Hyperledger Fabric

Hyperledger Fabric is a permission distributed ledger technology (DLT) platform, designed for business organizations. Here the fabric was intended to provide framework for developing solutions with a modular architecture. And hyperledger allows the components to be plug and play. It is a private and permission Blockchain system which means the members enroll through a Membership Service Provider (MSP). It gives the ability to a group of organizations to form a channel to perform transactions within the channel and maintain a separate ledger. The organizations that take part in building the Hyperledger Fabric network are called the “members”. Each member organization in the network will setup their peers for participating in the network. All of these peers need are configured with appropriate certificates and permission information. Peers within the member organization receive transaction invocation requests from the clients of the organization.

A client can be of any specific application/portal serving specific organization/business activities. Client application uses the service provided by the Hyperledger Fabric network to initiate the transactions. All the peers maintain one ledger per channel that they are registered to providing the distributed ledger within the channel. Unlike to Bitcoin/Ethereum where all peers are same, Hyperledger Fabric blockchain network peers have different roles.

Roles in Hyperledger Fabric network:

* Client / Peer
* Endorser peer
* Orderer peer
* Commit peer

**Peer:**

As described the peer is a member in the organization’s channel which initiates the transaction request. Based on the outcome of the endorsement result of the proposed transaction, the peer either forwards the proposal to ordering service in turn to commit service.

**Endorser peer:**

Endorsing peer on receiving the “transaction invocation request” from the client application validates the transaction. Checks certificate details and roles of the requester. It executes the smart contract and simulates the outcome of the transaction. At the end of the above two tasks the Endorser may approve to disapprove the transaction. But it does not update the ledger.

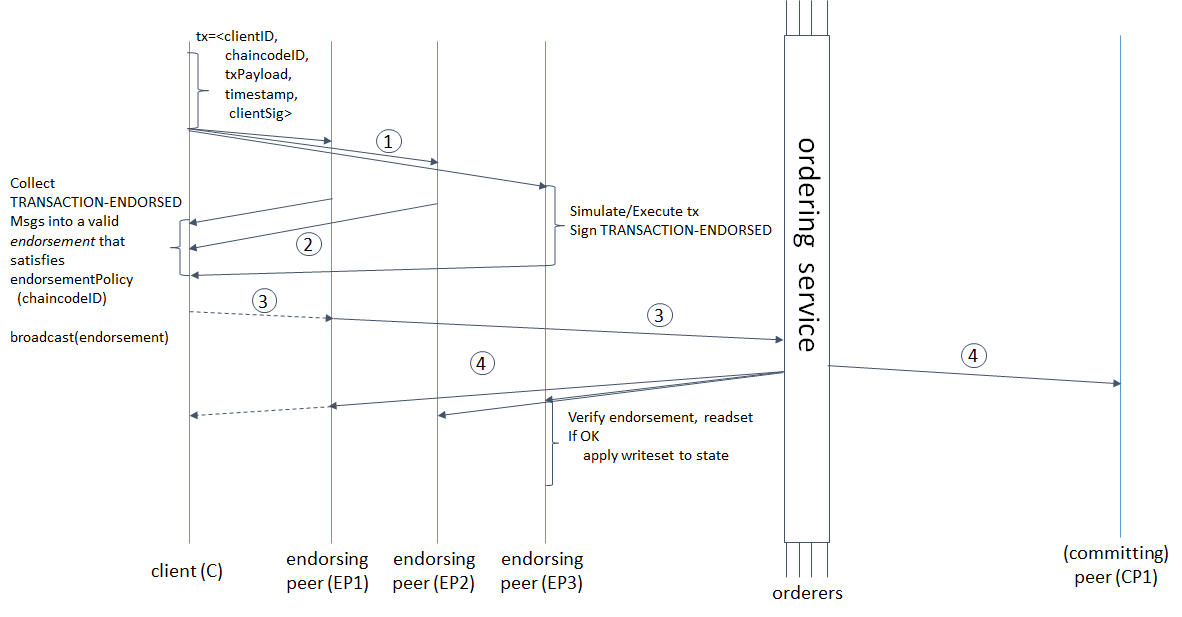
**Orderer peer:**

Orderer peer is considered as the central communication channel for the Hyperledger Fabric network. Orderer peer is responsible for consistent ledger across the network. It is responsible for creation blocks and delivers that to the network. Orderer is built on top of message oriented architecture. In production ready system Kafka is used to implement the orderer peer. Kafka is messaging software that has high throughput fault tolerant feature.

**Commit Peer:**

Commit peer is responsible for the committing the blocks given by the orderer. Committing peer updates the state resulting from executing valid transactions of the system.

Transaction flow:

****

1. Participant in the member Organization invokes a transaction request through the client application.
2. Client application broadcasts the transaction invocation request to the Endorser peer.
3. Endorser peer checks the Certificate details and others to validate the transaction. Then it executes the smart contract and returns the Endorsement responses to the Client. Endorser peer sends transaction approval or rejection as part of the endorsement response.
4. Client now sends the approved transaction to the Orderer peer for this to be properly ordered and be included in a block.
5. Orderer node includes the transaction into a block and forward the block to the committing peer.
6. Committing peer then commits the block in to the ledger and publishes the ledger to the network. By this the transactions which are only valid are being added to the ledger and made known to the peers in the channel.

