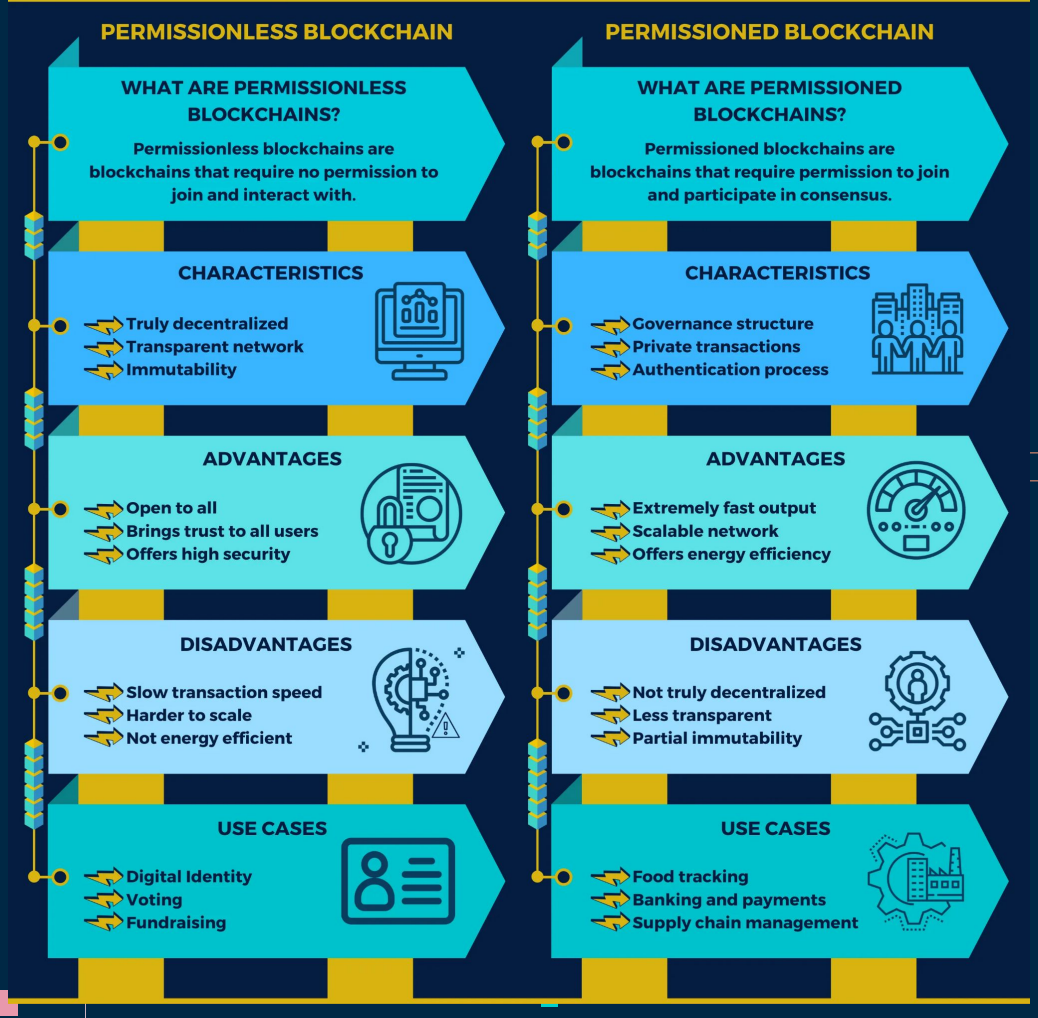
Blockchain Consensus Protocols in the Wild

Christian Cachin and Marko Vukolic ´
IBM Research - Zurich

This presentation is brought to you by Aabhas Tonwer, Anurag Dhasmana and Rachit Dhamija

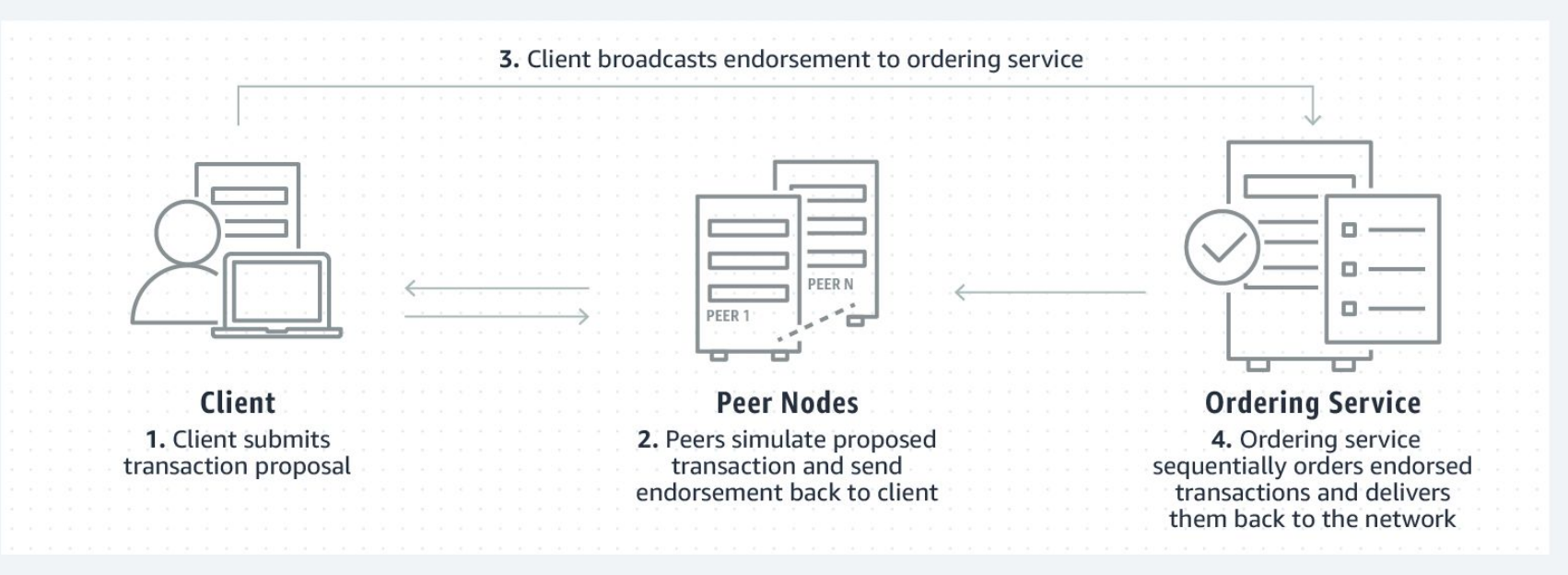


Credits:- https://101blockchains.com/permissioned-vs-permissionless-blockchains/

HYPERLEDGER

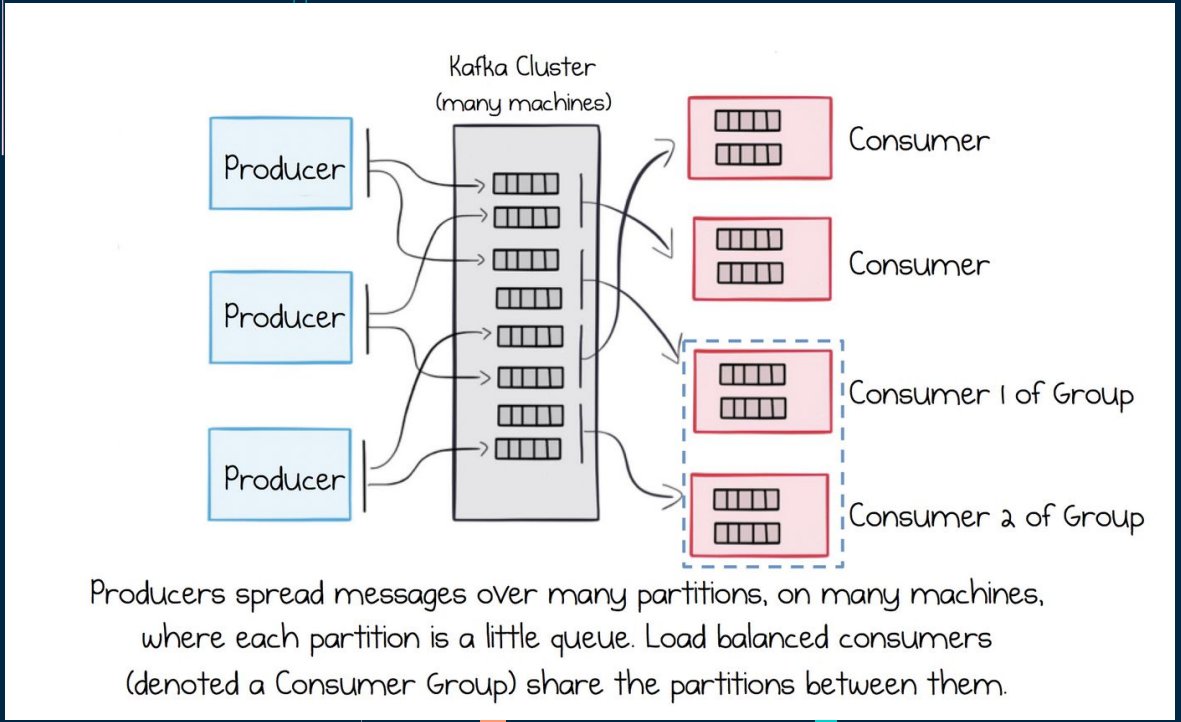
01

Simplified view of Hyperledger Fabric Transaction Flow



Credits:-https://aws.amazon.com/blockchain/what-is-hyperledger-fabric/

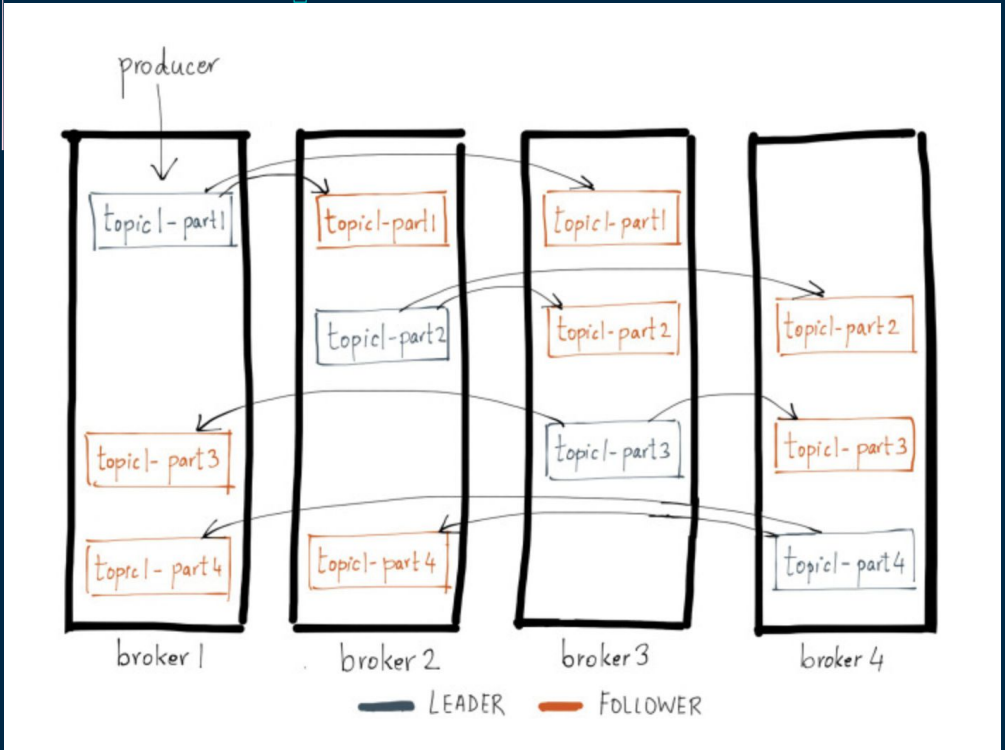
Kafka in Hyperledger Fabric Ordering Service (CFT)



What is Kafka?

