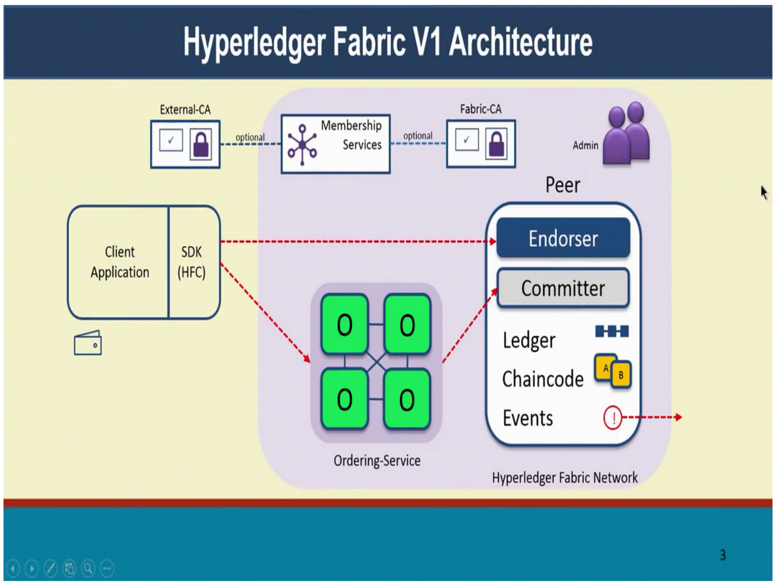
Blockchains Architecture, Design and Use Cases

Prof. Sandip Chakraborty
Prof. Praveen Jayachandran
Computer Science and Engineering
IBM Research, India
Indian Institute of Technology, Kharagpur

Lecture - 21
Hyperledger Fabric - Transaction Flow

Hello everyone. Welcome back to the next lecture on in our Block Chains course Architecture Design and use cases. So, in this lecture we are going to delve into hyper ledger fabric itself and look at the transaction flow that we have right and look at some of the key concepts: how are transaction is going to get committed by all the peers in the block chain network in a consistent manner. So, we need every one to be committing the same transactions in the same order and we going togoing to talk about how we achieve that in a completely descent life's fashion.

(Refer Slide Time: 00:46)



So, let us look at the high level architecture. So what are the key components of the hyper ledger fabric architecture itself? So let us start from the top right. So, there is a membership service, like I mentioned this is going to be providing the notion of identity for the users who are going to be transacting on the block chain.

So, this identity is going to be a digital certificate. And users are going to be using this digital certificate to signed that transactions and submit them to the block chain and the benefit of signing this transaction is two foot: one they authenticate with the block chain that they are a legitimate user and two it also ensures that they get the right access privileges on the block chain for the transactions they are performing. Like I mentioned we do have notions of access control. So certain users are allowed to perform certain transactions and if you do not have the right access then your transaction will get rejected.

So, your certificate is going to have all that information about you what privileges you can have what attributes you have and so on. And we use that to transact on the block chain. Now how do the certificate where do the certificate come from you, it comes from traditional certificate party right. Fabric implements a certificate authority, so this is the fabric CA and this certificate authority is optional.

So, it only it all it does is issued the certificate. So you go to the certificate authority and say I am Praveen Jayachandran I would give me a certificate and it will then issue this certificate saying this public key belongs to Praveen Jayachandran, Praveen has these following attributes and so on right. So, that is the role of the certificate authority. Then the certificate authority is a pluggable module, so it is very much possible for someone else to bring their own certificate authority that issues the certificates.

So, you can bring an external certificate authorities as well and then notion of certificate authorities is nothing new right. If the internet users certificate authority is all the time right whenever you have any secure communication between to a website you are going to use some notion of certificates. So, this certificate authority can be external there are large number of organizations that provide these services and those can be used as well and all of this relies on public key infrastructure.

So, you do have public key and a private key associated with you as a user, the private key needs to be private only to you. So, you are the only one who holds that private key, the public key is something that you advertise to the entire world, and the certificate authority helps you to advertise that. So, to say that you are recognized in this network and this public key belongs to you and this public private key pair is used to authenticate with the block chains.

Now, the client application can be written in absolutely any language of your choice and we provide a set of SDKs to interact with the block chains, so it is called the hyper ledger fabric client. This is the HFC and the SDK today is available in multiple languages it is available in node js it is available in java it is available in python as well. So, you can use any of those SDKs to perform your transactions on block chain.

Now, coming to the block chain it is a what are the various components a important component is of course the peer. So, there can be multiple organizations that are each running one or more peer. So, it is possible that one organization runs multiple peers several reasons for fault tolerance reasons for just separating out different applications it is part of. So, it is possible for one organization to run multiple peers and of course, there are multiple organizations there are run in the network.