Corda

Corda is a blockchain-inspired distributed ledger technology that currently targets finance use cases. Like Hyperledger Fabric, it is a permission network where all participants have verifiable identities using public-key infrastructure. Corda allows parties to transact directly, with value and does not use a blockchain to record transactions. Smart Contracts allow Corda to perform transactions between the participants of a transaction. This capability has broad applications across industries including finance, supply chain and healthcare.

In general the blockchain network replicates the ledger across the members of the network. In the Bitcoin and Hyperledger Fabric platform the ledger is shared to either every member of the network or the members in the channel. Corda views this approach as privacy concern and restricts the member to maintain their states with themselves. The only entities that should be aware of a transaction are the parties directly involve. If an organization issues a transaction that it owes a customer *n* dollars, only the organization, the customer, and relevant regulatory organizations need to know anything about the transaction.

**Communication System:**

To make sure only the involved parties know about a transaction, the first step is to control who the transaction is sent to. In general, blockchain networks heavily use *broadcast*, where messages are distributed to every node in the network. This is more appropriate to Bitcoin where the members are unknown. A Corda network includes a network map service that publishes information about how to reach every identity in the network. This allows any entity to specifically contact any other entity directly.

**No Blocks:**

According to Corda transactions are no longer needed to be batched together into blocks. The one reason blockchains batch transactions together into blocks are that it takes time to distribute a block across the entire network. In that time, many new transactions can occur. Especially in a proof-of-work system where every node is trying to calculate the *next* block, it’s important that the network have a uniform idea of what the *last* block was. This means that blocks need to be created slowly enough that each has already been distributed across the network before the next one is created.

Corda uses point to point communication, so it does not have to account for the time it takes to broadcast a message to the entire network. Each transaction is sent directly to the involved parties which is very quick to process. However, because transactions are not broadcasted to the entire network, no entity has an entire history of all the transactions in the network.

**Role of members:**

As discussed corda assigns the roles/identities to its members in the initial stage .

* Client
* Doorman
* Notary

**Client:**

The Client is a participant of the network which would like to perform a transaction of assets. In Corda the assets are defined based on the digital representation called Unspent Transaction Output (UTXO). Each client obtains these UTXOs either from an initial transaction performed by central issuing authority or by the involving in a valid transaction.

**Doorman:**

The Doorman is responsible to registering the members in the network and providing the necessary the network services to interact with the other members. Doorman also serves as the authentication service to authenticate the members in each session.

**Notary:**

A notary is a network service that provides uniquenessconsensus by attesting that, for a given transaction, it has not already signed other transactions that consumes any of the proposed transaction’s input states.

Upon being sent asked to validate a transaction, a notary will either:

* Sign the transaction if it has not already signed other transactions consuming any of the proposed transaction’s input states
* Reject the transaction and flag that a double-spend attempt has occurred otherwise

From the above discussion, reaching the point of finality in the corda system is the responsibility of Notary service. Until the notary cluster’s signature is obtained, parties cannot be sure that an equally valid, but conflicting, transaction will not be regarded as the “valid” attempt to spend a given input state. However, after the notary cluster’s signature is obtained, we can be sure that the proposed transaction’s input states have not already been consumed by a prior transaction. Hence, notarization is the point of finality in the system.

Every state has an appointed notary cluster, and a notary cluster will only notarize a transaction if it is the appointed notary cluster of all the transaction’s input states.

**Consensus algorithms:**

Corda has “pluggable” consensus, allowing notary clusters to choose a consensus algorithm based on their requirements in terms of privacy, scalability, legal-system compatibility and algorithmic agility.

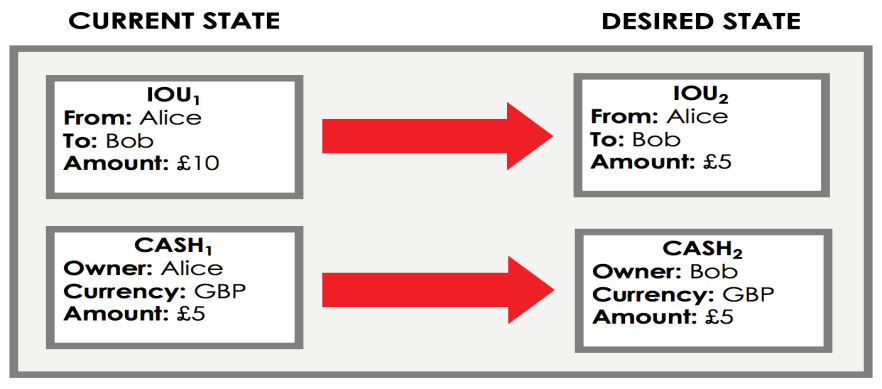
In particular, notary clusters may differ in terms of:

* **Structure** - a notary cluster may be a single node, several mutually-trusting nodes, or several mutually-distrusting nodes
* **Consensus algorithm** - a notary cluster may choose to run a high-speed, high-trust algorithm such as RAFT, a low-speed, low-trust algorithm such as BFT, or any other consensus algorithm it chooses