Credits: https://www.confluent.io/blog/apache-kafka-for-service-architectures/

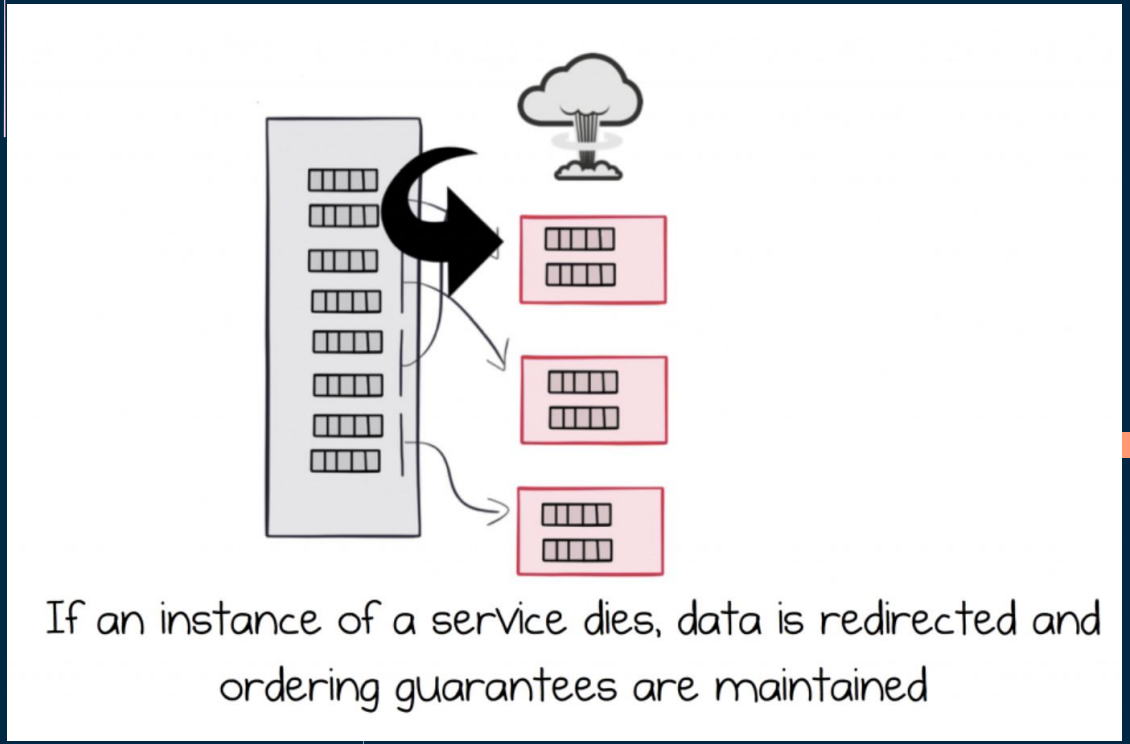
Kafka in Hyperledger Fabric Ordering Service (CFT)



Kafka's brokers, topics and partitions

Credits: https://www.confluent.io/blog/hands-free-kafka-replication-a-lesson-in-operational-simplicity/

Kafka in Hyperledger Fabric Ordering Service (CFT)

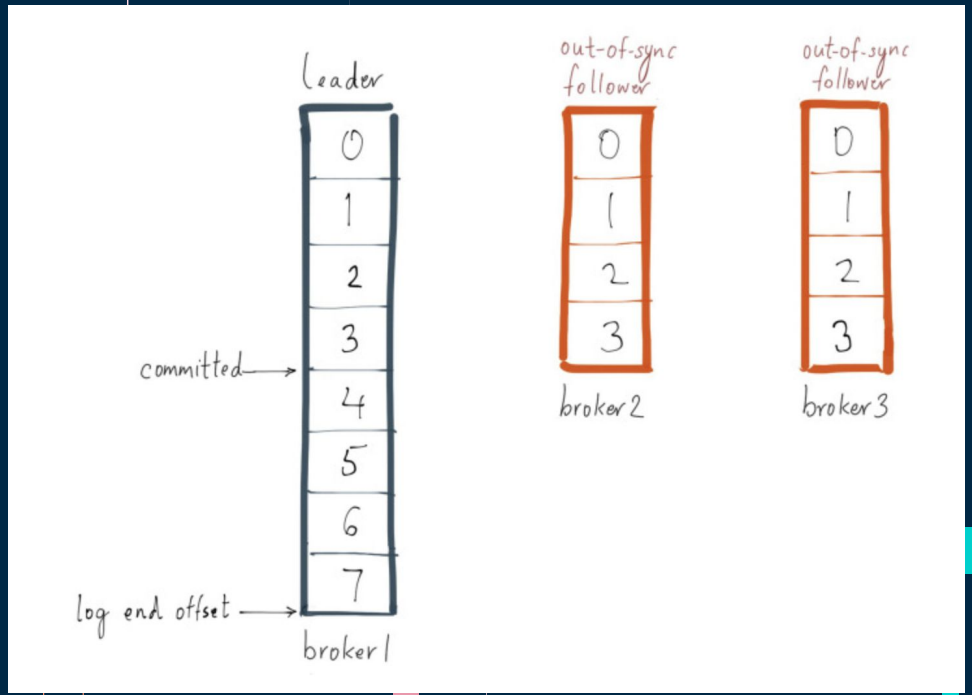


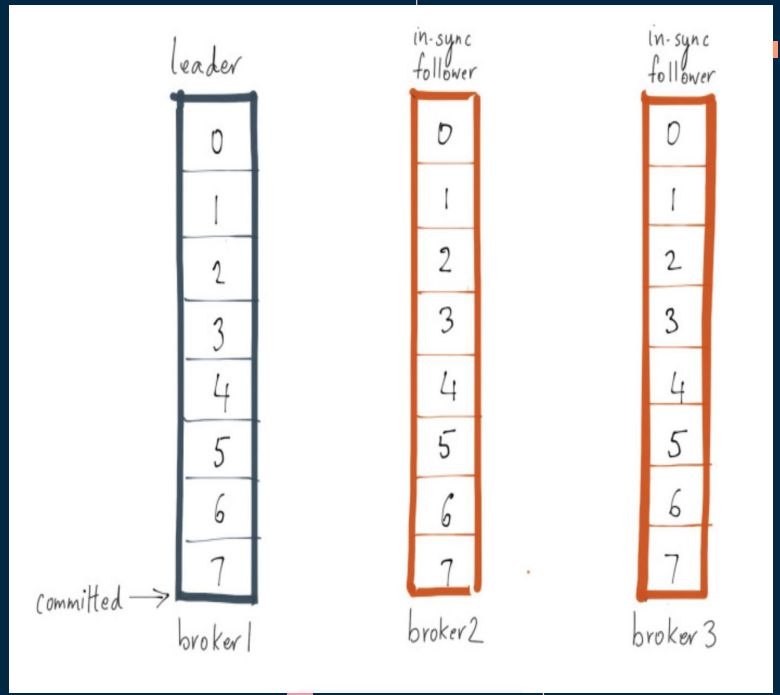
Kafka's Resilience
to Crashes

Credits: https://www.confluent.io/blog/apache-kafka-for-service-architectures/

Kafka in Hyperledger Fabric Ordering Service (CFT)

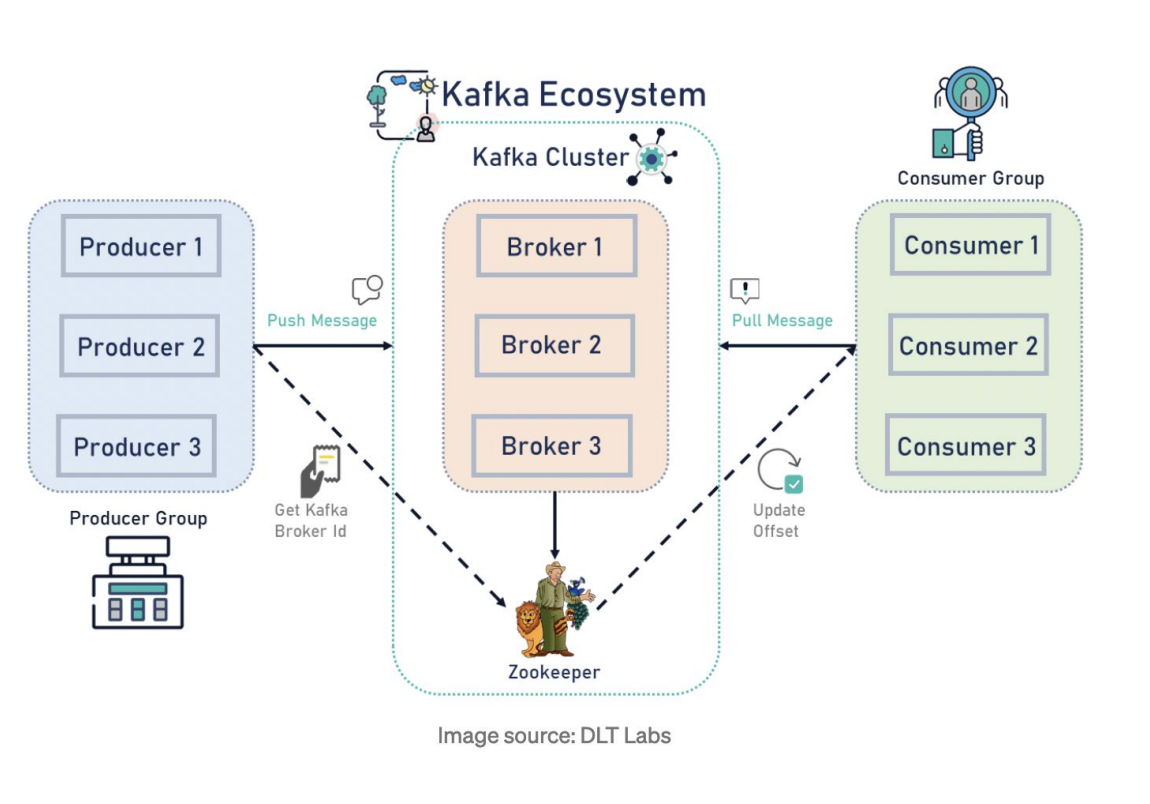
In/Out of Sync Replicas in Kafka





Credits: https://www.confluent.io/blog/hands-free-kafka-replication-a-lesson-in-operational-simplicity/

Kafka in Hyperledger Fabric Ordering Service (CFT)



Kafka with Zookeeper

Credits: https://www.confluent.io/blog/apache-kafka-for-service-architectures/

Kafka in Hyperledger Fabric Ordering Service (CFT)

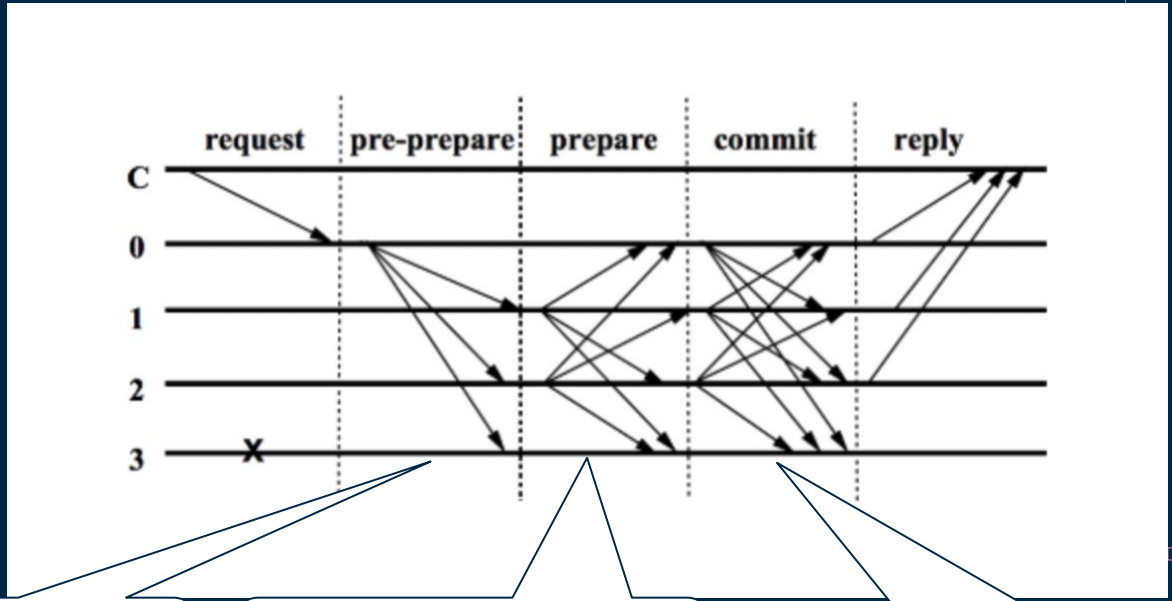
Resilience and Trustworthiness

- Leader does the ordering. Only in-sync replicas can be voted as leader.
- Provides crash fault tolerance

	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/2$	–
Liveness	n	$t < n/2$	–

Credits: https://drops.dagstuhl.de/opus/volltexte/2017/8016/pdf/LIPIcs-DISC-2017-1.pdf

PBFT implementation of Hyperledger (BFT)



Make sure I don't receive same sequence number for different messages

I know that no one received the same sequence number for different messages

Everyone know that no one received the same sequence number for different messages

Credits:- https://pmg.csail.mit.edu/papers/osdi99.pdf

PBFT implementation of Hyperledger (BFT)

