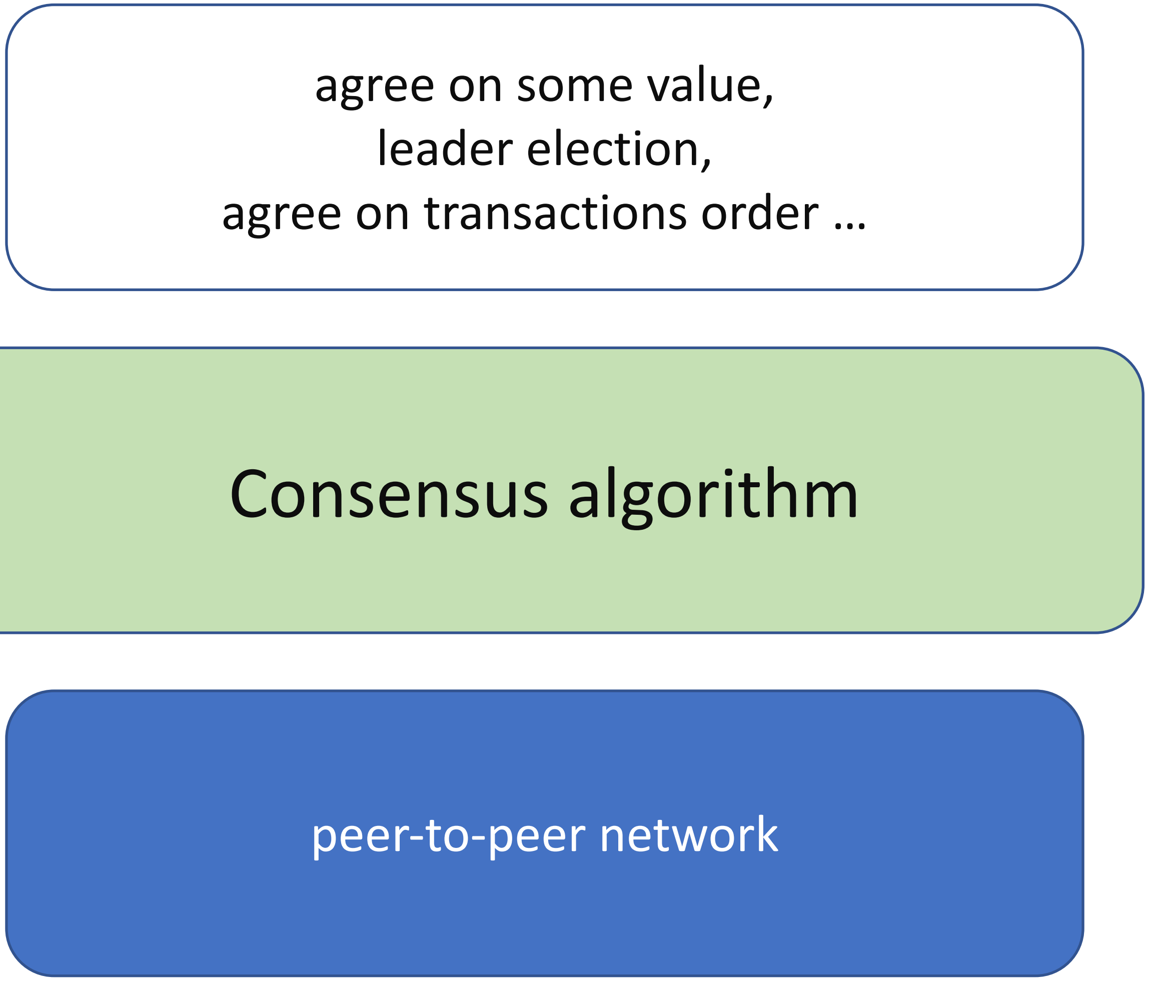