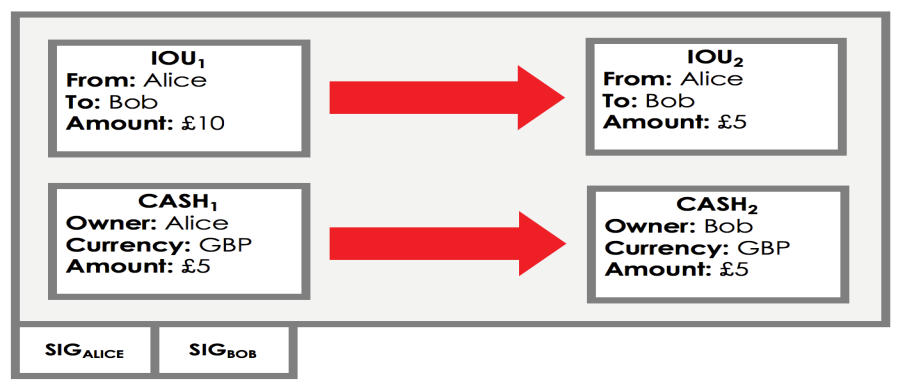
**Transaction Flow:**

The transaction flow in the corda is simpler compared to the other distributed ledger platforms. Transactions in Corda are constructed in stages and contain a number of elements. A transaction is just a proposal to update the ledger.

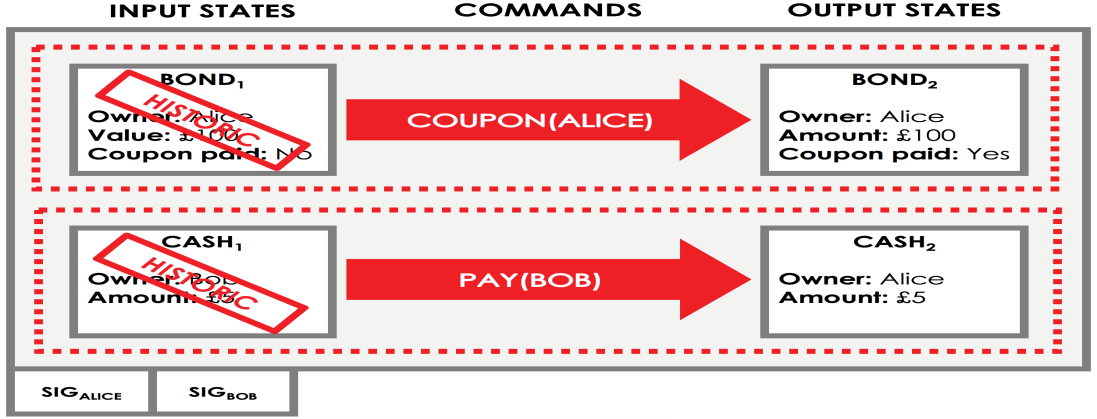
1. Both parties of an individual transaction initiates the transaction with a set of input state references that will be consumed by the final accepted transaction and their respective output references.



1. Upon agreeing on the output state of the reference objects each participant signs on the transaction.



1. The group of transactions upon agreeing on the output state will be forwarded to the notary service for transaction validation.
2. Notary service validates the input state references of each transaction . Basically it verify the below two conditions.
   1. Check if the input state references provided are valid UTXOs.
   2. Check if the given UTXOs is not used in another transaction.
3. Based on the outcomes of the conditions Notary updates its state which is a collection of UTXO’s in the system .
4. Update by the notary can be referred as the commit to the ledger of the system
5. The clients upon receiving the success response from the notary makes their input reference as history and the output references at their current state.



The history of the state references (UTXOs) of each individual client is stored in their respective vaults. Though the discussed steps are only the basic transaction flow, Corda provides many types of flows based on the assets.

Quorum Systems

Quorum is an Ethereum-based distributed ledger protocol that supports transaction and contract privacy. The primary features of Quorum are:

* Transaction and contract privacy.
* Voting-based consensus mechanism.
* Network and peer permissions management.
* Higher performance.

Apart from these features, Quorum includes the powerful feature of support for private and public transactions.

**Private transactions:**

Transactions whose payloads are only visible to the network participants whose public keys are specified in the “*privateFor*” parameter of the transaction. “*privatefor*” can take multiple addresses in a comma-separated list.

**Public transactions:**

Transactions whose payloads are visible to all participants of the same Quorum network. These are created as standard Ethereum transactions in the usual way. The treatment of both types of transactions is different. Public transactions are sent to an account that holds contract code. Each participant will execute the same code and their underlying StateDBs will be updated accordingly. For private transactions, it replaces the original transaction payload with a hash of the encrypted payload that it receives from the constellation. Participants who are party to the transaction will be able to replace the hash with the actual payload via their constellation instance, whilst those participants that are not parties will only see the hash.

**Participants in Quorum System:**

The participants of the network are defined as below.

1. Quorum Node
2. Constellation & Tessera -Transaction Manager
3. Constellation & Tessera - Encalve

**Quorum Node:**

Quorum Node is a simple node whose major responsibility is to maintain two databases public and private. Quorum node accepts the transaction requests from the clients/dapps and then forwards the same to the network.

**Constellation & Tessera:**

Constellation and Tessara are a general-purpose system for submitting information in a secure way. They are comparable to a network of MTA (Message Transfer Agents) where messages are encrypted with PGP. It is not blockchain-specific, and are potentially applicable in many other types of applications where you want individually-sealed message exchange within a network of counterparties. The Constellation and Tessera modules consist of two sub-modules:

* The Node (which is used for Quorum's default implementation of a PrivateTransactionManager)
* The Enclave

**Transaction Manager:**

Quorum’s Transaction Manager is responsible for Transaction privacy. It stores and allows access to encrypted transaction data, exchanges encrypted payloads with other participant's Transaction Managers but does not have access to any sensitive private keys. It utilizes the Enclave for cryptographic functionality (although the Enclave can optionally be hosted by the Transaction Manager itself.)

