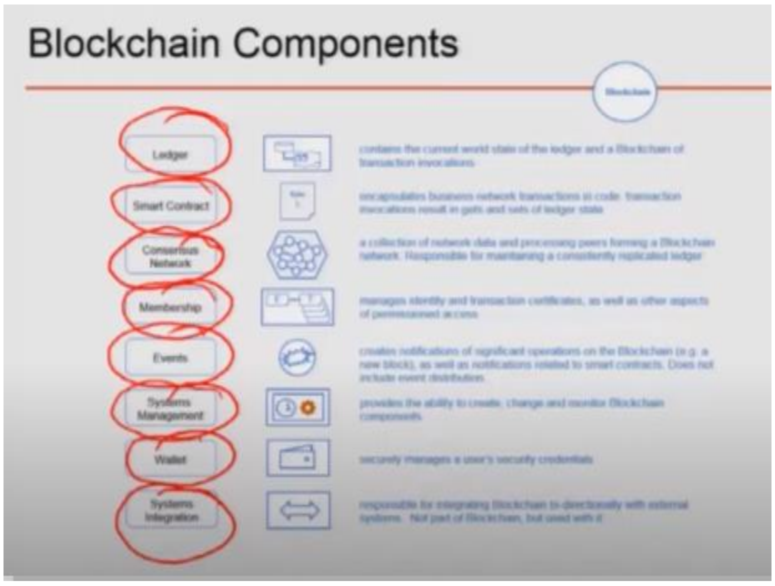
Introduction to Blockchain Technology & Applications

Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

Lecture - 20

Welcome to another session of blockchain technology and applications.

(Refer Slide Time: 00:19)



Blockchain Components

Blockchain

Ledger	contains the current world state of the ledger and a Blockchain of transaction invocations
Smart Contract	encapsulates business network transactions in code. transaction invocations result in gets and sets of ledger state
Consensus Network	a collection of network data and processing peers forming a Blockchain network. Responsible for maintaining a consistently replicated ledger
Membership	manages identity and transaction certificates, as well as other aspects of permissioned access
Events	creates notifications of significant operations on the Blockchain (e.g. a new block), as well as notifications related to smart contracts. Does not include event distribution.
Systems Management	provides the ability to create, change and monitor Blockchain components
Wallet	securely manages a user's security credentials
Systems Integration	responsible for integrating Blockchain bi-directionally with external systems. Not part of Blockchain, but used with it

So last time where we stopped was a very superficial overview of the Hyperledger fabric, which is a permission blockchain, a private blockchain and we were discussing what are the different components. So we talked about ledger, where the transaction history is hash chained for integrity checking. We talked about smart contracts and smart contracts in case of Hyperledger is, is can be written in any language.

And currently, most of the smart contracts for Hyperledger is written in Go language, but the way Hyperledger executes the smart contracts is inside a Docker and they do not use a special EVM for this like Ethereum uses EVM and therefore, it is for a specific language for which bytecode has to be generated. In case of Hyperledger, any general purpose language can be used. But most common is the Go language.

And then of course, there should be a consensus framework. And as we will see that in Hyperledger, the consensus is pluggable in the sense that you can decide what kind of consensus algorithm you need and accordingly you can plug that in into the

Hyperledger framework. There is also something called membership because we are talking about permission blockchain.

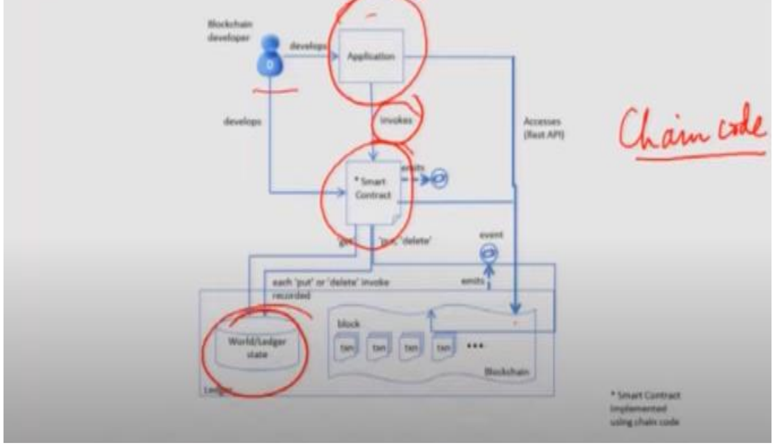
So there has to be a way to prove your own identity. And therefore, there is a membership which is pluggable. So you can have a digital certificate authority connected or you can have your self-signed digital certificate system or you can have a default membership system that obviously is not very secure, this kind of stuff. It also has the notion of events.

So you know like any other smart contract, the smart contracts can actually generate events, which may actually trigger other activities including invocation of other transactions or smart contract functions. Then of course, the system has to be managed. So there are chain codes or there are contracts, smart contract like programs that are called does some system management functions.

