Mini-Seminar in Foundations of Distributed Systems: Blockchain and Distributed Ledger in relation to Ethereum

Anastasiya Merkushova,
Takudzwa Togarepi, Yannick Martin

22 December 2023

Contents

1 Introduction

During the 1970s, the realization that computers would play a pivotal role in aircraft control highlighted the need for redundancy in mission-critical systems. Ensuring multiple machines maintained a consistent view and made coherent decisions became imperative. NASA's sponsorship of the Software Implemented Fault Tolerance (SIFT) project aimed to construct a resilient aircraft control system capable of withstanding component faults. This initiative led to the groundbreaking work by Lamport, notably introducing the "Byzantine Generals" problem [1], laying the groundwork for distributed consensus. Over time, distributed consensus has evolved significantly and found applications in various domains.

In recent decades, corporate giants like Google and Facebook have integrated distributed consensus into their computing infrastructure. This integration serves to replicate vital services such as Google Wallet and Facebook Credit. The emergence of Bitcoin in 2009 and subsequent cryptocurrencies marked a milestone in distributed consensus. They demonstrated the feasibility of achieving consensus in a decentralized, permissionless environment where participation is open to all.

In 2014, Ethereum was introduced through Vitalik Buterin's whitepaper [2]. Ethereum presented a platform enabling users to execute arbitrary code in the form of smart contracts. To combat denial-of-service attacks stemming from infinite loops in code, Ethereum introduced the concept of metered execution. This approach necessitates a fee, paid in Ether (Ethereum's native currency), for each operation conducted on the blockchain. With the introduction of smart contracts, Ethereum ushered in a versatile realm where operations are not confined to mere bitcoin-style value transfers. Instead, users can execute diverse business logic on-chain, courtesy of Ethereum's Turing-complete design. Currently, Ethereum stands as the most widely used blockchain platform for smart contracts.

This report will offer a concise overview of Ethereum in four main aspects: Data, Communication and High-Performance Computing.

2 Data Part

In this part we aim to look into some of the general aspects on which Ethereum is built on, that is we will look the basics of what distributed ledgers are and how they connect with Ethereum. Also, we will epxplore the structure of an Ethereum block as well as look into the main data structure(Merkle Patricia Trie) employed by Ethereum. Lastly, we would briefly discuss some of the Ethereum properties relating to how data is stored that is, is it replicated, is it consistent?

2.1 What is distributed ledger technology (DLT)?

In its simplest form Distributed Ledger Technology(DLT) refers to a database whereby there exists several identical copies of such a database which are distributed among participants and are updated in a synchronised manner by consensus of the participants [3].This database is what is referred to as a ledger.

Distributed ledger technology employs the peer-to-peer(P2P) network architecture. Also it makes use of encryption more specifically asymmetric encryption so as to authenticate the sender, to ensure the integrity of the message and makes it difficult for third parties to access the information in case they intercept it. Consensus mechanisms are employed to allow multiple participants in a network to agree on the addition of new entries to the ledger. There are two main categories of ledgers namely public and private ledgers.

2.2 What is Blockchain(Ethereum)?

Blockchain builds upon the distributed ledger technology. It is a type of distributed ledger which can fall under either public or private ledgers depending on application of the blockchain. However Ethereum is a blockchain that employs the public ledger. The term blockchain refers to blocks chained together with cryptographic hashes of the previous blocks. An Ethereum "block" is a collection of transactions and other data that is verified by the participants.

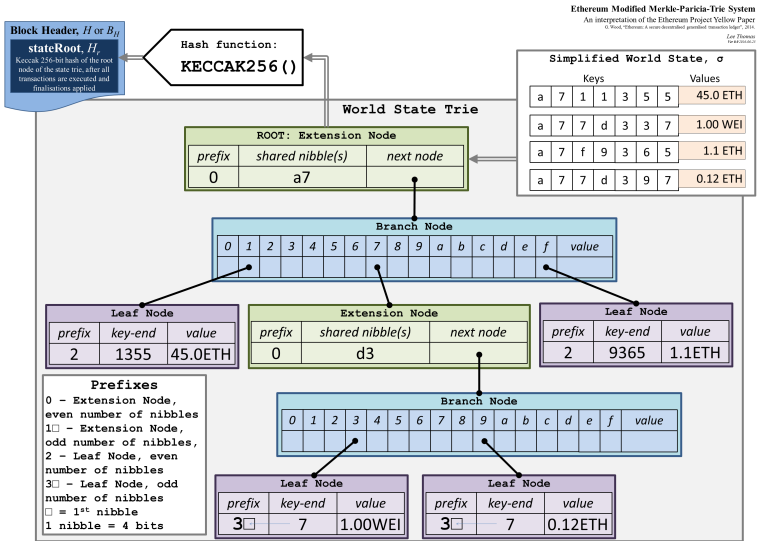
Each block is divided into two main parts namely block header and block body. The block head contains several components including *parent hash, ommer hash(uncle hash), timestamp, state root, difficulty target, nonce, mixHash* where:

- *parent hash* - This refers to a Keccak 256-bit hash of the previous(block N-1) block's header.
- *ommer hash* - This refers to the Keccak 256-bit hash of the combined

list of ommers(uncles) in a block. Blocks that are not included in the main blockchain but still considered valid are called ommers.