Resilience and Trustworthiness

- All instances do ordering
- Provides Byzantine fault tolerance

	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/3$	$f < n/3$
Liveness	n	$t < n/3$	$f < n/3$

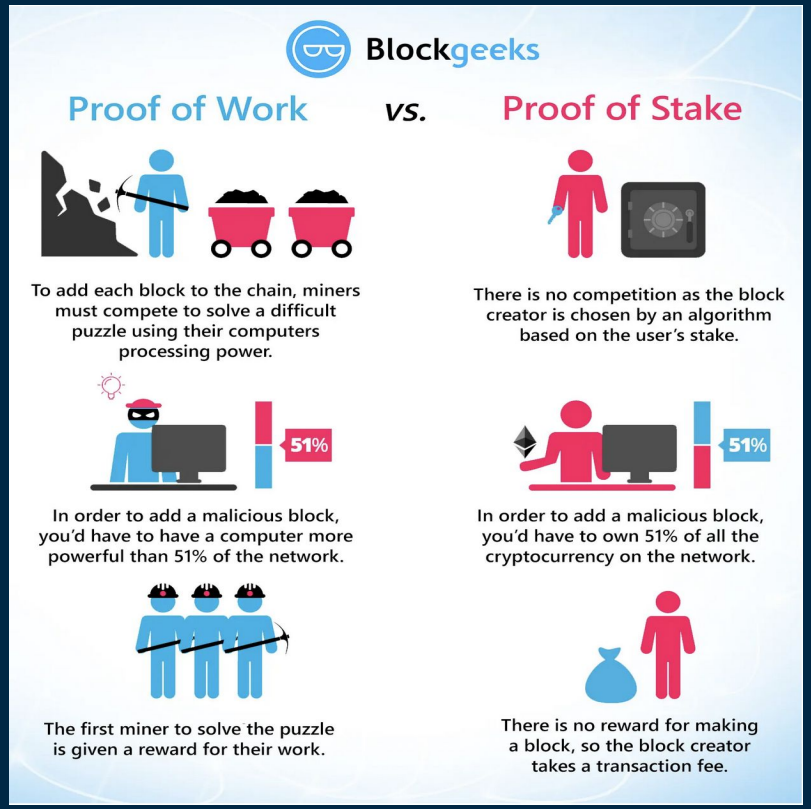
Credits: https://drops.dagstuhl.de/opus/volltexte/2017/8016/pdf/LIPIcs-DISC-2017-1.pdf

TENDERMINT

02

Tendermint

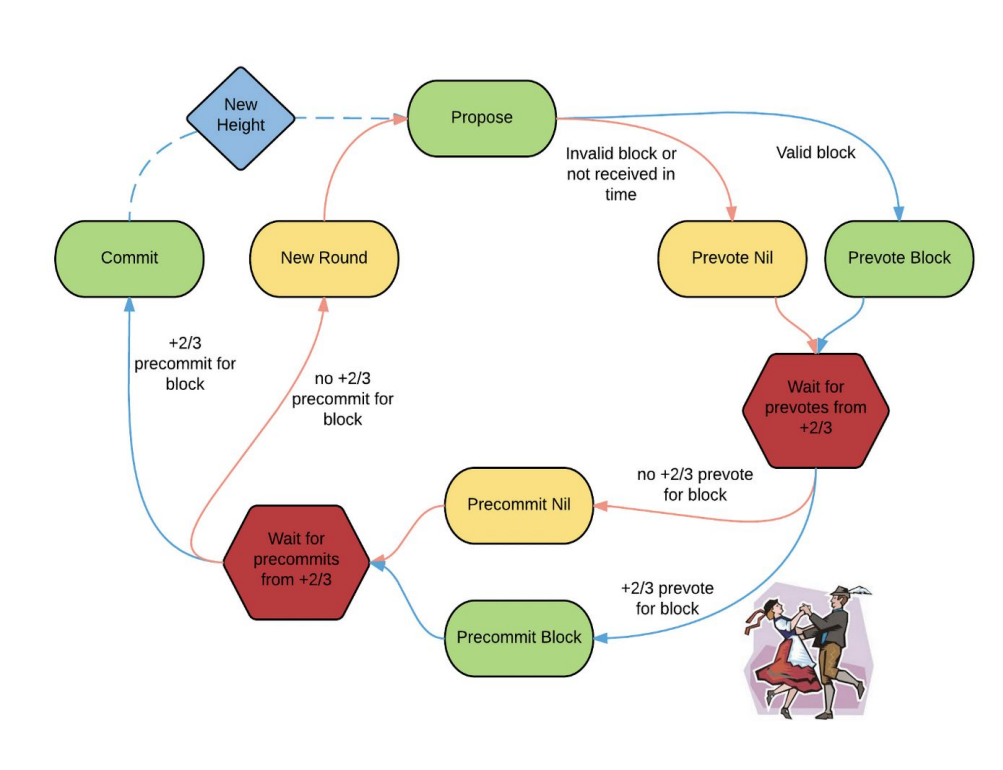
Motivation

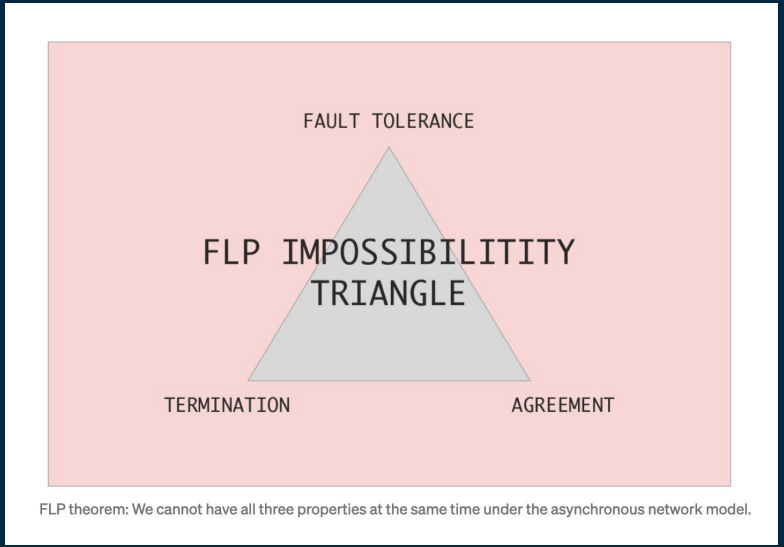


Credits:- https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/

Tendermint

Protocol flow and properties



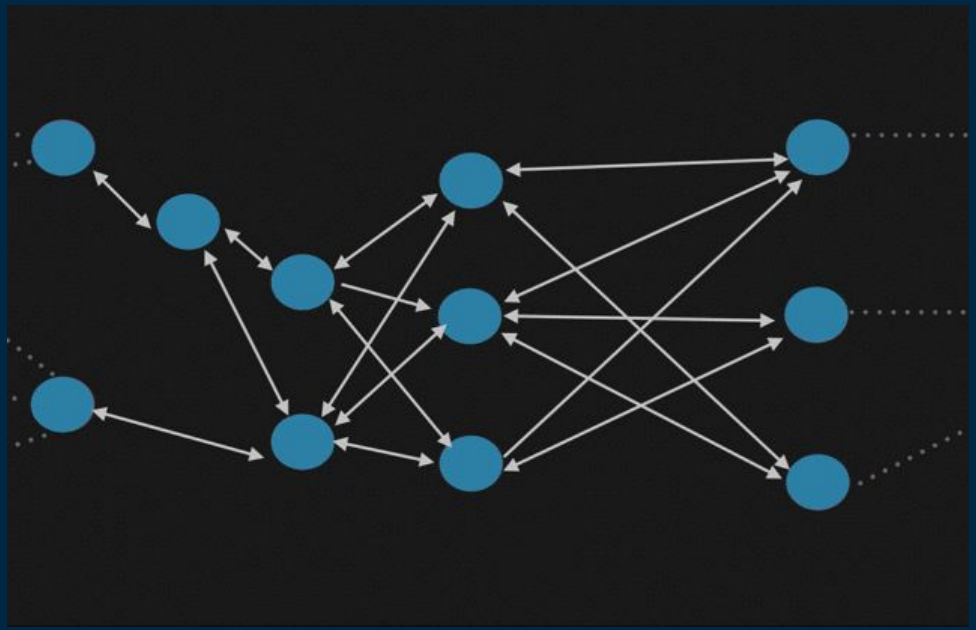


Credits:- https://medium.com/softblocks/explaining-how-tendermint-consensus-works-433066cbc465

Tendermint

Differences from PBFT

- Continuous leader change policy
- Optimized for gossip-based communication and is designed for a high number of nodes



Credits:- https://managementfromscratch.wordpress.com/2016/04/01/introduction-to-gossip/

Tendermint

Resilience and Trustworthiness

	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/3$	$f < n/3$
Liveness	n	$t < n/3$	$f < n/3$

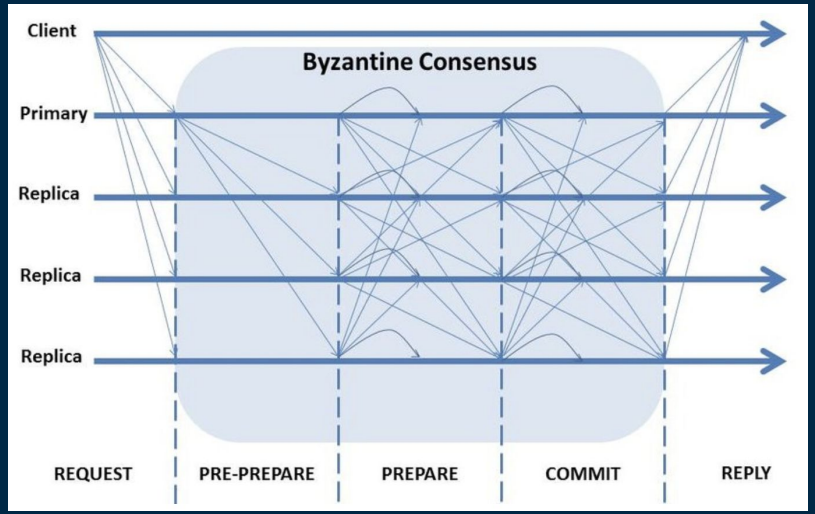
Credits: https://drops.dagstuhl.de/opus/volltexte/2017/8016/pdf/LIPIcs-DISC-2017-1.pdf

SYMBIONT – BFT-SMART

03

Symbiont – BFT-SMaRt

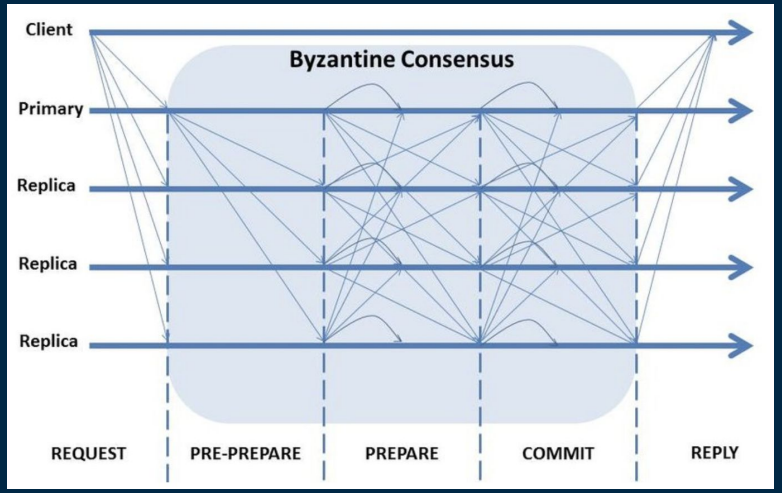
- Symbiont uses its own reimplementation of BFT-SMaRt
- Performance numbers of 80’000 transactions per second

