Introduction

Blockchain technology is one of the most prominent emerging technologies in the 21st century. Blockchain is at the core implementation of any digital currency like BITCOIN , ETHEREUM . The potential of blockchain is not just limited to digital currency, it can be used any field of science. This is due to the features it provides. By definition blockchain is as shared and open ledger that keeps a record of the transactions and cannot be modified. To define in simple words blockchain is a chain of blocks where each block contains a set of transactions. Each block is connected to its neighboring block by the hash of the respective preceding block.

The very existence of the blockchain technology is to remove the dependency of centralized systems and maintain an open ledger of transactions being executed. The concept of open ledger is to maintain the peer to peer relationship between the parties involved in the transaction with no trusted third party.

A distributed ledger can be defined as a database that is consensually shared and synchronized across network spread across multiple sites, institutions or geographies. It allows transactions to be public and thereby making a cyber attack more difficult. The participant at each node of the network can access the recordings shared across that network and can own an identical copy of it. Further, any changes or additions made to the ledger are reflected and copied to all participants in a matter of seconds or minutes.

**Distributed Ledger Technology (DLT):**

DLT is based on the peer-to-peer (P2P) technologies at its core. The P2P technology is the base of the internet and its applications such as email, sharing music or other media files, and internet telephony are driven by the same technology. However, the internet-based transfers of asset ownership have long been elusive, as this requires ensuring that an asset is only transferred by its true owner and ensuring that the asset cannot be transferred more than once, i.e. no double-spend. The asset in question could be anything of value.

A paper written by Satoshi Nakamoto named “Bitcoin: A peer to peer Electronic Cash Systems ” in 2008 proposed a novel approach of transferring “funds” in the form of “Bitcoin” in a P2P manner. The underlying technology for Bitcoin outlined in Nakamoto’s paper was termed as Blockchain, which refers to a particular way of organizing and storing information and transactions. Subsequently, other ways of organizing information and transactions for asset transfers in a P2P manner were devised which lead to term “Distributed Ledger Technology” (DLT) to refer to the broader category of technologies.

Presently the state art of distributed ledger technology has become an emerging research area both in academia and industry. These have to lead to the development of multiple DLT platforms. Some of them are Bitcoin, Ethereum, Hyperledger Fabric, Corda, etc. Though these platforms comes in different flavors, addresses this concerns of distributed system, the main aim of all the platforms is one. That is to provide peer to peer transfer of assets without central trusted system. Some of the DLT platforms are discussed in detail in the report.

The sections in the report are structured as Section 2 provide the necessity of distributed systems, Section 3 – Promises of Blockchain technology/DLT ,Section 4 - analysis of existing distributed ledger technologies, Section 5 - proposed framework and compare with the current centralized system and existing DLTs.

Central Systems and Requirements

A ledger by definition it is a book of record keeping all the financial transactions of the organization. Since ancient times, ledgers have been at the heart of economic transactions to record contracts, payments, buy-sell deals or movement of assets or property.

The journey which began with recording on clay tablets or papyrus, made a big leap with the invention of paper. Over the last couple of decades, computers provided the process of record keeping and ledger maintenance great convenience and speed. Today, with innovation, the information stored on computers is moving towards much higher forms which is cryptographically secured, fast and decentralized.

In modern days computerized ledger came into existence and the general ledger works as a central repository for accounting data transferred from all sub ledgers cash management, fixed assets, purchasing and projects. The general ledger is the backbone of any accounting system which holds financial and non-financial data for an organization. The collection of all accounts is known as the general ledger. In a manual or non-computerized system this may be a large book. Each account in the general ledger consists of one or more pages.

**Central System:**

To create and manage the central ledger the parties involved in the transaction has to trust a central party. Thus making the central trusted party as more powerful in the sense of access to data and execution of the transaction.

In the current system of banking, a client has to put his trust on the bank which is the trusted third party to execute a simple funds transfer. For example a fund transfer in the central system is explained as in the below figure.

Funds Transfer Transaction

Steps Involved:

1. Sender initiates the transaction “Send $100 to receiver” by sending the request to his/her Sender bank
2. Sender bank on receiving the request for initiation of the fund -transfer and transfers the request to the central bank only after verifying the necessary conditions.
3. The central banks upon receiving the request from bank1, executes the request and sends the response to the sender bank and also to the receivers bank
4. Receivers bank on receiving the message from central bank updates the account of the receiver, to reflect the current updated balance.

Some of the observations

1. Each transaction is broken in to multiple hops which are independent of the result of prior hop.
2. Sender is not aware of the processing being careered out by the banks.
3. Reconciliation issues which could be due any software issues

Though the central systems have been over for a long time in the systems they still have some issues which need to be addressed.

