7-2024

Decentralized consensus and governance for collaborative intelligence

Huiwen LIU

DECENTRALIZED CONSENSUS AND GOVERNANCE FOR COLLABORATIVE INTELLIGENCE

LIU HUIWEN

SINGAPORE MANAGEMENT UNIVERSITY
2024

Decentralized Consensus and Governance for Collaborative Intelligence

Huiwen Liu

Submitted to School of Computing and Information Systems
in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy in Computer Science

Dissertation Committee:

Feida Zhu (Supervisor / Chair)
Associate Professor of Computer Science
Singapore Management University

Jun Sun
Professor of Computer Science
Singapore Management University

Paul GRIFFIN
Associate Professor of Information Systems
Singapore Management University

Qiang Tang (External Member)
Senior Lecturer of Computer Science
The University of Sydney

Singapore Management University
2024

I hereby declare that this PhD dissertation is my original work
and it has been written by me in its entirety.
I have duly acknowledged all the sources of information
which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree
in any university previously.

Huiwen Liu

Huiwen Liu
11 July 2024

Abstract

The data economy today is becoming increasingly collaborative in nature. Take business intelligence, for example. To unleash the full potential of big data, it is essential to integrate multi-source data depicting entities from a multi-faceted and multi-modal perspective, which, not surprisingly, is not achievable by any company alone. In collaborative intelligence, there are two core issues, namely "trust" and "incentive". The core mechanisms to solve these two problems are consensus and tokenization separately.

To solve the trust problem, we propose a systematic consensus evaluation framework to investigate whether existing consensus algorithms can do so. After a lot of research and evaluation, new challenges arise accordingly in achieving both correct model training and fair reward allocation with collective effort among all participating nodes, especially with the threat of the Byzantine node jeopardizing both tasks. So, we propose a blockchain-based decentralized Byzantine fault-tolerant federated learning framework based on a novel Proof-of-Data (PoD) consensus protocol to resolve both the "trust" and "incentive" components. By decoupling model training and contribution accounting, PoD can enjoy not only the benefit of learning efficiency and system liveliness from asynchronous societal-scale PoW-style learning but also the finality of consensus and reward allocation from epoch-based BFT-style voting. To mitigate false reward claims by data forgery from Byzantine attacks, a privacy-aware data verification and contribution-based reward allocation mechanism is designed to complete the framework.

To solve the incentive problem, we investigated the mainstream DAO governance system from the tokenization governance perspective and proposed a four-layer governance evaluation framework. Through the core decentralized analysis, we aimed to illustrate the challenges of building a truly decentralized governance system and prepare for the design of a relatively perfect tokenization governance system.

Contents

Acknowledgments

My deepest gratitude goes first and foremost to Professor Feida Zhu, my supervisor, for his constant encouragement and guidance. He has walked me through all the stages of the writing of this thesis. Without his consistent and illuminating instruction, this thesis could not have reached its present form.

Second, I would like to express my heartfelt gratitude to the committee members who have contributed their time, thoughts and skills to this thesis. I am also greatly indebted to the programme administrators, who have instructed and helped me a lot in the past years.

Last my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I also owe my sincere gratitude to my friends and my fellow classmates who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the thesis.

Chapter 1

Background and Motivation

1.1 Collaborative Intelligence

The term collaborative intelligence refers to collaborative human-AI systems that leverage the different attributes and strengths of each agent to achieve further improvements in work outcomes [79]. It is widely acknowledged that the data economy today is becoming increasingly collaborative in nature. Take business intelligence, for example. To unleash the full potential of big data, it is essential to integrate multi-source data depicting entities from a multi-faceted and multi-modal perspective, which, not surprisingly, is not achievable single-handedly by any company alone. It is mutually beneficial for companies to leverage each other's data for collective model training. Specifically, in commercial applications, banks need to know their customers' consumption patterns to serve them better. Meanwhile, tel companies also need to know the history of enquiries. When data flows between banks and telcos, information is exchanged to better understand customer needs. The same is true in multiparty settings.

Although data, dubbed as "new oil", is increasingly important for all businesses in almost all industries, private and public sectors alike. However, most businesses do not have sufficient data they need within their own ecosystem, but data from different communities across different silos can complement one another to unleash the true power of big data with collaborative intelligence.

