


Blockchains Architecture, Design and Use Cases
Prof. Praveen Jayachandran
Prof. Sandip Chakraborty
IBM Research
Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 22
Hyperledger Fabric Details

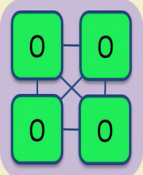
Welcome to the next lecture of our Blockchains course on Architecture, Design and Use Cases. We have been going through some of the details of Hyperledger Fabric and we will be continuing to do that in this lecture. So, we are going to go delve a little deeper into looking at some of the components, some of the principles, concepts and hyperledger fabric; how it is designed and so on. We talked about the ordering service a little bit in the previous lecture, right.

(Refer Slide Time: 00:41)

Ordering Service



The ordering service packages transactions into blocks to be delivered to peers.
Communication with the service is via channels.



Ordering-Service

Different configuration options for the ordering service include:

- **SOLO**
 - Single node for development
- **Kafka** : Crash fault tolerant consensus
 - 3 nodes minimum
 - Odd number of nodes recommended

◀ ▶ 🔍 ↺ ↻ 3

So, the ordering service as I mentioned is responsible for ordering a sequence of transactions into blocks and it goes to deliver this sequence of totally ordered transactions to all the peers in the network.

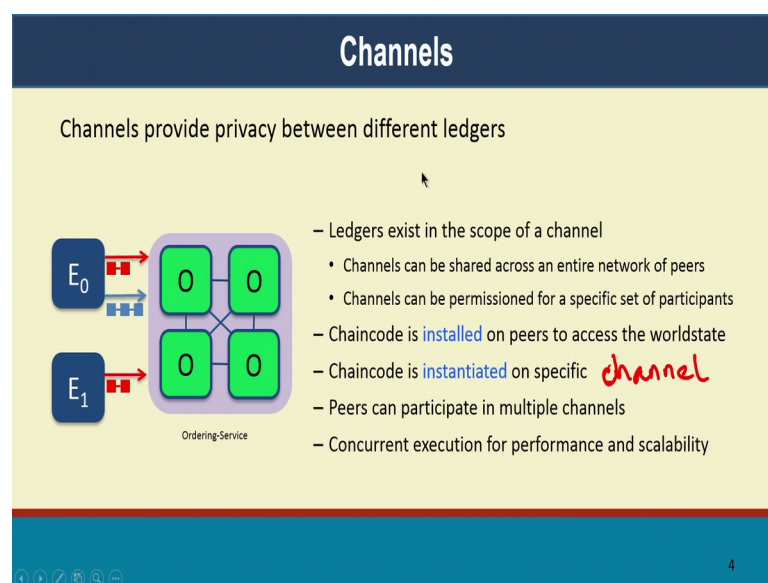
The communication with the service is via channels. So, we will talk about channels in the next slide. So, there are different configuration options possible for the ordering service and the ordering service needs to be set up before you adopt the block

chain itself, right. So, the ordering service and the details of its configuration has to be included in the genesis block of the block chain.

And there are two ordering services that are implemented today in hyperledger fabric; one as I mentioned is a SOLO ordering service, it is a single node, it is meant for development purposes and it is easy to setup and you do not have to worry about some of the distributed consensus life's aspects of the ordering service and this single node will just do a FIFO order. So, first in first out whichever transactions it sees first, it will order that first.

Kafka consensus is a crash fault tolerance consensus like I mentioned in the previous lecture and it needs at least a three nodes at a very minimum and we recommend an odd number of nodes in the network so that we can do a majority consensus. So, that is the ordering service and it needs to be setup before you setup everything else; anything in the network.

(Refer Slide Time: 02:01)



So, the next notion is the notion of channels and the channels what it provides you is the notion of privacy or actually transaction privacy between different ledgers.

So, the same set of peer nodes. So, E 0, E 1 can actually be part of multiple block chain ledgers. So, I can have E 0, E 1, E 2 be part of one ledger or one block chain; E 0, E 1 and E 3 might be part of another block chain, right. So, these are two different chains of

block, they can have their own; each of them can have a different ordering service if they want to; it can have same as well and they are all going to maintain a different scoped ledger. So, each of them; each block chain is going to maintain the same ledger.

