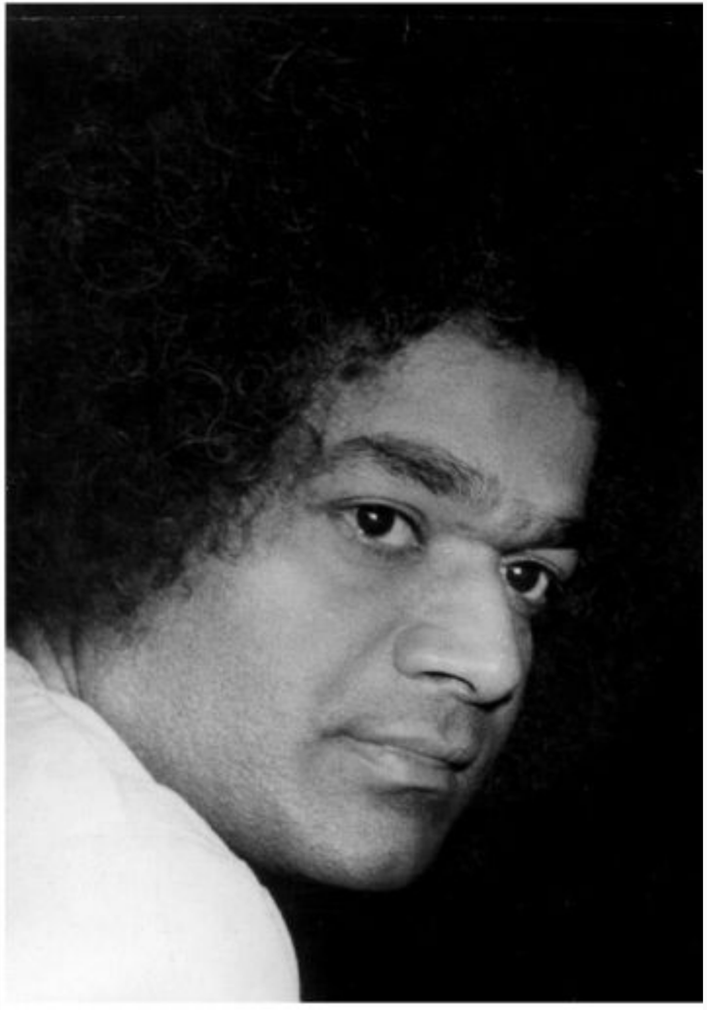
Dedicated at Thy Lotus Feet



BlockChain Consensus
&
Smart HBasechainDB

Bangarugiri Sateesh • 18555

Supervisor : Dr. Pallav Kumar Baruah

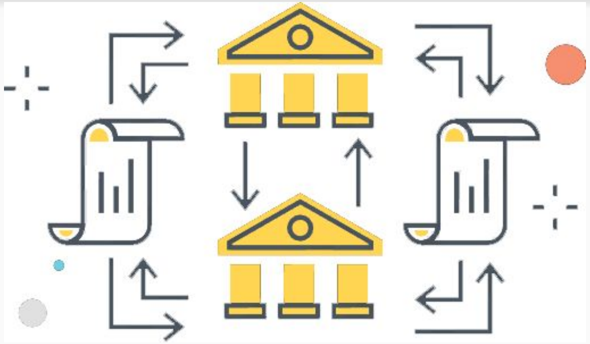
Contents

PROJECT INTERIM REVIEW

- Introduction
- Problem Statement
- Objectives
- Forking Resilient Kafka Orderer
- Smart HBasechainDB
- Future Work

Source[1]

What is Blockchain ?

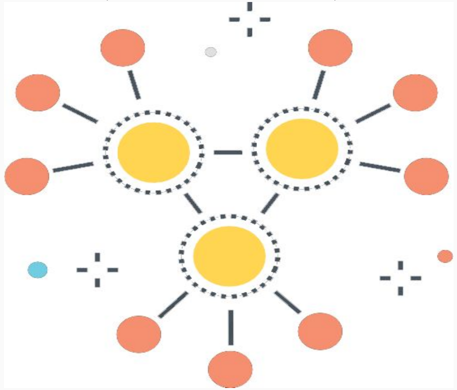


Distributed Ledger Technology
(acronym: DLT)

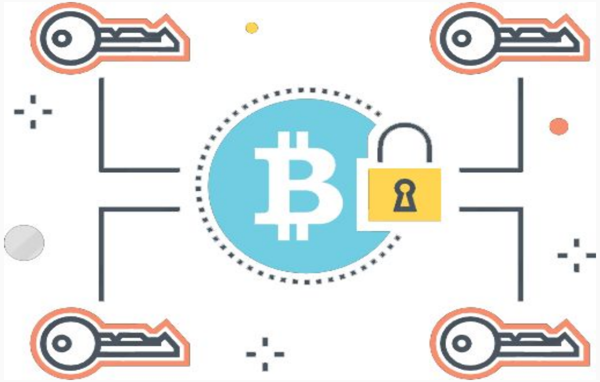
Ledger records
ALL transactions

Source[2]

'key/value' database with
current state (optional)