So that is the system management code and then wallet in case of Hyperledger is about this, you know the account and the corresponding identity that is given to it by the membership service and then the entire thing has to be integrated, because, you know you can customize your particular implementation of Hyperledger with a different permission membership service with a different consensus service and therefore, you have to do an integration.

(Refer Slide Time: 03:29)

Blockchain Applications and the Ledger

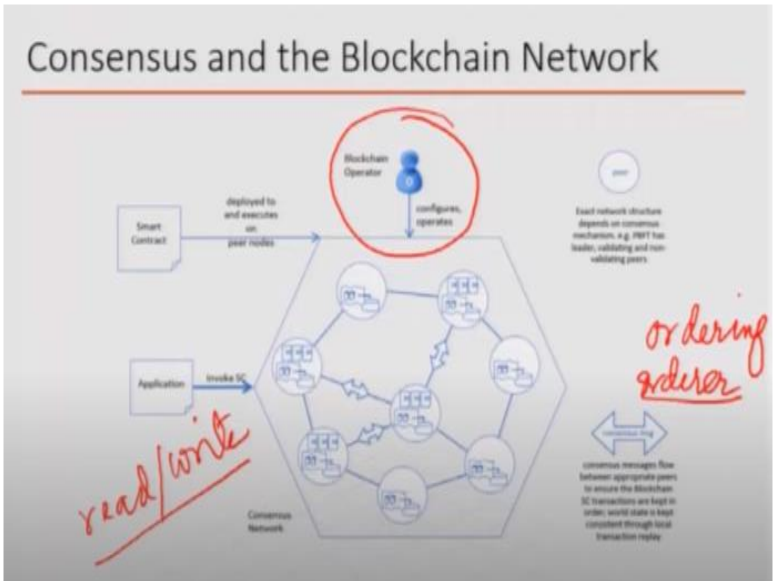


So what happens in this is that, a blockchain developer develops an application which invokes smart contracts. In case of Hyperledger, they are called chain code. So since this is blockchain so the code that runs in the blockchain system is they call it chain code. And then there is the blockchain and the state of the blockchain which is kept in a database.

It will be more clear as we go deeper into these Hyperledger architecture and system description, but the idea here is that the developer develops an application which is capable of doing transaction, but before doing transaction it has to verify or prove its identity and then it can invoke through the transaction request it can invoke smart contract which are chain code.

And then that once that is done, there is you know the transaction has to be persisted for integrative verification in the future for audit function and so on. And then the state of the system has to be also stored because next time you invoke a transaction, it has to be from the right state.

(Refer Slide Time: 04:50)



Consensus and the Blockchain Network
Blockchain Operator
configures, operates
Smart Contract
deployed to and executes on peer nodes
Application
invoke SC
Consensus Network
peer
Exact network structure depends on consensus mechanism, e.g. PBFT has leader, validating and non-validating peers
consensus messages flow between appropriate peers to ensure the blockchain SC transactions are kept in order, world state is kept consistent through local transaction replay
read/write
ordering
orderer

So here is the consensus mechanism. So unlike the Ethereum or bitcoin where the consensus happens first through proof of work or proof of stake or whatever consensus mechanism is inbuilt into the system. And then the consensus what it does is that everybody who is interested in building the consensus, they are called miners.

And what the miners do in such systems in case of bitcoin and Ethereum etc., is that they select the transactions that they want to put into the block, they execute them and make sure that the transactions are valid by looking at the signature etc. And then once they execute them, they actually try to solve the whatever proof of work or proof of stake problem.

And then whoever wins among all this miners, whoever wins will be doing the next block. so the next block will be percolated everywhere. And then every node will accept or try to diverge, but eventually everything will come to a same main chain. In Hyperledger, however, that is not how it is done. So there are certain nodes called ordered nodes. So ordering nodes or ordered nodes, which are actually responsible for the consensus.

The other difference is that the consensus mechanism is all about the ordering the transaction because when you do multiple transactions from different clients, if you do not order them correctly, then if the different nodes order them in their own way, then what would happen is that the result of the set of transactions will be different at different nodes.

So the way this is solved in bitcoin or Ethereum is that eventually only one block wins. So the order that is selected in that particular block is what is you know percolated and persisted everywhere. In this case, this is not how it is done. So what happens is that you order the transactions using this ordering service and this transaction ordering is actually done by solving a consensus.

There are only certain nodes which are involved in that process. And once they do the ordering, they will now create the block, create the hash and then send it to all other nodes. All the other nodes will check whether the transactions that are in the block that they are receiving are valid. So note that here the person who is doing the consensus or ordering is not even checking whether the transactions are valid.

His or her only job is to check is to order the transactions and then agree. If there is multiple nodes which are involved in this ordering process, they have to all agree or have a consensus that this is the order that we are going to persist. And then they will send it to everybody else in a block and the others are responsible for validating that the transactions are valid.

If they are not valid, then we will mark those transactions which are not valid and persist the rest and then they will change the state of the state store according to what those transactions would achieve. So each transactions will change the values of certain states, right. So those values will be changed in everybody's database.