So, that is the peer itself and apart from the peer there is an ordering service, so this ordering service can also be run by multiple organizations or one organization you can architect it. However you choose and then the goal of the ordering service or the function it provides is to give you totally ordered set of transactions. So, different peers might be submitting transactions to the ordering service and the ordering service is just gone over take this transaction and put them in some order. And now this ordering service is also a decentralized and the transactions that are coming in from the various peer need to be ordered by this ordering service.

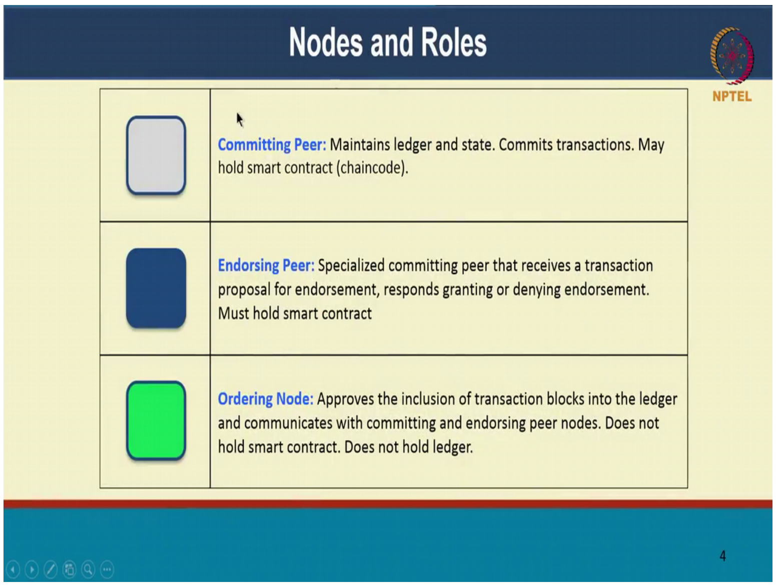
So, again it is possible that some of the peer running organizations also own the ordering service that is possible. So, there might be the once running some of these notes, but just for abstractions purposes we have kept these separate functions that anyone can perform. Within the peer itself there are many notions right. Of course, the peer is going to be maintaining the ledger this is the ledger of all transactions all the state information that is getting stored. It has the smart contracts we call them chain code, so this is the code that is gone a be running on the block chain itself. So, every node in the network will be executing this chain codes.

When I say every node that can be exceptions we will come to that and of course, once the transactions are getting committed on to the block chain, you can emit events those can be integrated with existing system. Now coming to the peer itself the peer can have each peer node can have 1 or 2 functions right. So, it can serve as an endorser what the endorser means is that it is going to execute the chain code. So, it has a copy of the chain code with it, it is go when I be responsible for executing the chain code and signing the output of that chain code.

Whereas I say again sign this peer also has an identity on the block chain, it is going to use it is certificate to sign this transaction saying this is the output of the transaction and I agree with this output right. So, that is the endorser apart from just signing the transaction at that point it is not yet added to the block chain itself, it as to go to the ordering service everyone has to agree that this is transaction that can be committed on to the block chain. And once you ordering service comes back with the full order of transaction then each of this transaction need to be committed on to the block chain.

So, this committing function think is also us can be thought of a separate function. So, it is possible that certain peers are both endorsers and committers, where are certain other peers are only committers. So, I might not execute the transaction as such, but whatever other peers have agreed as the output of the transaction I will only commit. Let us so some of this actually helps in scalability we will come to some of those notions.

(Refer Slide Time: 07:14)



Nodes and Roles

⬜	**Committing Peer:** Maintains ledger and state. Commits transactions. May hold smart contract (chaincode).
🟦	**Endorsing Peer:** Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. Must hold smart contract
🟩	**Ordering Node:** Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. Does not hold smart contract. Does not hold ledger.

