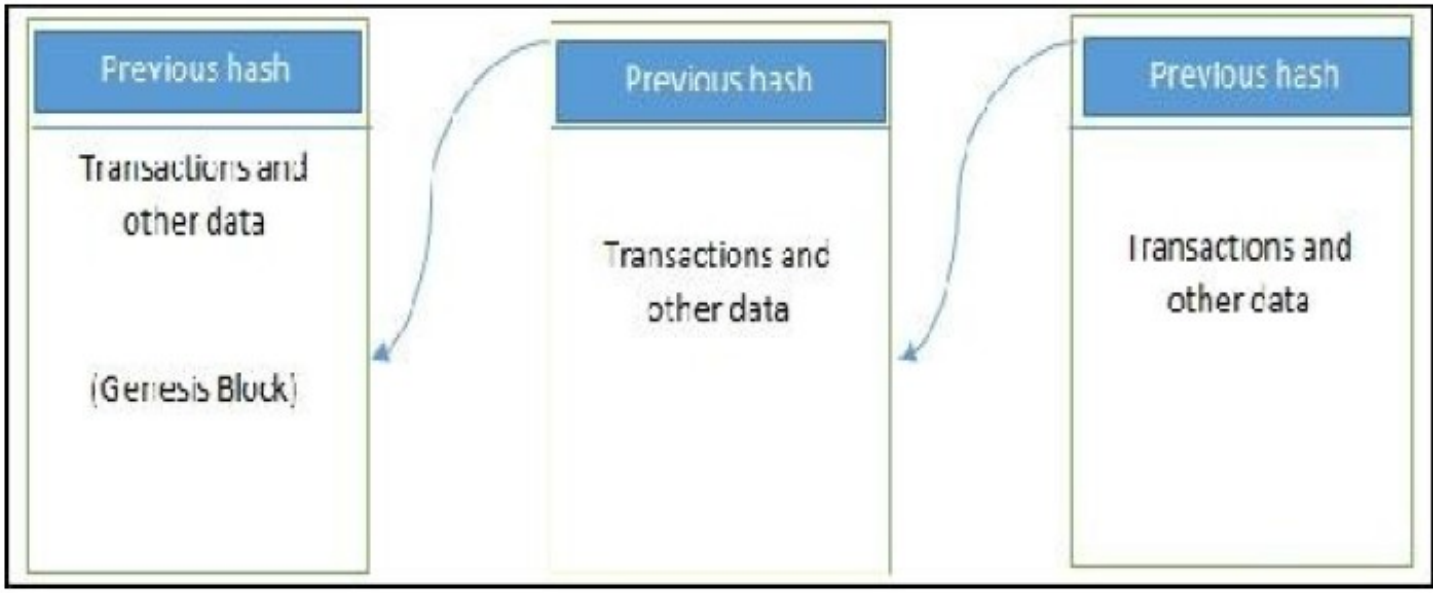
19ZO03 - INTRODUCTION TO BLOCKCHAIN TECHNOLOGY

Syllabus

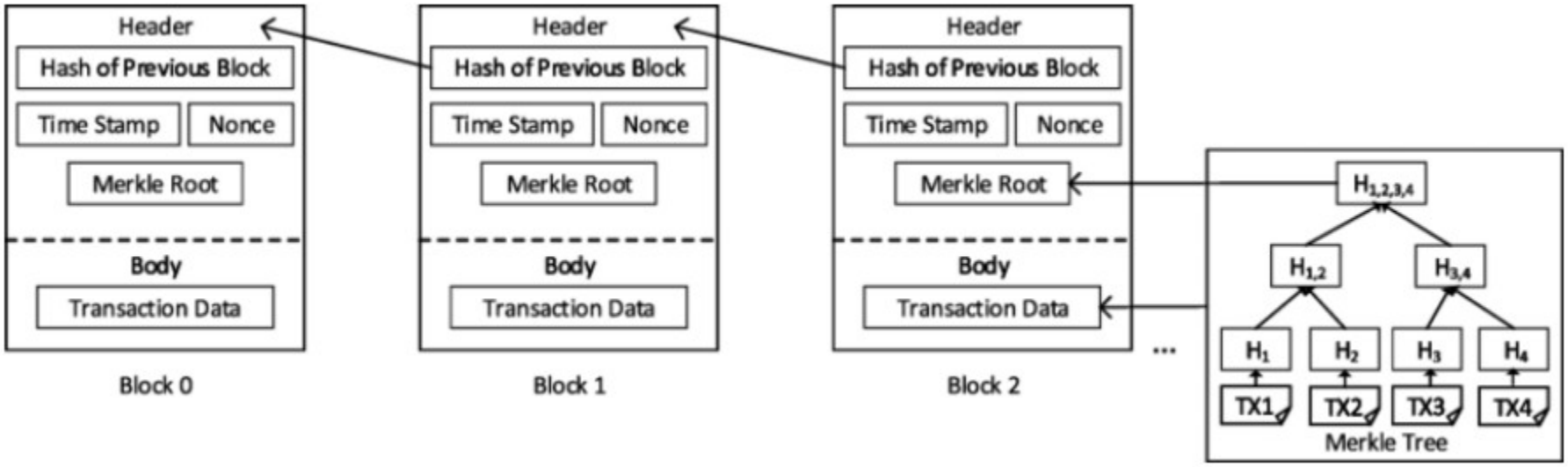
- **INTRODUCTION:** Distributed System, P2P system, Hadoop Distributed File System, Distributed Hash Table, ASIC resistance, issues, Distributed Ledger Technology- Private, public and permissioned ledgers - Cryptographic primitives- public key cryptography- Digital Signature Algorithm -Hashing- Blockchain evolution- Structure of Blockchain – Life of Blockchain application - consensus – Byzantine General problem and Fault Tolerance (11)
- **BLOCKCHAIN 1.0 - BITCOIN AND CRYPTOCURRENCY :** Block Hash - structure of block – syntax , structures, and validation - transaction life cycle- transaction types – Hash computation and Merkle Hash Tree -Bit coin and importance- Creation of coins–Bitcoin P2P Network-, Bitcoin protocols - Mining strategy and rewards – PoW and PoS – Difficulty, hash rate– Wallets- Double spending – forking- Token, Coinbase - practice on MTH (12)

Structure of Blockchain



Generic structure of a blockchain

Structure of Blockchain



Generic elements of a blockchain

- **ADDRESSES**
  - Unique identifiers (Sender , Receiver) that are used in a transaction on the blockchain.
  - A public key or derived from a public key Algorithm
  - As a good practice it is suggested that users generate a new address for each transaction in order to **avoid linking transactions to the common owner**, thus avoiding identification.

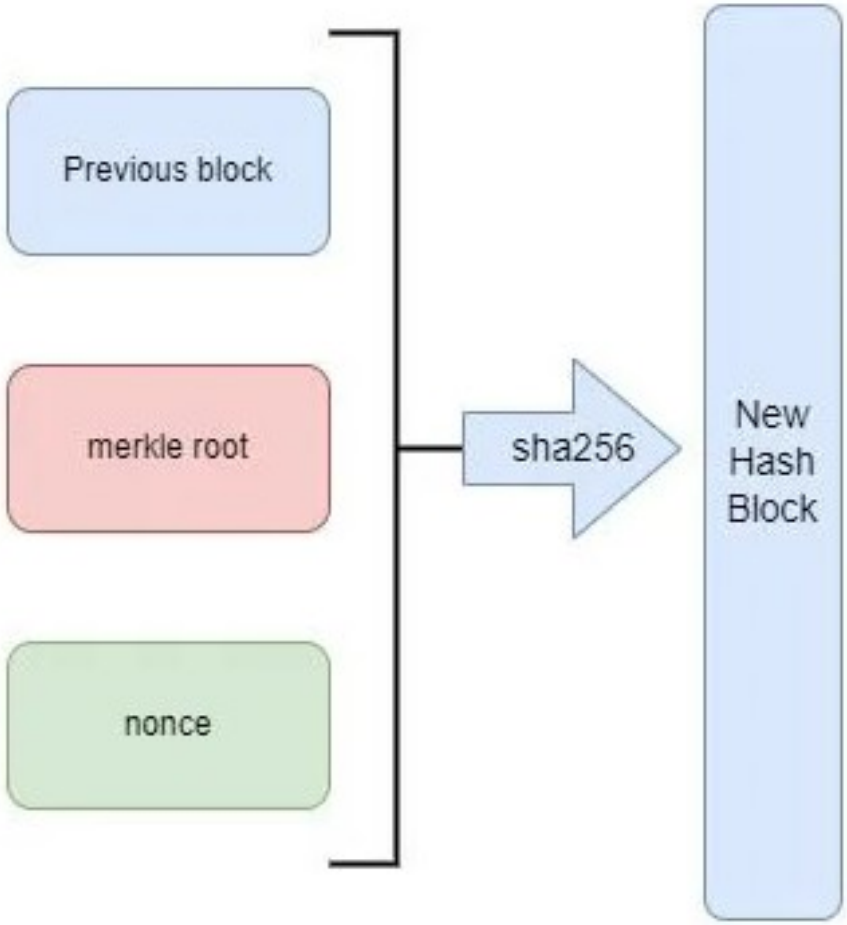
Generic elements of a blockchain

- **TRANSACTION**
  - Fundamental unit of a blockchain.
  - Represents a transfer of value from one address to another.
- **BLOCK**
  - A block is composed of multiple transactions and some other elements such as the previous block hash (hash pointer), timestamp, and nonce.

Nonce

- In blockchain, a nonce (short for "number used once") is a random or pseudo-random number that is used in the mining process to find a valid hash for a new block.
- It's a crucial component of the proof-of-work (PoW) consensus mechanism used in many cryptocurrencies like Bitcoin.
- The nonce, along with other block data, is fed into a cryptographic hash function, and the miner's goal is to find a nonce value that produces a hash output meeting specific criteria (e.g., starting with a certain number of zeros).

kettan007



Miners must find a nonce value that, when plugged into the hashing algorithm, generates a hash value that is lower than the target difficulty.

The nonce is the only field in the header that isn't predetermined.

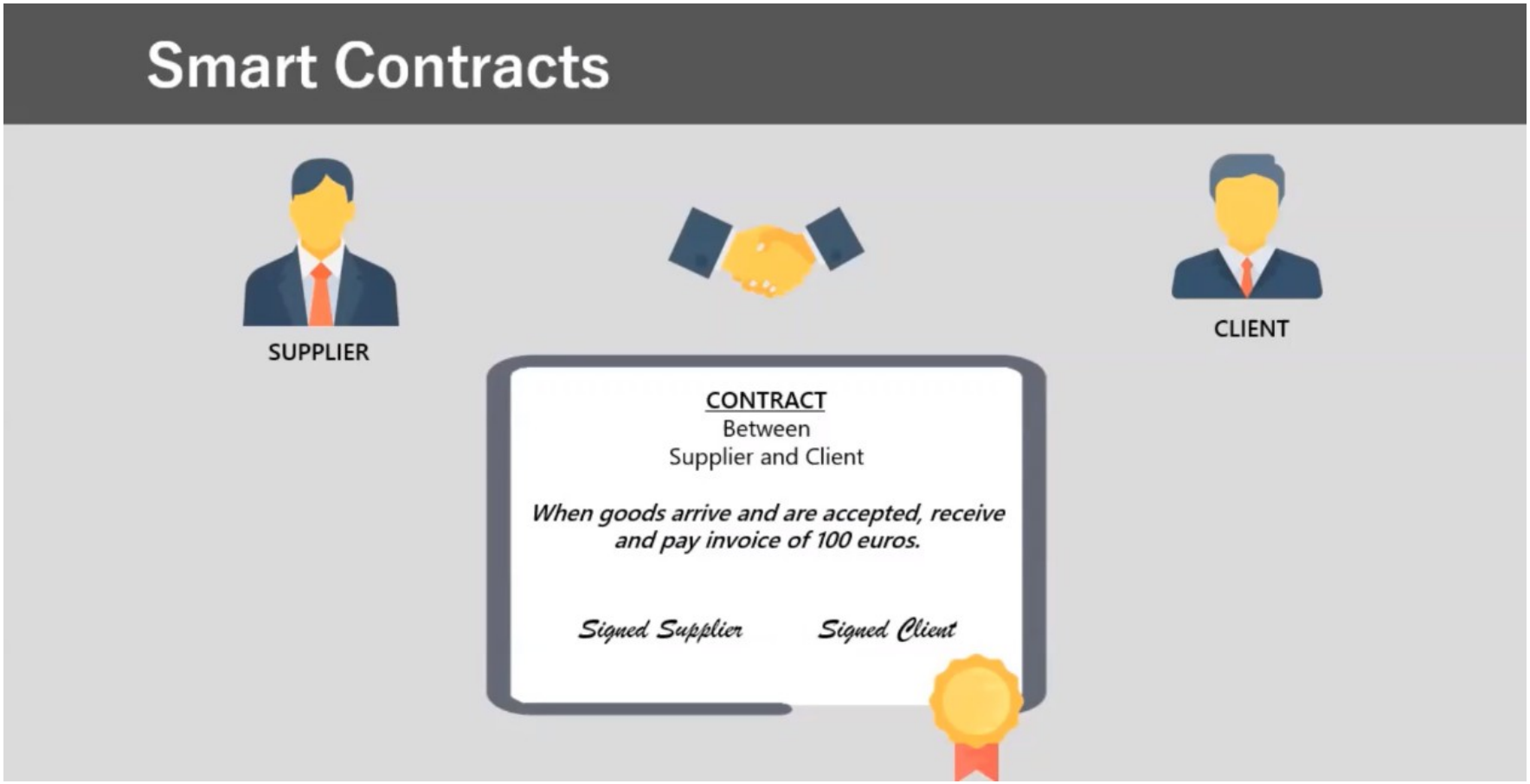
Generic elements of a blockchain

- **NODES**
  - Performs various functions depending on the type of the blockchain used and the role assigned to the node.
  - **Types of node:** Full Nodes ,Light Nodes
    - Full nodes: Store a complete copy of the Blockchain ledger
    - Light nodes: Only store the necessary data to verify transactions.
    - Miners: Find the valid block, store a copy and distribute it to other nodes) – Powerful Computers / hardware
      - ✓ **Validation of Transactions**
      - ✓ **Gathering transactions for a Block**
      - ✓ **Broadcasting a valid transaction and block**
      - ✓ **Consensus on next block creation**
      - ✓ **Chaining of blocks**

Generic elements of a blockchain

- **SMART CONTRACTS**
  - These programs run on top of the blockchain and encapsulate the business logic to be executed when certain conditions are met.
  - The smart contract feature is not available in all blockchains but is now becoming a very desirable feature due to the flexibility.

SMART CONTRACTS



Smart Contracts

SUPPLIER

CLIENT

```
contract Token {
    mapping(address => uint256) balances;
    using Balances for *;
    mapping(address => mapping (address => uint256)) allowed;

    event Transfer(address from, address to, uint amount);
    event Approval(address owner, address spender, uint amount);

    function balanceOf(address tokenOwner) public view returns (uint balance) {
        return balances[tokenOwner];
    }
    function transfer(address to, uint amount) public returns (bool success) {
        balances.move(msg.sender, to, amount);
        emit Transfer(msg.sender, to, amount);
        return true;

    }
```

Smart Contracts

SUPPLIER

CLIENT

```
contract Token {
    mapping(address => uint256) balances;
    using Balances for *;
    mapping(address => mapping (address => uint256)) allowed;

    event Transfer(address from, address to, uint amount);
    event Approval(address owner, address spender, uint amount);

    function balanceOf(address tokenOwner) public view returns (uint balance) {
        return balances[tokenOwner];
    }
    function transfer(address to, uint amount) public returns (bool success) {
        balances.move(msg.sender, to, amount);
        emit Transfer(msg.sender, to, amount);
        return true;

    }
```

Smart Contracts

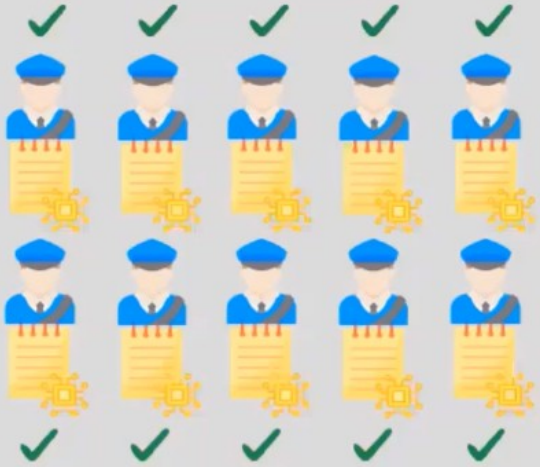
SUPPLIER

CLIENT

Smart Contracts

SUPPLIER

CLIENT



```
contract Token {
    mapping(address => uint256) balances;
    using Balances for *;
    mapping(address => mapping (address => uint256)) allowed;

    event Transfer(address from, address to, uint amount);
    event Approval(address owner, address spender, uint amount);

    function balanceOf(address tokenOwner) public view returns (uint balance) {
        return balances[tokenOwner];
    }
    function transfer(address to, uint amount) public returns (bool success) {
        balances.move(msg.sender, to, amount);
        emit Transfer(msg.sender, to, amount);
        return true;

    }
```

Generic elements of a blockchain

- **VIRTUAL MACHINE**
  - It is an extension of a transaction script
  - Allows Turing complete code to be run on a Blockchain
  - Various blockchains use virtual machines to run programs, for example **Ethereum Virtual Machine (EVM) and Chain Virtual Machine (CVM).**

Generic elements of a blockchain

- **STATE MACHINE**
  - Blockchain can be viewed as a state transition mechanism.
  - A state is modified from its initial form to the next and eventually to a final form as a result of a transaction execution and validation process by nodes.