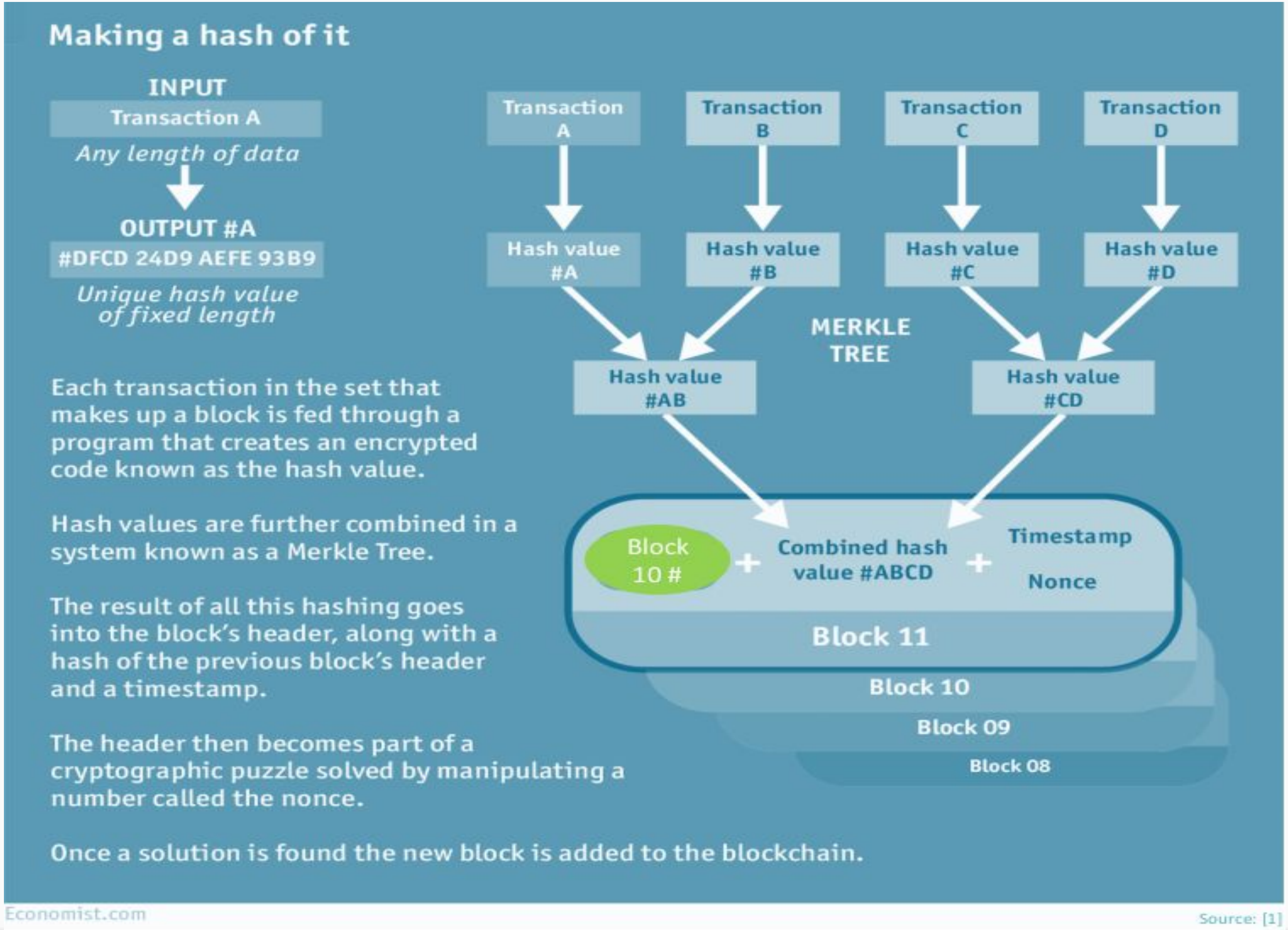


Decentral peer-to-peer
network of nodes



Public key-cryptography
without central authority

Any transaction added is
validated by multiple entities



BUNDLING TRANSACTIONS INTO A BLOCK

CHAIN OF BLOCKS BY ADDING HASH OF THE PREVIOUS BLOCK TO CURRENT

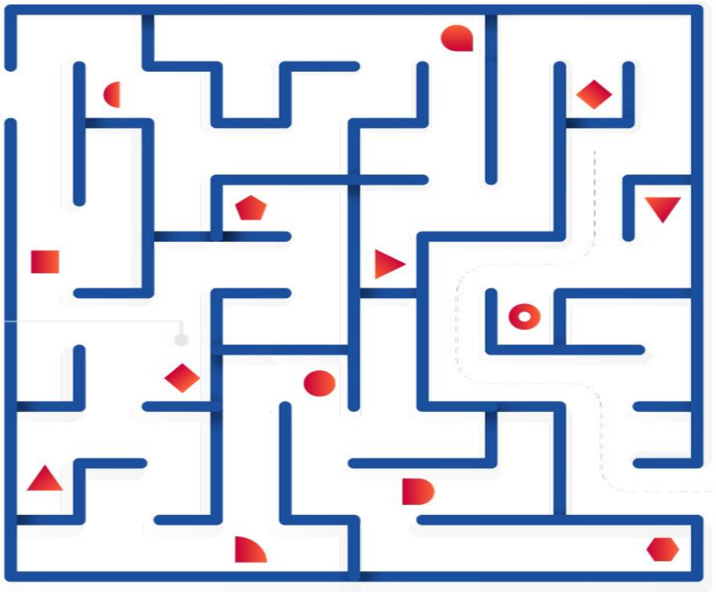
TAMPER-PROOFING

Proof of Work

*The system is called **proof of work** because the probability of mining the block is increased with the amount of work that is put in.*

1 A very complex mathematical challenge is proposed to the blockchain network

2 The miners have to compete to find the solution, which takes time and resources, making it costly for the contestants.

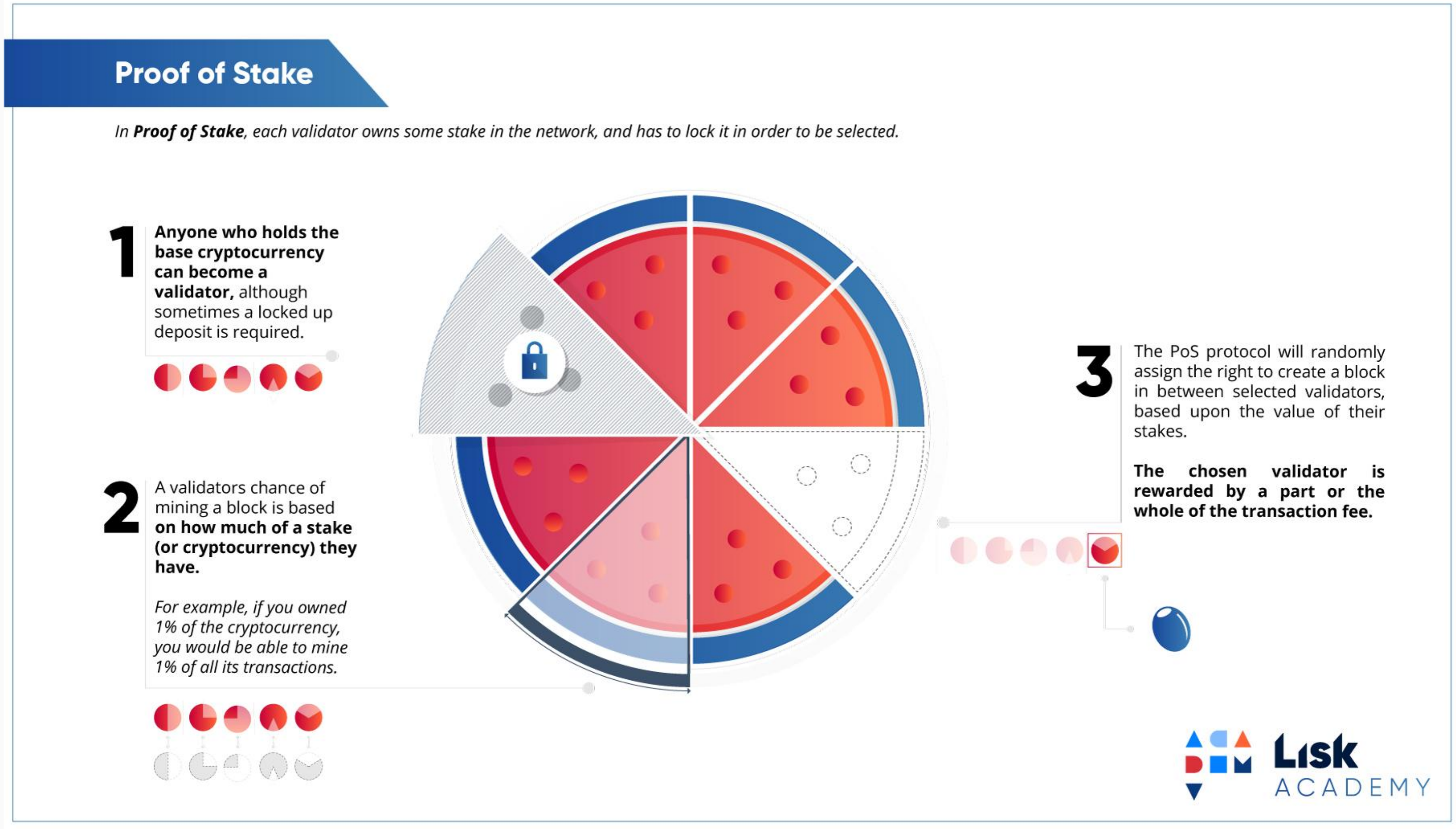


3 The first miner to solve the problem has the ability to validate transactions and create a new block, receiving a reward afterwards.

Lisk ACADEMY

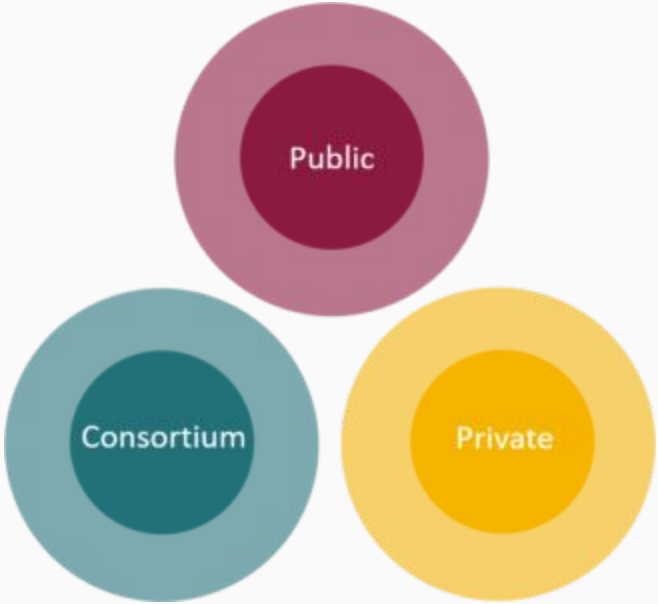
Ethereum

Source[4]

