

# Distributed Consensus Algorithm

*By- Amit Kumar Vishwakarma, IIT Kanpur and Tsinghua University*





# Table of Content

1. Consensus Algorithm
2. Problem and Challenges of Distributed Consensus
3. Fault tolerance
  - Crash Fault
  - The Byzantine Fault
4. Practical Byzantine Fault Tolerance
  - Crash Fault Tolerance
  - The Byzantine Fault Tolerance
5. Proof-of-Work
6. Proof-of-Stake



# Consensus Algorithm

- Consensus algorithms are a decision-making process for a group, where individuals of the group construct and support the decision that works best for the rest of them.
- It's a form of resolution where individuals need to support the majority decision, whether they liked it or not.
- Consensus algorithms do not merely agree with the majority votes, but it also agrees to one that benefits all of them. So, it's always a win for the network.



# Objective of Consensus Algorithm

- **Coming to an agreement:** The mechanism gathers all the agreements from the group as much as it can.
- **Collaboration:** Every one of the group aims toward a better agreement that results in the groups' interests as a whole.
- **Co-operation:** Every individual will work as a team and put their own interests aside.
- **Equal Rights:** Every single participant has the same value in voting. This means that every person's vote is important.
- **Participation:** Everyone inside the network needs to participate in the voting. No one will be left out or can stay out without a vote.
- **Activity:** every member of the group is equally active. There is no one with more responsibility in the group.

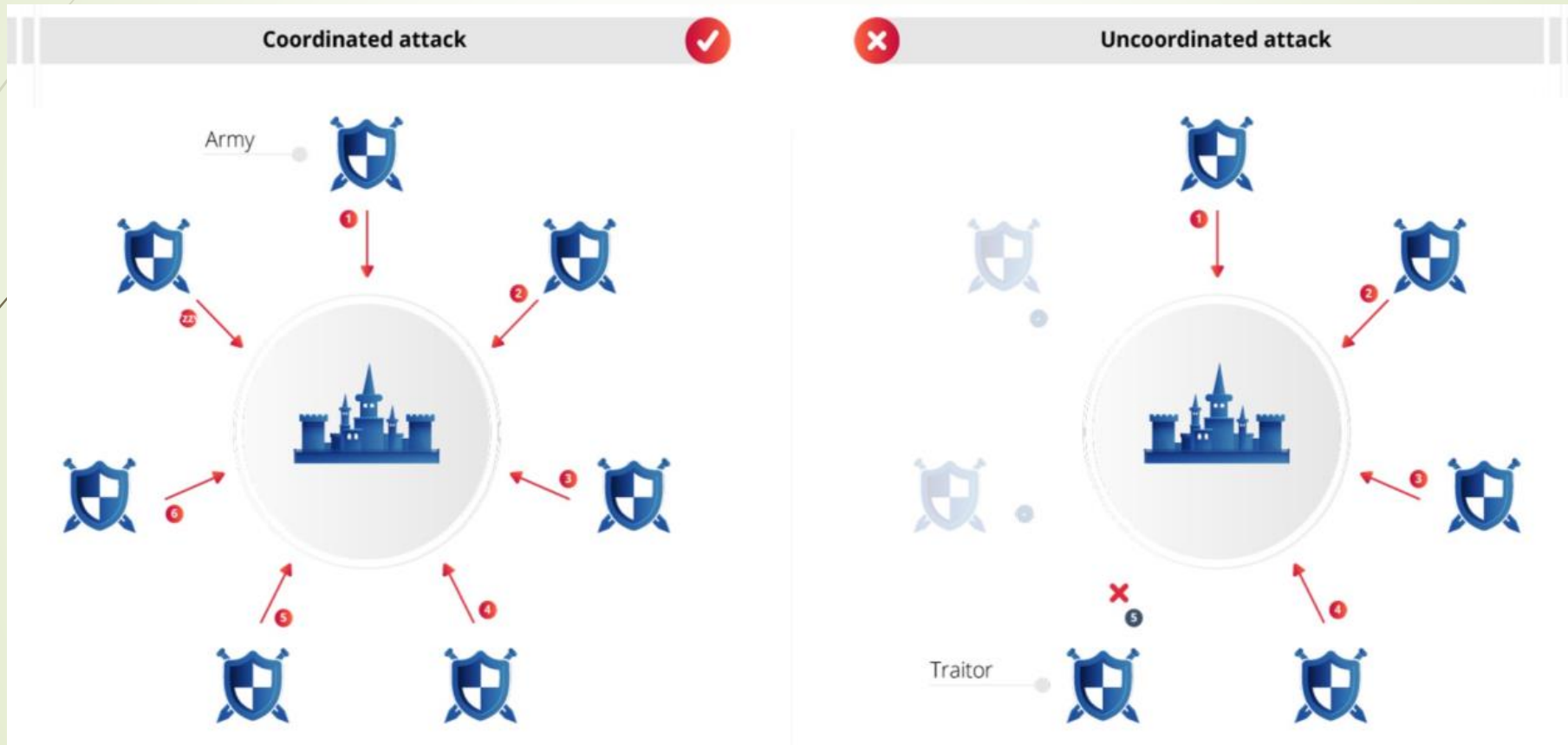
# Problem and Challenges of Distributed Consensus

