Blockchain for B2B Contracts
(A view from Uncontrolled Shipments)

BITS ZG628T: Dissertation

by

Sudhindra. C

2015HT13101

Dissertation work carried out at

SAP Labs India Pvt. Ltd. Bangalore

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)

October 2017

Blockchain for B2B Contracts
(A view from Uncontrolled Shipments)

BITS ZG628T: Dissertation

by

Sudhindra. C

2015HT13101

Dissertation work carried out at

SAP Labs India Pvt. Ltd. Bangalore

Submitted in partial fulfillment of M.Tech. Software Systems degree programme

Under the supervision of
Karthikeyan Loganathan,
SAP Labs India Pvt. Ltd. Bangalore

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE
PILANI (RAJASTHAN)

October 2017

Academic Area of Work: Data Storage Techniques and Networks

CERTIFICATE

This is to certify that the Dissertation entitled Blockchain for B2B contracts (A view from Uncontrolled Shipments) and submitted by Sudhindra. C having ID-No. 2015HT13101 for the partial fulfillment of the requirements of M.Tech. Software Systems degree of BITS, embodies the bonafide work done by him under my supervision.

_____
Signature of the Supervisor

Karthikeyan Loganathan
Chief Development Architect
SAP Labs India Pvt. Ltd. Bangalore

Place : Bangalore

Date : 17-10-2017

_____
Name, Designation & Organization & Location

Birla Institute of Technology & Science, Pilani

Work-Integrated Learning Programmes Division

First Semester 2017-2018

BITS ZG628T: Dissertation

ABSTRACT

BITS ID No. : **2015HT13101**

NAME OF THE STUDENT : **Sudhindra. C**

EMAIL ADDRESS : **sudhindra.chandrashekar@sap.com**

STUDENT'S EMPLOYING ORGANIZATION & LOCATION : **SAP Labs India Pvt. Ltd. Bangalore**

SUPERVISOR'S NAME : **Karthikeyan Loganathan**

SUPERVISOR'S EMPLOYING ORGANIZATION & LOCATION : **SAP Labs India Pvt. Ltd. Bangalore**

SUPERVISOR'S EMAIL ADDRESS: **karthikeyan.loganathan@sap.com**

DISSERTATION TITLE : **Blockchain for B2B contracts (A view from Uncontrolled Shipments**

ABSTRACT:

A Blockchain network can be visualized as an open, public and an anonymous distributed ledger network. Blockchain is the framework/operating system on which Bitcoin operates. Bitcoin is one of the many transactional systems which runs on Blockchain network.

The Linux foundation's Hyperledger fabric is the Blockchain that is built for businesses. Hyperledger Fabric proves to be:

- **Cost effective** as they increase the speed of the transactions and reduce the overhead costs.
- **Highly efficient** as the transaction is recorded only once and is visible to all parties through the distributed network.
- **Tamper Evident** as a transaction recorded in a Blockchain network cannot be changed and it can only be reversed with another transaction and these transactions are recorded over a continuously growing list of records called blocks. These blocks are linked to each other securely using cryptography

The fundamental difference between Hyperledger Fabric (Blockchain for business) and Bitcoin network is that, Hyperledger Fabric is a closed network which has only the people of the organization transacting on the network, and the network on which Bitcoin runs is an anonymous network. Hence in the Bitcoin network it takes immense computing power to maintain the safety and integrity of the transactions transacted. Since Hyperledger Fabric network is a private network it is much more cost effective whilst still having all the security aspects of what the Blockchain network offers.

This project focuses on how Hyperledger Fabric can be used to store the B2B contracts of SAP TM. Especially in an Uncontrolled shipment scenario these B2B contracts are triparty contracts. These contracts are subject to changes and when multiple parties are involved, it becomes hard to keep the changes transparent.

The results show that Hyperledger Fabric can store intangible assets like the B2B contracts, of SAP TM, in a private network and the changes to the contract is distributed to all the participants of the network, which runs on smart contracts. Smart contracts are business logic embedded in the network which helps in interacting with the blocks of the blockchain. It is also possible to endorse these transactions by certain parties in the network which are called as endorsement policies.

The results also show that, with the parties of the Blockchain network have access to the latest version of the contract and this makes the transactions, which consume these contracts, error free. This makes the overall business processes run smoother.

Broad Academic Area of Work: Data Storage Techniques and Networks

Key words: Blockchain, Cloud Foundry, Docker, Go Lang, Java, REST API

_____	_____
Signature of the Student	**Signature of the Supervisor**
Name: _____	**Name: _____**
Date: 17/10/2017	**Date: 17/10/2017**
Place: Bangalore	**Place: Bangalore**

Acknowledgements

I would like to thank the following people without whom this dissertation would have been a daunting task.

- My mentor ***Mr. Karthikeyan Loganathan***. Your immense expertise and experience is a true motivation. The guidance you provided was very helpful
- My additional examiner ***Mr. Pushpendu Sarkar***. Your sound business acumen helped me to understand the business processes and the pain points
- My manager ***Mr. Mayank Bhatnagar*** for giving me the opportunity to pursue my higher studies
- ***Mr. Raghavendra Rajarao Deshmukh*** for your suggestions and ideas on the technical aspects of this dissertation
- ***Mr. Nitin Verma***. Your sound technical knowledge in Blockchain is commendable. Thank you for the initial and continuous support. You were my natural choice when I was wedged with a problem

Contents

List of Figures

List of Tables

1. Introduction:

1.1 Objective

Use Blockchain to store the B2B contracts in SAP TM, which are created for Uncontrolled Shipments, to bring transparency and trust among the organizations and increase the efficiency of the B2B processes between the organizations.

1.2 Business Domain

The transportation industry is a service industry that is driven by contracts. These are B2B contracts that exists between various business partners who are stakeholders in the logistics process. These contracts can have different variants (Quotation, Long term, Spot Quotation, volume/value contracts) and are an integral part of this industry, which define and shape the billions of transportation transactions daily.

To explain the logistics process, it is necessary to introduce some business partner roles:

- Shipper/Ordering Party: A business partner role who manufactures goods and ships the goods to the various locations
- Carrier: A business partner role who physically moves the goods using the transportation resources (trucks, barges, cargo aircraft, etc.)
- Logistics Service Provider (also referred as LSP): A business partner role who performs the role of an aggregator. An LSP usually co-ordinates the planning and the execution of the transportation. They might or might not own the transportation resources and might further subcontract the transportation to the carrier.