There are three elements of intelligence: data, model and computational power. Among them, data is the foundation and focus of this research. The two critical

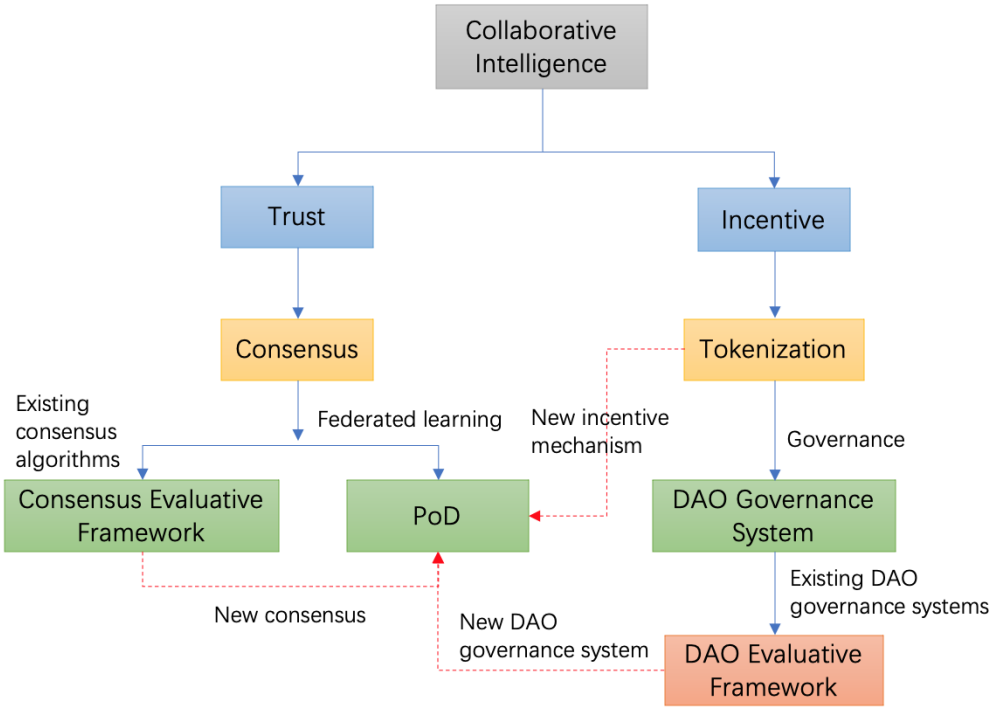


Figure 1.1: Reaching structure

enabling components of collaborative intelligence are trust and incentive. Trust and incentive are both necessary. If there is only trust, it is impossible for people to collaborate because there is no benefit, and if there is only incentive, it will not work, and even the whole data ecosystem would be destroyed. So, both trust and incentive are necessary and indispensable.

To achieve trust and incentive, we encounter some challenges as follows:

- The most important and prevalent setting is a decentralized one where no trusted central entities exist to coordinate and motivate.
- Besides the traditional attack models, Byzantine nodes are more concerned with attacks aiming for unwarranted benefits gained inappropriately for data contribution.
- Data privacy must be incorporated throughout the entire design and Incentive mechanism design is indispensable for a sustainable and lively ecosystem

So, if we want to design a solution to solve the challenges discussed above, it must contain several important components. The first is the consensus protocol, which is used to reach an agreement for decision-making in model training, profit attribution and allocation. The second is distributed ledger, which is used to record all transactional and operational data in an immutable way. The third is cryptography, which is used to verify data quality and privacy by design. The last is tokenomics, which is used to incentivize various parties in the ecosystem. So, our approach incorporates all of these components, and Figure 1.1 shows our reaching structure, which contains all of these components, next, I will introduce the crucial backgrounds of my research.

1.2 Decentralized Federated Learning

Privacy and security concerns have long been major roadblocks in cross-entity data exchange. Among all the approaches proposed to resolve these data silo issues, federated learning [49] has gained growing popularity due to the fact that participating training nodes can retain all their data on-premise, train models locally and exchange only model parameter information to cooperatively obtain a common model better than what each can individually train, maximally protecting their data privacy and security. This is the learning environment of collaborative intelligence we will focus on in this thesis.

Unfortunately, existing federated learning models focus mostly on settings with a central entity to coordinate all other nodes in the training process, which we refer to as the "centralized" setting. Specifically, geo-distributed devices contribute to private data directly, and the central servers aggregate massive information from edge devices and store it in their own database, as shown in Figure 1.2. They then use this information to train specific machine learning models with large storage capability and computation performance. Although in some collaborative scenarios, some encryption methods have been used to protect user privacy, this direct sharing of private data still poses a serious privacy problem because, in this mode, all the private data of users are open to the central servers or other third parties, which can reveal sensitive information of users such as address information, health information, shopping hobbies and so on.