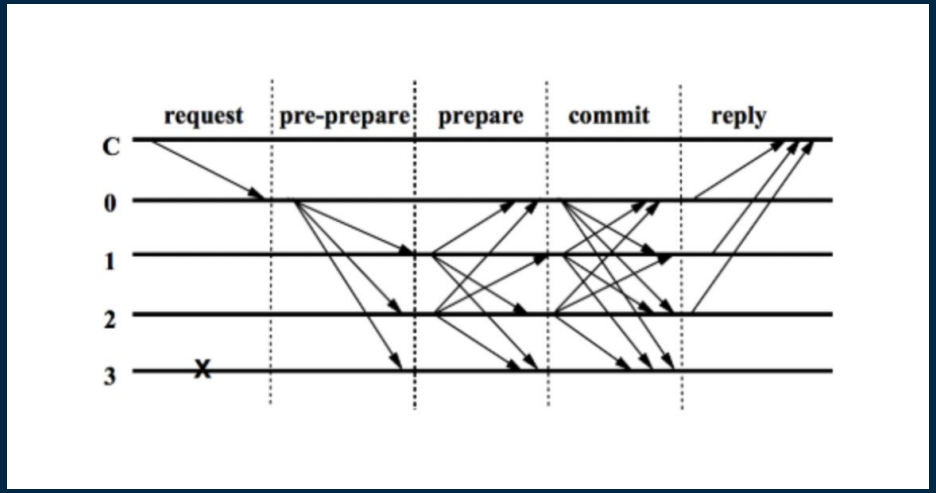


Credits:- https://www.di.fc.ul.pt/~bessani/publications/dsn14-bftsmart.pdf

Symbiont – BFT-SMaRt

Comparison with PBFT

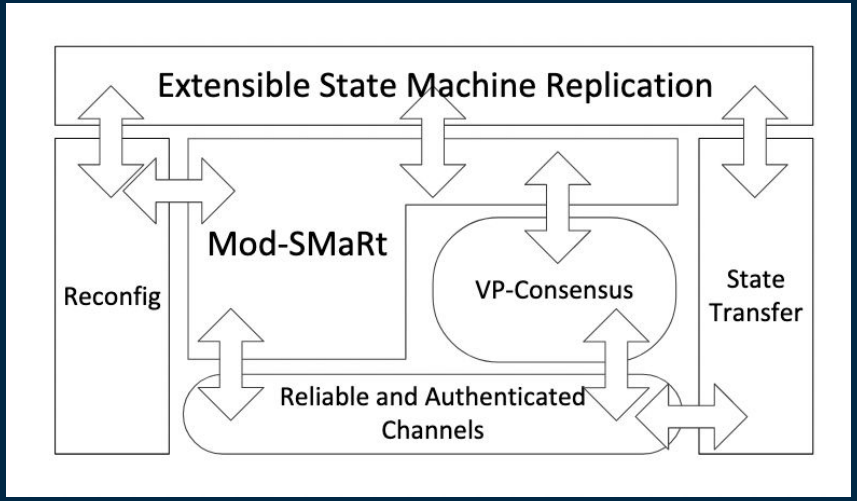


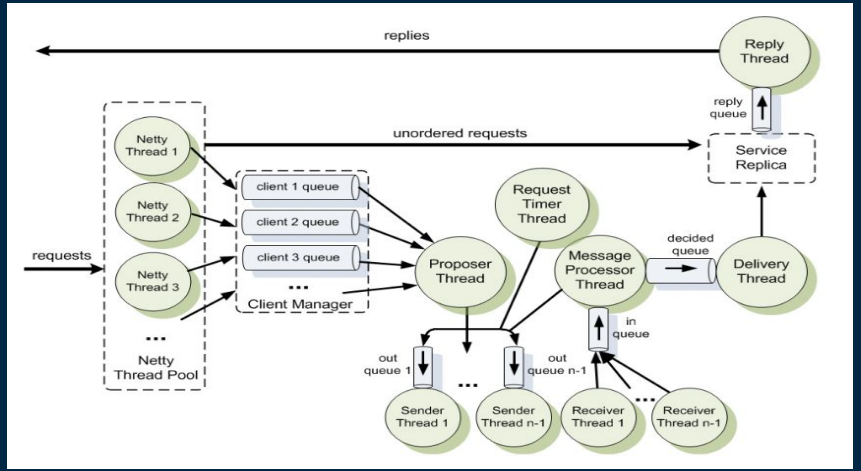


Credits:- https://www.di.fc.ul.pt/~bessani/publications/dsn14-bftsmart.pdf

Symbiont – BFT-SMaRt

Comparison with PBFT





Credits:- https://www.di.fc.ul.pt/~bessani/publications/dsn14-bftsmart.pdf

Symbiont – BFT-SMaRt

Resilience and Trushworthiness

	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/3$	$f < n/3$
Liveness	n	$t < n/3$	$f < n/3$

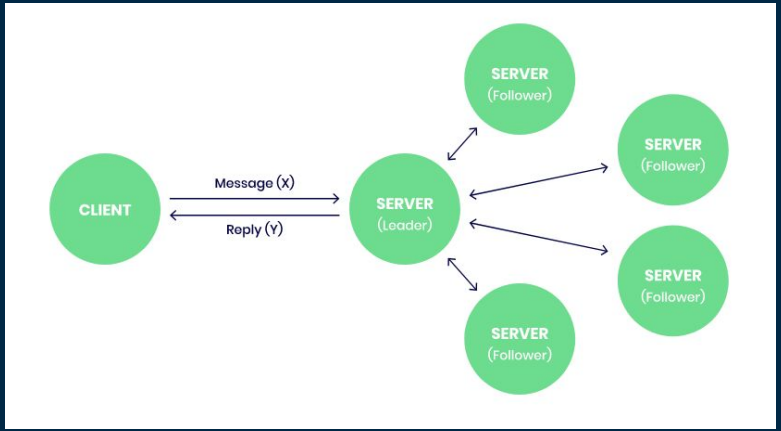
Credits:- Credits: https://drops.dagstuhl.de/opus/volltexte/2017/8016/pdf/LIPIcs-DISC-2017-1.pdf

RAFT

04

OVERVIEW

- Consensus algorithm that is designed to be easy to understand
- Leader Based: At any given time, one server is in charge, others accept its decisions; Clients communicate with the leader

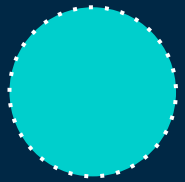


Credits: https://launchpad.settlemint.com/documentation/raft-consensus-protocol

STATES

FOLLOWER

Everyone starts here, totally passive



CANDIDATE

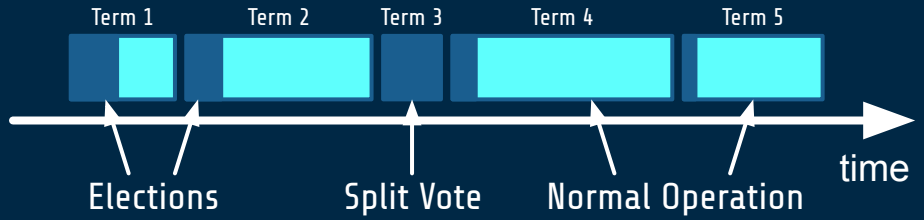
Used for electing new leader

LEADER

Accepts client requests, manages replication

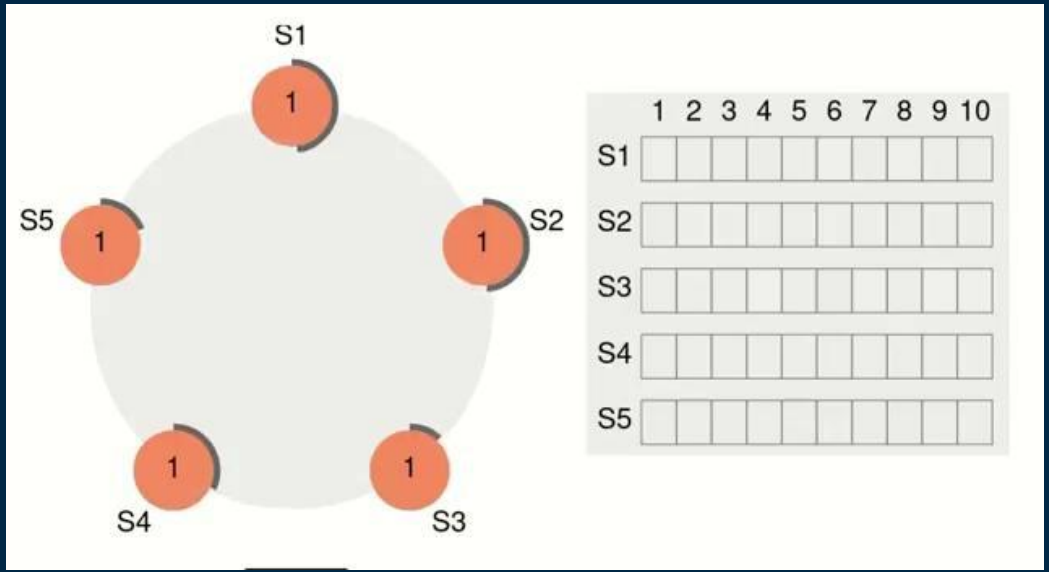
Normal operation: 1 leader, N-1 followers

TERMS, HEARTBEATS & TIMEOUTS



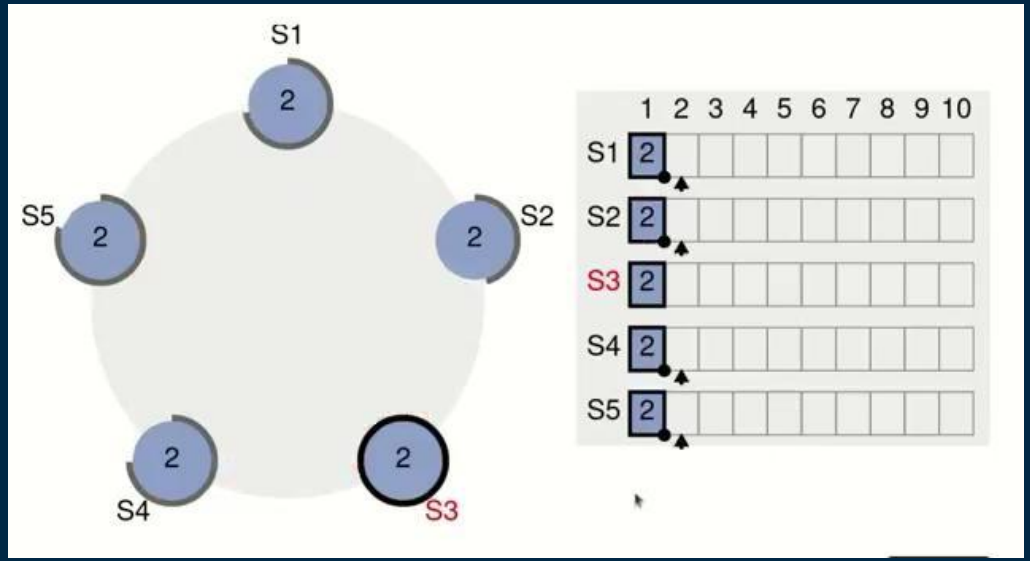
- Servers start up as followers
- Followers expect to receive RPCs from leaders or candidates
- Leaders must send **heartbeats** to maintain authority
- If electionTimeout elapses with no RPCs:
  - Follower assumes leader has crashed
  - Follower starts new election