When the shipper wants to move the goods that they manufacture they either:

- Get in touch with the carrier for the movement of goods. This is the case where the shipper performs the planning on their own. In this case the B2B contract exists between the shipper and the carrier.
- Get in touch with the LSP for the movement of goods. This is the case where the shipper does not perform the planning or they give Less than Truck Load (LTL) shipment or Less than Container Load (LCL) shipments. In this case the B2B contract exists between the shipper and the LSP.

The above scenarios are Controlled Shipment processes where the shipper pays the carrier or the LSP for the transportation service they have provided.

Uncontrolled Shipment: In this case the shipper has a contract with the LSP and a contract with the carrier. The shipper directs the LSP to use the carrier, with whom the shipper has the contract, to execute the transportation. The LSP uses the contract between the shipper and the carrier to calculate the freight costs.

1.3 Existing system and Problem statement

In these circumstances the LSP cannot access the contract that exists between the shipper and the carrier as this is not in the LSP's landscape. This contract exists in a system to which the LSP will not have direct access. To circumvent the problem the traditional approach is to maintain a copy of the contract in the LSP system. This approach has the following issues:

- Reduced transparency
- Disputes between the LSP and the shipper as the copy of the contract that the LSP maintains might be out of date
- Invoice verification process becomes cumbersome
- Manual effort for the LSP to manage the changes to this contract

1.4 High Level Requirements

Given the above problem statement, we try to answer some of the following business questions:

- How can we make the process of such freight rate management completely transparent so that every stake holder is completely aware of its evolution?
- How can we make the process hack-proof such that every stakeholder can only participate in the process as per his or her defined roles (E.g. Shipper and carrier can propose a rate change, LSP can use the contract but can't modify it etc.)?
- How do we get rid of duplicate maintenance of such agreements?
- Can we generalize this process where even standard shipper-carrier agreements can be included?

2. Literature Review

2.1 Blockchain

Blockchain is a distributed ledger that keeps the transactions in the form of blocks. Blockchain is the framework for Bitcoin, the famous cryptocurrency. In the case of Bitcoin, the Blockchain network is public network where anyone can participate. Blockchain network works based on a Peer to Peer distributed time stamp server. Blockchain ensures that there is no modification of the previous records. Every node/peer in the Blockchain keeps a complete copy of the data that has been validated by itself. When a transaction is placed on the Blockchain network the distribution of the transaction data is guaranteed by following a series of consensus or endorsement rules. When the data in the peers of the blockchain network have the same block chains, it can be concluded that the peers are in consensus.

2.2 Data storage techniques

There are many ways to store and access data. For this report, we will focus on the following categories:

2.2.1 Centralization of Data

Client server model is a distributed application model that partitions the tasks or workloads between the resource providers which are called as servers. When the request for data is placed to the client then the clients connects to the server, using different protocols to get the data and serve the request. The client server model can be represented by the figure below:

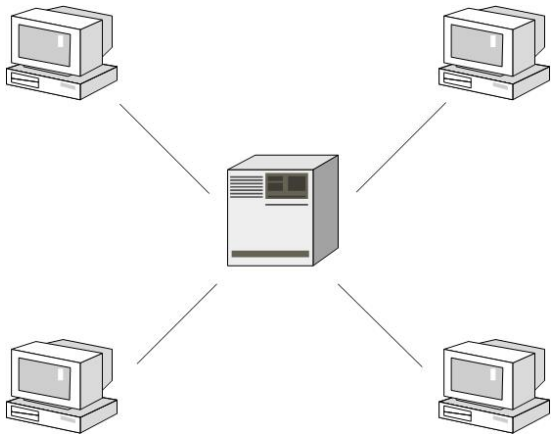

Figure 1: Client Server Model

2.2.2 Decentralization of Data

A peer to peer model is a distributed application model that partitions the tasks or workloads between the peers of the network. In this scenario, the peer holds a copy of the complete data. When a request is placed on the client/peer for the data the peer can answer to the request independently. In a P2P there needs to be a mechanism by which the data is shared between the different peers of the network. There are also fault tolerance mechanisms in

place to ensure that the data is consistently shared across the network. A typical P2P network can be depicted via the diagram below:

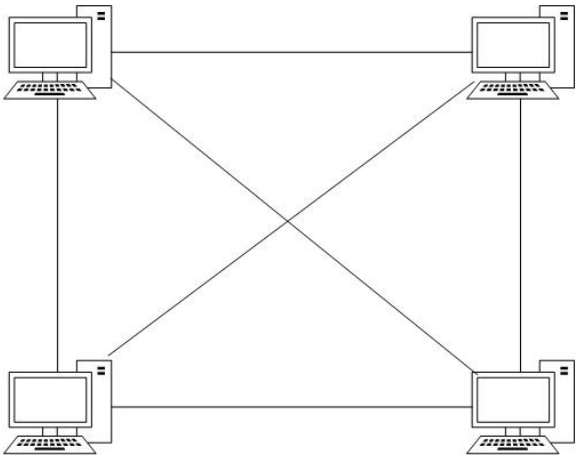

Figure 2: A Peer to Peer network

With crypto currencies, the concept of decentralization and decentralization of the ledgers emerged. We can now see that these concepts can not only be used for crypto currencies but also to store any kind asset on the decentralized ledger. We also see the decentralization being defined as "We stand at the edge of a new digital revolution. The Internet is beginning a new phase of decentralization" [1].

Prior to the invention of Blockchain the way to store the assets mentioned above were managed by a centralized and a competent authority like a bank. Centralized businesses were in the center of production, aggregation, distribution of resources and services [2]. However, with the introduction of blockchain, smart contracts and digital currencies, some practitioners suggest that the world of commerce and finance may soon be re-invented [2].

2.3 Consensus Mechanisms

As mentioned above in a P2P the peers/nodes hold the complete set of data. When there is an update to the data in one peer, then that data needs to be distributed to all the peers in the network. This mechanism is called peer to peer distribution. In Blockchain the mechanism of updating ledger state to the peers is facilitated by the Consensus mechanism. The consensus mechanism/protocol has three main properties [3]:

- Safety: The consensus protocol ensures that the output produced by one node is consistent across all the other nodes' output. This is also referred as the consistency of the shared state
- Liveness: The consensus protocol guarantees that all the non-faulty nodes which are part of the network eventually produce a value. It guarantees eventual consistency
- Fault tolerance: If there are any failed nodes in the network the consensus protocol ensure fault tolerance mechanisms

The FLP Impossibility Result, by Fischer, Lynch and Paterson [4], states that no deterministic consensus protocol can guarantee safety, liveness and fault tolerance in an asynchronous system. While fault tolerance is crucial for globally distributed networks to operate, distributed systems tend to choose between safety and liveness depending on their system requirements and assumptions.

