

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

Lecture – 14
Permissioned Blockchain – I (Basics)

Welcome to the course on blockchain. So, in the last few lectures we have looked into the details of a permission less block blockchain settings, where we have looked into the architecture and system internals of a bitcoin based protocol. And we have went to the details of how are bitcoins based system works and what are the different design internals of a bitcoin based architecture where blockchain has been used as a fundamental building block. So, the bitcoin type of blockchain network that we have learnt till now it uses a model which we call as the permission less model.

So, in that permission less model setting ideally what happens that every node can join in the network anytime without having any preauthentication or pre authorisation mechanism. It is like that anyone can join in the network anytime without going to an authentication procedure and that is why we had a settings in permission less blockchain were you would require a consensus among the nodes.

But that kind of consensus we cannot use, the conventional distributed system protocols and we went to our channel challenge response base method, where the network has thrones certain challenge to the users or the nodes in the blockchain. And then the users have collectively try to solve a problem and the node with wings to find out the solution for the challenge that particular node ensures or advertise the updated blockchain to it is neighbor.

In contrast to the permission less setting where it is more like an open environment we have another kind of initiative from the industries specifically which we call as the blockchain to and in that particular settings the industry was more focused towards the blockchain application or towards the application of public ledgers, where you have certain number of the nodes which is a kind of closed environment.

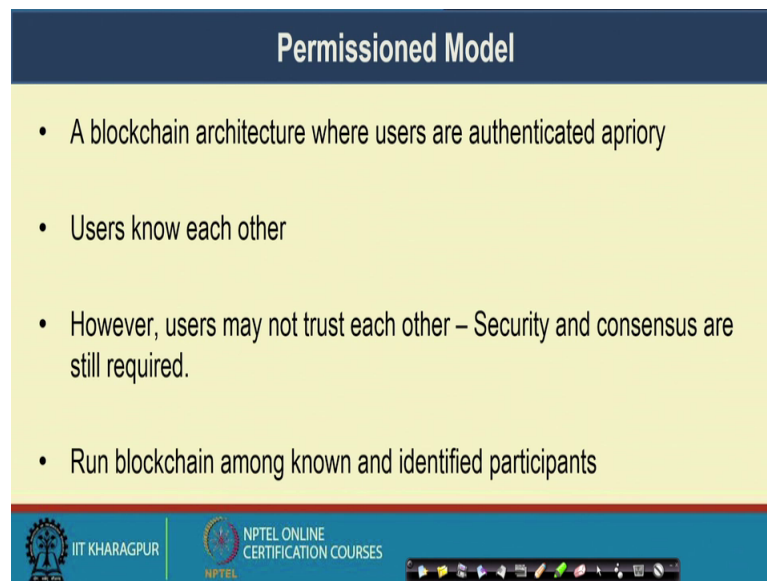
So, the nodes which knows each other and deep tries to come to a common platform where there is no such centralized database or data server rather you still want a

complete decentralized architecture, but the difference is that now rather than having a complete open environment you have a closed environment.

So, you have a set of nodes which knows each other a priori, but they may not trust each other. So, that particular settings we call it as a permission the environment in contrast to a permission less environment where nodes cannot join to the blockchain network anytime rather before joining to the blockchain network they have to go through some authentication or preauthorization mechanism through which they can validate themselves.

So, in this particular lecture series so, in the next couple of hours we will look into the details of this permission blockchain environment and the different aspects inside.

(Refer Slide Time: 03:42)



Permissioned Model

- A blockchain architecture where users are authenticated a priori
- Users know each other
- However, users may not trust each other – Security and consensus are still required.
- Run blockchain among known and identified participants

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, as we have mentioned that of permission model is something where the blockchain architecture inside the blockchain architecture the users are authenticated a priori. In that particular settings that uses know each other, but they may not trust each other it may happened that certain nodes in the network in the blockchain network although they got a pre authenticated they are authenticated to use the system.

