Whitepaper of DP-Chain

Mingrui Cao

2022.06.08

Abstract

This paper is to describe the methods and implementation of Distributed and Parallel Blockchain (DP-Chain) system.

Contents

1 Introduction

Blockchain is defined as an immutable ledger recording transactions in a distributed network formed by mutually untrusting nodes. Every node has a copy of the ledger locally. Consensus mechanism runs on every node to validate transactions, wrap transactions into block and append the block to the leader (or say the chain). According to the consensus mechanism, transactions are stored in the ledger in order, and the consistency of the distributed ledger is achieved.

Bitcoin [1] brings out the blockchain technology which is widely regarded as a promising way to make trusted exchanges in a distributed system. As bitcoin first introduces a public blockchain that anyone can participate without a specific identity, public blockchains typically involve a native cryptocurrency and often use consensus based on “proof of work” (PoW) and economic incentives.

However, in some scenarios where participants are known and identified, blockchain could be reconstructed as a consortium one. A consortium blockchain provides a way to secure the interactions among a group of entities that have a common goal but which do not fully trust each other, such as businesses that exchange funds, goods, or information. By relying on the identities of the nodes, a consortium blockchain can use traditional Byzantine-fault tolerant (BFT) consensus.

The consensus protocols used in blockchain platforms vary greatly these days. Most enterprises only want to construct consortium blockchain platforms, thus, abandoned the inefficient and wasteful PoW in favor of some variant of a quorum-based consensus protocol. These protocols reach a decision on a sequence of values which in the blockchain case is the identity and order of transactions when a quorum Q of nodes out of a predetermined assembly N are in agreement.

At the very beginning, we must state that DP-Chain should be a consortium blockchain system whose competitors are Hyperledger Fabric, Fisco Bcos, and etc. Therefore, we assume that all nodes in DP-Chain have been identified and verified when accessing the network. All discussions about security and stability of DP-Chain later are based on this fundamental assumption.

DP-Chain is a information shared platform, in which all information is recorded on a distributed ledger, with authority and cannot be tampered.

One of the biggest scope of designing a blockchain system is to maintain the consistency of the distributed ledger. For public chains like Bitcoin and Ethereum

2 Architecture

The architecture of DP-Chain is shown as Fig.1.

3 Methods and Theories

3.1 Unequal Nodes

In a public chain, people assume that every node maintains the blockchain is motivated by some potential reward. Thus, every node has the right to be a miner as it has the ability. This makes the inherent concept of blockchain that every node in blockchain should seen in equal. However, it is not the case in a consortium blockchain. To accelerate the system