There are two types of faults in the network:

1. Fail-stop faults cause the peers to stop participating in the network. These are also benign faults which are caused due to hardware or software faults and the node just stops responding
2. Byzantine faults, which cause the nodes to behave erratically. This category of faults was identified and characterized by Leslie Lamport as the Byzantine General's Problem [5]. Byzantine faults occur due to software bugs which cause the nodes to behave unreliably

Some of the Consensus models are listed below:

2.3.1 Proof of Work

With every transaction placed in the network the legitimacy of the transaction is proven by solving a mathematical puzzle associated with that transaction. This is called the Proof of work or also called as mining. A reward is given to the to the first miner who solves the puzzle in the form of cryptocurrency who would also validate the transaction and once validated the transaction is placed in the public network. Since this method involves in solving a complex mathematical puzzle there is a lot of computing power that goes into this process. Here the people who solve this puzzle are called miners.

2.3.2 Proof of Stake

Proof of Stake is a different algorithm which chooses the originator of the block or transaction in a deterministic way. The parameters that are used to decide the above can be depending on its wealth/stake. In this case the validation of the transaction is done by the peers but there is no block reward but there is a transaction fee. In this case the miners are called as forgers. This method of validating the transaction and placing it on the public network is energy saving.

2.3.3 Proof of Elapsed Time

This was initially proposed by Intel and is now officially under the Linux foundation's Hyperledger Fabric. This method employs a random leader election to finalize the block. The random election model chooses the leader among the available peers and this is also the way to deal with the un-trusted/unavailable nodes. For this model to work the algorithm must distribute the leader election among all the available peers of the network and requires a secure way to ensure the selection of the leader was correct and was not manipulated.

This is intended to run in a Trusted Execution Environment (TEE), such as Intel's Software Guard Extensions (SGX). The leader election works in a way that every validating peer request for a wait time from the code running in the TEE. The validator/peer with the shortest wait time wins the lottery and is chosen as the leader which validates the transaction.

3. Solution Approaches

The above problem statement/requirements can be realized via the following solution approaches:

3.1 Centralized Database

The B2B contract is created and maintained in a central database and all the business partners have access to this database.

The advantages and disadvantages of having a centralized database can be listed as follows:
Advantages:

- Confidentiality: With financial transactions being one of the front runners for blockchain there is always a need for confidentiality of data. This can be best achieved with a centralized database
- Performance: In a centralized database, the transactions need to be committed only once which is beneficial from a performance point of view

Disadvantages:

- A centralized database must be managed by a central administrator and there needs to be a centralized application logic to enforce business constraints
- The transactions should be validated by a competent authority
- Failsafe and robustness of a central database is often questionable
- Data retrieval at the peers/clients is often performance intensive as the centralized database might get a lot of requests to serve

3.2 Distributed Database

B2B contracts stored and maintained in a distributed database system. The different database nodes communicate with each other and the transactions between them are validated by nodes/peers in the distributed network. In other words, use Blockchain to store the versions of the B2B contract.

The second solution approach has the following advantages and disadvantages:
Advantages:

- Blockchain removes the necessity of a competent authority to validate the transactions. This is a core value of blockchain which enables a database to be shared across boundaries of trust, without requiring a central administrator
- Having a single competent authority like the government, bank, trade associations etc. attract a great deal of time and money which can be saved using blockchain
- Blockchain or a distributed ledger is known for its robustness and it employs fault tolerant mechanisms to ensure transaction completeness

Disadvantages:

- Performance: Usually the transactions carried out in a distributed network are slower than a centralized database as the transactions must be committed to all the databases

3.3 Decision on the solution approach

After an examination of the above solution approaches and the limitations posed by each solution approach it is beneficial to use a distributed ledger that can record transactions efficiently and securely. This is exactly what "Blockchain" provides. The solution proposal can be pictorially represented as below:

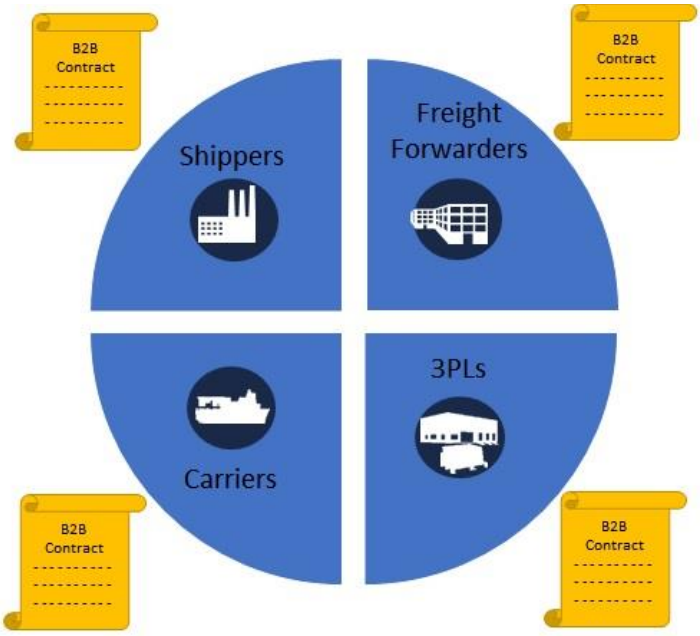

Figure 3: Decentralized view of B2B Contracts

There are multiple implementations of Blockchain framework like:

- Hyperledger Fabric
- Multichain
- Corda

We will use Hyperledger Fabric for the above-mentioned use case.

Hyperledger Fabric is a platform for distributed ledger solutions, underpinned by a modular architecture delivering high degrees of confidentiality, resiliency, flexibility and scalability. It is designed to support pluggable implementations of different components, and accommodate the complexity and intricacies that exist across the economic ecosystem.

Hyperledger Fabric delivers a uniquely elastic and extensible architecture, distinguishing it from alternative blockchain solutions. Planning for the future of enterprise blockchain requires building on top of a fully-vetted, open source architecture.

