BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

SECURITY AND PERFORMANCE ANALYSIS OF AVALANCHE DISTRIBUTED CONSENSUS PROTOCOL

ANALÝZA BEZPEČNOSTI A VÝKONU DISTRIBUOVANÉHO KONSEZUÁLNÍHO PROTOKOLU AVALANCHE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **FILIP SAPÁK**
AUTOR PRÁCE

SUPERVISOR **Ing. MARTIN PEREŠÍNI**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Intelligent Systems (DITS) Academic year 2021/2022

Bachelor's Thesis Specification

Student:	**Sapák Filip**
Programme:	Information Technology
Title:	**Security and Performance Analysis of Avalanche Distributed Consensus Protocol**
Category:	Security

Assignment:

1. Get familiar with existing categories of distributed consensus protocols, especially DAG-based protocols. Make a theoretical comparison of these protocols in terms of throughput, storage, scalability, security, failure tolerance, liveliness, safety, and finality.
2. Analyze DAG-based consensus protocol Avalanche.
3. Study existing simulation frameworks for distributed consensus protocols.
4. Implement simulation of Avalanche protocol and perform a set of experiments to evaluate its performance, storage overhead, and security.
5. Discuss achieved results and compare them with the information from literature (white-paper).

Recommended literature:

- Homoliak, Ivan, et al. "The security reference architecture for blockchains: Towards a standardized model for studying vulnerabilities, threats, and defenses." *arXiv preprint arXiv:1910.09775* (2019).
- Perešíni, Martin, et al. "DAG-Oriented Protocols PHANTOM and GHOSTDAG under Incentive Attack via Transaction Selection Strategy." *arXiv preprint arXiv:2109.01102* (2021).
- Aoki, Y., Otsuki, K., Kaneko, T., Banno, R. and Shudo, K., 2019. SimBlock: a blockchain network simulator. arXiv preprint arXiv:1901.09777.
- Zhang, Ren, and Bart Preneel. "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security." *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019.
- https://www.avalabs.org/whitepapers

Requirements for the first semester:

- Items 1 and 2.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Perešíni Martin, Ing.**
Consultant:	Homoliak Ivan, Ing., Ph.D., UITS FIT VUT
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	November 1, 2021
Submission deadline:	May 11, 2022
Approval date:	November 3, 2021

Abstract

In this thesis, we discuss the analysis of the "Proof-of-Stake" (PoS) distributed consensus protocol Avalanche. We describe blockchain, consensus algorithms and the simulation tools designed to simulate the blockchain. After getting acquainted with all the necessary facts, the thesis proceeds with the implementation of a simulator of the consensus algorithm implemented in the Avalanche protocol, which mimics the functioning of the Avalanche consensus. Experiments with simulator were conducted with various parameters and helped us to understand properties of Avalanche.

Abstrakt

V tejto práci sme sa zaoberali analýzou „Proof-of-Stake“ (PoS) distribuovaného konsenzuálneho protokolu Avalanche, problematikou blockchainu, konsenzuálnych algoritmov a simulačnými nástrojmi určenými na simuláciu blockchainu. Po oboznámení sa so všetkými potrebnými skutočnosťami, práca pokračuje implementáciou simulátora konsenzuálneho algoritmu implementovaného v protokole Avalanche, ktorý napodobňuje fungovanie Avalanche konsenzu. Experimenty so simulátorom boli vykonané s rôznymi parametrami a pomohli nám pochopiť vlastnosti Avalanche.

Keywords

Blockchain, Proof-of-Stake, Node, Block, Transaction, Consensus, Avalanche

Klíčová slova

Blockchain, Proof-of-Stake, Uzol, Blok, Transakcia, Konsenzus, Avalanche

Reference

SAPÁK, Filip. *Security and Performance Analysis of Avalanche Distributed Consensus Protocol*. Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Martin Perešíni

Rozšířený abstrakt

Táto práca sa zaoberá analýzou „Proof-of-Stake“ (PoS) distribuovaného konsenzuálneho protokolu Avalanche a implementáciou simulátora konsenzuálneho algoritmu implementovaného v protokole Avalanche. V práci sa zaoberáme problematikou „blockchainu“, jeho štruktúrou, architektúrou a všetkými komponentami, ktoré sú dôležité pre jeho pochopenie. Ďalej sú spomenuté aj iné konsenzuálne modely, ako napríklad „Proof-of-Work“ (PoW) alebo „Practical byzantine fault tolerance“ (PBFT), ktoré sme navzájom porovnali. Popísané sú aj rôzne blockchain simulátory a ich porovnanie kompatibility s konsenzuálnym modelom PoS, ktorý práve používa aj sieť Avalanche, a je teda veľmi dôležitým parametrom pri výbere simulačného nástroja. Sieť Avalanche je momentálne veľmi populárna medzi vývojármi vďaka možnosti spustenia Web3 aplikácií na vlastných „subnetworkoch“ - výhodou sú rýchlejšie transakcie a nižšie poplatky za transakcie pre užívateľov konkrétnej Web3 aplikácie ktorá beží na „subnete“. Pre podporu rôznych decentralizovaných finančných nástrojov a blockchainových hier je sieť Avalanche veľmi populárna aj medzi užívateľmi. Motivácia analyzovať protokol Avalanche vznikla faktom, že je to relatívne nový a aktuálne jeden z najlepších PoS protokolov ktorý by mal byť podľa tvrdení nástupcom Nakamoto konsenzu. Chceli sme sa zamerať hlavne na konsenzuálny algoritmus, jeho vlastnosti a nastaviteľné parametre v rôznych situáciách, ktoré môžu v sieti Avalanche vzniknúť. Konsenzuálny algoritmus Avalanche sa skladá z 3 hlavných algoritmov: *Slush*, *Snowflake* a *Snowball*, ktoré prebiehajú v kolách a ich funkcionalita je implementovaná v navrhnutom simulátore. *Slush* je základom rodiny týchto algoritmov, nie je odolný voči Byzantským uzlom, no slúži ako základ pre na algoritmy *Snowflake* a *Snowball*, ktoré na ňom ďalej stavajú a tento problém odstraňujú. Zamerali sme sa hlavne na možnosť simulovať priebeh týchto algoritmov ku dosiahnutiu konsenzu vo veľkých sieťach ktoré operujú nad desiatkami tísic uzlov. Výsledky implementácie sa dajú považovať za úspech, pretože navrhnutý simulátor zvláda simulovať veľké siete pomerne efektívne a v krátkom čase vďaka spôsobu výpočtu výsledkov v jednotlivých kolách konsenzuálneho algoritmu Avalanche. Vďaka implementovanému simulátoru sme mohli uskutočniť experimenty týkajúce sa vplyvu parametrov konsenzuálneho algoritmu na rýchlosť dosiahnutia konsenzu a jeho náchylnosť na „škodlivé“ alebo „zákerné“ uzly v sieti, ktoré sa snažia sieť zvrhnúť. Výsledky experimentov potom odhalili vplyv rôznych parametrov na sieť Avalanche a pomohli nám priblížiť vlastnosti distribuovaného konsenzuálneho protokolu Avalanche.

Security and Performance Analysis of Avalanche Distributed Consensus Protocol

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Martin Perešíni. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Filip Sapák
May 11,2022

Acknowledgements

I would like to thank my supervisor Martin Perešíni for his supervision of this work and also my family and closest friends for the emotional and physical support they created in my academic path.

Contents

Chapter 1

Introduction

The term blockchain is being used more and more nowadays, mostly because it is associated with cryptocurrencies that offer high returns and attract investors. Most consider blockchain a new technology even though the early stages of development of blockchain-like protocols date back from 1980's to 1990's. Blockchain is a shared, immutable, distributed ledger system that secures data integrity, transparency and decentralization. The first real decentralized blockchain protocol was announced on October 31, 2008 when Satoshi Nakamoto, the anonymous founder of Bitcoin, sent the Bitcoin whitepaper to the Cypherpunk email list. Today, Bitcoin is the world's largest cryptocurrency in terms of capitalization and is widely used and popular for it's security and decentralization among investment companies and investors. Although the application of blockchain nowadays is mostly found in financial sector, new types of applications, like in the governance systems, medical systems, virtual reality or art trade are emerging every day.

Blockchain is spread across nodes whose job is to verify transactions on the network and help keep the network safe - using the community to help verify if transactions are genuine. The rules by which nodes (a.k.a.,participants,devices) process transactions are declared by consensus mechanism implemented on particular blockchain. Anyone can submit information to be stored onto a blockchain but all nodes must agree on the state of the network - achieving consensus, following the predefined rules or protocol.

Different blockchain protocols use different consensus models. The best known is the concept of Proof-of-Work (PoW) which is based on performing computationally intensive operations. The problem with this concept is its energy inefficiency. Bitcoin, using the PoW model, can consume more energy per year than the Netherlands [15]. Another problem is the low number of transactions that can be achieved with this type of consensus. Compared to centralized systems with a throughput of approximately 50,000 transactions per second, systems based on PoW have lower performance (5 to 10 transactions per second). Protocols based on the Proof-of-Stake (PoS) model are a potential solution. Instead of challenging operations, different approaches are used to select the participants who are responsible for the correct functioning of the network. Not only does this approach save electricity, but also increases the overall throughput of the blockchain. However, PoS protocols also have their vulnerabilities. For this reason, they need to be studied and compared thoroughly. This is precisely the purpose of of this thesis, with the goal of analyzing Avalanche consensus in terms of its performance and security using simulation.

Document structure

Chapter 2 explains the blockchain to the ordinary user and informs about blockchain architecture (2.2) with all its layers and blockchain structure (2.3) with all its components - blocks, transactions, smart contracts, virtual machines, nodes and incentive mechanisms.

Chapter 3 describes consensus models Proof-of-Work (3.1), Proof-of-Stake (3.2) and Practical byzantine fault tolerance (3.3).

Chapter 4 describes the Avalanche consensus protocol in detail with its main part being consensus algorithm (4.2).

For the simulation needs of Avalanche consensus, we looked at some of the existing simulation tools in chapter 5. In addition, we created a table (5.1) that summarizes the support of Proof-of-Stake protocol simulation by each of the mentioned tool.

Next chapter (6) is dedicated to implementation of the proposed Avalanche consensus simulator and evaluation of experiments done with various Avalanche consensus parameters. Lastly, we discuss achieved results in chapter 7.

Chapter 2

Blockchain technologies

To further understand any consensus protocol, we need to have a basic understanding of blockchain. The following chapter will explain the working principle of this technology.

2.1 Definition

Blockchains are tamper evident and tamper resistant digital ledgers implemented in a distributed manner (i.e., without a central repository) in which data records (i.e., blocks) are linked using a cryptographic hash function, and each new block must be agreed upon by participants (a.k.a., nodes) usually without a central authority (i.e., a bank, company or government). At their basic level, they enable a community of users to record transactions in a shared ledger within that community, such that under normal operation of the blockchain network no transaction can be changed once published [16].

Features

- **Immutability:** No participant can change or tamper with a transaction once the transaction in published. In order to do it, the majority of the nodes need to agree upon that.
- **Decentralization:** The transactions are not processed through a central entity (i.e., central server), but through the participants (a.k.a., nodes) in parts of the network, which function based on a set of rules that they must follow.
- **Security:** Blockchain uses cryptography to sign data in order to prove that a transaction was approved by the sender. Each running node of network has a copy of all the information contained in that blockchain. Thus, the data is protected even if a certain amount of the nodes are compromised by an attack or network issues.
- **Transparency:** All the actions on a public blockchain are viewable and accessible for everyone.
- **User anonymity:** The user identities are hidden, only the digital addresses and interactions within the network are visible.

Types

Based on permission model, which determines who can join the consensus protocol and publish blocks, we can distinguish the two main blockchain network types:

- **Permissionless** blockchain networks are open to anyone,participants can join and publish blocks without needing permission from any authority. As a result, malicious users can join and attempt to publish blocks in a way that subverts the system. To prevent this, permissionless blockchain networks usually require a verification of the transactions through consensus methods such as Proof-of-Work, Proof-of-Stake and so on, which require users to expend or maintain resources when attempting to publish blocks. These consensus systems promote non-malicious behavior through rewarding the publishers of protocol-conforming blocks.
- **Permissioned** blockchain networks are under the control of a centralized or federated authority, which decides which participants can publish blocks and maintain the network. The authority has such control over the network, it also has a power to restrict who can issue transactions and to restrict read access. Like in permissionless networks, permissioned blockchain networks use consensus models for publishing blocks, but because of the establishment of participants identities and the level of trust between them, these consensus methods often do not require the expense or maintenance of resources [10].