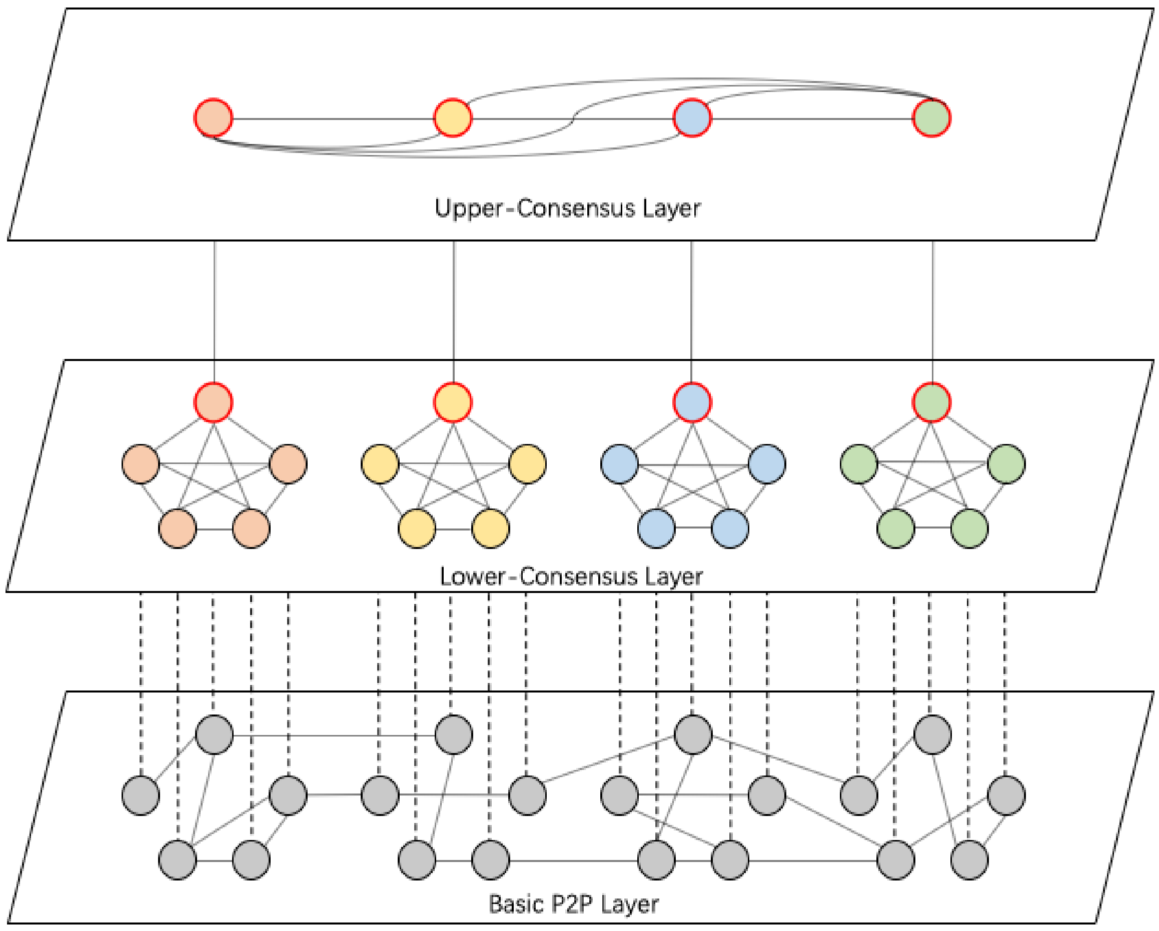

Figure 1: The architecture of DP-Chain.

efficiency, some kinds of nodes could be designed to maintain the blockchain without any reasons. In DP-Chain, one important kind of nodes is boot node. Boot nodes help other nodes to enter DP-Chain and form the P2P network, and also take on the work of configuring DP-Chain and coordinating the consensus. Though boot nodes play an important role at the beginning of DP-Chain, they could not be seen as the administer of DP-Chain. This is because boot nodes only provide service and have no privileges.

3.2 Transactions and Smart Contract

3.3 Trimmed PBFT Consensus

In the lower-consensus layer, every subnet runs its own trimmed PBFT consensus. The traditional PBFT of blockchain systems contain 3 stages to achieve consensus which are pre-prepare, prepare, and commit stages. The trimmed PBFT of DP-Chain also contains these three stages, but have some differences.

The workflow of the trimmed PBFT is elaborated in Fig.2. Let the leader nodes of subnet A be denoted as SNL-A, and the three follower nodes are denoted as SNF-1, SNF-2, and SNF-3 respectively.

There is a trigger on SNL-A. Every time a certain interval passes or enough transactions are collected in the transaction pool of SNL-A, the trigger launches a consensus request. Then, SNL-A would order transactions in the transaction pool. After that, SNL-A

Table 1: The data structure of the transaction.

Transaction		
Head	Sender	Account Address
	Nonce	Transaction Index
	Version	Current Block Hash
	Lifetime	Max Lifetime
	Signature	User Signature
Body	Contract	Contract Address
	Function	Function Name
	Args	Function Arguments
Tail	Checklist	Checklist for Validation

broadcasts a pre-prepare messages to other nodes in the subnets. The pre-prepare message contains the order of transactions in this round of consensus as shown in Tab.2. Once followers (SNF-1, SNF-2, and SNF-3) receive the pre-prepare message, they'll check out whether the ordered transactions have been collected in their transaction pool. If transactions are right there, they broadcast a prepare message of their each own.

The data structure of prepare message is shown in Tab.3. When a follower receives enough prepare messages, it begins to validate the transactions in order, and form a block containing the tidied order of transactions which are valid. Notice that the validation is very simple, because transactions execution is not done at this time. If every node runs properly, the block after tiding on each is the same.

Finally, every node signs its tidied block and send back a commit message as shown in Tab.4 to the leader SNL-A. In the traditional PBFT, commit messages are broadcast in the network, and when enough commit messages are received, the block is regarded as valid. This part of PBFT is trimmed in the lower-consensus layer of DP-Chain. The leader SNL-A will combine the block signatures in commit messages and represent the subnet to make the block in consensus during the upper-consensus layer.

Table 2: The data structure of the pre-prepare message.