Block structure

Block header	
Name	Description
Version	Block version number
Hash	The block's hash value
Parent hash	The previous block's hash value
Difficulty	The proof-of-work target difficulty
Timestamp	Creation time of the block
Merkle root	The root of Merkle Tree of transactions
Nonce	A random counter for proof-of-work

Block body: Transactions
Transaction 1, Transaction 2 … Transaction n

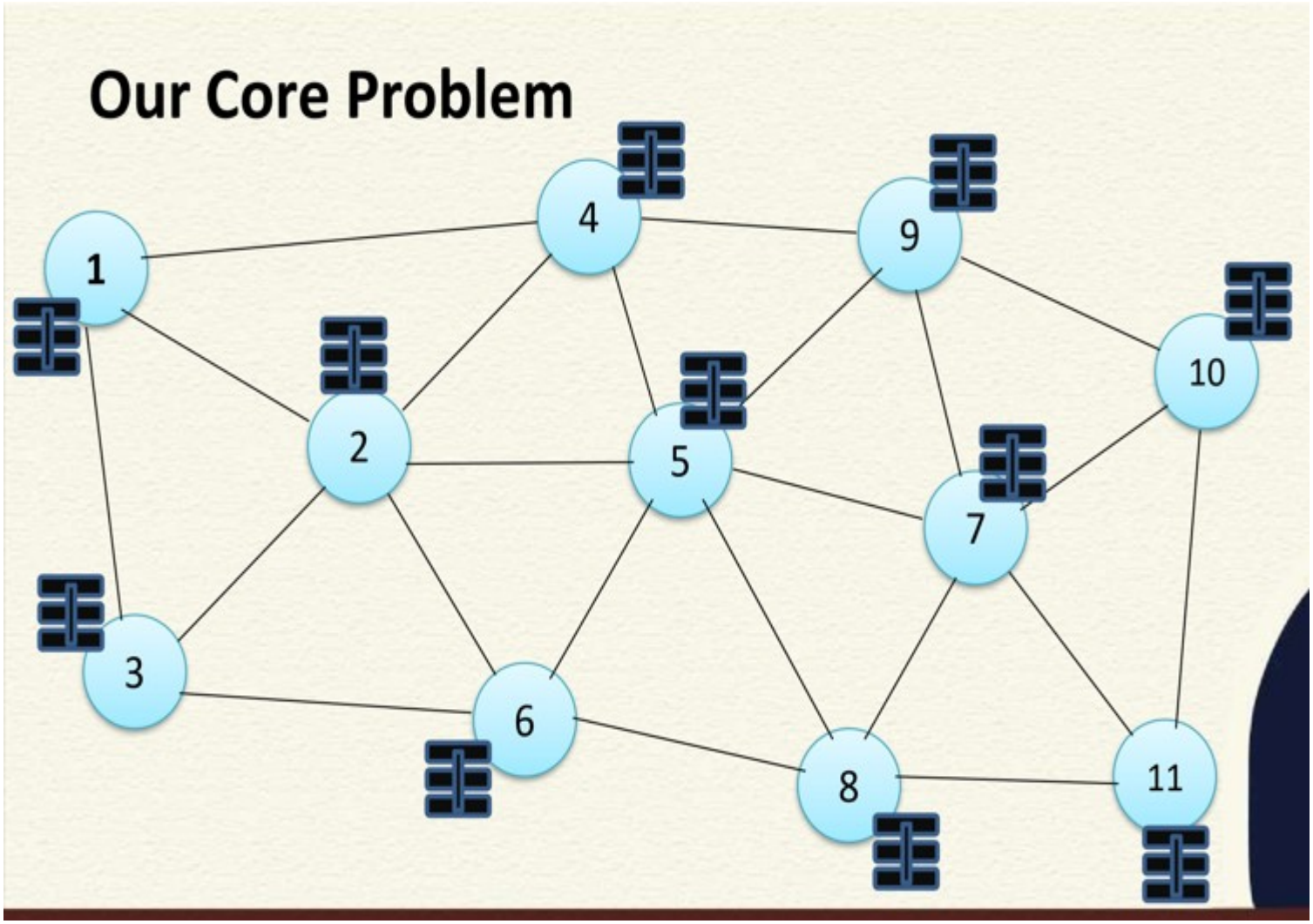
Transaction header	
Hash	The transaction's hash value
Block number	Block containing the transaction
Order	The transaction's number in the block
Timestamp	Creation time of the transaction
Sender	Sender's ID
Receiver	Receiver's ID
Signature	Sig{The transaction's hash value}

Payload
data 1, data 2..data n

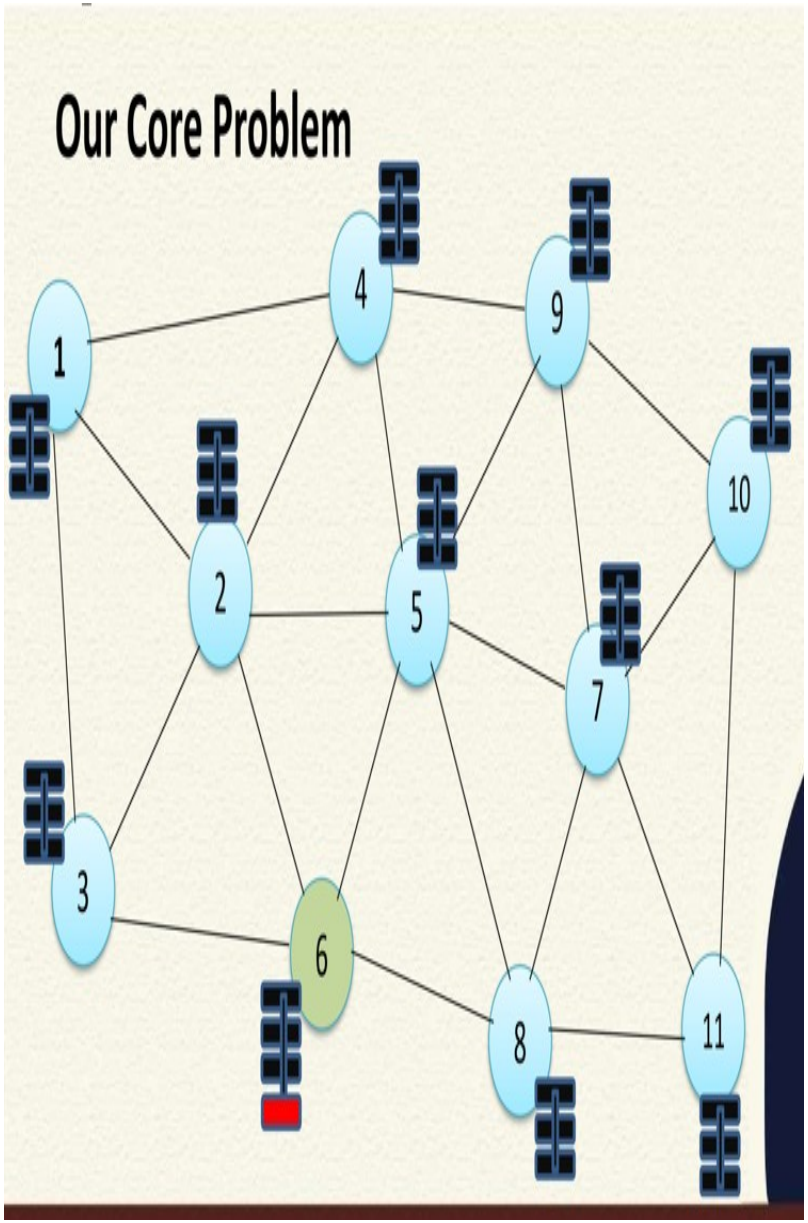
Consensus Mechanisms

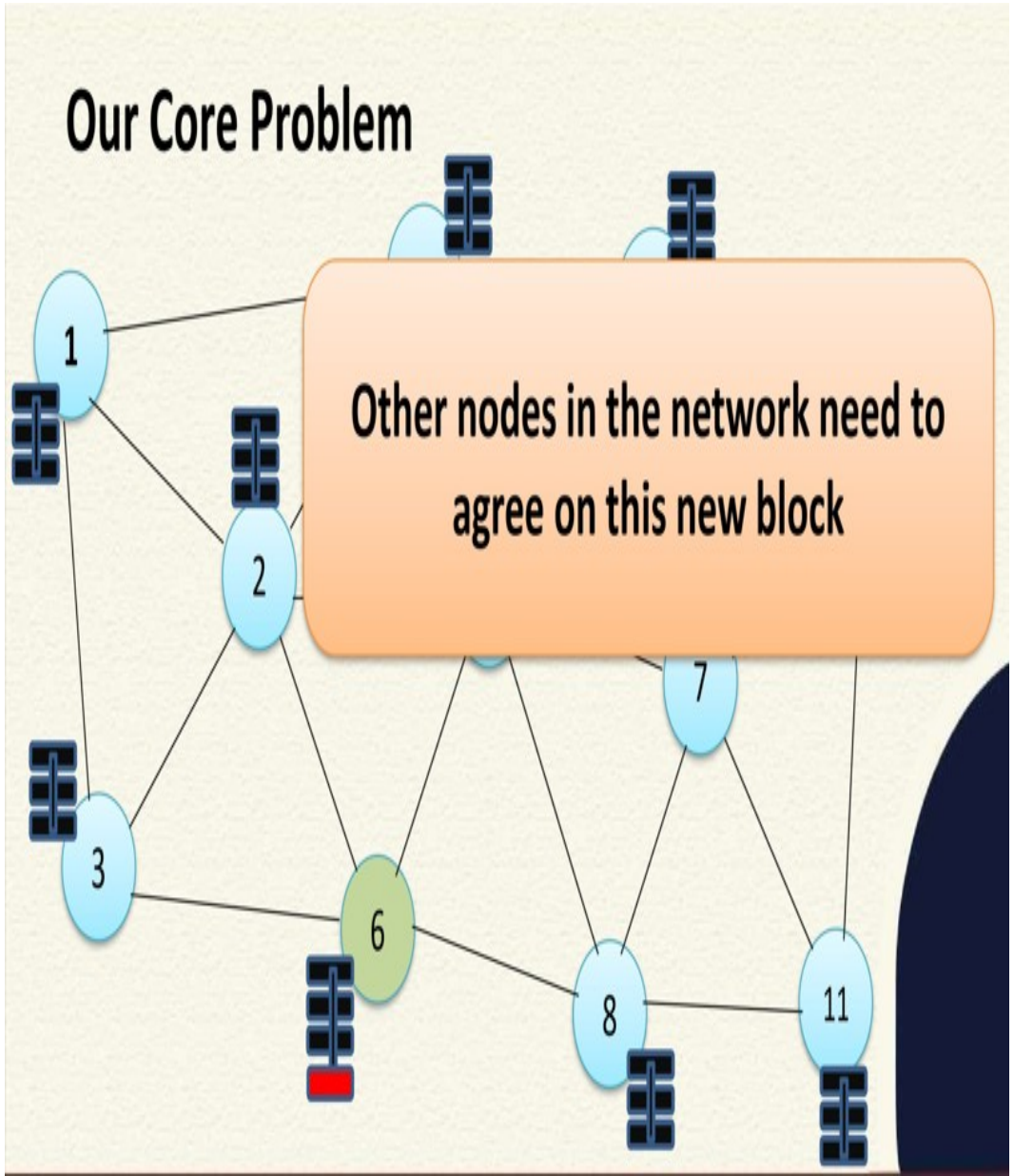
- **Consensus:** It is a process of agreement between distrusting nodes on a final **state of data.**
- This concept of achieving consensus (agree on a single value) between multiple nodes is known as **distributed consensus**.
- A consensus mechanism is a set of steps that are taken by all, or most, nodes in order to agree on a **proposed state or value**.