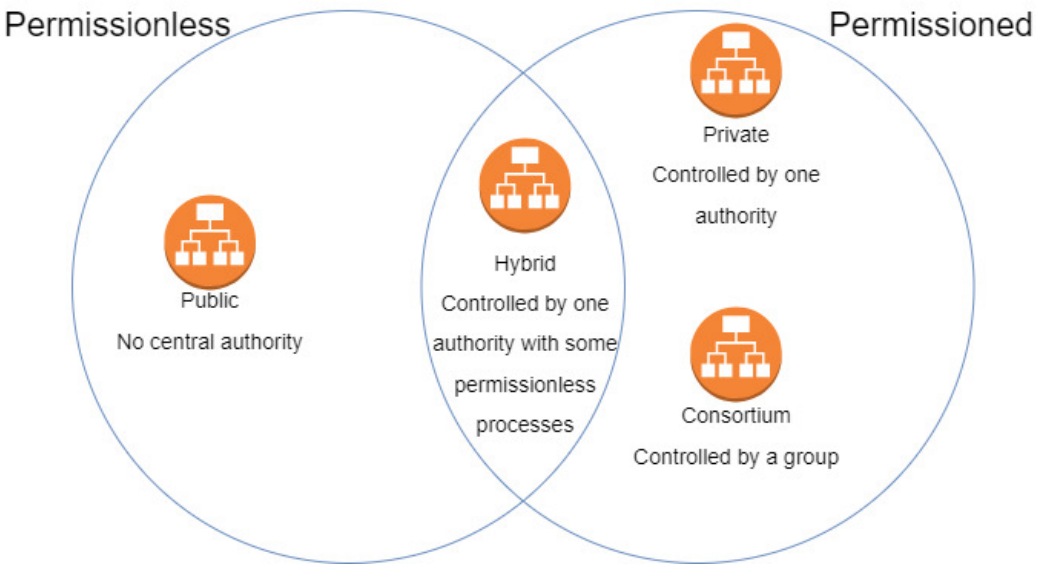

Figure 2.1: Blockchain types [8].

2.2 Architecture

This section discusses components involved in the blockchain structure.

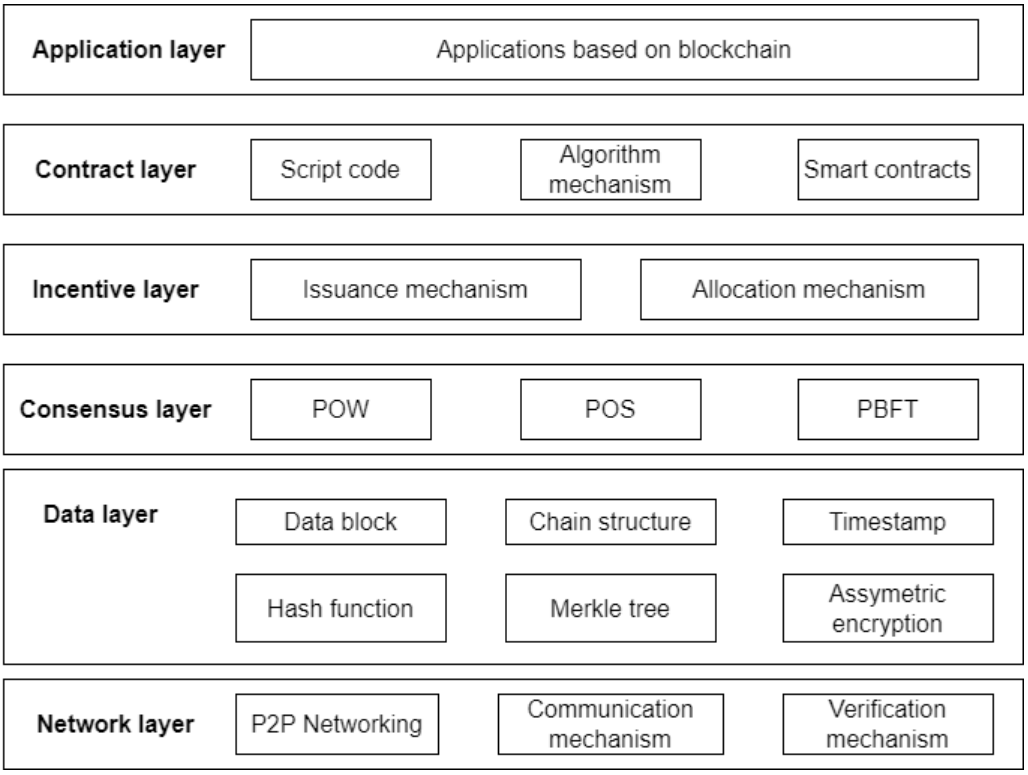

Figure 2.2: Technology architecture of blockchain system [17]

We can divide blockchain system architecture into layers:

- **Application layer** encapsulates various application scenarios and cases of the blockchain.
- **Contract layer** contains various scripts, algorithms and smart contracts, which is the basis of blockchain programmable features.
- **Consensus layer** mainly encapsulates all kinds of consensus mechanisms while the incentive layer integrates economic factors into blockchain technology.
- **Network layer** includes a peer to peer distributed networking mechanism, a data propagation and verification mechanism.
- **Data layer** encapsulates the underlying data block, basic data and algorithms related to data encryption and time stamping.

2.3 Structure of blockchains

Blocks

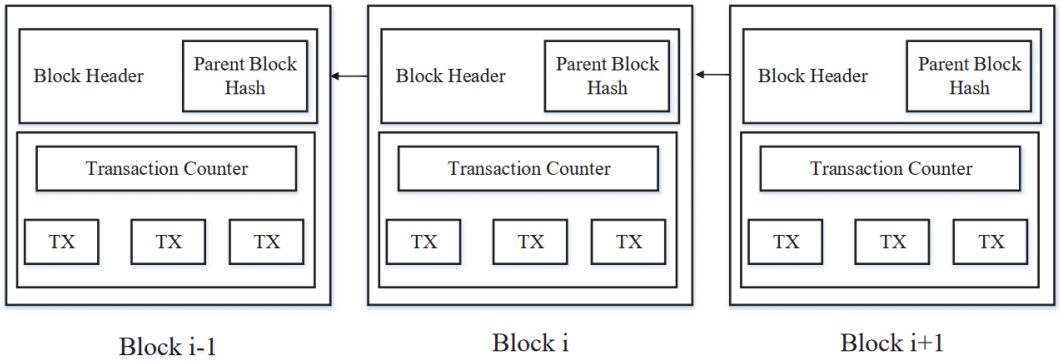

Figure 2.3: Chain of blocks [18]

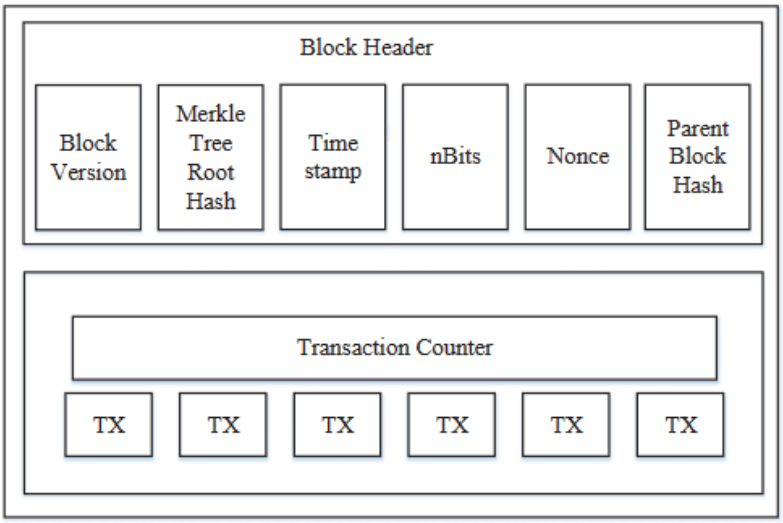

Figure 2.4: Simplistic structure of a block [18]

The blockchain data structure is represented by an append-only ordered list of blocks which consist of transaction records as shown in Figure 2.3. A block contains individual transactions and transaction counter. The maximum number of transactions that a block can contain depends on the block size and the size of each transaction. A block also holds block header which consists of:

- **Block version:** refers to the blockchain version that the particular block is using, which rules must be used to validate the particular block

- **Merkle tree root hash:** the hash value that represents all the transactions that are recorded in particular block, it is found by repeatedly hashing pair of transactions together until a single hash value remains
- **Timestamp:** the time of block foundation, used as a parameter to verify the authenticity of a particular block
- **nBits:** represents the target value/threshold for a new valid block
- **Nonce:** 32-bit number that has to be equal or lower than given target value. Once node finds the correct value, it broadcasts the block to other nodes that must validate it in order to append the particular block to blockchain.
- **Parent block hash:** a 256-bit hash value that stores pointer to the previous block, the first block in the blockchain is called the Genesis Block and has no previous block hash value

As a result of each block containing the hash digest of the previous block's header, they form an imaginary chain, hence the name blockchain. If a hash of already published blocks is changed, it implies a change in a particular published block's data. This change would cause change in all the later published blocks' hashes, which makes it easy to trace and reject that particular block which caused this alteration.

Transactions

Transactions are encrypted in a form called Merkle tree, a data structure where all transactions in block are hashed and combined until a single root hash is found that represents the entire structure, which is then stored in Merkle tree root hash in the block header of particular block.

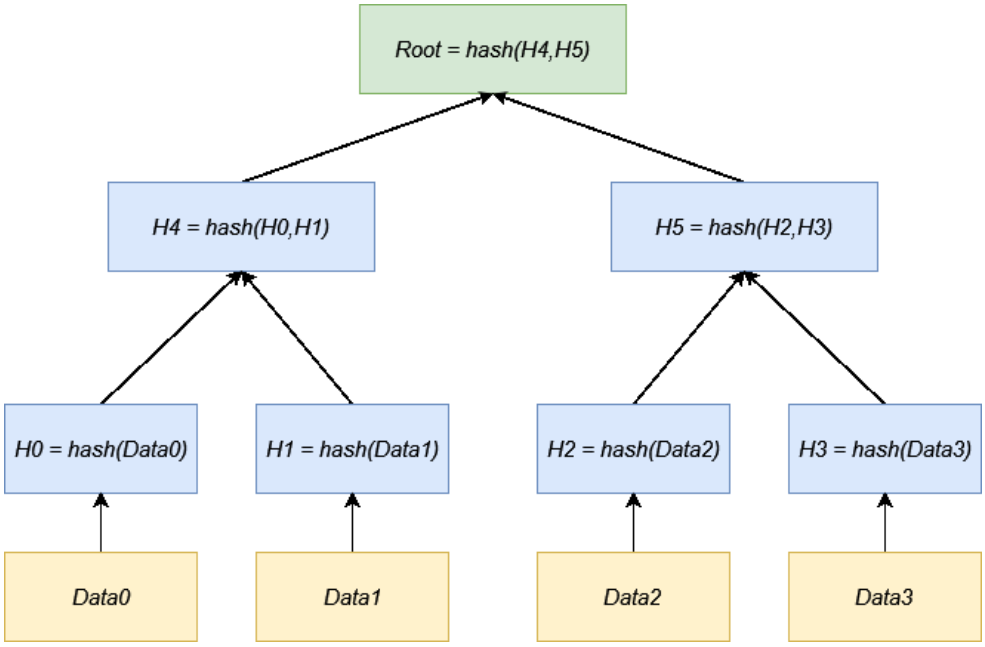

Figure 2.5: Merkle tree [16]

Hashing is a method of calculating a relatively unique fixed size output from any input. The encryption in the tree is done by bottom-up approach, the nodes at the bottom represent each transactions data which will be used as input to hash function. The transaction information can be different for every blockchain information, it may contain sender's address, sender's public key, a digital signature, transaction inputs and transaction outputs. Next level nodes hold the hash result from previous level which will then get paired and used as input to hash function to form a new single encrypted node. This process continues until all the nodes are combined into the root node at the top of the tree - the Merkle tree root hash. The final block will then combine the Merkle tree root hash with previous block encrypted header hash [7, 16].

Smart contracts

A smart contract in blockchain is a special type of transaction, a program code which is stored, verified and then run on a blockchain to facilitate, execute and enforce the terms of an agreement. Smart contracts promise low transaction fees by automatically executing the terms of an agreement once the specified conditions are met as opposed to traditional systems that require a trusted third party to enforce and execute the terms of an agreement.

