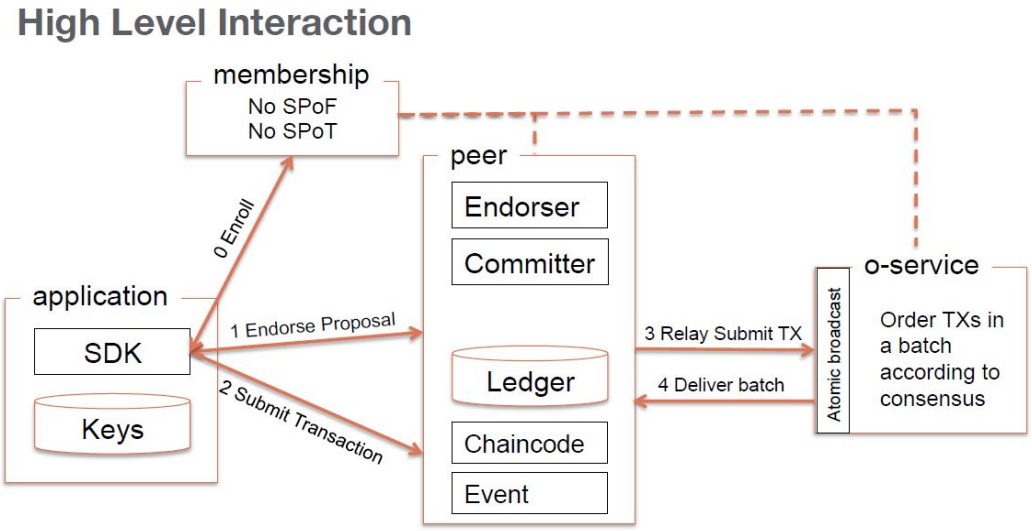
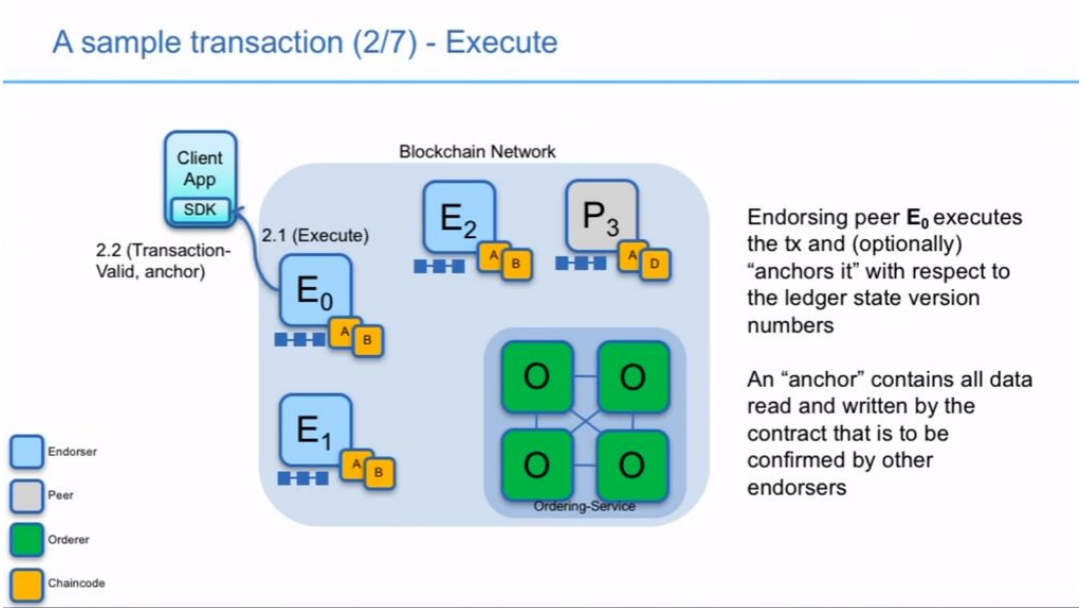
GT Nexus BlockChain Project Report
7/24/2017

BlockChain Terminology





Member

The members on Fabric network. Each member(organization) holds a list of peers and may subscribe to different channels.