In some sense channels in you can think of it as providing multi tendency, you can have different applications with the different set of peers executing on each channel and the notion of ledgers that we talked about exist in the scope of a channel. So, each channel has its own ledger and the data in that ledger is available for the peers in that channel. So, all the peers in the channel can see the data maintained by the ledger of that channel and it is available to all this. So, all the peers in the networks have accessed to the entire ledger.

And it is possible that channels are permissioned for a specific part of participants like I mentioned. So, you can have some sub sets of the network form the channels amongst themselves to executes transactions that are private amongst them. So, you can think of let us say a consortium of 20 banks that come together, there is a one channel that has all the 20 participants amongst themselves. But some sub set may be 2 banks or 5 banks might come together to form a private channel that only they have knowledge of; the other nodes in the network who are not permissioned as part of that channel will not even know the existence of that channel. So, that sort of privacy is possible and the notion of channels also segregates chain code.

So, the chain code is the smart contract and the chain code is installed on the peers that provide access to the world state, right. So, the installation of the chain code; think of it as you are installing a piece of software and that is done on a peer to peer basis, on each peer you will have to go and install the chain code. So, a copy of the chain code is going to be running on each peer that needs to run that chain code.

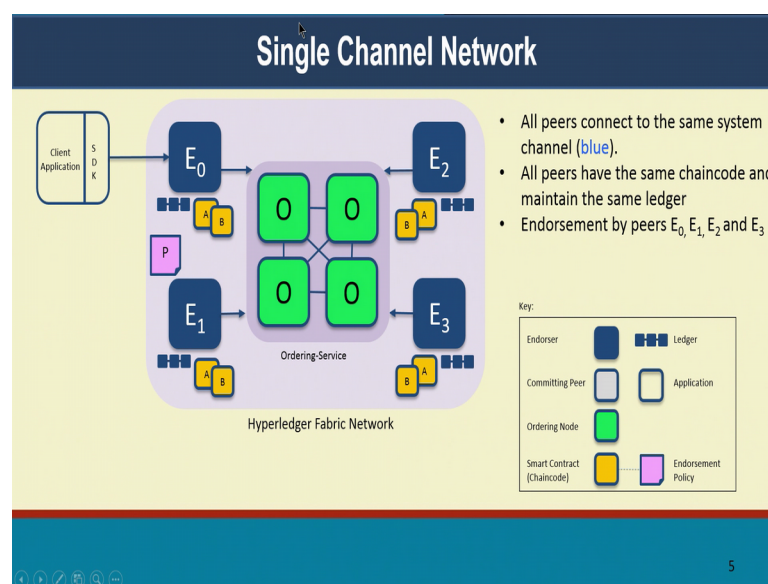
But chain code is instantiated for a specific channel; so, for each channel here. So, what we mean by this is each peer is going to be running a copy of the chain code that it needs to run, but the chain code I can say that only channels 1 and 2. So, in this example here, there are two channels; red and blue, E 0 is part of both of these channels, E 1 is part of only the red channel. So, I can say this chain code is running on E 0 and E 1, for E 0; it needs the chain code on both these channels; on the both the red and the blue channel.

Whereas E 1 needs this chain code only on the red channel because it does not know of the existence of the blue channel. So, chain code has to be instantiated on each channel specifically and the state maintained by the chain code is the part of ledger as I mentioned and it is separate for each of the channel. So, the chain code state in one channel is different from the chain code state in other channel. So, these are kept private and separate and peers can participate in multiple channels. So, it is the same peer process, it is just one peer process that can be participating in multiple channels and this becomes very useful when you are building applications for easy segregation of data and transactions across nodes in the networks itself.

And this also alerts for concurrent execution for performance and scalability. So, it is possible like I said; let us say 20 nodes in a network, 5 of them can form a private channels among themselves where as another set of let us say, 10 nodes can form another private channels among themselves and these two trans; these two channels can execute and completely parallelly with each other.

And that can help improve scalability of the entire network itself. So, they can all executes transactions themselves without interfering with each other. So, that is the notion of channels.