So you see a big difference here, that you are giving the consensus activity, you are giving the responsibility of consensus to a different specific set of nodes, which we call the orders. And they all together form what is called the ordering service. And these orders are not checking the validity of the transaction. They are only checking the, they are only deciding the order of the transactions.

And then they are telling everybody that this is what the order should be. And this is the block of transactions. So you should all check that the transactions are valid, and then you will persist it. And then all the other nodes will do that, add that node, add that block to their blockchain copy, and then change the state and that is how this thing works. So here you see that the application invokes a smart contract.

Every node, many nodes will be executing different set of smart contracts depending on what they are doing. The smart contracts are actually executed inside a Docker. So each smart contract runs inside its own Docker and this Docker makes it possible that the smart contract can be written in any programming language. So it does not have to be Go.

It can be C, C++, Java, whatever language you prefer, you can write the smart contract. Smart contract is basically the application logic. So once this application invokes this smart contracts, then the smart contracts will execute. And then it will realize that if I execute this transaction, then this will be the change in the state. These variables, these state variables will change.

For example, if they are keeping track of money, then how much money is being deducted by a transaction, who is which account gets that money, all that information, they will gather. But they will not persist it because persistence time is later. So they will send back to the application the result of executing those transactions. So in that sense, we say that they simulates the transaction. They really do not execute the transaction.

So they simulate the transaction. They know what the if the transaction happened, what would be the result, they will send the result as what we call a read/write set. So what value is read from the state and what value will be written. For example, if they are keeping track of money, then which account has to be whose balance has to be read, and whose balance has to be written to.

Maybe both the balances have to be written to because you are say you are sending some money from this account to that account. So both accounts have to be read. And both accounts have to be written to. So that read/write set will be sent back to the application. And then the application will then send this information to the ordering service.

The ordering service will now will be getting similar transactions and their read/write sets from multiple different clients and therefore, they will put them together in a block. But before they put them in a block, they will decide what is the right order. And to decide the right order, they will go through the consensus mechanism. And once they have decided the right order, they will create the block, they will compute the block hash and then send it to everybody else.

There is a sequence number that they will also provide to the block. But that is more detailed implementation to avoid confusion, if the same block is sent multiple times etc. So confusion avoidance is done by this sequence number. In any case, the order will then send it to all the others or what we call peers.

And these peers will then check the validity of the transactions and then they will go ahead and only valid transactions, they will actually persist and then they will check the read/write set and then they will change the store. So they will change the blockchain by adding a new block that was given to them by the orders and then they will change their state, database to reflecting the read/write sets that came with these transactions.

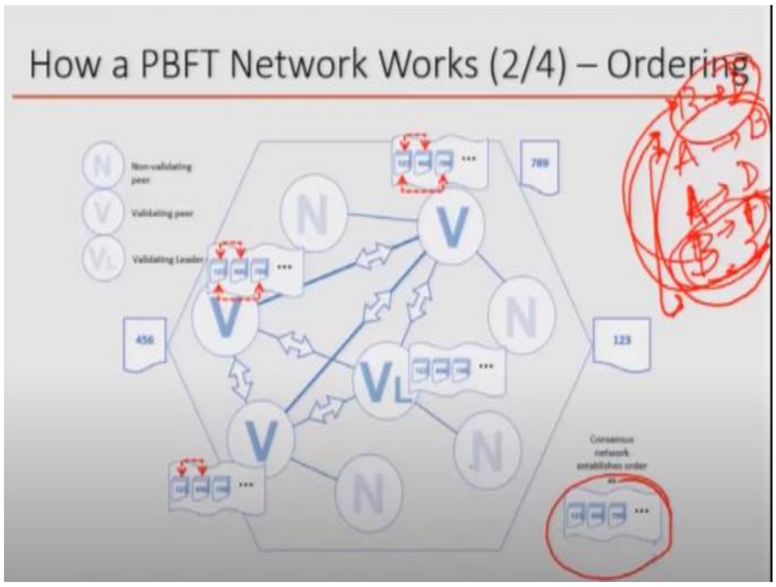
So this is how this works. So blockchain operator actually creates. Why is there a blockchain operator? Because it is a private permissioned blockchain. So let us say I am State Bank of India and I am I am keeping all my transaction reports in this blockchain. So therefore, The State Bank of India's IT department or somebody will set up the blockchain.

They will set up the chain codes, they will set up the applications. And then the clients, that means, you and I, when we actually do a internet transaction will be actually when I fill in that I am going to transfer this amount of money to this other person's account, that information will invoke a suitable smart contract. And that smart contract will give me back the information about what will happen if this transaction executes.

And that information in a read/write set will be given back to me. Back to me means in the background, I will not see it. It will be the code that is running on behalf of the State Bank at the backend of my let us say web based front, web based client side application. Then the back end that is behind my client is going to now collect these and send it to the ordering service. The ordering service will order them.