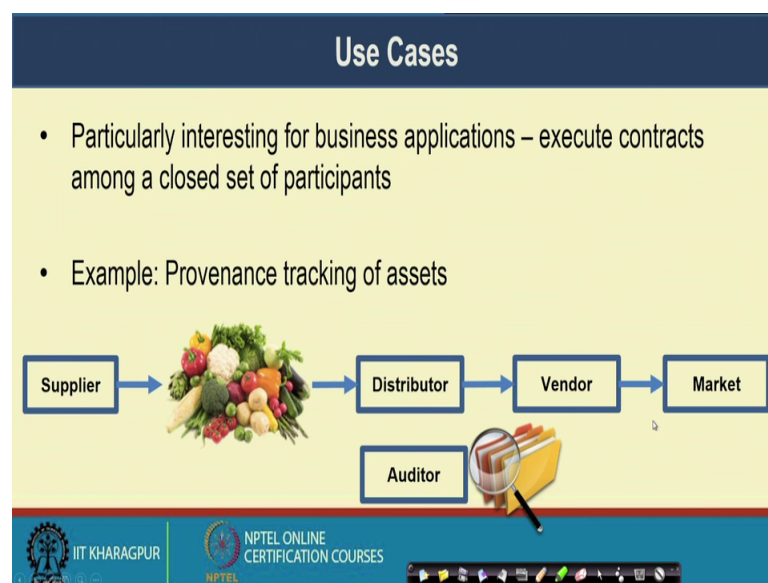
But they have started behaving maliciously typical use case you can think about certain business environment where you want to go for a business with a set of different levels of users.

Now, in a in a think of think of a scenario of a typical organization or a typical company in industry where in the company every user is authenticated every user knows each other or at least there is certain database where you have the records of the users who belongs to that particular company, but that does not prevent the users to behave maliciously.

So, the users can behave maliciously anytime they can make some kind of cheating or fraud with the with the entire system and under that particular scenario we have an architecture where the users are pre authenticated the user knows each other, but they may not trust each other because some users can behave maliciously.

So, here our objective is to run the blockchain among this known and identified set of participants.

(Refer Slide Time: 05:24)



So, let us look into certain use cases of this permission environment to an interesting use case of permission environment is for business application where you want to execute certain contracts among a closed set of participant.

So, earlier we have look into the details of this concept of smart contracts where rather than having the contracts on paper you write down the contacts in the form of a code and that code will be uploaded to a blockchain environment where the code will get executed that is a that is the typical scenario of a permission blockchain environment.

So, one interesting use case of this can be like this provenance tracking of asset so, in this provenance tracking of assets. So, what you want to do you want to ensure that whenever certain as it is moving from say one particular suppliers to the distributors to a vendor and going to the market every stage there is certain locks which are being maintained and inside that lock you make an entry that this particular asset has now move from location A to location B.

Now, the interesting fact is that why do not you want to use a centralized server to have this entire lock of data and then anyone can just look into that central lock and can verify indeed. Now a days we use that kind of architecture say for example, whenever you are sending certain courier service via say blue dart b h p or any kind of courier platform what you normally do that whenever the you are you are later or your courier is moving from one particular location to another particular location.

The courier agency they are making an entry to their central database that will this particular location say the location A the centre at location a has received as the parcel and they have verified the parcel and then they have verified the parcel send a parcel to say location B the courier agent at location B. Now, whenever it is moving from one agent to another agent that different location across the entire pass from the source to destination every individual agent this making and entry to central database.

Now everyone else in that blue dart or b h p they can look into that lock and they can find out that where that particular asset is moving, for how that asset is moving from the user one user to the destination to the final discipline. Now in the centre architecture the users are also in the loop. So, whenever you want you can make a login to their blue dart blue dart website or you can provide a reference number or what they call as the waybill number and. So, that waybill number you can track that where your parcel it is currently located that and what is the expected time that it will reach at the destination.

Now, within this kind of centralized architecture whenever the entire thing is under control blue dart it is good to use a centralized server because the centralized server is maintained by blue dart, but think of a typical scenario when the particular thing is moving from one say courier company to another courier company and third courier company. So, a typical use case can be like you are sending a postal mail from say India

to somewhere in USA. So, whenever you are sending a postal mail to India to somewhere in the USA.