**Issues:**

1. Single point of Failure
2. Fault Tolerance
3. Scalability
4. Reconciliation Overhead.

**Single point of Failure:**

A single point of failure (SPOF) is a potential risk posed by a flaw in the design, implementation or configuration of a circuit or system in which one fault or malfunction causes an entire system to stop operating. In a data center environment, a single point of failure can compromise the availability of the system or the entire data center depending on the location and interdependencies involved in the failure.

For example consider a data center where a single server runs a single application. The underlying server hardware would present a single point of failure for the application’s availability. If the server failed, the application would become unstable or crash entirely. Preventing users from accessing the application, and possibly resulting in some of data loss. In normal practice use of a backup server would suffice but poses the same challenge to the second server too.

**Fault Tolerance:**

Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of some (or one or more faults within) of its components. If the operating quality for a system decreases, the decrease will be proportional to the severity of the failure. Compared to a native designed system in which even a small failure can cause total breakdown (SPOF). Fault tolerance is particularly sought after in high-availability or life-critical systems.

Currently the fault tolerance of the system is being addressed by technique redundancy or replication.

**Redundancy/ Replication:**

Redundancy is the provision of functional capabilities that would be unnecessary in a fault-free environment. This can consist of backup components that automatically "kick in" if one component fails. Two kinds of redundancy are possible space redundancy and time redundancy. Space redundancy provides additional components, functions, or data items that are unnecessary for fault-free operation. In time redundancy the computation or data transmission is repeated and the result is compared to a stored copy of the previous result.

But considering the state of art of fault tolerance research there current fault tolerant mechanisms are not the best to rely on . Techniques like state machine replication in a distributed environment can guarantee better fault tolerant systems.

**Scalability:**

Scalability is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged to accommodate that growth. For example, a system is considered scalable if it is capable of increasing its total output under an increased load when resources are added. It is a highly significant issue in electronics systems, databases, routers, and networking. A system, whose performance improves after adding hardware, proportionally to the capacity added, is said to be a ***scalable system***.

With the centralized system in place the scalability it is always a property to be addressed keeping different constraints. Presently scalability is viewed by the number of user requests, flavors of implementation and hardware. It is very hard to scale the present system as the scaling needs all the data, state of the machine and capacity of the system needed to be changed.

**Reconciliation Overhead:**

Reconciliation is an accounting process that uses two sets of records to ensure figures are correct and in agreement. It confirms whether the money leaving an account matches the amount that's been spent, and making sure the two are balanced at the end of the recording period. The purpose of reconciliation is to provide consistency and accuracy in financial accounts. Reconciliation is particularly useful for explaining the difference between two financial records or account balances. Some differences may be acceptable due to the timing of payments and deposits.

**General causes of reconciliation:**

With centralized system it could be taken granted that the general ledger will be maintained and updated by the central systems. But in practical scenarios this is not the case with businesses. There could be various reasons where the client could expect different values in their accounts than displayed. Some are discussed as below with banking sector as example.

1. Interpretation of calculation:

In general businesses are carried on by performing multiple transactions ie., exchange of information, money or services. This expects the both the parties should have understood the business processes and their calculations. This can happen in one of two ways. You might make an error when you’re recording a transaction in your general ledger. This can be as simple as transposing two numbers or recording a transaction twice, or be more serious, such as forgetting a transaction. The other way math errors occur is when both the ledger and bank statement match, but you make a mistake during your comparisons.

1. Electronic Fees:

With more and more businesses using electronic transactions, the different fees charged can cause confusion. A simple case might be like a credit card payment for $250 and record that as a $250 credit in your ledger. However, in bank statement might show the credit card processor has taken a percentage of that $250, depending on your arrangement, and deposited less than $250 into the account. There could be monthly shopping cart, secure gateway and card processing fees, one or more of which differ each month. If the interest earned on the bank account balance, the statement would have shown after a cumulative amount earned which could lead to confusion.

1. Potential Fraud:

Another reason a bank reconciliation statement might show a difference is because a stolen or lost card/credential. A person making general ledger entries might record cash transactions in your general ledger, but not deposit the money. This might be a one-time event, or occur on a daily or weekly basis, with the person recording the fraudulent entries. To protect from such kind of frauds, checking the bank balance electronically on a regular basis, ideally daily.

1. Outstanding payments/checks

Another reason bank statements and general ledgers may not agree is because a payment recorded in the general ledger might have been a check that did not clear by the bank until after statement closing date. For example, if you pay a vendor $105.08 on March 15 and the vendor does not cash the check until March 30, your bank account probably won’t be debited $105.08 in March. Your ledger will have the $105.08 payment debit recorded in March, however. If you find your numbers are off by $105.08, you might quickly find a $105.08 transaction in your ledger, but not on your bank statement. For this reason, it’s common practice to record how transactions are paid in the general ledger.

With the above issues, it is certain that there are businesses needs much more and better systems than the current centralized system to meet their requirements.