Peer

A peer is a node(machine) on Fabric network. A peer can be a committer which sends the transaction proposals or it can be an endorser which endorse the transaction

proposals. After collecting the endorsed proposals and verified the transaction, a peer will send this verified transaction to orderer. A peer receives the blocks sent by orderer and maintain blockchain/ledger.

Orderer

An orderer orders the transactions sent from the peers, aggregating the transactions into one block and sent the block/batch copies back to all the peers by channel definition. We can have multiple orderers run as an ordering service.

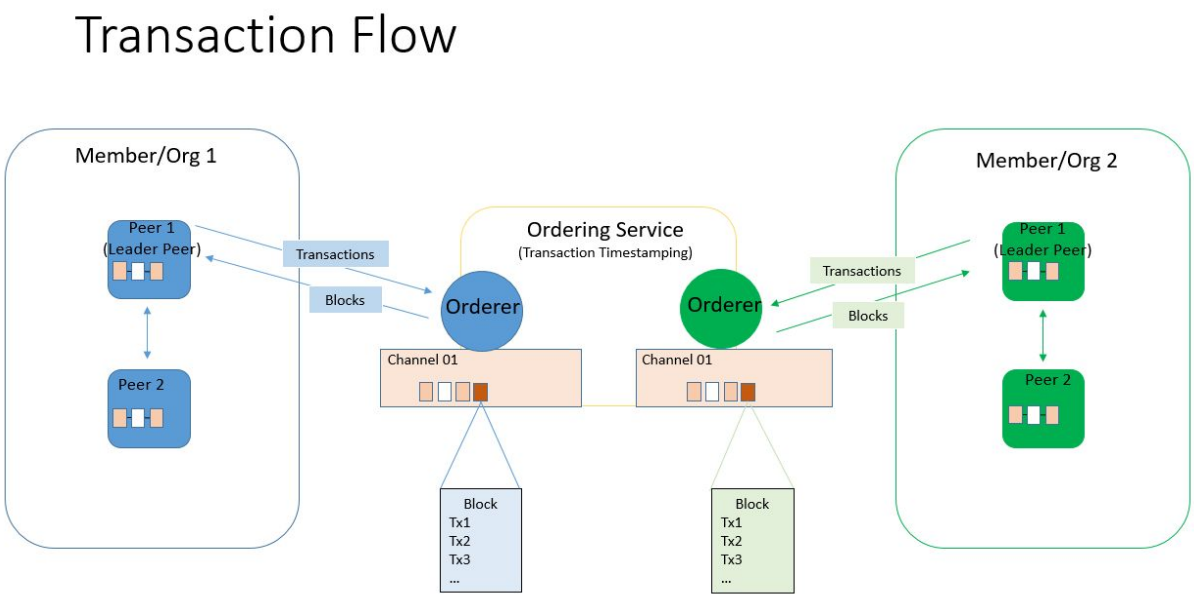
EventHub

Listen to the events on channel

User

An application user can use SDK to send the request to peers. A user may have roles like admin or user.

Channels



A Hyperledger Fabric **channel** is a private “subnet” of communication between two or more specific network members, for the purpose of conducting private and confidential transactions. A channel is defined by members (organizations), anchor peers per member, the shared ledger, chaincode application(s) and the ordering service node(s).

Each transaction on the network is executed on a channel, where each party must be authenticated and authorized to transact on that channel. Each peer that joins a channel, has its own identity given by a membership services provider (MSP), which authenticates each peer to its channel peers and services.

Channel genesis blocks(on orderers) stores the policy of configuration, saying e.g. the majority of members need to approve for updating the channel configuration.

To create a channel, you need to send transaction to a system chaincode. To update the channel, you will send an updated genesis block. You will have the signing party who has a privilege to create channel sign the transaction and send it to orderer.