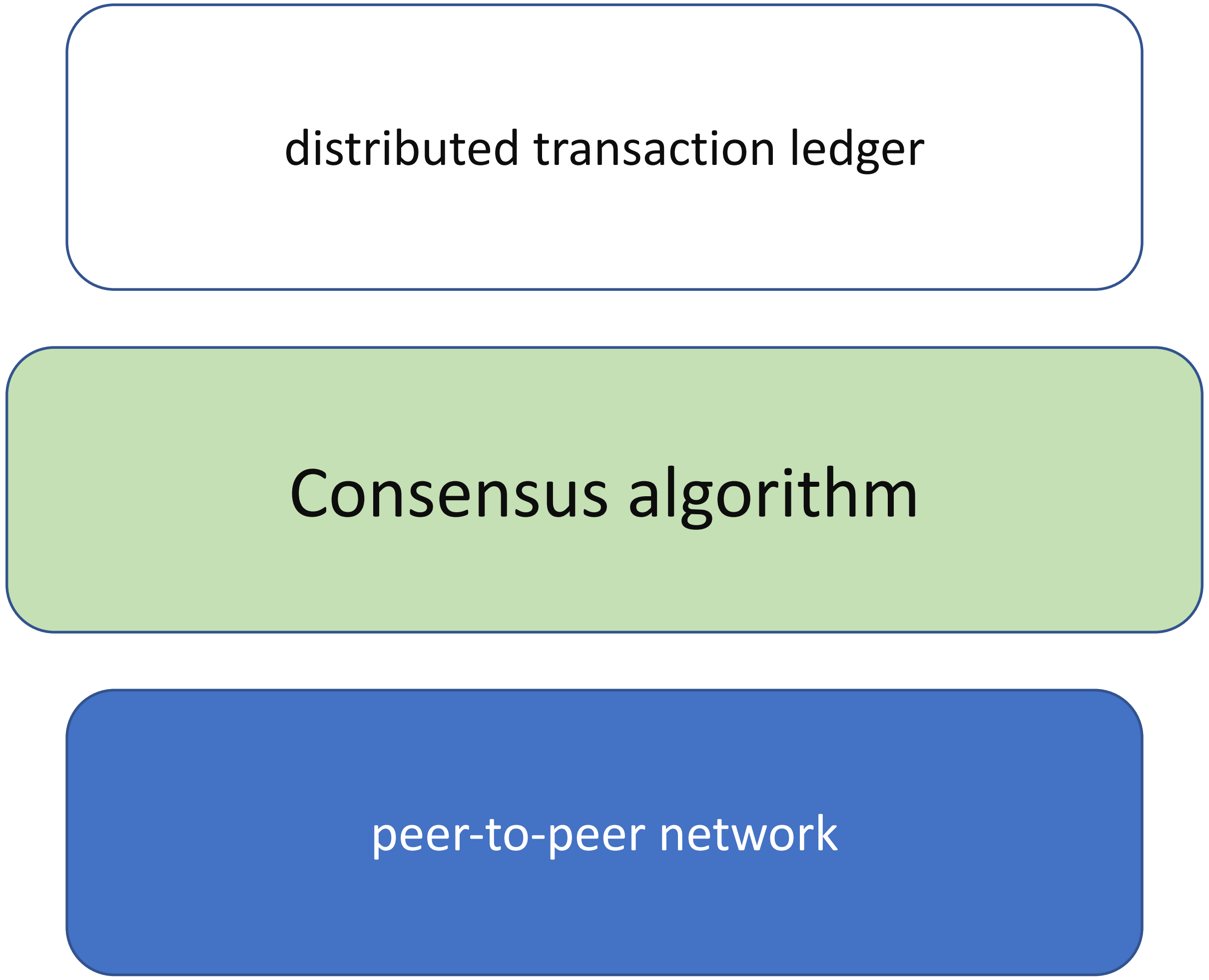