So, the India post they are one authenticated domain and India post will basically do the transfer of the post up to their border gateway or the border point from there they will send it to some international agency or some international courier transfer postal transfer who will then take the mail or take the courier to your USA post and then USA post will do it transfer it internally.

Now, in this entire process you have multiple authorities who take cares of your parcel, but whenever you have this kind of multiple authorities it is difficult, but there is a kind of trust issues whenever you are going to rely on a centralized server so, the question that comes first of all who will host that server.

If you want that will India post the will host the server then the question comes that why USA post will trans or believe the data which is there in the India server and also the USA a postal they have to have an access to the central database which is being maintained by the India post. So, because they have different kind of policies or different kind of environment it becomes difficult to handle this kind of multi authoritative scenario when you have multiple authoritative domains and they try to put the information to a central data base.

Now the second problem comes like if none of the India post or USA post will host that particular database central database rather they may purchase the database from some third party agent like they may think of that well we will now host this things to central database the which will be hosted by a third party cloud. Now, if they host it on a third party cloud then the questions comes like you have to pay a significant amount of money for that third party cloud.

The second thing is that because they are multiple parties multiple authoritative domains they require certain kind of access to that central server and the question comes that how will you guarantee that the date of which is entered by the USA agent or entered by the USA post that is not getting tampered by the data which is a being entered by India post or vice versa.

So, whenever there is this kind of multiple authoritative dominos in the loop you all are you have this kind of problems in trust relationship and that is why are people do not want to go for any kind of centralized server. Now it is if you want to send something from India post to USA post or vice versa what you have to do like whenever the get or whenever your parcel is within India you can login or you can look into the India post website to look into the tracking details and then whenever it is going to USA post we have to looking to the US post website to go to looking were go to the details of the tracking.

Now the information that is shared by India post to USA post or vice versa that will only give you some insight or a kind of partial information like how your asset or your post is moving your parcel is moving from India to USA or coming to India from USA. Now for this kind of provenance tracking of assets, we are interested to use the blockchain environment and the beauty of the blockchain environment is that you do not require any kind of centralize server to be hosted the data need not to be in a centralized server the data would be in the individual hand of India post and USA post, but everyone will be able to validate adders data.

So, that is the typical use case of permission blockchain environment where you know that well the use cases are going to be the India post or going to be the USA post. So, you have a closed set up participants who are participating in the entire blockchain environment, but it is like that there is still the trust relationship is not there and. So, you have to maintain a certain kind of security or you have to ensure that the data is not getting tampered while transferring from one authoritative domain to another authoritative domain. .

So, the similar case happens whenever you are transferring certain kind of goods between the suppliers and distributors. So, you have multiple suppliers multiple distributors and multiple vendors. Now, it is like that every individual supplier distributor or vendor their individual authoritative domain they have their own policy of entering data, but a third party auditor should have access to this entire data and they should be able to reliably verify that whatever data is being passed through supplier distributor vendor and the final market those data are indeed the correct data.

To ensure this kind of environment in a distributed setting we want a permission model or you want to use the permission model. Now well one point here is like you can always use a permission less model for this kind of settings, but there are certain disadvantages of using a permission less model because you are going for a open environment and be whenever you are going for a open environment your network or your system becomes more complex.

And you have to handle a lot of things all together and that is why we want to move from a permission less model to a permission model. So, we will look into the certain aspects of a permission model indeed it is well in respect to this permission model earlier we have briefly looked into the concept of smart contracts here.

(Refer Slide Time: 15:26)



The slide is titled "Smart Contracts" in a dark blue header. The main content is on a light yellow background and consists of two bullet points. The first bullet point defines a smart contract as "A self executing contract in which the terms of the agreement between the buyer and the seller is directly written into the lines of code" and includes a URL: <http://www.scalablockchain.com/>. The second bullet point discusses "bitcoin scripts" and lists two sub-points: "Your friend can use that money immediately" and "Your friend can use that money after 2 months". At the bottom of the slide, there is a blue footer containing the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a navigation bar.