- *timestamp* - This is a scalar value which denotes when the block was created.
- *state root* - This refers to the Keccak 256-bit hash of the root node of the Merkel Patricia Trie(MPT). This hash summarizes the current state of the blockchain.
- *transaction root* - The transaction root refers to the Keccak 256-bit hash of the transactions in a block. It is calculated from the same Merkel Patricia Trie(MPT) as the state root.
- *receipt root* - Similarly to the transaction root, the *receipt root* is a Keccak 256-bit hash of the transaction receipts included in the block. The gas used and any logs generated are some of the information included in the *receipt root*.
- *difficulty target* - This refers to one of the parameters in the proof of work consensus algorithm. The difficulty target represents computational effort required to find a valid block hash. It is dynamically updated by the Ethereum protocol at 15 second intervals.
- *nonce* - A block header nonce is a 64-bit number that is meant to represent a counter. It is also used in the block creation process as well as proof of work consensus algorithm.
- *mixHash* - Like the nonce the *mixHash* is also a component of proof of work consensus algorithm. It is used together with the nonce to create a puzzle miners need to solve to create a valid block.

2.3 Merkle Patricia Trie(Ethereum data structure)



We have 4 main nodes in Merkle Patricia Tries namely, *Extension*, *Branch*, *Leaf* and *Empty* nodes. Data is stored in the form of Key-value pairs in the Trie. In an example of just Ether transfers the account address would be the key whilst the value is the amount. The keys are in base 16(hexadecimal form). In the diagram above if we want to lookup value 1.1ETH we start in the root extension node. And since all our keys have shared first 2 characters a7, we take a7 as our shared nibble which just is a reference of the data we're looking for. This prompts us to go to the next node which is the branch node then we look for key f. This leads us to a leaf node which contains the remaining part of the key ("9365") and the actual value 1.1ETH. The contents of the root extension node are what is Keccak 256 hashed and is known as the stateRoot.

2.4 Properties of Ethereum

- *atomicity* - Transactions in Ethereum are considered to be atomic transactions. That is operations of a transaction are executed sucessfully or none of them are.
- *consistency* - Due to the distributed nature of Ethereum data is not instantly consistent after a transaction. However we do reach eventual consistency by using merkle tries and consensus mechanisms.
- *freshness* - Similarly to the consistency in Ethereum, there do exist brief periods of data staleness although data is mostly considered fresh. For example network congestion among othe factors could temporarily affect the data freshness.

3 Communication Part

In a seminal paper by Lamport [1], the consensus problem is vividly demonstrated through an example involving divisions of the Byzantine army encamped outside an enemy city. Each division is under the command of its own general, and these generals can only communicate via messengers. Their objective is to collectively decide on a plan of action after observing the enemy. However, a challenge arises: some of these generals might be traitors aiming to disrupt the loyal generals' ability to come to an agreement.

Let's say there are n generals, and among them, one is designated as the commanding general. The commanding general seeks to propose an order, either *attack* or *retreat*, to all generals, under the following conditions:

All loyal generals must unanimously agree on the same decision. If the commanding general is loyal, all loyal generals will follow the commanding general's order. Lamport labeled this dilemma the *Byzantine Generals problem*, also commonly known as Byzantine Broadcast (BB). It's crucial to note that if the commanding general is guaranteed to be loyal, the issue becomes straightforward: the commanding general can send their order to all other generals, who can then comply, assuming they can verify the authenticity of the order. However, even the commanding general might be a traitor, proposing different orders to different generals. Consequently, the straightforward solution would lead to conflicting decisions.

Given this scenario, can the loyal generals still reach a consensus on an attack plan by communicating with each other, despite the presence and influence of corrupted generals?

The above imaginary situations serve as an analogy with a computer system in which one or more components can fail. More precisely, the problem

of distributed consensus in a distributed system is that while some nodes in the system might act in an arbitrary manner, the correctly functioning nodes still need to agree on a common value among themselves. Real-life consensus protocols typically need to repeatedly reach consensus over time, such that nodes jointly maintain an ever-growing, linearly-ordered log of transactions, called a blockchain. The primary motivation for replication for blockchain is "decentralization," meaning to ensure that responsibility for the protocol is distributed over many machines, with no one actor having significant control over its state and execution.