Pre-Prepare Message		
Common	Consensus	DP-Chain 1.0
	Type	101/denotes pre-prerare
Message	Version	Current block hash
	Nonce	Consensus round index
	Leader	Node Address
	Order	Transactions order
Summary	Round ID	Hash(Common+Message)

It should be noted that signatures are no need during pre-prepare and prepare stages, because every node in the subnet is connected with each other directly with security.

In addition, the transactions tidying should be customized to enable the special requirement of some contracts. This is an essential part of implementing in some scenarios where user is node.

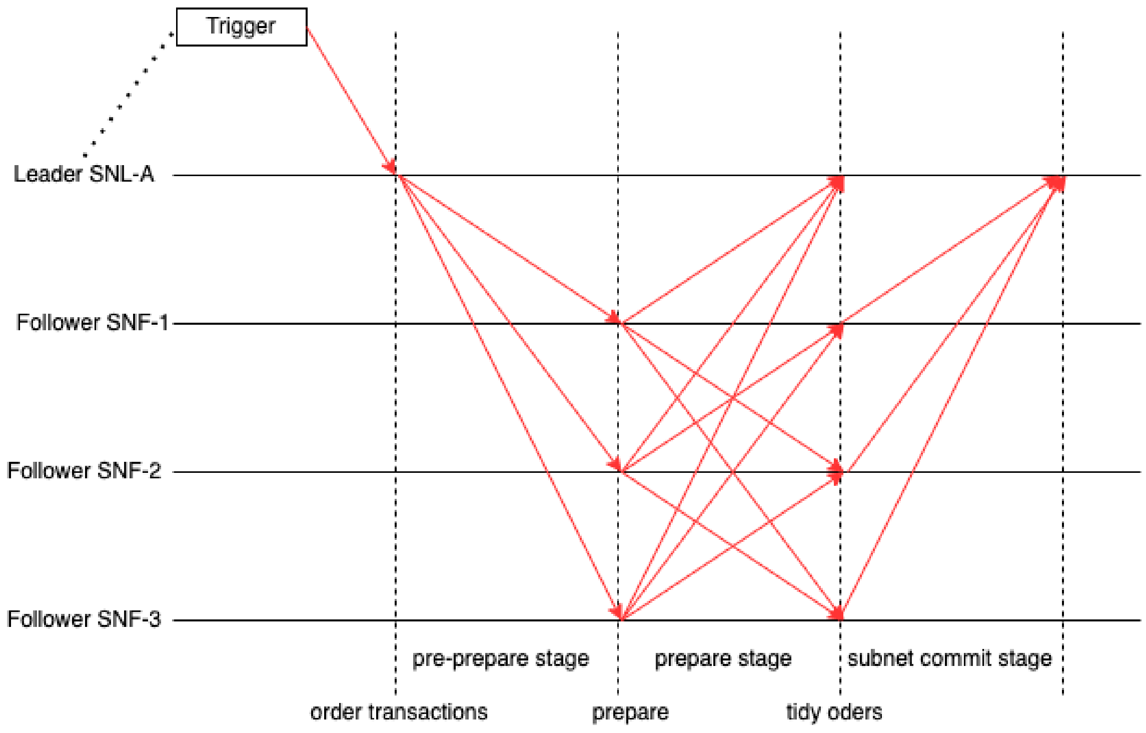

Figure 2: Three stages of the trimed PBFT.

Table 3: The data structure of the prepare message.

Prepare Message		
Common	Consensus	DP-Chain 1.0
	Type	102/denotes prepare
Message	Round ID	Hash of pre-prepare message
	Follower	Node Address

Table 4: The data structure of the commit message.

Commit Message		
Common	Consensus	DP-Chain 1.0
	Type	103/denotes commit
Message	Follower	Node address
	Round ID	Hash of pre-prepare message
	Result	Results after tidying
Vote	Signature	Sign and vote the block

3.4 Modularized Upper-Level Consensus

The upper-level consensus of DP-Chain is modularized, but should be with strong consistency to make every block come from different subnets in same order on blockchain. In this paper, we apply the boot node to provide block ordering service to help subnet leader nodes do upper-consensus as shown in Fig.3.

After a leader node of a subnet goes through a round of the lower-level consensus (trimmed PBFT), a block with the signatures of the subnet nodes is broadcast to other subnet leader nodes and boot nodes, then the block would be added to blockchain in order

Block		
Subnet Commit	BlockHash	The block hash
	Subnet	The ID of the subnet
	Leader	The leader account address
	Version	Current block hash in subnet
	Nonce	The index of this block
	Transactions	Ordered transactions
	Subnet votes	Signatures to vote the transaction order
Block Ordering Service	PrevHash	Previous block hash
Leader Validation	Receipt	The validating results of transactions
	Leader Signatures	Leader signatures to vote this block

Table 5: The data structure of the block

based on the block ordering service.