Transaction

There are system level transactions which call system chaincodes. And there are channel level transactions which call the normal chaincodes. In Hyperledger document, sometimes they have “tx” stand for transaction.
To know more about transaction flow: http://hyperledger-fabric.readthedocs.io/en/latest/txflow.html

Chaincode

Chaincode is a program that implements a prescribed interface. Chaincode runs in a secured Docker container isolated from the endorsing peer process. Chaincode initializes and manages ledger state through transactions submitted by applications. We can install different chaincodes on peers for the channel. A chaincode typically handles business logic agreed to by members of the network, so it may be considered as a “smart contract”.

Each chaincode may be associated with an endorsement and validation system chaincode (ESCC, VSCC)
– ESCC decides how to endorse a proposal (including simulation and app specifics)
– VSCC decides transaction validity (including correctness of endorsements)

System chaincode has the same programming model except that it runs within the peer process rather than in an isolated container like normal chaincode. Therefore, system chaincode is built into the peer executable and doesn’t follow the same lifecycle described above. In particular, install, instantiate and upgrade do not apply to system chaincodes.

System chaincode is used in Hyperledger Fabric to implement a number of system behaviors so that they can be replaced or modified as appropriate by a system integrator.

The current list of system chaincodes:

LSCC Lifecycle system chaincode handles lifecycle requests described above.
CSCC Configuration system chaincode handles channel configuration on the peer side.
QSCC Query system chaincode provides ledger query APIs such as getting blocks and transactions.
ESCC Endorsement system chaincode handles endorsement by signing the transaction proposal response.
VSCC Validation system chaincode handles the transaction validation, including checking endorsement policy and multi-versioning concurrency control.

To know more about chaincode:
http://hyperledger-fabric.readthedocs.io/en/latest/chaincode.html
http://hyperledger-fabric.readthedocs.io/en/latest/chaincode4noah.html?highlight=system%20chaincode

Hyperledger API walkthrough

1.Construct a channel

To create a channel, the sdk take a channelConfiguration file (channel_name.tx) generated by using the tool of configtx. This file contains the predefined channel information such as the memberships(MSPs), Consortium, Readers and Writers.

Configuration is stored as a transaction of type HeaderType_CONFIG in a block with no other transactions. These blocks are referred to as *Configuration Blocks*, the first of which is referred to as the *Genesis Block*.

To subscribe to a channel, a Peer's certificate must be signed by 1 of the members authorized to the channel. The members authorized to the channel are encoded in a configuration transaction on the channel, starting with the genesis block. That is, the genesis block of every channel contains a configuration transaction.

By using SDK API, a Peer can be instructed to subscribe to existing channels. Access control on whom may join a channel is performed by the Orderers with the information given during channel creation or reconfiguration thereafter.

2. Install chaincode on a channel

Create an install chaincode proposal request providing the information of chaincodeID, chaincode file location, and chaincode version. Send the proposals to systemChannel and then the systemChannel will it send to all the peers on the channel. Note that only users with admin role privileges can install and instantiate chain code. MSP determines if someone has the admin role by doing a strict byte comparison with certificates found in the admincerts directory. For chaincode install, this is the local MSP and for chaincode instantiate, it is the channel MSP in the config block.
https://jira.hyperledger.org/browse/FAB-3752

3. Initiate chaincode

Every chaincode program must implement the Chaincode interface whose methods are called in response to received transactions. In particular the Init method is called when a chaincode receives an instantiate or upgrade transaction so that the chaincode may perform any necessary initialization, including initialization of application state. When

initiating chaincodes, a chaincode deployer can specify that endorsements for a chaincode be validated against a specified endorsement policy. So different chaincode can have different endorsement policy by mapping with the chaincodeID.

4. Run the chaincode

A user can send proposals to a channel such as calling the method of invoke in chaincode. After getting the verified proposals, the chaincode on each peer will execute the program by the arguments passed in. For example, if we pass String[] args { "query", key} into the invoke chaincode, it will execute a query for the data with the given key stored on the ledger on the peer. Note that each peer has chaincode installed and each peer maintains it’s ledger under channel consensus. The orderer plays the role of maintaining the consensus of channel.