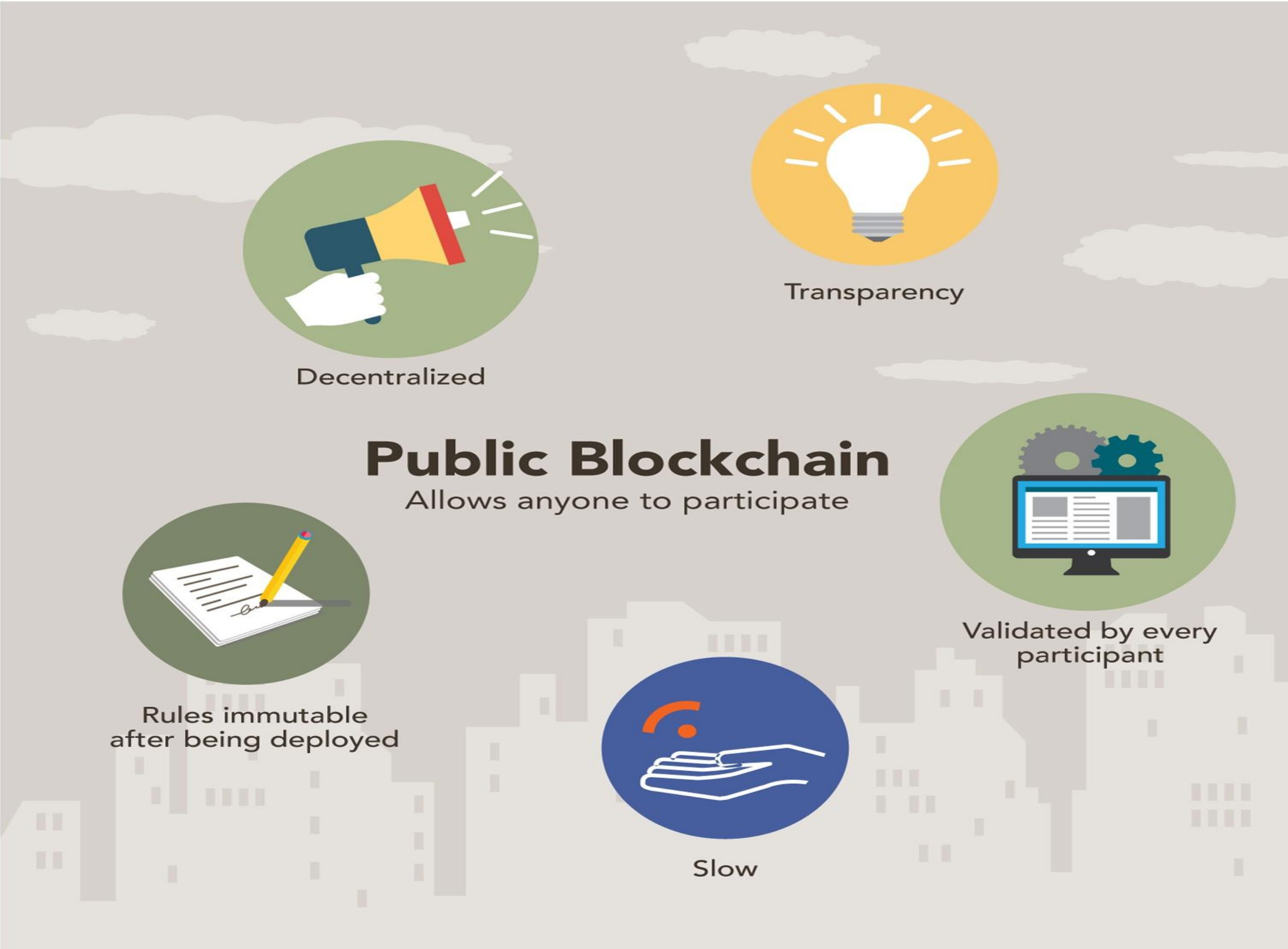

Proof of Stake
In **Proof of Stake**, each validator owns some stake in the network, and has to lock it in order to be selected.
1
Anyone who holds the base cryptocurrency can become a validator, although sometimes a locked up deposit is required.
2
A validators chance of mining a block is based **on how much of a stake (or cryptocurrency) they have.**
For example, if you owned 1% of the cryptocurrency, you would be able to mine 1% of all its transactions.
3
The PoS protocol will randomly assign the right to create a block in between selected validators, based upon the value of their stakes.
The chosen validator is rewarded by a part or the whole of the transaction fee.
Lisk
ACADEMY

Source[5]