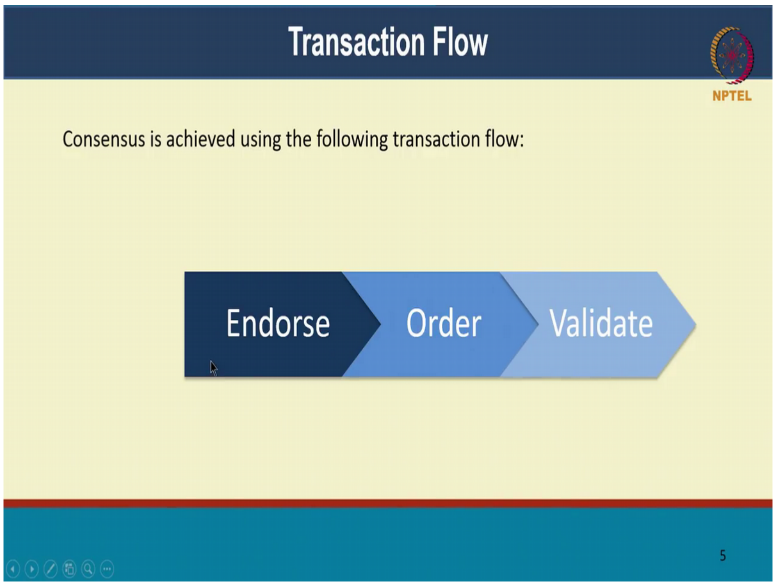
4

So, like I mentioned some of the roles that organization can take on is like I mentioned, all you can be doing is just committing all the transactions that others are performing so that is a committing peer. So, it is maintains the ledger and the state but it does not execute any transactions and it does not hold the smart contracts themselves then next notion is the endorsing peer. So, this is specialize notion so beyond just committing all the transactions you are also responsible for executing the transactions coming up with the output. And then agreeing whether this is legitimate transaction or not. So, that is the notion of endorsement. And obviously, because you are going to execute the transaction you need to hold a copy of the smart contract and you need to be executing that smart contract.

And apart from the peers we have an ordering node which is going to approve the inclusion of transactions into a particular block. So, it is going to order the transaction that may be seen across the networks from multiple peers and it is going to communicate to all the peers what is the order in which the transaction must be committed. So, the ordering service ensures that all the transactions are totally ordered every node in the network is going to commit the same transactions in the same order and that is what will ensure consistency across the entire network.

So, if you think any node if you ask any node any peer in the network what the status, they all have to tell you the same answer, right. It is it should not be that different node tells you different answers in which case consistency you will be lost and the ordering node importantly does not hold the copy of the ledger. It need not even see what the content of the transactions are all it will do is just order the transactions in some manner, so that all node see the same order.

(Refer Slide Time: 09:07)



Transaction Flow

Consensus is achieved using the following transaction flow:

Endorse
Order
Validate

5

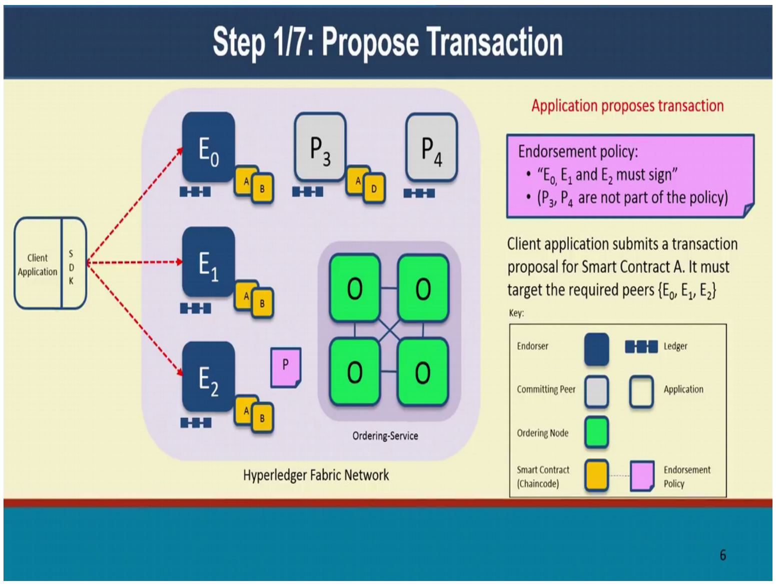
Coming to the transaction flow itself the way in which consensus is achieved is following this order right. So, the client application is going to submit the transaction to a few peers and these peers going to execute this transaction, and agree that the output is the same across all of them. So, there are all going to find the output of the transaction and they will add their signature to it, so that is the part of endorsement. So, the client application has to collect endorsement at endorsements from multiple peers in the networks to say that this transaction is the valid transaction and all the outputs are the same.

So, that is the endorsement part once you collected sufficient signatures, we will talk about what is that sufficient is once if you collected sufficient signatures then you can submit the transactions for ordering. So, this now multiple applications might be submitting multiple users might be submitting these transactions to ordering. And now the ordering service will ensure that all of them are fully ordered right. And totally ordered across all the nodes, so that is part of the ordering service you gone order the transactions. And once you determine the order of all the transactions then the resolution of validation.

So, for instance I should not be performing 2 transactions as simultaneously that modify the same state. So, this is equalent to the double spending problem that you would have seen in bit crime, I should if I have just 10 rupees balance in my bank account I should not be transferring 10 rupees to 2 different people trying to double spend that money. So, that is the validation part and you can think of a generalize motion of that which is for any state right. Let us say I have a state variable which has the value 100 there should not be 2 simultaneous transactions 1 trying to change that 100 to 200 another trying to change that 100 to 50. Now these 2 transactions are going simultaneously only one of them can succeed and which one succeeds will depend on the ordering service.

The first transaction that modify the data element in succeed, every other transactions in that same block there are trying to modify the same transaction will get in validated, so that is the validation step. So, this is the order in which this it is going to execute. So, we will look through these steps in detail in detail know.