In a decentralized system, ensuring security without relying on a central authority indeed presents a significant challenge. Participants in such systems often don't trust each other, and establishing a reliable record of transactions becomes crucial.

The validation of transactions and prevention of fraudulent activities in a decentralized system are typically managed through *consensus mechanisms.* A consensus mechanism is an algorithmic protocol used to validate and authenticate transactions or records on a decentralized ledger. It serves as the foundational pillar, ensuring the integrity and trustworthiness of the distributed network.

Satoshi Nakamoto's, founder of Bitcoin, introduction of the proof-of-work *consensus mechanism* [4], which underlies Bitcoin's functioning. It involves proving computational effort to verify transactions and append them to the ledger. This process creates a distributed, immutable, and secure ledger of transactions. However, there isn't just one *consensus mechanism.* Over time, various alternatives have emerged, each with its unique characteristics shaping the associated network's attributes. Ethereum initially used proof-of-work but later switched to proof-of-stake with Serenity's release in September 2022, altering the network's attributes.

In this chapter, we'll delve into the foundational concepts underlying Ethereum's *consensus mechanisms*, aiming to provide an understanding of how Ethereum achieves agreement among its decentralized network participants.

3.1 Proof-of-Work Algorithm

The nonce and mixHash fields of a header prove that a sufficient amount of computation has been carried out on the block. In other words, they represent a proof of work.

Definition 1 *A proof of work is cryptographic evidence that a certain amount of computational effort has been expended.*

Producing proofs of work is done in Ethereum by using ETHash, a custom algorithm derived from the Dagger-Hashimoto algorithm [5]. We can

consider ETHash as a one-way, strong collision resistant hash function with two inputs and two outputs (in: header, nonce and out: result, mixHash).

Definition 2 *PoW verification: Given a header H, and its truncated version H_t (i.e. without the nonce and mixHash fields), $H.nonce$ and $H.mixHash$ make up a valid proof of work when the following condition holds:*

$$result \leq \frac{2^{256}}{H.difficulty} \quad and \quad mixHash = H.mixHash$$

where $(result, mixHash) = ETHash(H_t, H.nonce)$.

The specific process of finding values for the nonce and mixHash fields of a header satisfying this condition is called *mining*.

3.2 Proof-of-Stake Algorithm

Nakamoto's consensus first showed how to achieve consensus in a permissionless, decentralized environment with open enrollment. This was amazing, butthe community soon realized that Proof-of-Work (PoW) based approaches are undesirable, partly due to the enormous energy waste induced by the protocol. According to the bitcoin energy consumption tracker at Digiconomist, as of June 2019 Bitcoin consumed 66.7 terawatt-hours per year. That's comparable to the total energy consumption of the Czech Republic, a country of 10.6 million people!

To avoid this drawbacks, many blockchain projects put their hope on a new paradigm called Proof-of-Stake (PoS). In PoW-based consensus, nodes have voting power proportional to their mining power, and consistency and liveness are guaranteed if the (super-)majority of the mining power is honest. Analogously, in PoS, nodes have voting power proportional to the amount of cryptocurrency they hold, and we hope to guarantee consistency and liveness as long as, roughly speaking, the (super-)majority of the stake participating in consensus behaves honestly.

3.2.1 Casper: Ethereum's Proof-of-Stake Algorithm

Buterin et al. introduced Gasper [6], an innovative protocol amalgamating Casper FFG (Friendly Finality Gadget) and LMD GHOST (Latest Message Driven Greediest Heaviest Observed SubTree) to construct Ethereum's "beacon chain" for its sharding design. Interestingly, this construction can also function independently as a solo blockchain without sharding.

Casper FFG serves as a "finality gadget," marking specific blocks in the blockchain as finalized, enabling participants with partial information to have complete confidence that these blocks are part of the official chain. It operates agnostically atop various blockchain protocols, be it proof-of-work or proof-of-stake. Meanwhile, LMD GHOST acts as a fork-choice rule,

wherein validators (participants) endorse blocks, akin to a voting system, signaling their support for these blocks.

Gasper represents a complete proof-of-stake protocol, serving as an abstracted model of the proposed Ethereum implementation.

4 High Performance Computing Part

As we have seen in the lecture, the goals of High Performance Computing are maximizing performance, resiliency and sustainability, and minimizing the energy consumption. One of the challenges presented in the lecture to achieving these goals is scalability. Especially with blockchain, scalability is a very important topic as more users using blockchain, and still every transaction has to be processed by every node. As a result, the costs to add a block are getting increasingly higher, which jeopardises the above-mentioned goals of High Performance Computing, especially performance[7].