Let us go to the little details of this concept of smart contract. So, as we have discussed earlier that as smart contract is a self executing contract, in which the terms of agreement between the buyer and seller is directly written into the lines of code. So, it is like that you are not using a pen and paper to write down the contract rather the entire contract is written inside the code. So, the question comes that how this kind of contract may get executed or how a blockchain can help you to execute this type of contracts.

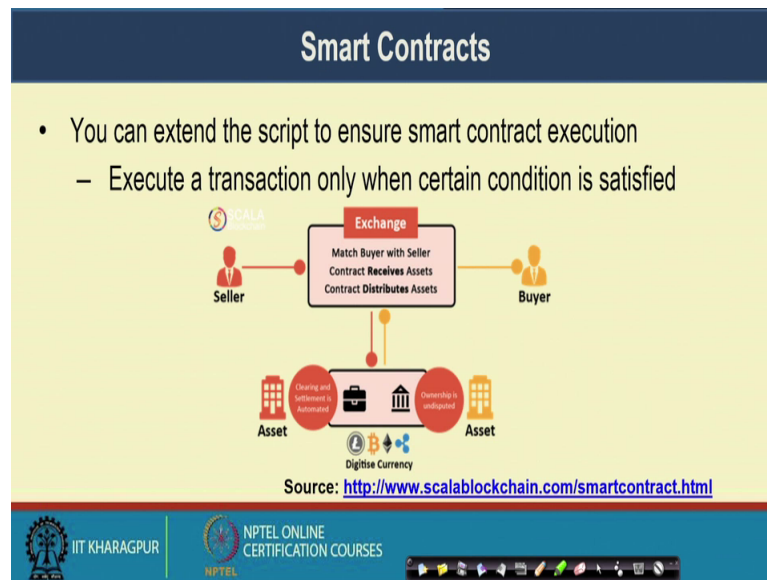
So, if you remember the concept of bitcoin script that we discussed earlier. So, bitcoin script is certain thing or certain small code through which you can change the control on how the money that is you are transferring to your friend that will be spent latter on. So,

by writing into the bitcoin script by writing the rules into the bitcoin script if you remember we had this input script and output script and our task could be to map the output of the previous transaction to the input of the next transaction which will give you an idea or which will give you a verifiable way of finding out that how the bitcoin that you are transferring to your friend or how the bitcoin that you are transferring to your claim that is going to be used later on.

So, you can write a script where your friend can use that money immediately or at the same time we have seen that you can write a script inside bitcoin where you can prevent your friend to use that money for 2 months it is just like that 1 or 2 month, month type line duration gets over your friend will be able to use that particular bitcoin for further transactions. Now in case of smart contract we actually expand the idea further where it is not like that some small simple scripts which are being used by bitcoin rather with detailed that is scripts through some programming language or some general purpose programming language.

So, can you can execute complex type of contracts like if certain conditions are met then only this particular bitcoin will be spendable. So, if your friend meets certain condition or if some environmental condition that can come from the other third parties they get validated or they made, during only during that time the contract will get executed and it will be used further or the money can be used further after all these conditions get verified it. So, the concept of smart contract comes from this fact like you can extend those simple script which is used in bitcoin to ensure the execution of smart contract one typical.

(Refer Slide Time: 18:20)



Example is that you execute a transaction when certain condition is satisfied. So, you have a seller you have a buyer and you have certain assets and the contract mix the match with the buyer and seller now whenever you are buy something there are certain scripts that will be executed at the seller side and there will be certain scripts that will be executed at the buyer side. So, whenever someone is selling something they have a contract like if I get this much amount of money then I can transfer the asset from myself to you and the buyer the idea is that the moment I have transferred this much of money the asset should be in my hand within certain duration.