Classical Distributed Consensus Problem

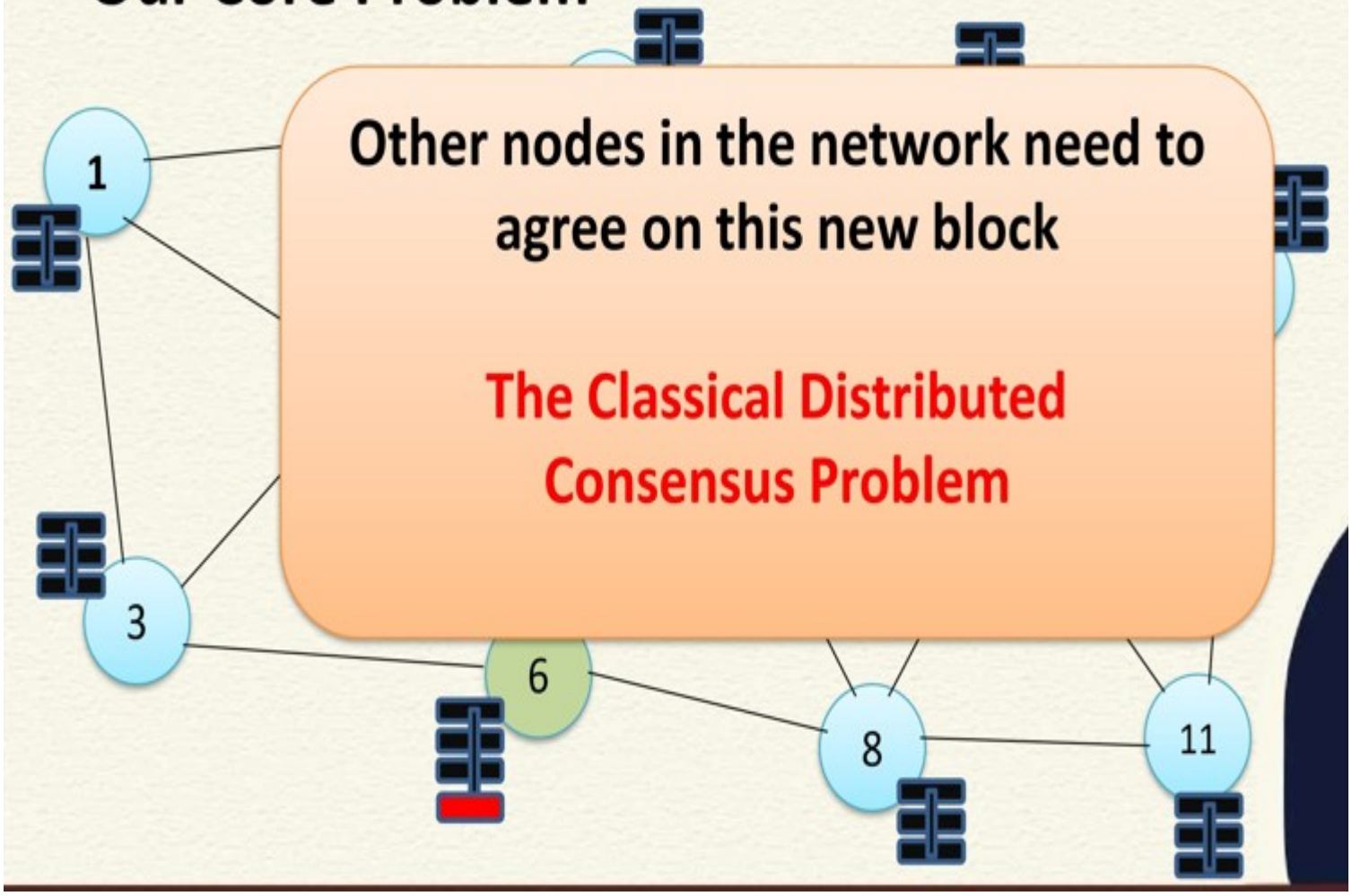


Classical Distributed Consensus Problem

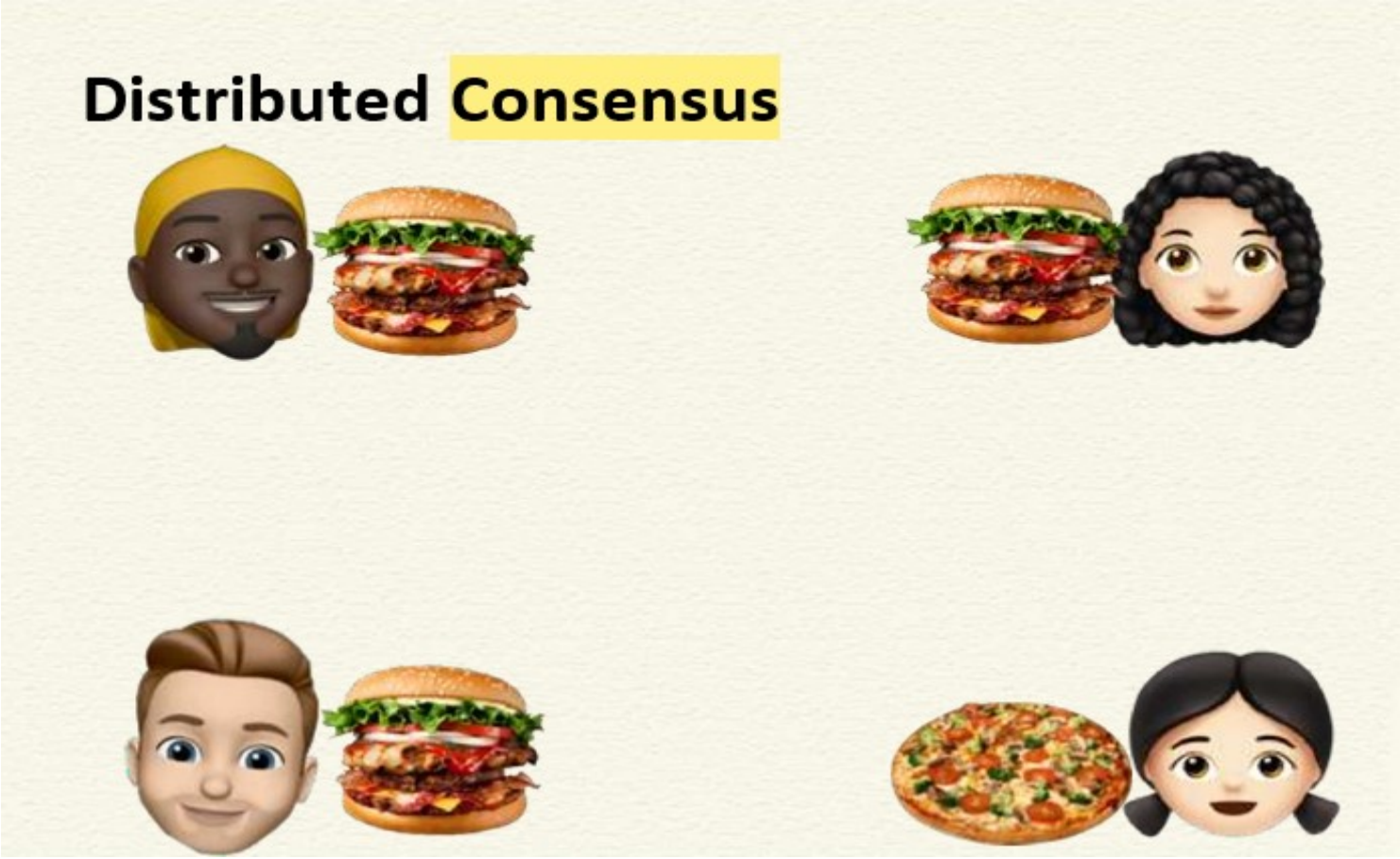




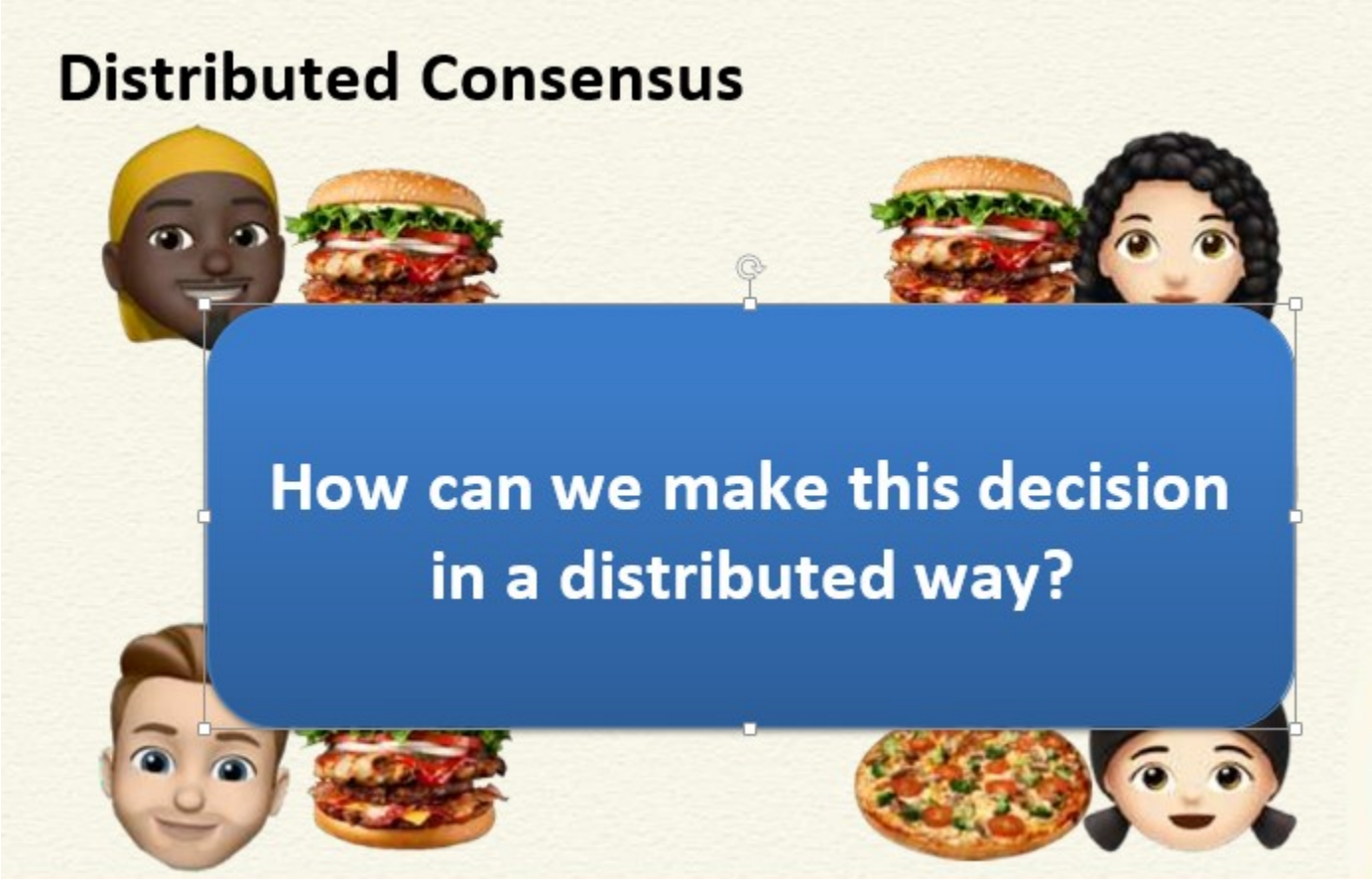
Our Core Problem



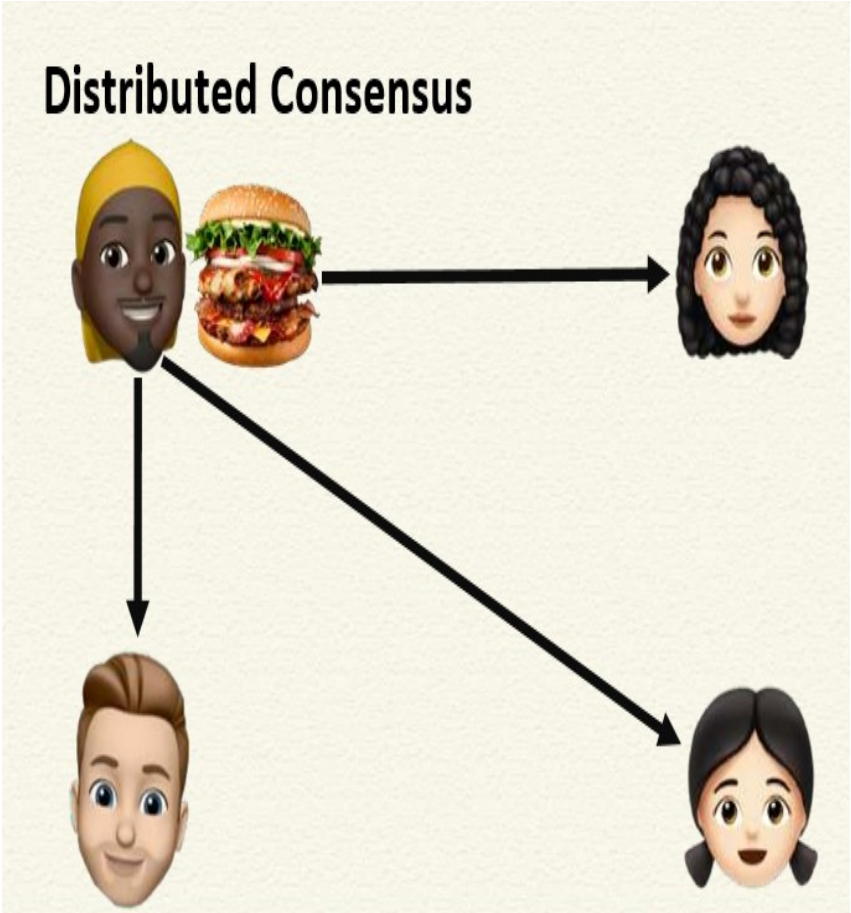
Distributed Consensus

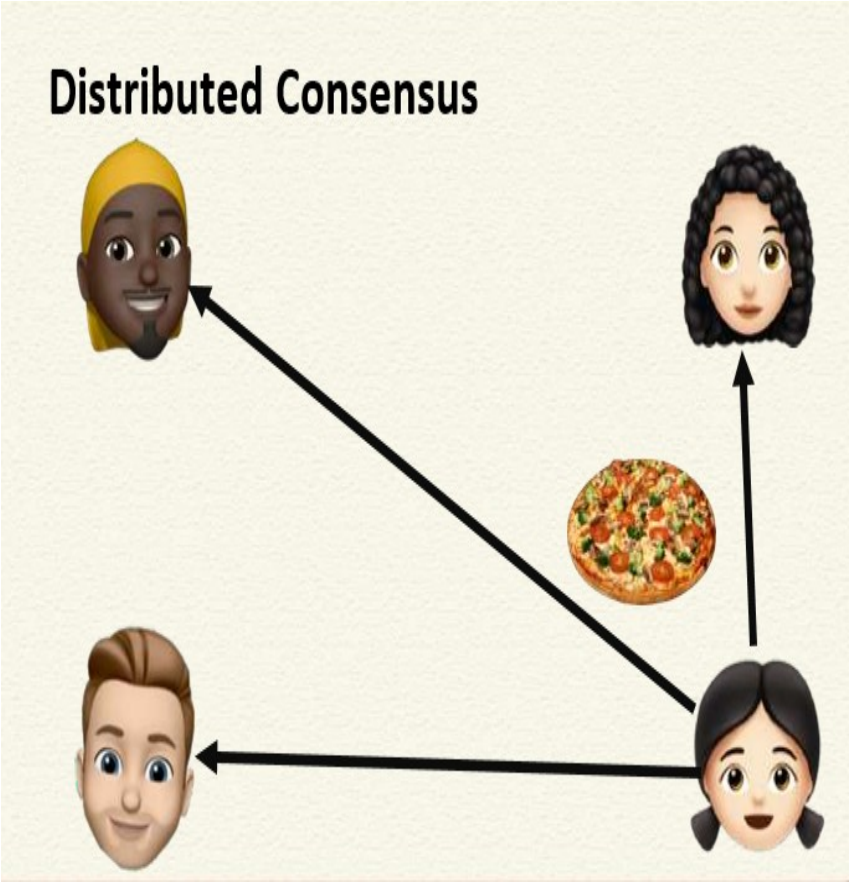


Distributed Consensus

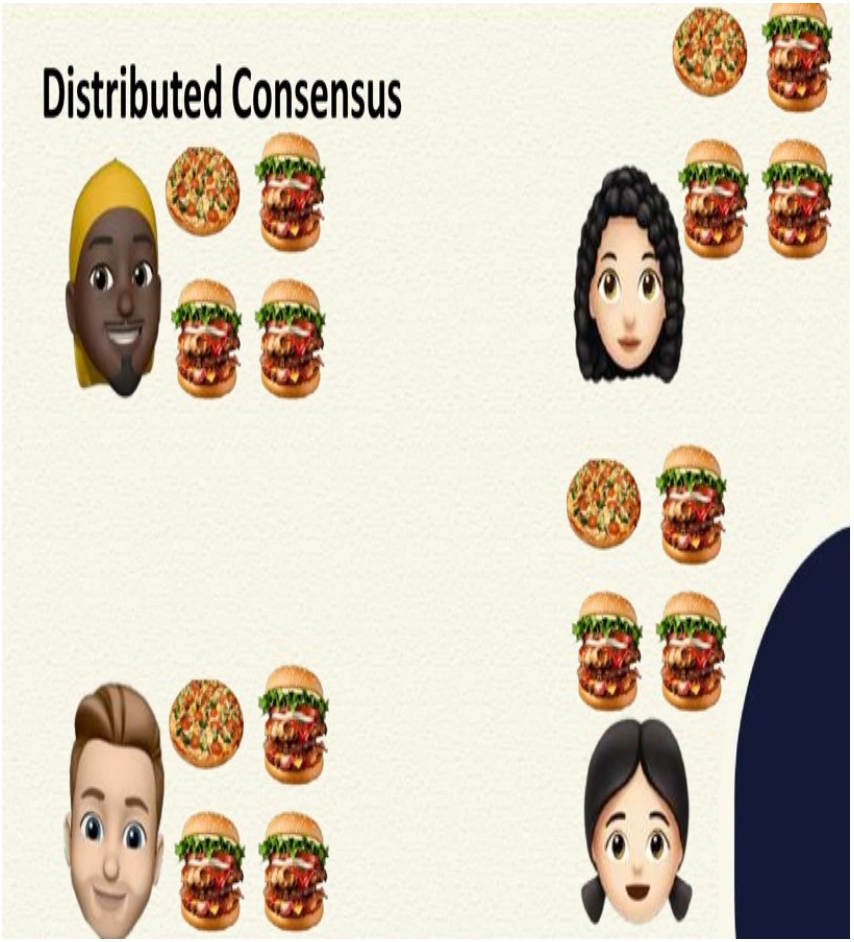


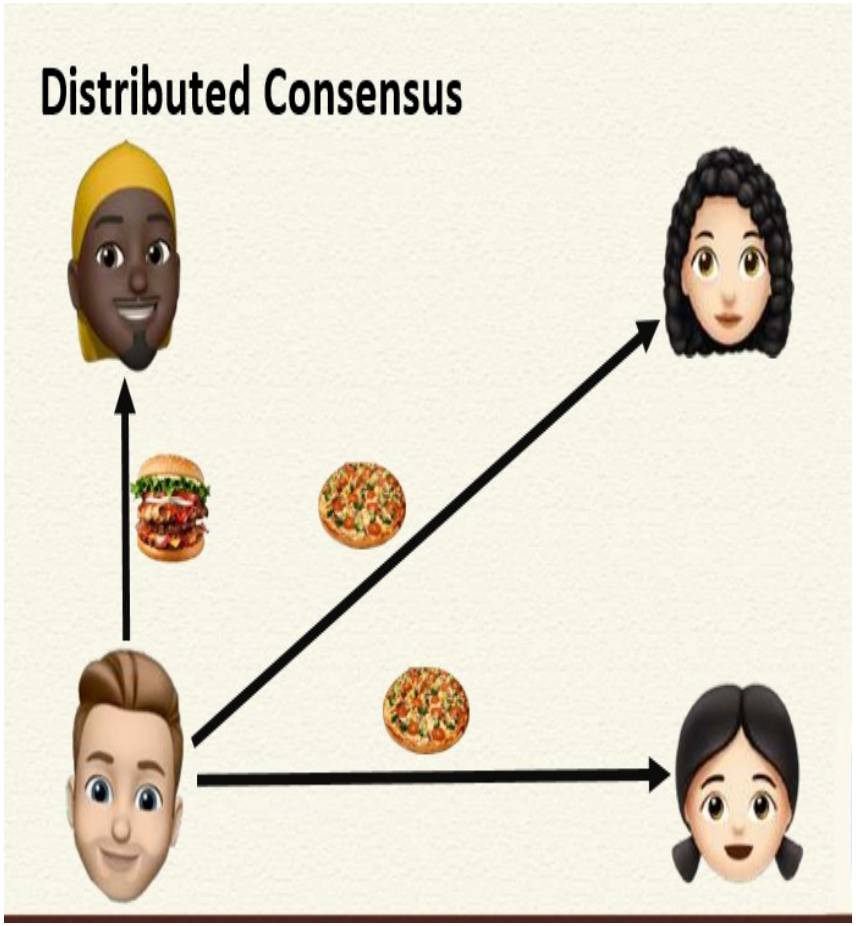
Distributed Consensus

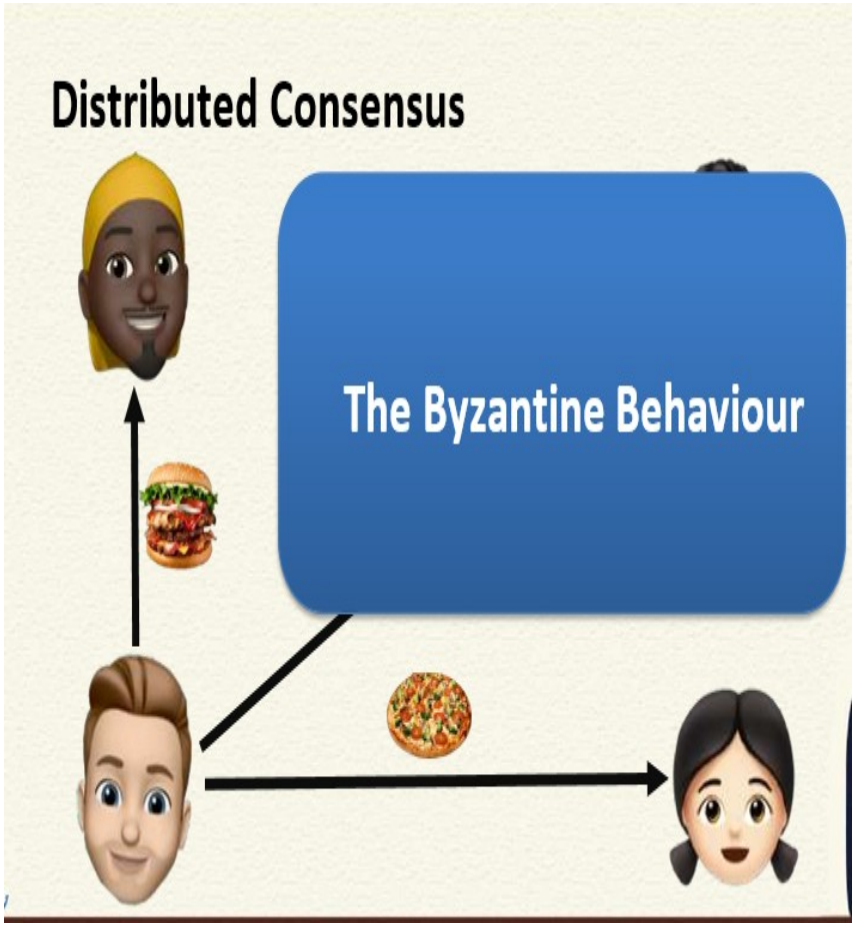




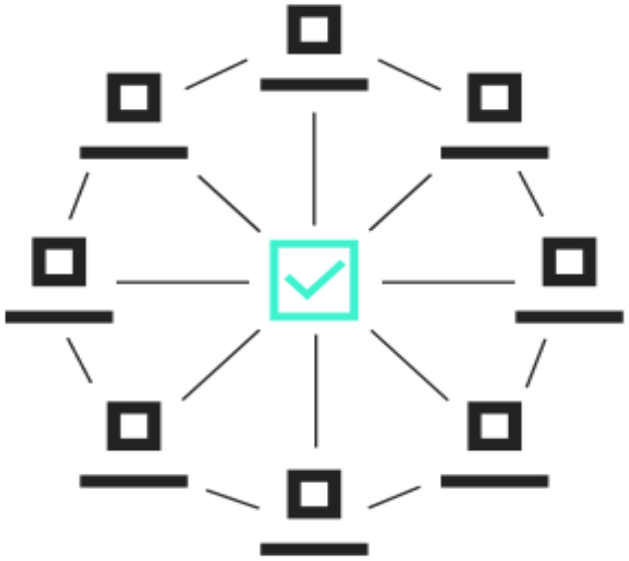
Distributed Consensus




Distributed Consensus


Distributed Consensus
The Byzantine Behaviour

Consensus Mechanisms

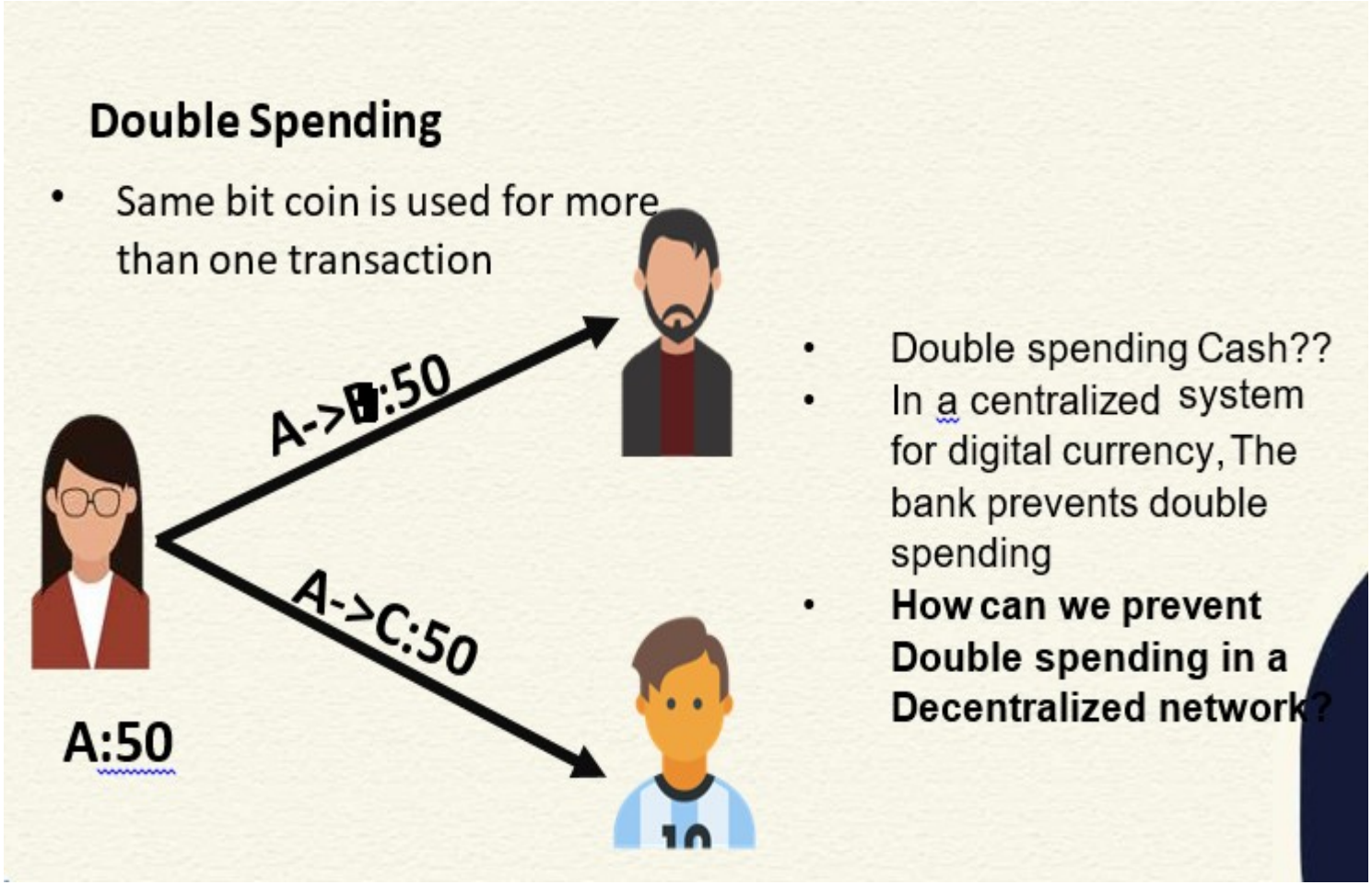


Decentralized Consensus

Centralized Consensus

Process prevents errors and secures the network against threats such as double-spending or Sybil attacks

Double Spending



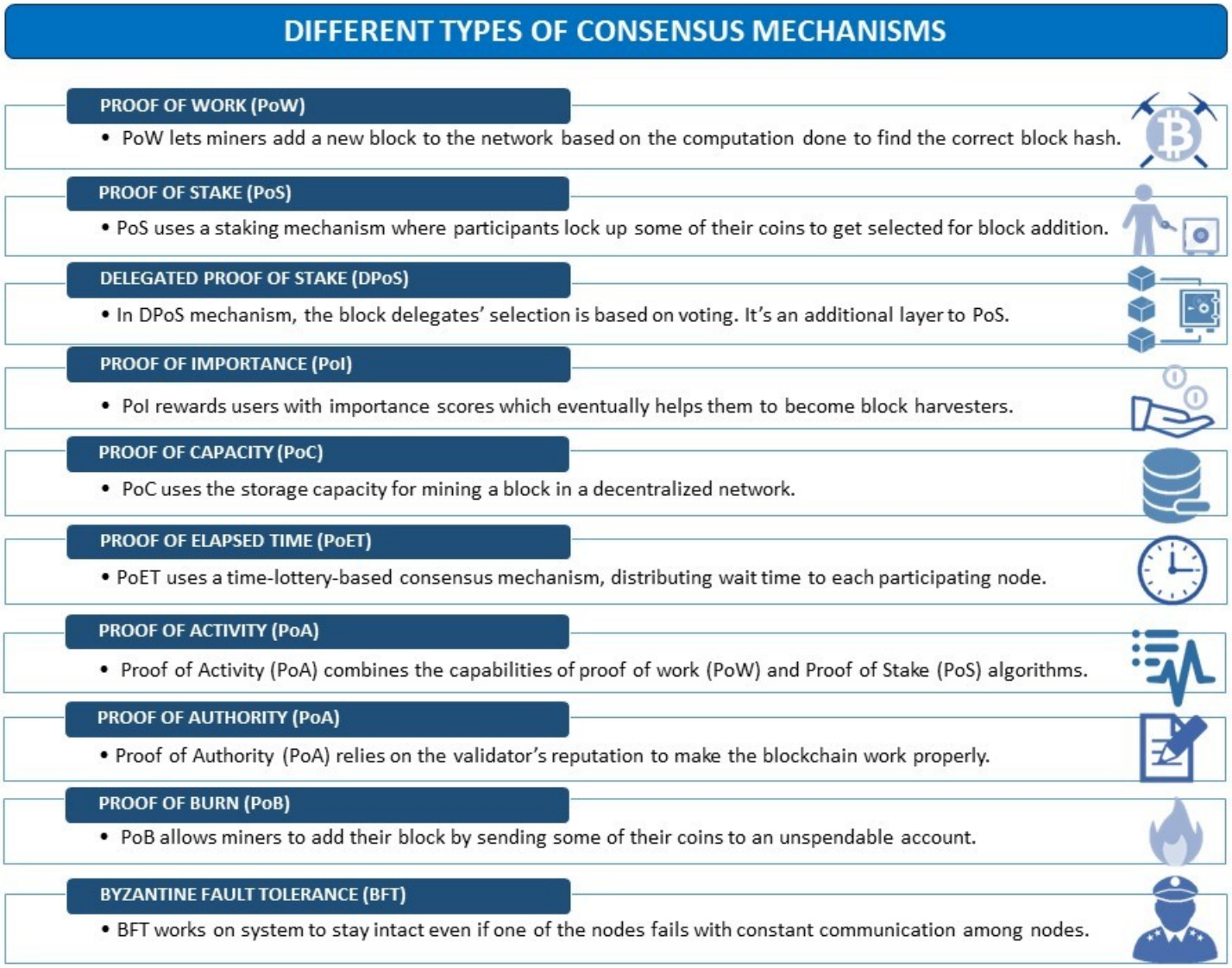
CONSENSUS MECHANISMS

- More than three decades this concept has been researched by computer scientists in the industry and academia
- Consensus mechanisms have recently come into the limelight and gained much popularity with the advent of bitcoin and blockchain
- There are various requirements which must be met in order to provide the desired results in a consensus mechanism.

Consensus Requirements

- **Agreement:** All honest nodes decide on the same value**.**
- **Termination:** All honest nodes terminate execution of the consensus process and eventually reach a decision.
- **Validity:** The value agreed upon by all honest nodes must be the same as the initial value proposed by at least one honest node.
- **Fault tolerant:** The consensus algorithm should be able to run in the presence of faulty or malicious nodes (Byzantine nodes).
- **Integrity:** This is a requirement where by no node makes the decision more than once. The nodes make decisions only once in a single consensus cycle.

Types of Consensus Mechanisms



TYPES OF CONSENSUS MECHANISM