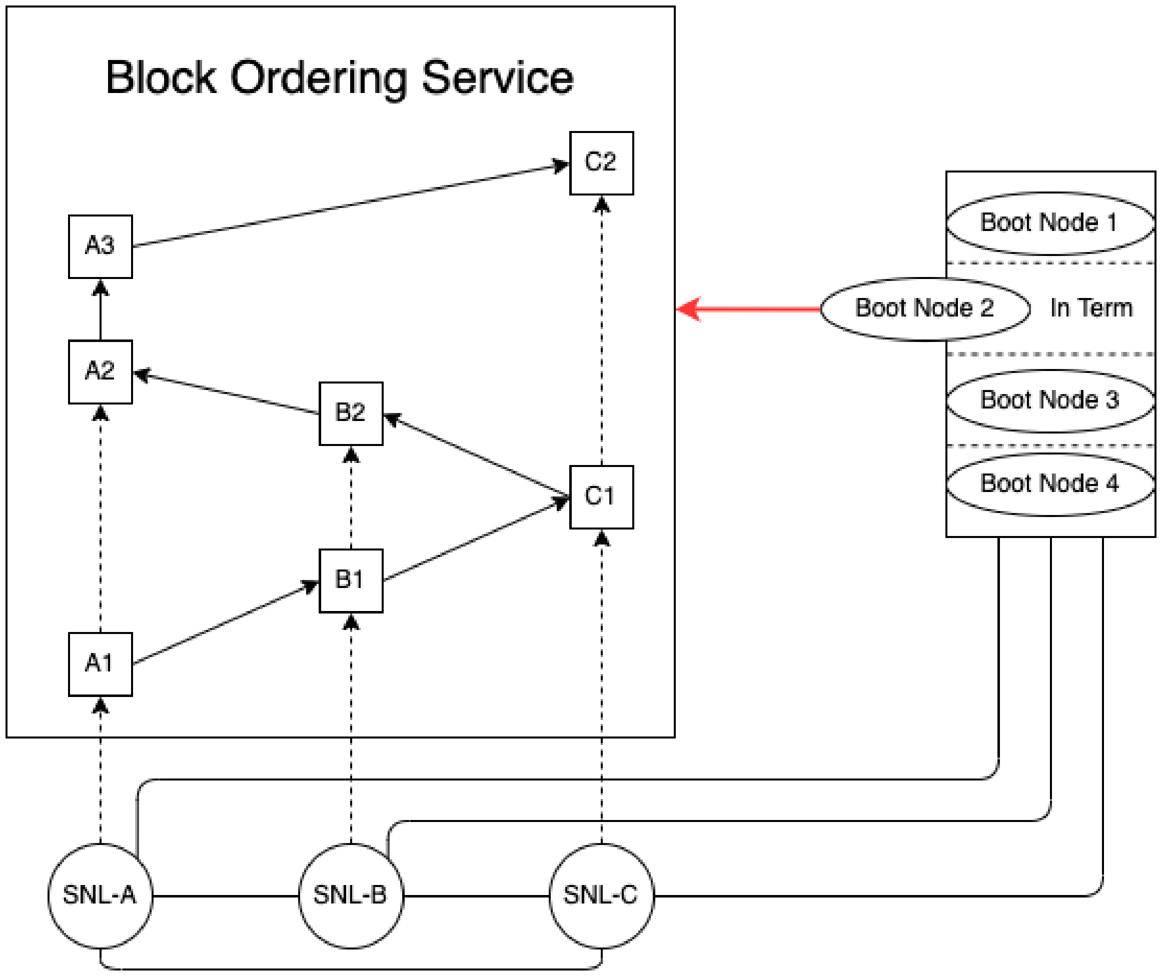

Figure 3: The ordering service provided by boot nodes.

Only one boot node is enough to provide the block ordering service in theory. However, to avoid the one-fault problem, multiple boot nodes should be involved and runs Raft consensus to determine which one is in term to provide the block ordering service.

In addition, subnet leader node can be the boot node at the same time. In this case, the whole system is much more decentralized but it requires subnet leader node with more communication and computation resources to keep high efficiency.

4 Why DP-Chain

To clarify the innovation and indeed requirement of constructing DP-Chain, we should know the architecture and workflow of some common blockchain systems. By comparing with these current resolutions, the pros and cons of DP-Chain would be well understood.