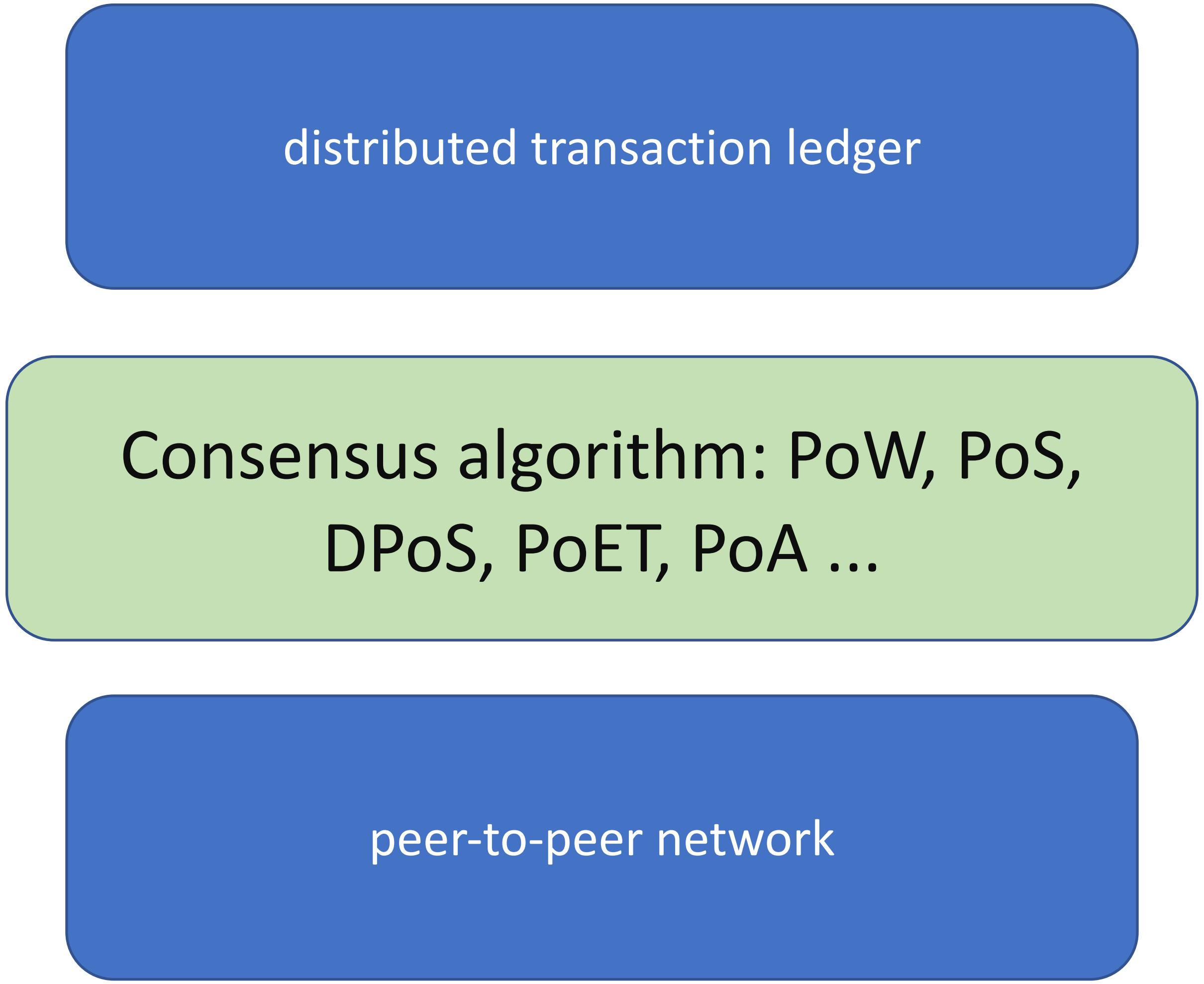
Blockchain