Once the contract is deployed into the blockchain, the contract code cannot be changed. To run a contract, users can simply send a transaction to the contract's address which will then be executed by every consensus node in the network to reach a consensus on its output and update the contract's state. The contract can, based on the transaction it receives:

- store money into its account balance
- read or write to its private storage
- send or receive messages or money from users or other contracts
- create new contracts

It is often used as part of larger applications. The term smart contracts was first used in 1997 well-known cryptographer Nick Szabo. Protocols which work with its functionality are for example Ethereum and Cardano [1].

Virtual Machine

A Virtual Machine (a.k.a.,VM) defines the application-level logic of a blockchain. In technical terms, it specifies the blockchain's state, state transition function, transactions, and the API through which users can interact with the blockchain and allows smart contracts from multiple sources to interact with one another [3].

Nodes

In general, every participant in a blockchain network is a node. Most blockchains involve three native types of parties that according to their roles and actions can be organized into a hierarchy.

(1) Consensus nodes can read blockchain and write to blockchain by appending new transactions and prevent malicious behaviors by not appending invalid transactions or ignoring an incorrect chain. They can also validate the blockchain and check the other consensus nodes actions correctness.

(2) Validating nodes read blockchain, validate it, and propagate transactions. They can detect malicious behavior sice they hold copies of the entire blockchain, but unlike consensus nodes, validating nodes cannot prevent malicious behaviors as they cannot write to the blockchain.

(3) Lightweight nodes have limited information about blockchain and they rely on consensus nodes and validating nodes. Lightweight nodes can detect only a limited set of attacks as they can read only fragments of the blockchain and validate only a small number of transactions that concern them [6].

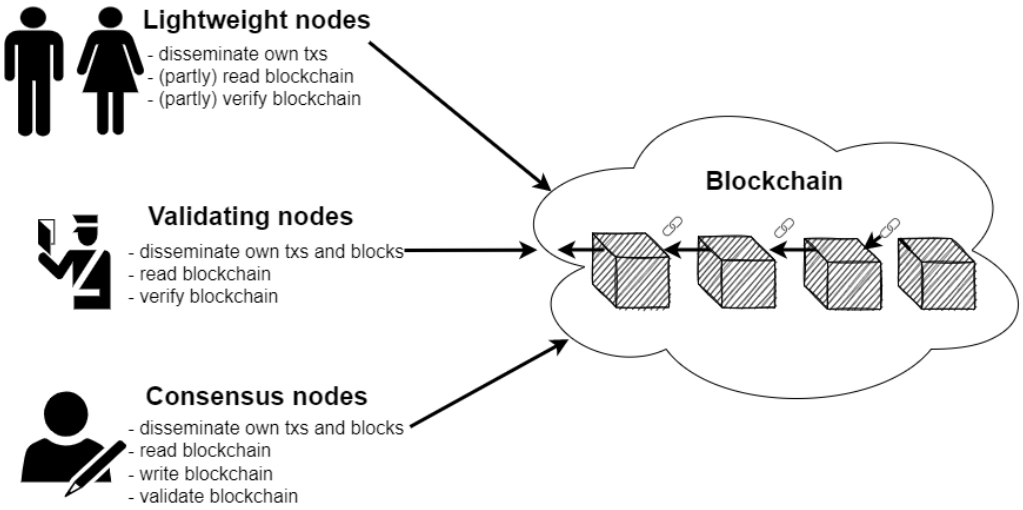

Figure 2.6: Interaction of nodes with the blockchain [6].

Incentive mechanisms

In public chains, it is of great importance to encourage nodes to participate in the maintenance of the blockchain through incentive mechanisms to prevent tampering with the ledger. It is done so by rewarding nodes that follow the rules with different forms of tokens or cryptocurrencies and penalizing nodes that do not. In the original Bitcoin blockchain, participants provide their calculation and accounting services through ”mining” to create new blocks, thus jointly maintaining the continuation of the blockchain. The winning miners receive Bitcoin rewards, some of which come from mining and partly from commissions in the transactions [17].

Chapter 3

Consensus models

One of the aspects blockchain technology must address is determining which user publishes the next block and how to solve situations when one or more users want to publish the block at the same time. There are generally many publishing nodes competing to publish the next block at the same time. The reason for this competition between nodes is usually financial gain, users are trying to win cryptocurrency and/or transaction fees without caring about the well-being of the other publishing nodes or even the network itself. To resolve conflicts between nodes and make it work, blockchain uses consensus models to enable a group of mutually distrusting users to work together [16].

There are two types of consensus protocols, the ones using synchronous assumption found in traditional consensus algorithms and the ones that are non-deterministic that are used in most blockchain networks. In traditional consensus algorithms as opposed to consensus algorithms in blockchain networks, each node must be aware of and communicate with every other node in the network. Therefore, these algorithms often scale poorly. Scalability in addition to decentralization and security, is one of the blockchain's most crucial aspects. Blockchains are often forced to make trade-offs between these three aspects that prevent them achieving them all. Various blockchain consensus protocols try to solve this trilemma differently.

3.1 Proof of Work

Proof of Work is designed for permissionless public distributed ledger and consumes computational resources of the system, it assumes that the nodes will use computational effort for the chance to decide the next block where the new transactions will be stored. This process is called mining where nodes are supposed to solve the cryptographic puzzle of finding a random integer that leads to hashes with a specified number of leading zeroes. The miners would change the nonce in block header frequently to get different hash values.

In PoW, the consensus requires that the calculated value must be equal to or smaller than a certain given value. The solution is the proof they have performed work. Once node finds the correct value, it broadcasts the block with a valid nonce to full nodes that must validate it in order to append the particular block to their own blockchains and then forward the transaction to their peers. This process is repeated until the transaction has propagated throughout the network. It is designed such that solving the puzzle is difficult but checking that a solution is valid is easy as only a single hash needs to be done to check to see if it solves the puzzle. The target value may be modified over time to adjust

the difficulty to influence how often blocks are being published. The publishing nodes (i.e. miners) attempt to solve this computationally difficult puzzle to claim a reward of some sort usually in the form of a cryptocurrency offered by the blockchain network. The prospect of being rewarded for extending and maintaining the blockchain is referred to as a reward system or incentive model.

One of the prime examples of blockchain using PoW is Bitcoin which adjusts the puzzle difficulty every 2016 blocks by increasing or decreasing the number of leading zeros required therefore making publication rate around once every ten minutes. With increasing number of leading zeros the puzzle becomes more difficult to solve as there are fewer possible solutions. Correspondingly, with decreasing number of zeros required the the puzzle becomes less difficult as there are more possible solutions. This adjustment is made to maintain the computational difficulty of the puzzle and therefore maintain the core security mechanism of the Bitcoin network as it ensures no entity can take over block production. Available computing power increases over time, as does the number of publishing nodes, so the puzzle difficulty is generally increasing. Due to the significant resource consumption of some proof of work blockchain networks, there is a move to add publishing nodes to areas where there is a surplus supply of cheap electricity.

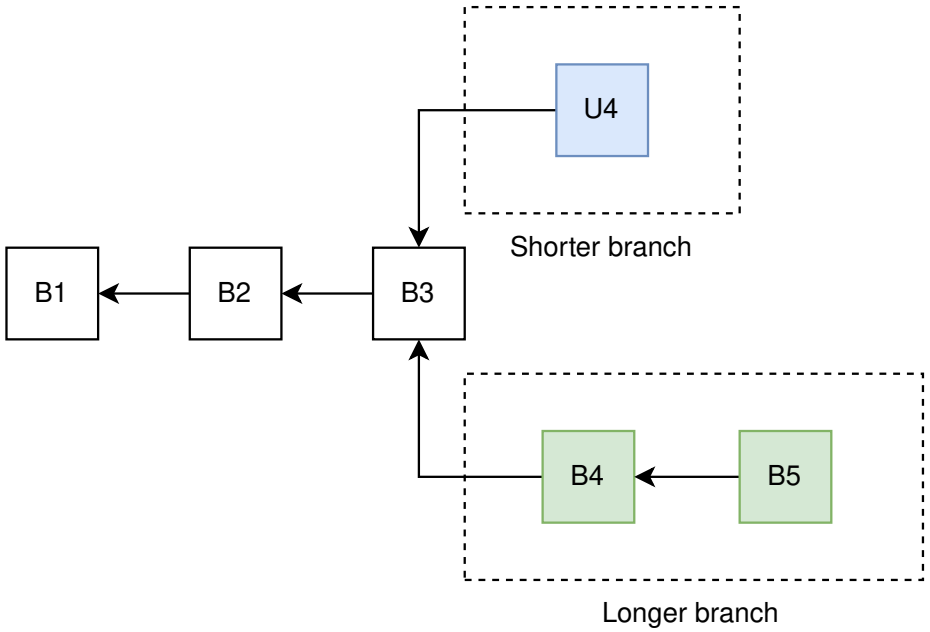

Figure 3.1: Blockchain branches [18]

For many PoW based blockchain networks, publishing nodes tend to organize themselves into „pools“ where they work together to solve the puzzles and then split the reward. Dividing up the work amongst many more machines yields much better results, as well as more consistent rewards in a proof of work model.

The use of a computationally difficult puzzle helps to combat the “Sybil Attack” – a computer security attack (not limited to blockchain networks) where an attacker can create many nodes (i.e., creating multiple identities) to gain influence and exert control. The proof of work model combats this by having the focus of network influence being the amount of computational power (hardware, which costs money) mixed with a lottery system (the most

hardware increases likelihood but does not guarantee it) versus in network identities (which are generally cost-less to create) [16, 18].

In the decentralized PoW networks, valid blocks might be generated simultaneously when multiple nodes find the suitable nonce nearly at the same time which may be create branches as shown in Figure 3.1. However, it is highly unlikely that two competing forks will generate next block simultaneously. In PoW protocol, a chain that becomes longer is judged as the authentic one. Figure 3.1 shows fork created by simultaneously validated blocks U4 and B4. Miners keep mining their blocks until a longer branch is found. B4,B5 forms a longer chain, so the miners on U4 would switch to the longer branch [18].

3.2 Proof of Stake

Proof of stake reduces the amount of computational work needed to verify blocks and transactions with the idea that the more stake a user has invested into the system, the more likely they will want the system to succeed, and the less likely they will want to subvert it. The owners offer their coins (i.e. cryptocurrency) as collateral for the chance to validate blocks. Coin owners with staked coins become validators. Once staked, the cryptocurrency is generally no longer able to be spent. PoS blockchain networks mostly use the amount of stake a user has as a determining factor for publishing new blocks - the likelihood of a blockchain network user publishing a new block is tied to the ratio of their stake to the overall blockchain network amount of staked cryptocurrency - users with more stake are more likely to publish new blocks.

Most PoS blockchain networks are designed such that all the cryptocurrency is already distributed among users rather than new cryptocurrency being generated over time. The reward for block publication is then usually the earning of user provided transaction fees.

The approach for how the blockchain network uses the stake to choose the validator for next block can vary depending on the project design:

- **Random selection** - choosing next block publisher based on their ratio of stake to the overall amount of cryptocurrency staked.
- **Multi-round voting** - selecting several staked users to each create a proposed block which will be voted on by all staked users. This method allows all staked users to have a voice in the block selection process.
- **Coin aging systems** - staked cryptocurrency has an *age* property. After a certain amount of time the stake owning user can be selected to publish the next block resetting the staked cryptocurrency *age* property. This prevents users with more stake to dominate the system since there is a cooldown timer.
- **Delegate systems** - users vote for nodes to become publishing nodes. Blockchain network users' voting power is tied to the amount of their staked coins - the lager the stake, the more weight the vote has. Nodes which recieve the most votes become publishing nodes (representing their voters) and can validate and publish blocks. Users can try to remove established publishing nodes by voting against them. The voting occurs continuously to promote competitiveness and incentivize nodes to not act maliciously. Users can also vote on delegates who participate in the governance of the blockchain and propose changes, which will be voted on by users [16].

3.3 Practical byzantine fault tolerance

Comparing to the Proof based consensus such as PoW, where the security threshold is 51 percent, i.e., absolute secure transaction can be achieved if the malicious user occupies no more than half of the overall resource, PBFT requires the number of malicious users under 33 percent of total participants to ensure the system immune from the malicious attacks.

In PBFT, nodes in the system share messages among each other to commit a block to the chain. Malicious nodes, in this case, may broadcast tampered blocks, as a result, the block which is considered valid by a most number of nodes is considered valid by the entire network. The process of voting can be divided into three phases: *pre-prepare, prepare, commit*. Before a voting round begins, transactions are broadcasted among nodes so that all nodes have the same transactions in their pool. After a sufficient number of transactions is in the pool, nodes start a new round. A proposer is chosen in a round-robin fashion and sends a *pre-prepare* message which contains a proposed block and each node enters *pre-prepared* state. Each node sends a *prepare* message if they agree upon the proposed block. After a certain amount of such messages, nodes change state to *prepared*. Prepared nodes send *commit* messages to each other, upon a certain amount of commits, nodes move to *commit* state and add the block to the chain. After adding the block they move to the *final commited* state and the process can be repeated.