4.1 Others: Ethereum

Ethereum is a public blockchain system for decentralized applications. It is a programmable version of Bitcoin, and people can make changes and add functionalities using smart contract on it. Though Ethereum is not a consortium blockchain, it does help the whole blockchain community to step into the consideration of enabling block more functional by leveraging smart contracts. Coincidentally, the smart contract becomes the key to consortium blockchain.

The innovation of Ethereum is somewhat easy to understand, but not so does the real implementation. In this paper, we have no need to be a prof of Ethereum but just go through the transaction workflow from a top level shown in Fig.4 to know the difference between public chain and consortium chain.

Unlike Bitcoin uses UTXO model in which the balances of accounts should be collected by looking through the blockchain, Ethereum introduces the concept of world state to retrieve balances and other important data. Accounts and balances are stored as key-value pairs in a database, and this database is copied by every node in Ethereum, thus, the state of this database is so called the world state. Transactions and blocks are generated to transit the world state, and the main challenge of the blockchain system is to maintain the consistency of the world state. Every time a new block containing several transactions is appended to the blockchain according to consensus mechanism, the world state changes.

As shown in Fig.4, we assume a user wants to send a transaction which invokes a function of a smart contract. A user uses an application (e.g. Geth) on client nodes to create a transaction Tx X which calls the function. Then the application would use the information provided by the user to sign Tx X and broadcasts it to the network. After a while, Tx X is known by the mining nodes. Take one of the mining nodes for a look, it first adds Tx X into its transaction pool. Every time, the mining node chooses some proper transactions (transactions are in different privileges which determined by the mining node itself) to validate, and then creates a half-formed block with sequence of transactions. Then, every transaction in the block is executed (smart contract is executed at this time) and a new world state is got on the mining node temporarily (that is to say this new world state has not achieved in consensus). Then the mining node runs the PoW (proof of work) consensus to strive for the right to add its new block to the blockchain with the other mining nodes. Finally, a block containing the Tx X is in consensus and the whole network of Ethereum synchronize the world state. In brief, the life circle of transaction is ordering, execution and consensus.

Ethereum is a public chain, it allows any node to take part in the network and any user to user to send transactions. To use the service of Ethereum, the user should pay the fee. Though fee is not in consideration in most scenarios of consortium blockchain, it does play an important role to make the public chain work properly. In Ethereum, every

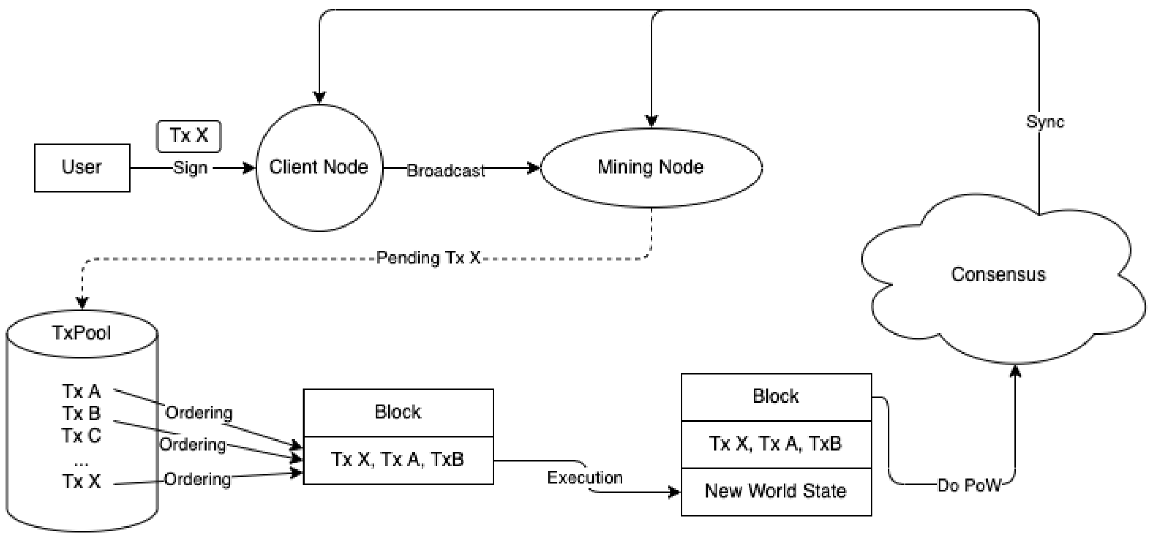

Figure 4: Transaction workflow of Ethereum.

transaction should be attached with some assets (i.e. ethers) when they are created. Mining nodes would only pend transactions attached with corresponding assets, which resists the DDoS attacks. Data read, write, operation and so on all create a fee when executing the transaction. The mining node would sum fees in real-time. Once the transaction runs out of its assets, the transaction execution is terminated and attached assets are not back any more. The fee constrains anonymous users to invoke smart contract function without any consideration. Thus the execution consumption (time, energy and storage) of transactions are limited and the system efficiency improves.