Consensus Protocols

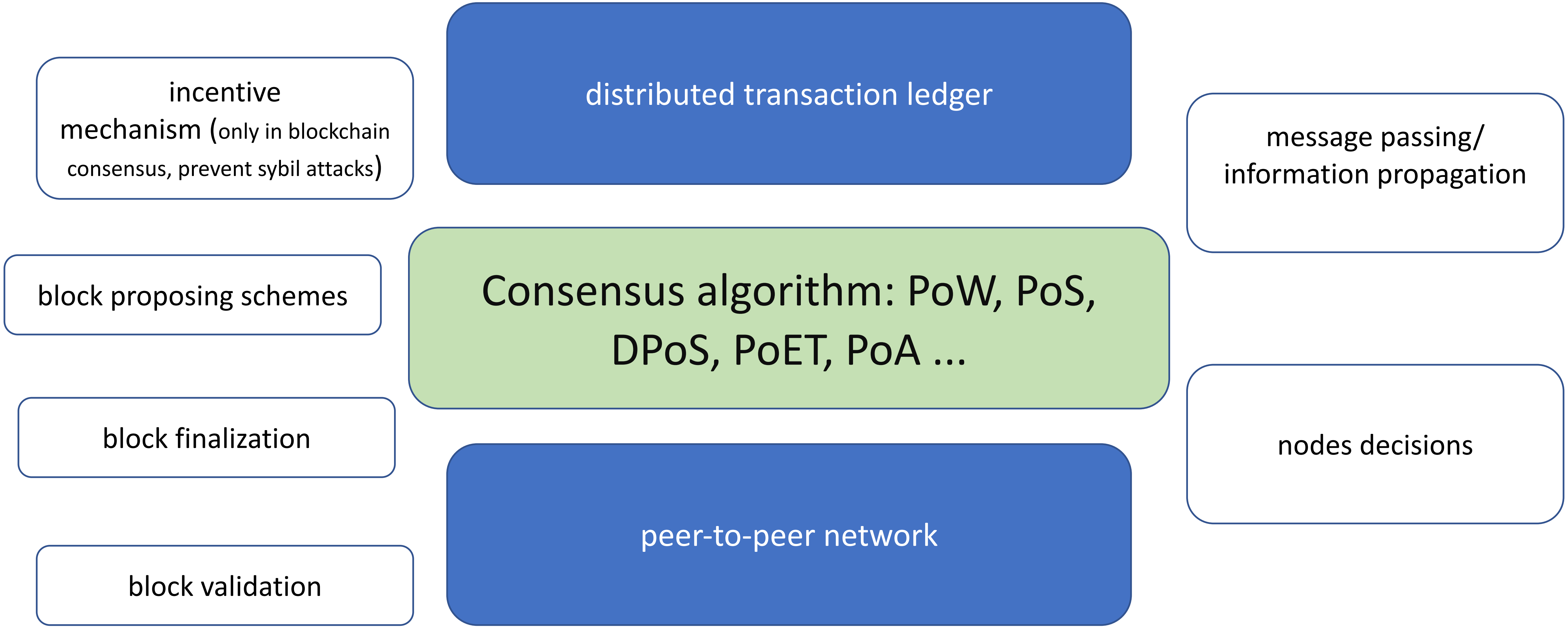

agree on some value,
leader election,
agree on transactions order …
Consensus algorithm
peer-to-peer network


distributed transaction ledger
Consensus algorithm
peer-to-peer network

Consensus protocols: Technology that enables decentralization. Ensures all participants agree on a unified transaction ledger without a central authority.

