Tendermint

In distributed systems, replication of information on all machines is fundamental. In our case, Tendermint is responsible for replicating securely and consistently the state-machine amongst the nodes of the system.

It is composed of a Byzantine-Fault-Tolerance (BFT) Proof-of-Stake (PoS) consensus algorithm and a peer-to-peer networking protocol. It communicates to the application layer through the ABCI protocol.

Tendermint allows developers to focus on the application level and not take care of peer discovery, block propagation, consensus, etc …

Byzantine-Fault-Tolerance (BFT)

In short, BFT represents the ability of a system to continue operating even if some of its nodes fail or act maliciously. In the Tendermint case, it can only tolerate up to 1/3 of failures, meaning that the blockchain will halt momentarily until 2/3 of the validator set comes to consensus.

Unlike Nakamoto consensus where it is subject to 51% attack (meaning that 51% of the actors acting maliciously could attack and alter the blockchain), Tendermint is more resistant as it is subject to a 66% attack.

Consensus

Tendermint consensus module relies on Proof-of-stake and BFT.

Let's take a look at the consensus process from a high-level standpoint:

1. Transactions are received by the node and go into a local cache mempool
2. Before going into the node mempool, Tendermint asks the application if the transaction is valid through "CheckTx" ABCI message
3. If it's valid, transaction is added to the mempool and broadcasted to the peer nodes
4. A new consensus round is initiated with a Proposer Node
5. The Proposer selects transactions in the mempool to be included in a new block
6. This proposed block is broadcasted to all nodes (Pre-vote phase) and nodes verify that the block is valid, simultaneously verifying that the Proposer is also valid and sign the pre-vote message
7. If >2/3 of nodes pre-vote for this block, block is valid (but not committed) and we enter the pre-commit phase
8. Same as pre-vote, Tendermint will wait for >2/3 of nodes to sign the pre-commit message
9.
    - There are two stages of voting to tolerate Byzantine faults, meaning that the pre-commit ensures that enough validators witnessed the result of pre-vote stage
10. *
11. The block is then committed, broadcasted to all nodes and transactions in this block executed by the application to update its state (for example account balance update etc)
12.
    - Once a block is committed, there is no way to revert it and that gives an instant finality
13.
    - All nodes are processing the transactions of every block even if they are not the block proposer
14. *
15. A new round is proposed and a new proposer is designated
16.

For a graphic view of this consensus process check What is Tendermint?

This consensus ensures that all nodes maintain the same blockchain, i.e. the same list of blocks containing the past transactions and that all nodes could propose a block through Proposer rotation.

Validator voting power - important for DPOS

Users, known as delegators here, can choose which validator they want to delegate based on their reputation, stability, security and infrastructure. The amount of the native chain token owned and delegated to a validator represents its voting power, giving them the opportunity to be a new block proposer more often. Decentralization here is "measured" by the voting power distribution amongst the validators and not the number of validators.

ABCI

ABCI layer is the communication protocol for Tendermint to make requests to the Application, like CheckTx (as we saw in the consensus explanation), indicate a Begin or End of a block, Deliver transactions to the application through DeliverTx, Query the application for account balance for example.

Finally, below a figure representing the Tendermint stack and all elements we went through in this paragraph:

Source : Tendermint in a Nutshell

Last updated11 days ago On this page * Byzantine-Fault-Tolerance (BFT) * Consensus * ABCI

Was this helpful? Edit on GitHub Export as PDF