Then send it back to their all the other nodes. These nodes are there to create replica of the blockchain. And the replica is needed, they could see currently, you know everything is probably put in into a database, which is a single database. Maybe it has a single replica because of the backup and so on. But now there will be many replicas, many peers, many machines on which this whole thing will be persisted. So that is the idea.

(Refer Slide Time: 15:52)


How a PBFT Network Works (2/4) – Ordering
Non-validating peer
Validating peer
Validating Leader
789
456
123
Consensus network establishes order

So if you look at this, so this is basically the ordering. So let us say on one client submits the transaction with a transaction ID of 456. There is another client here at the same time he is doing also another banking transaction and his transaction ID is 789. And another client which is also probably checking balance or doing something else some transaction, that will be let us say 123.

So each of these will now send this information to this ordering service nodes. So these are the ordering service nodes that they have sent to. Now ordering service nodes they will this means here is the ordering messages that the ordering nodes or consensus nodes will exchange with each other in order to achieve a consensus. And at the end, they will achieve a consensus and then that information will be sent to everybody else.

So look here, that there are nodes which are not part of the consensus. They are just there to keep the replica of the blockchain in order to have enough redundancy in the system. So they are not doing the consensus. And that is a major difference with the regular block chains that we have seen earlier that everybody can be part of the consensus mechanism. But here certain nodes are configured to be part of the consensus.

Some nodes are configured not to be part of the consensus, but keeper of the blockchain. So now these things will be ordering these transactions. So the lot of messages will pass and then a consensus will be reached about the actual ordering.

And let us see, eventually they decide that this is the order 123 first although remember, like when we were describing, we said that 123 came slightly later, etc.

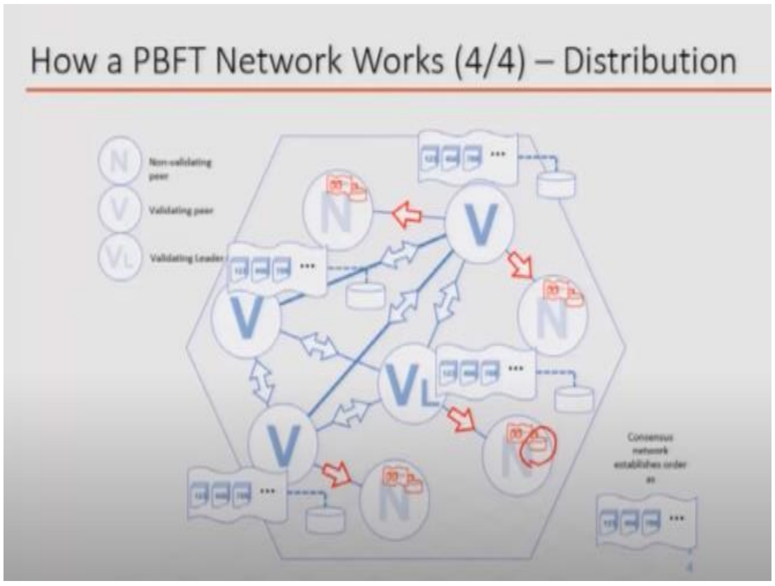
But we are doing this in the correct ordering. Now if 456 let us say this transaction is about two clients A giving money to B and this one is C giving money to D and this one is E giving money to F. Then any order would have been fine because there is no conflict between them. But suppose this one is A giving money to B.

And this one is A giving money to D and this one is B giving money to D. In that case, one has to decide an order. For example, A giving money to B would mean B's balance will go up. If this was put before and let us say B did not have enough balance earlier, then this transaction will not be valid. But if I order it this way this transaction will be can be done. So therefore the ordering will matter if there are such conflicts.

Otherwise the ordering could be any ordering, but the consensus network will decide what the order should be. And if the order they choose still remains the same, then maybe when these guys get the blocks and they will say oh B to D, let us say came first, let us say this is what the orderers decide. Then they will say this transaction is invalid because B cannot give money to D.

So then this client has to again put the transaction back after he gets the money from B. So the so the client or the customer will may actually get a denied transaction and he tries after some time and he will get a fulfilled transaction, because in the first time he tried at that time the ordering service ordered it in a certain way. This can happen.

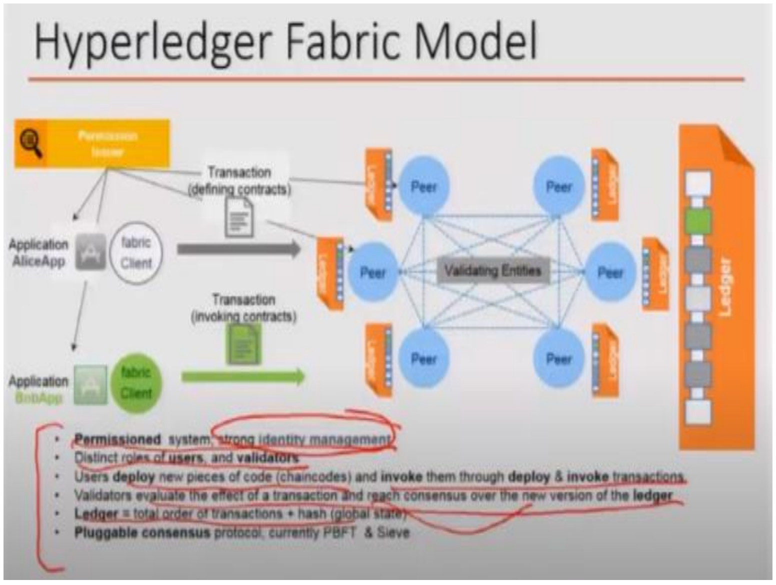
(Refer Slide Time: 19:51)