LEADER ELECTION

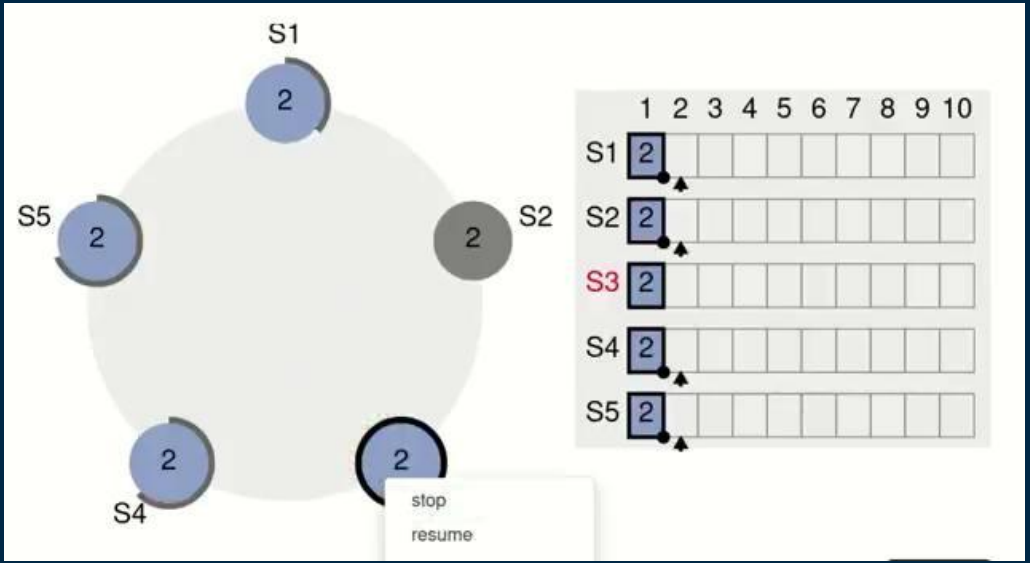


Credits: https://raft.github.io/

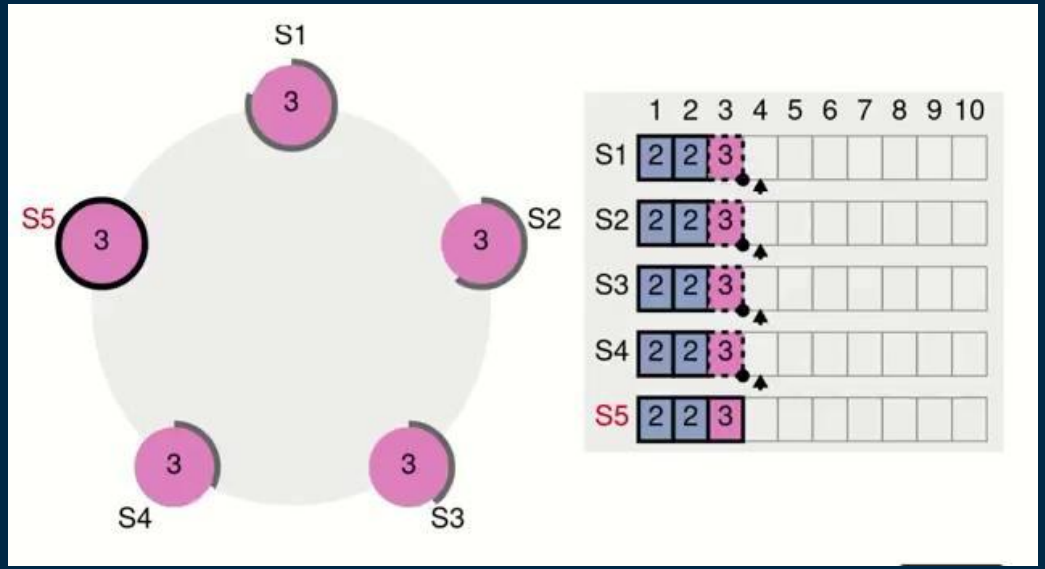
COMMITING A TRANSACTION



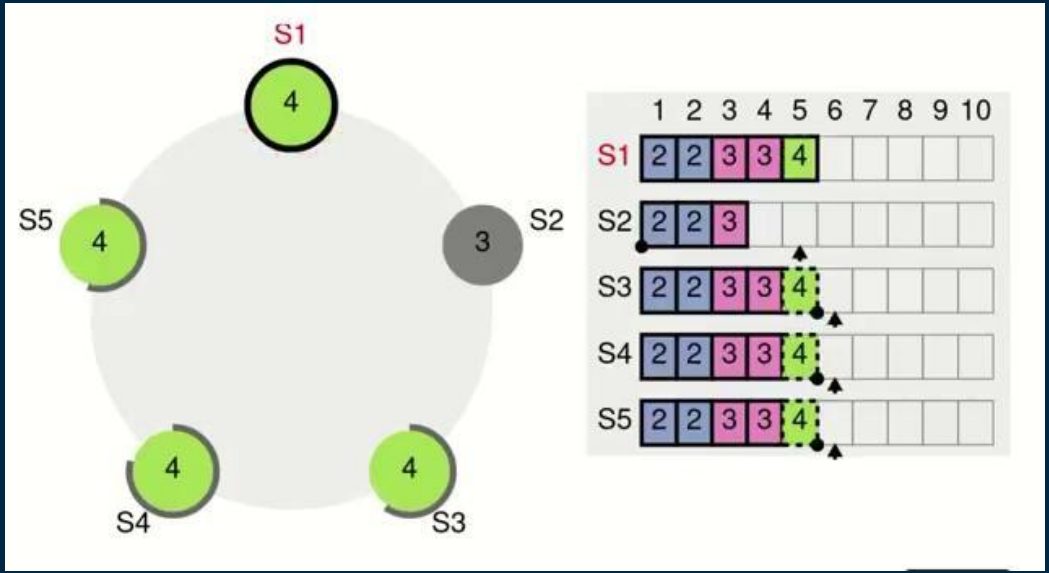
NODE FAILURE



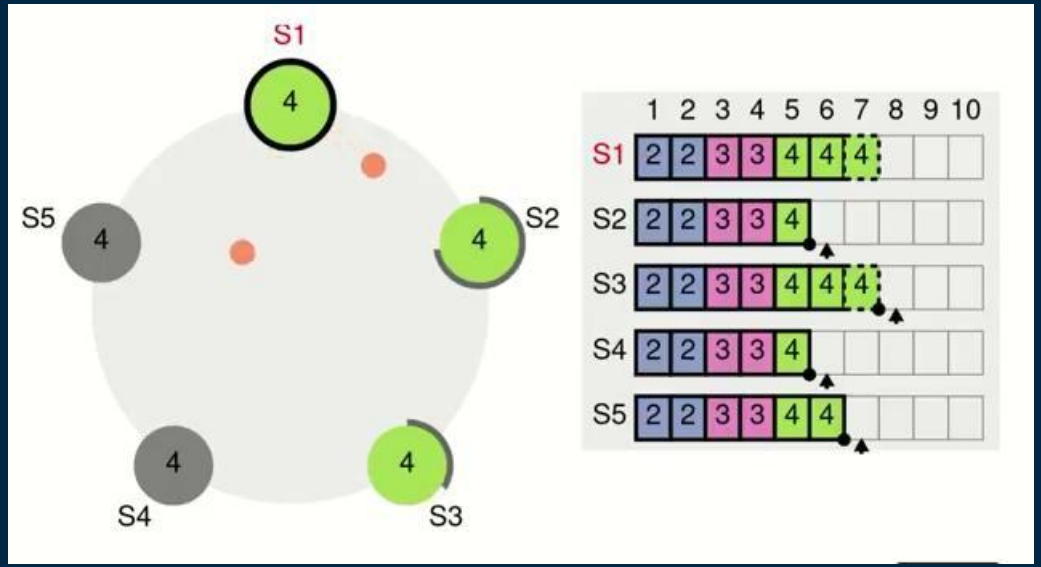
TIMEOUT - NEW ELECTION



NODE FAILURE

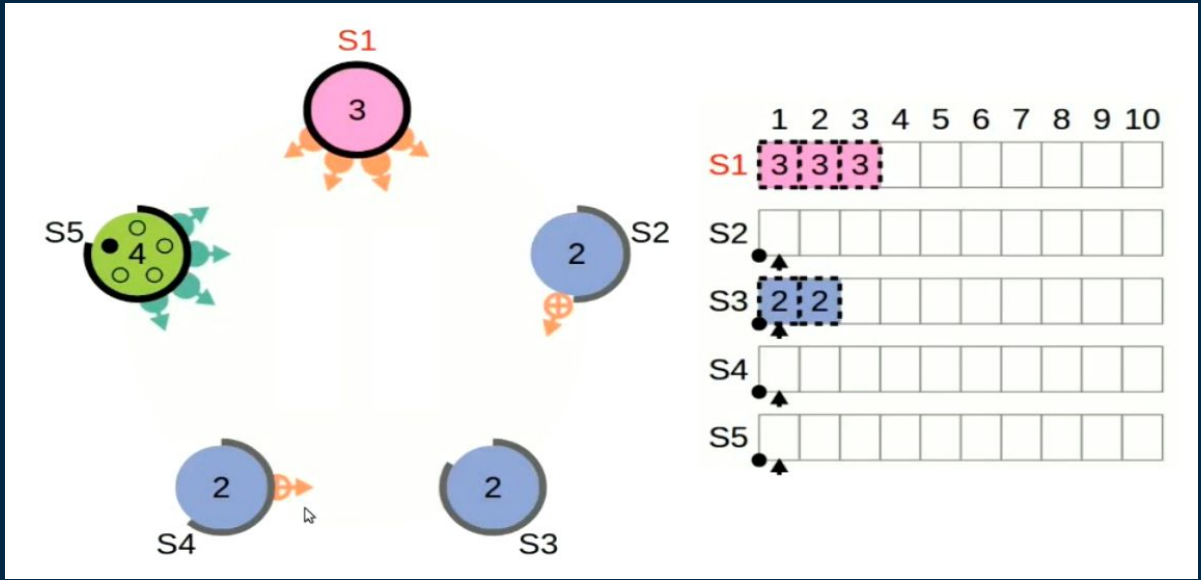


NODE FAILURE - TEMP MINORITY



NO VOTE FOR BAD CANDIDATES

Followers don't vote for candidates with worse terms than them



SAFETY AND LIVENESS

Safety – At most one winner per term

B can't also get majority

Voted for candidate A

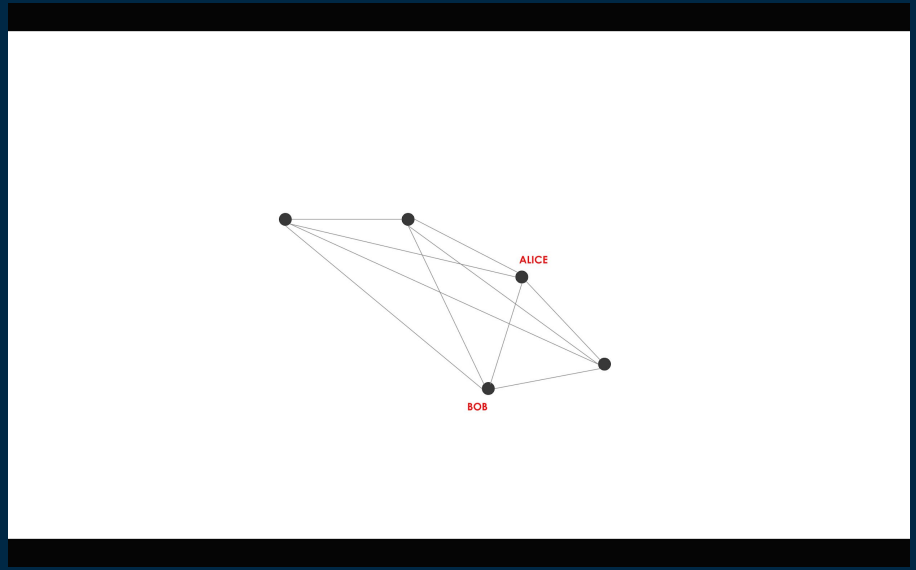
Liveness – Someone eventually wins

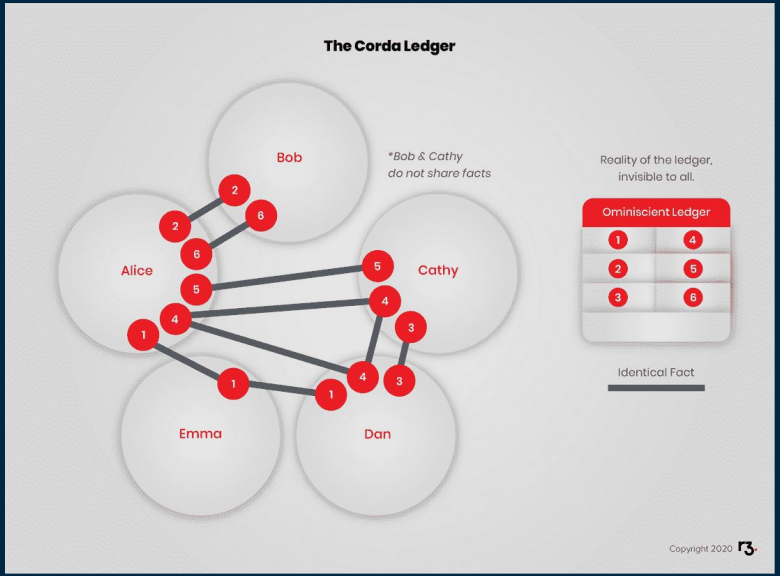
Works if Timeout (T) is much larger than message broadcast time (t)