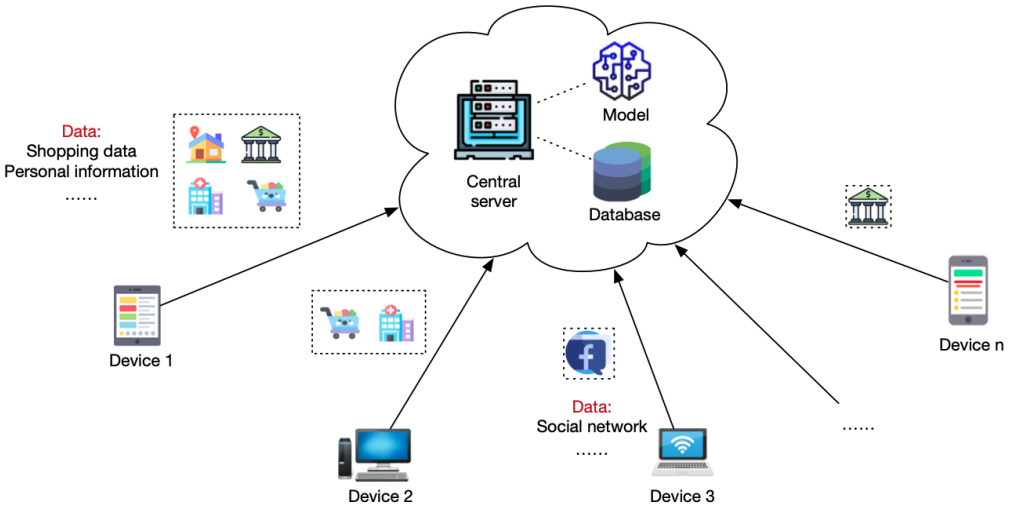


Figure 1.2: Traditional machine learning

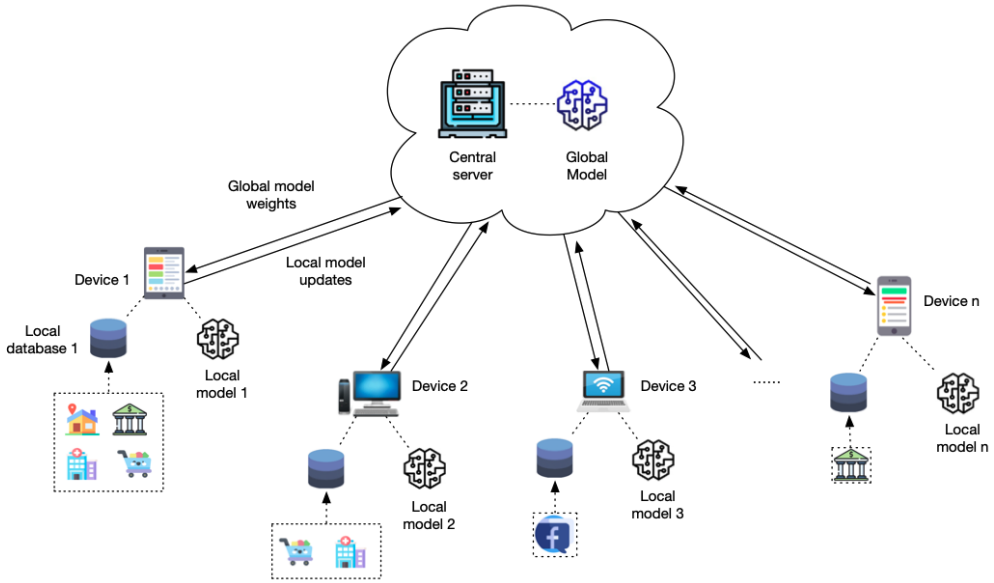


Figure 1.3: Federated learning

However, due to these devices' growing computational power and concerns over transmitting private information, it is increasingly attractive to store data locally and push network computation to the edge, making training transfer possible, as shown in Figure 1.3. This new intelligent collaboration mode is called federated learning, which is a type of model collaboration that stores data locally and pushes network computation to edge devices. In the federated learning model, each device