How a PBFT Network Works (4/4) – Distribution

So then once this ordering has been done for this set of transactions, they will be sent to all the peers. And all the peers will then add the new block to the blockchain and also change the state of the state variables in the databases.

(Refer Slide Time: 20:10)



Hyperledger Fabric Model

- **Permissioned** system, strong identity management
- Distinct roles of **users**, and **validators**
- Users **deploy** new pieces of code (chaincodes) and **invoke** them through **deploy** & **invoke** transactions.
- Validators evaluate the effect of a transaction and reach consensus over the new version of the **ledger**
- **Ledger** = total order of transactions + hash (global state)
- **Pluggable consensus** protocol, currently PBFT & Sieve

So that is the idea. So the basic point here is that Hyperledger is a permission system. So there is a strong identity management, everybody has an identity that they have to get either from some kind of a Digital Certification Authority. Or if they are all part of the same company or something then the company can provide the self-signed certificates or any other mechanism by which a person's real identity will be connected to his identity here.

Again, you see in the permissionless, I can create a pair of public and private keys and I can say this public key hash is my identity. There is no connection to that of my real world identity. Here there will be a strong identity management. So your real world identity would be connected to your identity in the system.

For another difference with these previous ones is that in case of Ethereum, or in case of bitcoin, what we saw is that every member or every participating node or every participating member of the blockchain have the same role. Now some of them might actually say I do not have enough computation power, I will not do mining. And they just do a light, they just maintain a lightweight client, but that is their choice.

So but they could if they have the right if they can try to mine, and they might not ever win because they have only just a laptop, but they can do everything that anybody else can do, except that they may not have the right computation power. Here, the blockchain is configured such that all the users have different roles. Some are just clients. Some are just peers, some are actually orderers or consensus making notes.

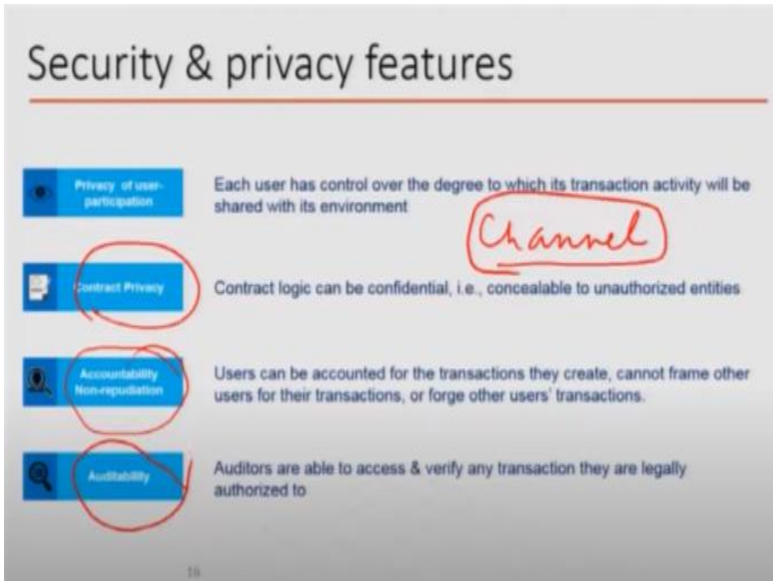
So everybody has their certain roles and this is pre-configured by this blockchain administrator. The users can deploy smart contracts or chain codes and invoke them through deploy and invoke transactions. Validators evaluate the effect of transaction and reach. Well, the validators do not reach the consensus. This is slightly misrepresented in this slide.

The orderers do the consensus and then orderer decided what should go as the next block. And then validators actually do the main job of validating the transaction and persisting the transaction. Ledger will eventually give you a total order of the blocks. So the transactions are ordered inside the block. But that ordering is what is decided by the consensus making nodes. And then the blocks will never diverge right.

So here there is no scope for creating forks right. So even in Ethereum or bitcoin forks can happen, but those forks will eventually you know converge, because after a while the longest surviving chain will be what others will add to. Here there is no such scope, because there is every replica will be a single linear, you know chain of blocks. The consensus is pluggable.

And then there are multiple different consensus. We will talk about that as we discuss more details.