$$T >> t$$

R3 CORDA 05

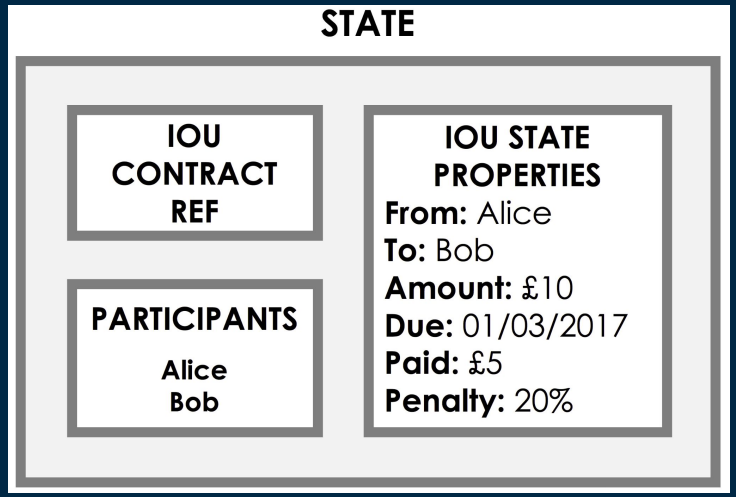
OVERVIEW

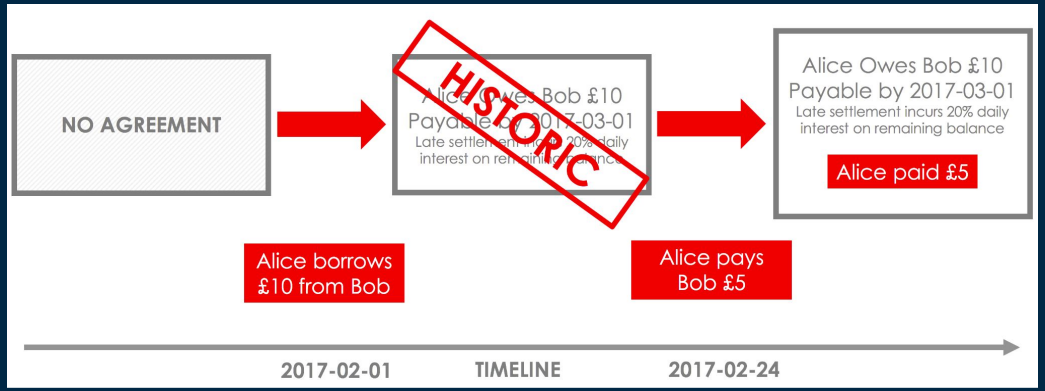




Credits: https://docs.r3.com/en/platform/corda/4.5/open-source/key-concepts.html

STATES & STATE SEQUENCES

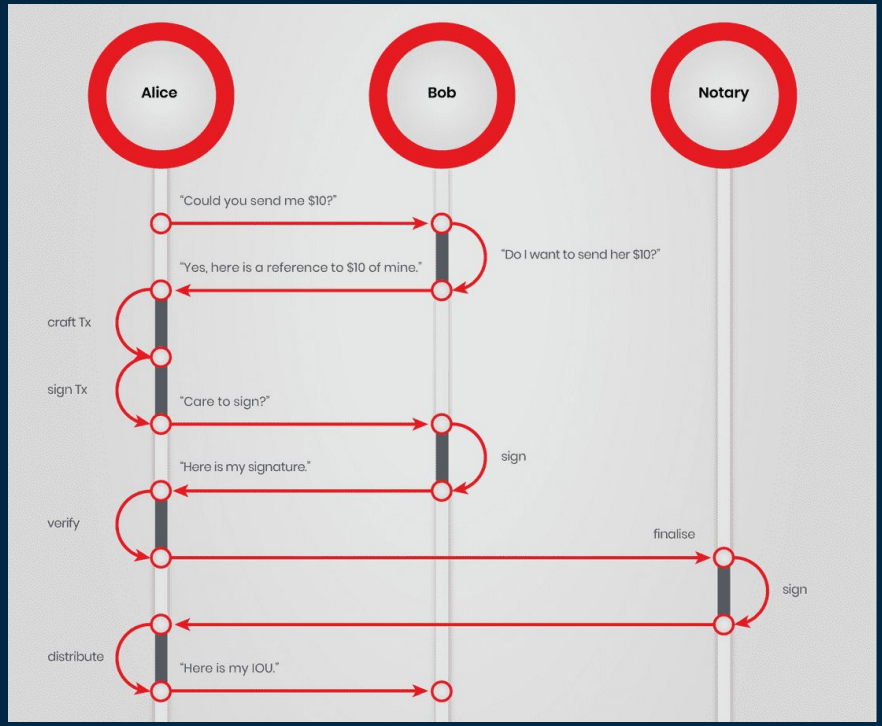




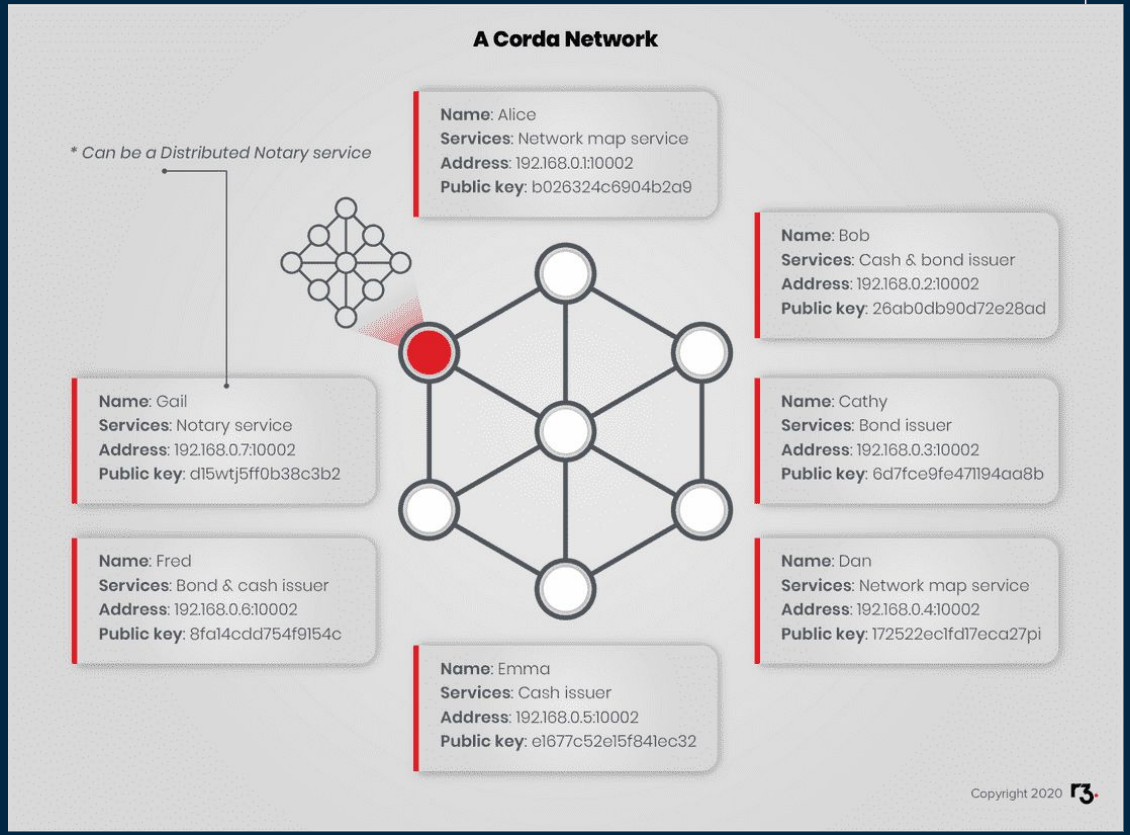
Credits: https://docs.r3.com/en/platform/corda/4.5/open-source/key-concepts.html

CONSENSUS

- **Validity consensus**: Proposed transaction has valid input and output states and required signatures
- **Uniqueness consensus**: There exists no other transaction, over which there is previously reached consensus – Notary required for that



Credits: https://docs.r3.com/en/platform/corda/4.5/open-source/key-concepts.html



Credits: https://docs.r3.com/en/platform/corda/4.5/open-source/key-concepts.html

RESILIENCE

	Notary nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/2$	–
Liveness	n	$t < n/2$	–

RAFT Based notary service

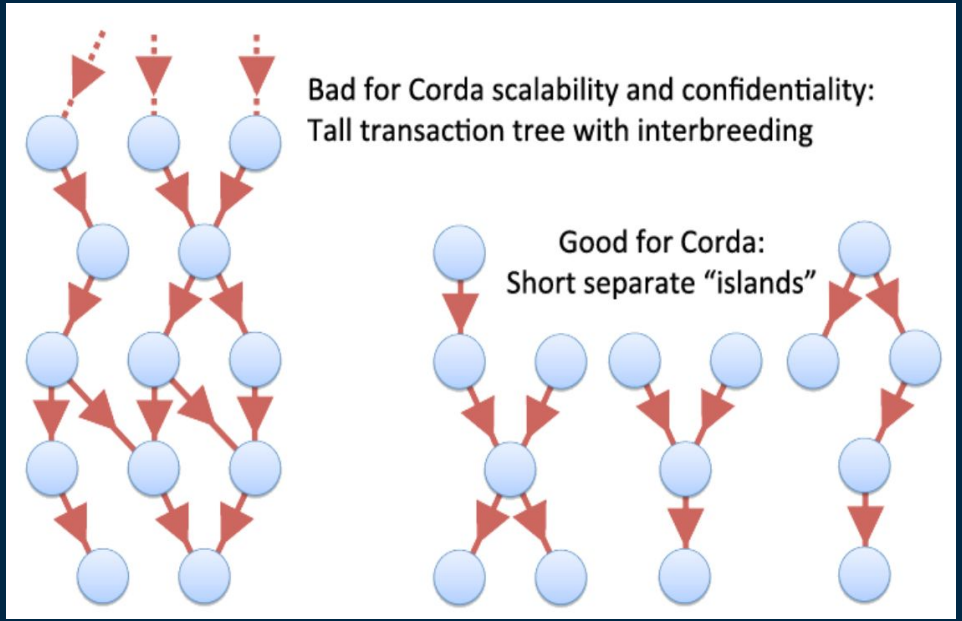
	Notary nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/3$	$f < n/3$
Liveness	n	$t < n/3$	$f < n/3$

BFT-SMaRt-based notary service

Credits: https://drops.dagstuhl.de/opus/volltexte/2017/8016/pdf/LIPIcs-DISC-2017-1.pdf

ISSUES

- Confidentiality - If Alice pays Bob $10, then Bob sends that $10 on to Charlie, Charlie's node has to be shown the transaction between Alice and Bob, even though it doesn't involve him.
- Scalability - Tall transaction trees would cause considerable transaction delay. In blockchain, nodes have to verify transactions that might not be relevant to them.

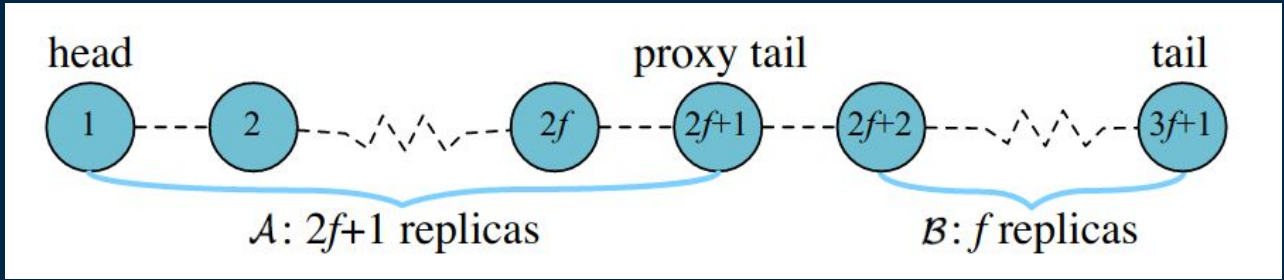


Credits: https://launchpad.settlemint.com/documentation/raft-consensus-protocol

B-CHAIN

06

OVERVIEW



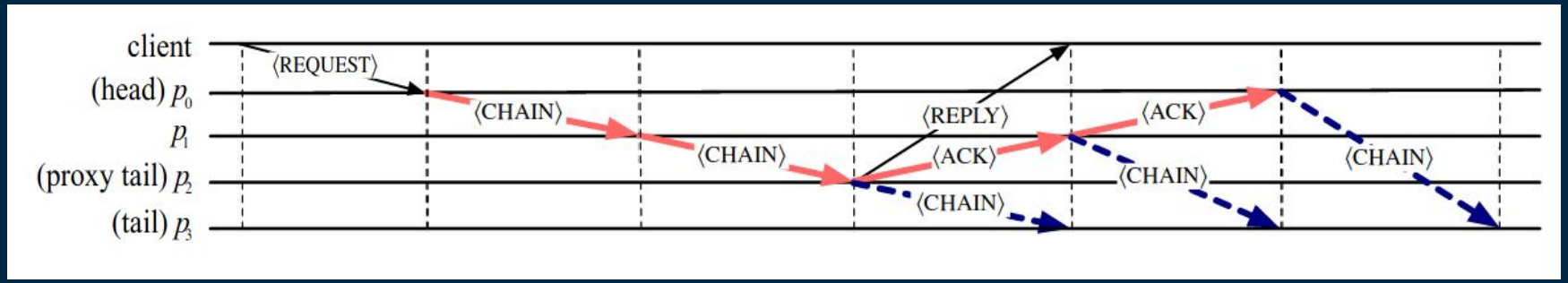
Credits: https://www.cs.ucdavis.edu/~peisert/research/2014-OPODIS-BChain.pdf

Predecessor set P - A replica in the first f + 1 replicas -> all the preceding replicas
For every other replica -> the preceding f + 1 replicas in the chain

Successor set S - If one of the last f + 1 replicas -> all the subsequent replicas
For every other replica -> subsequent f + 1 replicas.