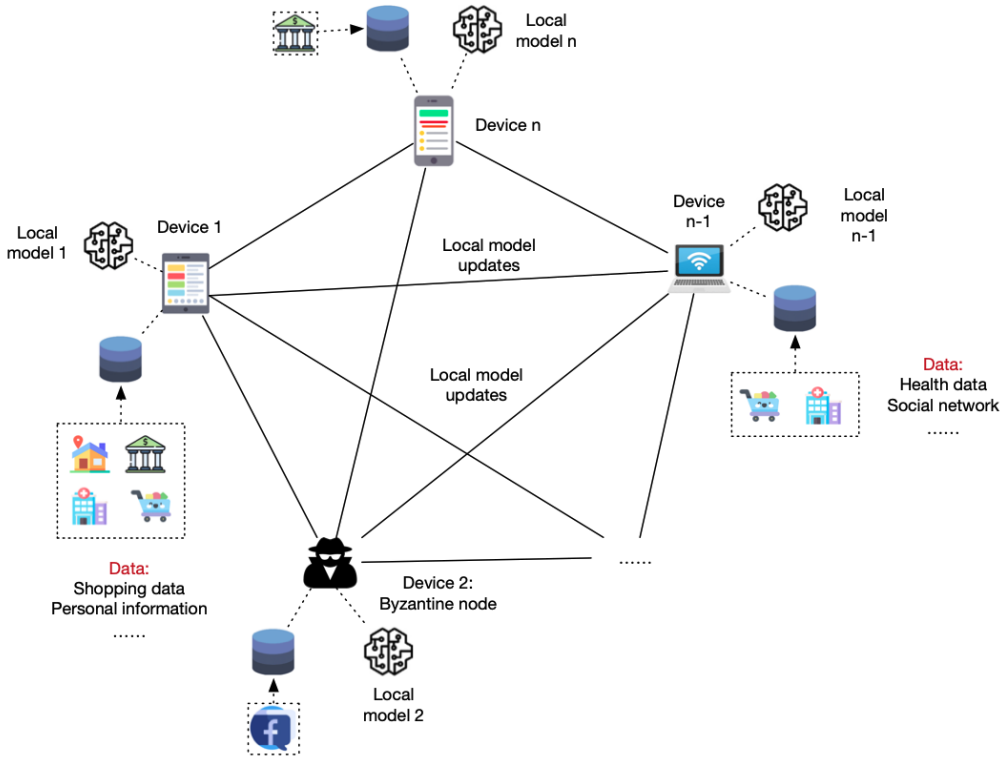


Figure 1.4: Decentralized Byzantine fault-tolerant federated learning

trains the specific machine learning model locally with their local private dataset and just needs to transmit because each device just needs to transform the local model weights to the central server. The central server aggregates the local model weights and generates the global model weights with specific protocols. It doesn't get any private data from the local device. In this mode, all local devices can effectively protect their own private datasets from exposing to the public.

However, this centralized federated learning mode has a fatal drawback. If there is a problem with the central server, the whole federated learning system will be affected. Therefore, researchers tried to design a federated learning mode with no central server, and decentralized federated learning [39, 47, 60, 57] then came into view as shown in Figure 1.4.

In decentralized federated learning system, all edge devices train the local model with their own private dataset and broadcast the local model weights through the P2P network. Because there is no central server, so the main task of the DFL is to design consensus protocol to make the all locally models keep same.

Intuitively, decentralized federated learning system has a global model, and each device in the system has a local view of the model, so all the devices must roughly agree on what the global model is. The solution to solve the problem is consensus protocol, which serves to make sure a valid agreement is reached continuously among a group of decentralized devices. As we have discussed, there is no system free of faults, especially Byzantine faults in an open-access system running environment. So, we also need to consider the attacks from the Byzantine nodes.

1.3 Consensus protocols

Consensus protocol is not a novel research topic and has been studied for decades with the advent of distributed systems. The earliest consensus protocol specifies how to make all active nodes agree on a value in a distributed system. In this period, an algorithm achieves consensus if it satisfies the following two conditions simultaneously related to resilience [70, 71]:

- **Agreement.** all non-faulty processes decide on the same output value.
- **Termination.** all non-faulty processes eventually decide on some output value.

It 's worth noting that consensus algorithms at this period do not take the benign fault into consideration in which all processes are considered honest and don't behavior maliciously.