**Business Requirements:**

Different businesses could have different requirements to be met by the underlying frameworks or systems which enable them to do business. Current centralized systems even though meet the business requirements they fall short in addressing some requirements or the current state of art of the solutions implemented could be not he right matching. Below are some of the most common requirements from the community to be met by the service/framework providers.

**Requirements:**

1. Guaranteed message delivery
2. Guaranteed on results of business processes
3. Business continuity
4. End-to-end security
5. Privacy
6. Data integrity
7. Latency
8. Asynchronous processing

**Guaranteed message delivery:**

This is most common and necessary requirement for system to provide. This is mostly related to the communication trust level provided by the service. Each business process communication depends on the delivery of the message sent. And this would not expect just only delivery but the order of message deliver. With the current implementations of message delivery systems is guaranteed that the messages from the two parties are delivered as expected.

**Guaranteed on results of business processes:**

Businesses expects for a smoother and correct processes. This could also mean the actual execution of the business agreements should be as expected. Though the requirement is on results of the business processes it is more related to the terms and conditions both the parties agreed upon but technology is the one which provides platform to execute the processes. For example the different environment of executions should results in same result of the processes.

**Business continuity:**

Business should be able to continue even with the existence of systems failures. This mostly deals with the fault tolerance provided by the systems. Presently fault tolerant systems are available in the market namely the DCDR systems. But the current implementations of the fault tolerant techniques used are not very promising.

**End-to-end security:**

The current centralized system of doing transactions was not able to provide the end-to –end security. The end-to-end security can defined as the “Only the participants should be able to view the contents of a transaction”. But in central systems / decentralized systems , a transaction is performed in multiple hops. The details of the transaction is being viewed by all the participants involved in all the hops a single transaction. Making the details not only available to participants other than the intent receiver, but also not revealing the data to the actual sender and receiver. This could be viewed as a serious issues in a business transaction as the initiator (sender) of the transaction will not be aware of the status of the transaction until he/she hears from the receiver acknowledging for that specific transaction . So there is a very much requirement of a framework which provided end-to-end security

**Privacy:**

As discussed in the above requirement each transaction in the central/decentralized systems is carried out in multiple hops. At each hop the data transmitted will be viewed and perform the necessary steps. This can also be viewed as the loss of privacy of the sender and receiver of the transaction. In business community the data shared to perform a transaction could be of highly sensitive.

With the introduction of General Data Protection Regulation (GPDR), there is a huge requirement to provide services to either businesses or to individual in a private preserving ways.

**Data integrity:**

Data integrity is the assurance that information is not corrupted and can only be accessed or modified by those authorized to do so. Integrity involves maintaining the consistency, accuracy and trustworthiness of data over its entire lifecycle.

To maintain integrity, data must not be changed in transit and steps must be taken to ensure that data cannot be altered by an unauthorized person or program. Such measures include implementing user access controls and version controls to prevent erroneous changes or accidental deletion by authorized users. Other measures include the use of checksums and cryptographic checksums to verify integrity. Network administration measures to ensure data integrity include documenting system administration procedures, parameters and maintenance activities, and create disaster recovery plans for occurrences such as power outages, server failure or security attacks. If data become corrupted, backups or redundancies must be available to restore the affected data to its correct state.

**Latency:**

Latency by definition is the time lapse between two events. In the systems dealing with the transactions latency is defined according to the time taken to complete a transaction.

In the current systems latency is at acceptable level that is a transactions completion a very low but these are achieved with a trade-off between privacy. Since the central party cloud complete the transaction just by updating its state based in the input provided by the intermittent nodes the latency is low.

**Asynchronous processing:**

The capability of a system to perform transactions asynchronously is known as asynchronous processing. Current systems do provide asynchronous processing of each clients, but mostly depends on the hardware, network and the processing speed of the system . If there is an increase in the incoming requests asynchronous processing could be affected.

Promises of Blockchain

As discussed, blockchain “is as shared and open ledger that keeps a record of the transactions and cannot be modified ” . This is based on the properties of linked list data structure and the concept cryptography. With linked list the data in set the form of blocks(collection of data points/rows) are linked to one an another . In the header (meta data) , the hash of the most recent (latest) block is embedded. Here the hash function is used to maintain the sanctity of the data being added to in the new block.

Below are some of the benefits provided by the blockchain technologies.

1. Immutability
2. Transparency
3. Business Continuity
4. Privacy
5. Disintermediation
6. Consensus
7. Trust
8. Smart Contracts

Immutability:

Immutability can be defined as the property of an object of not being able to change its structure due to changes in its environment or external factors. It is difficult to obtain the perfect immutability in the current systems or in the blockchain technologies.

The immutability definition is slightly modified in the blockchain technology and defined as the data in blockchain is hard to modify and easy to verify. This means the data in the blockchain is made to be very hard to modify . This is due to the process involved in the creation of blocks and making the blockchain as append only. Append only constraint makes the participants of the blockchain to only append a new block and cannot modify the existing block.