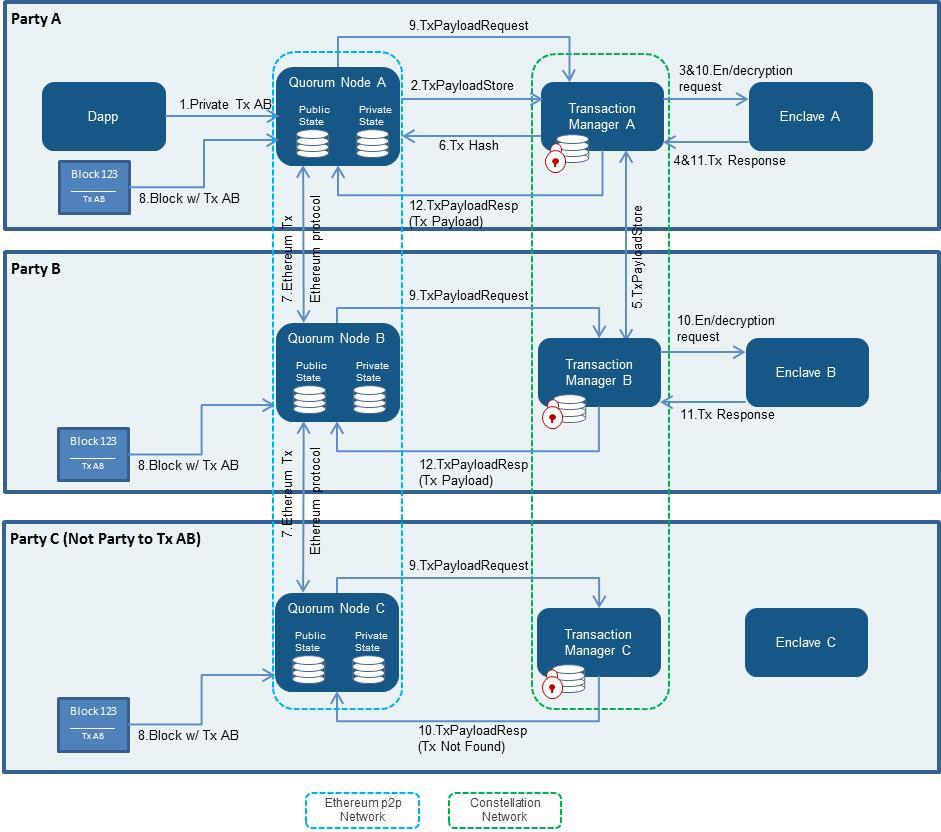
The Transaction Manager is restful/stateless and can be load balanced easily.

**Enclave:**

Distributed Ledger protocols typically leverage cryptographic techniques for transaction authenticity, participant authentication, and historical data preservation (i.e. through a chain of cryptographically hashed data). In order to achieve a separation of concerns, as well as to provide performance improvements through parallelization of certain crypto-operations, much of the cryptographic work including symmetric key generation and data encryption/decryption is delegated to the Enclave.

The Enclave works hand in hand with the Transaction Manager to strengthen privacy by managing the encryption/decryption in an isolated way. It holds private keys and is essentially a “virtual HSM” isolated from other components.

**Internals of Quorum**



The transaction flow can viewed from the above picture. The transaction includes the private transaction happening between A and B. The third member C belongs to the network but will be unable to view the transactions between A and B.

1. The client sends the request corresponding Quorum node (i.e., A => Quorum Node A, including transaction A to B).
2. A’s Quorum node passes the transaction on to its paired transaction manager (Transaction Manager A), requesting for it to store the transaction payload.
3. A’s Transaction manager makes a call to its associated enclave to validate the sender and encrypt the payload.
4. A’s enclave checks the private key for Party A and, once validated, performs the transaction conversion.
5. Party A’s transaction manager then stores the (encrypted payload , encrypted symmetric key). And then securely transfers the (hash, encrypted payload, encrypted symmetric key) that has been encrypted with Party B’s public key to Party B’s Transaction Manager.
6. Party B’s Transaction Manager responds with an Ack/Nack response.
7. A’s Transaction Manager returns the hash to the Quorum Node, which then replaces the Transaction’s original payload with that hash.
8. Then transaction is propagated to the rest of the network using the standard Ethereum P2P protocol.

A block containing Transaction AB is created and distributed to each party on the network.

1. All the parties attempt to process the transaction. A and B make a call to its enclave, passing in the encrypted payload, encrypted symmetric key, and signature. However the member C will receive a *NotARecipient* message.
2. The respective enclaves tries to validate the signature and then decrypts the symmetric key using the party’s private key that is held in the enclave.

The transaction is then decrypted to payload using the now-revealed symmetric key, and returns the decrypted payload to the transaction manager.

1. The transaction managers for parties A and B then send the decrypted payload to the EVM for contract code execution. This execution will update the state in the Quorum node’s private StateDB only.