1.3.1 Extended failure model

The underlying model is a complicated environment on which the consensus algorithm runs. Algorithms that solve consensus greatly depend on the assumptions that are made about the system. The more general the underlying running environment, the higher the applicability of the consensus algorithm. The underlying model mainly includes two parts. The first is the network reliability assumption. In a reliable message transmission network, all messages are eventually delivered intact exactly once. The second is timing assumption. In a synchronous [26] tim-

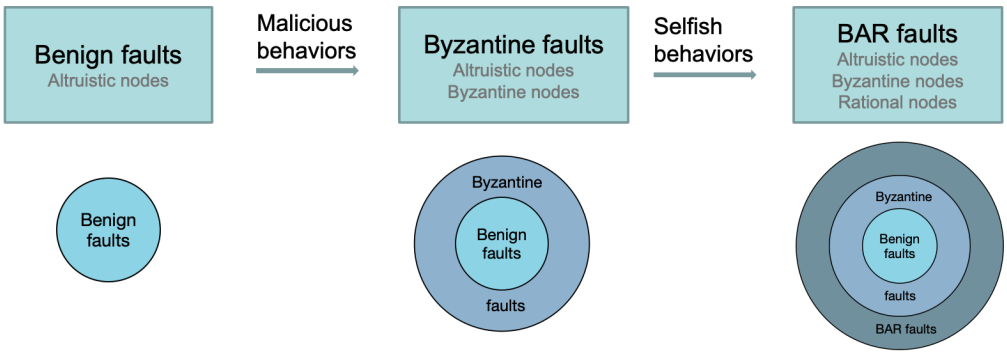


Figure 1.5: Extended failure model

ing model, communicating message delays and process delays are bounded, which enables communication in synchronous rounds.

But it's impossible to have a system free of faults, and the failure model mainly specifies the faults that the consensus can tolerate. In the original distributed system, consensus only considered the fail-stop faults [70], in which faulty nodes stop and do not send messages. But with the practical application of consensus algorithms, a new type of fault is taken into consideration in the consensus design-Byzantine faults [54], in which faulty nodes may send arbitrary messages. As consensus is applied to more and more practical problems, especially large-scale financial applications, attacks from malicious nodes become more complicated. Regardless of Byzantine and altruistic nodes in the system, a new kind of node emerges, rational nodes [1]; they are selfish and driven by benefits and only follow the protocol if it suits them.

So, it is necessary for us to extend the definition of the failure model to accommodate more and more complicated applicative scenarios, as shown in Figure 1.5. Specifically, there are three types of nodes in the BAR fault-tolerant system:

- `Byzantine`: Byzantine nodes aim to harm the system with malicious behaviour all the time.
- `Altruistic`: Honest nodes always follow the protocol.
- `Rational`: Rational nodes only follow the protocol if it benefits them.

In the early, researchers focus on benign faults, which means there is no malicious behaviors in the system, mainly including crash faults and omission faults.

Driven by the applications, benign fault-tolerant system can not meet the practical requirements. Researchers began to think about malicious behaviors of the nodes, thus the Byzantine fault-tolerant consensus algorithms were born. In this stage, only general malicious behaviors of Byzantine nodes are delt with. With the advent of Bitcoin [65], a new kind of node emerged: rational nodes; they are selfish and driven by benefits and only follow the protocol if it suits them. So, they tried their best to design specific attacking models to get more benefits. With the emergence of rational nodes, distributed consensus requires more complex user behaviours to be considered, and distributed systems become more complex.

1.3.2 FLP impossibility result

Work [32] has demonstrated that it is impossible to have a deterministic protocol that solves consensus in a message-passing asynchronous system in which, at most, one process may fail by crashing. In other words, asynchrony, determinism and fault-tolerance are incompatible with a consensus protocol. However, for a practical DLT-based system, fault tolerance is necessary. So, researchers focus on the perspective of asynchrony and determinism to overcome the FLP impossibility result, Including the synchronous assumptions, randomness and hybrid methods, which combine synchronous assumptions and randomness. These three methods represent the exploration of consensus protocols at different levels of system complexity.

1.3.3 Four stages of consensus development

Picture1.6 shows the timeline of consensus development. Firstly, I want to emphasize that the distributed consensus is not a new problem and has been studied for decades. In 1980, the first fault-tolerant consensus [71] was proposed, but it can only tolerate benign faults. In 1982, Lamport proposed the Byzantine generals problem [54], and then a new branch of the Byzantine fault-tolerant consensus emerged. However, constrained by the FLP impossibility result, the performance of early Byzantine fault-tolerant algorithms is generally poor. Although the proposed PBFT [12] broke through the limitation of practical application, it is difficult to form large-scale commercial applications. It was not until 2008 that this dilemma was finally broken. Nakamoto [65] creatively proposed the proof of work in