4.2 Others: FISCO BCOS

FISCO BCOS is an efficient consortium blockchain platform which supports smart contract. It was launched by FISCO open-source working group in 2017. Now its open-source community has grown as the largest and most active consortium blockchain ecosystem in China, where thousands of institutions and ten thousands of individual developers collaborate together. It has been constantly updated for multiple versions, but the basic architecture of it has not been changed much through the years. Generally speaking, the architecture of FISCO BCOS is not difficult to understand as it is more or less like the Ethereum.

To have a basic realization of FISCO BCOS, let's go through the workflow of it as shown in Fig.5. Note that, FISCO BCOS are modularized with different consensus mechanism, cryptography technology and smart contract, and Fig.5 only shows the common case.

The concept of world state is also used in FISCO BCOS, where a distributed database is maintained by the whole network with consistency. We assume a user needs to send a transaction to invoke a smart contract function. The user uses an application to create the transaction and sign it. Then signed transaction is sent to the access node (in most scenarios the application could be deployed on an access node directly). Access nodes would roughly validate the transactions collected and broadcast the valid ones to consensus nodes. Consensus nodes then runs PBFT consensus mechanism. Due to the characteristics of PBFT, every consensus node votes to determine the orders of transactions sealed in a block, and so does the result block executing. After block executing, the new world state is only stored in the block. Once a block is accepted by most of the consen-

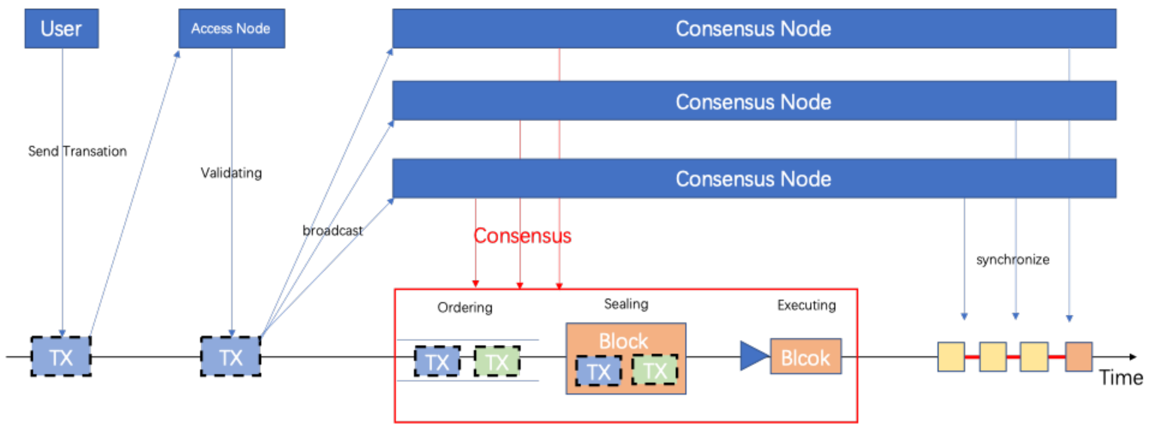

Figure 5: FISCO BCOS Workflow.

sus nodes, the block would be added to the blockchain and finally be seen by every node through synchronization. After synchronization, the world state is formally updated to that contained in the new block.

The FISCO BCOS is a natural evolution from public chain to consortium chain. The biggest difference between FISCO BCOS and Ethereum lays on the consensus mechanism. The consensus mechanism based on voting (e.g. PBFT) benefits the system efficiency of FISCO BCOS in three aspects as follows.

- 1) Fee is no need, every node in consortium blockchain are forced to maintain the world state without any reasons. Thus, a smart contract function is allowed to be frequently called. Smart constructs could be designed with less constrictions.
- 2) Block sealing is agreed on before block executing, unlike PoW in which every mining node seal a different block with different transactions order. This feature makes the block executing amd validating much easier.
- 3) No forking problems should be considered about. Once a new valid block is synchronized on the local ledger, the world state is updated immediately. In contrast, nodes in public chain have to wait for a long time to use a new seen block in order to avoid forking problems.