(Refer Slide Time: 11:19)



Step 1/7: Propose Transaction

Application proposes transaction

Endorsement policy:
- "E_0, E_1 and E_2 must sign"
- (P_3, P_4 are not part of the policy)

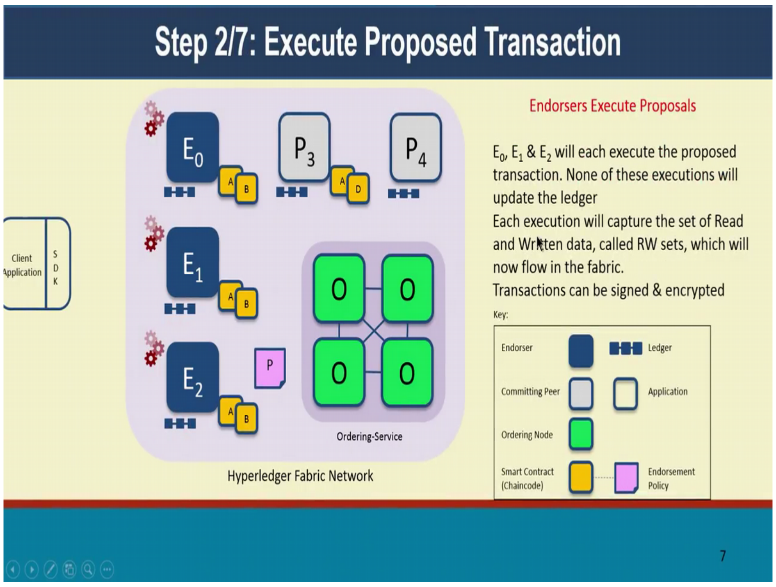
Client application submits a transaction proposal for Smart Contract A. It must target the required peers {E_0, E_1, E_2}

So, the step 1 of the whole transaction flow is what we call proposing at transaction. So, a client application or a user is going to propose a particular transaction. So, it will say I want to I am this user I have this identity on the network, I want to invoke this particular function of smart contract. And here is my transaction here are the inputs to that transactions and this the client application will send to multiple endorsing peer. So, E 0 E 1 and E 2 in this figure are all endorsing peer, and E 0 E 1 and E 2 are endorsing peers they will all execute this transaction. But they are not going to commit into the block chain yet they are all going to execute the transactions and they will say this is the output I see, and here is my signature. So, all these endorsing peers will execute so that is the proposal part.

Now, there is on so how many of these peers should I send to right, the client application and needs to determine that and that is determined by an endorsement policy right. The endorsement policy is something that you can defines for each smart contract in your network right. So, for instance one of the endorse the policy might say I want E 0 E 1 and E 2 to sign for sure or I could say between E 0 E 1 and E 2 if any 2 out of the 3 is sign then I am then this is a valid transaction.

So, there is an endorsement policy that this is pre specified along with the smart contract and that the client application has to satisfy that endorsement policy. It has to it has to communicate with enough peers to gather endorsements for that transaction. Then P 3 and P 4 in this example are committed only nodes. So, we will see what is their function is so over the next few steps and there is a ordering service that I have shown here. So, this is a key here that shows what the difference is and the pink is the endorsement policy and the here the endorsement policy in the example is to say that E 0 E 1 and E 2 must sign this transaction.

(Refer Slide Time: 13:27)



Step 2/7: Execute Proposed Transaction

Endorsers Execute Proposals

E_0, E_1 & E_2 will each execute the proposed transaction. None of these executions will update the ledger

Each execution will capture the set of Read and Written data, called RW sets, which will now flow in the fabric.

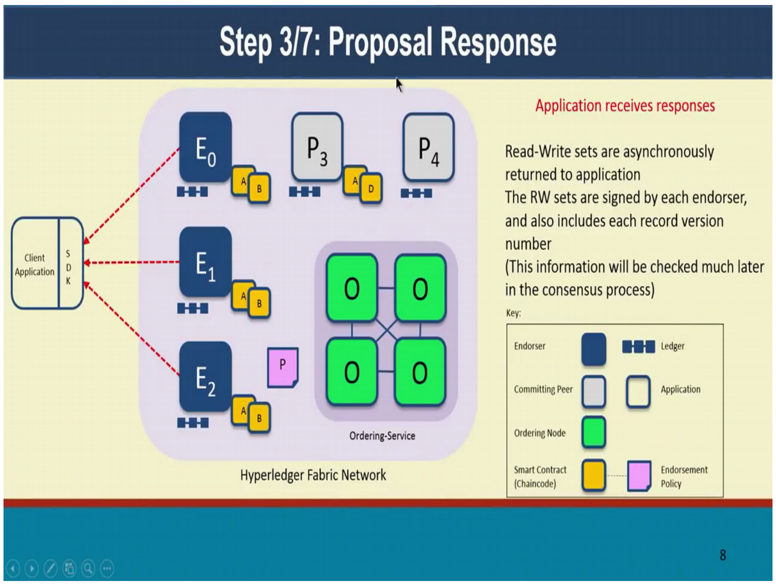
Transactions can be signed & encrypted

So, that is the first step of submitting a transaction or a proposing a transaction. Now the second step is each of these endorses like I mentioned have to execute this transaction. So, the endorses execute this proposal so E 0 E 1 and E 2 are going to execute these transactions and the each execution will capture the read and write set.