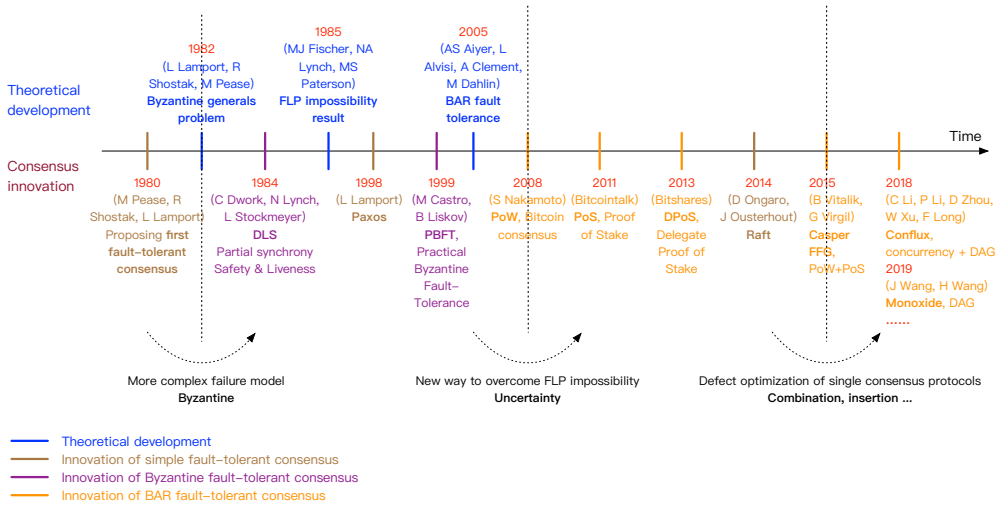


Figure 1.6: Timeline of consensus development

Bitcoin, which used new way randomness to overcome FLP impossibility and even tolerate BAR faults. PoW liberates the performance bottleneck of the traditional distributed consensus and makes large-scale commercial applications possible, but the introduction of uncertainty also leads to the problem of probabilistic finality. In addition, the huge consumption of electricity is also a problem. So, researchers began to design new replacements, such as PoS [5], Tendermint [50, 10] researchers tried to optimize the single consensus protocols with combination, insertion and other technologies. Casper FFG of Ethereum [11] was one of the early experiments, followed by Conflux [56], Monoxide [87] and so on, based on different goals and forming new tradeoffs.

According to the different levels of the system fault-tolerance and ways to overcome the FLP impossibility result, we can divide the consensus development into four stages. In the first stage, the consensus protocols serve as benign fault-tolerant systems, which mainly use timeouts to overcome the FLP impossibility result, and the algorithms only need to meet the agreement and termination to achieve consensus. Compared to the first stage, the consensus in the second stage can tolerate more complex Byzantine faults, which exist in malicious behaviors. And consensus for Byzantine fault-tolerant systems must meet safety and liveness simultaneously. For safety, regardless of the agreement, we also need to consider the validity of the consensus results. With the birth of Bitcoin, we have entered

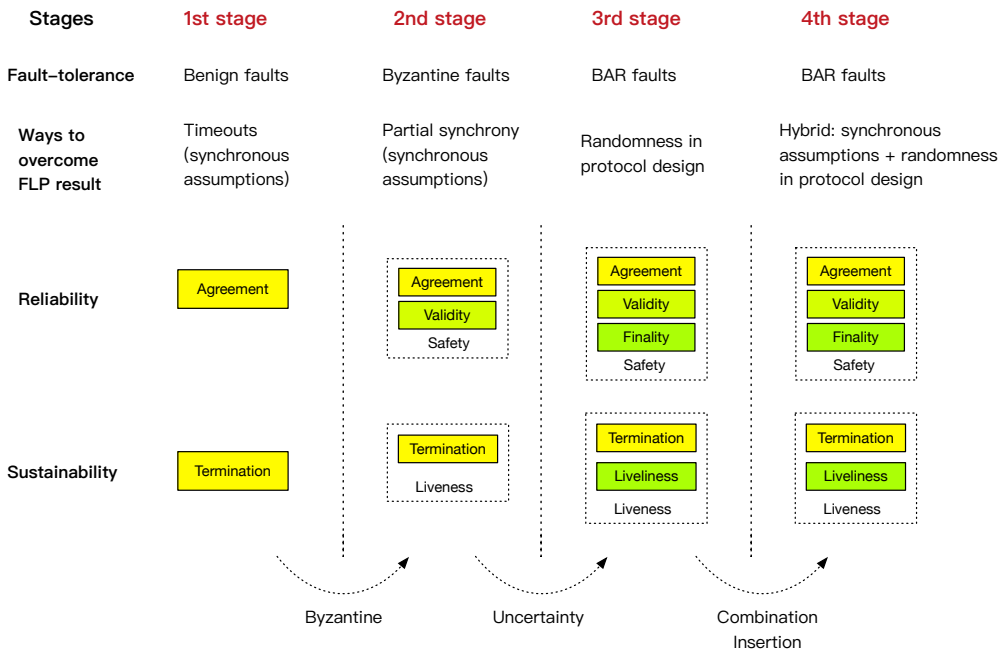


Figure 1.7: Four stages of consensus development.