Blockchain provides the benefit of easy to verify to check if the block is modified (incase). With the hash of each preceding block embedded in new block’s headers provides an easy technique to verify the sanctity of a block. In this way the data added to the blockchain can be considered near to immutable object.

**Transparency:**

Transparency can be defined as a property where all the parties getting involved in the transaction knows exactly what actions are being taken on what data. In public or private blockchain technologies, all the transactions performed are added to the blockchain/ledger. As the ledger is accessible by all the participants (private) transparency could be achieved. By making the ledger available on the network that is by broadcasting, the risk of cyber attacks can also be reduced.

**Business Continuity:**

Businesses are purely based on the service provided by them to its client. This makes the availability of business as one of the top requirements for any business. No matter what are the faults in the system, the system is expected to support the business continuity. Present centralized systems are prone to single point failures. Though they provide the fault torrent systems , the implementation of those software could pose some unexpected challenges like “synchronization of data” at the time of fallback .

In blockchain, data is available to each and every participant; this reduces the effort of synchronization of data at fallback time. And the systems in the blockchain are connected, so single server failures cannot affect the continuity of the service.

**Privacy:**

Privacy of the data being shared between the parties of the each transaction is a very important feature for any system. In public blockchain all the data is shared to all the members of a network. But as businesses cannot be performed on such a public platforms , private blockchain have been introduced where data being shared can be restricted to be viewed by only some participants. Platforms like hyperledger frabic , Multichain ,Quorum and Corda provide high level of privacy making suitable for businesses.

**Disintermediation**

Disintermediation is the removal of intermediaries in any transaction based systems, or cutting out the middleman in connection with a transaction or a series of transactions**.** Blockchain technologies provide a system with no intermediary to carry out transactions. Here any valid member can initiate a transaction with any member without any intermediary. Each transaction can be verified as valid or invalid transaction by all the participants based on the information in the blockchain. Thus removing the intermediaries to carry out the transaction.

**Consensus**

A consensus mechanism is a fault-tolerant mechanism that is used in computer and blockchain systems to achieve the necessary agreement on a single data value or a single state of the network among distributed processes or multi-agent systems. It is useful in record-keeping, among other things.

In any centralized system, like a database holding key information about all the transactions, a central administrator has the authority to maintain and update the database. The task of making any updates - like adding/deleting/updating is performed by a central authority who remains the sole in-charge of maintaining genuine records.

Public/private blockchains that operate as decentralized, self-regulating systems work on a global scale without any single authority. They involve contributions from hundreds of thousands of participants who work on verification and authentication of transactions occurring on the blockchain.

In such a dynamically changing status of the blockchain, these publicly shared ledgers need an efficient, fair, real-time, functional, reliable, and secure mechanism to ensure that all the transactions occurring on the network are genuine and all participants agree on a consensus on the status of the ledger. This all important task is performed by the consensus mechanism,

which is a set of rules that decides on the contributions by the various participants of the blockchain.

**Trust**

The problem blockchain is trying to solve is how to run a trusted system with trustless people. With just immutability property of blockchain alone trusted systems cannot be produced . It has to decouple from the operating team(human interference).

Blockchain and DLT ( Distributed Ledger Technology) together eliminate the vulnerability of human interference, as a system built with both technologies can operate without having to trust system administrators. There is also no hierarchy of users in a DLT system, no super-administrators or privileged accounts that can delete transactions in the system. Each transaction must be endorsed by other users.

**Smart Contracts**

A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract.

Smart contracts allow the performance of credible transactions without third parties. Contents of smart contracts claim that many kinds of contractual clauses may be made partially or fully self-executing, self-enforcing, or both. The aim of smart contracts is to provide security that is superior to traditional contract law and to reduce other transaction costs associated with contracting.

With all the assurances provided by smart contracts , the execution environment should also be trusted . Though the smart contracts assure the business processes execution, the environment where these are executed will also affect the result of the smart contracts. So there should be an agreement on the execution environment of these smart.

Existing Blockchain Technologies

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.