(Refer Slide Time: 23:38)



Security & privacy features

Privacy of user-participation	Each user has control over the degree to which its transaction activity will be shared with its environment
Contract Privacy	Contract logic can be confidential, i.e., concealable to unauthorized entities
Accountability Non-repudiation	Users can be accounted for the transactions they create, cannot frame other users for their transactions, or forge other users' transactions.
Auditability	Auditors are able to access & verify any transaction they are legally authorized to

Channel

Then the question is that since every user's real world identity is connected to the actual identity. So what you might ask is that what about privacy because in blockchain, even though in bitcoin or Ethereum, even though all transactions are in the open ledger, public ledger, at least I can nobody knows who I am. So whoever I am giving money and etc., will be difficult to find out, although it is a pseudo anonymity not full anonymity, but what about in this case?

So in this case, normally, this blockchain is used for businesses right. So therefore, the privacy is not the same as privacy of individuals or in money transaction in bitcoin and blockchain. But even then, a group of users may belong to a particular organization and another group of users may belong to another particular organization and they are doing transactions on the Hyperledger.

Then you may not want that this what this group are transacting and doing should be known to the other group and vice versa, right. So therefore, this Hyperledger has a notion of channels. So channels basically means that in the same set of computing nodes, I can run a Hyperledger, but I can define channels. So certain users will only transact on a particular channel, other set of users will transact on a different set of channel.

These two channels cannot see each other's transaction and they maintain a different blockchain. So let us say State Bank of India and UBI decide that all their interbank transaction etc., will be done through blockchain. So that there is transparency, auditors can see and everything. So what they can do is that they can have a channel in which certain nodes from both organizations are there.

Through which the only the interbank transaction type of transactions will be put in a separate interbank transaction blockchain. On the other hand for their internal customers, their own customers, let us say their transactions are being recorded. So there will be a channel that is SBI will have another channel that UBI will have and they will not see each other.

Only the third channel in which the interbank transactions are being put that will be kept. Now why is this channel why not two, three different Hyperledgers? Because they can share resources. For example, ordering service can be shared by all these three channels. The ordering will never mingle the transactions or blocks of each channel.

It will know which peers belong to which channel and then accordingly send the blocks that it cuts to those peers. So there will be separation and therefore, in that sense there will be institutional privacy. But individual privacy is not a big issue here because everybody is there working on a business and official capacity.

The other thing that the Hyperledger has compared to Ethereum for example, so in Ethereum, all these smart contracts are on every node, is actually on the blockchain. And every node that has a copy of the blockchain has all the smart contracts. And they also execute all the smart contracts, because that is how the ledger works.

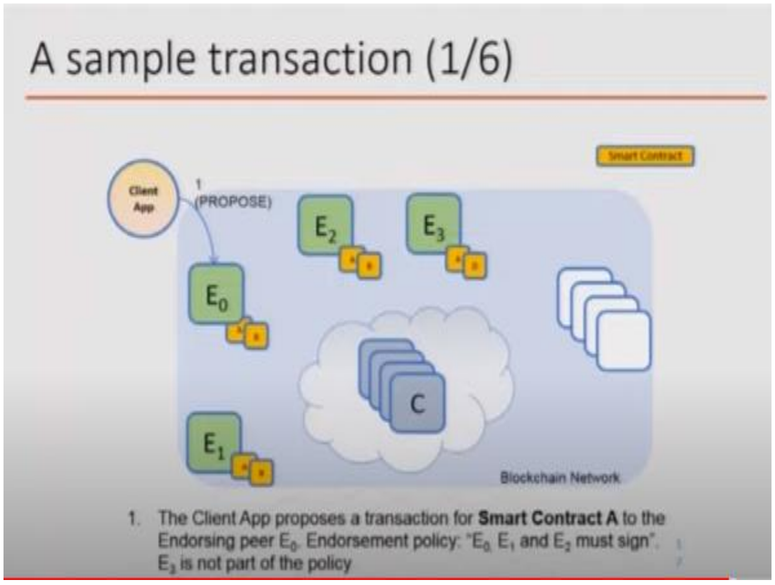
They have to execute and make sure that the transactions are actually valid, and also they have to change the state. What we will discuss shortly is that in Hyperledger, this is basically means that my business logic and your business logic will be known to each other. In a business setup that may not be the good things.

For example, if SBI is having particular logic for deciding who to give loans for example, and UBI has a different set of logic and they do not want their business secrets to mingle. So in that case, this person's smart contract and that person's smart contract may not be, you know visible to each other. It is not desirable to them. So that can be done in this setup in Hyperledger setup, but not through channel.

You can be on the same channel, but you can have separate trust domains, and therefore, only people in your trust domain, the peer senior smarter trust domain can only see your smart contract, others cannot see. So that contract privacy is another thing that Hyperledger gives. The accountability and non-repudiation comes from strong identity, right. So everybody has a strong identity.

So they if they put in a transaction, they cannot say I did not do this transaction. And therefore, there is also auditability. Auditors can access and verify transactions that they are legally authorized.