a new era of using randomness to overcome the FLP impossibility result, which adds uncertainty to the consensus process. In this stage, we consider the more complex BAR fault, which exists in rational nodes that only follow the protocols if it suits them in the system. The addition of uncertainty makes us need to consider the consensus results' finality and the consensus persistence's liveliness. However, consensus algorithms have many drawbacks at this stage. So, researchers try to combine the consensus in the previous three stages to design more resilient, efficient and lively consensus algorithms. At present, the development of consensus is in the fourth stage.

1.4 Decentralized governance and MakerDAO

As Bitcoin was proposed in 2008 [65], blockchain has deeply changed financial markets. Various debates evolve around the potential democratization of financial services [82], blockchain competition and services improvement, and investment opportunities in new tokenized assets. It is widely accepted, though, that the prominent disruption lies in the disintermediation of financial institutions from their

centralized role. The absence of centralized third parties, e.g., central banks, in the blockchain universe and circumventing traditional barriers to financial markets' participation are the major attributes of this market revolution. There is no or limited role for a central authority. Such decentralized frameworks theoretically allow all participants to be part of prominent decision-making and spread risk among each other. Decentralization, therefore, is logically regarded as the core value proposition of blockchain.

Decentralized Finance (DeFi) is a blockchain-based financial application designed to replicate most economic activities in traditional markets, e.g., lending and borrowing. Theoretically, governance in DeFi is decentralized since all members are decision-makers. In conventional finance, governance is inevitably centralized, which can be the origin of several problems. The most intractable issue is probably the agency problem, where the owners and managers of an organization have different interests. Managers can pursue their own profits at the expense of the owners' benefits [30]. Therefore, the most challenging objective of governance is to align the interests between owners and managers. As discussed by Lee [55], the decentralized nature of blockchain brings forward the idea of a 'token economy', where capital is better directed to those users actually contributing with content and services. Within the DeFi context, owners and managers are theoretically identical, which creates an opportunity to investigate the premises of this debate once again.

Stemming from this background, evaluating how efficient DeFi is a very crucial task. Decentralized Autonomous Organization (DAO) is one popular solution to decentralized governance and decision-making. In a DAO, all members are the owners of the organization, and they have decision-making power to the development of it. Usually, the suggested changes will be written in the form of an Improvement Proposal (IP), which is then voted through an established poll, where all members can make public their options. DAO members state their options through governance tokens. Usually, these governance tokens are also tradable cryptocurrencies. The votes are weighted by the amount of governance tokens held by voters. In other words, governance in DAO is tokenized. Currently, DAO is one of the most common governance mechanisms adopted by DeFi [82]. Technically, DAO can be deployed on blockchain, and currently, most DAOs rely on Ethereum [91], which is a programmable blockchain. The core of DAO governance is based on standard

smart contract codes instead of human actors. In other words, DAO's governance is tokenized. In practice, DAO-based protocols usually have their own governance token and governance token holders can vote on changes to the protocols.

MakerDAO [27] is one of the most influential DAOs based on the Ethereum DeFi platform. It develops and manages the Maker protocol. An innovative selling point of the Maker protocol is decentralized governance. In the Maker protocol, governance can be divided into two parts: on-chain governance and off-chain governance. In on-chain governance, there are two types of votes, namely the Governance Polls and Executive Votes. Any MKR holders can vote using the Maker Protocol's on-chain governance system. Governance polls, which are about non-technical changes, measure the sentiment of MKR holders. Executive votes "execute" technical changes to the protocol. The voting results are documented on blockchain. Off-chain governance is mainly about informal discussion, e.g., discussion on the MakerDAO forum. Both MKR holders and the larger community can express their opinions. The amount of MKR weights voting power that a voter owns and represents, making the voting mechanism a token-weighted one [85]. One MKR equals to one vote and the option with the largest votes wins. In the Maker protocol, Maker IPs are structured and formalized for a voting event, and key issues and changes to the system are rigidly defined in Maker IPs. Usually, the Maker Foundation will draft the initial Maker IPs, and any community members can propose competing IPs. Then, the final decisions will be made by MKR voters through the current Maker governance process. MKR holders can be voters and directly choose their options on the Maker Governance Portal. On the other hand, they can select a Vote Delegate to be their representative. As a result, delegates gain voting power from MKR holders, and these MKR holders can indirectly vote. The voting results are weighted by the amount of MKRs who voted for a proposal.