Types of Blockchains

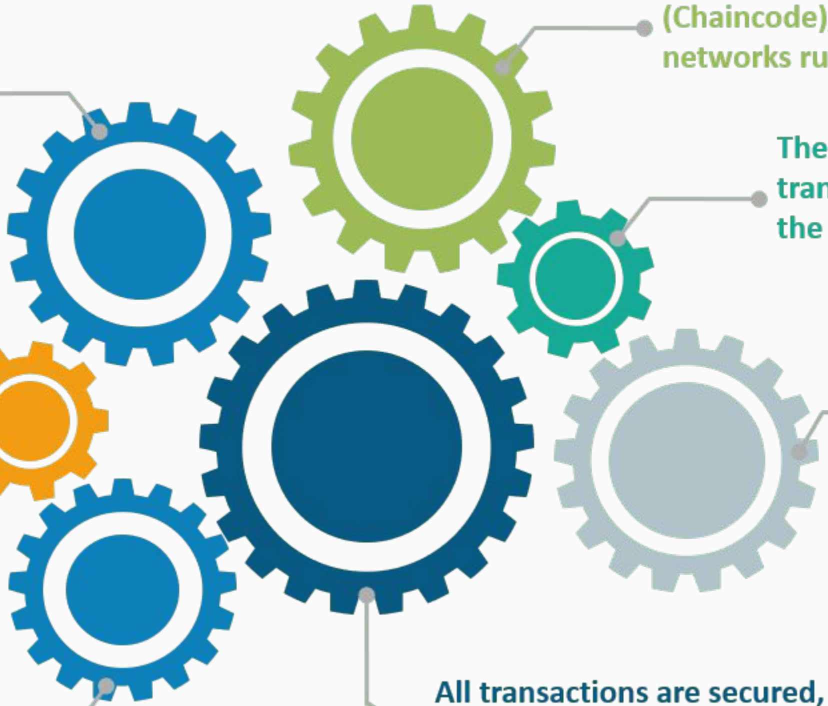


Public Blockchain Features

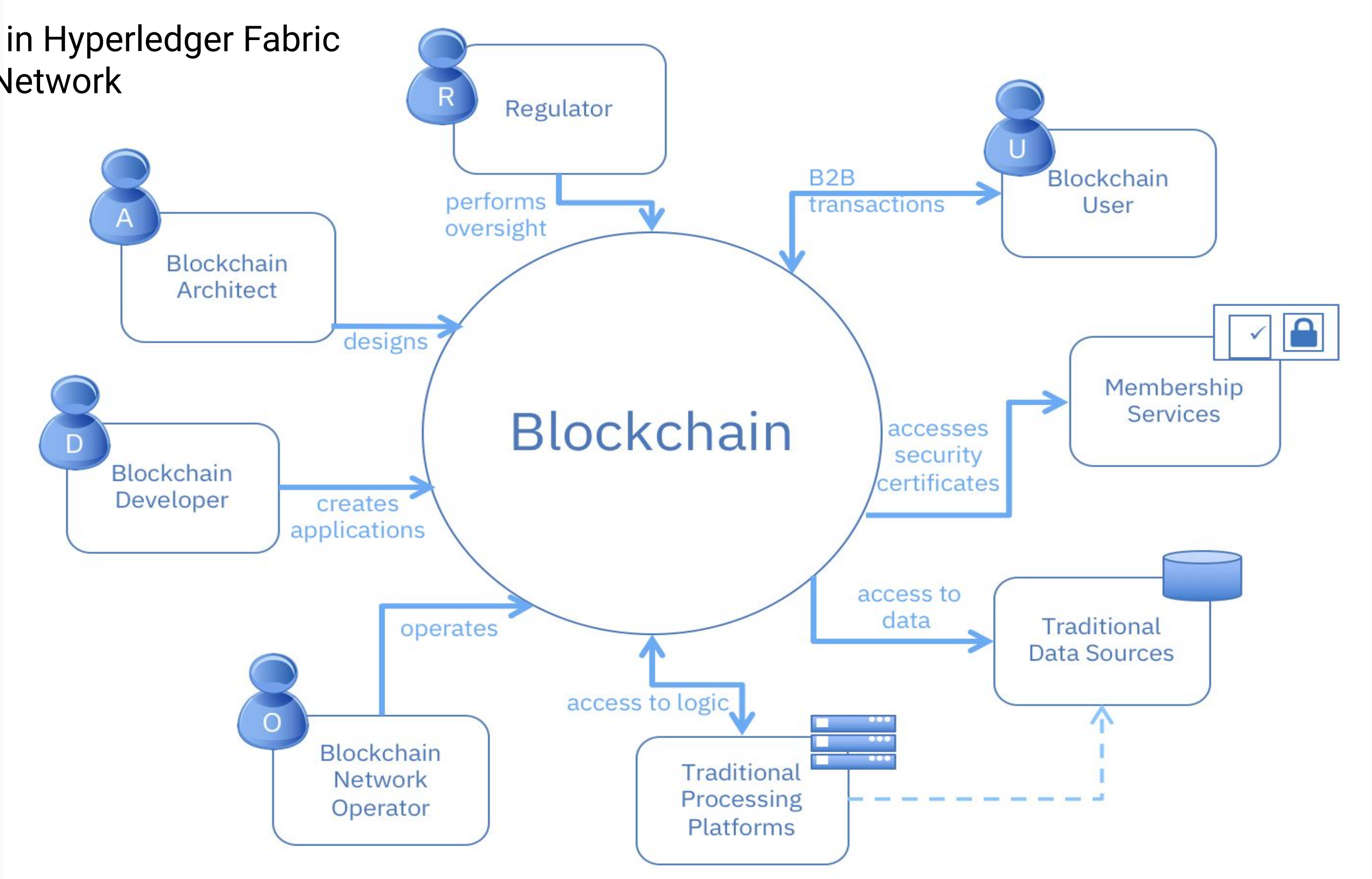


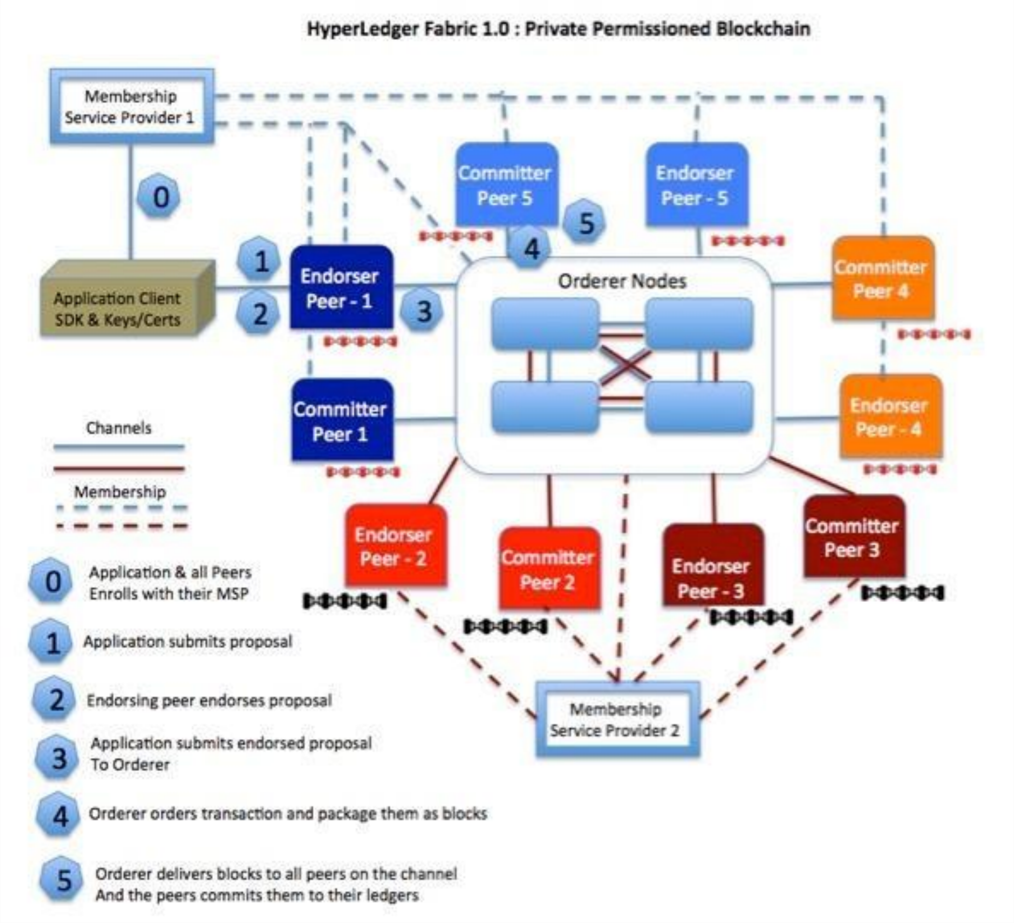
HYPERLEDGER FABRIC

Source[8]

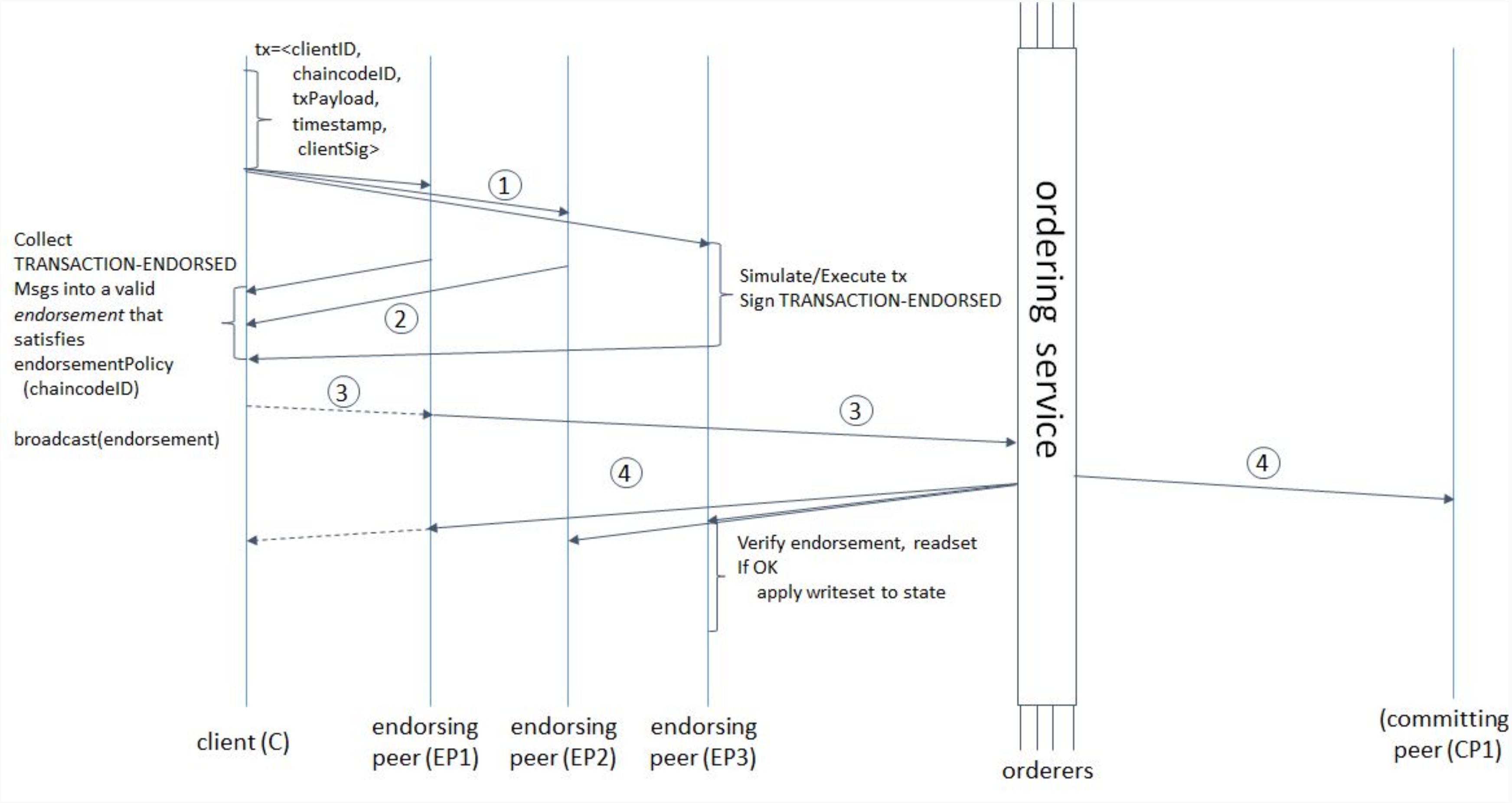

Fabric issues transactions with derived certificates that are unlinkable to the owning participant
Relies on a smart contract system (Chaincode), which every peer of the networks runs in Docker containers
The events are structured as transactions and shared among the different participants
All participants must register proof of identity to membership services in order to gain access to the system
The transactions are executed without a cryptocurrency
The content of each transaction is encrypted to ensure only the intended participants can see the content
All transactions are secured, private, and confidential. Fabric can only be updated by consensus of the peers