Working Report

Things we have done

- **Configurations:** Set up configurations for our Network of Network business case. The setting is for 3 orgs on one channel. Please refer to the section of CONFIGURATIONS in the **README.md**

- **Chaincode:** Implemented our chaincode interface in go language. In myCC.go file, we have defined our Init() and Invoke() methods, and also defined the data structure of each trading partner we will put into the ledger. In the generic method of invoke(), we have implemented functions of “add”, “query”, “delete”, and “modify”.

- **Hyperledger API:** Tried to understand java-sdk by reading the sample test of End2EndIT and the codes in java-sdk. We have made our Hyperledger API for developers to develop an application easily. We made generic methods for different stages from constructing a new channel, installing chaincode to run the channel. In the End2EndIT test, it only has one org running on one channel. We have figured out how to bring three orgs on one channel. And we also made the API methods compatible with our customized chaincode(myCC.go).

- **Application UI**: Implemented the UI for the application. We used dropwizard for creating a simple web server that would be able to make calls to our Hyperledger API. The web pages themselves were created using HTML and JQuery. A user may log in from a browser and add/query trading partners on the network.

- **Query Proposal:** We learned that unlike other methods in invoke(), a query proposal doesn’t need endorsement process. A user can send the query proposal to only the peers from the user’s org and obtains the data from ledger. The sdk differentiates the behaviors of query() and other methods like add()/edit()/remove() by the type of proposal it sends. So it doesn't matter that we have query() as one of the function in invoke() in myCC.go. The query proposal skips the endorsement process due to it's proposal type (queryProposalRequest).

- **Transaction Proposal:** We tested a few things on Hyperledger and it turned out that the SDK doesn't help send transaction proposal requests to the right

endorsing peers. It only checks the number of signatures for this tx defined in endorsement policy. That's why we are sending the proposal to all the peers on channel for now. We wonder if there's a method in SDK that help to connect to as many peers as are required by the endorsement policy for the chaincode.

Things under testing/studying

- **Updating channel configuration to bring new orgs onto channel.**

> The standard usage is expected to be:
>
> 1. SDK retrieves latest config
> 2. `configtxlator` produces human readable version of config
> 3. User or application edits the config
> 4. `configtxlator` is used to compute config update representation of changes to the config
> 5. SDK submits signs and submits config

We have tried using the tool of configtxlator to provide a write_set (steps 2 to 4). We found that there is no method in the current SDK that supports retrieving the latest config from a channel. We expect java-sdk will support this feature in the future (see Jira issues below) and will keep following up with it.

After consulting with a Hyperledger-Fabric developer *@jeffgarratt* at Hyperledger developer discussion channel, we were informed that in step 5, the submitter of the TX would have to have current write rights on the channel, and the signers of the TX envelope would have to be based upon the mod_policy of the application group, which commonly is ALL. Meaning a meta policy that represents the signature of the channel config admins from each organization currently in the channel.

Reference of how to use configtxlator:
http://hyperledger-fabric.readthedocs.io/en/latest/configtxlator.html
Jira issue- No means to get channel config bytes for update:
https://jira.hyperledger.org/browse/FAB-5368
Jira issue- Missing channel upgrade method:
https://jira.hyperledger.org/browse/FAB-5408

- **Revising the endorsement process**
- **Setting up the Fabric network with peers on different machines**

Questions to be followed up

- How can we do a more complicated endorsement process with our business case requirements?
    - We have created an endorsement system through chaincode, where after every query, the payload would invoke another chaincode, which could be specified per organization. This second chaincode would look at the data and determine if this data should be approved or rejected.
    - However this system would still be inside the network. Ideally, we'd like it to be endorsed by an outside party, maybe even a human user

- We wonder if there's a method in sdk-java that helps to connect to as many peers as are required by the endorsement policy for the chaincode? (So we don't have to always send the transaction proposal request to all the peers on channel)

Resources

Hyperledger Developer Discussion channel:
https://chat.hyperledger.org/channel/fabric

Hyperledger fabric-adk-java:
https://github.com/hyperledger/fabric-sdk-java

Hyperledger Fabric online document:
http://hyperledger-fabric.readthedocs.io/en/latest/getting_started.html