## Crash Fault

A crash fault in a distributed network often may be related to one of the following issues:

- Nodes or replicas may experience downtime at any time, stop running for a short time and recover later.
- The network may be interrupted at any time.
- A sent message may be lost during delivery and cannot be received.
- A sent message may be delayed and received after a long time.
- Messages may experience the out-of-order problem during the delivery process.
- The network may be divided. For example, due to poor communication between clusters in China and the US, the entire network may be divided into two sub-networks for the China clusters and US clusters, for instance.

# The Byzantine Fault







# Fault Tolerance



## Crash Fault tolerance:

- for distributed components in many companies such as distributed storage, message queue, and distributed services, we only need to consider CFT.
- The reasons for this is as follows: The entire enterprise network is closed and protected by multiple firewalls, making external access and attacks unlikely. Individual nodes are deployed in a unified manner, and it is very unlikely that the machines and running software gets changed without proper authorization.

# Byzantine Fault Tolerance

- BFT is related to the entire distributed network being evaluated in a larger environment.
- In addition to physical hardware, it is also necessary to take some "man-made" factors.
- After all, it is specific persons instead of machines that perform misconduct.
- Assume that a distributed network is relatively open, for example, a private network of tens of companies in a specific industry. Or assume a completely open network, for example, a network that anyone has access to.
- Node machines and software on these machines are deployed by individual companies or individuals themselves.
- If the benefit is tempting enough, a person may launch DDoS attacks on one of these nodes, making authorized, often malicious changes to software code and to the code execution logic, or even the data that is persisted on disks in the network. In such case, we face bigger challenges.
- In addition to the unreliable communication networks and machine hardware, we need to consider and deal with the "troublemakers" in the system.