(Refer Slide Time: 28:56)

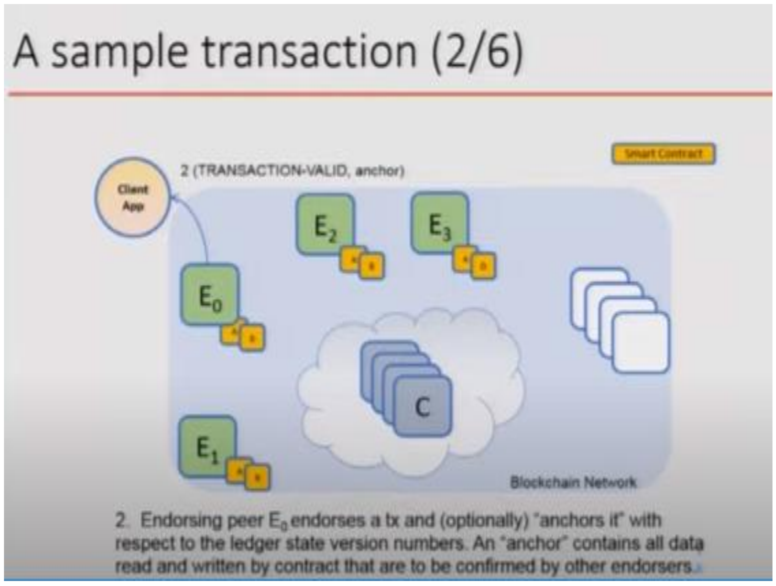


A sample transaction (1/6)

1. The Client App proposes a transaction for **Smart Contract A** to the Endorsing peer E_0. Endorsement policy: "E_0, E_1 and E_2 must sign". E_3 is not part of the policy

So a sample transaction would be like a client application. Like I am a customer to SBI and I am doing a transaction. So I log in through my client application and I propose a transaction. I say give this much money to this person. So then it goes to certain nodes or what we call endorsers. So for each transaction when it is proposed, it has to be endorsed by a set of peers.

(Refer Slide Time: 29:24)



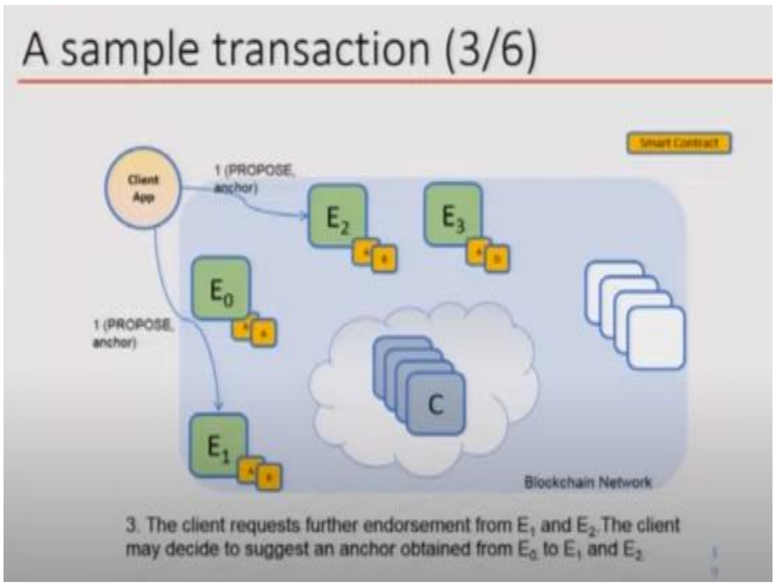
A sample transaction (2/6)

2. Endorsing peer E_0 endorses a tx and (optionally) "anchors it" with respect to the ledger state version numbers. An "anchor" contains all data read and written by contract that are to be confirmed by other endorsers.

So let us say E 0 is my endorser, and I only need one endorser. Let us say this is the endorser. It is a it basically contains chain code, let us say the smart contract A. It also runs other smart contracts because it is a computer. It can run multiple smart contracts but for let us say, for transaction processing A is the smart contract that is required. So what it will do is, it will execute, simulate the execution of the transaction see if you have balance.

And what would be your final balance if you give that money, and what would be the final balance of the other account and it will create a read/write set and send to you.

(Refer Slide Time: 30:02)