Participants in Hyperledger Fabric Network



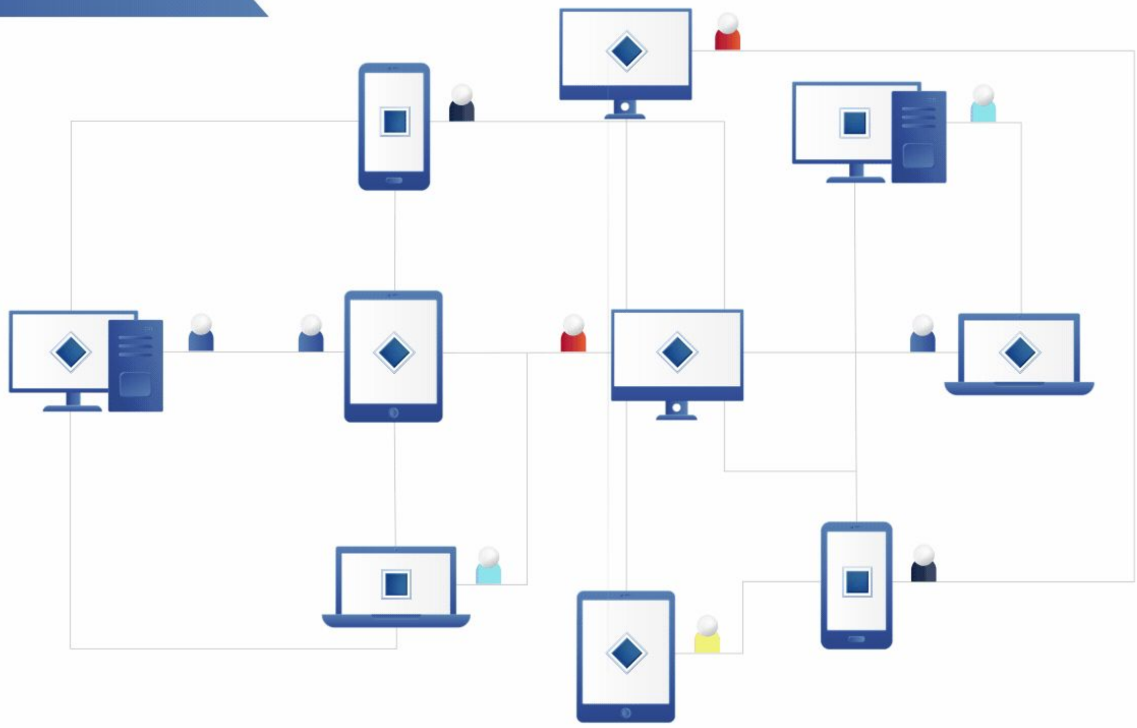

HyperLedger Fabric 1.0 : Private Permissioned Blockchain
Membership Service Provider 1
0
Application Client SDK & Keys/Certs
1
2
Endorser Peer - 1
3
Committer Peer 5
5
4
Endorser Peer - 5
Orderer Nodes
Committer Peer 4
Endorser Peer - 4
Committer Peer 1
Channels
Membership
Endorser Peer - 2
Committer Peer 2
Endorser Peer - 3
Committer Peer 3
Membership Service Provider 2
0 Application & all Peers Enrolls with their MSP
1 Application submits proposal
2 Endorsing peer endorses proposal
3 Application submits endorsed proposal To Orderer
4 Orderer orders transaction and package them as blocks
5 Orderer delivers blocks to all peers on the channel And the peers commits them to their ledgers

Source[11]


tx=<clientID,
chaincodeID,
txPayload,
timestamp,
clientSig>
1
Collect
TRANSACTION-ENDORSED
Msgs into a valid
endorsement that
satisfies
endorsementPolicy
(chaincodeID)
2
Simulate/Execute tx
Sign TRANSACTION-ENDORSED
ordering service
3
broadcast(endorsement)
3
4
4
Verify endorsement, readset
If OK
apply writeset to state
client (C)
endorsing
peer (EP1)
endorsing
peer (EP2)
endorsing
peer (EP3)
orderers
(committing
peer (CP1)

Source[II]

Consensus Protocol



Crash Fault Tolerance vs Byzantine Fault Tolerance

- **Crash Fault Tolerance** (CFT) is one level of resiliency, where the system can still correctly reach consensus if components fail. (eg. Kafka Orderer and Raft Orderer)
- Crash failure = the process halts
- Fail-Stop failure is a simple abstraction that mimics crash failure when process behavior becomes arbitrary.
- If a system cannot tolerate fail-stop failures, it cannot tolerate crash failures.

- **Byzantine Fault Tolerance** (BFT) is more complex and deals with systems that may have malicious actors.
- Numerous possible causes.
- Most difficult kind of failure to deal with.
- Some of the arbitrary node failures are given below : **Failure to return a result, Respond with an incorrect result, Respond with a deliberately misleading result, Respond with a different result to different parts of the system.**

Problem Statement

To build a forking resilient Kafka ordering service by modifying existing one and ensuring safety guarantees.

To make existing HBasechainDB support Smart Contract by modifying its architecture and check for flexibility.

Objectives

- ❑ To study the Bitcoin blockchain technology for zeroing on problems to work upon
- ❑ To explore newer blockchain implementations – Ethereum and Hyperledger
- ❑ To write smart contracts in the Ethereum framework and acquire an understanding of its use cases
- ❑ To study approaches taken to addressing Byzantine Faults in blockchain
- ❑ To explore consensus algorithms like Paxos, Raft, Kafka Orderer, Honey Badger
- ❑ To study the HBasechainDB1.0 and HBasechainDB2.0 for understanding its architecture
- ❑ To implement modified Kafka Orderer for a forking resilience in Hyperledger Fabric in GO language
- ❑ To accommodate the changes in architecture of HBasechainDB2.0 for supporting Smart Contract and check for flexibility

Work Done

- ✅ To study the Bitcoin blockchain technology for zeroing on problems to work upon
- ✅ To explore newer blockchain implementations – Ethereum and Hyperledger
- ✅ To write smart contracts in the Ethereum framework and acquire an understanding of its use cases
- ✅ To study approaches taken to addressing Byzantine Faults in blockchain
- ✅ To explore consensus algorithms like Paxos, Raft, Kafka Orderer, Honey Badger
- ✅ To study the HBasechainDB1.0 and HBasechainDB2.0 for understanding its architecture
- ❑ To implement modified Kafka Orderer for a forking resilience in Hyperledger Fabric in GO language
- ❑ To accommodate the changes in architecture of HBasechainDB2.0 for supporting Smart Contract and check for flexibility

Source[IV]

Literature

- Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed, Cocco, Jason Yellick. (2018). "Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains".

- Christian Gorenflo, Stephen Lee, Lukasz Golab, S. Keshav, (March 2019). “ FastFabric: Scaling Hyperledger Fabric to 20,000 Transactions per Second”.

- Arati Baliga, Nitesh Solanki, Shubham Verekar, Amol Pednekar, Pandurang Kamat and Siddhartha Chatterjee, “Performance Characterization of Hyperledger Fabric”, 2018 Crypto Valley Conference on Blockchain Technology.