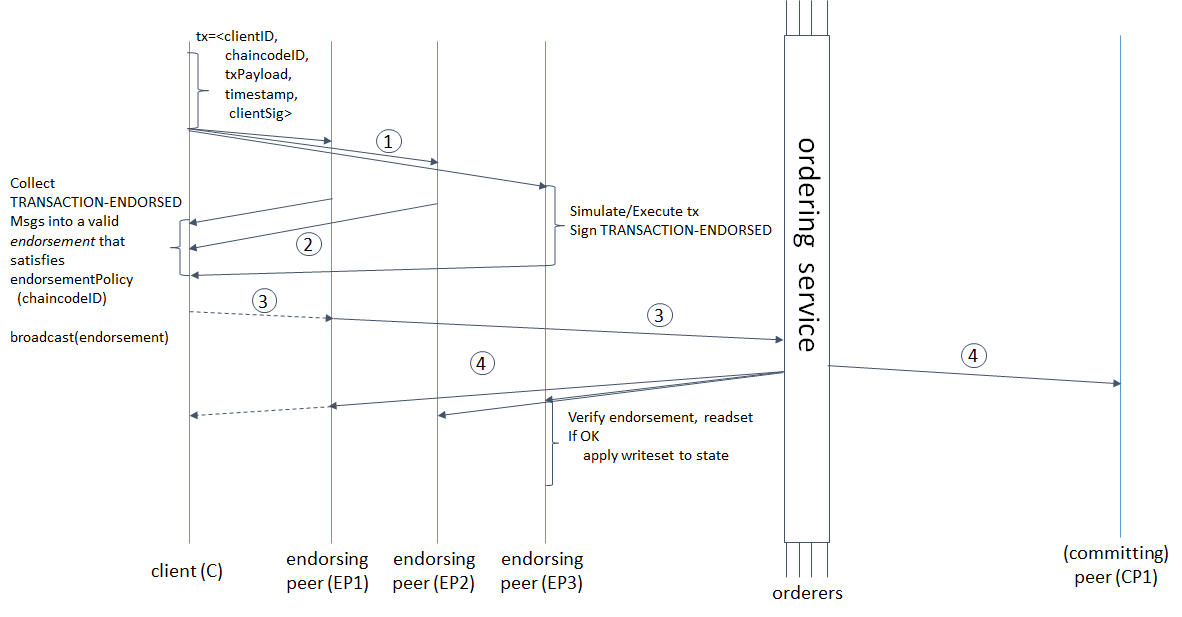
**Orderer peer:**

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.

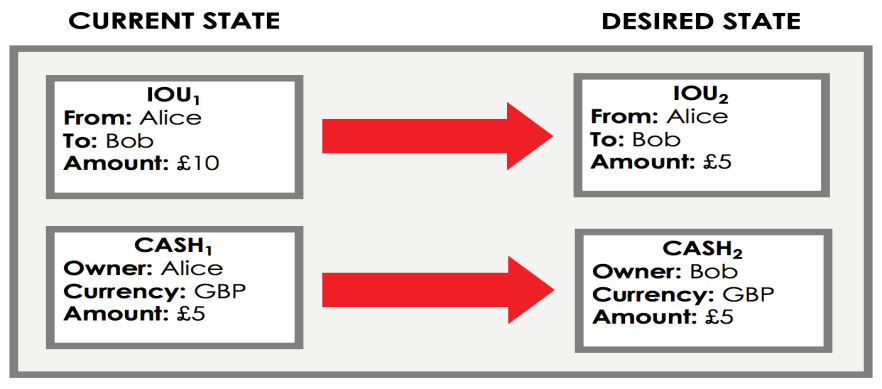
In particular, notary clusters may differ in terms of:

* **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

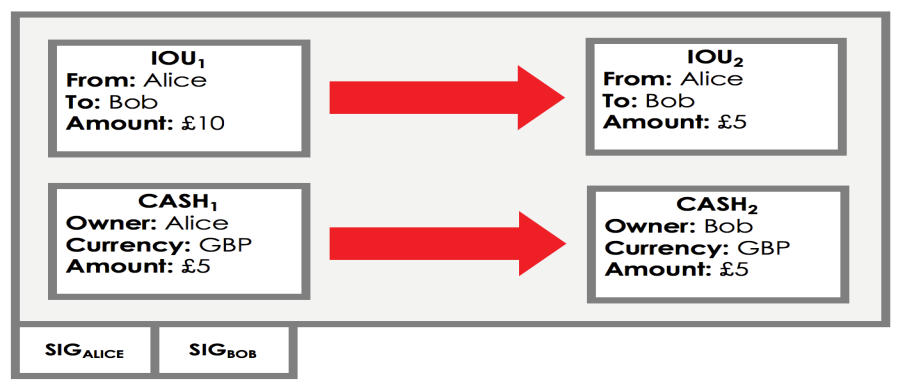
**Transaction Flow:**

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.