- **Byzantine fault tolerance-based: With no compute intensive** operations such as partial hash inversion, this method relies on a simple scheme of nodes that are publishing signed messages.

  Eventually, when a certain number of messages are received, then an agreement is reached.

- **Leader-based consensus mechanisms:** This type of mechanism requires nodes to compete for the *leader-election lottery and the node* that wins it proposes a final value.

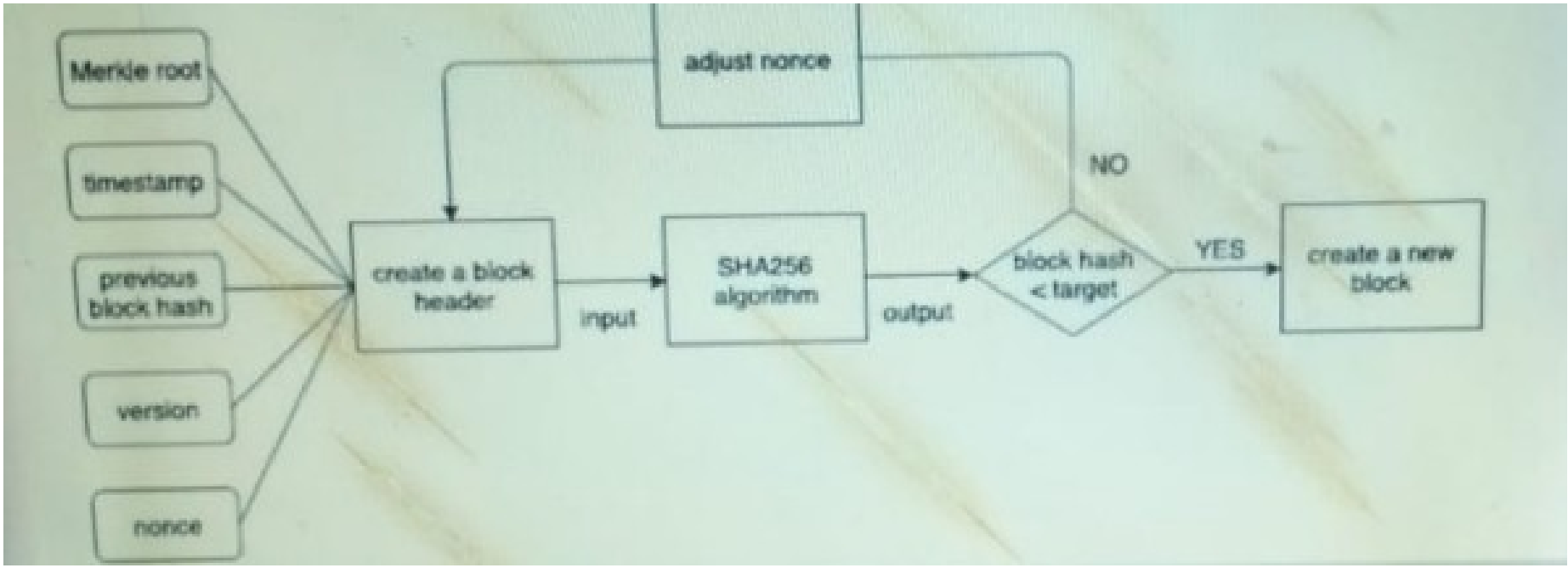
Types of Consensus Mechanisms

- Many practical implementations have been proposed such as **Paxos, the most famous protocol introduced by Leslie Lamport in 1989.**
- **In Paxos nodes are assigned** various roles such as Proposer, Acceptor, and Learner.
- Nodes or processes are named replicas and consensus is achieved in the presence of faulty nodes by agreement among a majority of nodes.
- Alternative to Paxos is RAFT, which works by assigning any of three states, that is, Follower, Candidate, or Leader, to the nodes.
- A Leader is elected after a candidate node receives enough votes and all changes now have to go through the Leader, who commits the proposed changes once replication on the majority of follower nodes is completed.

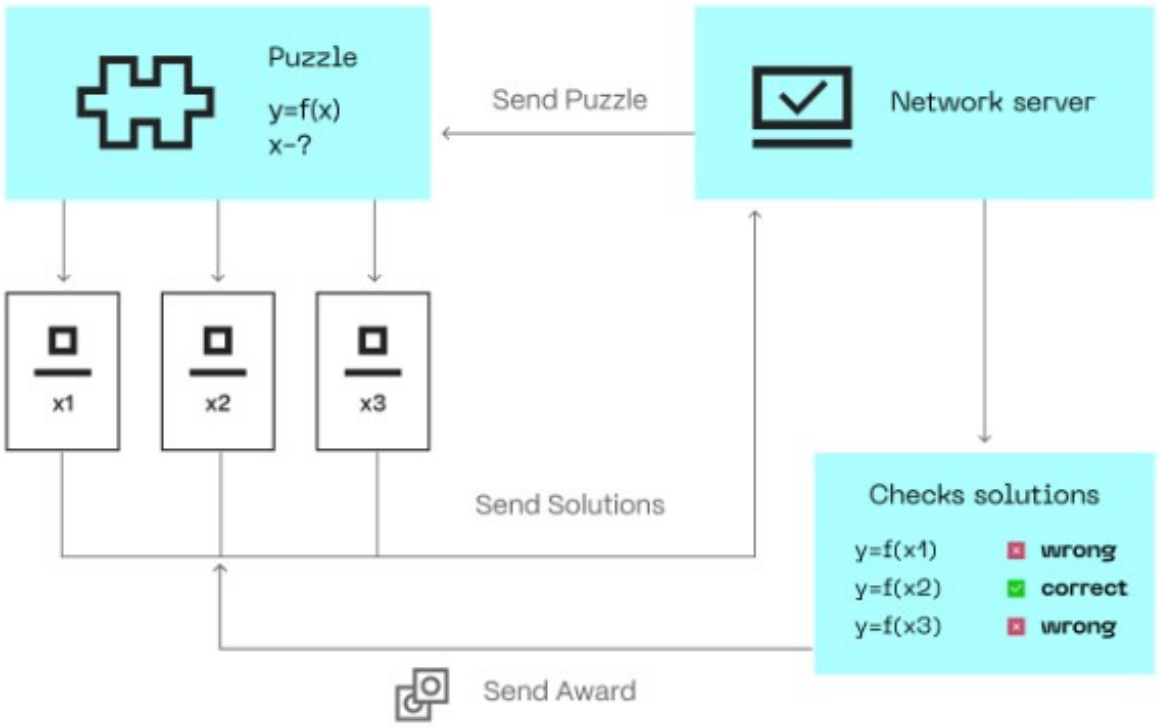
TYPES OF CONSENSUS MECHANISM

- **Proof Of Work:** is one of the earliest consensus algorithms, which works based on game theory.
- Many popular blockchains adopted it, including **Bitcoin, Litecoin, and Dogecoin.**
- There are high-level computational tasks that miners have to do in discovering new blocks, called mining.

Proof Of Work(PoW)



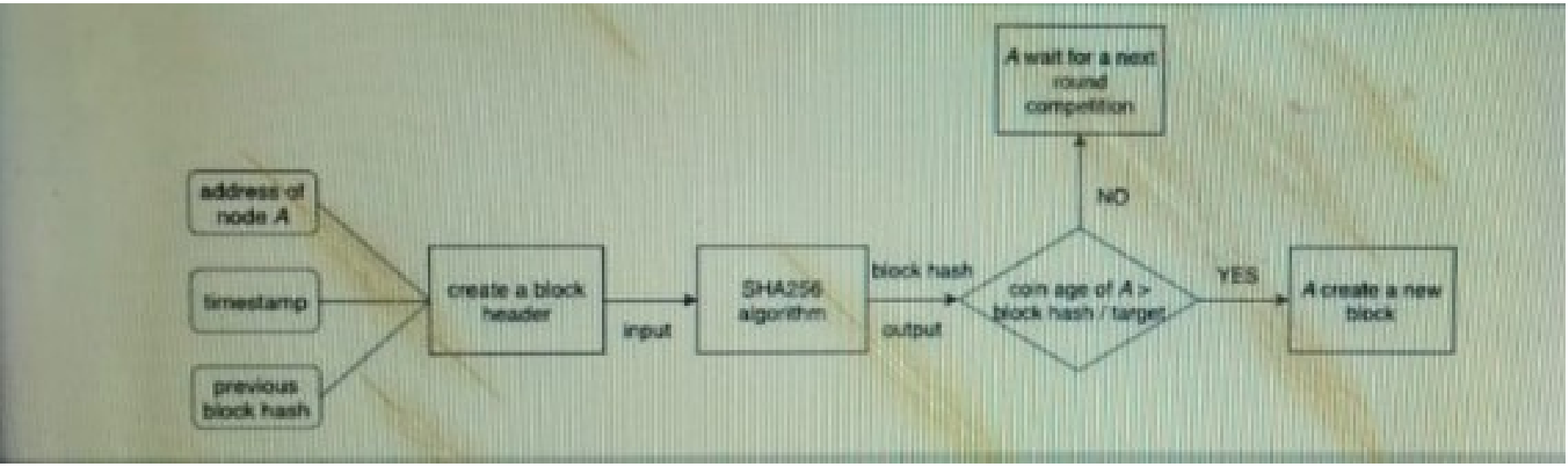
Proof Of Work(PoW)



Imagine there is a **puzzle tournament** and **whoever solves it first wins the game**. The grid was 9 x 9 at the start, but the organizers increased it to 27 x 27 as hundreds of people joined. Thus, making it harder to solve. Some of the participants developed custom machines to automate the solution. This will give them a higher chance of winning, right?