Given the fact that each node has to query other nodes, traditional single layer PBFT based blockchains are difficult to scale up.

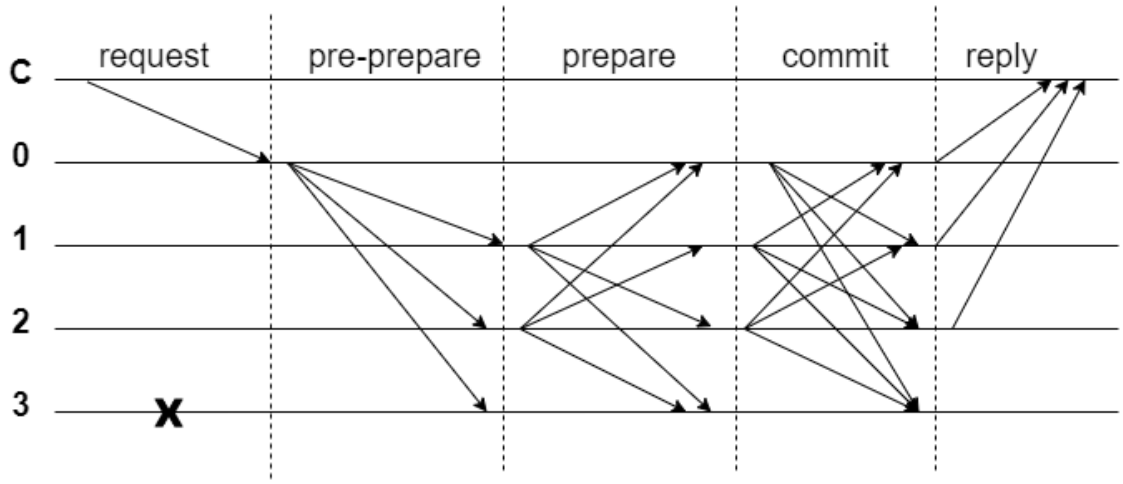

Figure 3.2: Execution of PBFT protocol. Node 0 is primary, rest of the nodes are replicas where replica number 3 failed [9].

Consensus mechanisms compared

Consensus	Nakamoto (PoW)	Avalanche (PoS)	Classical (PBFT)
Scalable	✓	✓	x
Robust	✓	✓	x
Highly decentralized	✓	✓	x
Low latency	x	✓	✓
High throughput	x	✓	✓
Lightweight	x	✓	✓
Sustainable	x	✓	✓
Resilient to 51%attacks	x	✓	x

Table 3.1: Comparison of consensus mechanisms [3].

Avalanche consensus, which is based on Proof-of-Stake, claims to have the best properties among Nakamoto consensus based on Proof-of-Work and PBFT, which is a Classical consensus. In the next chapter, we will describe the structure of Avalanche network and also how Avalanche consensus protocol works.

Chapter 4

Avalanche

Avalanche is an open-source platform for launching decentralized applications and enterprise blockchain deployments in one interoperable, highly scalable ecosystem that aims to address the blockchain trilemma of scalability, security and decentralization through Proof of Stake mechanism. It supports smart contracts written in the Solidity language and aims to create greater blockchain interoperably by integrating a number of decentralized finance ecosystems, including well-established projects. The native token of the Avalanche platform - AVAX, is used to power transactions in its ecosystem and serves as the means to distribute system rewards, participate in governance and facilitate transactions on the network by paying fees. There are three primary aspects of its design that distinguish it from other blockchain projects: its consensus mechanism, its incorporation of subnetworks and its use of multiple built-in blockchains.

Avalanche uses a novel consensus mechanism that builds on the PoS foundation. When a transaction is initiated by a user, it's received by a validator node that samples a small, random set of other validators, checking for agreement. The validators perform this sampling procedure repeatedly, "gossiping" with each other to ultimately reach consensus. In this way, one validator's message is sent to other validators, which sample more validators, which sample even more validators – again and again, until the whole system reaches agreement on an outcome. Just as a single snowflake can become a snowball, a single transaction can eventually turn into an avalanche. Validator rewards scale according to the amount of time a node has staked its tokens, called Proof of Uptime, and if the node has historically acted according to the software's rules, called Proof of Correctness.

Avalanche users can launch specialized chains that can operate using their own sets of rules. This system is comparable to other blockchain scaling solutions, like Polkadot's parachains and Ethereum 2.0's shards. Consensus on these chains is reached by subnetworks, which are groups of nodes that participate in validating a designated set of blockchains. All subnet validators must also validate Avalanche's Primary Network [3, 12].

4.1 Platform

Avalanche is built using three different blockchains in order to address the limitations of the blockchain trilemma. All 3 blockchains are validated and secured by the Primary Network. The Primary Network is a special subnet, and all members of all custom subnets must also be a member of the Primary Network by staking at least 2,000 AVAX. Digital assets can be moved across each of these chains to accomplish different functions within the ecosystem.

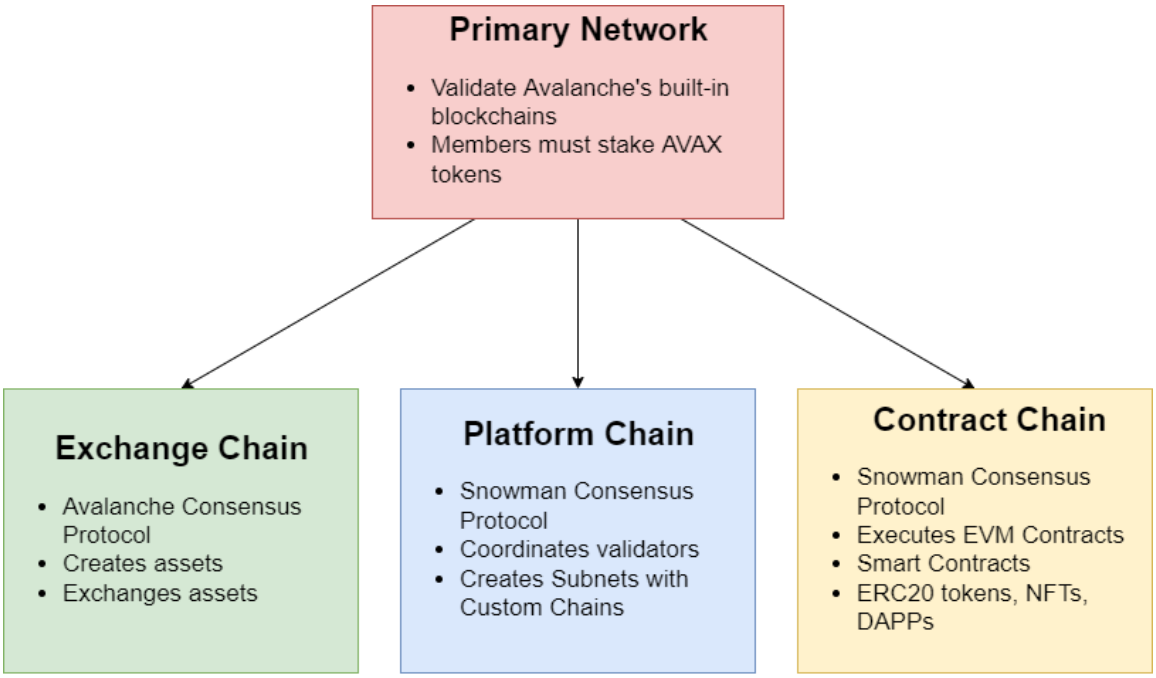

Figure 4.1: Avalanche network structure [3].

- **The Exchange Chain (X-Chain)** is the default chain based on Direct Acyclic Graph and acts as a decentralized platform for creating and trading digital smart assets, a representation of a real-world resource (e.g., equity, bonds) with a set of rules that govern its behavior. This includes Avalanche's native token, AVAX. The X-Chain is an instance of the Avalanche Virtual Machine (AVM). The X-Chain API allows clients to create and trade assets on the X-Chain and other instances of the AVM.
- **The Contract Chain (C-Chain)** allows for the creation and execution of smart contracts using the C-Chain's API. Because it is based on the Ethereum Virtual Machine (EVM), Avalanche's smart contracts can take advantage of cross-chain interoperability.
- **The Platform Chain (P-Chain)** is the metadata blockchain on Avalanche and coordinates validators, keeps track of active subnets, and enables the creation of new subnets. The P-Chain implements the Snowman consensus protocol.

  The P-Chain API allows clients to create subnets, add validators to subnets, and create blockchains

DAG

Directed Acyclic Graph is a data structure that uses topological ordering and is mostly used for solving problems such as data processing, finding the best route for navigation, scheduling, and data compression. As opposed to blockchain, DAG eliminates the block creation and the transactions go directly into the validation process. After validation, every transaction is linked to a new and also an existing transaction on the network thus keeping the network width under a certain range that can support quick transaction and its validation. Time to confirm and the speed of executing a transaction does not depend on block-size but on the bandwidth between nodes. Therefore the scalability of transactions is increased in the DAG network [4].

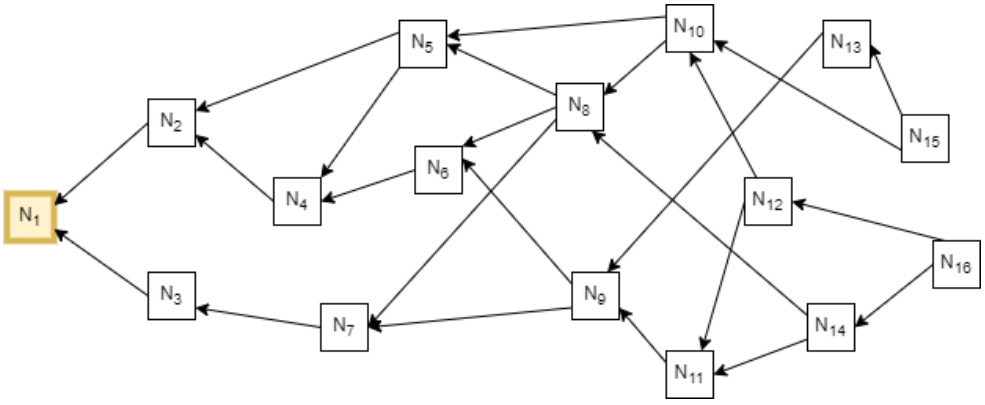

Figure 4.2: A DAG holds a genesis transaction defining an initial state [11]

Subnetworks

Subnetwork (a.k.a.,subnet), is a dynamic set of validators (a.k.a.,nodes) working together to achieve consensus on the state of a set of blockchains. Each blockchain is validated by exactly one subnetwork. A subnetwork can validate many blockchains. A node may be a member of many subnetworks. A subnet manages its own membership, and it may require that its validators have certain properties. Some examples of requirements include:

- Validators must be located in a given country.
- Validators must pass a KYC/AML checks.
- Validators must hold a certain license.

These requirements do not apply to the Avalanche Primary Network [3].

4.2 Consensus algorithm

A key difference between Avalanche and other decentralized networks is the consensus protocol which employs a novel approach to consensus to achieve its strong safety guarantees, quick finality, and high-throughput without compromising decentralization.

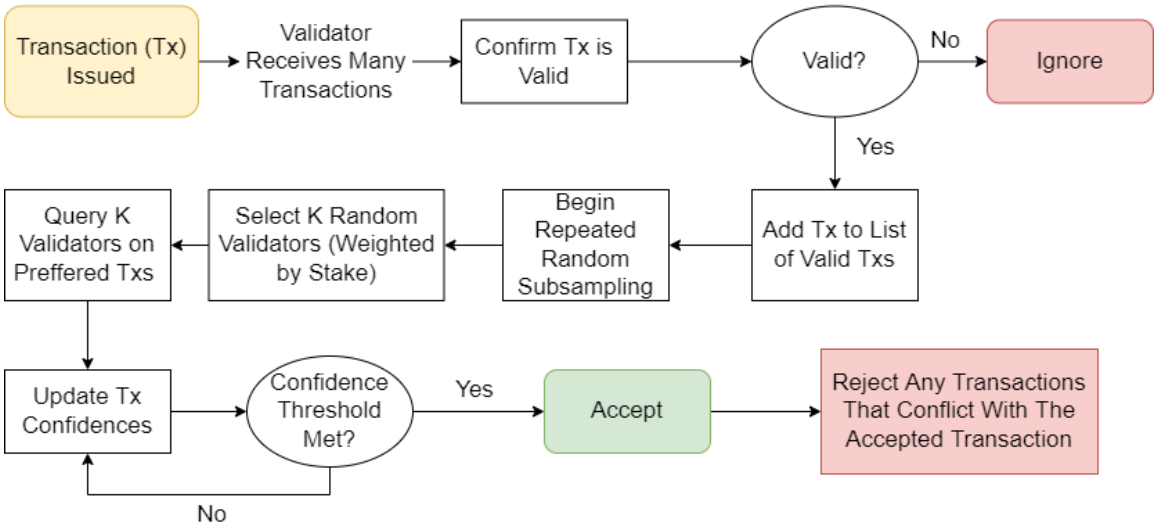

Figure 4.3: Avalanche consensus flowchart [3].