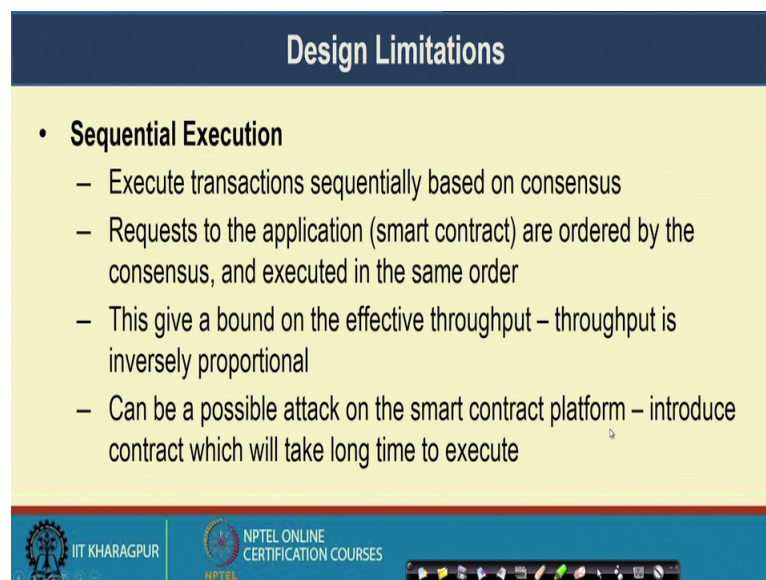
So, you can write a contract to ensure this kind of this kind of environment. So, it is like that the moment the buyer transfers the money to the seller. So, the moment of buyer transfer the money to the seller the ownership of the asset goes from the seller to the buyer and the code that you have written that particular code, actually validate whether the buyer has transferred at certain amount of money in general you can think of in terms of bitcoin or any kind of digital currency. It is like that the buyer the moment the buyer has send that certain amount of currency to the seller and the contract actually verifies that well this amount of money has been transferred from the buyer to the seller and then it sends of sends the ownership of that particular asset from the seller to the buyer.

So, the seller will not be able to claim the ownership on that asset any further whereas, the buyer will be able to claim his or her ownership on that particular asset. So, that is

the concept of smart contract which we can execute in closed environment with the help of blockchain. So, here the fact is that in a typical business platform you have fixed number of sellers or fixed number of buyers or you can say that there are limited numbers of buyers and sellers in the market and the buyers and sellers can always register to a central portal so, that everyone can know each other.

So, we are making an insurance that everyone knows each other the buyer knows seller and the seller knows buyers, but there may not be any trust relationship between them it happened that some seller is fraud and that seller it is just taking the money and give not giving the asset to you. So, that kind of fraud that, kind of malicious activities in the system we want to prevent with the help of a permission blockchain.

(Refer Slide Time: 21:26)



The slide is titled "Design Limitations" and contains a bulleted list of points. The background is light yellow with a dark blue header and footer. The footer includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a navigation bar.

- **Sequential Execution**
 - Execute transactions sequentially based on consensus
 - Requests to the application (smart contract) are ordered by the consensus, and executed in the same order
 - This give a bound on the effective throughput – throughput is inversely proportional
 - Can be a possible attack on the smart contract platform – introduce contract which will take long time to execute

So, when let us look into many of the design limitations which are there in a permission environment. So, if you look into the blockchain concept which actually comes from this typical permission less environment like a bitcoin type of examples they are one issue is that we execute the transaction sequentially based on the consensus. So, it is like that if certain transaction if it gets verified or if it gets committed in one transaction that will first executed and the transaction that is committed latter on that will be executed next.

So, the request to the applications, the applications here of the smart contracts they are ordered by the order of the consensus in which the individual application of the individual contacts get a consensus and they are executed in that particular order.

So, these kind of sequential order actually gives about on the effective throughput because you want to ensure that certain consensus or certain ordering of transactions are made we apply that proof of work base techniques in case of your permission less model were the network or the system close the challenge to that users individual users. And every user tries to solve that particular challenge individually and the nature of the challenge is such that it is difficult to find out a solution for that challenge, but once a solution is found everyone can verify it very easily. .