4. Hyperledger Fabric [5]

This chapter introduces the overview and terminology of Hyperledger Fabric

4.1 Hyperledger Terminology

4.2.1 Blockchain Network

A blockchain network is a network of organizations which share a common ledger. These organizations can share either a tangible (house, car, etc.) or an intangible (contracts, patents). The network must contain at least on peer who is responsible for endorsing and committing a transaction. This is also called as an ordering service. The network also contains a membership services component which is responsible for distributing and revoking the cryptographic material for the user identities and the permissions.

4.2.2 Peer

Peer is an artifact that maintains a ledger of transactions. The peer is always a committer but not necessarily an endorser. Peers do not play a role in the ordering of the transactions that are placed on the network.

4.2.3 Ordering Service

Ordering service is a component that orders transaction in a block. The ordering service is responsible to distribute the ledger state to the different peers of the network. This can either be a "solo" orderer, or Kafka for fault tolerance, or sBFT/PBFT for Byzantine fault tolerance. We can also have our own protocol to distribute the data to the network and plug to Hyperledger.

4.2.4 Channel

A channel is a subset of a broader blockchain network. A channel contains a collection of orderers and peers. The transactions placed in the channel are local to the channel and are not visible outside the channel. A peer can be part of multiple channels however they have the visibility of the transactions in the subscribed channels. Each channel has a unique ledger with which the confidentiality is ensured.

4.2.5 Genesis Block

A genesis block is a configuration block that initializes a blockchain network or a channel. The genesis block is also the first block on a chain.

4.2.6 Ledger

A ledger is an append-only transaction log maintained by the peers. The ledger has two meanings: peer and validated. The peer ledger contains the transactions that have originated from the ordering service. These transactions may not be valid as it might not have gone through the consensus logic. The validated ledger contains all the endorsed and validated transaction blocks. The transactions in the validated peer would have been endorsed, ordered and validated.

4.2.7 Endorsement Policy

Before any transaction is written to the ledger it must perform a transaction proposal on the peer/s. When this is done, the transactions must be endorsed by a minimum number of endorsing peers. The number of endorsing peers are defined during the instantiation of the chaincode application. Policies can be defined per application and can also be tuned based on the desired flexibility against unavailability (deliberate or not) of the peers.

4.2.8 Membership Service Provider (MSP)

The MSP is a component of the system that provides credentials to the clients and peers of the network. Clients use these credentials to authenticate themselves on the network and to submit transactions. The peers also use these credentials to authenticate the transactions. Since this component is a separate component in the Hyperledger network, alternate implementations can easily be plugged into Hyperledger Fabric.

4.2.9 Membership Services

Membership services or also called as the Fabric-ca is the certificate authority in the Hyperledger network. The Fabric-ca contains the client and server that handles the distribution and revocation of the cryptographic material, required for the enrollment of users in the network. The `MembershipSrvc` code (MSP) runs on the peer, and is used by the peer to authenticate users on the network.
Membership Services provides the enrollment artifacts by combining the elements of the Public Key Infrastructure (PKI) and consensus. Hyperledger Fabric being a permissioned blockchain mandates the entities to register themselves for long-term identity credentials, also called as Enrollment Certificates. These certificates are also entity specific. For users, the Enrollment Certificate authorizes the Transaction Certificate Authority (TCA) to issue pseudonymous credentials. These certificates authorize the transactions submitted by the user. These certificates are also stored on the blockchain, which help to identify the transactions.

4.3.0 Chaincode (Smart Contracts)

Chaincode is the business logic that is embedded in the blockchain that defines the rules of the transactions. Developers write chaincode applications, which are deployed into a chain by an authorized user, usually by the peer administrator. These chaincode applications are invoked by end users via a client side application that connects to the peer. These chaincode applications implement a predefined interface which gives the ability for the applications to append the transaction data to the shared ledger.

4.3.1 Hyperledger Fabric Client SDK

A client SDK provides a powerful set of methods to expose the capabilities and functionalities of Hyperledger Fabric. Some of the examples are Enrollment of the user, Re-Enrollment of the user, which directly interacts with the Fabric-ca to issue certificates and key pair to the user to transact on the blockchain network. The client SDK also provides capabilities to submit transaction proposals by setting the appropriate user context. Fabric SDK has support for Node.js, Java and Python thus allowing developers to write client applications in any of the above-mentioned programming languages.

5. Solution Architecture

5.1 Solution Block Diagram

With Hyperledger Fabric as the solution to store the B2B contracts in the Blockchain network the block diagram can be depicted as below:

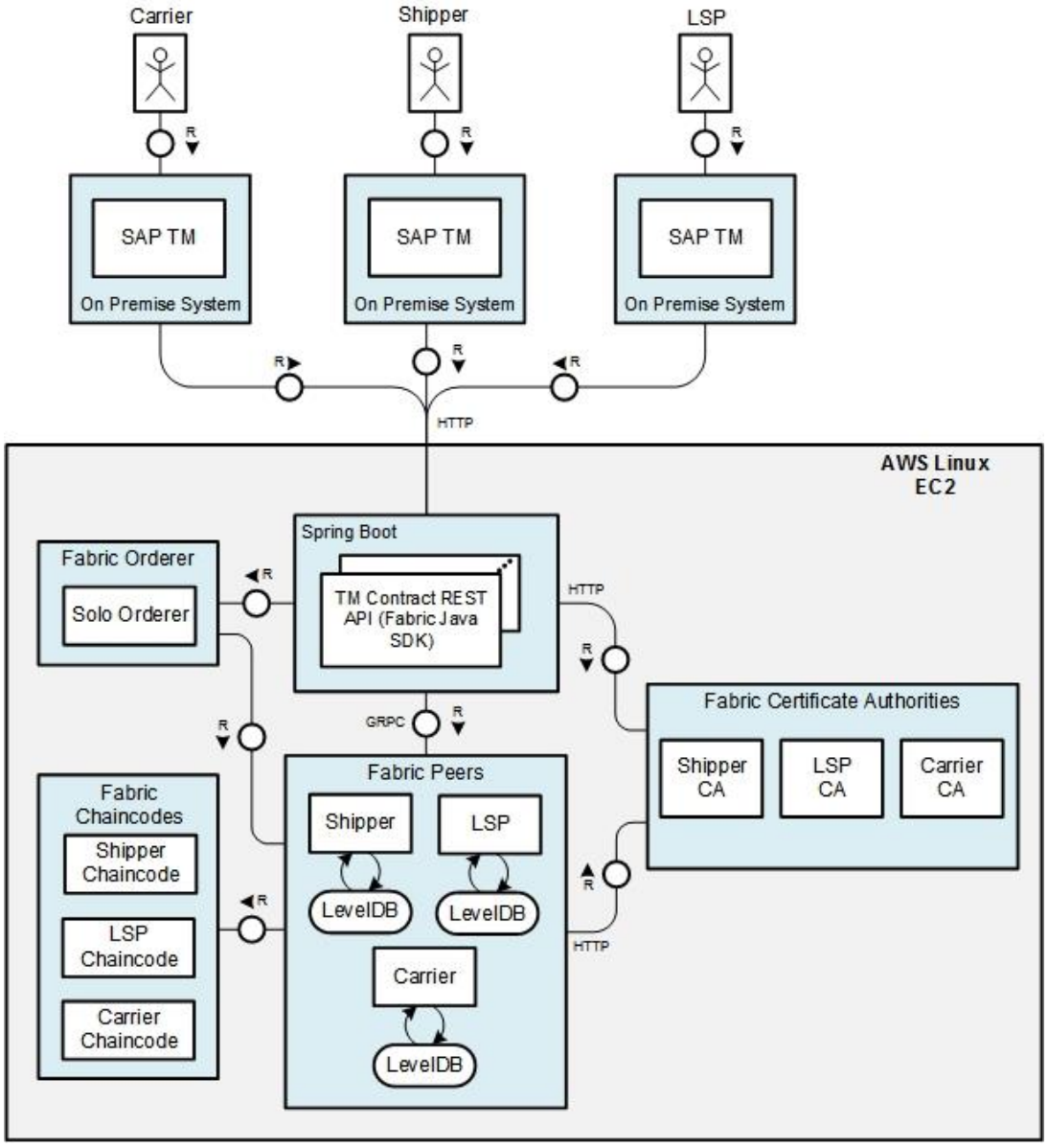

Figure 4: Solution Block Diagram

In the above architecture, we can see the different SAP TM systems owned by the shipper, carrier and LSP. These systems interact with the TM Contract API provisioned as a Spring Boot Application deployed on an AWS EC2 instance. This TM contract API uses the Hyperledger Fabric Java SDK to perform transaction proposals on the Blockchain network. This facilitates CRUD operations on the ledger.

5.2 Blockchain Network Configuration

Hyperledger provides Docker Images which runs various peers/nodes as Docker containers. With the usage of Docker Compose Topology it is possible to run multiple Docker containers via a simple configuration file.

The different entities of the Blockchain network can be represented via the tables below:

Table 1: Network Entities

Organization	Domain	Entity Type	MSP ID
Orderer	orderer.saptm.com	Orderer	OrdererMSP
ShipperOrg	peer0.shipperorg.saptm.com	Peer	ShipperOrgMSP
CarrierOrg	peer0.carrierorg.saptm.com	Peer	CarrierOrgMSP
LSPOrg	peer0.lsporg.saptm.com	Peer	LSPOrgMSP

Once we run the above network with the Hyperledger Fabric delivered Docker images we will have the following Docker containers running

Table 2: Docker Containers

Container Name	Image Type	Protocol	Port
orderer.saptm.com	fabric-orderer	GRPC	7050
peer0.shipperorg.saptm.com	fabric-peer	GRPC	7051
ca_ShipperOrg	fabric-ca	HTTP	7054
peer0.carrierorg.saptm.com	fabric-peer	GRPC	8051
ca_CarrierOrg	fabric-ca	HTTP	8054
peer0.lsporg.saptm.com	fabric-peer	GRPC	9051
ca_LSPOrg	fabric-ca	HTTP	9054
CLI (Command Line Interface)	CLI	NA	NA

The Shipper, Carrier and the LSP Organizations are part of a consortium called the "SAPTMConsortium" and form a channel, called as "saptmchannel", in Hyperledger. These organizations have a single peer within them for the integration tests.

With the above we can simulate a channel with three organizations: ShippperOrg, CarrierOrg and an LSPOrg. Every organization has a CA (Certificate Authority) which is responsible to issue signed certificates and credentials for users who are part of these organizations. For testing purposes the orderer is configured as a "solo" orderer which is a simple set up. This orderer can be replaced with a Kafka ordering service, which has more features in terms of fault tolerance.

5.3 Activity Diagram

The overall activity diagram of the system can be depicted via the Activity diagram below:

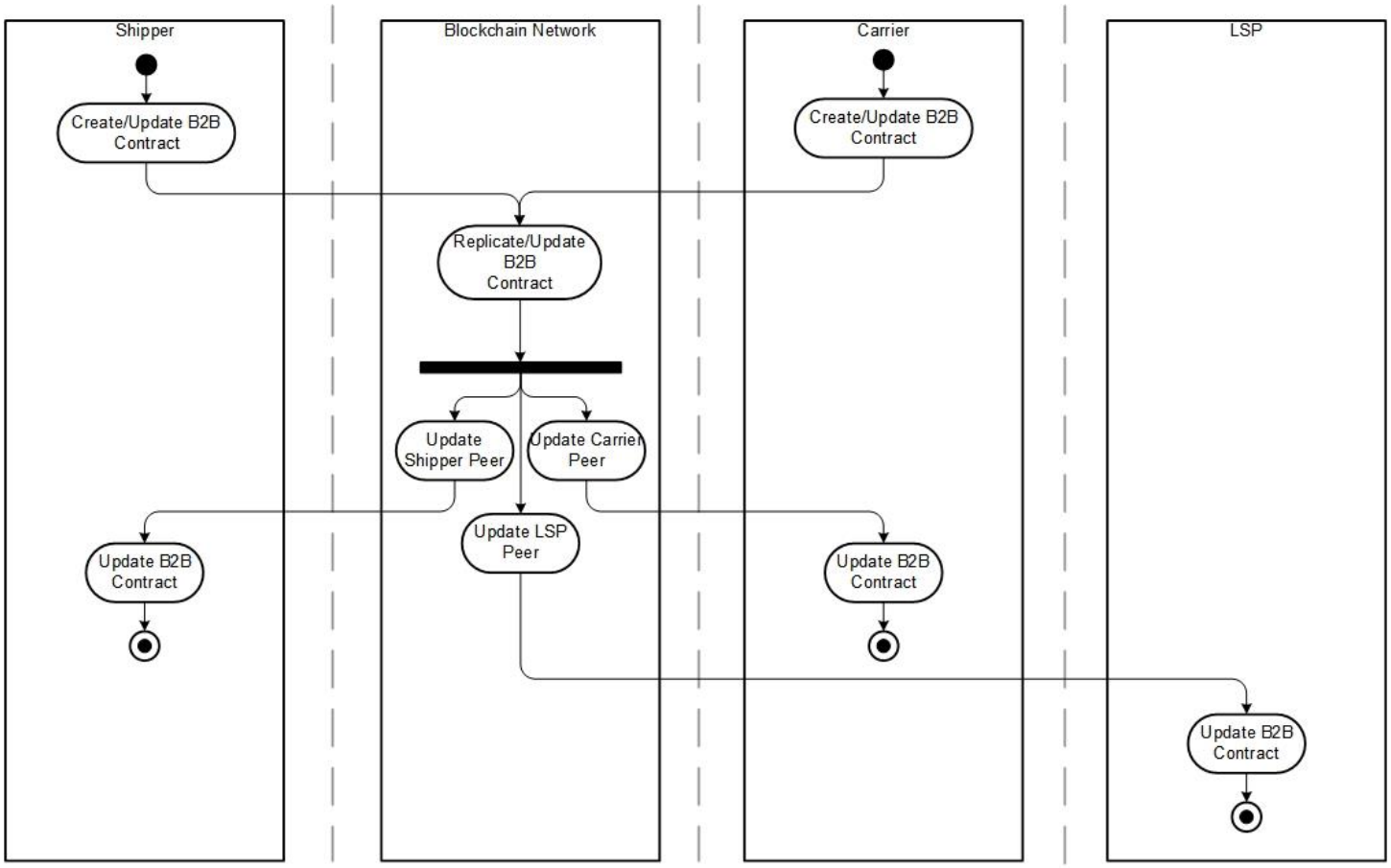

Figure 5: Solution Activity Diagram

In this report, we will only focus on the capability that either the Shipper or the Carrier can submit the changes to the B2B contract. When the update to the contract is done in the respective SAP TM system the transaction proposal is sent to the respective blockchain peer in the network, via the TM Contract API, and this contract is replicated to all the peers by the orderer. The SAP TM system then runs batch jobs to pull the updated contract from the blockchain network and appropriately update the contract in the SAP TM system.

5.4 Sequence Diagram

The sequencing of the different events in the system can be depicted via the below sequence diagram:

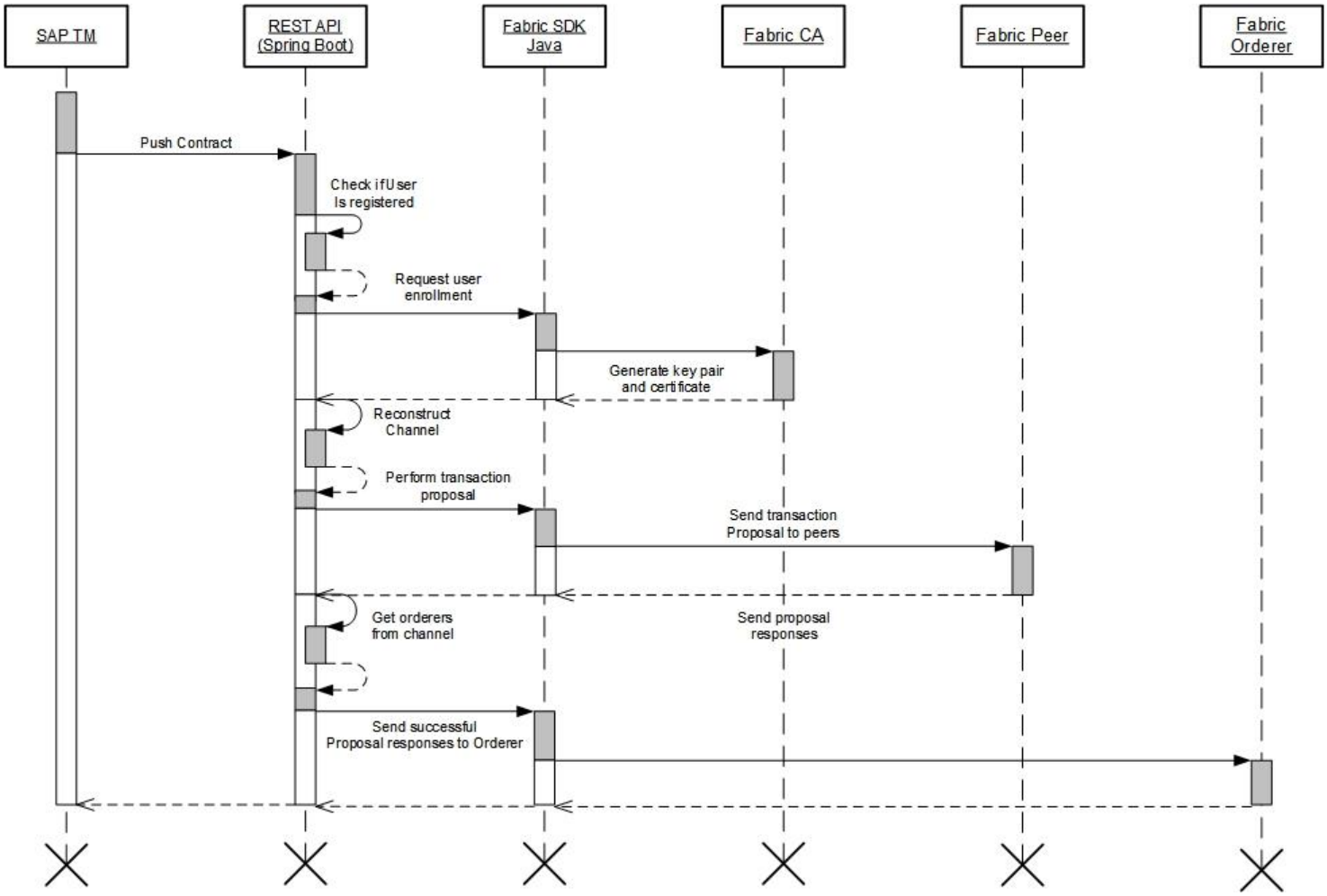

Figure 6: Sequence Diagram

6. System Specifications

This chapter gives the technical components of the solution.