Avalanche starts with a protocol called Slush and progressively build up to Snowflake and Snowball, all based on the same common majority-based metastable voting mechanism. These protocols are single-decree consensus protocols of increasing robustness. This section provides full specifications for the protocols inspired by epidemic or gossip protocols [12, 3].

Slush

The simplest protocol shown in Algorithm 1, Slush, is the foundation of this family of protocols. Slush is not tolerant to Byzantine faults, only crash-faults (CFT), but serves as an illustration for the Byzantine fault tolerant protocols that follow. For better understanding, Slush will be described using a decision between two conflicting colors, red and blue.

In Slush, a node starts out initially in an uncolored state. Upon receiving a transaction from a client, an uncolored node updates its own color to the one carried in the transaction and initiates a query. To perform a query, a node picks a small, constant sized (k) sample of the network uniformly at random, and sends a query message. Upon receiving a query, an uncolored node adopts the color in the query, responds with that color, and initiates its own query, whereas a colored node simply responds with its current color. Once the querying node collects k responses, it checks if a fraction $\geq \alpha$ are for the same color, where $\alpha > \lfloor k/2c \rfloor$ is a protocol parameter. If the α threshold is met and the sampled color differs from the node's own color, the node flips to that color. It then goes back to the query step, and initiates a subsequent round of query, for a total of m rounds. Finally, the node decides the color it ended up with at time m.

Algorithm 1 Slush [12]

```
1: procedure onQuery(v, col')
2:     if col = ⊥ then col := col'
3:     RESPOND(v, col)
4: procedure slushLoop(u, col_0 ϵ {R,B,⊥})
5:     col := col_0 //initialize with a color
6:     for r ϵ {1...m} do
7:         //if ⊥, skip until onQuery sets the color
8:         if col = ⊥ then continue
9:         //randomly sample from the known nodes
10:        K := SAMPLE(N\u, k)
11:        P := [QUERY(v, col) for v ϵ K]
12:        for col' ϵ {R,B} do
13:            if P.COUNT(col') ≥ α then
14:                col := col'
15:    ACCEPT(col)
```

Slush has a few properties of interest. First, it is almost memoryless: a node retains no state between rounds other than its current color, and in particular maintains no history of interactions with other peers. Second, unlike traditional consensus protocols that query every participant, every round involves sampling just a small, constant-sized slice of the network at random. Third, Slush makes progress under any network configuration even fully bivalent state (i.e. 50/50 split between colors), since random perturbations in sampling will cause one color to gain a slight edge and repeated samplings afterwards will build upon and amplify that imbalance. Finally, if `m` is chosen high enough, Slush ensures that all nodes will be colored identically with high probability. Each node has a constant, predictable communication overhead per round, and `m` grows logarithmically with `n`. The Slush protocol does not provide a strong safety guarantee in the presence of Byzantine nodes. In particular, if the correct nodes develop a preference for one color, a Byzantine adversary can attempt to flip nodes to the opposite so as to keep the network in balance, preventing a decision. This issue is addressed in Snowflake, the first BFT protocol that introduces more state storage at the nodes [12].

Snowflake

Snowflake augments Slush with a single counter that captures the strength of a node's conviction in its current color. This per-node counter stores how many consecutive samples of the network by that node have all yielded the same color. A node accepts the current color when its counter reaches β, another security parameter. Algorithm 2 shows the amended protocol, which includes the following modifications:

- Each node maintains a counter `cnt`.
- Upon every color change, the node resets `cnt` to 0.
- Upon every successful query that yields $\geq \alpha$ responses for the same color as the node, the node increments `cnt`.

Algorithm 2 Snowflake [12]

```
1:  procedure SNOWFLAKELOOP(u, col_0 ϵ {R,B,⊥})
2:      col := col_0, cnt := 0
3:      while undecided do
4:          if col = ⊥ then continue
5:          K := SAMPLE(N\u,k)
6:          P := [QUERY(v,col) for v ϵ K]
7:          maj := false
8:          for col' ϵ {R,B} do
9:              if P.COUNT(col') ≥ α then
10:                 maj := true
11:                 if col' ≠ col then
12:                     col := col', cnt := 1
13:                 else cnt++
14:                 if cnt ≥ β then ACCEPT(col')
15:         if maj = false then cnt := 0
```

When the protocol is correctly parameterized for a given threshold of Byzantine nodes and a desired ϵ-guarantee, it can ensure both safety and liveness. As will be later shown, there exists an irreversible state after which a decision is inevitable. Correct nodes begin to commit past the irreversible state to adopt the same color, with high probability. There also exists a phase-shift point, where the Byzantine nodes lose ability to keep network in a bivalent state [12].

Snowball

Snowflake's notion of state is ephemeral: the counter gets reset with every color flip. Snowball augments Snowflake with confidence counters that capture the number of queries that have yielded a threshold result for their corresponding color Algorithm 3. A node decides if it gets β consecutive chits for a color. However, it only changes preference based on the total accrued confidence. The differences between Snowflake and Snowball are as follows:

- Upon every successful query, the node increments its confidence counter for that color.
- A node switches colors when the confidence in its current color becomes lower than the confidence value of the new color [12].

Algorithm 3 Snowball [12]

procedure SNOWBALLLOOP($u, col0 \in$ {R,B,$\perp$})
  $col := col0, lastcol := col_0, cnt := 0$
  $d[R] := 0, d[B] := 0$
  while undecided **do**
    if $col = \perp$ **then continue**
    $K :=$ `SAMPLE`$(N \backslash u, k)$
    $P :=$ [`QUERY`(v, col) **for** $v \in K$]
    $maj := false$
    for $col' \in$ {R,B} **do**
      if P.`COUNT`$(col') \geq \alpha$ **then**
        $maj := true$
        $d[col'] + +$
        if $d[col'] > d[col]$ **then**
          $col := col'$
        if $col' \neq lastcol$ **then**
          $lastcol := col', cnt := 1$
        else $cnt + +$
        if $cnt \geq \beta$ **then** `ACCEPT`(col')
    if $maj = false$ **then** $cnt := 0$

Conclusion

In this work, we want to simulate properties and behaviour of Avalanche consensus algorithm to analyze its consensus performance and security when it comes to large networks with tens of thousands of nodes with different properties.

Chapter 5

Simulation tools

Several simulation tools were developed for the purpose of testing the properties of blockchain protocols. Most of them have been created for simulating Proof-of-Work protocols mainly due to their considerable prevalence. In our case, we will investigate the performance of the consensus algorithms implemented in Avalanche and their behavior with various properties or under various attacks. Let's take a look at few interesting simulation tools used to test blockchain networks.

5.1 SimBlock

SimBlock is event-driven simulator, that considering block generation and message transmission/reception as events. Each node generates messages and events informing the mining process. The user has the ability to edit parameters such as:

- Block size: The size of the block generated by the node.
- Block generation interval: Block generation interval targeted by the blockchain.
- Number of nodes: The number of nodes involved in the blockchain network.
- Number of neighbor nodes: The number of neighbor nodes of each node.
- Location of the node. Network parameters are determined by the region.
- Block generation capacity: The block generation capacity of each node. The block generation difficulty is obtained from the sum of the block generation capacity of all the nodes and the target of the block generation interval.
- Network bandwidth: The upstream and downstream bandwidths for each region.
- Network propagation delay: The average value of propagation delay between regions.

It supports the creation of a model for any type of consensus, with models already implemented by the authors for the Proof-of-Work protocols Bitcoin, Litecoin and Dogecoin. The tool is written in Java and supports visualization from the output stored in JSON format [2].

5.2 BlockSim

The BlockSim tool is based on a stochastic simulation model. It can model events based on the probability distribution of their occurrence. This requires changes in the state of the system in discrete time. BlockSim controls operations running in continuous time in certain time intervals, thus making them discretized and at the same time less demanding to run. This makes it possible to monitor hundreds of nodes simultaneously. Users are allowed to take advantage of this tool to simulate existing protocols in already created models or create new models.

BlockSim allows specification of various parameters including latency or duration of block creation. It is also possible to create factories for nodes or transactions, that will later be broadcasted to nodes. Due to its dynamic nature, it allows nodes to connect and also disconnect from the simulated system. The architecture shown in Figure 5.1 includes a module for monitoring the simulation from captured metrics and a module for creating reports. The entire simulation can be edited by the user in the BlockSim programming interface using Python. It is primarily designed for testing the performance of protocols and does not support behavioral testing for different types of attacks on the blockchain [5].

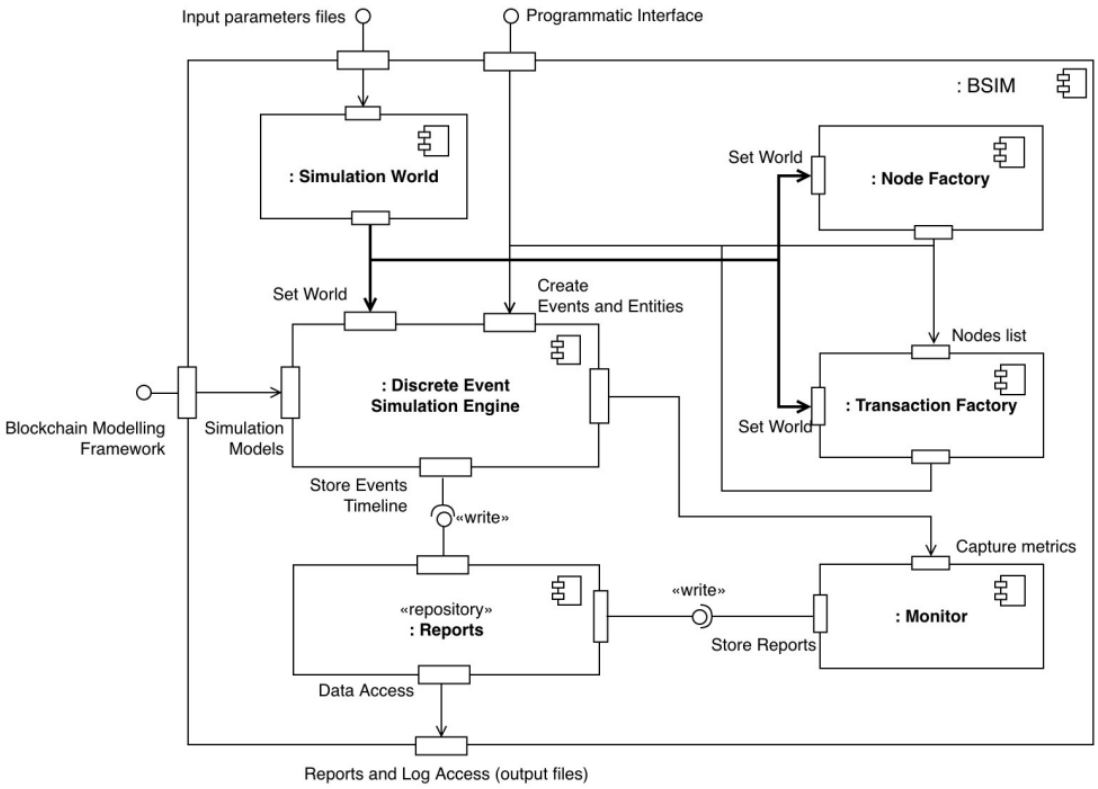

Figure 5.1: BlockSim architecture [5].

5.3 VIBES

VIBES is a blockchain network simulator built on peer-to-peer technology. It differentiates itself from other blockchain simulators in its ability to simulate blockchain systems beyond bitcoin and its support for large-scale simulations with thousands of nodes. Its big advantages are optimal scalability and high speed. It supports configuration of various input parameters such as network topology, area size, latency, bandwidth, number of nodes, number of miners, block size, block confirmation time, number of transactions per block, transaction size, electricity cost, smart contract time, smart contract intensity and smart contract density, percentage of attacker nodes or percentage of failing nodes. The output of the simulation are values representing statistical information about the transactions, total time to process, total number of transactions processed, throughput (transactions per second), block propagation delay for, client bootstrap time, cost per transaction, probability of an attacker taking over at each stage, and a log of all transactions [14].