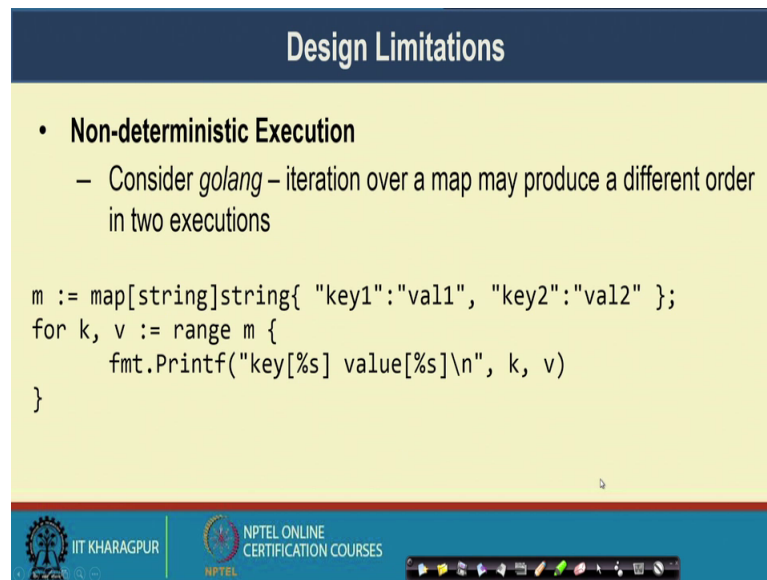
So, with that particular challenge response based method than nodes try to come to the consensus, but as we have seen like the challenge is very hard at the challenge is taking certain time to get a solution and that is the reason that if you want to this kind of serializable order of transactions execution you get a bound on the effective throughput. So, ideal in the throughput is inversely proportional to the commitment latency. So, it is like that if your commitment latency gets increased your throughput will get then decreased.

So, it is like that in a permission less, environment in a bitcoin type of environment if you increase the difficulty of your challenge if or if you increase the complexity of your challenge the effective throughput that you will get that throughput here in terms of number of transactions that can be committed per second per unit time. So, the effective throughput that you will get it will drop in inversely proportional to the commitment latency.

So, this can be a possible attack on the smart contract platform. So, you introduced contract an attacker can introduce contract which will take long time to execute and that is why if certain contract takes the huge time to execute the other contracts will not be able to execute in further, because once the consensus for the previous contract has been reached then only you will be able to execute the contracts which are been submitted latter on.

So, you maintain a kind of serializability order of the transactions which is here preventing you to execute latter contract until the previous contract gets executed. If you introduce a malicious contract in the system which will take a huge amount for execution, you will be able to kind of launch a kind of denial of service attack on your consensus algorithm.

(Refer Slide Time: 25:03)



The slide is titled "Design Limitations" and contains the following content:

- **Non-deterministic Execution**
 - Consider *golang* – iteration over a map may produce a different order in two executions

```
m := map[string]string{ "key1":"val1", "key2":"val2" };
for k, v := range m {
    fmt.Printf("key[%s] value[%s]\n", k, v)
}
```

The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a navigation bar with various icons.

So, that is the kind of first problem, the second problem is like the implementation of your smart contract. Now, as I have mentioned to implement a smart contract you need to go to some language which will give you more power compared to bitcoin script. So, if you remember that bitcoin script is a not during complete language and because it is not during complete it does not support all the contracts which can be there.

For example, if does not support loops or it has certain limitations in execution, but to implement a smart contract, because you want to increase the power of that particular script so, that you can write down any general purpose contract in the form of a code. We use different kind of language for that so, the develops are develop, different types of languages. So, one interesting language is that golang which is widely used in this kind of a contract execution platform.