Proof of Stake (PoS)

- PoS: consensus **eliminates the high energy consumption** by PoW. PoS uses a **staking mechanism** in which **miners (or validators) hold some of their earned coins** in the network to get selected for adding a block.
- It was developed as a better option for PoW. In 2022, Ethereum dropped PoW for PoS because it is more energy-efficient and decentralized.
- Other prominent blockchains, such as Ethereum 2.0 Tezos and Cardano, also incorporated PoS into their protocols.

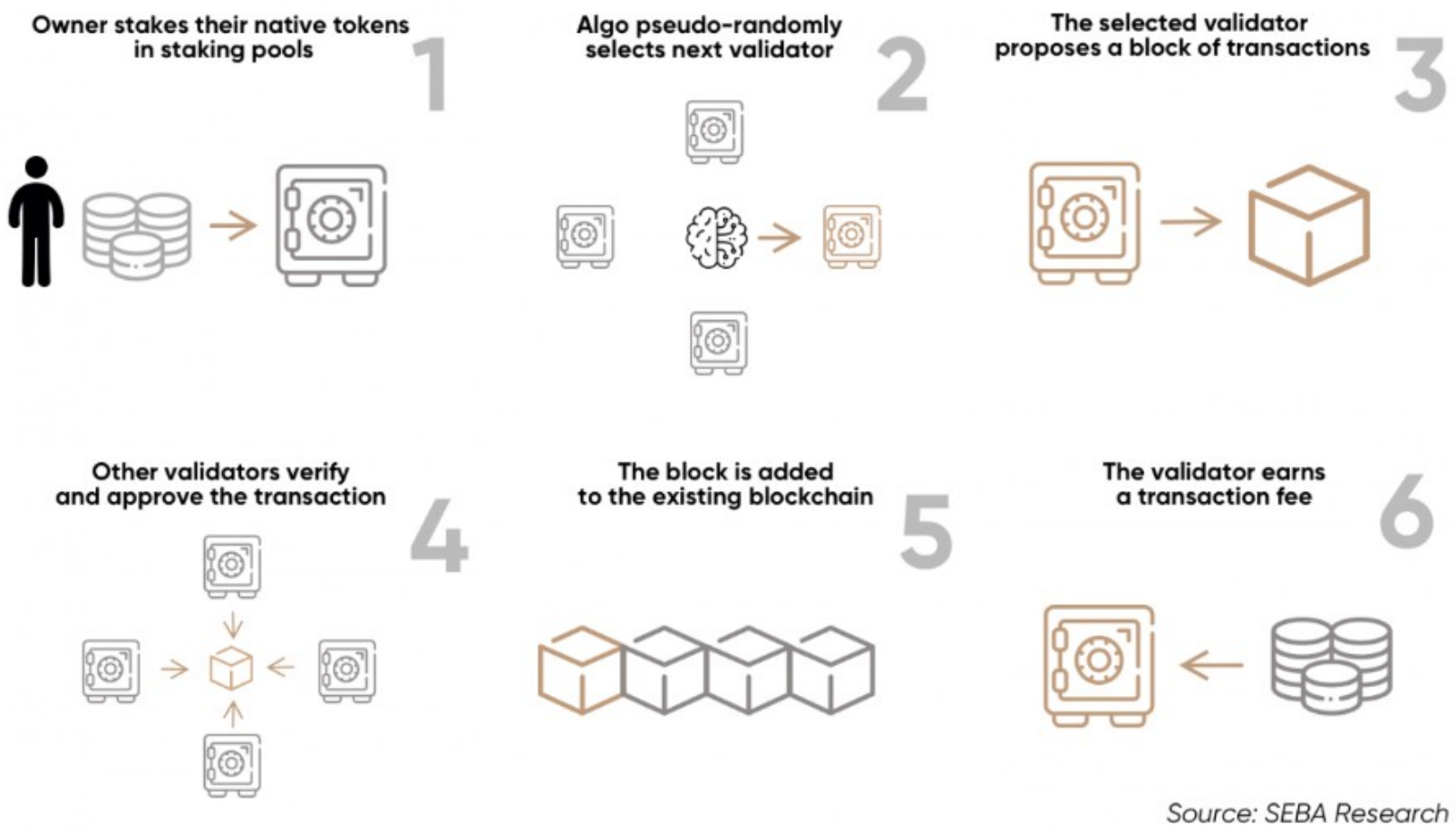

A wait for a next round competition
address of node A
timestamp
previous block hash
create a block header
input
SHA256 algorithm
block hash
output
coin age of A > block hash / target
NO
YES
A create a new block

1. Nodes perform transactions. The PoS algorithm puts all these transactions into a pool.
2. All nodes fighting to become validators for the next block raise the stake.

   This stake is combined with other factors such as “coin age” or “ random block selection” to select a validator.

   - Coin-age-based selection: The algorithm keeps track of how long each candidate validator node remains a validator. The older the node, the higher the chance of becoming a new validator.
   - Random Block selection: The validator is selected by combining “lowest hash value” and “highest stake.” The node that has the best-weighted combination of them becomes the new validator.
3. The validator verifies all transactions and publishes the block. His bet remains locked, and the forgery reward has also not yet been awarded. This is so that nodes in the network can “OK” a new block.
4. The validator will get the stake back and the reward if the block is OK. If the algorithm uses a mechanism based on coin age to select validators, the validator for the current block has its coin age reset to 0. This puts it in low priority for the next validator election.
5. If other nodes in the network do not validate the block, the validator loses his stake and is marked as “bad” by the algorithm. The process starts again from step 1 to create a new block.

How staking works in PoS



Proof of elapsed time (PoET)

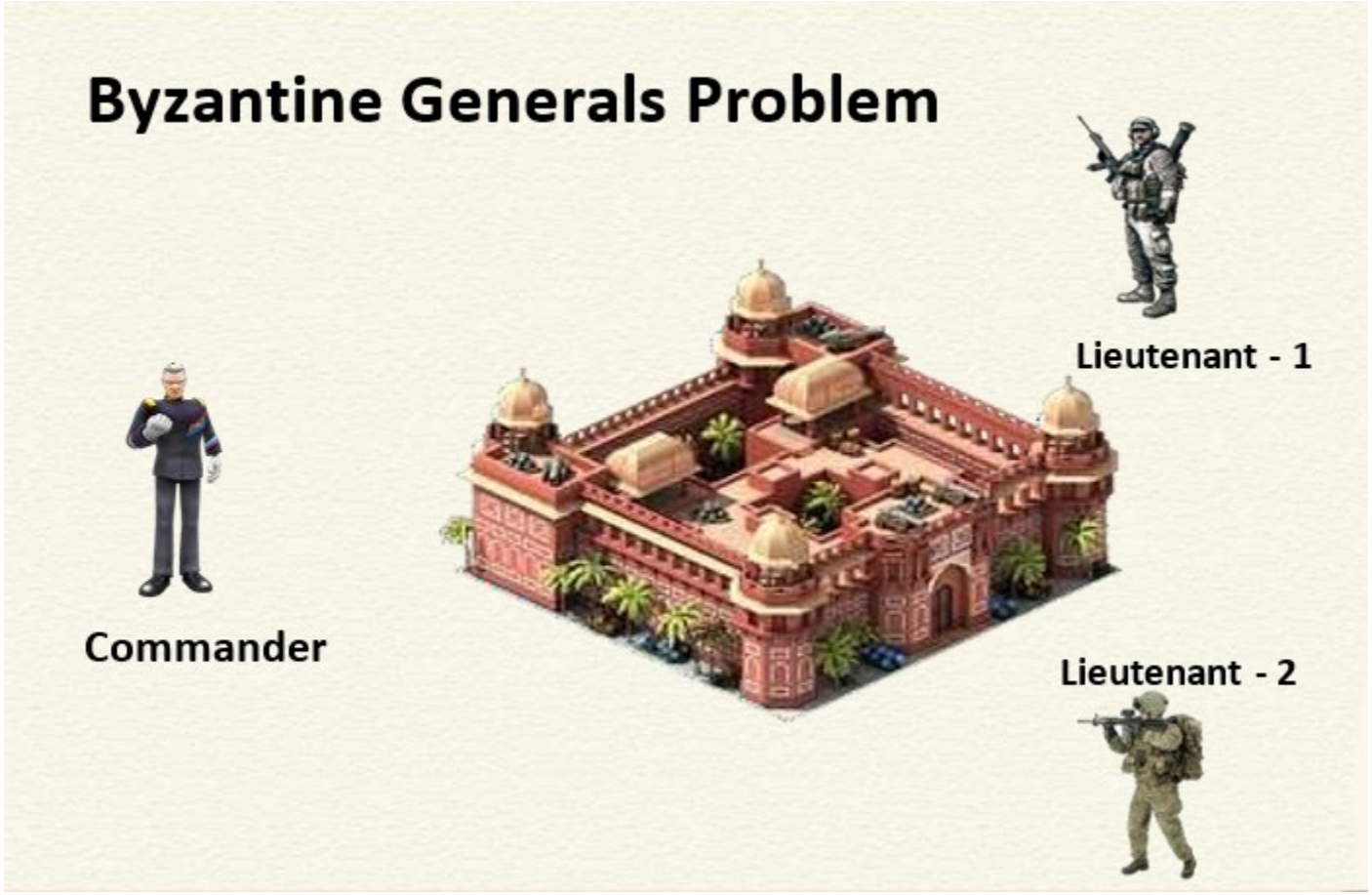
- Developed by **Intel Corporation** that enables **permissioned blockchain** networks to determine who creates the next block.
- PoET follows a lottery system that spreads the chances of winning equally across network participants, giving every node the same chance.
- The PoET algorithm generates a random wait time for each node in the blockchain network; each node must sleep for that duration.
- The node with the shortest wait time will wake up first and win the block, thus being allowed to commit a new block to the blockchain.
- The PoET workflow is similar to Bitcoin's proof of work (PoW) but consumes less power because it allows a node to sleep and switch to other tasks for the specified time, thereby increasing network energy efficiency.

Proof of Capacity (PoC)

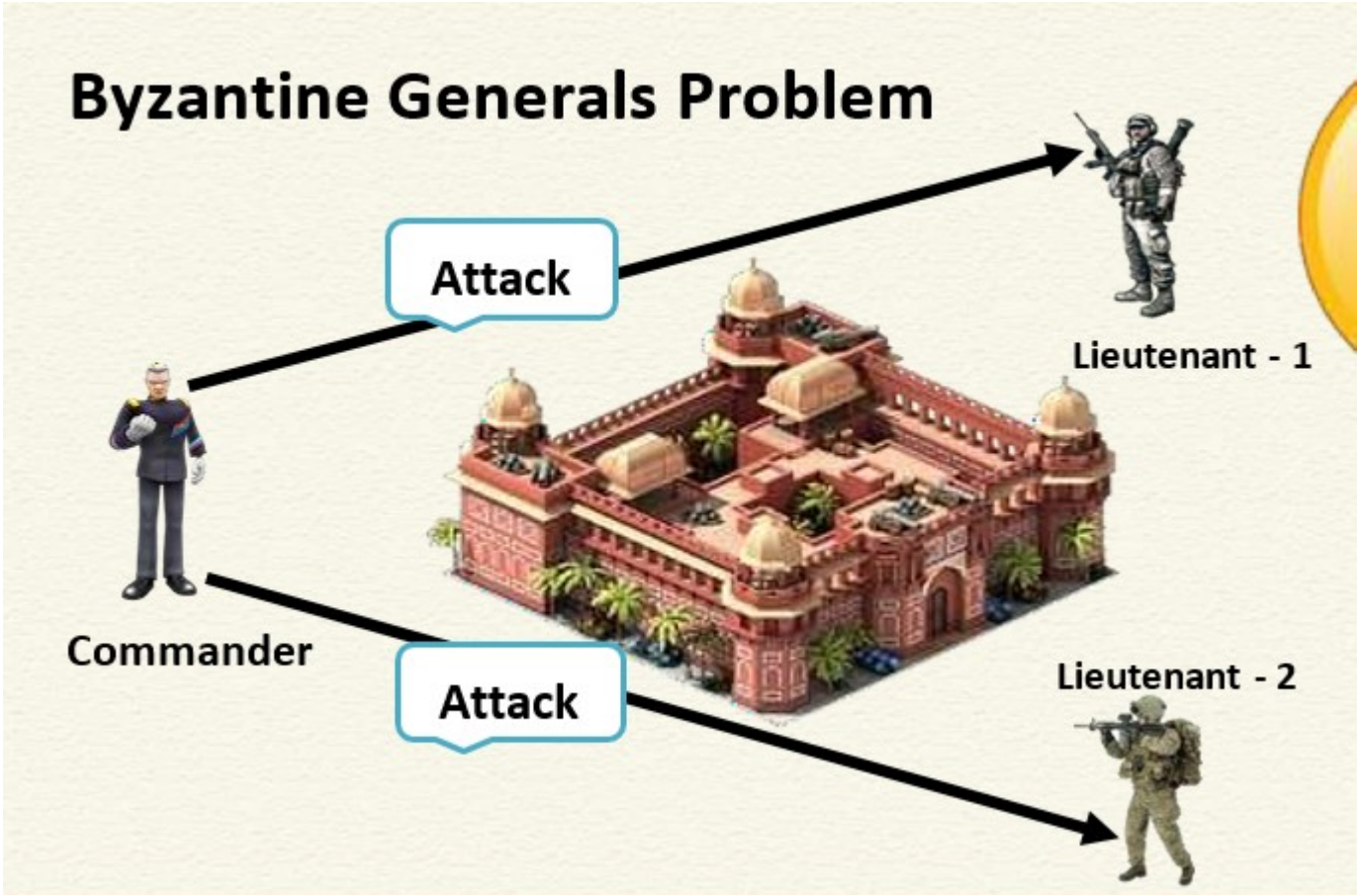
- PoC is also known as **proof of space,** is a consensus mechanism in the blockchain. Unlike Proof of Work (PoW) which uses separate mining hardware for computing power, or Proof of stake (PoS), which stakes miners' coins.

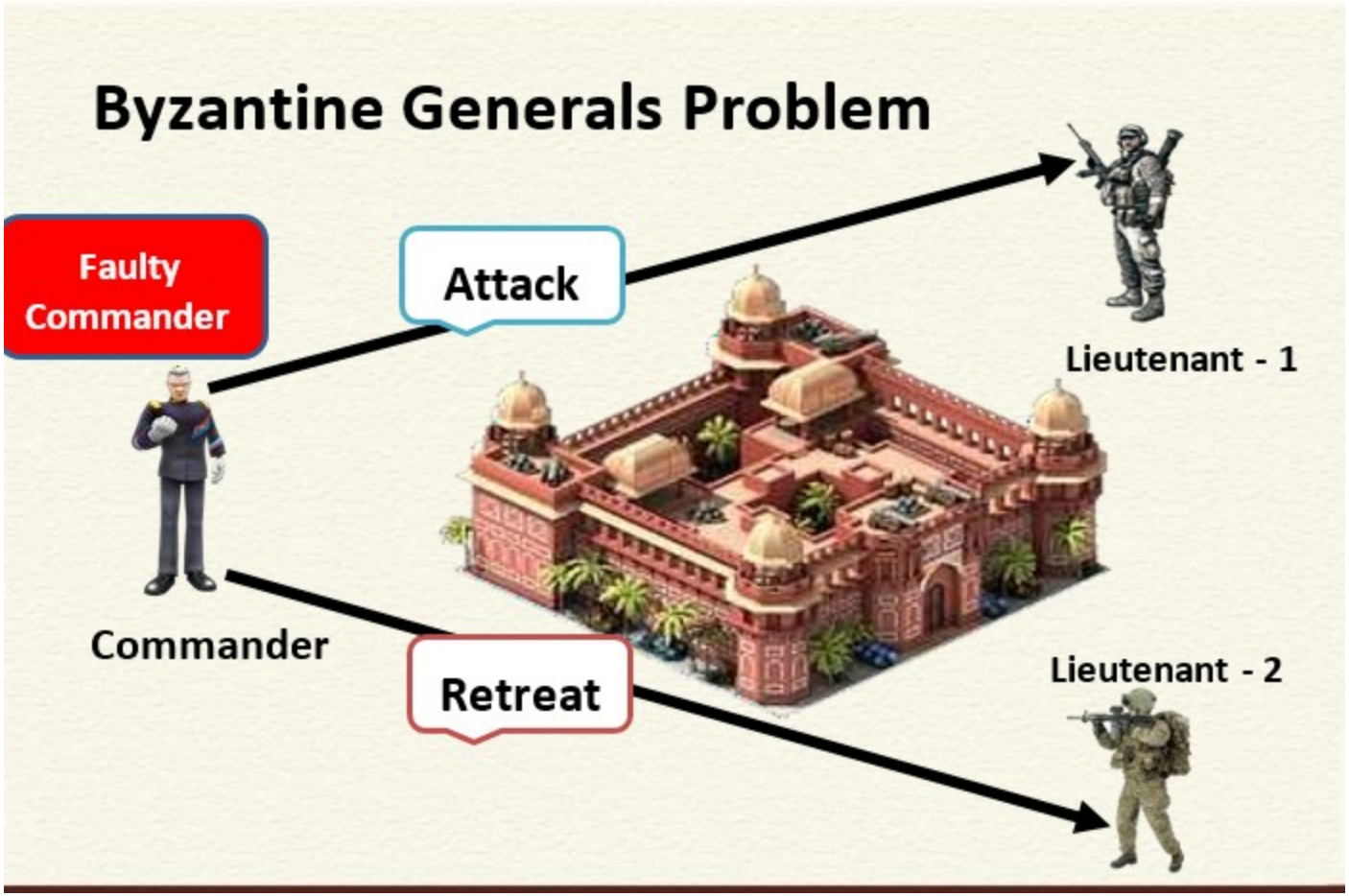
- PoC allows network participants (nodes) to use their available hard drives space for mining a new block and validating transactions**.**