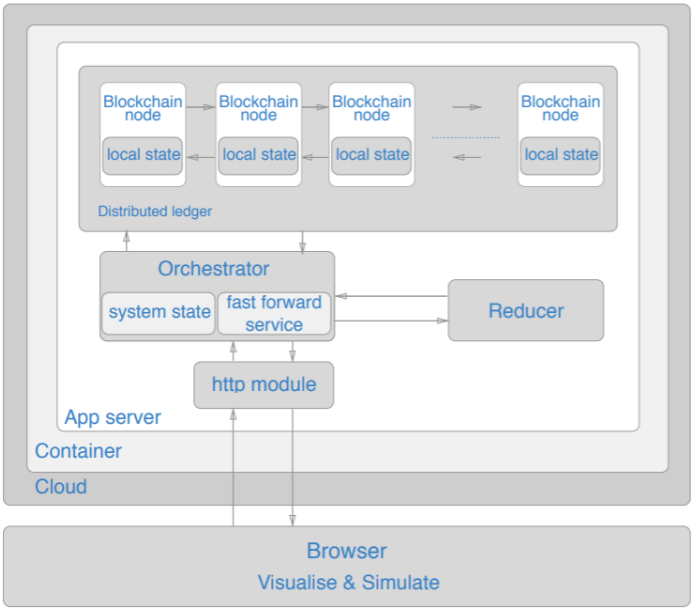

Figure 5.2: VIBES architecture [14].

One of the great advantages it provides is the possibility of accelerated simulation. If parameters specified at startup are theoretical values that represent the individual calculations, VIBES can skip the difficult calculations. For example, if the duration of the block creation time is 10 minutes, then VIBES in accelerated simulation mode will perform the necessary calculations and allow the node to create the block at a rate of a few milliseconds. Full simulation time is thus significantly shorter and the user is able to change the necessary parameters more frequently and obtain desired results. At the same time, VIBES can provide simulation speedup without loss of quality of the results. The VIBES simulator architecture shown in Figure 5.2 uses cloud services to run the application server [14].

5.4 BlockZoom

Similar to VIBES, BlockZoom supports the simulation of large-scale blockchain systems. The simulation environment is based on the Grid'5000 platform and seeks to create conditions similar to those in a production environment. The user is able to change the configuration of individual nodes and then compare the results of the simulations performed. In addition to the configuration of nodes, the system load can also be adjusted, which allows us to create different stress conditions for the blockchain. The simulations use private network between resources stored on several geographical locations in France and the Netherlands. This makes the simulation properties, such as network latency, very similar to the real ones. This tool is still under development at the time of writing and thus its properties are mainly theoretical. The architecture of the BlockZoom tool is shown in Figure 5.3 [13].

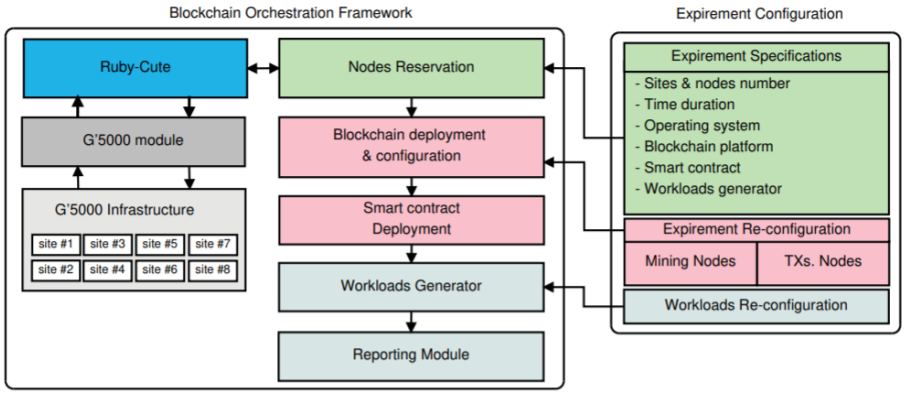

Figure 5.3: BlockZoom architecture [13].

5.5 Proof of Stake support

	Protocols	PoW and PoS support
SimBlock	Bitcoin, Dogecoin, Litecoin	PoW, other consensus mechanisms
BlockSim	Bitcoin,Ethereum	PoW
VIBES	Bitcoin	PoW
BlockZoom	Ethereum	PoW

Table 5.1: Comparison of supported protocols [13, 14, 5, 2].

These simulation tools do not state whether they are designed for the needs of simulating Proof-of-Work protocols or whether they also support Proof-of-Stake based protocols. However, mostly PoW protocols have been simulated by them. Table 5.1 gives a summary of the protocols that have been simulated by the authors of each tool and their support of PoW and PoS protocols.

For our work, we decided **not to use** any of the mentioned tools for the sake of **not supporting PoS protocols**. Instead we implement a simulator that focuses more on the Avalanche consensus algorithm and its properties.

Chapter 6

Implementation and evaluation

Choosing the right tool is the key to simulation. Each of the tools that have been mentioned are different and have their advantages or disadvantages. To test the impact of consensus settings on performance and safety, it is necessary to create a new tool that is built with Slush Algorithm 1, Snowflake Algorithm 2 and Snowball Algorithm 3 in its core.

We implemented a Python[1] script `avalanche-sim.py` that is able to simulate Avalanche consensus for thousands of nodes using NumPy[2] library for computing and Matplotlib[3] library for visualization of the results. It produces the Avalanche simulation array of round results for a given set of parameters. The user has the ability to edit parameters such as:

- Number of nodes in the network.
- Stake distribution of nodes.
- Maximum number of rounds.
- Sample size in polling mechanism.
- Virtuous commit threshold β.
- Percentage of malicious nodes in network.
- Number of simulations.

All the optional parameters are shown in Figure 6.1.

6.1 Simulator description

The core procedure `avalanche` simulates the Avalanche consensus algorithm for the provided number of nodes, sample size and stake weights. It applies the stake-weighted Slush polling mechanism and makes a random choice of a sample size for each element within the array of nodes. Then for each sample determines, if the α-majority of the sample indicates red or blue colors. However, if there is no α-majority within the sample, no changes are made.

[1] https://www.python.org
[2] https://numpy.org
[3] https://matplotlib.org

It also applies the percentage of population, which is malicious towards the network attempting to flip peer nodes without actually following the consensus rules if the needed parameter is given. Then the algorithm proceeds to apply the Snowflake conviction model onto the nodes and returns an array of results for each round.

```
> python .\avalanche-sim.py --help
usage: avalanche-sim.py [-h] [-N NODENUMBER]
[-D {normal,uniform,exponential,cauchy,equal}]
[-R MAXROUND] [-S SAMPLESIZE]
[-B BETA] [-M MALICIOUS] [-n SIMNUM]

Produces the Avalanche simulation array of round results for a given set of parameters.

options:
  -h, --help show this help message and exit

  -N NODENUMBER, --nodeNumber NODENUMBER
   number of nodes in the network(default: 10000)

  -D {normal,uniform,exponential,cauchy,equal},
  --distribution {normal,uniform,exponential,cauchy,equal}
    stake distribution(default: uniform)

  -R MAXROUND, --maxRound MAXROUND
    maximum number of rounds(default: 20)

  -S SAMPLESIZE, --sampleSize SAMPLESIZE
    sample size (default: 20)

  -B BETA, --beta BETA virtuous commit threshold (default: 15)

  -M MALICIOUS, --malicious MALICIOUS
    Share of malicious nodes in [0.0, 0.5](default: 0.0)

  -n SIMNUM, --simNum SIMNUM
    number of simulations(default: 1)
```

Figure 6.1: The run-time arguments of `avalanche-sim.py`.

Once the simulator was implemented, tests were performed and the results made it possible to approximate the properties of Avalanche protocol. In addition to standard tests, such as the effect of number of rounds on consensus algorithm, various other experiments were performed. Using them we monitored the tolerance of the Avalanche consensus protocol to malicious nodes and we also simulated scenarios with various adjustments to see what is the optimal setup. In this chapter we will describe the method of testing and the observed results.

6.2 Experiments

In blockchain systems, it is important to be able to quickly process large amounts of transactions. The more transactions that can be logged into the ledger, the better. However, throughput is closely related to the liveness of the protocol, because if we are able to insert blocks or transactions into the ledger faster, the number of transactions written will also increase. We can also assume the number of rounds in which we can reach consensus also greatly affects the throughput of the protocol, the lesser the rounds the consensus is reached within, the faster the transactions are processed.

We investigated how many rounds does it usually take to reach consensus with a certain number of participants and sample size in a fully bivalent state (i.e. 50/50 split between colors). Another important aspect is security, however, with our simulator we are only able to investigate the security of the consensus,therefore we investigated how malicious nodes are affecting reaching of consensus. In terms of scalability, we investigated how a large number of nodes achieves consensus as many may expect, that by drastically increasing the participants in the network by an order to tens of thousands, a similar increase of the sample size and rounds would be required to effectively reach consensus.

First set of experiments

Firstly, we concluded basic experiments using the same parameters as the actual mainnet, the number of rounds is 20, the size of subsamples is also 20 and the quorum rate is 70% of the subsample size with all the attributes shown in Table 6.1 and the results are shown in Figure 6.2, Figure 6.3, Figure 6.4 and Figure 6.5.

Nodes	Sample size	Rounds	Simulations
1000	20	20	100
10000	20	20	100
50000	20	20	100
100000	20	20	100

Table 6.1: First set of experiments.

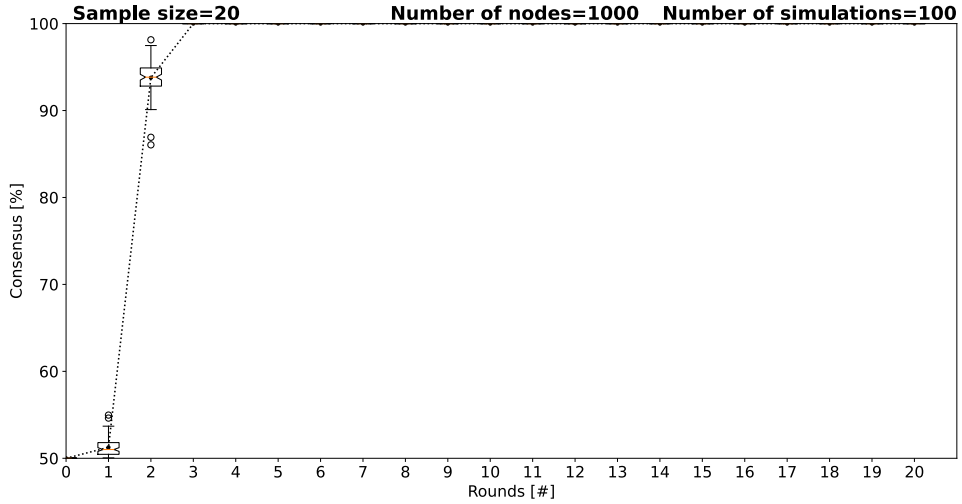

Figure 6.2: First set of experiments. Nodes = 1000.

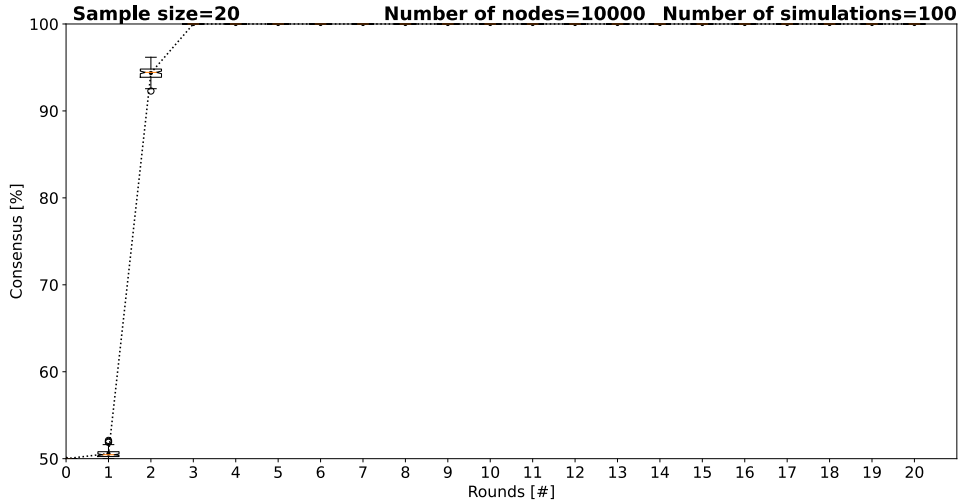

Figure 6.3: First set of experiments. Nodes = 10000.