- Diego Ongaro and John Ousterhout, (May 2013). “In Search of an Understandable Consensus Algorithm”

- Leslie Lamport, (01 Nov 2001). “Paxos Made Simple”

- Medium Blogs

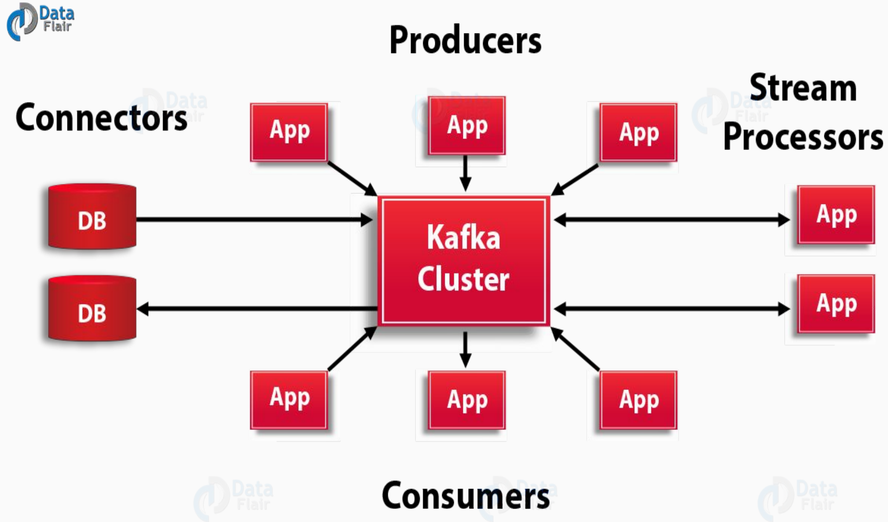
APACHE kafka®
A distributed streaming platform

Distributed publish subscribe messaging system

- It was developed at LinkedIn and later on became a part of Apache Project.
- It is fast, agile, scalable and distributed by design.
- A message handling system, that uses the Publish-Subscribe Model.
- Consumers subscribe to Topic to receive new messages , that are published by Producer.

Source[V]

Kafka Architecture and Terminology



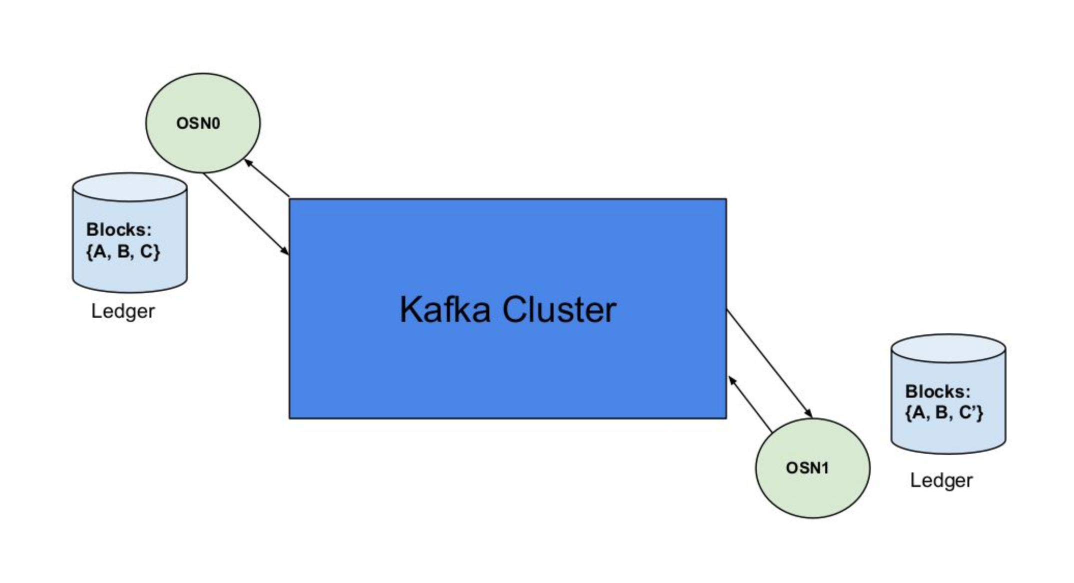
- Topic : A stream of messages belonging to a particular category is called a topic
- Producer : A producer can be any application that can publish messages to a topic
- Consumer : A consumer can be any application that subscribes to topics and consumes the messages
- Broker : Kafka cluster is a set of servers, each of which is called a broker

Kafka Orderer in Hyperledger Fabric

- For every channel, we have a separate partition.
- Each channel maps to the single partition topic.
- The OSNs after confirming the permissions in the chain relay the incoming client transactions (received via the Broadcast RPC)
- The OSNs can then consume that partition and get back an ordered list of transactions that is common across all ordering service nodes.
- The transactions in a chain are batched, with a timer service. That is, whenever the first transaction for a new batch comes in, a timer is set.
- The block (batch) is cut either when the maximum number of transactions are reached (or when the timer expires, whichever comes first).
- Each OSN maintains a local log for every chain, and the resulting blocks are stored in a local ledger.
- The transaction blocks are then served to the receiving clients via the Deliver RPC.
- In the case of a crash, the relays can be sent through a different OSN since all the OSNs maintain a local log. This has to be explicitly defined, however.

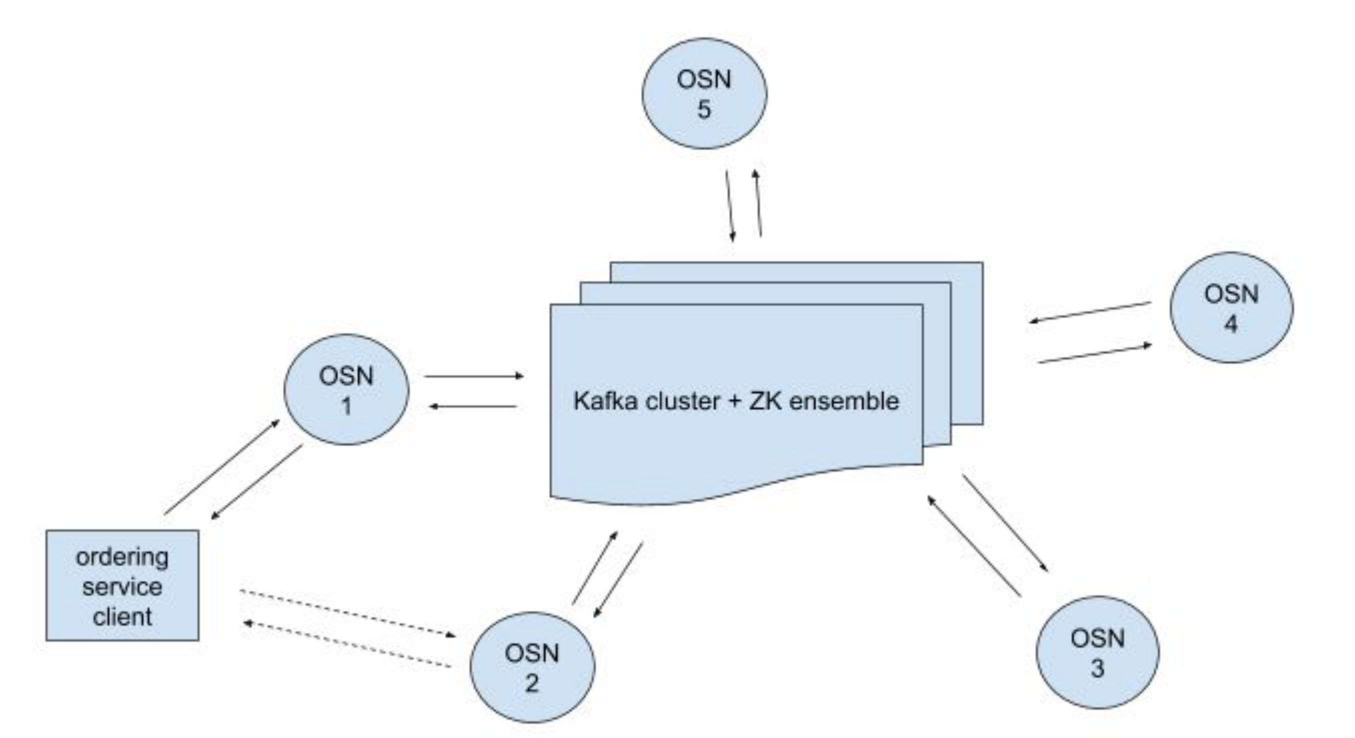
Forking Resilient Kafka Orderer

- BlockChain Fork: Occurs when two or more blocks have the **same block height** and the block height of a particular block is defined as the number of blocks preceding it in the blockchain.
- Fork happens when two miners find a block at nearly the same time in Bitcoin blockchain.
- Forking in each and every blockchain is different, based on the design architecture.
- Fork in Hyperledger Fabric:



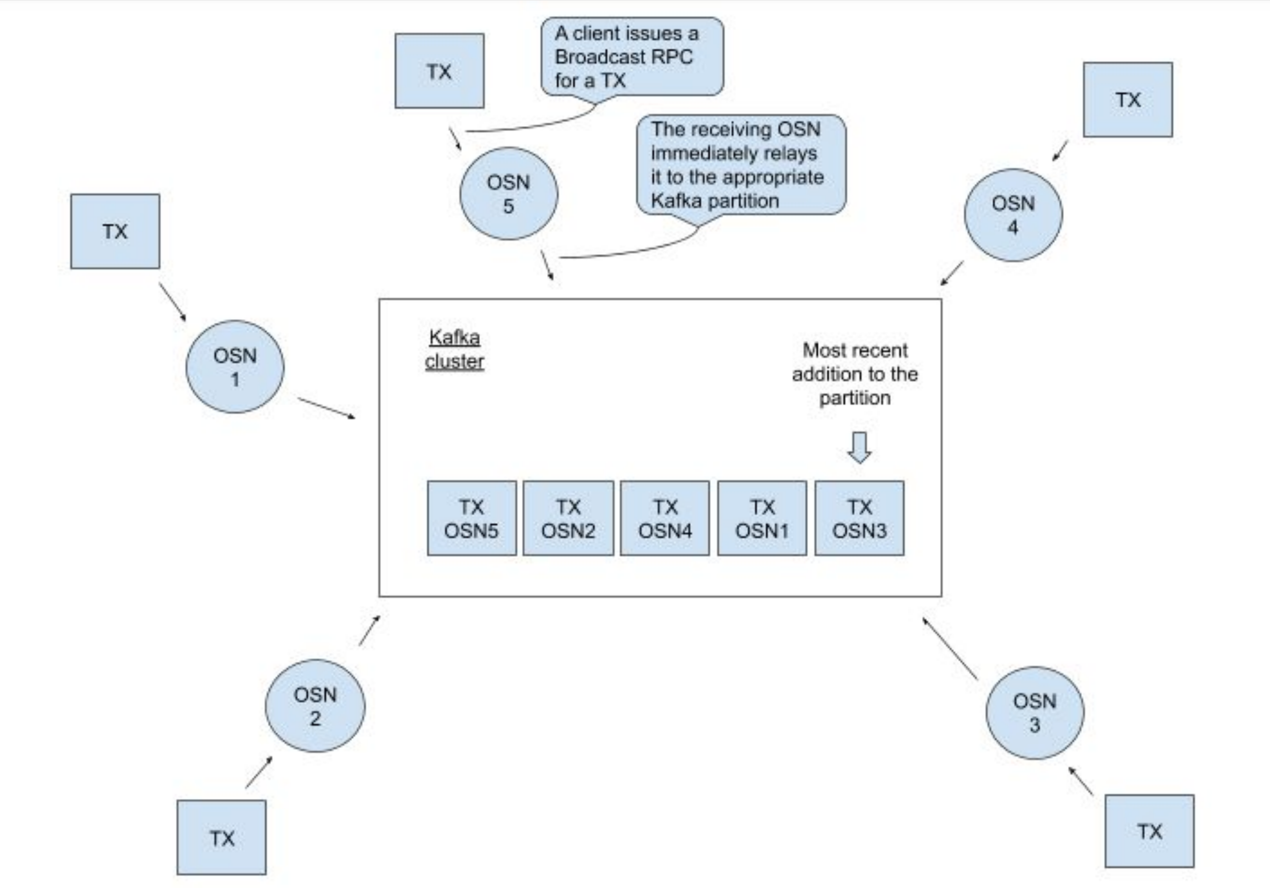
Source[VI]

Kafka Ordering Service in Hyperledger Fabric



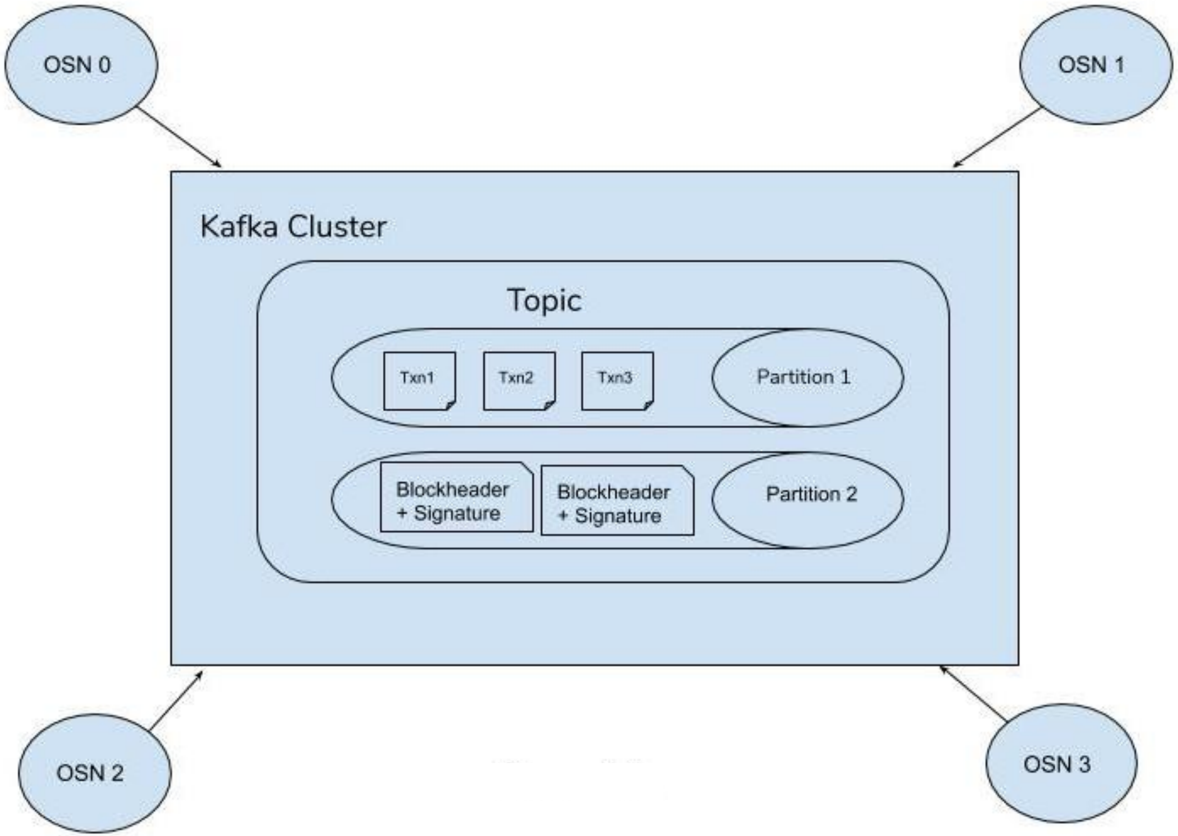
Source[VII]

Kafka Ordering Service in Hyperledger Fabric



Source[VIII]

Modified Kafka Ordering Service in Hyperledger Fabric for Forking Resilience



Source[VI]

Steps to avoid Forking in Kafka Orderer

- Create a second partition in the kafka topic for each channel apart from the first partition from where the OSNs consume the transactions.
- Have each OSN produce its signature over the block header, but before committing it, produce the signature and block header into the second partition.
- Each OSN commit their block into their local logs after getting a sufficient number of matching signatures from the second partition and all these signatures must be added into block metadata.