Byzantine General problem

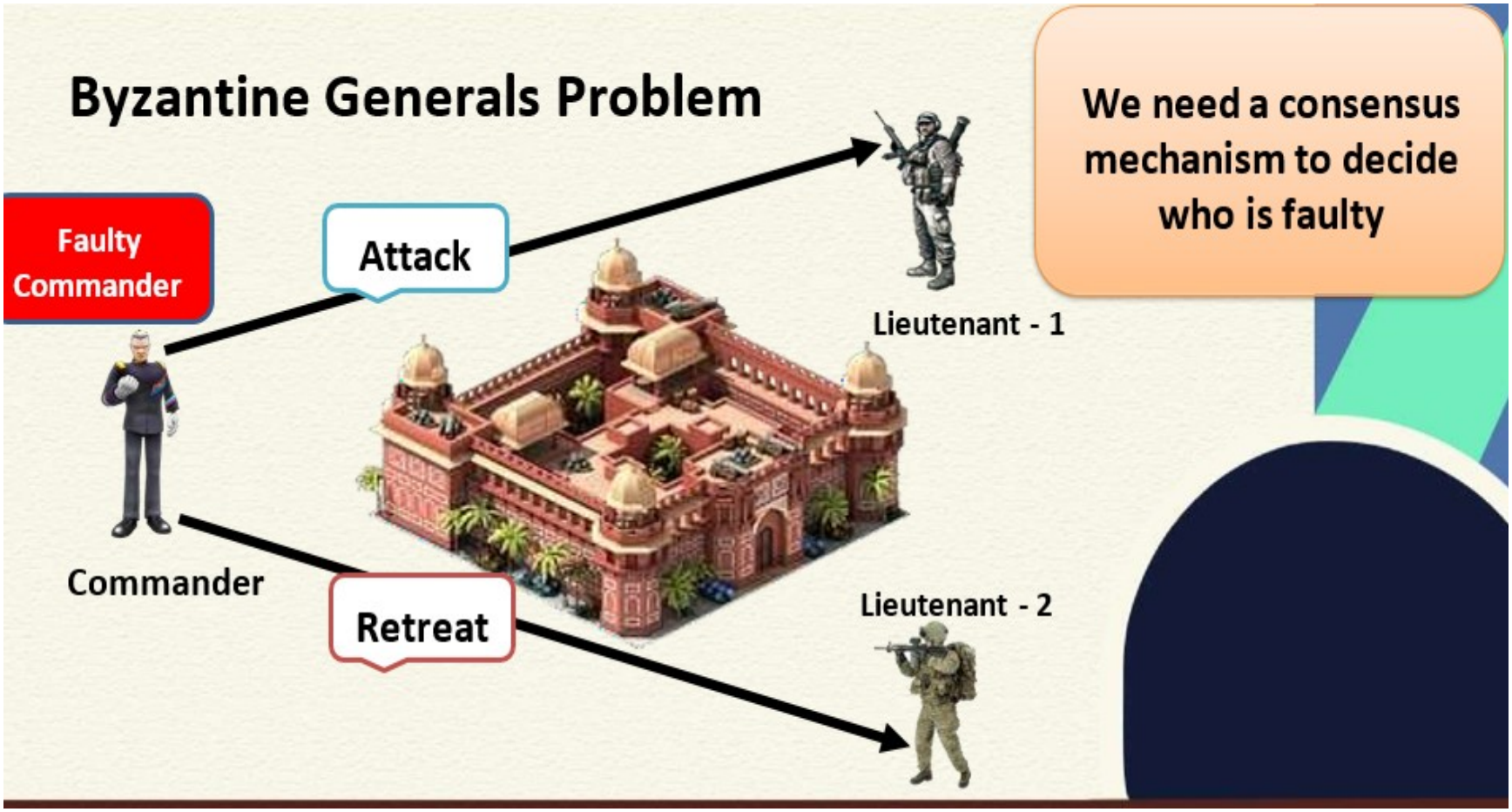


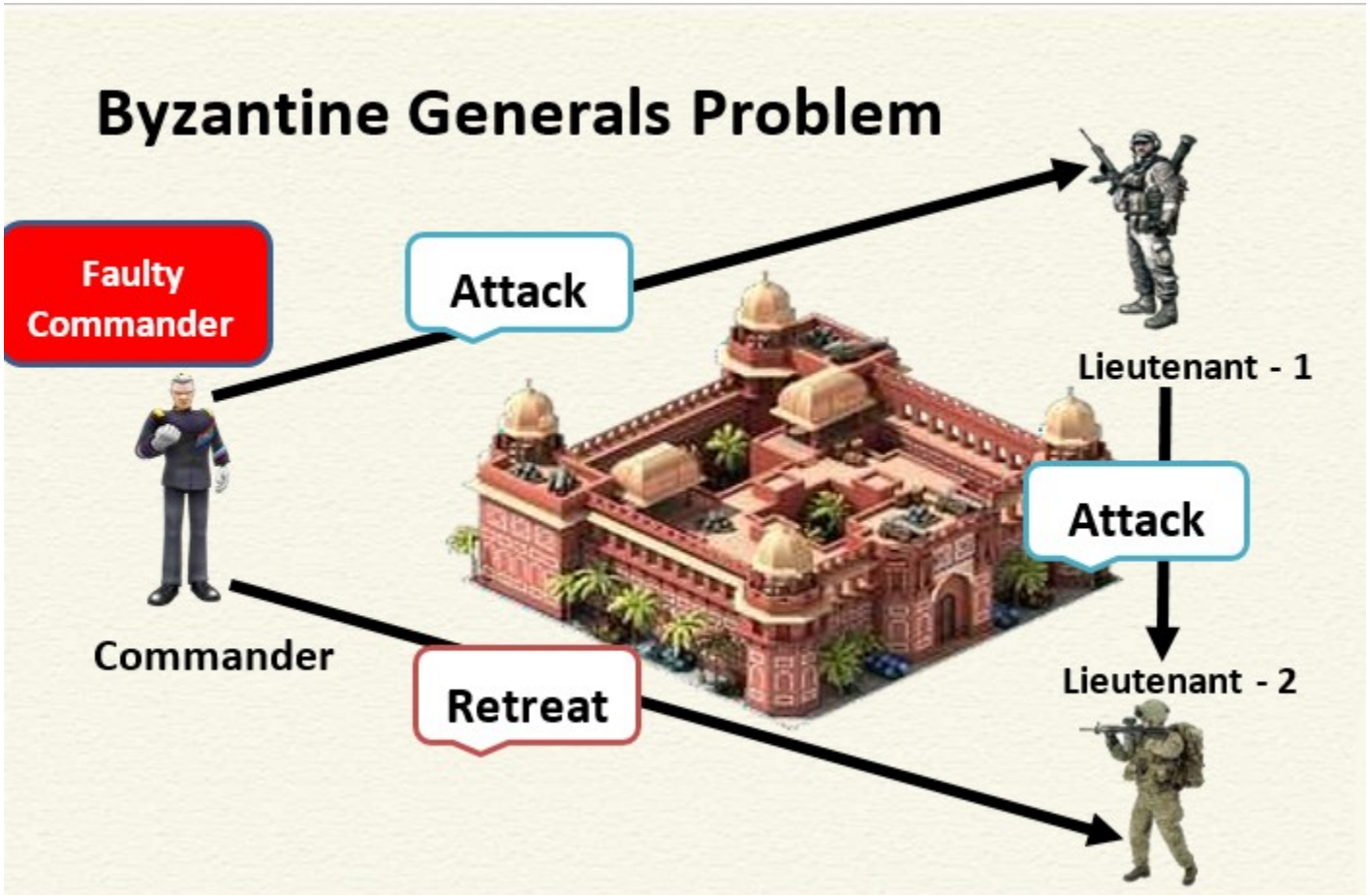
Byzantine General problem

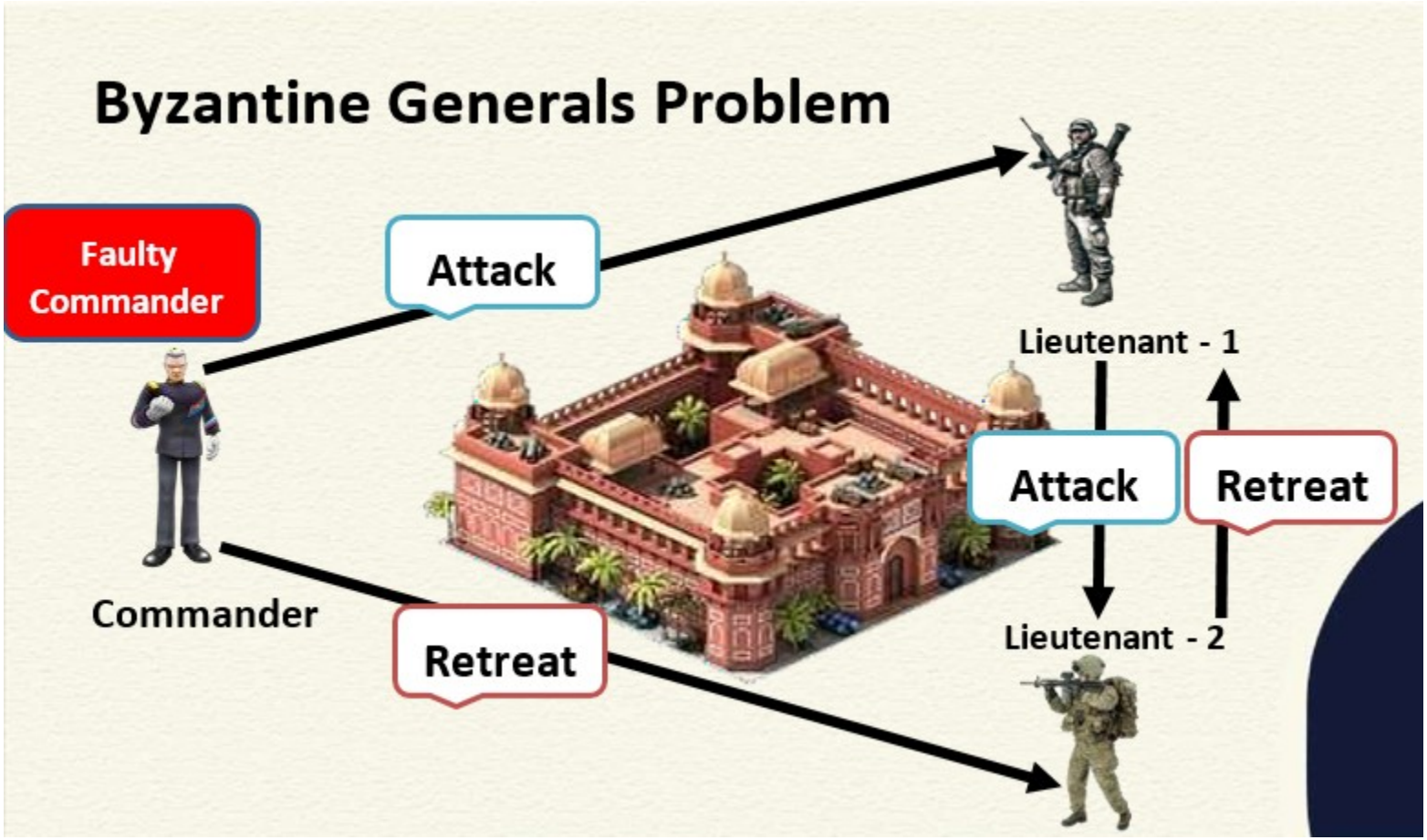


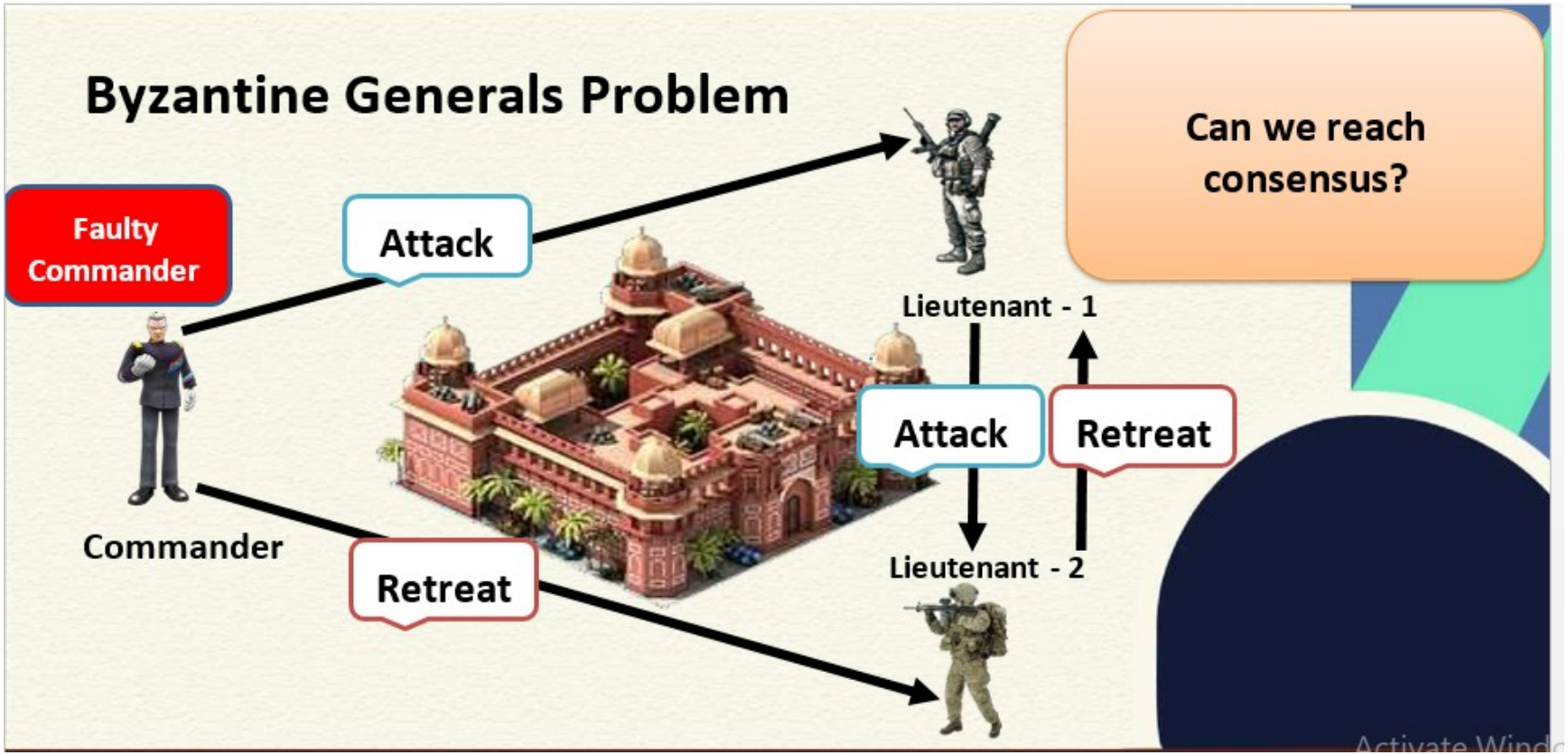

Byzantine Generals Problem
Faulty Commander
Attack
Lieutenant - 1
Commander
Retreat
Lieutenant - 2