6.1 Hardware Requirements

Since the solution is envisioned to be a cloud application the options would be to either use an IaaS (Infrastructure as a Service) provider or a PaaS (Platform as a Service) provider. Since we would require some amount of software installation to be done the choice was to go for a IaaS provider. For the solution, an AWS EC2 instance running Linux was chosen. Since the initial version of the solution will not be that resource intensive a "t2.micro" which runs a single Virtual CPU with a 1 GB memory and 8 GB of SSD storage is sufficient.

The only requirement would be the OS installed must support virtualization for the Docker containers to run.

6.2 Software Requirements

The solution requires the following software:

- Maven: Following the installation steps documented here
- Java SDK 1.8 & Spring: The REST APIs are provisioned using the Spring Boot library. Java is the programming language that is used for the application logic
- Docker & Docker Compose: Follow the link for the installation steps. With the installation of Docker CLI (Command Line Interface) commands are available through which one can execute the Docker commands
- Hyperledger Fabric: There is a small toolset that is provided by Hyperledger to generate the certificates and the keypairs for the various organizations in the network which requires installation. This toolset also downloads the latest Hyperledger Fabric provided Docker images.
- Go: Chaincode which is the programming language used to write business logic for Hyperledger Fabric is written in Go. It is necessary that the Go library is installed which is used to compile the Chaincode. The installation steps can be found here
- LevelDB: The ledger in Hyperledger Fabric uses LevelDB as its persistence. It is not a rule to use LevelDB. Even Couch DB can be used. The default Docker containers provisioned by Hyperledger uses LevelDB. As mentioned since this is already provisioned in the Docker Image a separate installation is not required
- Git: Once you install Git from here, you should be able to clone Git repositories which contain project source code and run the same

Hyperledger Fabric clearly documents all the pre-requisites for running the Blockchain network and it can be found here

Apart from the above software it is also required to have a running SAP TM system running for the integration tests. Since the source system of the B2B contracts is the SAP System this is required for testing

6.3 Programming Languages

- Java: REST APIs will be exposed to perform CRUD (Create, Read, Update, Delete) operations on the Blockchain network. To maintain the transaction integrity the delete in Hyperledger places only a delete marker on the given key of the document. Hyperledger Fabric provides:

    - Java SDK (Software Development Kit)
    - Node SDK
- Go: Chaincode (Smart Contracts) which is the business logic for Blockchain is written in Go. The program needs to implement the Shim interface provided by Hyperledger to "Get" or "Put" the state to the ledger
- ABAP: ABAP is the proprietary language from SAP. The business logic implemented in SAP TM is written in ABAP. In order to send the B2B contracts to Hyperledger the REST APIs exposed must be consumed using ABAP.
- BASH Scripting: To bring up the different Docker Containers, creating the keypairs and certificates we will be using BASH scripts to automate this process
- YAML: A markup language that will be used for writing the different configuration files like:
    - Docker compose files
    - Blockchain configuration files
        - MSP (Membership Service Providers) configuration
        - Crypto Configuration files to generate the certificates and the key pairs
    - Swagger files for the REST API documentation

7. Summary

With the help of this project it was possible to bring up a Blockchain network containing the three organizational entities, the Shipper organization, the Carrier organization and the LSP organization. The blockchain was running with a chaincode application that could store the state of a B2B contract in a shared ledger.

The interface between the SAP TM system and the Blockchain network was a REST server which consumed the Java flavor of Hyperledger SDK. It was seen how the update to the B2B contract pushed this object to the right organization in Hyperledger Fabric network and how the orderer deployed in the network took care of distributing this state of the B2B contract to the other organizations in the network.

Below are the steps that need to be followed to get the above-mentioned results.

7.1 Bringing up the Blockchain Network

The BASH script `saptmn.sh` provided with the project automates the following steps:

1. Read the `crypto-config.yaml` file to generate the necessary cryptographic material (Key pair and X.509 certificates) to all the entities in the network
2. Read the `configtx.yaml` file to generate the genesis block and the channel files to configure the different organizations in the channel.
3. Use the `docker-compose-cli.yaml` file to spin up the Dockers containers for the organizations as mentioned in the network topology mentioned in section 4.2
4. The Docker compose files also maps some volumes from the local system to the file system of the Docker containers. This facilitates:
    a. Storing the certificates in the required folders of the Docker containers so that the signature validation can be done
    b. Copy the chaincode executable to the Docker containers
5. Once the Docker containers are running the following steps are performed in the CLI container (automated using the `script.sh`)
    a. Create the saptmchannel
    b. Join the ShipperOrg, CarrierOrg and the LSPOrg to the channel
    c. Set the Anchor peers for the Organizations. Every organization has one anchor peer called as peer0
    d. Install the chaincode on all the peers
    e. Instantiate the chaincode on the ShipperOrg peer only (The instantiation on the other peers is done later while executing the transactions)

With the above steps now the Blockchain network is running.

7.2 Deploying and running the REST Server

The next steps are to run the application that provide the REST API endpoints to interact with the Blockchain network. The project provided is a Maven project which uses Spring Boot to handle web requests. To run the application navigate to the root directory of the project and execute `mvn spring-boot:run`

The Spring Boot application has an inbuilt tool called Swagger, which makes it easy to test the REST endpoints using the browser. Once the project is running the Swagger UI can be accessed by `http://<host>:<port>/swagger-ui.html`. For unit tests this can be used to test the REST Endpoints

7.3 Integration Tests and results

The integration test was performed using multiple SAP TM systems, each presumed to be owned by a different organizational entity.

It was seen that with Blockchain:

- The B2B contracts were seamlessly distributed across the different SAP systems
- Every change to the contract reflected in the different system almost instantaneously
- The changes to the B2B contracts were also digitally signed by the different organizations that made the change keeping the change process transparent
- Since the organizations were using the updated B2B contracts for the transactional documents the follow business processes were error free

8. Conclusion

Hyperledger Fabric is a relatively new technology started in 2015 by the Linux Foundation. It has already gained a lot of traction in the market and the surrounding community support for the same is increasing by the day. With Hyperledger Fabric offering capabilities like:

- SDK support for client application written in different languages,
- Fault tolerance mechanisms to share data to the different entities of the network with the usage of reliable messaging services like Kafka,
- Have different endorsement policies for every business domain,
- The ability to integrate to different Certificate Authorities or LDAP (Lightweight Directory Access Protocol), etc.

make Hyperledger Fabric the natural choice to implement Blockchain solutions for businesses.