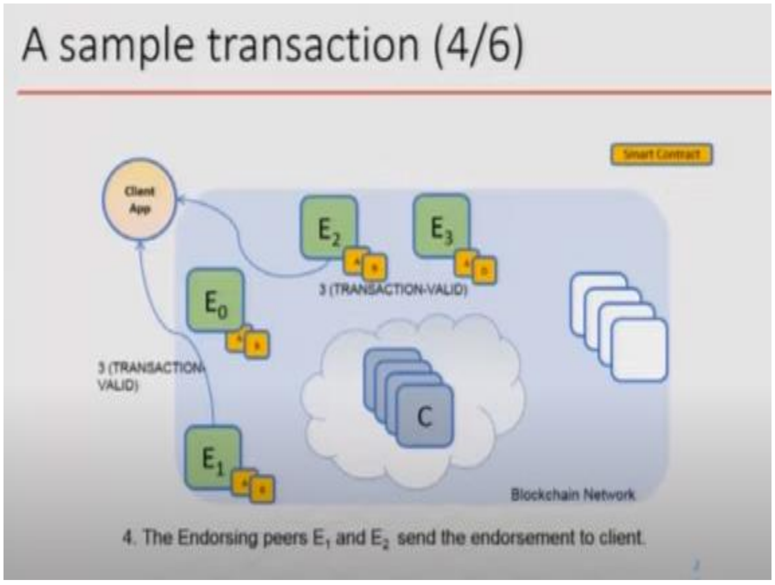


A sample transaction (3/6)

3. The client requests further endorsement from E_1 and E_2. The client may decide to suggest an anchor obtained from E_0 to E_1 and E_2

Then let us say the endorsement policy says that you have to get at least maybe two endorsers to say that what you are doing is allowed. So then it will send to the other nodes which are also in its endorsement policy.

(Refer Slide Time: 30:16)

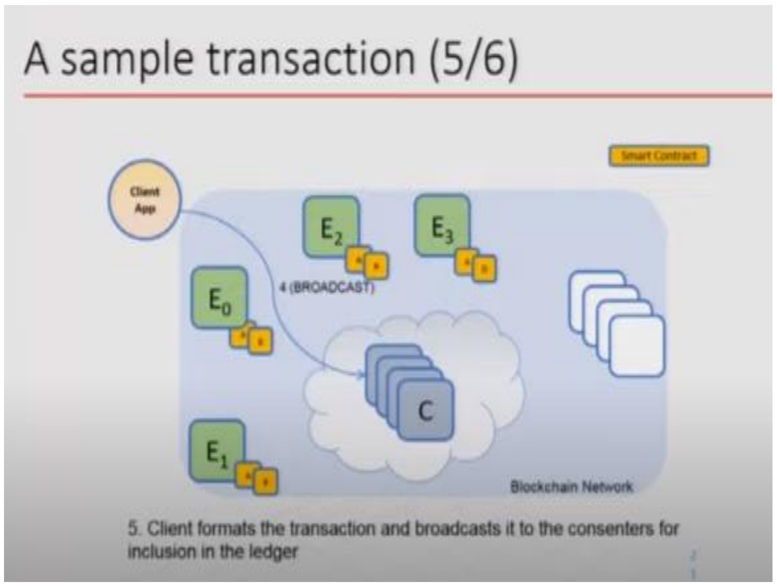


A sample transaction (4/6)

4. The Endorsing peers E_1 and E_2 send the endorsement to client.

And then they will also send you that same kind of information that if you did the transaction, what would be the read/write set what would be the changes in the state.

(Refer Slide Time: 30:27)

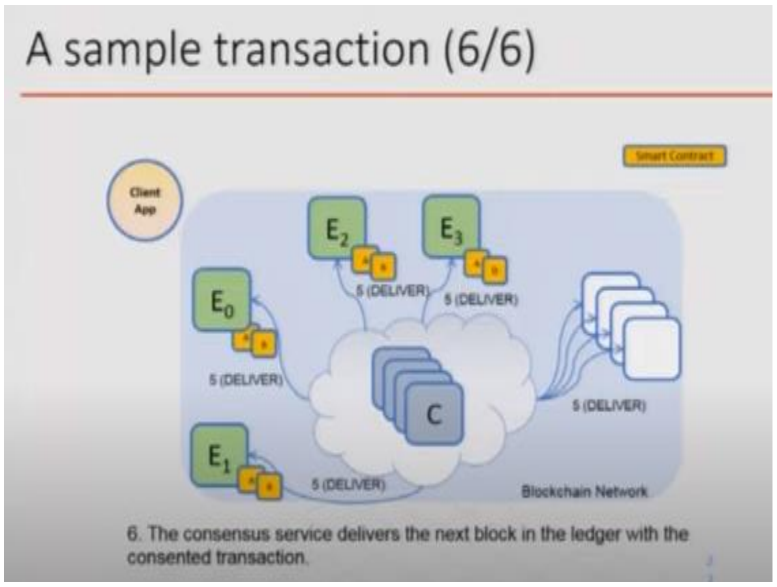


A sample transaction (5/6)

5. Client formats the transaction and broadcasts it to the consenters for inclusion in the ledger

So then it will send it to the blockchain’s ordering service.

(Refer Slide Time: 30:33)



A sample transaction (6/6)

6. The consensus service delivers the next block in the ledger with the consented transaction.

The ordering service will then create the ordering, and then it will deliver the blocks to all the other nodes and other nodes will then including these nodes who are the endorsers, and then they will all persist that blockchain. So this is an overall view of the Hyperledger blockchain. Now when we come back, we will go into a lot more details about how Hyperledger works and the more theoretical underpinning of the design of the Hyperledger blockchain.