So, which data elements where read which data elements where written is captured by every node in the endorser set. And once they capture the read write set it is that read write set that they are the endorsers are going to sign saying. This is what I see from executing the transactions, these 3 elements where written these 4 sorry these 3 elements where read these 4 elements where written.

So, that read write set is collected for that particular transaction and each endorsers will find on it, right. And it can also be encrypted, so you all these communications between parties are encrypted for security purposes right. So, that is now the second step of the flow.

(Refer Slide Time: 14:35)



Step 3/7: Proposal Response

Application receives responses

Read-Write sets are asynchronously returned to application

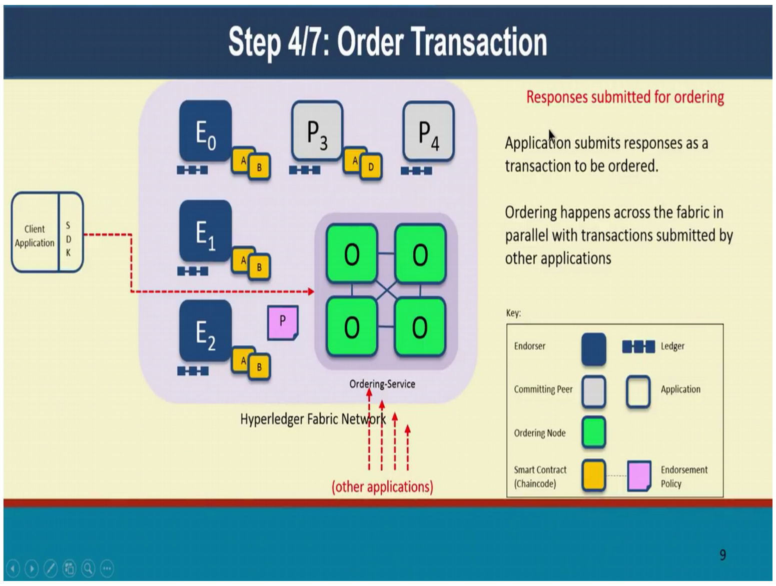
The RW sets are signed by each endorser, and also includes each record version number

(This information will be checked much later in the consensus process)

The third step of the flow is for the endorsers to communicate fact that this is the output they see along with their signature. So, the application or the SDK or the client is now gone receive these read write sets; hence all of these are happening in parallel right. So, you can think of these as a synchronous things it was a it is possible that E 0 and E 1 respond first, whereas E 2 takes little bit more time to come back with the response.

So all of these are happening a synchronously and the information this information about how many endorsements who signed this endorsement whether these were valid endorsement will be checked later in the consensus. So, we will in the flow; we will talk about when these are actually checked. So, this is the third step where the client receives the responses from all the endorsers that it has communicated.

(Refer Slide Time: 15:27)



Step 4/7: Order Transaction

Responses submitted for ordering

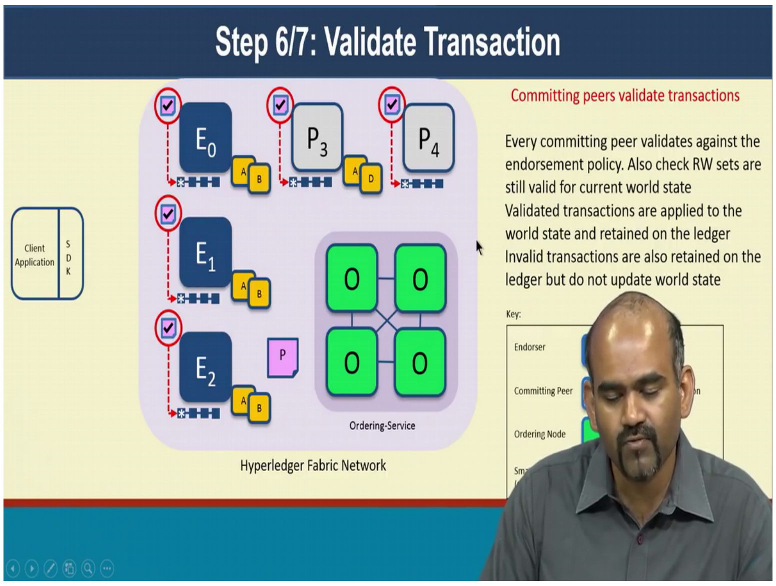
Application submits responses as a transaction to be ordered.

Ordering happens across the fabric in parallel with transactions submitted by other applications

With the next step is ordering. So once the client has received some sufficient endorsements that it thinks satisfies the endorsement policy. In this case it needed signatures from E 0 E 1 E 2 it has got responses from all 3 of them, it is going to then submit the transaction to the ordering service.