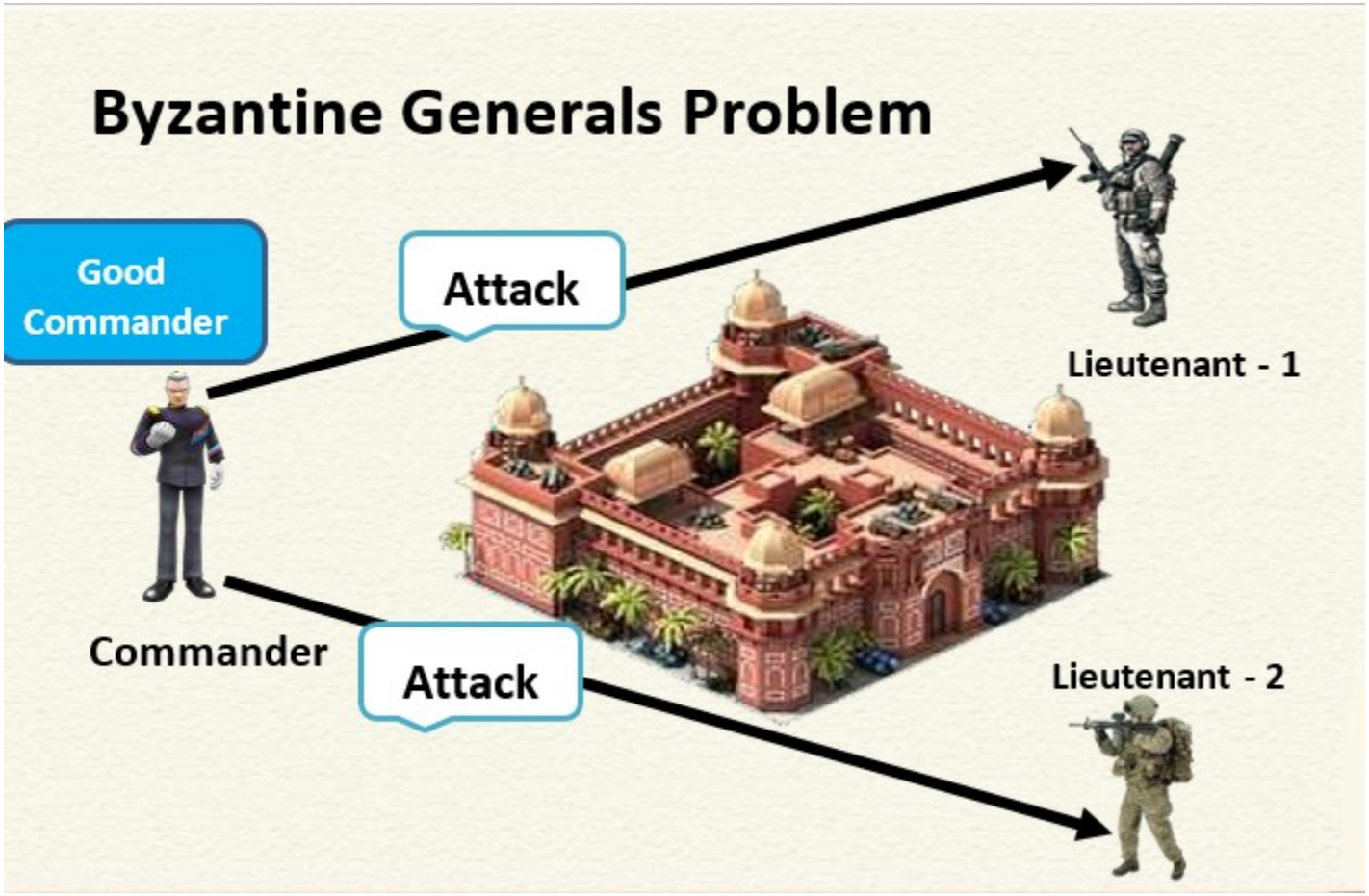
Byzantine Generals Problem

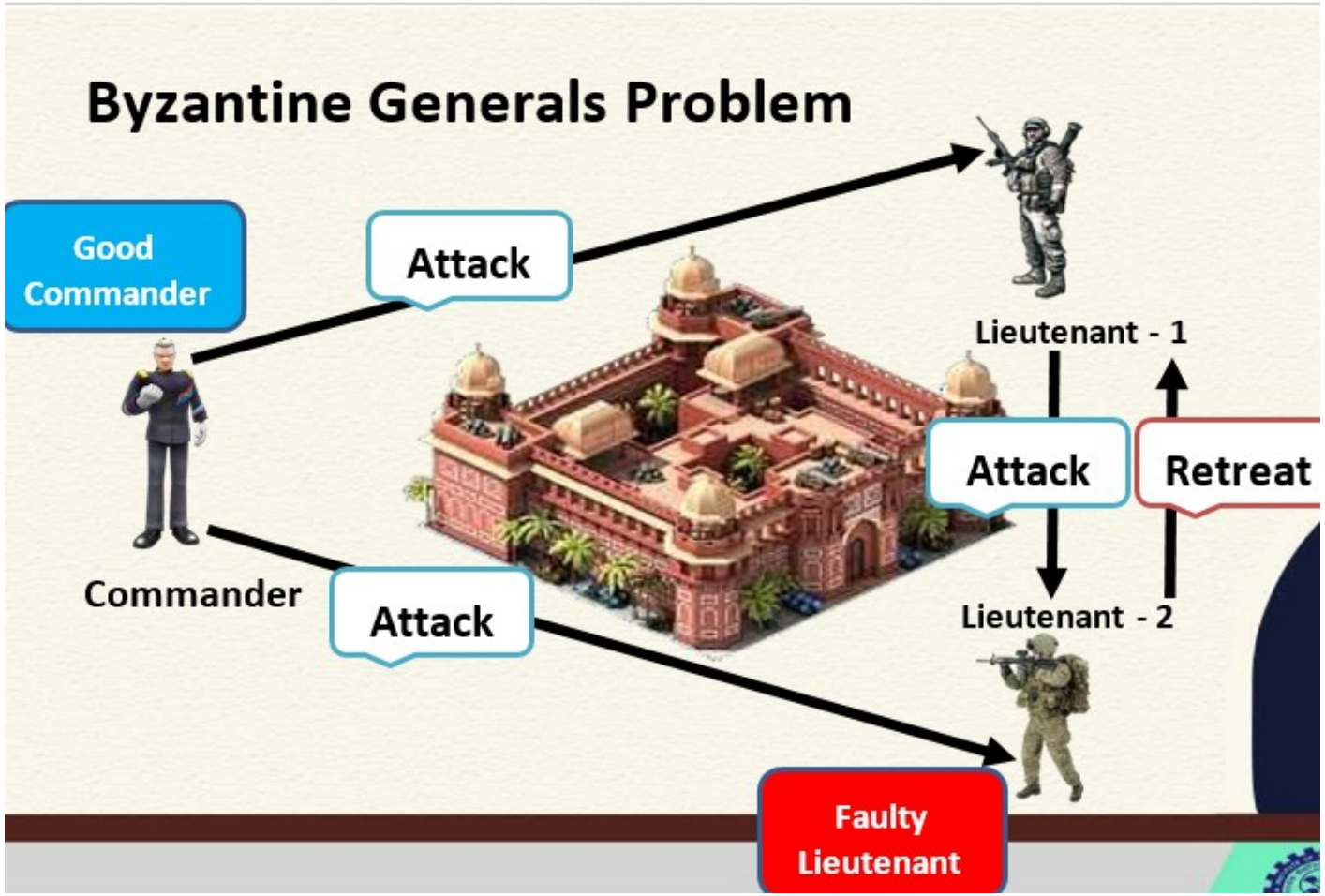


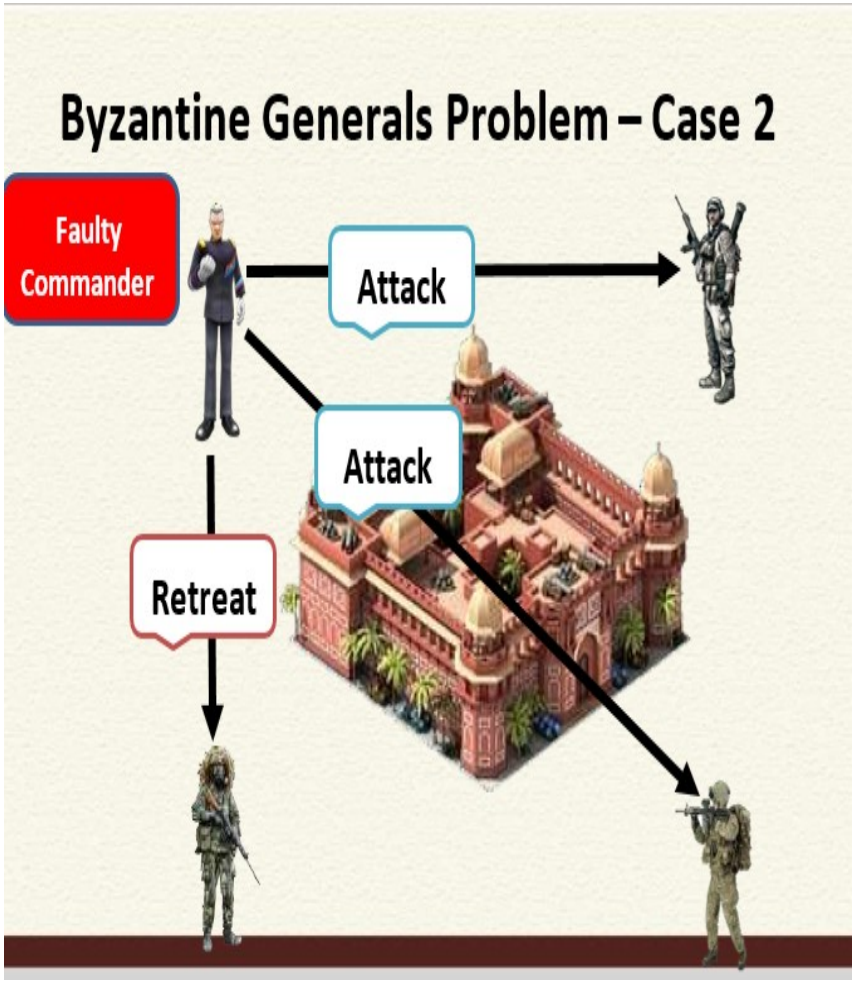

Byzantine Generals Problem
Faulty Commander
Attack
Lieutenant - 1
Attack
Commander
Retreat
Lieutenant - 2

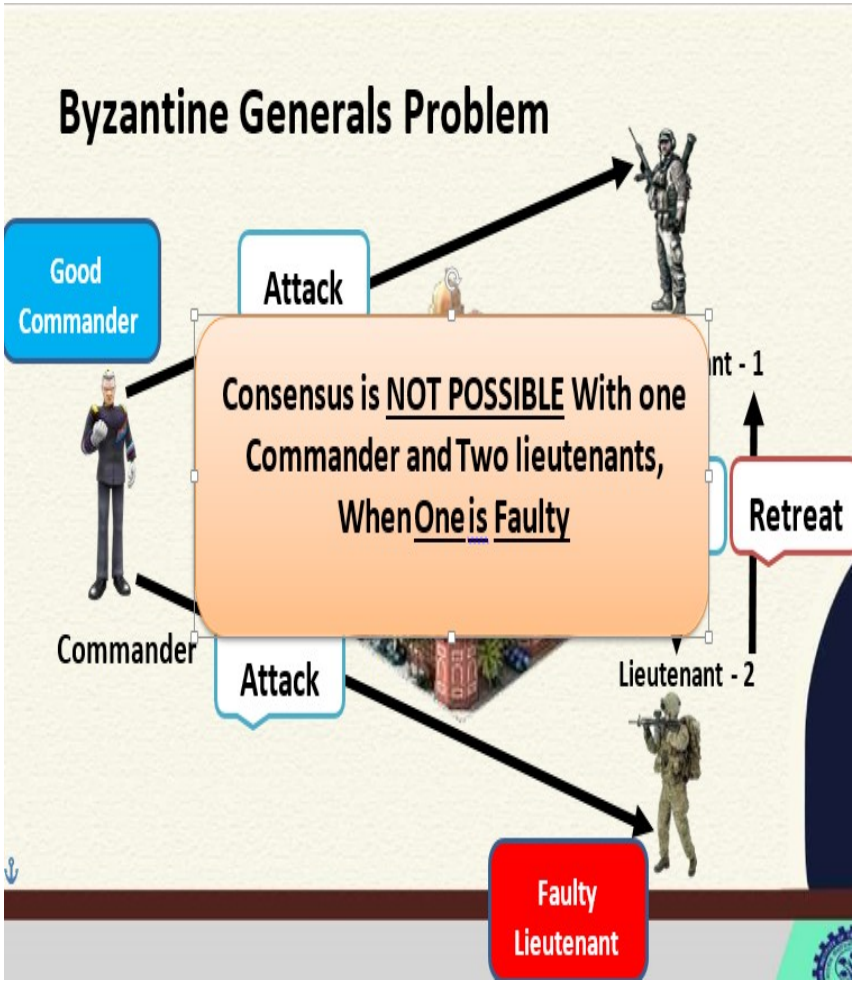

Byzantine Generals Problem
Faulty Commander
Attack
Commander
Retreat
Lieutenant - 1
Attack
Retreat
Lieutenant - 2

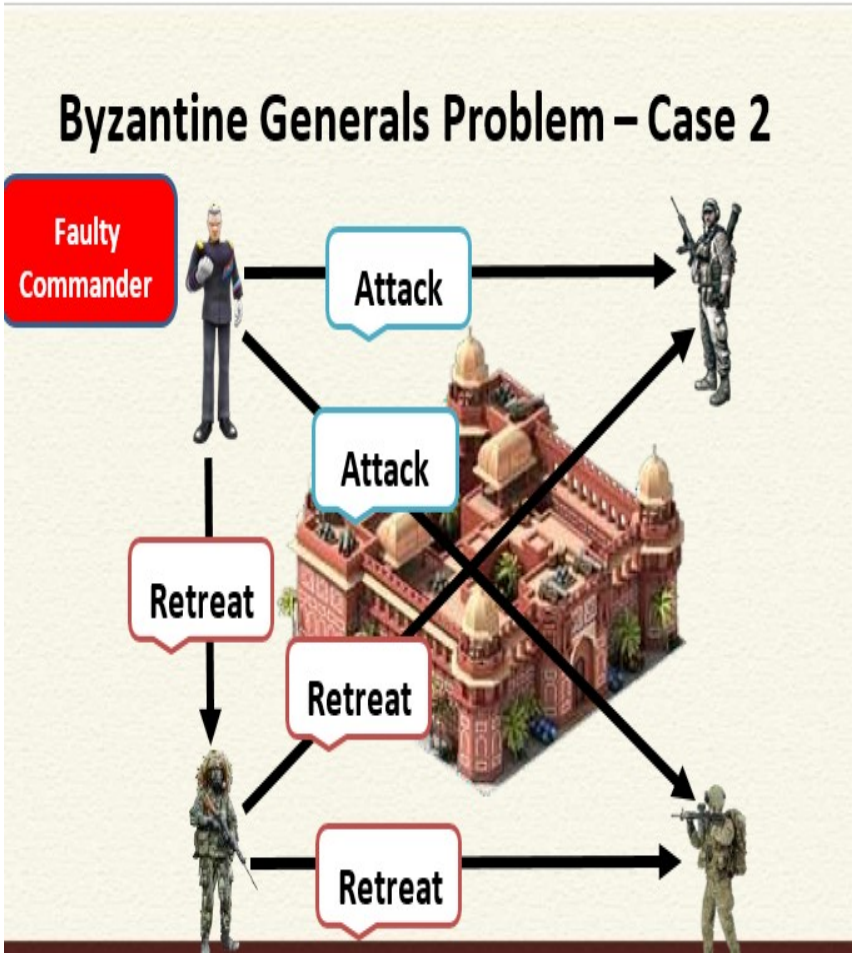

Byzantine Generals Problem
Can we reach consensus?
Faulty Commander
Attack
Lieutenant - 1
Attack
Retreat
Commander
Retreat
Lieutenant - 2

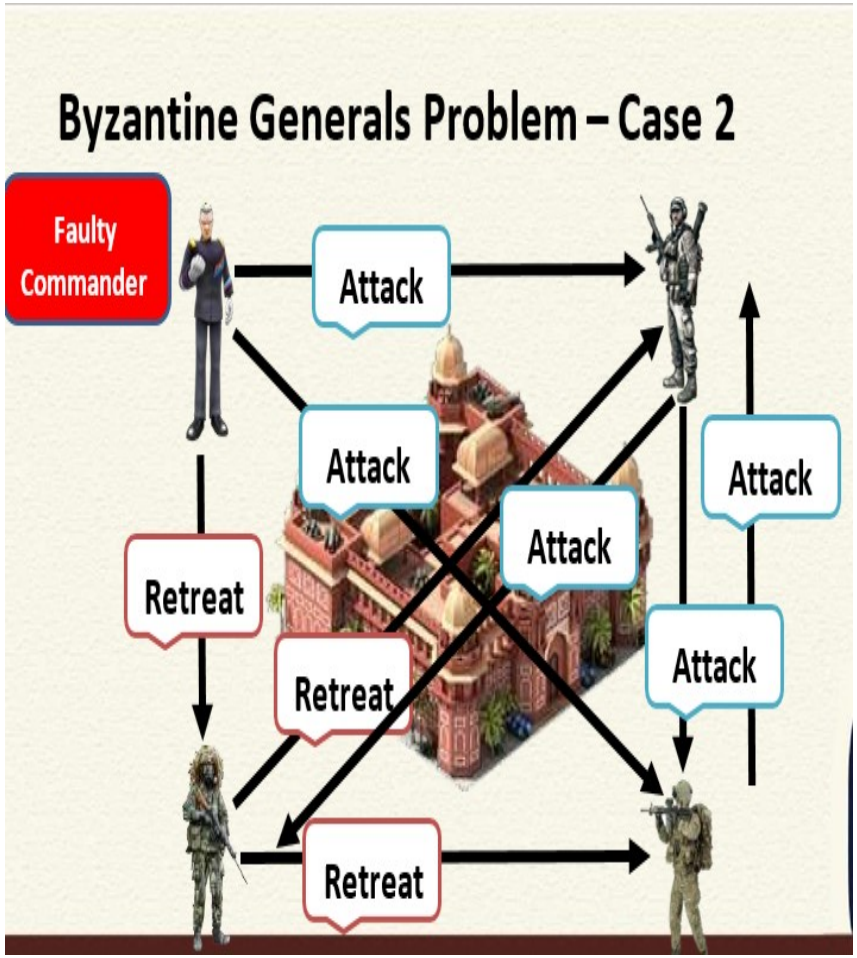

Byzantine Generals Problem
Good Commander
Attack
Lieutenant - 1
Commander
Attack
Lieutenant - 2

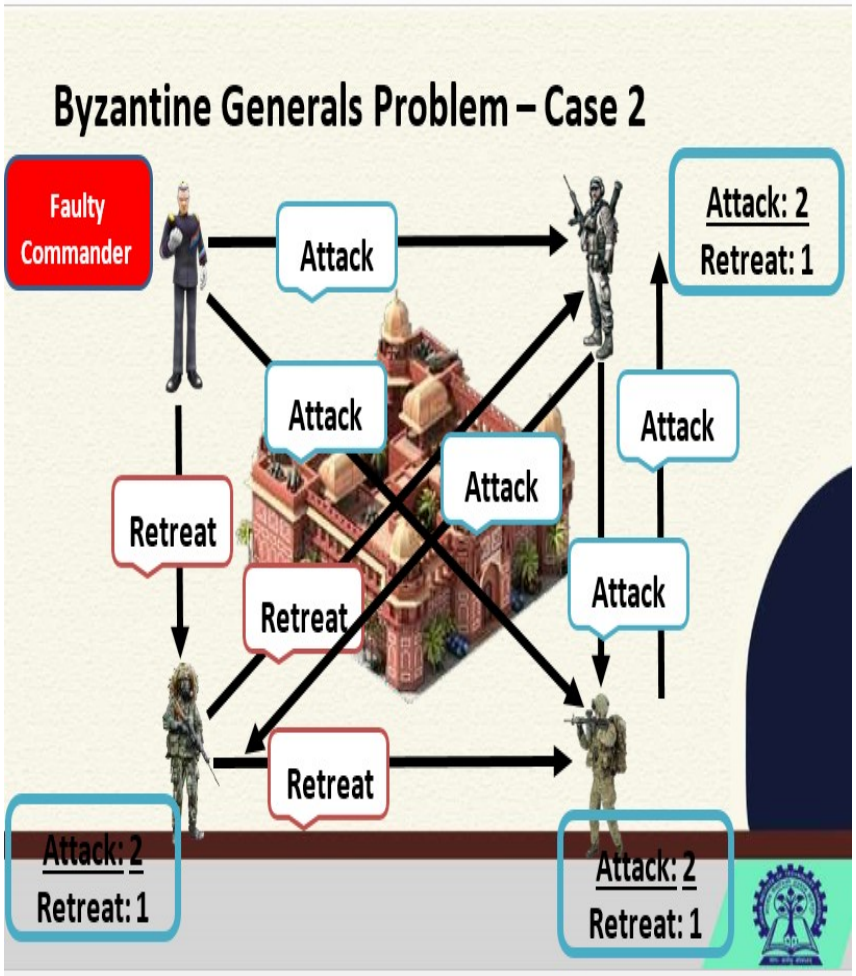

Byzantine Generals Problem
Good Commander
Attack
Lieutenant - 1
Attack
Retreat
Commander
Attack
Lieutenant - 2
Faulty Lieutenant