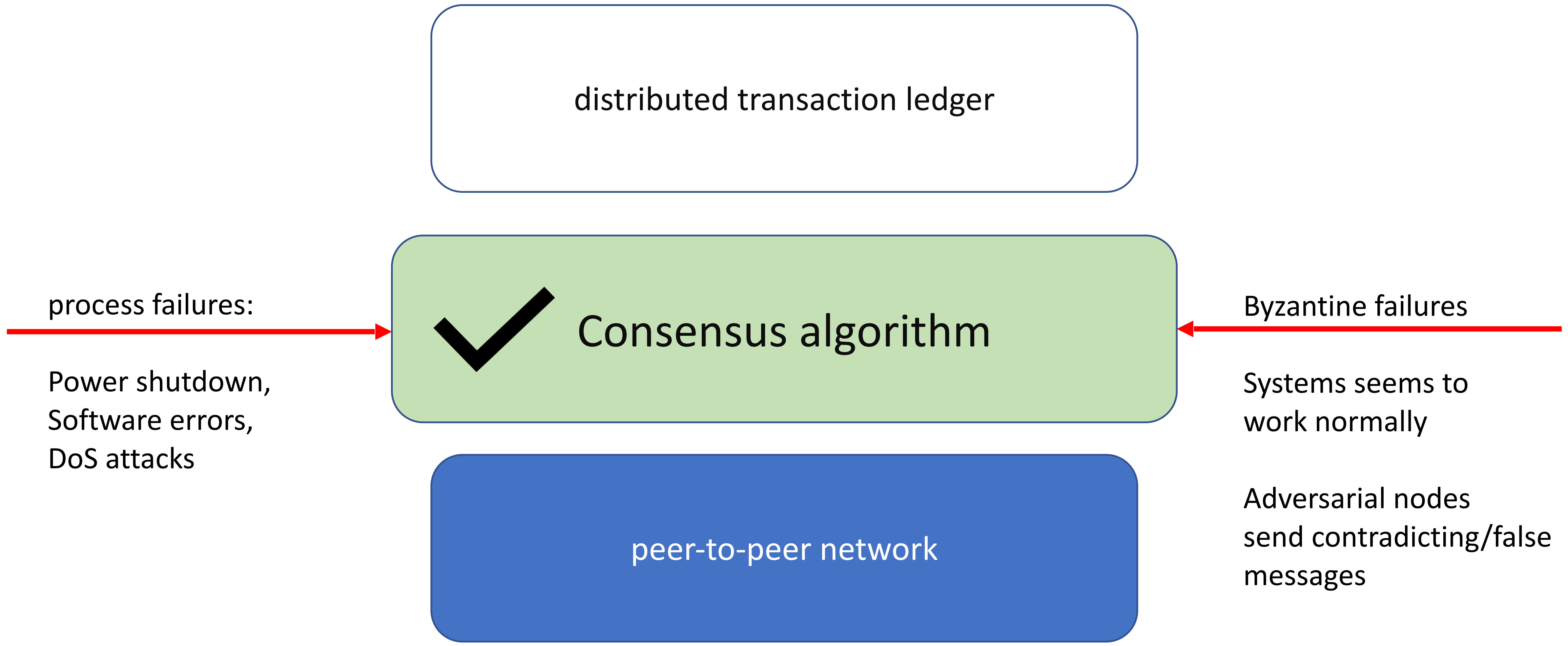

distributed transaction ledger
Consensus algorithm: PoW, PoS, DPoS, PoET, PoA ...
peer-to-peer network

Consensus protocols: Technology that enables decentralization. Ensures all participants agree on a unified transaction ledger without a central authority.


incentive mechanism (only in blockchain consensus, prevent sybil attacks)
distributed transaction ledger
message passing/ information propagation
block proposing schemes
Consensus algorithm: PoW, PoS, DPoS, PoET, PoA ...
block finalization
nodes decisions
block validation
peer-to-peer network

Bitcoin protocol

- Block generation: PoW find nonce, hash satisfies a difficulty target
- Block propagation: gossiping any block (received or locally generated) should be *advertised* to peers and *broadcast*
- Block validation: check block header and transactions
- Longest-chain rule: Blocks should always extend the longest chain.
- Rewards: coinbaise transactions.



Fault tolerant: ability of a distributed network to reach consensus despite system failures/malicious nodes sending incorrect messages.

Consensus protocols

- On blockchain: ensures all participants agree on a unified transaction ledger without the help of a central authority.
  - All agents (nodes) in a distributed system agree on the same value.
  - Agents agree on a majority value.
- Process failure
  - power shutdown, software errors, attacks (DoS) etc.
- Byzantine failure
  - Node sending faulty messages, acts as a non-faulty node.

Consensus protocols requirements

PoW

Fault tolerant?

Performance?

Scalability?

Transaction capability?