(Refer Slide Time: 06:08)



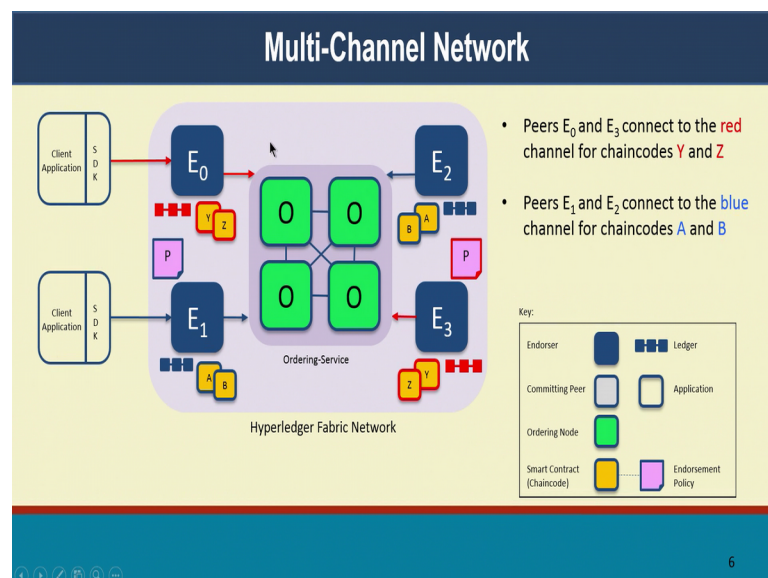
So, we will take a couple of examples. So, here is a network with just a single channel, as before a client application is going to submit a transaction to the network, it will let us

say contacts E 0. So, E 0, E 1, E 2, E 3 are the peers or the endorsing peers in this network, there is an ordering service and there is a exactly one channel in this system and all the peers connects to that channel. So, this is the blue channel here. So, they are all represented by these blue boxes here.

And this channel is going to maintain sequence of blocks or a sequence of transactions and that is what is represented here. Now, A and B are smart contracts. So, these are chain codes that are deployed. So, these chain codes are all installed on every node in the network and they are also instantiated on that channel, right, on the blue channel. So, when our transaction is submitted by the client application to the network, let us say it is invoking a particular function in smart contract A, then all the 4 peers E 0, E 1, E 2, E 3; they are all going to be executing the same function, they all have a copy of that function, they are also going to be executing the same function across all the nodes and then of course, they will go through ordering and the consensus and then final validation.

And the endorsements can also be obtained from all the. So, that is single channel network, it is an example of a single channel network where all the peers are part of that single channel.

(Refer Slide Time: 17:30)



Now, you could also have a multi channel network. Now there could be two different applications one application is transacting on the red channel, where as another application is transacting on the blue channel and these applications may not even know

of the existence of the other channel. If you look at the network itself, there is no way for them to do to even know if they are not permitted, they will not know the existence of the other channel.

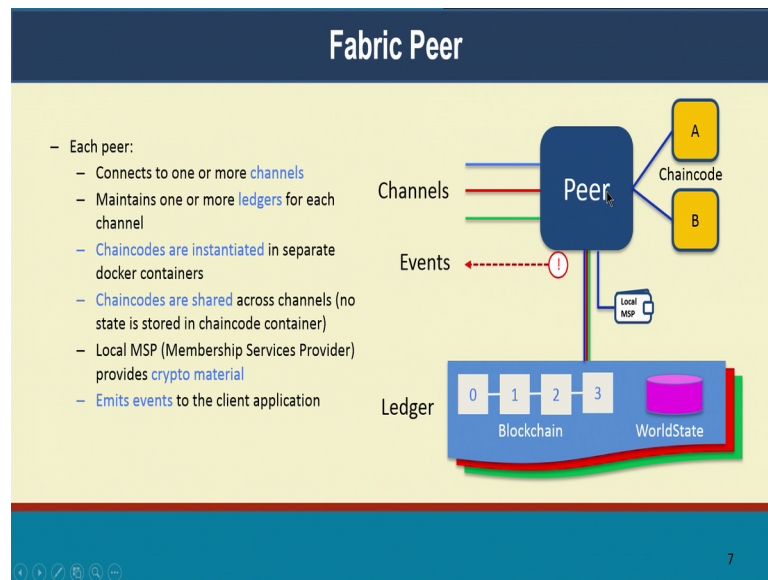
Now, in this example; E 0 and E 3 are only part of the red channel right and they are only aware of smart contracts that are or chain codes that are deployed on that channel. E 1 and E 3 are only part of the blue channel and they do not know of existence of the red. So, this is the two completely disjointed channels in this network and in this example they are actually using the same ordering service.