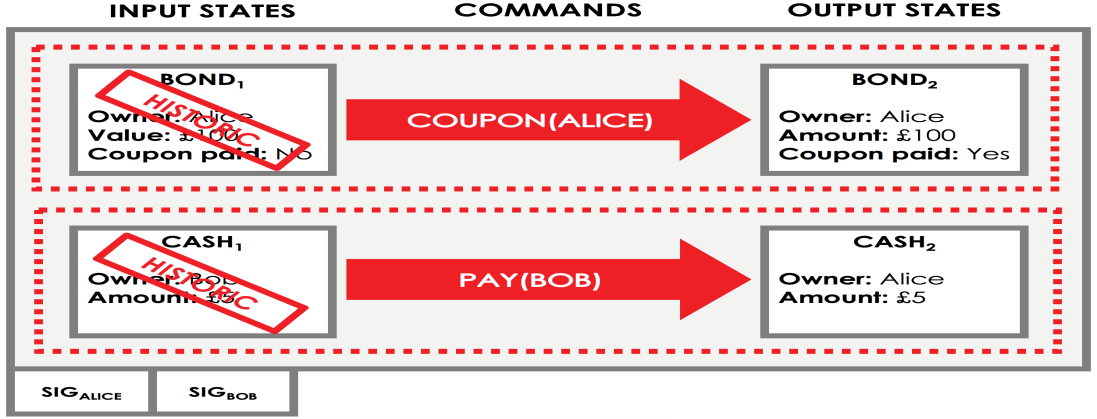
1. Both parties of an individual transaction initiates the transaction with a set of input state references that will be consumed by the final accepted transaction and their respective output references.



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.)

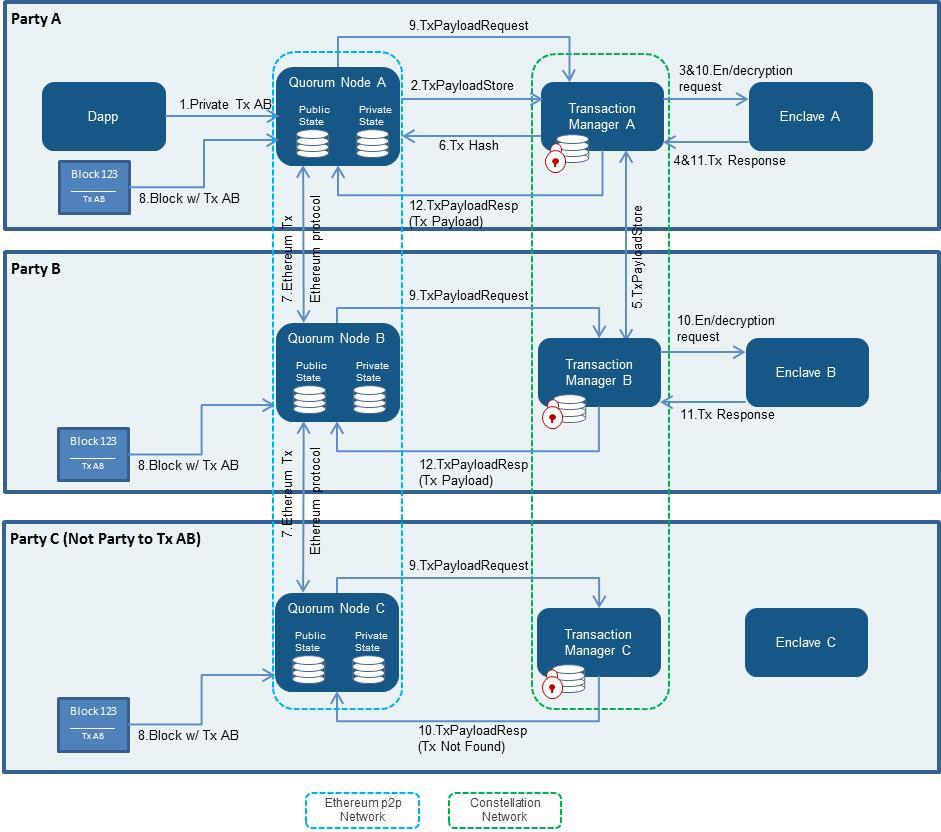
The Transaction Manager is restful/stateless and can be load balanced easily.

**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 include 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.

Proposed BCT Platform

Based on the discussion of the existing current central systems and the blockchain technologies, it is observed the there are some inevitable to trade-offs in achieving complete freedom from the third trusted party. This section deals with the proposed blockchain platform and also it comparison with current systems.

Before discussing the proposed platform, a quick overview of the trade-off is discussed in the below section.

Inevitable Trade-offs:

1. Consistency Vs Latency:

Consistency is distributed systems can only be achieved by consensus or agreement by all the parties in the systems . Unlike in central system where the its is the whole responsibility of the central administration / central server to verify the system status and maintain the status. So it can viewed that consistency is maintained by the central system.

To achieve consensus on the status of the system all the servers have to come to an agreement on the state of the system. This induces latency in the transaction execution time. Thus , consistency and latency is distributed systems is an avoidable trade –off .

1. Privacy vs Trusted flexible interoperability

Public blockchain technology allows the block created to broadcast to all the participants of the system. This could be termed as loss of privacy of the participants. To achieve privacy which is one of the basic requirements of a business, private blockchain technologies have been introduced. The main goals of these blockchain technologies are provide privacy of the data for each participant yet providing the benefits of the blockchain.