For SAP TM, this is one of the pilot scenarios to showcase how supply chain execution scenarios can be built using the Blockchain technology. SAP TM also offers other functionality like Strategic Freight Management, Carrier invoicing, dispute management and integration with SAP Financials. At a very high level some of these scenarios can also be modeled in Blockchain. There are also other collaborative scenarios that are planned to be developed within SAP TM and this is a very good candidate for Blockchain.

The primary intention of Blockchain is to increase the trust between the organizations so that doing B2B business can be seamless. When the network of organizations is trustworthy then the business flourish is such networks.

9. Directions for Future Work

- Currently the SAP TM system is running on its own HANA (In memory) database. The current project focuses on enabling the automatic update of the B2B contracts in the SAP TM system via batch reports that are written in SAP TM. The next step could be either:
    - Have the SAP TM system running on the same database as the Blockchain network so that the updated contracts are available immediately or
    - Have a publish subscribe mechanism by which the SAP TM system is notified of a change in the B2B contract and there are subscribers to this event which triggers an automatic update of the contract. This is possible with an integration layer in between

- In this project the update of the contract supports only flat rate changes. The next scope could also include the changes to the rate table. A rate table is a multidimensional mechanism to maintain the rates for the freight charges.

- The automatic onboarding of the user to the Blockchain network should be done. Currently the transaction proposals to the Blockchain network is done using the pre-defined user who has already been onboarded to the organization during the setup of the network.

- Currently there are no endorsement policies that are set up for the chaincode application. The endorsement policies can be defined based on the business needs to guard the updates to B2B contracts. One example could be that if the carrier has increased the rates of the shipment by more than a fixed percentage then the shipper can reject the update of the contract.

10. References

[1] Decentralized Blockchain Technology and rise of Lex Cryptographia (pdf)

[2] A Comprehensive Literature Review on the Blockchain Technology as an Technological Enabler for Innovation (link)

[3] Understanding Blockchain Consensus Models (pdf)

[4] Impossibility of Distributed Consensus with One Faulty Process (pdf)

[5] The Byzantine's general problem (pdf)

[6] Hyperledger Fabric Official Documentation (link)

Apart from the above references the following articles were also referred

The truth about Blockchain (link)

IBM pushes Blockchain into the Supply Chain (link)

10.1 Project References

The entire project is available in GitHub. The project is available here

11. Glossary

Term	Abbreviation	Definition
SAP Transportation Management	SAP TM	SAP Transportation Management is a comprehensive solution for performing all activities connected with the physical transportation of goods from one location to another
Spring		Spring is an application development framework for enterprise Java. Spring enabled application developers to focus on the application development and abstracts technical aspects. For more details refer https://spring.io/
Amazon Web Services	AWS	Amazon Web Services. Cloud computing services provided by Amazon.
Swagger		Swagger is the world's largest framework of API developer tools for the OpenAPI Specification(OAS), enabling development across the entire API lifecycle, from design and documentation, to test and deployment
Maven		Maven is a software management and comprehension tool. With Maven, it is easy manage dependencies in your project. This is based on the Project Object Model (POM). With the POM file it is easy to manage a project's build, reporting and documentation.
Docker and Docker Compose		Docker is a software container platform. With this the developers, can package the application and run it anywhere. The Docker containers only run the libraries and settings that are required for the application to run and this container can be deployed anywhere
Git		Git is a version control system to track the versions of the programs and files. This project uses GitHub for versioning and to store the source code
Hyperledger Fabric		Hyperledger Fabric is a platform for distributed ledger solutions underpinned by a modular architecture delivering high degrees of confidentiality, resiliency, flexibility and scalability. It is designed to support pluggable implementations of different components and accommodate the complexity and intricacies that exist across the economic ecosystem
Peer to Peer	P2P	A distributed application architecture that partitions tasks or workloads between peers
Kafka		A streaming platform that lets you publish and subscribe to a stream of records, allows you to store the stream of records in a fault tolerant way and it allows you to process the stream of records as they occur

Practical Byzantine Fault Tolerance	PBFT	Provides high-performance Byzantine state machine replication, processing thousands of requests per second with sub-millisecond increases in latency.
Simplified Byzantine Fault Tolerance	sBFT	An implementation of the Practical Byzantine Fault Tolerance

12. Checklist

1.	**Is the final report neatly formatted with all the elements required for a technical Report?**	Yes
2.	Is the Cover page in proper format as given in Annexure A?	Yes
3.	Is the Title page (Inner cover page) in proper format?	Yes
4.	(a) Is the Certificate from the Supervisor in proper format? (b) Has it been signed by the Supervisor?	Yes Yes
5.	Is the Abstract included in the report properly written within one page? Have the technical keywords been specified properly?	Yes Yes
6.	Is the title of your report appropriate? **The title should be adequately descriptive, precise and must reflect scope of the actual work done.** Uncommon abbreviations / Acronyms should not be used in the title	Yes
7.	Have you included the List of abbreviations / Acronyms?	Yes
8.	Does the Report contain a summary of the literature survey?	Yes
9.	Does the Table of Contents include page numbers? (i). Are the Pages numbered properly? (Ch. 1 should start on Page # 1) (ii). Are the Figures numbered properly? (Figure Numbers and Figure Titles should be at the bottom of the figures) (iii). Are the Tables numbered properly? (Table Numbers and Table Titles should be at the top of the tables) (iv). Are the Captions for the Figures and Tables proper? (v). Are the Appendices numbered properly? Are their titles appropriate	Yes Yes Yes Yes Yes NA
10.	Is the conclusion of the Report based on discussion of the work?	Yes
11.	Are References or Bibliography given at the end of the Report? Have the References been cited properly inside the text of the Report? Are all the references cited in the body of the report	Yes Yes Yes
12.	Is the report format and content according to the guidelines? The report should not be a mere printout of a Power Point Presentation, or a user manual. Source code of software need not be included in the report.	Yes

Declaration by Student:

I certify that I have properly verified all the items in this checklist and ensure that the report is in proper format as specified in the course handout.

Place: **Bangalore** **Signature of the Student**

Date: **October 17, 2017** **Name:** **Sudhindra. C**

ID No.: **2015HT13101**