4.3 Others: Hyperledger Fabric

Hyperledger Fabric is an open source project, released by the Linux Foundation [2]. After continuos version iterations, it now introduces an architecture for enterprise grade blockchain platforms based on permission. Fabric follows the novel paradigm of execute-order-validate for distributed execution of smart contracts (which is known as chaincode in Fabric).

Fabric aims to be customized and fit in most application scenarios of consortium blockchain. Thus, Fabric introduces a slightly complex system structure, so as some special terms. Before going through the system architecture, some special terms should be specified first to make it much more understanding.

Every instance contained in the Fabric blockchain system is a node. There are order nodes, peer nodes and etc. Order node are could be seen as thee administer of Fabric, and could be more than one. Peer node, so called peer, contains three kinds of nodes, which are endorser peer, committer peer and anchor peer. Order nodes help organizations to construct channel, and after that order node have no right to interfere the channel and only provide it with order service. There are multiple organizations in a channel, and every organizations have some peers. Usually, every organization would have at least one endorser peer. Chaincode should be installed on endorser peer. Chaincode contains the logic of a smart contract. Also there's a endorsement policy associated with the chaincode that states out which peers (usually the endorser peer) should endorse the transactions produced using the smart contract.

The architecture of Fabric could be realized as a three-layer one, which are ordering layer, channel layer, and external application layer. In ordering layer, there are order nodes. Order nodes take the duty of configuring the whole network and help organizations to construct channel. In channel layer, there are various kinds of peers belonging to different organizations. Every organization should have at least one endorser peer in a channel, because endorser peer is where chaincode installed and executed. An organization could have committer peers and anchor peers, too. Committer peers perform validation of transactions after ordering by the order node. Anchor peer is used when peer from one organization need to communicate with another organization in a channel. A node of an organization in a channel could be endorser peer, committer peer and anchor peer simultaneously. In the external application layer, users or clients use an external applications to send request to endorser peers in a channel to produce transactions according the installed chaincode (or say smart contract). The application is a shim, and is designed to help users and clients edit and send their transaction proposals. Though application is external as they are not a part of Fabric, an application could be executed on a peer node.

Let's dive into the realization of Fabric by going through the workflow as shown in Fig.6. We assume there are three order nodes to control the whole Fabric network and provide ordering service. There are three organizations in a channel, and endorser peers EP1, EP2, EP3 belong to them respectively. Also there are committer peers CP1, CP2, CP3 belong to these three organizations respectively. Every peer in this channel has a copied ledger and are the same. A chaincode has been installed on EP1, EP2, and EP3. An application APP is provided by EP1 to help client and user using the smart contract service. Now, a client is prepared to send a transaction according to the smart contract provided in this channel.

- At stage 1, the client use the APP to produce a transaction proposal which contains the identity of the client, transaction payload and signature. Then this transaction proposal is sent to EP1, EP2, and EP3.

- At stage 2, every endorser peer would validate and execute the transaction in a sand box, then get a value change set. The value change set would not be updated to the database of these endorser peer at this time, they would add this value change set into a transaction endorsement message. The endorsement messages is signed and sent back to APP by the endorsement peers respectively.

- At stage 3, the APP would collect the endorsement transactions and validate them. The APP determines if the specified endorsement policy has been fulfilled before submitting (i.e. did all endorser peers endorse). If so, the APP compresses the transaction proposal and collected endorsement responses into a full transaction and broadcasts it to the ordering service. As the APP is provided by EP1, EP1 plays a role in this stage to help APP ask order node OD1 for ordering service.

- At stage 4, OD1 collect transactions from the channel it is in charge. The transaction will contain the read/write sets, the endorsor peers' signatures and the Channel ID. After ordering transactions with the same Channel ID, OD1 creates a block containing these transactions and delivers it to all peers in the channel. The transactions within the block are validated by peers to ensure endorsement policy is fulfilled and to ensure that there have been no changes to ledger state for read set variables since the read set was generated by the transaction execution. Transactions in the block are tagged as being valid or invalid. Each peer appends the block to the channel's ledger, and for each valid transaction the write sets are committed to current state database.

After these four stages, an event is emitted by each peer to notify the APP that the transaction has been immutably appended to the chain, as well as notification of whether the transaction was validated or invalidated.