It is possible that even the ordering service is separate across the channels if you if that is how you wanted, but it is also possible that the same ordering service is ordering for both channels and the and the ordering for the two channels are kept separate. So, the transactions on one channel are only ordered for that channel, the transactions on the other channels are kept separate. Now the endorsement policy on these two channels can also be different or the same its sub to you and (Refer Time: 08:42) per smart contract basis, for the smart contract Y on the red channel, I can define a particular endorsement policy.

For the smart contract A on the blue channel, I can define a different endorsement policy, if I choose to. It is also possible that E 1 for instance is common to both channels. So, you can also have a scenario where E 0, E 1 and E 3 are part of the red channel, E 1 and E 2 are part of the blue channel. So, that is also possible in which case E 1 has knowledge of both the red and the blue channel and the other important thing that I missed mentioning is the fact that even the fact that this particular chain code was has been instantiated on all the node on all the nodes in the channels. So, in the E in the red channel chain codes Y and Z have been instantiated.

So, that instantiation is also a transaction that is recorded on the blockchain; not just the client invoke transaction, but is the instantiation of the chain code also forms a transaction on the block chain. So, there will be a transaction that says that this particular chain code has been instantiated on this channel so that will form a transaction in itself.

(Refer Slide Time: 09:47)



So, just a closer look at what the peer itself does. So, like I mentioned, each peer can connect to one or more channels. These channels are completely disjointed from each other, they are private. So, one channel's information is not mixed with another and so, in this case, there are three different channels; there is a blue, there is a red and there is a green.

So, the three different channels that this particular peer connects to and each channel has its own ledger and its own block chain and they are all kept separate. So, there is a blue ledger, a red ledger and green ledger, they are all kept separate. The actual process for the chain code; let us say that is chain code A and chain code B, these chain codes are processes, they run inside actually docker containers, you can package them as these processes or you can package them as docker containers. So, think of the chain codes as as docker containers and those processes are running, but across all the channels; let us say chain code A and B instantiated all three channels. So, let us say blue, red and green all have a need for chain code A and chain code B, they will be instantiated three times; one in blue, one in red, one in green.

But there will be only one process that is running. So, that is the multi-tenancy that we support, the same process will serve transactions in the blue ledger, serve transactions in the red and the green ok. So, that is just the one copy of the chain code, you could run multiple copies of the chain code if you choose to, but just one copy is sufficient.

And like I said chain codes are initiated separately in docker containers for each channel, but there is just one copy of the chain code running. So, that's that is that what says here; no state is stored in the chain code container itself, it is stored in the ledger that is maintained by the peer. So, there is going to be a communication between the chain code and the peer and communication between the peer and the ledger.

All of these happen through remote procedure calls, we use Google RPC, gRPC and these are all pluggable components. So, the ledger you can plug-in your own data base implementation of the ledger, you can plug that in, the peer implementation can be modified the chain code implementations are pluggable. So, all of these components are pluggable in nature.

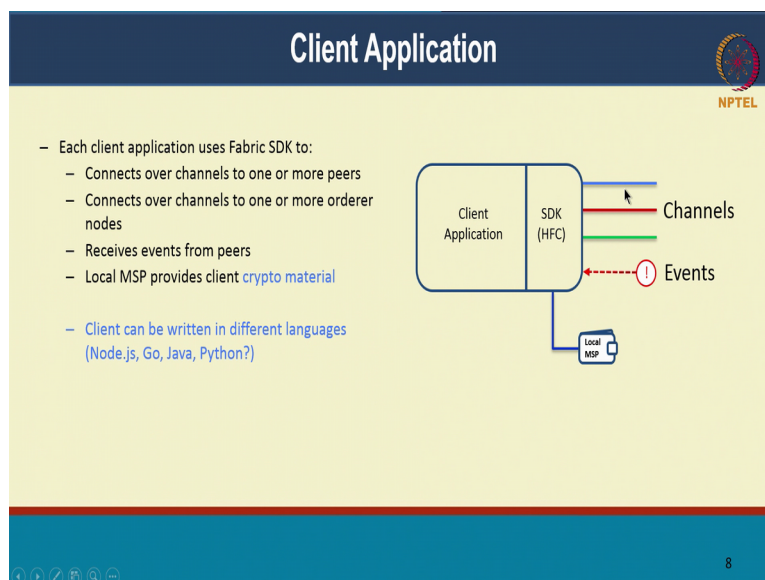
Today the peer is written in Golang, it is it is a new upcoming language for distributed processing. So, all of these components peer ledger are implemented in Golang and they communicate via gRPC. So, any language that gRPC supports can be supported; so, tomorrow you could potentially write the peer in java if you choose to it is possible; the chain codes; there are multiple languages, we support the chain code to be written in. So, we support Golang chain code that is the most popular that we have seen.