MultiChain

MultiChain is an off­the­shelf platform for the creation and deployment of private blockchains, either within or between organizations. It aims to overcome a key obstacle to the deployment of blockchain technology in the institutional financial sector, by providing the privacy and control required in an easy­to­use package. MultiChain solves the related problems of mining, privacy and openness via integrated management of user permissions. The aim of the multichain can be defined by the below points:

1. To ensure that the blockchains activity is only visible to chosen participants,

2. To introduce controls over which transactions are permitted, and

3. To enable mining to take place securely without proof of work and its associated costs.

Once a blockchain is private, problems relating to scale are easily resolved, since the chain’s participants can control the maximum block size. In addition, as a closed system, the blockchain will only contain transactions which are of interest to those participants. To understand permissions in MultiChain, we begin by noting that all crypto currencies manage identity and security using public key cryptography. Users randomly generate their own private keys and never reveal them to other participants. Each private key has a mathematically related public address which represents an identity for receiving funds. Once sent to a public address, those funds can only be spent using the corresponding private key to “sign” a new transaction. In this sense, access to a private key is equivalent to ownership of any funds which it protects. Beyond controlling access to funds, this type of cryptography enables any message to be signed by a user to prove that they own the private key corresponding to a particular address. MultiChain uses this property to restrict blockchain access to a list of permitted users, by expanding the “handshaking” process that occurs when two blockchain nodes connect:

1. Each node presents its identity as a public address on the permitted list.

2. Each node verifies that the other’s address is on its own version of the permitted list.

3. Each node sends a challenge message to the other party.

4. Each node sends back a signature of the challenge message, proving their ownership of the private key corresponding to the public address they presented.

If either node is not satisfied with the results, it aborts the peer­to­peer connection.

Since multichain is based on the blockchain technology but provides the execution of transactions in a private mode. Since it's a private Blockchain, there is a lot more control about the miners in the chain. This allows having specific mining settings for the chain like mining diversity, blocking size and frequency and as such solving the capacity problems.

A transaction in multichain is similar to that of blockchain, but the only difference is that each node performing a transaction should have the necessary permissions in subscribed private blockchain. Private Blockchain can be informally defined as the set of nodes performing transactions with necessary permissions in that chain.

Permissions Management:

In MultiChain, all privileges are granted and revoked using network transactions containing special metadata. The miner of the first “genesis” block automatically receives all privileges, including administrator rights to manage the privileges of other users. This administrator grants privileges to other users in transactions whose outputs contain those users’ addresses together with metadata denoting the privileges conferred. When changing the administration and mining privileges of other users, an additional constraint is introduced, in which a minimum proportion of the existing administrators must vote to make a change. These votes are registered by each administrator in a separate transaction, with the change applied once sufficient consensus is reached. The first few blocks of a chain define a “setup phase”, in which a single administrator is able to bypass this voting process. Future versions of MultiChain could also introduce “super administrators” who can assign and revoke privileges on their own.

Mining in MultiChain:

By restricting mining to a set of identifiable entities, MultiChain resolves the dilemma posed by private blockchains, in which one participant can monopolize the mining process. The solution lies in a constraint on the number of blocks which may be created by the same miner within a given window. MultiChain implements this scheme using a parameter called mining diversity, which is constrained by 0 ≤ mining diversity ≤ 1. The validity of a block is verified as follows:

1. Apply all the permissions changes defined by transactions in the block in order.
2. Count the number of permitted miners who are defined after applying those changes.
3. Multiply miners by mining diversity, rounding up to get spacing.
4. If the miner of this block mined one of the previous spacing­1 blocks, the block is invalid.

This enforces a round­robin schedule, in which the permitted miners must create blocks in rotation in order to generate a valid blockchain. The mining diversity parameter defines the strictness of the scheme, i.e. the proportion of permitted miners who would need to collude in order to undermine the network. A value of 1 ensures that every permitted miner is included in the rotation, whereas 0 represents no restriction at all. In general, higher values are safer, but a value too close to 1 can cause the blockchain to freeze up if some miners become inactive.

If mining is restricted to certain entities, which is similar to the centralized systems where a central system accepts the incoming transactions and updates the ledger then what is the advantage of going to private blockchain?

. The answer is threefold:

1. Each participant retains full control over its assets via their private key. Even miners cannot create transactions that spend another party’s funds.
2. Control of the database is distributed across many entities, so that no individual or small group can unilaterally decide which transactions are valid or will be confirmed.
3. Superior robustness, since the disappearance or malfunctioning of one server will not affect the continued processing of transactions by the network as a whole.

In practice, the MultiChain uses a bitcoin­style proof of work to regulate and randomize each node’s rate of block production, but this is not the basis of the blockchain’s security.

In a MultiChain blockchain, transaction fees and block rewards are zero by default. If the cost of mining a block is negligible, miners need no compensation for providing this service beyond their general stake in the blockchain’s smooth functioning. Alternatively miners might charge network participants a fixed annual service fee, paid by traditional off­blockchain means. If a blockchain’s sole purpose is to enable transactions in tokenized assets, its “native” currency might safely be ignored as an evolutionary artifact. However, if transaction scarcity is desired, MultiChain can also be configured to use a native currency for block rewards, minimum transaction fees and output quantities. In this case participants would need to purchase units of the native currency from miners, perhaps in exchange for a tokenized asset.

Connecting to Private Chain:

Transaction flow / Block Creation:

To create a new blockchain, the user has to perform two simple steps.

1. The user chooses a name for the chain, upon which MultiChain creates a configuration file containing the default settings.

This file can be modified by the user, although the defaults will be suitable for common use cases.

1. Second, the user launches the blockchain, upon which the genesis block is mined by MultiChain, granting its creator all user privileges.

At this point, MultiChain also embeds details of the genesis block along with a hash of all the blockchain’s parameters in the configuration file, in order to prevent subsequent accidental changes.