4.1 Gas

Gas, in Ethereum, is the unit that measures how much computational effort is needed to execute a transaction. In other words, how much energy is consumed during the transaction. The fee you have to pay is the amount of gas used for the transaction multiplied by the cost per gas. Gas fees exists on the one hand for network security to prevent bad actors from spamming the network but on the other hand and this is the more important point to pay the computational costs to integrate the transaction into the blockchain. The creators of Ethereum obviously want to keep the fees low so that more users use the Ethereum blockchain. And to reduce these energy costs and therefore also thee fees, the blockchain has to be scaled[8].

4.2 Scalability

As already mentioned above, scaling is not only important to reduce energy consumption but also a measure to improve performance. For this the aim is to increase the number of transactions per seconds so that Ethereum can be used in everyday life as a payment method, for example. Improving the performance is important since each node must provide the computational effort to validate each transaction. The Ethereum blockchain demands therefore high computing power, high bandwidth internet connection and also a lot of storage space. However, scaling is not easy, especially because Vitalik Buterin, the co-founder of Ethereum, has recognised the blockchain trilemma between scalability, security and decentralization. This means that you have to find a compromise between these three factors. This becomes even more difficult when you consider that decentralisation is a key property of the blockchain that you want by design and also security is an essential property because Ethereum is also used as a currency.[7]

Fortunately, there are still some different types of blockchain scalability solutions. These can be divided into 2 categories. They are called on-chain and off-chain scaling solutions. On-chain solutions work with the original blockchain, while off-chain solutions look for solutions outside the

blockchain. The change from proof of work to proof of stake for example, which was described in the Communication Part above, is an on-chain scaling solution. For a long time it looked like sharding is the method that will scale the blockchain. Sharding is also an on-chain scaling approach which divides the database into "shards". After that, not all validators but only a subset of them are responsible for a shard. This allows you to execute transactions more efficiently. However, off-chain scaling methods in particular Layer 2 Scaling have been developed, so much so that this is the favoured method for Ethereum, which we will discuss in more detail here.[9]

4.3 Layer 2 Scaling

Off chain scaling methods form a category where transactions are already executed outside of the blockchain. Layer 2 scaling works as the name suggests with a second layer, which is a separate blockchain. One of the mechanisms that use Layer 2 are rollups. The basic idea of rollups is executing the transactions outside of layer 1 namely on layer 2 like described above. The transactions are then bulked together and only minimal summary transaction data is posted to the original blockchain. This method scales the blockchain and keeps the security guarantees of Ethereum. There exist also two different categories of roll ups namely optimistic rollups and Zero-knowledge rollups. The main difference between these categories is the validation of transaction. Optimistic rollups take an "optimistic" approach, assuming that all transactions are valid unless proven otherwise. Zero-knowledge rollups use zero-knowledge proofs to validate transactions. This validation happens off-chain, and only the proof is posted on-chain.[9] [10]

5 Conclusion

In conclusion, Ethereum is one of the most popular blockchain in the world, making it a go-to solution for reaching agreement on a large scale. The blocks in the blockchain contains the transactions and are immutable. Merkle Patricia Trie is the data structure used to store the world state effectively. Ethereum recently adopted the proof-of-stake protocol, a more energy-efficient way of making decisions compared to proof-of-work. However, because more and more people want to use Ethereum, it needs to figure out how to handle a higher number of transactions at once. Therefore finding good scalability solutions are very import, so that Ethereum can be used on a daily basis. In summary Ethereum by itself is one of the most popular answer on the questions on how to reach agreement on the large scale nowadays.

References

[1] Leslie Lamport, Robert Shostak, and Marshall Pease. *The Byzantine Generals Problem*, page 203–226. Association for Computing Machinery, New York, NY, USA, 2019.

[2] Vitalik Buterin et al. Ethereum white paper.

[3] José Luis Romero Ugarte. Distributed ledger technology (dlt): introduction. *Banco de Espana Article*, 19:18, 2018.

[4] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[5] T. Dryja. S. Hashimoto. I/o bound proof of work. *Decentralized business review*, 2014.

[6] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper. *arXiv preprint arXiv:2003.03052*, 2020.

[7] S. Mustapha A. Hafid, A.Hafid. Scaling blockchains: A comprehensive survey. *IEEE Access*, 2020.

[8] Ethereum.org. Gas and Fees. `https://ethereum.org/en/developers/docs/gas/#priority-fee`. Online; accessed 7 December 2023.

[9] Alessia Renzoni. *Assessing the Impact of Rollups Scaling Solutions on Ethereum's layer-2 Ecosystem*. PhD thesis, 2022.

[10] Ethereum.org. Scaling. `https://ethereum.org/en/developers/docs/scaling/`. Online; accessed 7 December 2023.