We also support Java as well as JavaScript chain code. So, those are also available today for application developers to use. Another important notion of the peer itself, just made pleating reference to it before the peer itself has an identity, right. So, it has to be it has a certificate that issued by recognize certificate authority on the network and peer has a pk infrastructure as a public key and a private key.

And it is going to use those keys when it is communicating with the other peers in the other or with the ordering service or with client applications, event submitted by the peer are all signed by the certificate that the peer has and the membership service provider will have a actually a lecture talking about what the membership provider does. It provides the crypto material that the peer will use to authenticate users transacting on the block chain. There will be a set of notion of organization, notion of a user within an organization. So, the crypto materials for authenticating and validating these users on the block chain will be available to the peer using a local membership service provider that it maintains.

There is also a notion of global membership service provider, but we will talk about that later. So, that is the overall function of what the peer does.

(Refer Slide Time: 13:50)



Now, coming to the client application, I think we talked about this, it is possible for one client application to work with multiple channels just like before. So, these are all kept separate, they have they have to work with the particular peers on those channels.

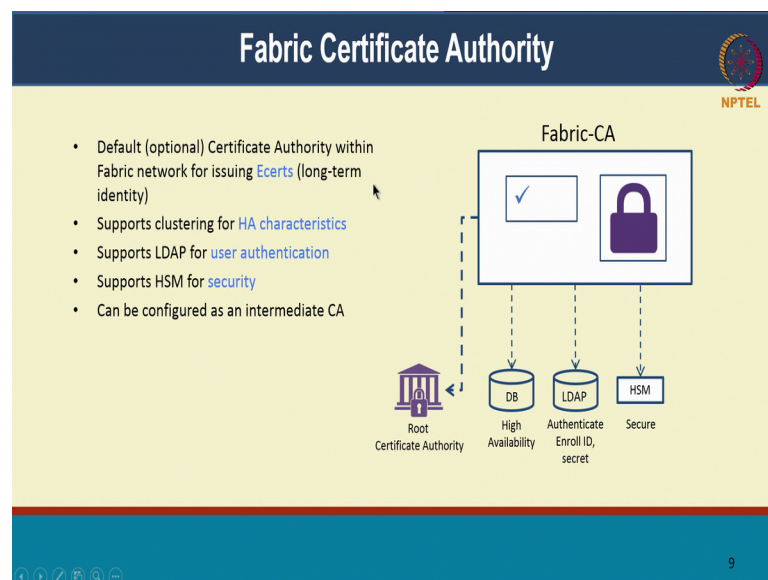
And it can also receive events from multiple channels and the SDK needs to be connected to again local membership service provider that has the crypto materials for the users using that applications, based on the channels the client might also have to communicate with one or more ordering services. So, the blue and red might be using one ordering service, green might be using another ordering service. So, the client application has to be aware of the ordering services and all this information is available in the genesis block like I mentioned.

So, who are the peers in the channel, who are the ordering service; what is the ordering service? All of that information is available to the client application in the genesis block and the peers in the channel can also be dynamic. So, it is possible that one peer leave the channel at some point during the course of execution, may be after the first 100 blocks, one peer leaves and that might be another peer or another organization that joints the network and they can then join the network from block 100, but they can get the history of all the previous blocks that they missed.

So, peers can enter and leave channels dynamically and the fact that the way peers entering and leaving the channel are also recorded as transactions on the block chains. So, they are called channel configuration transactions and they are also recorded on the block chain again in a decentralized manner. So, ever one has to agree that this peer is now to join this channel and they will be admitted into that channel and when a peer is admitted into the channel, of course, the crypto material for that peers has to be updated.