So, this is the typical example of a contract execution platform in case of this golang you have a construct call the map. So, map is a data structure where you have certain keys and you have certain values. So, every value is associated with one key, now if you run a loop. So, here I have defined the map with the string values the key under corresponding value and this map is stored in a map variable n in golang language and then we are just writing a for loop on top of the smart. So, whenever you have written a for loop over a map and just printing the key value pair and if you run this code multiple times and you

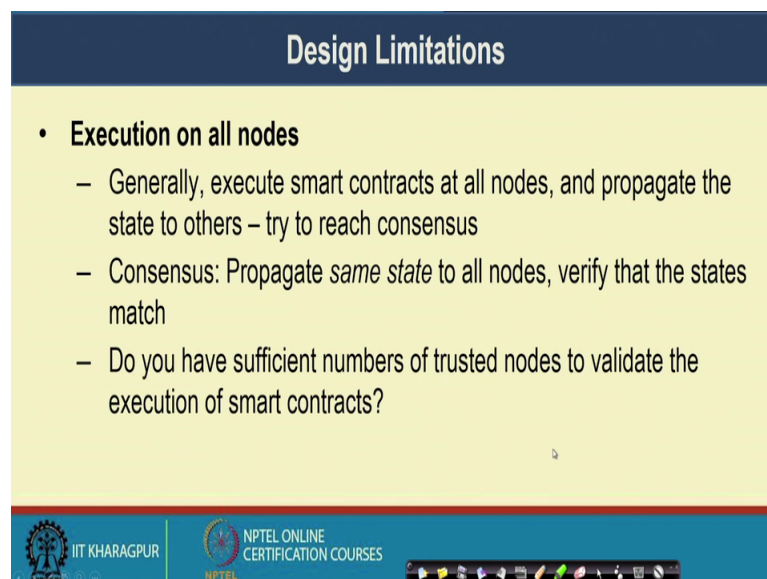
have multiple key value pairs in your map you will see that are different run the ordering of that key value pairs are different.

Now, this gives a non deterministic execution of certain programming construct which are there in your general purpose programming languages. So, here in the golang in the map data structure that it is design in certain way. So, that it is not necessary like every key is ordered in certain way. So, it uses it is internal contracts to fetch that keep certain key under corresponding value which is associated with that key. So, that is why different runs you may have different ordering of the keys.

So, the smart contract execution in contrary it should always needs to be deterministic otherwise it may happen that the system may lead to certain inconsistent state or you may have many forces in the system. So, one particular user it executes the contract get one result or one ordering of result another complex it may get another ordering of result. So, if you are getting two different ordering of results then it may be difficult for you to ensure that that you have the longest chain in the blockchain, although it by the use of the blockchain technology you may be able to ensure that, but you will have multiple unnecessary fork in the system which you want to prevent.

So, the solution here is to go for a specific domain specific language it will implement you are smart contract platform. So, for example, ethereum is a platform where you can implement smart contract with the construct that we have supporting.

(Refer Slide Time: 28:37)



Design Limitations

- **Execution on all nodes**
 - Generally, execute smart contracts at all nodes, and propagate the state to others – try to reach consensus
 - Consensus: Propagate *same state* to all nodes, verify that the states match
 - Do you have sufficient numbers of trusted nodes to validate the execution of smart contracts?

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, that third design limitation is the execution of smart contracts. So, generally we execute does not contract at all node and propagate the state to other. So, that we try to reach in a consensus. So, here the consensus is mean that you execute does contract and based on the input you get certain states and ensure that the states which have been propagated to all the nodes in your blockchain environment they get the same states.

But to ensure consensus the problem that comes that do you have sufficient number of trusted nodes to validate the execution of smart contact, if it happens that will your number of trusted nodes are less than the number of malicious nodes they may get control over the entire environment. But you can prevent that by going to a permission less setting like a proof of work base consensus mechanism, but in a proof of work base consensus mechanism the problem is like you will be you may be able to stuck to a particular like block or a kind of starvation scenario where a contract is taking long time to execute and all the contacts are getting backlock.

So, that is why in case of this permission setting you want to move from that challenge response based method to the traditional distributed system based consensus algorithms, but in that case you have to ensure that you have sufficient number of trusted nodes in the system.

(Refer Slide Time: 30:15)

The slide features a dark blue header with the title "Do We Really Need to Execute Contracts at Each Node" in white. Below the header, a yellow background contains a bulleted point: "• Not necessary always, we just need state synchronization across all the nodes". A network diagram with blue circular nodes and connecting lines is shown, with a red icon and the text "Execute Contract" pointing to one of the nodes. At the bottom, there is a blue footer with logos for "IIT KHARAGPUR" and "NPTEL ONLINE CERTIFICATION COURSES", and a small video inset of a speaker in the bottom right corner.