OVERVIEW

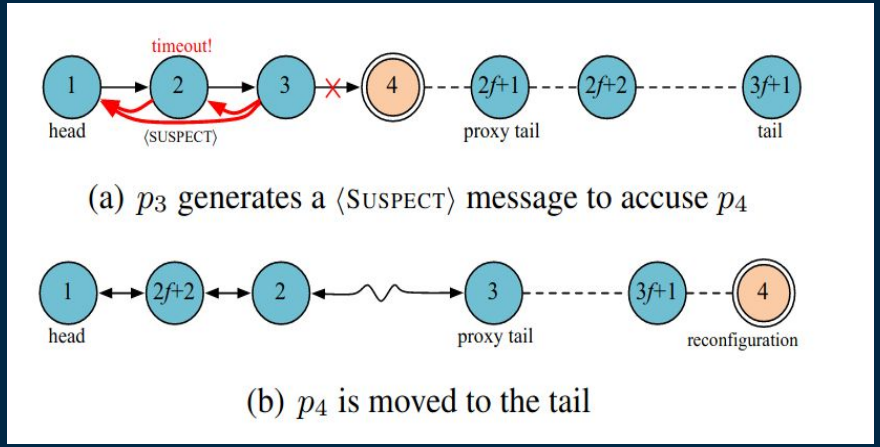
- The first 2f + 1 replicas reach agreement; last f replicas just update their states
- 2 types of messages - <CHAIN> and <ACK>
- Request is executed on accepting⟨CHAIN⟩message - check for valid signatures by replicas in P
- Request is committed on accepting the⟨ACK⟩message - check for valid signatures by replicas in S
- The client completes the request if it receives a⟨REPLY⟩message from the proxy tail with signatures by the last f + 1 replicas in the chain. Otherwise, it retransmits the request to all replicas.



Credits: https://www.cs.ucdavis.edu/~peisert/research/2014-OPODIS-BChain.pdf

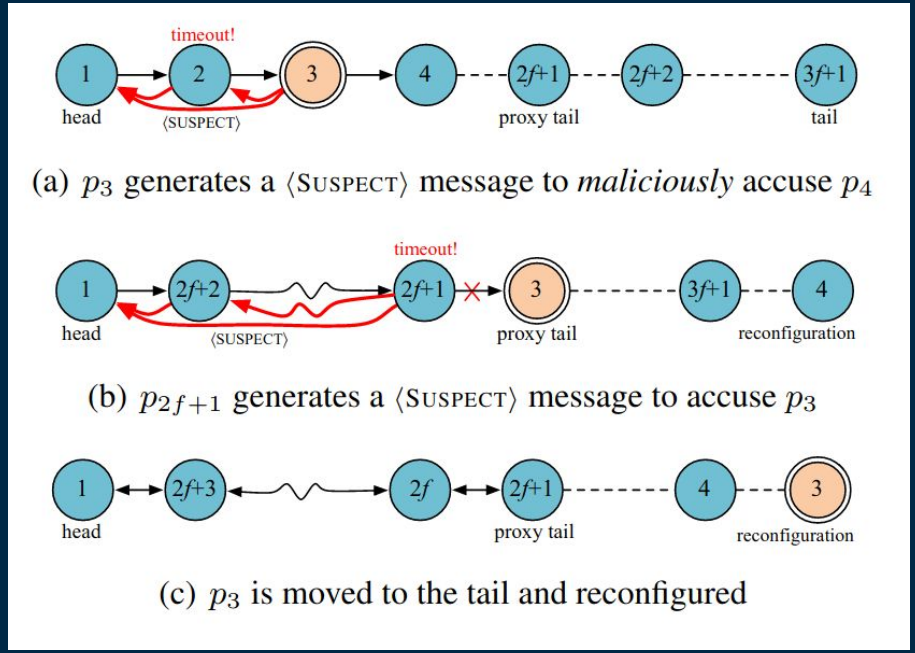
FAILURE DETECTION

- Every replica starts a timer after sending a ⟨CHAIN⟩message.
- Unless an⟨ACK⟩is received before the timer expires, it sends a⟨SUSPECT⟩message to the head and also along the chain towards the head.
- Upon seeing⟨SUSPECT⟩messages, the head starts the re-chaining, by moving faulty replicas to set B.
- If the head receives multiple ⟨SUSPECT⟩ messages, only the one closest to the proxy tail is handled.



Credits: https://www.cs.ucdavis.edu/~peisert/research/2014-OPODIS-BChain.pdf

FAILURE DETECTION



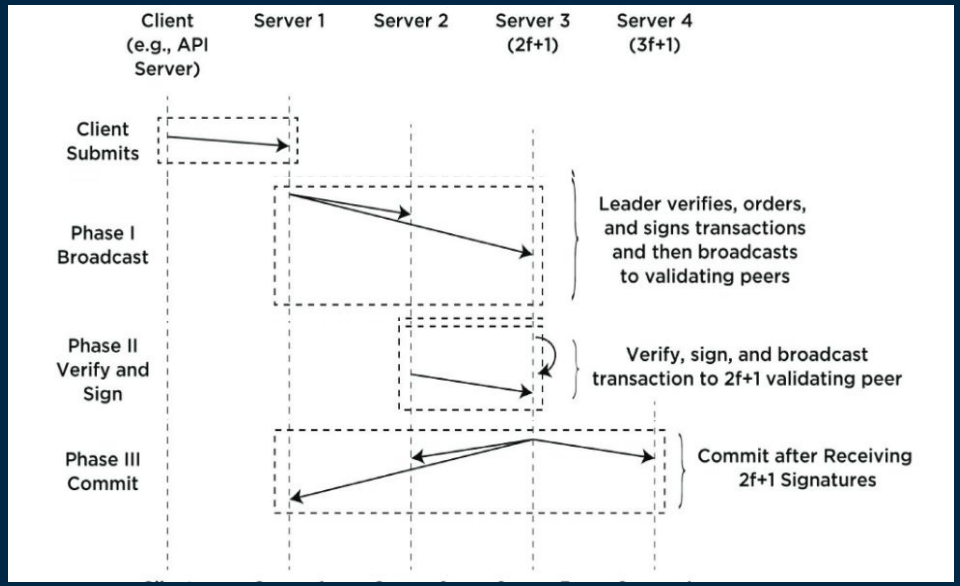
Credits: https://www.cs.ucdavis.edu/~peisert/research/2014-OPODIS-BChain.pdf

IROHA - SUMERAGI

07

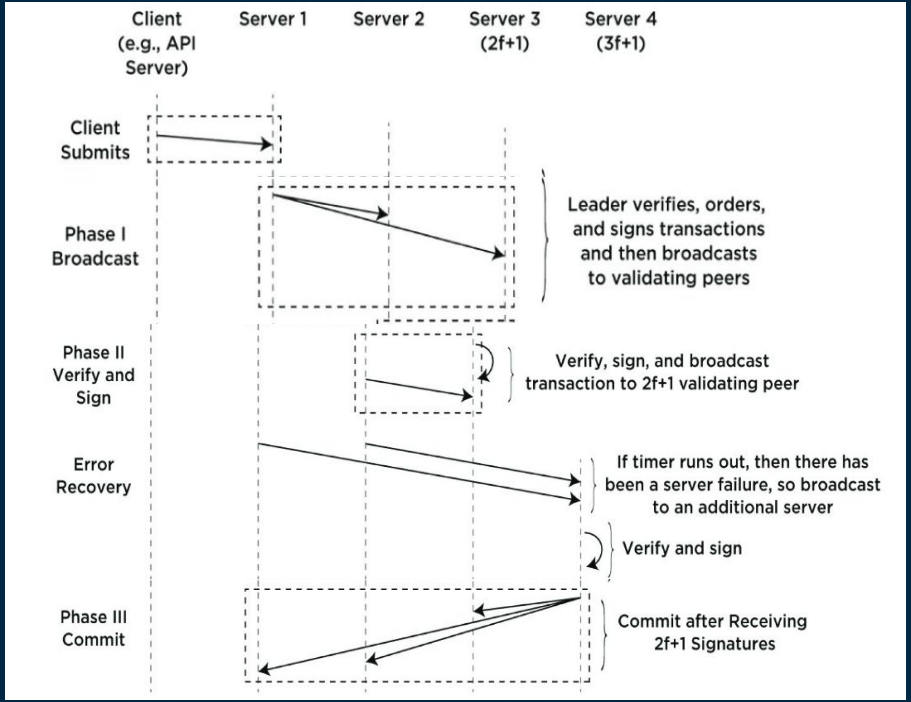
OVERVIEW

- Under the Hyperledger Project, tolerates upto f Byzantine faulty nodes
- “Heavily inspired” by BChain



Credits: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

FAILURE HANDLING



Credits: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

RESILIENCE

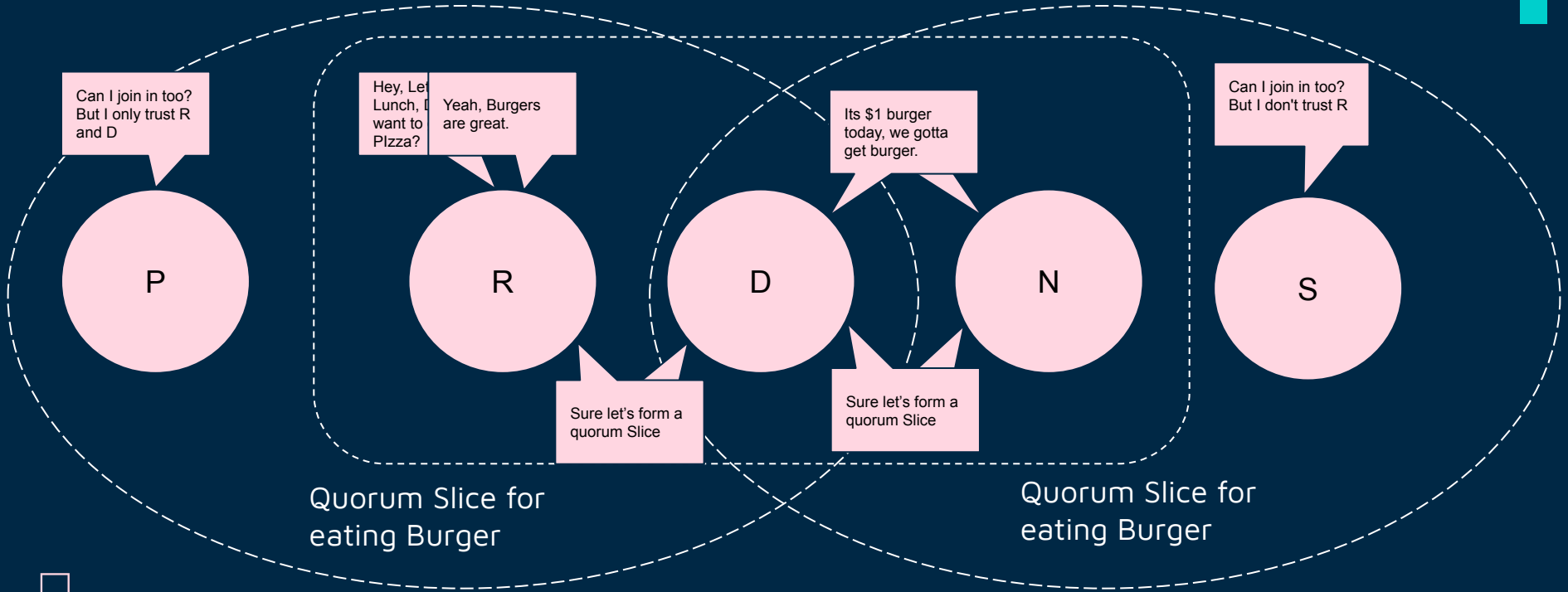
	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/3$	$f < n/3$
Liveness	n	$t < n/3$	$f < n/3$

