Introduction

Blockchain @ uninsubria

Roberto Vicario

2024/2025

Contents

1 Blockchain

The *Blockchain* is a registry technology that allows to record information in an immutable, transparent and secure way without the need for a central authority. It is called "blockchain" (chain of blocks) because the data is grouped into blocks, linked together in chronological order through cryptography.

> **Decentralization:** Control and decision-making aren't held by a central authority, like a company or government. Instead, power is distributed across a network of independent participants. Data is shared among many nodes (computers) distributed in a peer-to-peer network.

Key components:

- **Transaction:** A record of an action or event, like sending money.
- **Block:** A container for a list of transactions. Each block contains a unique identifier (hash) and a reference to the previous block, forming a chain. The first block in the chain is called the *Genesis Block*.
- **Node:** A computer that participates in the blockchain network, maintaining a copy of the ledger and validating transactions.

1.1 State Machine Replication (SMR)

A blockchain is an extension of the *State Machine Replication (SMR)* model. In this model, a distributed system maintains a consistent state across multiple nodes. Each node processes transactions in the same order, ensuring that all nodes reach the same state.

$\boxed{DEFINITION}$

Let S be a set of states, T be a set of transactions, and f be the function that maps a state and a transaction to a new state.

Assumption: If all replicas receive the same sequence of transactions $t_1, t_2, \dots, t_n$, they will all reach the same final state s_n and produce the same output.

The state machine replication model can be defined as:

$$S = \{s_0, s_1, s_2, \dots, s_n\}$$

$$T = \{t_0, t_1, t_2, \dots, t_m\}$$

$$f : S \times T \to S$$

Where:

- s_0 is the initial state of the system.
- s_i is the state of the system after processing transaction t_i.
- $f(s_i, t_i) = s_{i+1}$ is the function that updates the state of the system after processing transaction t_i.

1.2 Blockchain Stack

The main layers of the blockchain stack are:

1. **Network Layer:** The underlying infrastructure that connects all nodes in the blockchain network. It ensures that data can be transmitted securely and efficiently between nodes.
2. **Consensus Layer:** The protocol that allows nodes to agree on the state of the blockchain. It ensures that all nodes have a consistent view of the ledger, even in the presence of faulty or malicious nodes.
3. **Data Layer:** The actual data stored in the blockchain, including transactions and smart contracts. This layer is responsible for maintaining the integrity and immutability of the data.
4. **Execution Layer:** The user-facing applications and interfaces that interact with the blockchain. This layer includes wallets, decentralized applications (dApps), and other tools that allow users to interact with the blockchain.

2 Consensus Problem

In decentralized systems, no single person or computer is in charge, so how do all the independent nodes agree on what's true? That's the *Consensus Problem*:

> **Problem:** *How can a group of participants, who don't fully trust each other, agree on a single version of the truth?*

2.1 Consensus Algorithm

A *Consensus Algorithm* is a mechanism used in distributed systems to achieve agreement on a single data value among distributed processes or systems. It is essential for ensuring that all nodes in a blockchain network have a consistent view of the ledger. Consensus algorithms are designed to handle various types of failures, including network partitions, node crashes, and malicious behavior.

$\boxed{ASSUMPTIONS}$

An algorithm reaches consensus under the following assumptions:

- **Safety:** The system guarantees that all honest nodes will agree on the same value, even in the presence of faulty nodes.
- **Liveness:** The system guarantees that a decision will eventually be reached, provided that a sufficient number of honest nodes are present.

2.2 Permissioned vs. Permissionless

In a blockchain network, participants can be classified into two categories based on their access rights:

- **Permissioned:** Only authorized participants can join the network and validate transactions. This type of network is often used in private blockchains or consortium blockchains, where a group of organizations collaborates.
- **Permissionless:** Anyone can join the network and participate in the consensus process. This type of network is typically used in public blockchains, where anyone can become a node and validate transactions.

3 Network Models

A blockchain network can be implemented in various ways, depending on the requirements and goals of the system. The implementation can be classified into three categories based on the communication model used by the nodes:

3.1 Synchronous

In a *Synchronous* network, all nodes are assumed to have synchronized clocks and can communicate with each other within a known time frame. This means that messages sent between nodes will arrive within a predictable time limit.

$\boxed{ASSUMPTIONS}$

This model assumes that:

- **Timing:** All nodes have synchronized clocks, meaning they can agree on the current time.
- **Message Delivery:** Messages sent between nodes will arrive within a known time frame.
- **Processing Time:** The time taken to process a message is known and bounded.

> **Consideration:** These networks are easier to reason about, but they are less common in real-world scenarios due to network latency and clock synchronization issues.

3.2 Asynchronous

In an *Asynchronous* network, there are no assumptions about the timing of message delivery or the processing time of messages. Nodes can communicate with each other, but there is no guarantee that messages will arrive within a specific time frame. Assumptions are mainly the opposite of synchronous networks, the service could be delayed or unavailable.

> **Consideration:** These networks are more realistic for real-world scenarios, but they are harder to reason about due to the lack of timing guarantees. They have a higher risk of network partitions and message loss, and theoretical limits on consensus.

3.3 Partial Synchronous

In a *Partial Synchronous* network, there are some guarantees about message delivery and processing times, but these guarantees may not hold at all times. This model is a compromise between synchronous and asynchronous networks.

$\boxed{ASSUMPTIONS}$

This model assumes that:

- **Timing:** Nodes may not have synchronized clocks, but the network can eventually becomes stable enough for synchronously.
- **Message Delivery:** Messages sent between nodes will eventually arrive, but there may be periods of time when messages are delayed or lost.

- **Processing Time:** There exists a bound on the time taken to process a message, but this bound is unknown or does not hold at all times.

Consideration: These networks combine real-world practices with theoretical guarantees to reach consensus. It assumes that even if the messages are delayed, they will eventually be delivered.