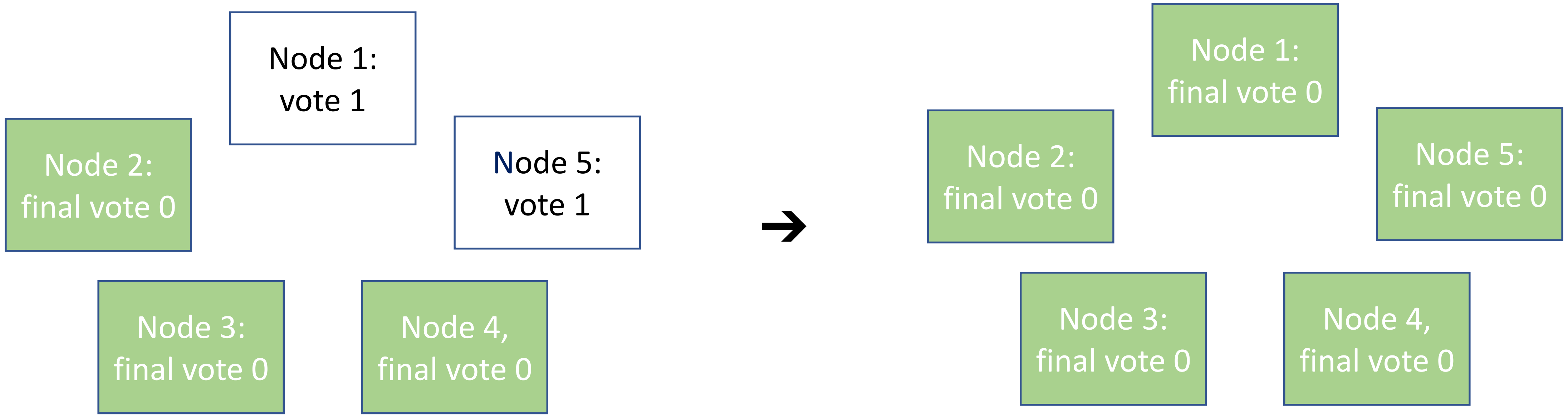
Security?

- energy consumption.
- small number of transactions/sec 7TPS.
- decreasing incentives.

Fault tolerant protocols requirements

- Agreement (Safety): "a bad thing never happens "

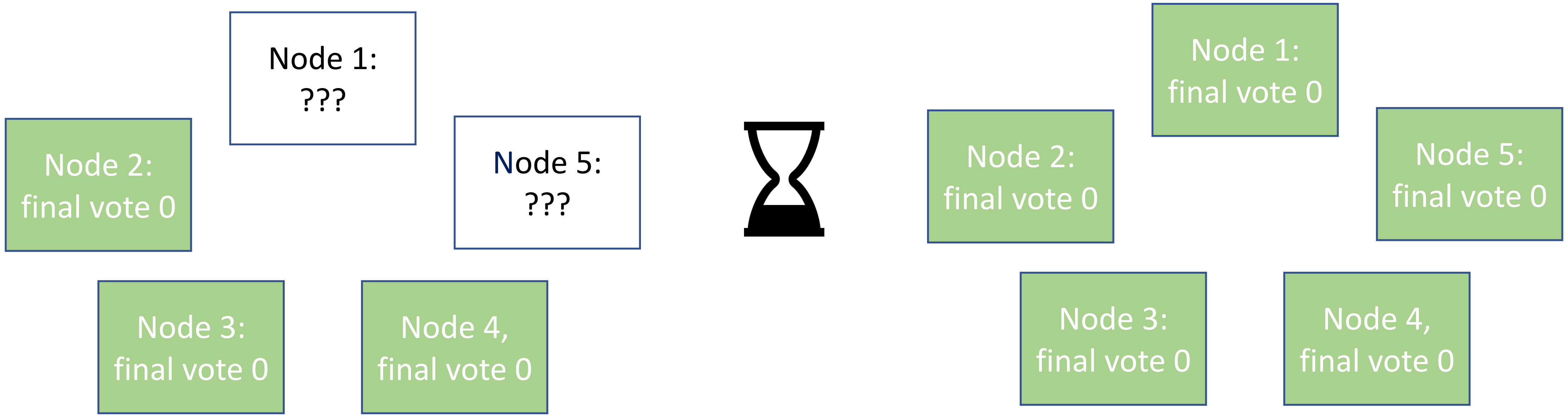
  Correct nodes agree on the same value.