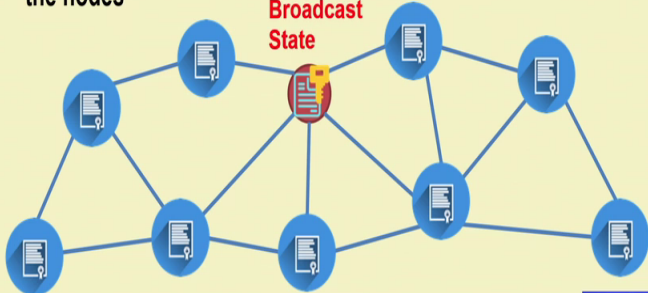
So, one interesting question that comes regarding the consensus of permission blockchain model that do you always need do you really need to execute the contacts are

each node indeed it is not necessary you just need state synchronization across all the nodes. So, it is like that you execute the contract in one node then after executing the contract in one node.

(Refer Slide Time: 30:42)

Do We Really Need to Execute Contracts at Each Node

- Not necessary always, we just need state synchronization across all the nodes



The diagram shows a network of 10 nodes, each represented by a blue circle with a document icon. The nodes are interconnected by blue lines. One node in the center is highlighted with a red and yellow icon and labeled "Broadcast State" in red text. The nodes are arranged in a roughly circular pattern with some internal connections.

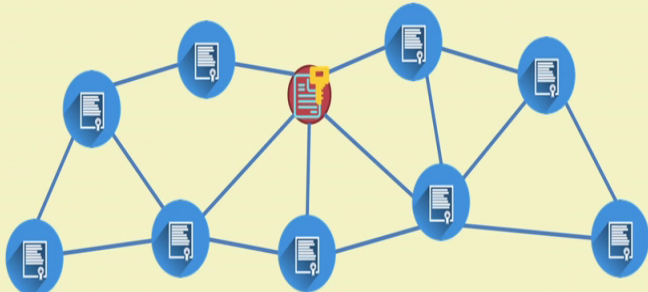
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

You propagate the state of the contract to your neighbouring node and that is states get further propagated. So, that way every nodes in the system it get the same states of the contract and they can be on the same page of your smart contract.

(Refer Slide Time: 31:00)

Do We Really Need to Execute Contracts at Each Node

- What if the node that executes the contract is faulty?



The diagram shows a network of 10 nodes, each represented by a blue circle with a document icon. The nodes are interconnected by blue lines. One node in the center is highlighted with a red and yellow icon, similar to the one in the previous slide, but it is not labeled. The nodes are arranged in a roughly circular pattern with some internal connections.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

But a typical question comes that what is the node that execute the contract it becomes faulty. So, what if this particular nodes become faulty now if this nodes becomes faulty vendor system gets down and the systems will not be able to make any progress further.

(Refer Slide Time: 31:15)

Do We Really Need to Execute Contracts at Each Node

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

The slide features a network diagram with 10 nodes. Three nodes are highlighted with a red 'X' icon, indicating they are faulty. The remaining seven nodes are blue with a document icon, representing active nodes. The nodes are interconnected in a mesh-like structure. At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a presenter.

So, in this particular scenario the idea is that you used the concept of state machine application. So, you execute the contract at a subset of node rather than a single node. So, you have multiple nodes which can be selected to run the particular contracts. So, here this 3 nodes are executing the contract and result of the 3 nodes are propagate to all the nodes in the network.

So, that way you can ensure a kind of should through this state machine replication you can ensure that that every node who are they are the part of the smart contract they are on the same page and they knows that will have to this much of the contract that got executed under remaining part need to be executed.

So, today will stop at this point in the next class will look into this the concept of state machine replication which is a powerful tool to ensure consensus in a permission blockchain environment; so, will come back next, in the next lecture with this concept of state machine replication.

Thank you.