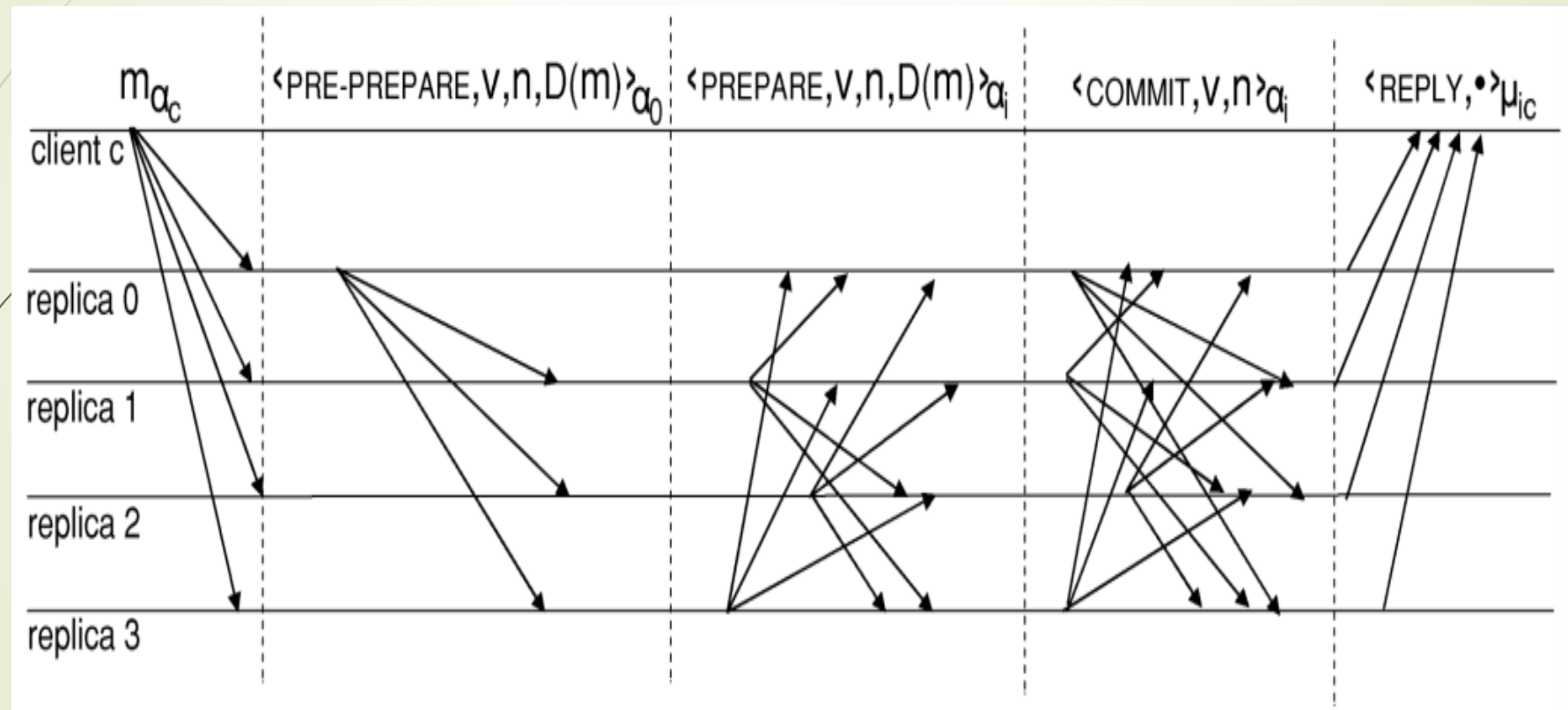




- 
- 
- Consensus algorithms can be divided into three major types by network model.
  - 1. Consensus protocols in partially synchronous models can tolerate up to  $1/3$  of any failures. In a partially synchronous model, the network latency is bounded, but we cannot know the boundary in advance. This type of fault tolerance also contains Byzantine faults.
  - 2. Deterministic protocols in an asynchronous model cannot tolerate faults. As mentioned before, in an asynchronous model, the network latency is unbounded. This conclusion is actually what the FLP impossibility theorem implies: Deterministic protocols in a completely asynchronous network cannot tolerate errors in even a single node.
  - 3. Protocols in a synchronous model can surprisingly support 100% fault tolerance, although they will limit node behaviors when the number of faulty nodes exceeds  $1/2$  of the total nodes. In a synchronous model, the network latency is bounded (smaller than a known constant).



# Practical Byzantine Fault Tolerance (PBFT)

- PBFT is the first algorithm of its kind with the complexity reduced from the exponential level to the polynomial level.
- PBFT enables several thousand TPS and feasible solutions to nodes acting maliciously in practice.
- It is proven that the PBFT algorithm will work normally if the number of malicious nodes in a system is no more than  $\frac{1}{3}$  of the total nodes.