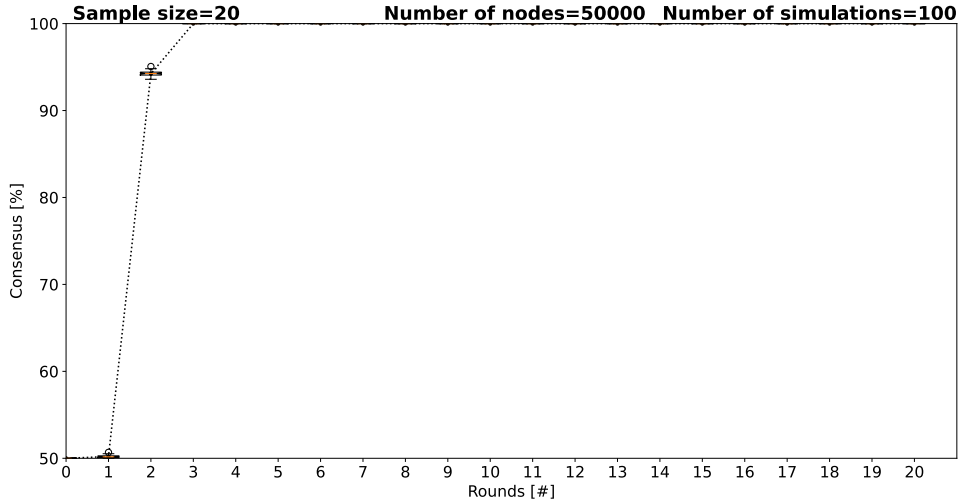

Figure 6.4: First set of experiments. Nodes = 50000.

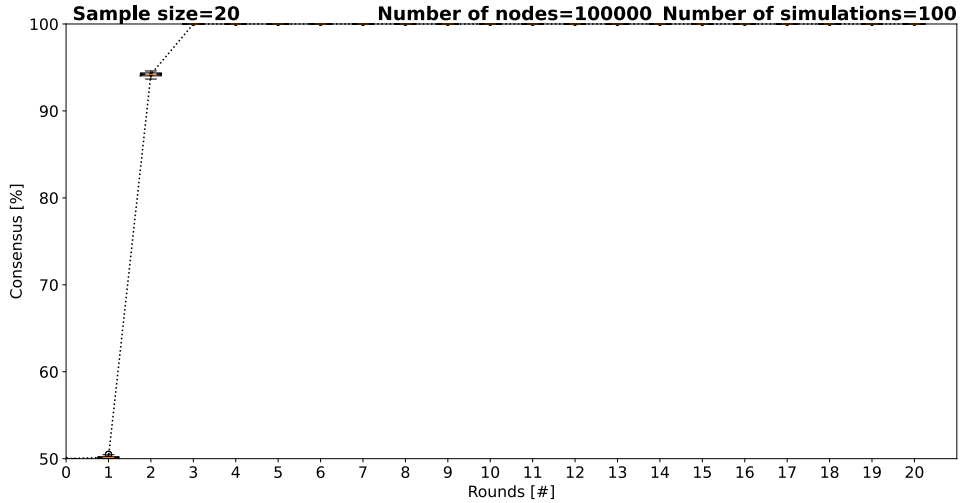

Figure 6.5: First set of experiments. Nodes = 100000.

As these experiments show, the parameters of the mainnet are set so that Avalanche consensus is highly scalable at the expense of speed. Even though the consensus is reached within three rounds, the sample size greatly affects the time spent in each round.

Second set of experiments

In the second set of experiments the size of subsamples has decreased to 5 to see if there are any significant changes in the number of rounds within the consensus can be reached. All the attributes are shown in Table 6.2 and the results are shown in Figure 6.6, Figure 6.7, Figure 6.8 and Figure 6.9.

Nodes	Sample size	Rounds	Simulations
1000	5	20	100
10000	5	20	100
50000	5	20	100
100000	5	20	100

Table 6.2: Second set of experiments.

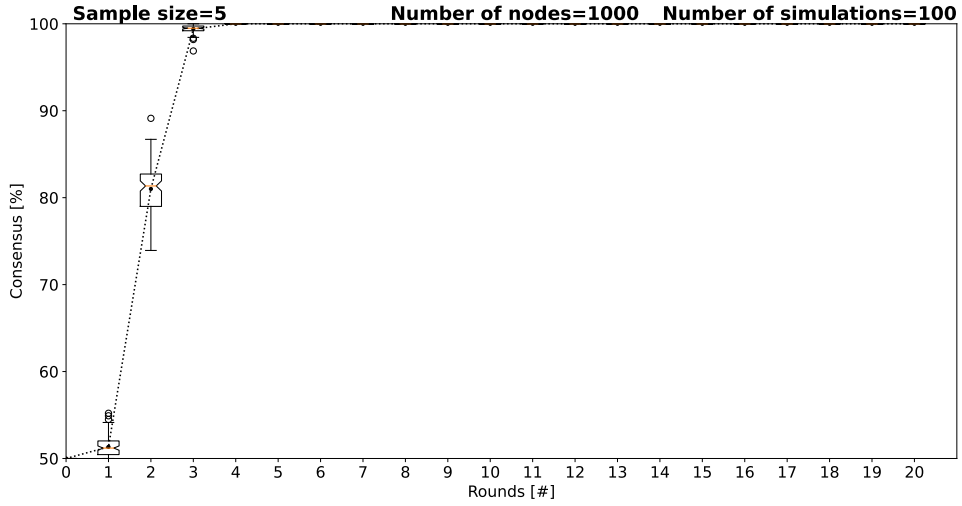

Figure 6.6: Second set of experiments. Nodes = 1000.

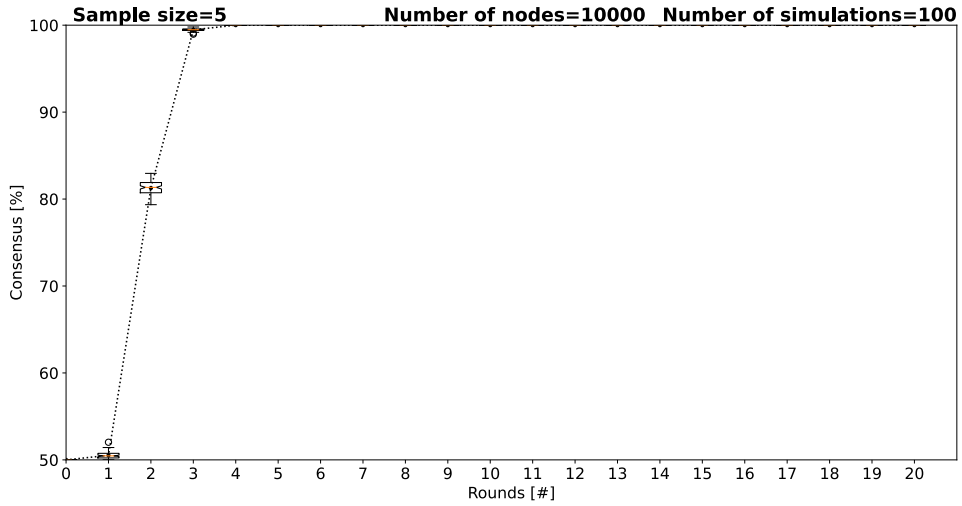

Figure 6.7: Second set of experiments. Nodes = 10000.

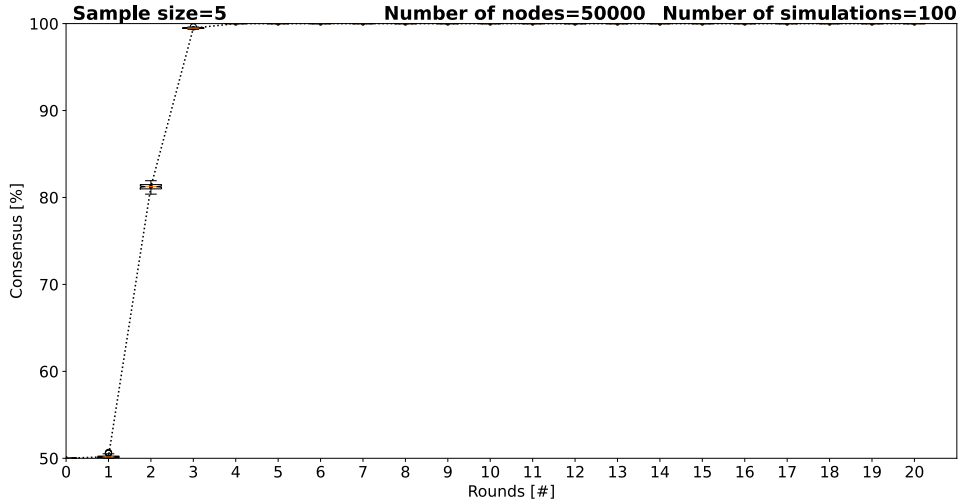

Figure 6.8: Second set of experiments. Nodes = 50000.

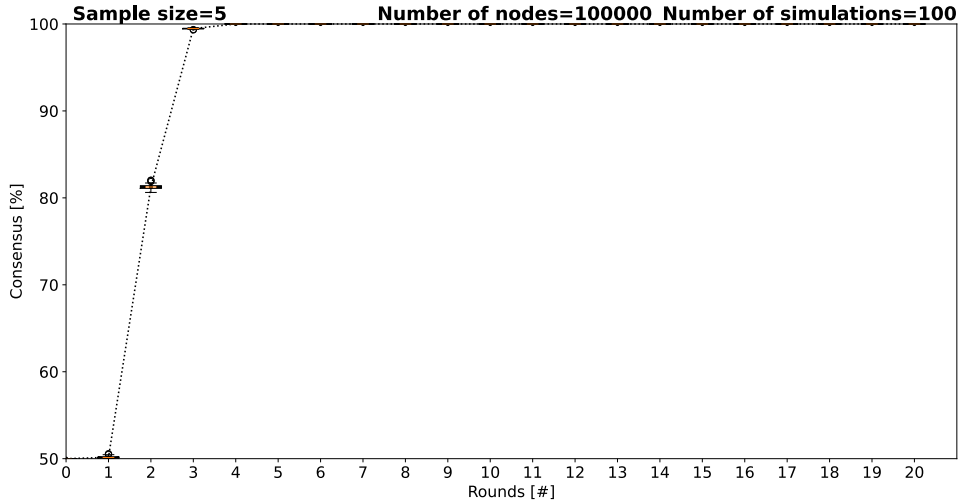

Figure 6.9: Second set of experiments. Nodes = 100000.

The second set of experiments shown that with sample size of 5 there is a small decrease in consensus. The consensus can be reached within 4 rounds as opposed to 3 rounds with sample size of 20. We can assume the time spent in reaching consensus is lower than with the sample size of 20 as the time spent with subsampling 5 participants is much lower than time spent subsampling 20 participants.

Third set of experiments

In the third set of experiments we apply malicious nodes where a percentage of the network participants by stake are modeled to be malicious and are trying to sabotage the network in each round from reaching the maximum possible consensus level. All the attributes are shown in Table 6.3 and the results are shown in Figure 6.10, Figure 6.11, Figure 6.12 and Figure 6.13.

Nodes	Malicious nodes[%]	Sample size	Rounds	Simulations
10000	20	5	20	100
10000	20	20	20	100
10000	40	5	20	100
10000	40	20	20	100

Table 6.3: Third set of experiments.

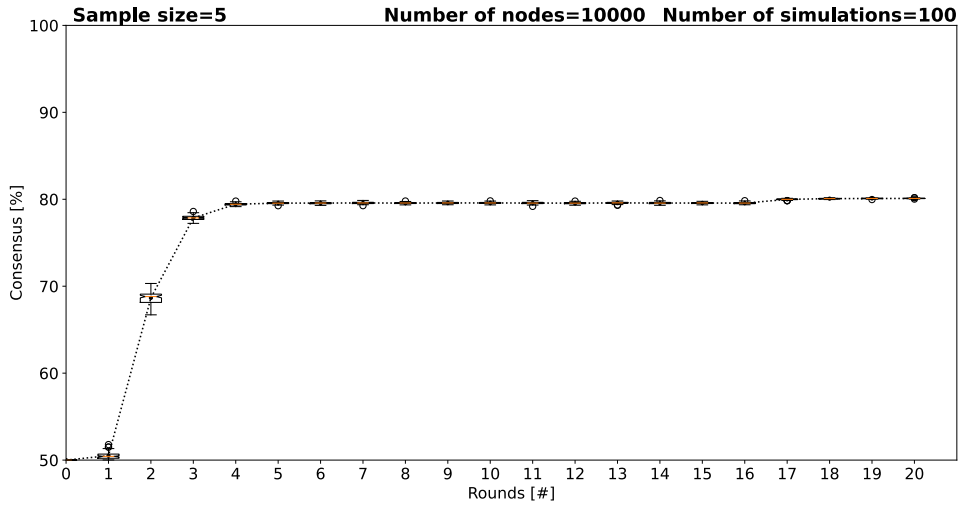

Figure 6.10: Third set of experiments. Malicious nodes make up 20% of all nodes.

