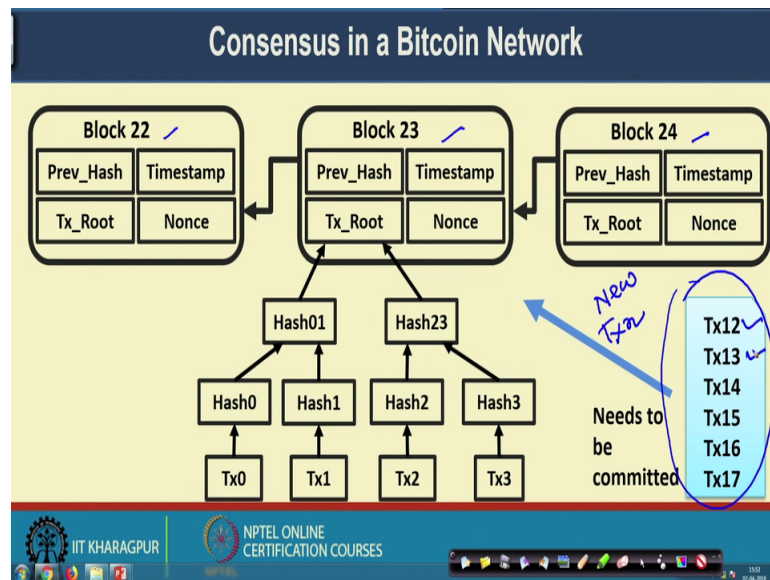
So, as I have mention that in this particular peer to peer network architecture and node does not know all the pairs in the network, because this is an open environment. So, in under that open environment some nodes can also initiate malicious transactions and you need to prevent the system from having this kind of malicious transactions.

(Refer Slide Time: 29:33)

The slide is titled "Consensus in a Bitcoin Network" and contains two bullet points. The first bullet point states: "Every node has **block of transactions** that has already reached into the consensus (**block of committed transactions**)". The second bullet point states: "The nodes also has a list of outstanding transactions that need to be validated against the block of committed transactions". At the bottom of the slide, there are logos for IIT Kharagpur and NPTEL Online Certification Courses, along with a navigation bar.

Now, in case of bitcoin network every node has a block of transactions that has already reached into the consensus. So, that are the blocks of committed transactions, which is stored in the form of a blockchain. And every node also has a list of outstanding transactions that need to be validated against the existing transactions in the block; that means the existing blockchain. So, that is an example.

(Refer Slide Time: 29:51)



So, you already have a block of transactions. So, these are the individual block of transactions block 23, 22, 24, inside every block the architecture we have looked earlier. So, you have a merkle root under that all the transactions are organized in the merkle tree. Now, every node has this blockchain and they have a set of new transactions. So, these are set of new transactions.

Now, this new transactions need to be committed to the block then need to be added in the blockchain. So, to commit these existing transactions the network need to validate, that these transactions are coming from the authenticated person and not from an attacker. And for this we require the consensus protocol in a bitcoin network.

(Refer Slide Time: 30:47)

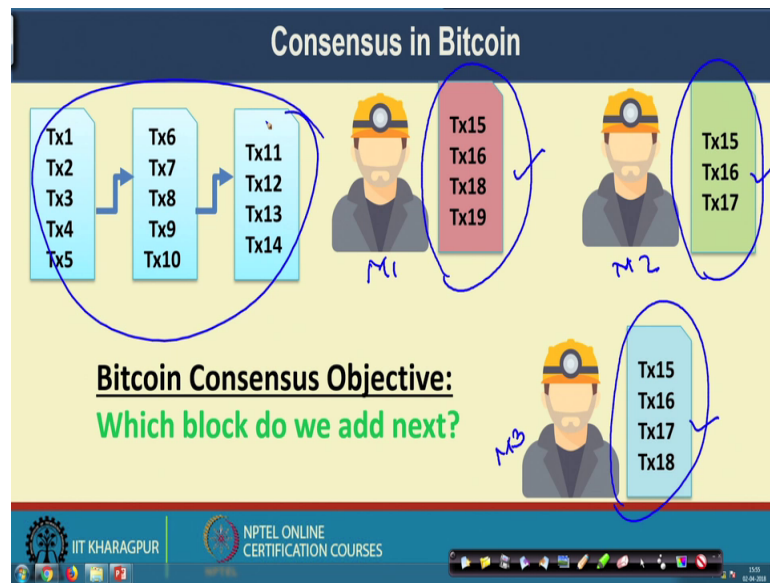
The slide is titled "Consensus in Bitcoin" and compares two consensus methods. On the left, "Per transaction consensus" is listed as "Inefficient". On the right, "Block based consensus" is shown with a "New Block of Transactions" containing Tx12, Tx13, Tx14, Tx15, Tx16, and Tx17. A blue arrow points from this block to the text "Apply consensus over the entire block of transactions", which is followed by the green text "Here comes the Blockchain". The slide footer includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a speaker.

So, whenever we talk about the consensus in a bitcoin network. So, this is an interesting fact that why we are applying something called a blockchain. So, you can always have a per transaction consensus. So, you take every individual transaction and then validate that transaction.

But, if you do for per transaction consensus it is inefficient, because you have to run the transaction algorithm for every individual transaction. So, what we do rather than running the transaction running the consensus algorithm for every individual transaction, you run the algorithm on a block of transactions. So, that way you can made the consensus algorithm much efficiency you do not need to run the consensus for every individual transaction rather you running for a block of transaction.

So, here comes the concept of blockchain that, we are we are representing the transactions in the form of blocks or and we are not representing it in the form of a transactions chain rather we are we are representing it in the form of a blockchain for an efficient implementation of a consensus.

(Refer Slide Time: 31:54)



So, that is the objective of bitcoin consensus algorithm. So, earlier we have looked into the concept of miners who actually participate in the consensus algorithm.

Now, here we have 3 individuals or 3 miners M1, M2 and M3. So, everyone has one blockchain in hand now everyone has constructed or observed T's transactions from the clients, now they construct a new block of transactions. So, M1 has constructed a new block of transactions from the transactions that he or she have heard of, M2 has constructed another block of transactions, M3 has constructed the third block of transactions.

And interestingly it is not necessary that every miner will heard the same set of transactions, there can be always the difference between the set of transactions that would be heard by the individual miners, and that is why the proposed blocks are different for different miners. So, the bitcoin consensus algorithm the objective is that among these 3 block which block I will add to this existing blockchain.

So, this is the problem of bitcoin consensus algorithm. So, with this we will stop today. In the next class we will look into that how the bitcoin proof of work algorithm actually achieves the consensus under an open environment. So, see you all during the next class and.

Thank you.