- five phases are experienced from the client launching requests to receiving responses.




- 
- 
- **Launch:** The client (client c) initiates the service request m to the cluster.
  - **Pre-prepare:** The leader node (replica 0) verifies the validity of the request message m, assigns the sequence number n to the request m in the view and broadcasts the assigned pre-prepare message to all the backup nodes (replica 1-3).
  - **Prepare:** The backup nodes verify the validity of the request message m and accept the sequence number n. If a backup node accepts the assignment scheme, it broadcasts the corresponding prepare message to the other nodes. In this phase, all the replicas are required to reach a globally consistent order.
  - **Commit:** Once the assignment agreement message is received from the cluster, all the nodes (primary and secondary nodes) broadcast the commit message to all the other nodes. In this phase, all the replicas have agreed on the order and confirmed the received request.

- 
- 
- **Execute and reply:** After receiving the commit message from the cluster, the nodes execute the request  $m$  and send replies to the client. The client awaits the same reply from  $f+1$  different nodes and considers that the request has been executed successfully.  $f$  represents the maximum number of potential faulty nodes in the clusters. That all the nodes directly return messages to the client is also to prevent the primary node from having problems during the request.




# Limitations

- ▶ BFT algorithms may cause problems in a permissionless (open permission, without permission control) open network.
  - ▶ If a distributed network is open and can be accessed by anyone and the cost of the network access is low, it is unknown how many potentially malicious nodes may be in the network.
  - ▶ The biggest limitation of BFT algorithms is that they can only coordinate a small number of nodes (for example, no more than 100 nodes).
  - ▶ If the nodes are in the thousands, the system shows very poor performance or even fails to reach consensus, affecting the liveness and availability of the system.
- 






# The Proof-of-Work (PoW) Mechanism

- The innovative solution to distributed consensus in open networks is the Proof-of-Work (PoW) mechanism in Bitcoin.
  - A bitcoin is actually an electronic chain of digital signatures.
  - The coin owner can transfer the coin by signing the hash value of the previous transaction and the public key of the next owner, and adding these to the end of the coin.
  - The payee verifies the chain formed by the coin owner by verifying the signature.
- 



The entire network runs as follows:


- New transactions are broadcast to all nodes.
  - Each node packs the received transactions into a block.
  - Each node performs the PoW task by constantly changing the nonce for the block, to make the hash of the block meet the specified conditions.
  - Once a node completes the PoW task, it broadcasts the block to all other nodes.
  - After receiving the block, other nodes verify the validity of transactions within the block, and accept the block if the verification passes.
  - How does a node express its acceptance of the block? That is, when the next block is added, the hash value of the accepted block is taken as the previous hash value of the next block.
- 

Rob wants to send  
0.3 BTC to Laura





# Why does PoW Work?

- **Effective incentive policies:** The incentive policies effectively encourage more nodes to participate in the Bitcoin peer-to-peer network. The more nodes, the safer the Bitcoin network.
  - **PoW:** Mining and generating blocks consume CPU computational power, which artificially creates obstacles and increases costs, thereby increasing the cost of attackers.
  - **Game theory:** Incentive strategies also take into account the game balance, so that rational nodes benefit more from keeping honesty.
  - **Communication efficiency:** The communication efficiency between Bitcoin nodes is not low. You may notice that the broadcast of transactions and blocks is also involved.
- 