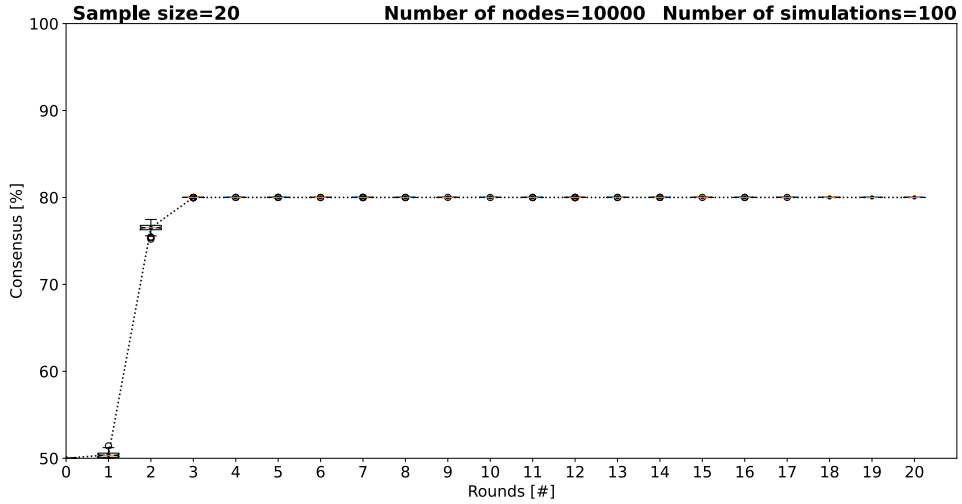

Figure 6.11: Third set of experiments. Malicious nodes make up 20% of all nodes.

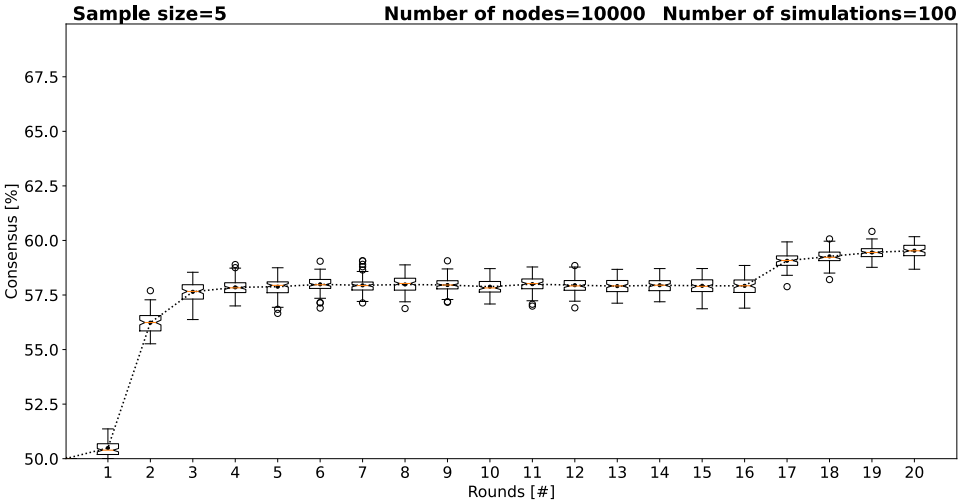

Figure 6.12: Third set of experiments. Malicious nodes make up 40% of all nodes.

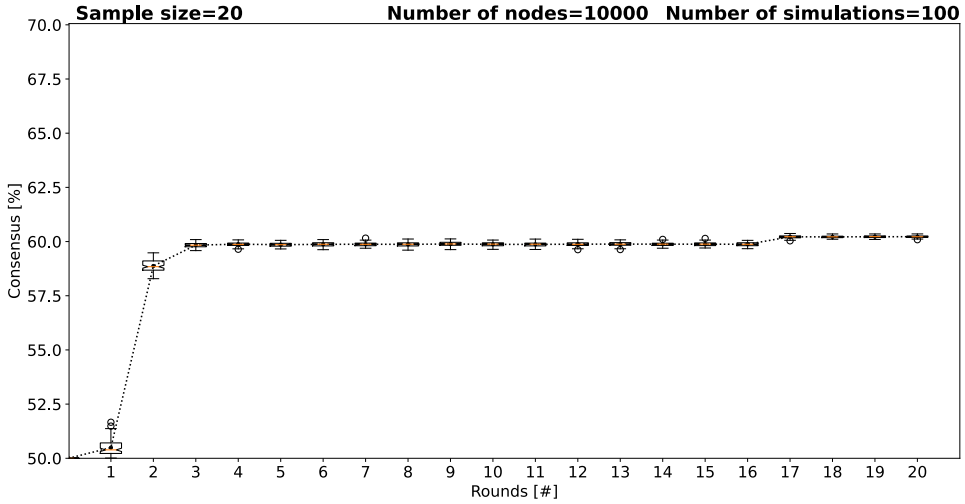

Figure 6.13: Third set of experiments. Malicious nodes make up 40% of all nodes.

The third set of experiments shown that with a sample size of 20, the network easily and very quickly reaches the maximum possible consensus level and then tightly maintains it despite constant interference by malicious network participants. The network with a sample size of 5, however, seem to be worse at maintaining the maximum possible consensus level as well as being slower reaching it.

6.3 Summary

While there seems to be an engineering **trade-off** between a lower sample size and a higher number of rounds, sample size of **5** seems to be a good choice when it comes to **private networks** that can be run on Avalanche subnetworks, as there would be no malicious nodes trying to overthrow the subnet and the **consensus would be achieved quite efficiently and quickly** in a small scaled subnet. However, when it comes to **public networks**, it is better to have more **security**. Therefore, the mainnet, which is a large network, using sample size of **20**, not only is **more resistent to malicious nodes** when it comes to achieving consensus, but also **reaches consensus faster** as a larger network works better with higher sample size. Another **security** measure seems to be the amount of **rounds**, which in case of mainnet and our experiments, is **20**. We could expect that these parameters can be even applied to a much larger networks of one million nodes without almost any significant decrease in consensus.

Chapter 7

Conclusion

The goal of this thesis was to implement a Avalanche consensus simulator that is able to accurately simulate network of thousands of nodes reaching consensus in various scenarios. We learned how blockchain networks work and the difference between the various consensus mechanisms and protocols. We learned about transactions, blocks, smart contracts and new approaches to distributed consensus protocols with Direct Acyclic Graph. We studied and analyzed individual consensus protocols with consensus mechanisms as well as blockchain simulators. Next, we figured out how to implement *Slush*,*Snowflake* and *Snowball* into the proposed Avalanche consensus simulator and conducted experiments upon the implementation. We summarized results of mentioned experiments and came to a conclusion about various parameters of Avalanche consensus algorithm. The experiments were used to analyze Avalanche consensus, but the real-world Avalanche network analysis would have to be done on much more robust and sophisticated simulator, which is also the weakness of this approach we used to implement this simulator. This thesis can be extended with the implementation of network or other layers, which could significantly improve the usability of the final simulator for other experiments. Nevertheless, we came to a conclusion that the overall approach of using the model of an Avalanche seems to be an efficient process of generating consensus among a very large population.

Bibliography

[1] Alharby, M. and Moorsel, A. van. Blockchain-based Smart Contracts: A Systematic Mapping Study. *CoRR*. 2017, abs/1710.06372. Available at: http://arxiv.org/abs/1710.06372.

[2] Aoki, Y., Otsuki, K., Kaneko, T., Banno, R. and Shudo, K. SimBlock: A Blockchain Network Simulator. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2019, p. 325–329. DOI: 10.1109/INFCOMW.2019.8845253.

[3] Ava Labs, I. *Avalanche*. 2021. https://docs.avax.network [Online; posted 2021].

[4] Benčić, F. M. and Podnar Žarko, I. Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph. In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. 2018, p. 1569–1570. DOI: 10.1109/ICDCS.2018.00171.

[5] Faria, C. and Correia, M. BlockSim: Blockchain Simulator. In: *2019 IEEE International Conference on Blockchain (Blockchain)*. 2019, p. 439–446. DOI: 10.1109/Blockchain.2019.00067.

[6] Homoliak, I., Venugopalan, S., Reijsbergen, D., Hum, Q., Schumi, R. et al. The Security Reference Architecture for Blockchains: Toward a Standardized Model for Studying Vulnerabilities, Threats, and Defenses. *IEEE Communications Surveys Tutorials*. 2021, vol. 23, no. 1, p. 341–390. DOI: 10.1109/COMST.2020.3033665.

[7] Academy, H. *Blockchain as a Data Structure* [online]. 2019 [cit. 2021-12-29]. Available at: https://academy.horizen.io/technology/expert/blockchain-as-a-data-structure/.

[8] Kathleen E. Wegrzyn, E. W. *Types of Blockchain: Public, Private, or Something in Between*. 2021. https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between [Online; posted 2021].

[9] Lei, K., Zhang, Q., Xu, L. and Qi, Z. Reputation-Based Byzantine Fault-Tolerance for Consortium Blockchain. In:. December 2018. DOI: 10.1109/PADSW.2018.8644933.

[10] Miller, A. Permissioned and permissionless blockchains. *Blockchain for Distributed Systems Security*. John Wiley & Sons. 2019, p. 193–204.

[11] Peresíni, M., Bencic, F. M., Malinka, K. and Homoliak, I. DAG-Oriented Protocols PHANTOM and GHOSTDAG under Incentive Attack via Transaction

Selection Strategy. *CoRR*. 2021, abs/2109.01102. Available at: https://arxiv.org/abs/2109.01102.

[12] Rocket, T. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies. *Available [online].[Accessed: 4-12-2018]*. 2018.

[13] Shbair, W. M., Steichen, M., François, J. and State, R. BlockZoom: Large-Scale Blockchain Testbed. In: *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 2019, p. 5–6. DOI: 10.1109/BLOC.2019.8751230.

[14] Stoykov, L., Zhang, K. and Jacobsen, H.-A. VIBES: Fast Blockchain Simulations for Large-Scale Peer-to-Peer Networks: Demo. In: *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*. New York, NY, USA: Association for Computing Machinery, 2017, p. 19–20. Middleware '17. DOI: 10.1145/3155016.3155020. ISBN 9781450352017. Available at: https://doi.org/10.1145/3155016.3155020.

[15] Times, T. N. Y. *Bitcoin consumption*. 2021. https://www.nytimes.com/interactive/2021/09/03/climate/bitcoin-carbon-footprint-electricity.html?smid=url-sharek [Online; posted 2021].

[16] Yaga, D., Mell, P., Roby, N. and Scarfone, K. Blockchain technology overview. National Institute of Standards and Technology. Oct 2018. DOI: 10.6028/nist.ir.8202. Available at: http://dx.doi.org/10.6028/NIST.IR.8202.

[17] Yu, Z., Liu, X. and Wang, G. A Survey of Consensus and Incentive Mechanism in Blockchain Derived from P2P. In: *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. 2018, p. 1010–1015. DOI: 10.1109/PADSW.2018.8645047.

[18] Zheng, Z., Xie, S., Dai, H., Chen, X. and Wang, H. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In: *2017 IEEE International Congress on Big Data (BigData Congress)*. 2017, p. 557–564. DOI: 10.1109/BigDataCongress.2017.85.

Appendix A

Contents of the included storage media

Source codes of the Avalanche consensus simulator are located on the attached compact disk in directory `sources` and the thesis including LaTeX source code in directory `thesis`.

- `sources` - here you can find zip file with all sources:
  - `*src.zip` that contains:
    - `avalanche-sim.py` - main simulation script
    - `README.md` - notes and instructions
- `thesis` - contains PDF and sources of this thesis
  - `xsapak05_sources.zip` - LaTeX sources to compile this thesis
  - `xsapak05.pdf` - Portable Document Format of this thesis