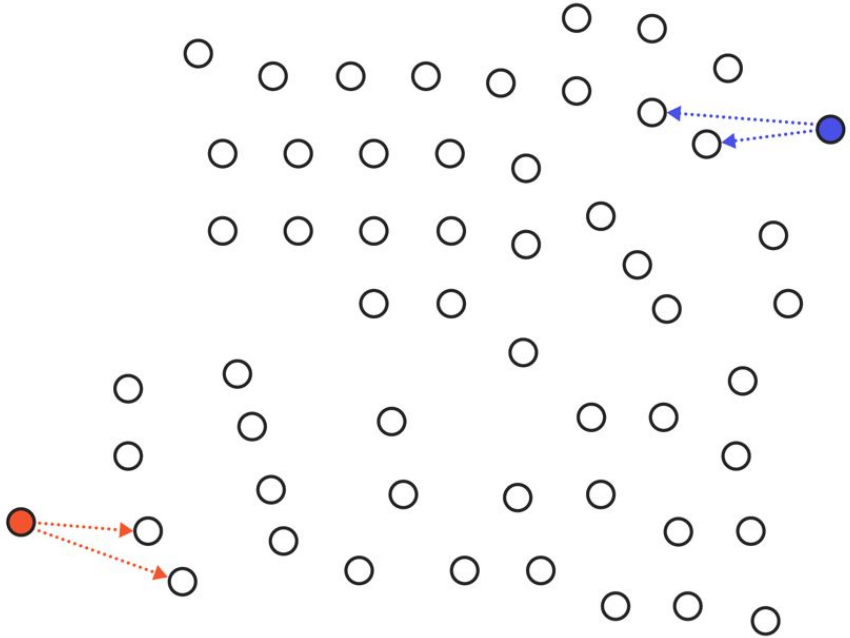
Credits: https://drops.dagstuhl.de/opus/volltexte/2017/8016/pdf/LIPIcs-DISC-2017-1.pdf

CHAIN - FEDERATED CONSENSUS

08

Quorum and quorum slice





Overview

1. For an institutional consortium to **issue and transfer financial assets** on **permissioned blockchain**
2. Uses a federated consensus protocol to reach consensus N nodes in the network
3. One of the nodes is configured as "**block generator**"
   - Periodically assembles non-executed transaction into blocks -> submit for approval to block signer
4. Other nodes are called "**block signers**"
   - Validates the block , checking the signature of the generator ->sign endorsement for the block
5. When a node receives q endorsement for a block (q = 2f + 1) -> appends the block to its chain

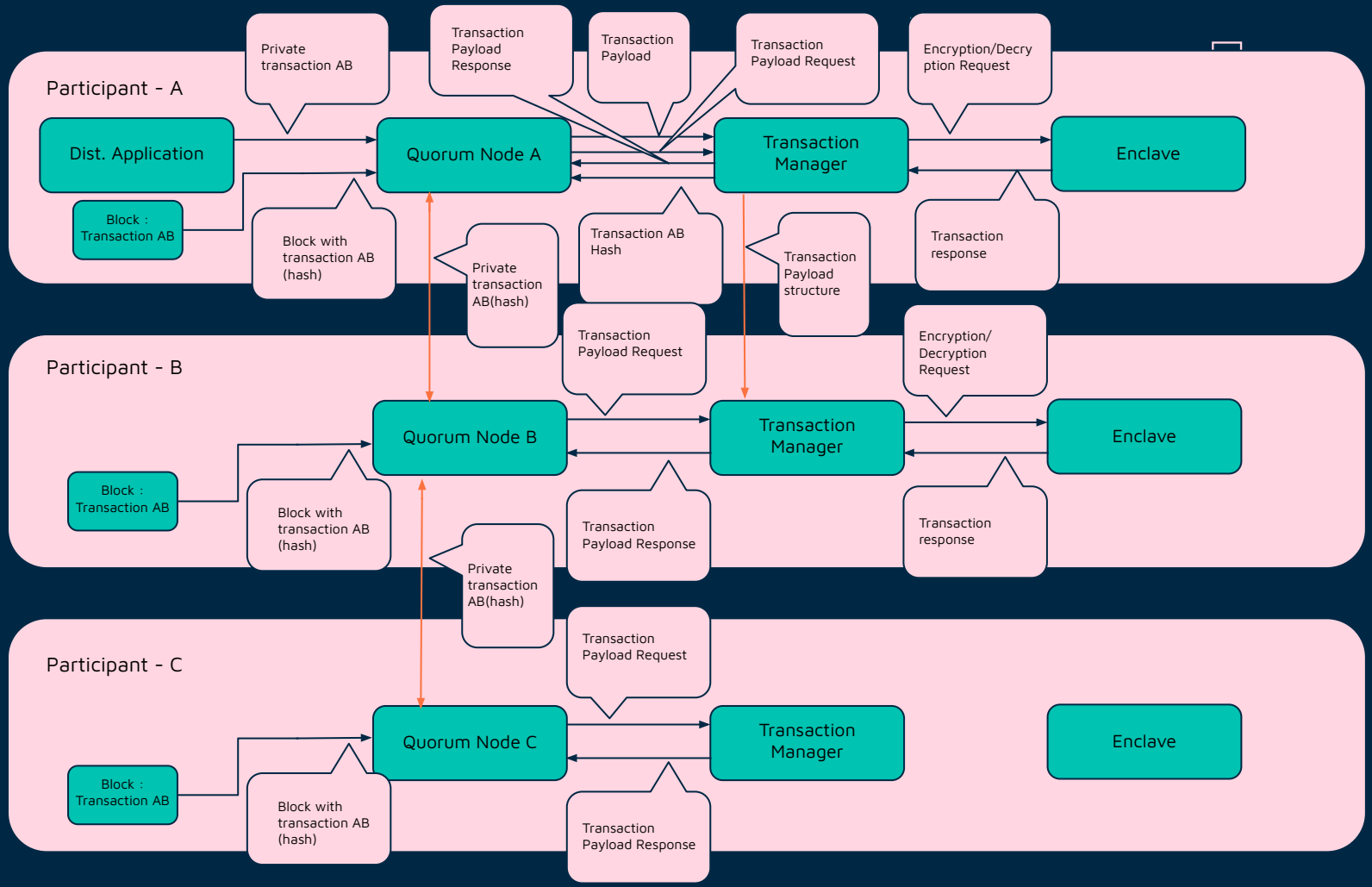
	Generic nodes	Any t nodes crash	Any f nodes subverted	Special nodes	Any s special nodes crash	Special nodes subverted
Safety	n	$t < n/3$	$f < n/3$	$m = 1$	–	–
Liveness	n	$t < n/3$	$f < n/3$	$m = 1$	–	–

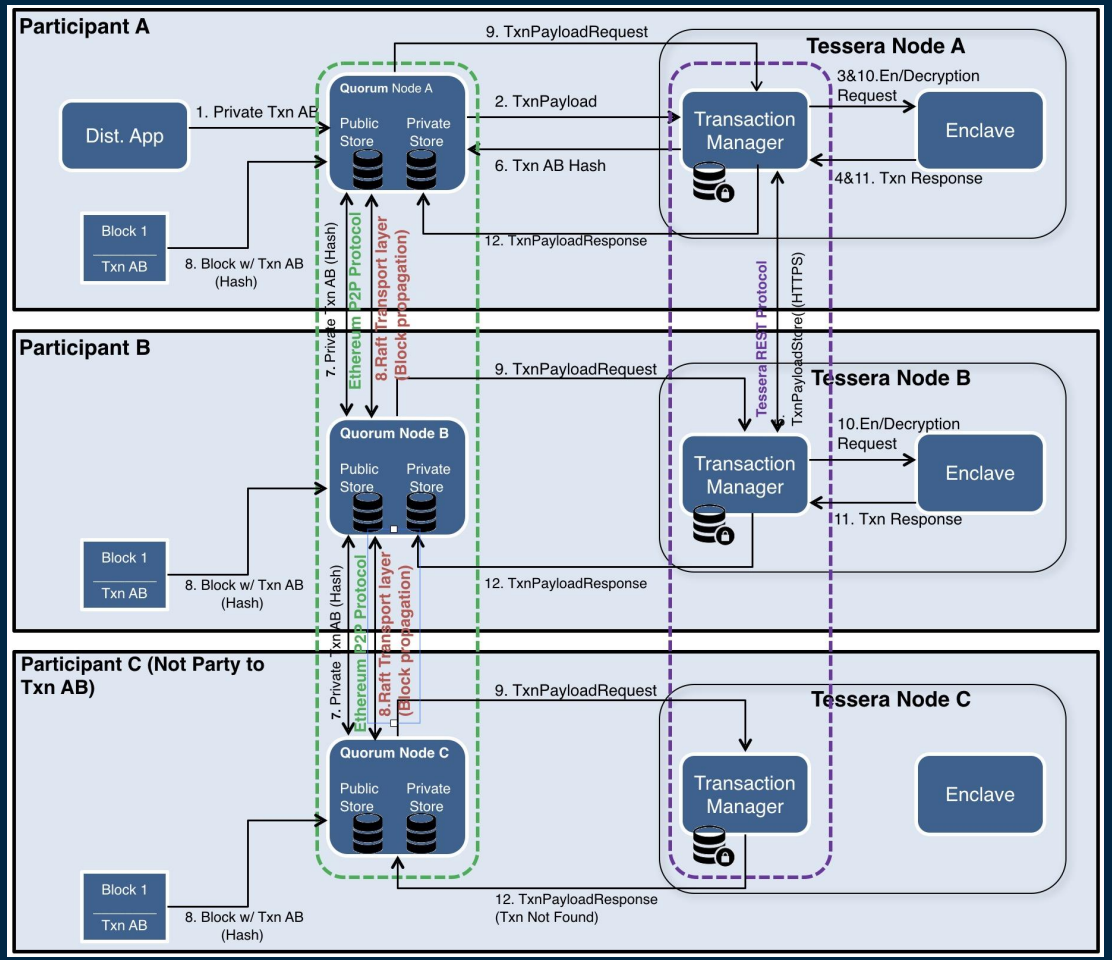
QUORUMCHAIN 09

Overview

- Uses **peer-to-peer gossip layer of Etherium.**
- Logic is formulated as a **smart contract** deployed with the **genesis block**.
- Nodes digitally sign every message they send.
- Block makers : permitted to propose block to be appended. Waits for random time and then create sign and propagate a new block on the chain.
- Voters: Validate the block and express their approval. Block with most votes appended in case there are multiple block with votes above threshold.

	Generic nodes	Any t nodes crash	Any f nodes subverted	Special nodes	Any s special nodes crash	Special nodes subverted
Safety	n	$t < n/3$	$f < n/3$	$m = 1$	–	–
Liveness	n	$t < n/3$	$f < n/3$	$m = 1$	–	–


Participant - A
Private transaction AB
Transaction Payload Response
Transaction Payload
Transaction Payload Request
Encryption/Decry ption Request
Dist. Application
Quorum Node A
Transaction Manager
Enclave
Block : Transaction AB
Block with transaction AB (hash)
Private transaction AB(hash)
Transaction AB Hash
Transaction Payload structure
Transaction response
Participant - B
Transaction Payload Request
Encryption/ Decryption Request
Quorum Node B
Transaction Manager
Enclave
Block : Transaction AB
Block with transaction AB (hash)
Transaction Payload Response
Private transaction AB(hash)
Transaction response
Participant - C
Transaction Payload Request
Quorum Node C
Transaction Manager
Enclave
Block : Transaction AB
Block with transaction AB (hash)
Transaction Payload Response


Participant A
Participant B
Participant C (Not Party to Txn AB)
Dist. App
1. Private Txn AB
Block 1
Txn AB
8. Block w/ Txn AB (Hash)
Quorum Node A
Quorum Node B
Quorum Node C
Public Store
Private Store
2. TxnPayload
6. Txn AB Hash
9. TxnPayloadRequest
12. TxnPayloadResponse
Tessera Node A
Tessera Node B
Tessera Node C
Transaction Manager
Enclave
3&10.En/Decryption Request
4&11. Txn Response
10.En/Decryption Request
11. Txn Response
7. Private Txn AB (Hash)
Ethereum P2P Protocol
8.Raft Transport layer (Block propagation)
Tessera REST Protocol
TxnPayloadStore((HTTPS)
12. TxnPayloadResponse (Txn Not Found)

QUORUM-RAFT 10

- Quorum based on **Raft protocol**.
- Quorum uses **etcd** for its implementations.
- **etcd** is a distributed reliable **key-value** store for the most critical data of a distributed system, with a **focus** on being **Simple, Secure, Fast and Reliable**.
- Raft **replicate transaction** for all participating node : Ensures that each node locally **outputs** the same **sequence of transactions**.

	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety	n	$t < n/2$	–
Liveness	n	$t < n/2$	–

MULTICHAIN

11

Overview

- Bitcoin blockchain is not yet suitable for institutional financial transactions.
- Permissioned blockchains in the financial industry and for multi currency exchanges in a consortium.
- Multichain : A dynamic permissioned model.
- There is a list of permitted nodes in the network at all times and only these node validate and participate in protocol

- Consensus mechanism called mining:
  - Nodes **do not solve computational puzzles**
  - Generate new blocks after waiting for a **random timeout**
  - **Resolves** the dilemma posed by private blockchains, in which **one participant** can **monopolize** the **mining process**
  - Define a parameter 0<= mining diversity <=1:
    - 1 means every permitted node must mine (Round robin)
    - 0 means no restriction

	Generic nodes	Any t nodes crash	Any f nodes subverted
Safety*	n	$t < n$	–
Liveness	n	$t < n$	–

Thank you