And the client the application itself can be written in any language, but the client that connects with the block chain; there are multiple SDKs that we provide, we provide SDK and Node js, Java and Python and you can use that to connect with the with the blockchain.

(Refer Slide Time: 15:52)



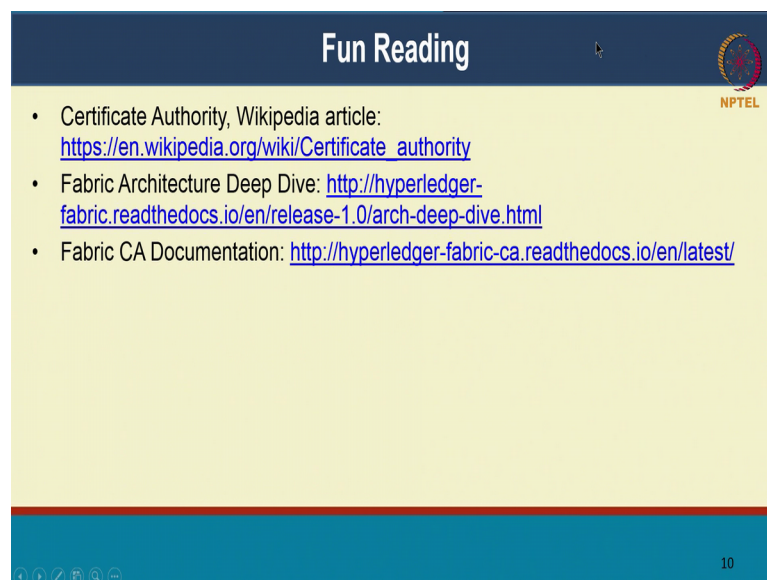
So, little bit on the fabric certificate authority. So, like I mentioned, this is one implementation of certificate authority that issues identities for users who users or components transacting on a blockchain. So, this could be for users, for peers, for ordering service; ordering service nodes, all of them have identities on the blockchain.

And a fabric CA can be used to issue all of those identities. So, there are many ways in which you can implement the certificate authority; one implementation is provided today, there is one more called IDE mixer which is also getting implemented, we might talk about it in advanced lecture later in the course on security and privacy. So, the fabric

CA is attached to root certificate authority and it can have multiple hierarchical level. So, you can have intermediate certificate authorities that link to the root CA and so on.

It has database for high availability and it can also support high availability characteristics. So, even if let us say one part goes down it will still be function out. It can be connected to other identity mechanisms like LDAP and so on and the keys that are issued by the fabric CA has stored securing in a hardware service module, right. Now what do the identity themselves looks like. So, there is notion of an enrollment certificate. The enrollment certificate defines your identity on the block chain and it is called your long term identity. So, we will talk about some of the notions of identity and how identity is used in the block chain itself in one of the subsequent lecture.

(Refer Slide Time: 17:27)



The slide is titled "Fun Reading" and features the NPTEL logo in the top right corner. It contains three bullet points with links to external resources:

- Certificate Authority, Wikipedia article: https://en.wikipedia.org/wiki/Certificate_authority
- Fabric Architecture Deep Dive: <http://hyperledger-fabric.readthedocs.io/en/release-1.0/arch-deep-dive.html>
- Fabric CA Documentation: <http://hyperledger-fabric-ca.readthedocs.io/en/latest/>

The slide has a blue header, a yellow body, and a blue footer with navigation icons and the number 10.

With that we are coming to the end of the lecture itself. So, again some fun reading for you; to gather some of the concepts introduced here; to learn more about some of these concepts. So, if you are not familiar with what are certificate authority is I will encourage you to read the Wikipedia article; it is actually well written article, you can gain a lot of knowledge about how the world works today using certificates. So, that is the good article to read, I will encourage you to read that the architecture of fabric itself detailed note of that is available online. So, there is the architecture deep dive page.

So, you can go read up end detail about the fabric architecture and the fabric CA itself how it architected and some of the functionality it provides, it is also available in the

documentation. So, I would encourage you all to read that as well. So, hope you got a good understanding of some of the components and components of hyper ledger fabric and how it is architected. So, we will look at some additional notions in the subsequent lectures.

Thank you.