Adding new Node:

1. To add a new node, MultiChain is run from another node with three parameters:

1. The destination blockchain name,
2. Its IP port number, and
3. The IP address of an existing node.

e.g. chain1@12.34.56.78:8571 .

At first a new node will not be permitted to connect, since the network is private and the node has not yet been granted connection privileges. MultiChain will display a message containing the new node’s self­generated public address, which must be sent to an administrator. The administrator grants connection privileges to this address via a simple command that creates the appropriate transaction. The new node can then reconnect successfully and automatically downloads the configuration file defining the blockchain’s characteristics. Any future connections to the same blockchain only require the chain name to be specified, with the handshaking process between nodes ensuring that they use identical parameters.

Transaction flow:

A message is sent from the originator to the recipient as follows:

1. The originating MultiChain node sends a message transaction to the recipient with metadata containing its IP address and a hash of the message’s content .
2. The receiving node receives the transaction and decodes this metadata.
3. The receiving node contacts the originating node via its IP address to retrieve the message, signing the request in order to prove its identity as the intended recipient. This communication takes place using the blockchain’s existing peer­to­peer protocol.
4. The receiving node verifies the message’s validity by checking its hash against the hash embedded in the original transaction.
5. If the message is valid, the receiving node completes the loop by sending back a second transaction to the sender containing the same message hash.

Once the first transaction is confirmed on the blockchain, the recipient can prove:

1. Who sent the message, since the sender reveals their address when signing the transaction,
2. The time the message was sent, since the transaction is embedded in a time stamped block, and
3. The message’s content, since the hash is part of the transaction that was signed.

However none of this is sufficient for the sender to prove that a particular message was received by the recipient. Indeed, malicious senders could even embed a hash of one message while sending a completely different message in step 3 above. Therefore a second transaction is required in which the recipient sends back a receipt containing the same hash. Once this transaction is confirmed, both parties can prove all the details of the correspondence that took place.

Ensuring privacy:

In any blockchain, all transactions are publicly viewable to all participants, and this creates a fundamental problem in terms of privacy. First, blockchains enable each participant to gain a global picture of the aggregate volume of assets being held and traded. Depending on the use case, this transparency may or may not be desirable. However a more serious problem is that participants learn the public addresses of other participants when transacting with them, enabling them to infer their counterparty’s full balance and trading activity in both the past and future.

Multichain addresses this problem by restricting the users to even view the hash of the public keys to an unregistered user. Though the users of same private blockchain will be able to view the transactions, hash keys within the same chain. This gives the users a medium level of privacy compared to the public blockchains.

Thus the multichain technology provides the private blockchain to perform the business level transaction privately in a distributed environment.

Classification of Existing BCTs:

As discussed in the above sections the existing BCTs can be classified based on the mode of communication , the data being shared in the system . Comparison can be viewed in the below table.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **M** | **TR** | **TC** | **BR** | **BC** | **SR** | **SU** | **C** |
| **Bitcoin / Ethereum** | p | p | p | p | p | p | p | p2p |
| **Hyperledger Fabric** | P | p\* | p\* | P | p\* | P | H |
| **Multichain / Quorum** | p# | p | P |
| **Corda** | P | - | - | P2P |

\* Within a channel, # Summary of transactions, p Open / Public, P Private / Permission

Membership (M):

Membership defines permissions and privileges a user can have in the system. Membership of a user in the systems could be public or private based on the platform chosen. Bitcoin and Ethereum consider all the users as the peers and the membership is public. User can disconnect from the system and join at anytime.

Hyperledger Fabric, Multichain, Quorum and Corda emphasis on the privacy of the user. These consider each user membership as private and restrict the user to view or share the data with other channels/chains. This makes these platforms more appropriate for the business community.

Communication (C):

Generally the communication in the distributed system is achived either by message passing system or by shared resources. Most of the present existing blockchain technologies communicate by message passing mechanism. Messages are broadcasted or sent only to a single party in the network making each message available to the parties. In public blockchain the processes of broadcasting the message poses a risk of loss of privacy , it provides the benefit of communicating to all the receivers in very less time.

In the existing platforms Bitcoin/Ethereum provides peer-to-peer communication where the message is been broadcasted to all the users of the system. In Corda , the communication is purely point-to-point where the users communicate only with the NOTARY and with users involved in a transaction. Hyperledger Fabric ,Multichain and Quorum provides a hybrid version of the message communication in the system. In Hyperledger Fabric the communication within the channel is peer-to-peer but the communication with the endorsement peer and ordering peer , the communication is point-to-point. This is the same case with multichain and quorum where the communication with in the same level is peer to peer where are communicating with other peers it is point to point.

Data :

Data in the system is defined as the data being generated and processed in the system. These include the data at three levels transaction, block and state. A comparison is made to verify who all can view the data at the mentioned levels and who can create the data at these levels.

Transaction:

In Bitcoin and Ethereum , a transaction can be viewed by all the members of the system. At the same time any user can create a transaction just by performing a transaction with any other user in the system. The actions transaction read and create are public in these systems.

In hyperledger farbric the transactions are viewable to only the users in the same channel. The same is true in the case of multichain and quorum . Where as in Corda , as there is not concept of distributed ledger , each user can only view the transaction data which it is involved.

Creation of a transaction is public in all the platforms , this is the very basic requirement that any platform can provide.

Block:

Block data is defined as the set of transactions arranged in a sequence and archived. Generally in public blockchains , the responsibility of creating a block is taken by miner. A miner user is same as the normal user in the system. Hence in the Bitcoin and etherreum block creation is public , anybody can create the block.