HBase

open source, multidimensional, distributed, scalable and a *NoSQL database*

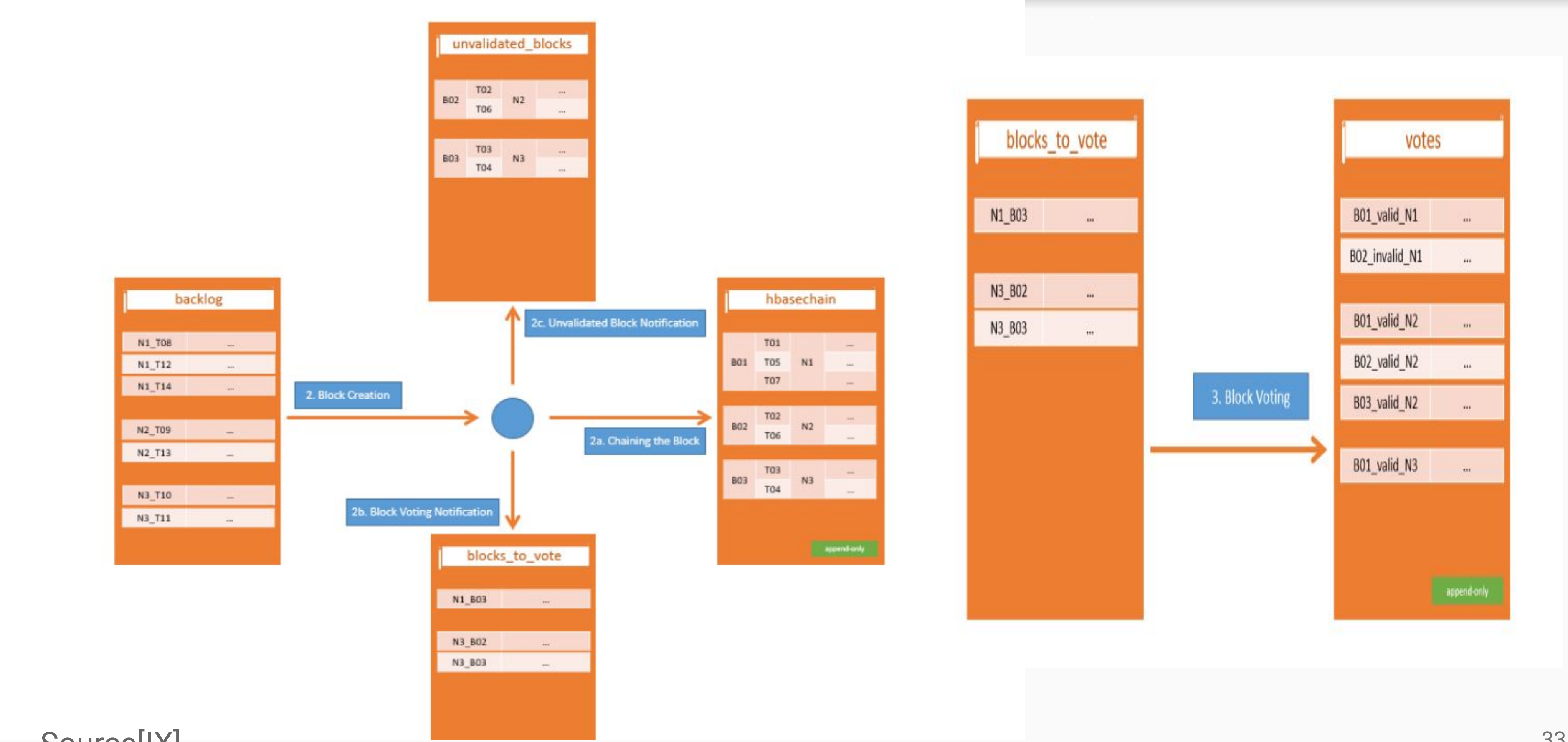
- It runs on top of HDFS(Hadoop Distributed File System)
- Designed to provide a fault tolerant way of storing large collection of sparse datasets.
- It achieves high throughput and low latency by providing faster Read/Write Access on huge datasets.
- It is the choice for the applications which require fast & random access to large amounts of data.

Source[V]

HBasechainDB

- A scalable blockchain framework on Hadoop ecosystem.
- It supports basic facilities like CREATION and TRANSFER of asset.
- It is operated using Federation of Nodes. All the nodes in the federation have equal privileges which gives HBasechainDB its decentralization.
- Any client can submit or retrieve transactions or blocks, but only the federation nodes can modify the blockchain.
- HBasechainDB's transaction model consists of; Transaction Id, Asset, List of Inputs, List of Outputs and Metadata.
- The current implementation of HBasechainDB uses six HBase tables: backLog, block, hbasechaindb, toVote, vote, reference.
- Operation of HBasechainDB involves a sequence of the following three steps: Transaction Insertion, Block Creation, and Block Voting.

Block Creation and Voting



Source[IX]

HBasechainDB

HBasechainDB1.0
Adarsh Saraf

HBasechainDB2.0
Subhankar Sahoo

Smart HBasechainDB

Timeline

Source[X]

Studied Blockchain basics(Bitcoin), Ethereum, Certification courses on Blockchain Technology.

Writing Chaincodes(in Fabric), Finalized Problem Statement, Studied Consensus algorithms(in Blockchain), Identifying Risk areas in Kafka Orderer

Installing HBase on 8-node Hadoop Cluster and HBasechainDB

JUNE | **JULY** | **AUGUST** | **SEPTEMBER** | **OCTOBER***

Writing Smart Contracts in Ethereum, Learnt Hyperledger Fabric System, Network setup for Fabric, Explored Cosmos Network and Edge Computing Platforms

Identified Fork situation in Kafka Order, Modifying Kafka Orderer, Studied HBasechainDB Architecture, Writing HiPC Paper on Forking Resilience in Kafka Orderer.

Future Work

Step 1

Installing HBase on 8-node Hadoop Cluster and HBasechainDB

Step 2

To implement modified Kafka Orderer for a forking resilience in Hyperledger Fabric in GO language and check for performance

Step 3

To accommodate the changes in architecture of HBasechainDB2.0 for supporting Smart Contract and check for flexibility

Acknowledgements

- Dr. Pallav Kumar Baruah
- Sri. Krishnamoorthy Balaraman
- Sri. Varun Vallampatla
- Jason Yellick, Blockchain Software Developer at IBM, North-Carolina Area, USA
- Soumya Nayak, System Engineer, First American India Pvt Ltd, Hyderabad
- Subhankar Sahoo

Image References

1. https://www.1point21gws.com/insights/blockchain/enterprise-cloud-security-is-blockchain-technology-the-missing-link/
2. “Build a decentralized blockchain application” by Robert van Mölken Oracle Developer ChampionBlockchain / Integration Specialist (AMIS)
3. “Blockchain-based Bigdata Solutions” Interim presentation by Adarsh Saraf
4. https://www.cryptolinenews.com/top-stories/facebook-plans-to-launch-its-own-cryptocurrency/
5. https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/proof-of-work
6. https://acadgild.com/blog/types-of-blockchain
7. https://medium.com/@yoa_project/types-of-blockchains-fc94ef840043
8. https://medium.com/coinmonks/beginning-with-the-hyperledger-fabric-241b54859476
9. https://www.edureka.co/blog/hyperledger-fabric/
10. https://medium.com/predict/stakeholders-in-a-blockchain-solution-d54591844c20
11. “Blockchain-based Bigdata Solutions” Interim presentation by Adarsh Saraf

Image References (Cont.)

I. http://websecuritypatterns.com/blogs/2017/07/07/unpacking-hyperledger-fabric-1-0-under-the-hood-of-a-permissioned-blockchain/
II. https://hyperledger-fabric.readthedocs.io/en/release-1.4/arch-deep-dive.html#swimlane
III. https://www.quora.com/What-is-consensus-in-blockchain
IV. https://www.shutterstock.com/search/coche+valid%C3%A9
V. https://kafka.apache.org/
VI. "Adding forking resilience in Kafka orderer even in the presence of Byzantine Faults" by Bangarugiri Sateesh, Pallav Kumar Baruah, Jason Yellick
VII. https://docs.google.com/document/d/19JihmW-8blTzN99lAubOfseLUZqdrB6sBR0HsRgCAnY/edit
VIII. https://docs.google.com/document/d/19JihmW-8blTzN99lAubOfseLUZqdrB6sBR0HsRgCAnY/edit
IX. "HBaseChainDB2.0" by Subhankar Sahoo and Pallav Kumar Baruah
X. https://gramboard.ai/blog/get-instagram-explore-page-2019/

References

- https://www.edureka.co/blog/hbase-tutorial
- https://www.geeksforgeeks.org/practical-byzantine-fault-tolerancepbft/
- https://codeburst.io/the-abcs-of-kafka-in-hyperledger-fabric-81e6dc18da56
- https://fabrictestdocs.readthedocs.io/en/latest/glossary.html
- http://homepage.divms.uiowa.edu/~ghosh/16612.week10.pdf
- https://www.cs.princeton.edu/courses/archive/fall16/cos418/docs/L9-bft.pdf
- https://www.investopedia.com/terms/b/block-height.asp
- DataFlair, Edureka.

Source[11]