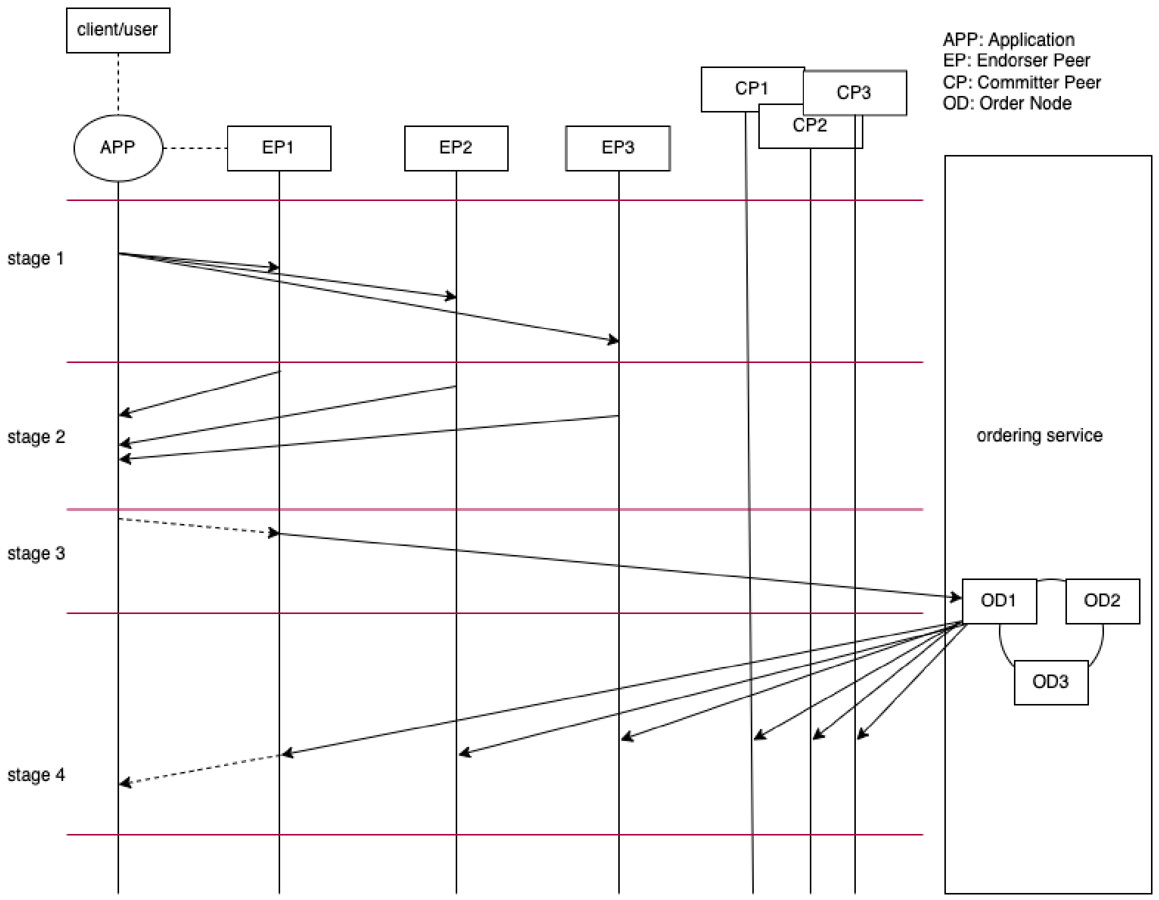

Figure 6: Fabric Workflow.

Fabric is a redesigned consortium blockchain system and is totally different from a public blockchain system.

Nodes in Fabric are highly identified and are only permitted to access the blockchain system with a specific certification. The proper work of Fabric rely on massive certifications and configurations.

4.4 DP-Chain work process

4.5 Pros and Cons

Pros: only two kind of nodes in DP-Chain: Subnet Leaders and Normal nodes.

5 Applications

5.1 Assets Scenarios

Assets application not only in financial field.

5.2 Federated Learning

Federated Learning on DP-Chain.

5.3 Information Records and Share in Industrial Network

Risk evaluation due to the business records.

5.4 MEC Scenarios

DP-Chain architecture can well fit the MEC backbones.

6 Implementation

Step 1: Build the base P2P network of DP-Chain.

Step 2: Configure each subnet and the leaders will be on behalf of their own subnets to create a subnet configuration information and spread it to the whole network. This configuration information spreading is realized by whisper protocol.

Step 3: Nodes in the same subnet will begin to run its own trimmed PBFT consensus at the same time which is configured. It is important to say there is a world state called subnets configuration which records the configuration of each subnets such as the leader of each subnets.

References

[1] Bitcoin.https://bitcoin.org.

[2] Apache Kafka.https://kafka.apache.org.