Hyperledger Fabric is a role based system where the roles for some users are predefined. In Hyperledger there endorsement peers and commiting peers to validate and commit the transaction into the block. The block create in hyperledger is only viewable to the users in the same channel. This is the same case with multichain and quorum. There is no concept of block in Corda.

State:

State of the system is defined as the number of transactions being processed and added to the blockchain and number of transaction being under process. State of the system is updated when transaction is being executed. In Bitcoin/Ethereum , since anybody can perform a transaction makes the state to update it is public to read and update the state of the system.

In hyperledger fabric , a user can read the status of the systems on within the channel and updating the state is done by the committing peer. Hence the state read is private outside the channel and purely private in updating the channel.

In Multichain and quorum the states of the system is related to the user’s state , that is the users can only read their own state . This is due to the state data being shared in these system is the form of hash. Even in these the state update is performed by a designated peer. In Corda , each individual needs to store its own state that is the history of all the transactions it is being involved. The users cannot read the state of the other users or the system making it more private.

**Comparison of Existing BCTs:**

Based on the detailed explanation of prominent blockchain technologies available in the present market, this section compares the systems on the requirements they will be guaranteeing.

**Guaranteed Message Delivery:**

Except in the Bitcoin/Ethereum, the message delivery is guaranteed in hyperledger fabric , multichain and quorum system. This is due to the reason the communication method used is point to point and hybrid (both peer to peer and point to point). Bitcoin and Ethereum used peer to peer communication where there is no guarantee on message being delivered.

**Guarantees on results of business processes:**

In the current centralized system the results of the business process can be varied due to the some below reasons.

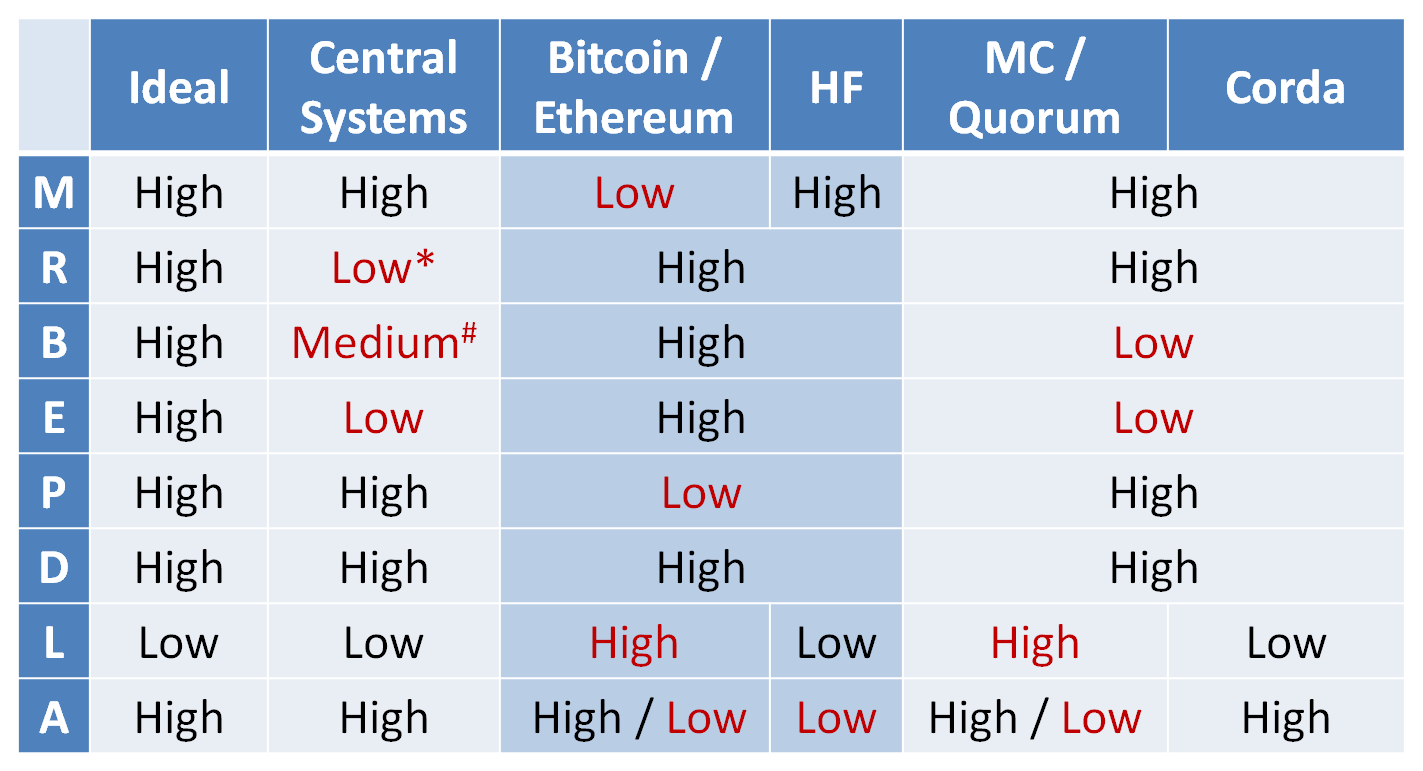
1. **Interpretation of the business process:**

Majority of differences in execution of the processes are due to the interpretation of the agreement. In general the interpretation of each parties could vary at multiple levels like, business agreement , implementation of contract.