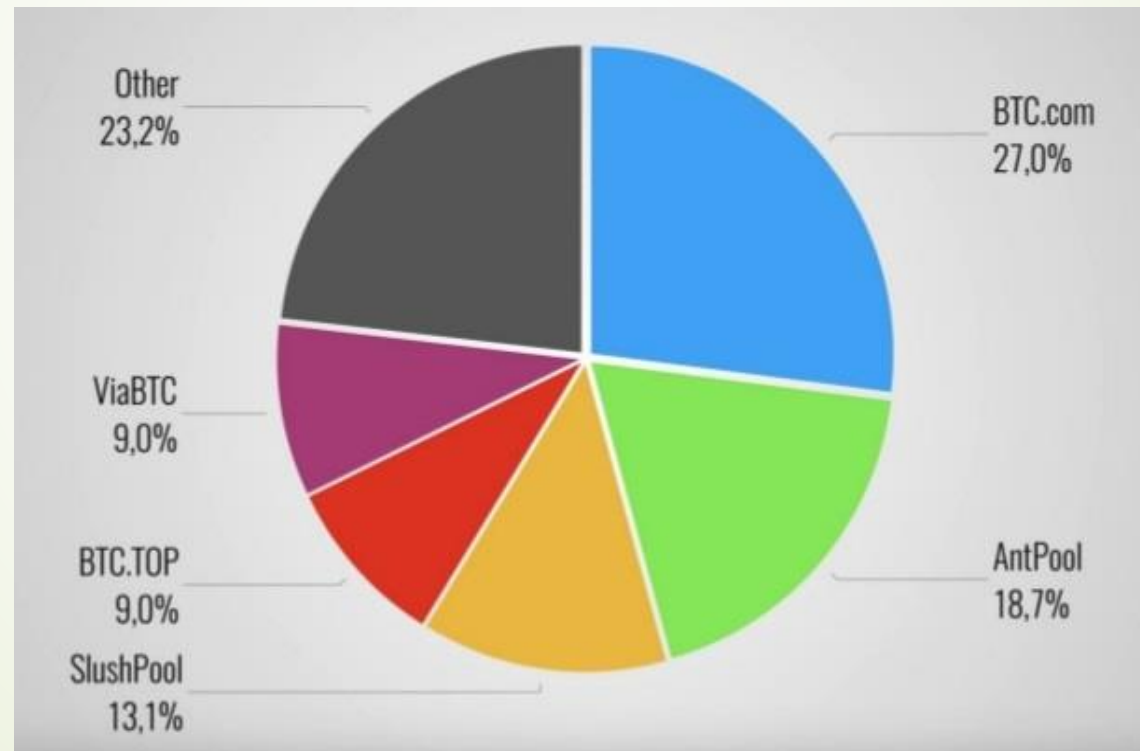
# Limitations of the PoW Mechanism

- Needs high computing power: in 2017, Bitcoin energy farms alone consumed 54 TWh (~5M US households or power Hungary/Ireland)
- Higher rewards are given to people with better and more equipment's: Higher hashrate, Higher the reward.
- Mining pools make blockchain more centralized than decentralized: Miners create mining pool to combine hashpower and share the profits evenly.
- With the emergence of specially crafted mining chips, such as ASIC and FPGA, it is almost impossible for ordinary personal PCs to dig up bitcoins.



# Concerns


- If three biggest mining pools combine they will take over the network and start approving fraudulent transactions.







# Proof-of-Stake

- Originated in 2011 by Quantum Mechanic (Bitcointalk.org)
  - Creator of a new block is chosen in a deterministic way, depending on its wealth, also defined as a stake.
  - No block rewards, so the validators take the transaction fees.
- 

# How it works?

Transactions are bundled together into what we call a block



Validators will stake (personal wealth as security deposit) to be randomly selected in a deterministic way



Validators will mint/forged a new block.



Validators take the transaction fees inside a block. They lose a part of stake if they verify fraud transactions.



Verified transactions are stored in the public blockchain

## PROOF OF WORK



The probability of mining a block is determined by how much computational work is done by the miner.



A reward is given to the first miner to solve the cryptographic puzzle of each block.



Network miners compete with one another using computational power. Mining communities tend to become more centralized over time.

## PROOF OF STAKE



The probability of validating a new block is determined by how large of a stake a person holds (how many coins they possess).



The validators do not receive a block reward, instead they collect network fees as their reward.



Proof of Stake systems can be much more cost and energy efficient than Proof of Work systems, but are less proven.

# Different Types of Consensus Algorithms