Now, such submission to ordering service can be happening from multiple users multiple client applications simultaneously across the network. Now any one node will not know who all submitted transactions who all submitted other transactions at the same time right. It is the ordering service that will determine that. So the ordering service might be getting many such transactions from across the network. So, that is what is shown here other applications are all submitting transactions to the ordering service and the ordering service is when going to determine how to order these transactions. And it will then make sure everyone sees the same order across the network that is the ordering part of it of the equation.

(Refer Slide Time: 16:32)


Step 6/7: Validate Transaction

Committing peers validate transactions

Every committing peer validates against the endorsement policy. Also check RW sets are still valid for current world state

Validated transactions are applied to the world state and retained on the ledger

Invalid transactions are also retained on the ledger but do not update world state

Now once the ordering service is determined an order, it is going to deliver that order set of ordered set of transactions to all the peers right in the network. And this ordered set of transactions is what we are going to call a block. So, each block the ordering service forms a block and communicates that with all the peers saying these are the 100 transactions which I have ordered all of you include this block in your block chain. So, this 100 transactions are now part of the block and they are communicated to all and it is possible that these peers themselves are architected in a way that there is a hierarchy.

So, may be P 4 is part of one organization which has other peers that are also that it also has as committing peers. So, this P 4 can then communicate to other peer then it is network whereas, it only P 4 communicates with the order right. So, it is possible that one peer in the organization takes responsibility for communicating with other peers with the ordering service, but internally all the other peers can then create a copy of the block.

Now how do you see ordering service form this ordered set of transactions, now this is pluggable component in hyper ledger fabric you can use any ordering service. So, there is rich literature of over 30 years where various consensus algorithms have been designed for ordering and for consensus. So, there are a few that are implemented by hyper ledger fabric today one is just a solo order. So, think of it has there is a dictator who is just a single node that is going to dictate what the order should be. And right now that single node implement of FIFO order first in first out. So, whatever I see first this single nodes is first that transactions is going to be ordered first.

So, that is the SOLO order the second ordering service that has been implemented is Kafka ordering service. So, Kafka is very popular event management service by apache it is open source again and internally Kafka implements notion of consensus and that notion of consensus is a crash fault tolerant consensus, you would have seen some of the consensus algorithms and earlier lectures.

So, what you mean by crash fault tolerant is the consensus algorithm or the ordering service can tolerate certain set of these nodes to fail at any point of time and even with certain failures among these ordering service nodes, you can still achieve consistence order across all the nodes right. So, that is the crash fault tolerant algorithm, so typically it will say that as not more than 50 percent of nodes have failed I can achieve consensus actually Kafka. It will ensure that even if one node is surviving out of the entire network we can still ensure that ordering is maintained and here like I mentioned it is a pluggable ordering service.

So, we are working towards a byzantine fault tolerant algorithm as well might come in the future and this is as I said hyper ledger fabric is a community owned project. So, any one in the community even yourselves you can write your own consensus algorithm implemented and contributed back to the (Refer Time: 19:42) community, so that is the ordering service and how it reaches consensus. Then the next step in the process is now all of these peers have received a block of transactions, let say there are 100 transactions in this block now all not all 100 of this transactions may be valid transactions.

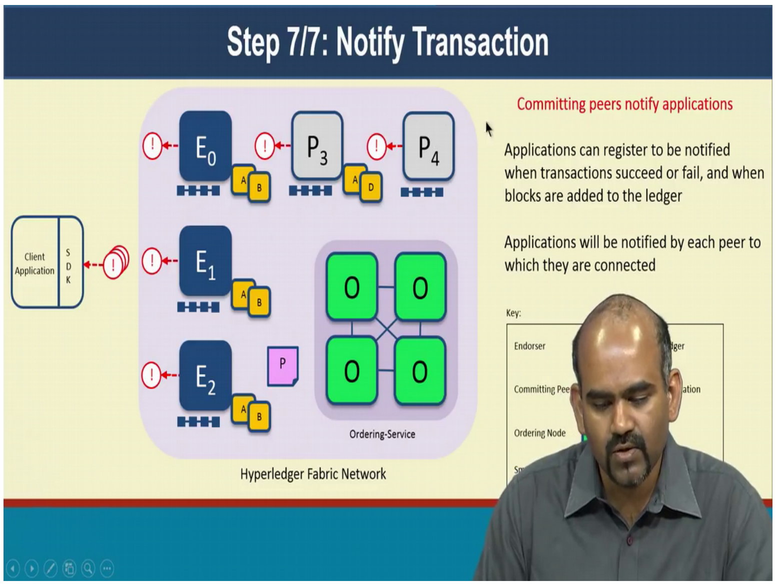
Now what are the some of reasons while transactions might be invalid? Now one reason why it may be invalid is because a transaction has not got sufficient endorsement. So, think of it let say the client application neither by mistake or it was malicious instead of getting endorsement from E 0 E 1 and E 2 it only got endorsement from E 0 and E 1, E 2 did not sign the transaction in that case, now in the validation step when all the peers see that only E 0 and E 1 have signed it E 2 has not signed it, then they will reject this transaction or they will mark that transaction as invalid. So, that is that might be one reason why it is invalidated.