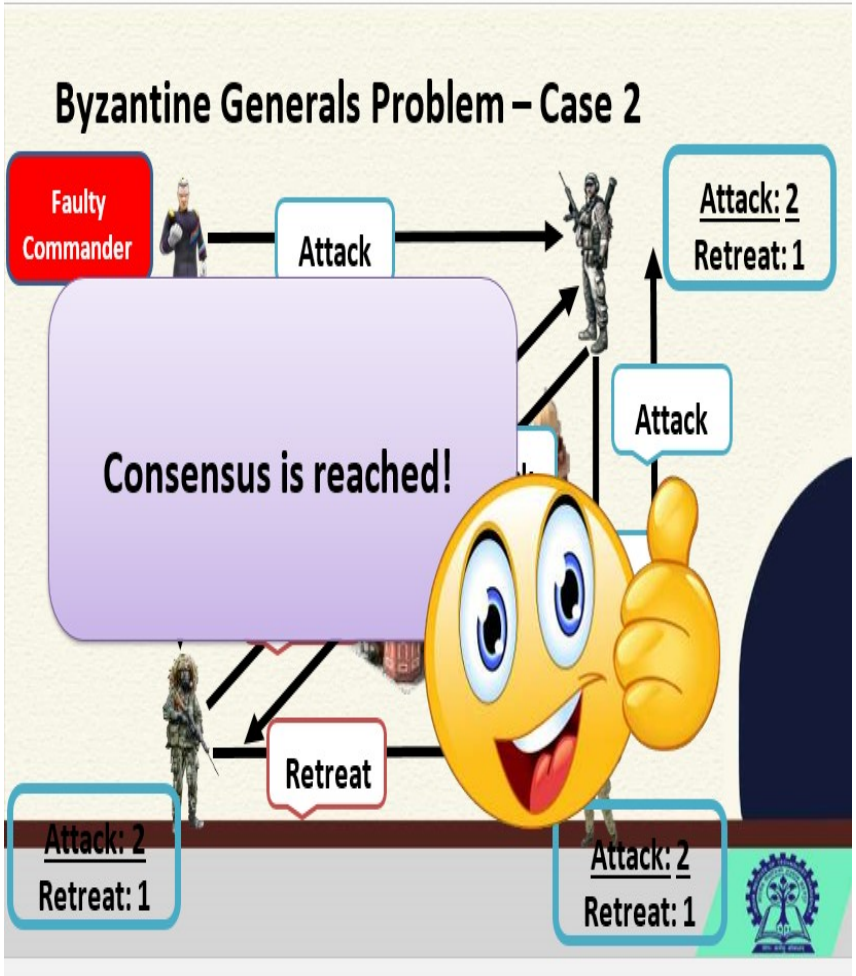

Byzantine Generals Problem – Case 2
Faulty Commander
Attack
Attack
Retreat


Byzantine Generals Problem
Good Commander
Attack
Consensus is NOT POSSIBLE With one Commander and Two lieutenants, When One is Faulty
Retreat
Commander
Attack
Lieutenant - 2
Faulty Lieutenant

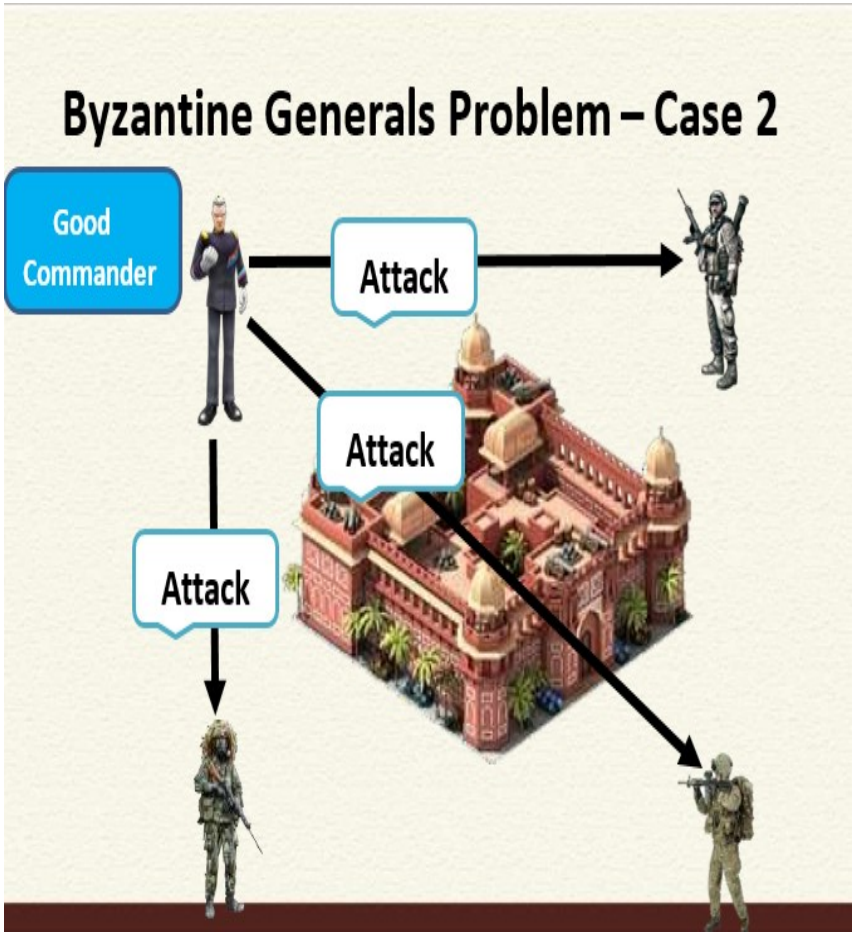

Byzantine Generals Problem – Case 2
Faulty Commander
Attack
Attack
Retreat
Retreat
Retreat

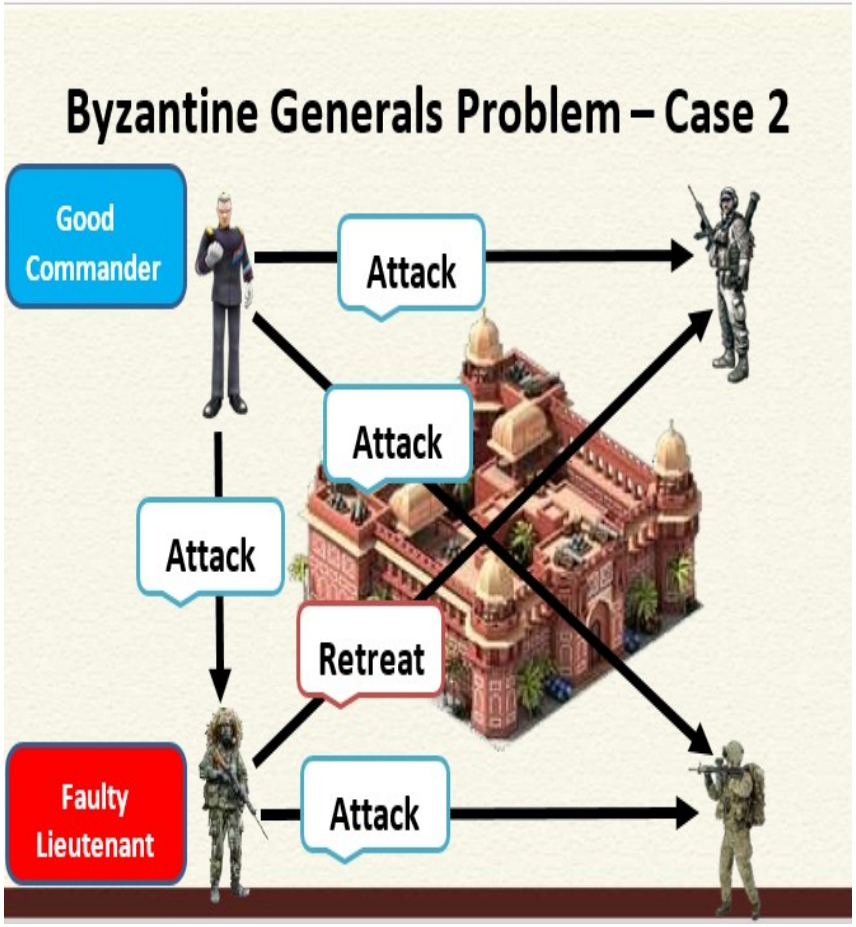

Byzantine Generals Problem – Case 2
Faulty Commander
Attack
Attack
Attack
Attack
Retreat
Retreat
Attack
Retreat

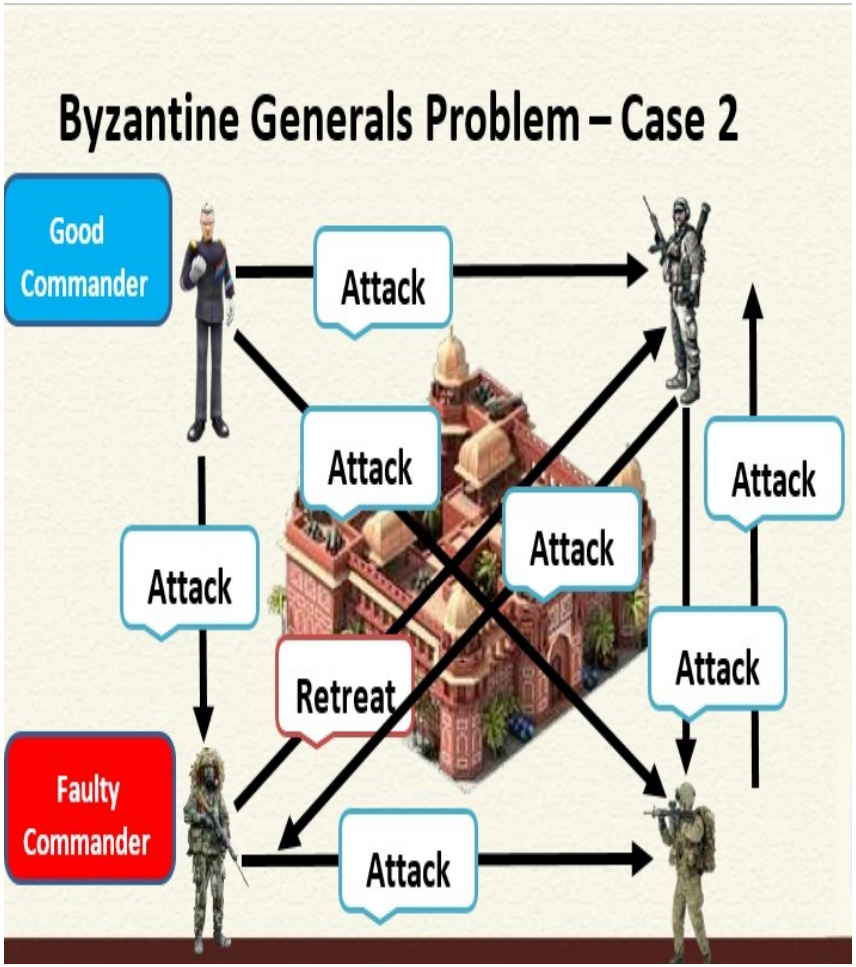

Byzantine Generals Problem – Case 2
Faulty Commander
Attack
Attack: 2
Retreat: 1
Attack
Attack
Attack
Retreat
Retreat
Attack
Retreat
Attack: 2
Retreat: 1
Attack: 2
Retreat: 1

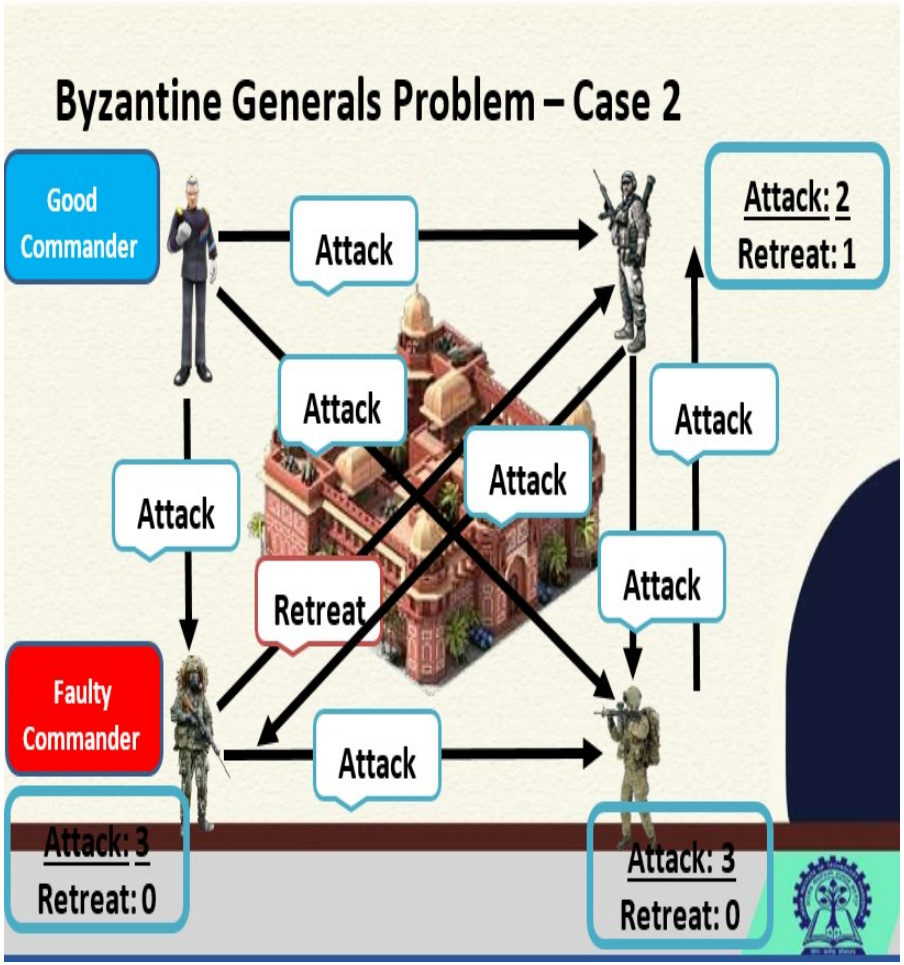

Byzantine Generals Problem – Case 2
Faulty Commander
Attack
Attack: 2
Retreat: 1
Consensus is reached!
Attack
Retreat
Attack: 2
Retreat: 1
Attack: 2
Retreat: 1

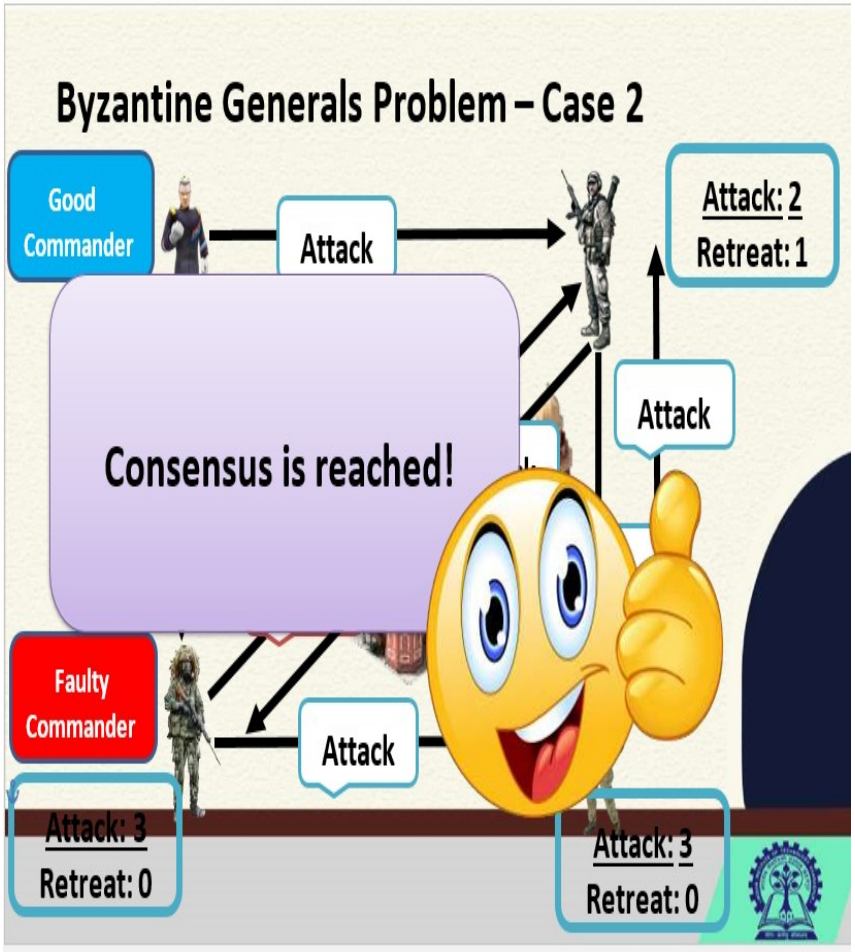
Faulty Lieutenant






Byzantine Generals Problem – Case 2
Good Commander
Attack
Attack
Attack
Attack
Attack
Attack
Retreat
Faulty Commander
Attack


Byzantine Generals Problem – Case 2
Good Commander
Attack
Attack: 2
Retreat: 1
Attack
Attack
Attack
Attack
Attack
Retreat
Faulty Commander
Attack
Attack: 3
Retreat: 0
Attack: 3
Retreat: 0



Asynchronous Byzantine Agreement

- F faulty nodes – Need 3F + 1 nodes to reach consensus
  - Faulty nodes create partition in the network

Byzantine Fault Tolerance

- Byzantine Fault Tolerance(BFT) is the feature of a distributed network to reach consensus (agreement on the same value) even when some of the nodes in the network fail to respond or respond with incorrect information.
- The objective of a BFT mechanism is to safeguard against the system failures by employing collective decision making(both - correct and faulty nodes) which aims to reduce to influence of the faulty nodes.
- BFT is derived from Byzantine Generals' Problem.