Private blockchain technologies like (hyperledger fabric, Multichain ) have achieved the privacy of each participant just by not reveling the data to all but to some participants. In hyperledger fabric data of the transaction is non-private with the channel but is private outside the channel. By this the participant’s privacy is preserved to an extend but completely. But these types of model have a trade off with the interoperability. The participants of one channel cannot do business with participants of other channel. This is an unavoidable trade-off where to achieve complete privacy there should be compromise on interoperability.

Considering the above discussed tradeoffs and the benefits of blockchain technology provided , the proposed model would be providing almost the same features of central system but in a distributed environment.

General Structure of Model:

The general structure of the proposed model is as shown in the figure. In simple terminology the model can be viewed as the collection of servers configured distributed performing a task independently. The model consists of below components

1. Consensus service
2. Data Service
3. Membership Service
4. Client / Proxy Interface

Consensus Service:

Consensus Service is a collection of systems (nodes) to execute a task/ transaction. Based on the literature of distributed systems the consensus on task could be achieved by many algorithms. Algorithms like PAXOS, PBFT, and Zyzyvza provide the basic understanding of the consensus between the distributed systems.

Here in the proposed model, Particle Byzantine Fault Tolerant (PBFT) algorithm is being implemented to provide consensus service. Communication between the nodes is achieved using the message passing technique. And each server in the system makes progress by state machine replication concept.

State Machine Replication (SMR):

SMR is a technique used to maintain the state of the machine using replication of state. A state of the machine is defined by the number of transactions / tasks performed by each node. State of the machine is changed only when the there is an input which either creates an object or updates an object.

A State Machine begins at the State labeled Start. Each Input received is passed through the transition and output function to produce a new State and an Output. The State is held stable until a new Input is received, while the Output is communicated to the appropriate receiver.

State Machine of a system should be deterministic that is if multiple copies of the same State Machine begin in the Start state, and on receiving the same Inputs in the same order should arrive at the same State having generated the same Outputs. The determinism of the system provides fault-tolerance. That is if there are multiple copies of a system exist, a fault in one would be noticeable as a difference in the State or Output from the others.

The PBFT algorithm is being used in this proposed model to handle any byzantine behavior of the nodes. Consensus server upon receving valid request from a client performs PBFT agreement protocol to get consensus on execution of the request. Based on the configuration of the system, which is discussed in later sections the response is being generated. After successfully obtaining the consensus on a request each independent node updates its current state to the next valid state. After N number of transactions, all the transactions in chronological order will added to create a block and saved. Using PBFT and SMR consensus server which can tolerate byzantine fault tolerant system is implemented and tested.

Membership Service:

Membership service is designed to address privacy in the proposed model. Membership service by the name provides registration and authentication service to the participants. This will provide the participants to know all the participants they are interacting with. Public and private key system is being used to authenticate and also to digitally sign each transaction a participant involves in.

Unlike in other blockchain platforms where membership service provides a way to register to a particular channel or quorum , here the membership service will only provide onboarding into complete system . But the clients should fetch the public of each participant to perform any transaction. This makes the clients perform transaction with any participant in the system.

Data Service:

Data Service is basically to store and retrieve data produced by the system. Data in the proposed system is either transaction information or a block. All transactions in the system are tracked in the data service component. Clients can query the data service for the data required.

Consensus service upon executing set transactions will create a block which gets added to the blockchain. After successful creation of the blocks consensus service updates the blocks data using the data service functionalities.

Client/Proxy Interface:

Client/Proxy interface provides the participants to interact with the proposed system. Below are current functionalities provided by the interface.

* Registration and login:

Registration of a participant involves in the interacting with the membership service. Registration process involves in creation of public and private keys of each participant. The client interface is used to store public key for future purposes (digital signature).

* Collect Responses from Consensus Service:

As discussed consensus server is a collection of nodes running independently . After a successful execution of a request each node in the consensus server sends out response to the clients. Client/proxy interface collects all the matching responses to a request and it informs the client only after receiving certain number of responses.

Currently the proposed system can be configured in three levels. The levels are based on the data that clients are ready to share with the system. Each level of systems expects more amount data and also provides more benefits as per the data shared.

Levels of System:

1. Message Level Consensus
2. Business Process Level Consensus
3. State Level Consensus

Message Level Consensus:

Business transaction can be viewed the exchange of information with the concerned parties. Each transaction contains specific data which needs to be processesd with a specific defined set of instructions. The set of instructions are steps agreed upon by each party to modify the data shared in the transaction . At the first level of the system participants can share only the intent of the transaction to be carried out . In formal language the data shared at this level could be the initiation of a transaction . The data could be just sender , receiver and the message. Consensus server upon receiving the data will perform the consensus algorithm and respond with a transaction id only. Sender on receiving the transaction id can transfer the actual data to the receiver where the execution of the transaction can take place. Consensus server will not be having any information on the status of the execution status of the transaction. Thus only the information of intiator and receiver of the transaction will be added to the blockchain . The transaction flow is shown in the below figure.