Fault tolerant protocols requirements

- Termination (Liveness): a good thing will eventually happen ”

  All nodes will eventually make a decision, in a finite amount of time.



Safety = all nodes agree on transactions

Blockchain,
agree on a majority value
= longest chain

Liveness = a new block will be added

Fault tolerant protocols requirements

- Validity:

  If every correct (non faulty) process begins with the same initial value *v* , then their final decision must be *v.*

- Integrity:

  The consensus value *v* must have been proposed by some nonfaulty process.

Blockchain consensus

- Termination: Honest nodes either discard or accept a transaction, within the content of a block.
- Agreement: An accepted transaction is accepted by all honest nodes. A block has the same sequence number at every honest node (same order).
- Validity: If a node receives a valid transaction/ block, it should accept it into the blockchain.
- Integrity: check hash, signatures, chronological order, no double spending.

Consensus protocols

- Synchronous systems processes operate in rounds of time.
    - Round $\rightarrow$ process receive messages sent to it in the previous round
      $\rightarrow$ process message and send messages
    - Centralized clock synchronization
    - Fixed message delivery delay.

- Asynchronous systems no time order on communications.
    - no centralized clock

- Partial synchronous systems
    - communications arrive within a certain (*not known*) time frame.

Synchronous systems

time T fixed & known

Asynchronous systems

time not fixed
(no upper bound exists)

Partial Synchronous systems

time T fixed &
not known

Consensus protocols

- FLP (Fischer, Lynch, Paterson) theorem [1]

  In asynchronous systems consensus cannot be guaranteed,

  even with a single crash failure.

  safety and liveness cannot be simultaneously guaranteed.

Atomic broadcast and consensus are equivalent problems. [2]

All servers start with the same state, receive the same sequence of requests and output the same execution results, ending in same state.

Practical Byzantine Fault Tolerance (PBFT)

Consensus protocol for partially synchronous networks

Practical Byzantine Fault Tolerance (PBFT)

- Subprotocols
  - Normal-operation
  - Checkpoint
  - View-change

- Works on the assumption that the number of malicious nodes in the network cannot exceed 1/3 of the total number of nodes.

$$N \geq 3f + 1$$

- Used in Tendermint , Hyperledger , Zilliqa

- PBFT Miguel Castro and Barbara Liskov [4]

PBFT

Normal operation round (view) phases:

1. request: Client sends the request to the primary replica;
2. pre-prepare: Primary replica sends the requests to the backup replicas at the same time;
3. prepare: Replicas send responses to all servers (replicas and primary).
4. commit: After receiving ≥ 2f +1 prepare messages, a server sends commit messages to all servers;
5. response: After receiving ≥ 2f +1 commit messages, a server sends response to client;