In the existing BCT platforms the business processes is being executed using the same smart contract. Smart contract provide the same setoff execution sequences at both ends which makes the differences in the interpretation almost to zero. Compared to centralized system all the BCT platforms provides better / higher guarantee on results of the business processes.

1. **Execution environment of the business**

Since in the central systems, the parties of the transactions executes the business process at their environment which may differ can lead to differences in the outcome of the business processes. All the existing blockchain technologies execute the code in the same environment for all the parties. And due to the consensus services the outcome of the business processes would be same at both ends. This makes all the BCT technologies better than the centralized system.



Comparison table of all the Blockchain technologies.

**Business Continuity:**

Business services should be continued even if there is a faulty node in the given system. Presently the centralized system though have the fault tolerant systems like DC DR , yet there are incidents where the service got discontinued for some time. This is due to the algorithms and techniques used to design the current fault tolerant systems. Blockchain /Distributed systems provide great fault tolerant system .

Bitcoin/Ethereum and also hyperledger fabric uses the consensus mechanism where the system is fault tolerant to certain number of failures. If a node fails by crashing or by behaving in byzantine behavior , the node can recover by updating its state from the latest block of the blockchain available in the system . This guarantees a higher level of business continuity of the system.

In Multichain and quorum , the blockchain shared in the system is not the complete status of the system. In Multichain the latest block can give information about only the channel the current user is registered. So maintaining the status of the system is rested within the channel. Quorum is very similar to multichain as per the business continuity.

In Corda as discussed the user will be wholly responsible to maintain the status of the system. Each individual stores the history of the transactions in their respective wallets. If there is loss of data from the wallet or any crash, business continuity in Corda will be stopped and there is no defined ways to fetch the updated system status. Hence the business continuity in Corda is very low compared to others.

**End-to-end security:**

As defined in the requirements end-to-end security of a transaction is to provide data and sequence security at each step in a transaction. Presently in the centralized system, the sender initiates the transaction and should wait for the reply to the transaction. Sender will not have any knowledge status of the transaction or on the data being received by the receiver. Sender or receiver rests their trust on central system .

End-to-End security of a transaction is high in Bitcoin and Ethereum due to the communication method used. In Bitcoin, all the transactions are performed are peer-to-peer. The sender’s message will be received by the receiver or the intended user only. Thus, making sure that there is no third party involved in completing the transaction.

In Hyperledger fabric , Multichain and quorum the transaction is actually executed by some designated (endorsement , committing ) peers. The level of end-to-end security provided by these systems is almost same as the centralilzed system due to some kind of intermediation.

**Privacy:**

In Bitcoin/Ethereum the blockchain / ledger is distributed to all the members of the system . And also the transaction initiated but completed yet will be floating in the system due to the peer-to-peer messaging system. With these systems the loss of privacy is very high. In Hyperledger fabric , the data is shared within the channels . This makes the privacy loss within the channel but privacy preserved outside the channel.

In Multichain and Quorum , the data is shared only within the private chains and quorums. The data shared in these systems is in encrypted format which provides a high level of privacy preservation of the data. Corda in contrary does not allow any data to be shared in the system. Only the parties involved in a transaction would be able view the details of other user related to that transaction only. Corda provides a higher level of privacy.

**Data integrity:**

Data Integrity of the systems is almost at same level for all the platforms. Data at all the level transaction, block and states is being updated only when there is an update transaction being executed. This makes the all the systems to have higher level of data integrity in the systems.

**Latency:**

Latency of a transaction is defined as the time taken by the system when a transaction is submitted and the corresponding response is received. Presently the current central system has the lowest latency (faster execution) which is due the design of the centralized system where the execution means only updating of the database.

In Bitcoin/Ethereum the consensus protocol is being used to obtain an agreement on executing a transaction. Consensus is achived by message passing which indirectly induces network latency of the system. Only on receiving the specific number of matching response the transaction is assumed to executed. Hence the latency in the Bitcoin and Ethereum is expected to high. This same as in multichain and quorum but the numbers of users involved in the agreement protocol is comparatively less than in Bitcoin/Etherrum .This makes the latency of these system almost near to Bitcoin/Ethereum.

In Hyperledger fabric , the decision on execution of a transaction is moved to the endorsement and committing peers . This makes transaction execution little faster that in Bitcoin/Ethereum.

Corda used notary service to decide on execution of each transaction where the notary just verifies the validity of the unspent transaction outputs (UTXOs) and updates its state. This makes it same as the centralized system providing a low latency.

**Asynchronous processing:**

The ability of the system to execute transactions asynchronously makes the system parallel. Bitcoin platform provide a high level asynchronous processing due to the architecture of the system. In Bitcoin, the consensus on a transaction is achieved before executing the transaction . This makes the nodes to perform consensus on multiple transactions and execute. Thus Bitcoin achieves a higher level of asynchronous processing.

Ethereum in contrary to Bitcoin , it adds the transaction to the blockchain and then calls for agreement on that transaction. This logic of execution of transactions will create gaps while executing the transaction. Hence Ethereum platform is more synchronous.

Multichain and Corda have a higher level of asynchronousprocess because the first uses consensus methodology and the later uses just updating the UTXOs . Updating the UTXOs is a trivial task to be performed and hence provides high asynchronous processing.

Quorum system due its architecture, a transaction can be executed only in sequence. This makes the quorum systems more sequential in nature.

This completes the comparison of the existing blockchain platforms and centralized system against the business requirements.