1. Request - (msg)
2. Response – (msg,seqNo)
3. Block - set of (msg,seqNo)

CS- Consensus server , C1,C2 – Participants

Guaranteed features:

At this level the system can provide some basic features guaranteed. The below table provides an overview of the guaranteed requirements (discussed earlier)

|  |  |  |
| --- | --- | --- |
|  | **Ideal** | **MLC** |
| **M** | High | High |
| **R** | High | Low |
| **B** | High | High |
| **E** | High | Low |
| **P** | High | High |
| **D** | High | High |
| **L** | Low | Medium |
| **A** | High | High |

Gaps:

From the table is clear that the message level systems have some gaps compared to an ideal system.

1. Guaranteed on business results:

As the data shared at this level is only the intent of the transaction and nothing more , consensus server cannot guarantee on the transaction execution status.

1. End- to- End Security:

Since the transfer on data is handled by the respective parties , the system has not control the flow of transaction data. To complete a transaction the data could be transferred in hops / direct depending on the parties.

1. Latency:

As the response to a request to the consensus server needs to go through the agreement algorithm and also due the network latency added , the latency of each request will be little higher than that on an ideal system.

Business Process Level Consensus:

At this level the clients can share the data of the transaction and also the business processes id . Prior to the acceptance of the transaction , the system would be configured on the sequence of steps involved in each business processes . If a participant wants to perform a transaction , it can share the message and also business processes id by which the consensus server can provide the details execution at each level.

Business processes can be viewed as the sequence of steps to be followed to complete a transaction. On sending the request to the system , consensus server respond to the participant with the sequence number , business process id and the completed step number to all the participants of the transaction .

Input : (msg , BPid, stepNo)

Output : ((msg,seqNo)(BPid, stepNo))

Block :set of ((msg,seqNo)(BPid, stepNo))

The above figure shows the transaction flow in the business processes level of the system .

Guaranteed features :

At this level the system provides some guaranteed features as show in the below table.

|  |  |  |
| --- | --- | --- |
|  | **Ideal** | **BLC** |
| **M** | High | High |
| **R** | High | Low |
| **B** | High | High |
| **E** | High | Medium |
| **P** | High | High |
| **D** | High | High |
| **L** | Low | Medium |
| **A** | High | High |

Gaps:

From the table is clear that the business level systems have some gaps compared to an ideal system.

1. Guaranteed on business results:

As the data shared at this level is only the intent of the transaction and the business process, consensus server cannot guarantee on the transaction execution status but guarantees on the steps of the execution.

1. Latency:

As the response to a request to the consensus server needs to go through the agreement algorithm and also due the network latency added , the latency of each request will be little higher than that on an ideal system.

When compared to the earlier message level consensus systems the business level consensus provide a better guarantee on the end-to-end security of a transaction. Since the business process is being shared by the participants , consensus could be performed on the steps involved in the processes. If there is a discontinuity in the steps of execution the consensus server can be drop that request. Thus, providing the end-to-end security for a transaction.

State Level Consensus:

The third level of system is based on the data shared to the system. Data shared to the system cloud be actual data objects of a transaction, business process id and step number which is same as in the case of business process level but here the consensus server is responsible for the execution of the transaction.

At setup stage the system would be loaded with the functions to be executed for a particular business process. Upon receiving the request the consensus server performs the agreement algorithm on the state of the objects being shared in the transaction data. Once the consensus is achieved the server executes the business process step and responds to all the parties of that transaction with the updated step. The server proceeds only on receiving the request from the next party to execute the next step . Thus consensus server makes sure that each party would be aware of the transaction status and also will be executing the transaction in sequence of steps provided. The transaction flow can be viewed in the below figure.

Input : (msg , BPid, stepNo)

Output : ((msg,seqNo)(BPid, stepNo))

Block :set of ((msg,seqNo)(BPid, stepNo),State)

Guaranteed features :

The state level of consensus system provides almost all the necessary requirements for an ideal system. This is due to the data shared includes the functions to be executed. This makes the execution of all the steps in the same agreed environment provided by the system.

Gaps:

The only gap in the state level consensus to the ideal system is that the latency in execution of a system. As discussed in the inevitable trade-offs section, consistency and latency cannot be achieved at same time. Here the consensus nodes have to achieve an agreement on the state of the objects of a transaction which would include network latency also. This adds some time lag in the complete execution of a transaction.

|  |  |  |
| --- | --- | --- |
|  | **Ideal** | **SLC** |
| **M** | High | High |
| **R** | High | High |
| **B** | High | High |
| **E** | High | High |
| **P** | High | High |
| **D** | High | High |
| **L** | Low | Medium |
| **A** | High | High |