PBFT

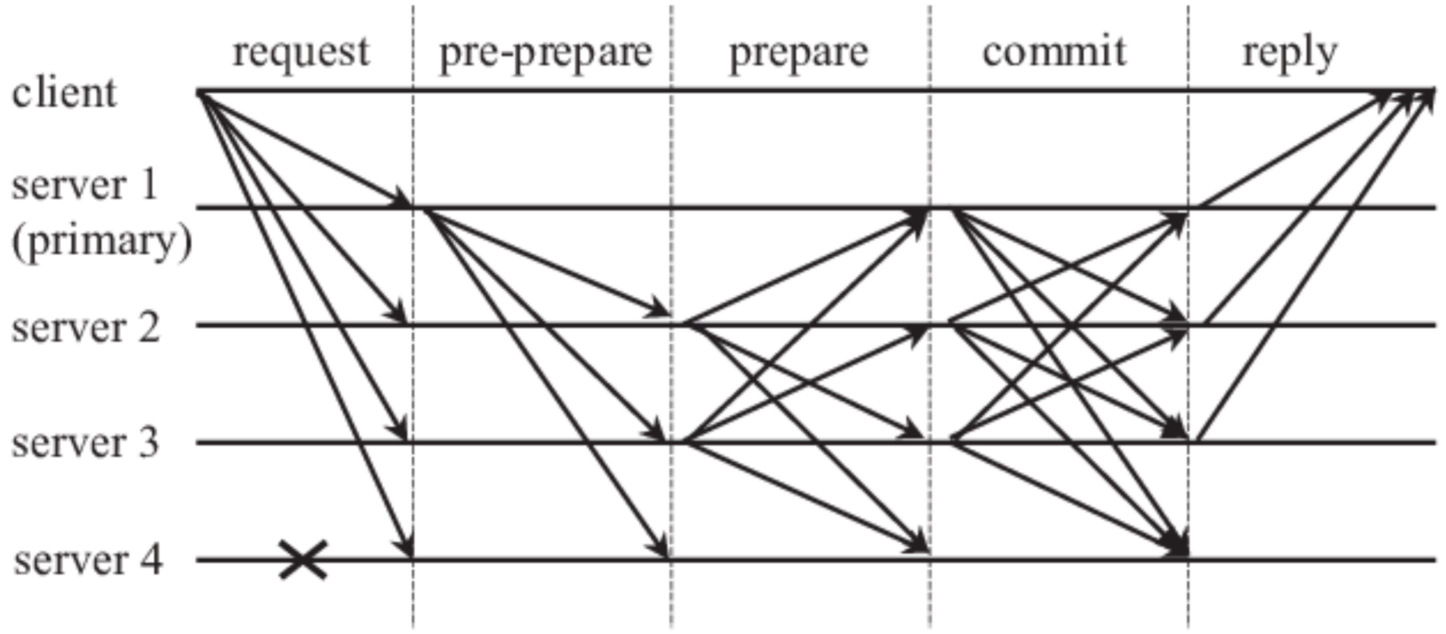
View change:

- Leader (primary replica) can be chanced if for a certain amount of time it hasn’t broadcast the request;
- A supermajority of honest nodes can decide that the leader is faulty and replace it.

Message complexity: normal operation $\mathcal{O}(n^2)$

Tolerance: If $N \geq 3f + 1$, PBFT can tolerate f byzantine failures. Client receive response is a majority of $2f + 1$ nodes agree on the result. Hence the malicious nodes are not able to alter the result.

PBFT normal operation



- image source [4]

Tendermint

- A node, the client, sends a proposed block to another node, the primary.
- Primary multicast the proposed block to backups.
    - The primary sends pre-prepare messages to backups. Pre-prepared messages are accepted, and node enters prepare phase.
    - Backup sends prepare message to other nodes. If prepare message is verified it enters commit phase
    - If the commit message is accepted the block is registered in the blockchain.

Tendermint

- Blockchain protocol used to replicate blockchain applications.
- If consists of:
  - A blockchain consensus engine (Tendermint Core)
  - A generic application interface (ABCI) process transactions in any programming language.
- Cosmos network: decentralized system of connected independent blockchains.
  - IoB internet of Blockchains
- ATOM cryptocurrency running on Cosmos network.

Tendermint

- Valiator
  - node that validates transactions on the network.
  - commits new blocks in the blockchain.
  - validators vote on proposals.
  - receive incentives.
  - weight of validator is determined by the amount of staking tokens.
  - Tokens can be self-delegated or delegated delagators.
  - Requirements: uptime, availability, minimum ATOM balance.

Tendermint

- Delegator
  - ATOM holders can delegate ATOM to a validator and obtain a part of their revenue in exchange.
  - If a validator misbehaves, each of their delegators are partially slashed in proportion to their delegated stake.
  - Validators can be in *jailed* state.

Bibliography

[1] Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. "Impossibility of distributed consensus with one faulty process." *Journal of the ACM (JACM)* 32.2 (1985)

[2] X. Defago, A. Schiper, and P. Urban, “Total order broadcast and ´ multicast algorithms: Taxonomy and survey,” ACM Computing Surveys, 2004.

[3] M. Castro, B. Liskov et al., “Practical byzantine fault tolerance,” in

OSDI, vol. 99, 1999, pp. 173–186

[4] Lei, Kai & Zhang, Qichao & Xu, Limei & Qi, Zhuyun. (2018). Reputation-Based Byzantine Fault-Tolerance for Consortium Blockchain. 10.1109/ PADSW.2018.8644933.