Another reason why it could get invalidated is because 2 transactions tried to update the same data. So, let say there is a variable that had a value 100 transaction 10 in this set of 100 transactions read that variable with value 100 and did certain operations it might have written a some amount of data. It might have written something may be it modified that variable from 100 to a value of 200 so that is if transaction 10. There may be a transaction let say 50 which also read this same element it is, because we are only reading all the committed state it will the transaction 50 will also read the value of 100 it does not have the value of 200 that the transaction 10 wrote, because they happen simultaneously and it tried to modify that 100 to let us say 50.

Now the transaction 50 has read stale data, because transaction 10 has already modified the data. So, that transaction 50 will be marked as invalid because, you transaction 50 read state that was invalidated by transaction 10. So, those are verifications on the read write set is being made is being performed by all the peers in the networks, both the endorsing peers as well as the committing peer will do the validation. And we will now out of the set 100 transactions they will mark these transactions has valid or invalid.

So, in the example that I stated the transaction 50 will be marked as invalid, whereas other transactions might go through. Now only the valid transactions right set will be updated in the world state, all the invalid transactions will have no effect on this state maintained by the block chain itself. So, that is how the overall block chain proceeds in terms if committing transactions. So, that is the validations step saying which transactions are valid which transactions are invalid and that flag of whether it is valid or invalid is included in the block itself.

(Refer Slide Time: 22:38)



Step 7/7: Notify Transaction

Committing peers notify applications

Applications can register to be notified when transactions succeed or fail, and when blocks are added to the ledger

Applications will be notified by each peer to which they are connected

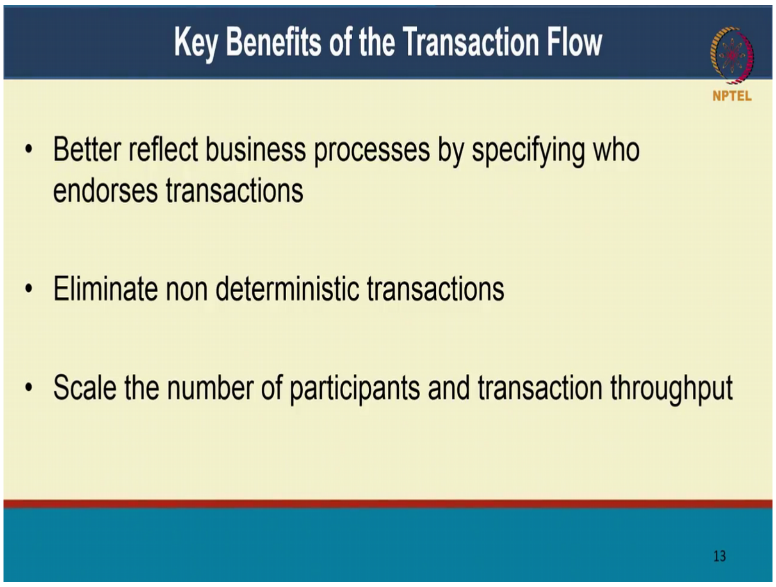
And now the final step is now all the peers are going to commit the set of valid transactions they gone commit the block, they gone add that block into the block chain and each of them will emit events. Now the events itself an hyper ledger fabric you can get multiple kinds of events for the block itself you can get an event, saying this block of transactions was committed.

For each transactions in the block whether it is valid or invalid for both of them you can get specific transaction level events and it is also possible for client applications to subscribe to specific transactions. For instance the client application can only subscribe to the transaction that it has submitted. It might not be interested in getting notifications for the other transactions. So, you can get transaction level events and it is also possible to embed events in the smart contract logic itself.

So, in this smart contract logic I can actually emit an event saying this variable was modified from 100 to 200. And actually give you a business justification or the business reason why it was transferred, may be it might say I got funds from somewhere else and I updated my latest account balance from 100 to 200. So, that business information can be embedded in an event those are called chain code events and those are also part of the block and emitted along with the block events and the transaction events

So, those that is forms the notification part of the whole transaction flow and these notifications are on a per peer basis. So, it is possible that P 4 is emitting events and same event is emitted by P 3 as well. So, the client application can listening on any of the peers that it has access to, may be within it is organization it might it is organization may be running let say 3 peers. It can listen to any of those peers and even if one of those peers fails it can get those revenge from the other peer. So, that is the fault tolerance that that the decentralized system will give you.

(Refer Slide Time: 24:30)



Key Benefits of the Transaction Flow

- Better reflect business processes by specifying who endorses transactions
- Eliminate non deterministic transactions
- Scale the number of participants and transaction throughput

13

So, why it be architected this way right, what is the reason for having the endorsement first then a ordering then a validation right. If you look at many of the existing block chain platforms, they actually have the thing inverted around they first do an ordering and then do the execution of that execution and validation of the transactions. So, it is an order and then execute is the traditional way in which many people have built block chain applications.

Whereas hyper ledger fabric for very specific reasons we took a deviation from that who do the execution first and then ordering and validation and some other reasons are as follows right. The notion of endorsements gives you a way for you to specify who in the system needs to validate a particular transaction. So, I can specifically say these 3 organizations have to except my transactions for it to go through, that can give to you a way to embedding business logic into the endorsement face itself. So, if these organizations do not accept that this is a valid transaction it cannot go through. So, that is a way which you can embedding business logic into endorsement.

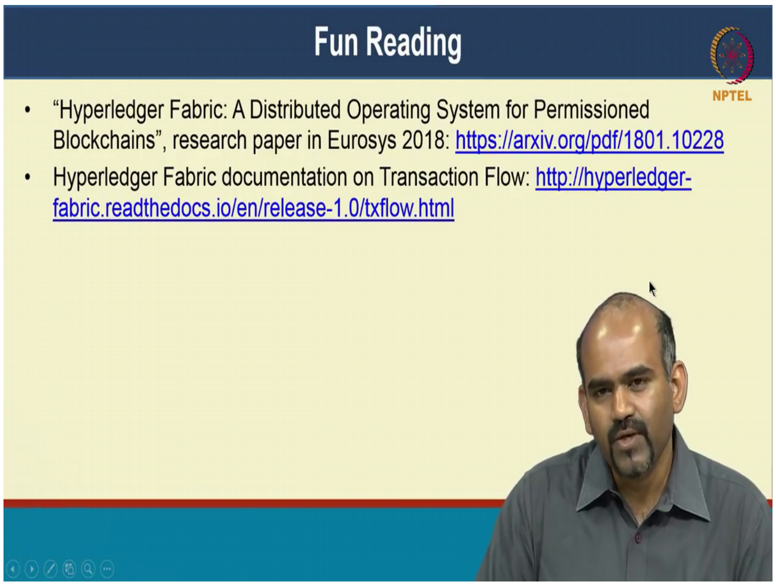
The second reason is to eliminate non deterministic transactions, non deterministic transactions can actually be play awoke in a block chain system what can happen. If it is non deterministic because of that fact that these transactions are executed by all the peers or multiple peers in the network. If they all come up with the different answers then you can it can lead to inconsistent state across the network and we do not want that we want all the nodes to maintain the same consisting state at any point of time.

So, we want to eliminate non deterministic transactions very early in the transaction flow. So, if the 3 endorses or how many our endorses execute the transactions first, they can find out that this is not deterministic 3 of us got 3 different answers. So we not, going to let this into the ordering at all. So, it does not even get to the next phase it is thrown away right there and the other advantage of separating out endorsers who has committers. And also doing the endorsements first is to be able to scale in the number of participants and really scale on the transaction through put.

So, think of it let us let say I have a 20 node network, it is possible that there are 4 sets of 5 nodes it is 20 let say it is divided as 4 sets of 5 nodes, each of these 5 nodes can be executing a different transaction parallely. So, nodes 1 to 5 may be executing 1 transaction, nodes 6 to 10 may be executing one of the transaction and all of these transaction executions can happen in parallel and they can all be submitted to the ordering service in parallel right.

As long as the endorsement policy says I want 5 nodes to accept this transaction. Based on the endorsement policy you can use that to also scale in terms of transaction through put, you can have multiple parallel transactions going through the system simultaneously. So, that was also reason for us to adopt this first execute then order and then validate ok.

(Refer Slide Time: 27:30)



Fun Reading

- "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains", research paper in Eurosys 2018: https://arxiv.org/pdf/1801.10228
- Hyperledger Fabric documentation on Transaction Flow: http://hyperledger-fabric.readthedocs.io/en/release-1.0/txflow.html

So, with that I think we conclude the way hyper ledger fabric executes transactions achieve consensus across distributed set of applications and users, where submitting transactions and how a decentralize set of nodes are committing these transactions; so couple of fund reading exercises for you at the end of this lecture. So, the whole hyper ledger fabric it is architecture the transaction flow along with some experimental results, has been published in research paper recently in EuroSys this year 2018. And the title is hyper ledger fabric distributed operating system for permission block chains, a pre print of that is available on our knife.

So, you can go look at that you can go read that I think it is a very good paper to read to help you motivate some of the requirements from enterprises applications. And why we architected hyper ledger fabric the way we did and extensive documentations for hyper ledger fabric. And specifically on it is transactions flow is available online and it is getting continuously improved with the lot of examples as well online.

So, you can go look at the documentation we will give you detailed explanation of how exactly all of this steps were happening, and I encourage you to encourage you all to go through that. With that concludes this lecture.

Thanks a lot for your time. I will see you at the next lecture.