By examining governance polls in the Maker protocol, we observe signals of centralized Maker governance. Compared with the rapidly increasing number of users, voters are centralized in a small group of members, and the most dominant voters are heterogeneous in characteristics. The unevenly distributed voting power, as a preliminary signal of governance centralization, leads to our measurements of governance centralization in Maker protocol.

1.5 Contribution

Two main challenges arise in the decentralized setting with Byzantine fault tolerance in collaborative intelligence: (I) How to collectively achieve the correct model training with comparable results as in centralized federated learning, and (II) How to design fair incentivization to properly reward participating training nodes for their contribution in terms of data. The most difficult part for both challenges is to handle the "Byzantine" nodes that are largely ignored in existing studies for centralized setting. It is worth noting that in our case, Byzantine nodes not only refer to malicious nodes in traditional consensus research — those attacking the system by compromising the consensus, which is the main task of challenge I — but also to nodes harmful to the system by scheming for unwarranted value allocation from their data contribution, the main task of challenge II.

To overcome these challenges, we first propose a systematic consensus evaluation framework to investigate whether existing consensus algorithms can solve the trust problem in the above setting. After a lot of research and evaluation, new challenges arise accordingly in achieving both correct model training and fair reward allocation with collective effort among all participating nodes, especially with the threat of the Byzantine node jeopardising both tasks. So, we propose a decentralized Byzantine fault-tolerant federated learning framework incorporating distributed ledger technologies with a novel partially asynchronous consensus called *Proof-of-Data* (i.e., PoD). To solve the incentive problem, we investigated the mainstream DAO governance system from the tokenization governance perspective and proposed a four-layer governance evaluation framework. The contributions of this thesis can be summarized as follows:

- We propose a general consensus evaluative framework from three perspective: resilience, performance and governance. Through the analysis of the consensus algorithms, we can provide useful reference for the designers so that they can choose the most appropriate existing algorithm or design a new efficient consensus algorithm for the new distributed ledger system.

- We propose a novel partially asynchronous consensus protocol called Proof-of-Data (i.e., PoD) tailored for decentralized federated learning settings with

Byzantine fault tolerance. PoD combines Proof-of-Work (i.e., PoW) style asynchronous consensus with epoch-based consensus locking enabled by BFT-style component, taking the best of both by endowing, on one hand, the missing consensus finality to the practically robust yet theoretically-flawed PoW consensus and lending, on the other hand, the scalability necessary for societal-scale application setting to the otherwise sound PBFT consensus.

- Finally, we propose a general evaluative framework for DAO governance of tokenization based on decentralization degree distribution. Through the core decentralized analysis, we aimed to illustrate the challenges of building a truly decentralized governance system and prepare for the design of a relatively perfect tokenization governance system.

1.6 Thesis Synopsis

The rest of this thesis is organized as follows. We introduce the consensus evaluative framework in chapter 2. Chapter 3 describes the proposed consensus algorithm PoD and analyzes the resilience, governance, and anti-attacks of the PoD with the proposed evaluative framework. Next, we introduce the general DAO governance evaluative framework for the tokenization system in the chapter4. Finally, we conclude the thesis in chapter 5.

Chapter 2

Evaluation Framework for Consensus in DLT

2.1 Introduction

Distributed ledger technology (DLT), initially fueled by the boom of Bitcoin [66], has started to attract increasing attention due to its great promise and potential for massive applications where a trusted third party is no longer needed. DLT distinguishes itself from traditional distributed databases with a decentralized, immutable and self-organized ledger system, which hinges upon its most important technical component — the consensus algorithm. A consensus algorithm serves to make sure a valid agreement is reached among a group of mutually distrusted nodes continuously. Widely studied by academia and industry for closed distributed systems, this technology's application to open-access distributed ledger systems has revitalized the field and led to a plethora of new designs. The groundbreaking feature of DLT is that the ledger is not maintained by any central authority. Multiple untrusted or semi-untrusted parties can directly and transparently interact with each other without the presence of a trusted intermediary. While the projected capabilities and value of the DLT might seem overly optimistic, its key properties of decentralization, integrity, resilience, and transparency make it an attractive option for lots of applications, including the financial industry, voting, publication records, social media and so on.

The major hurdle in the widespread adoption of the DLT is its inherent properties, mainly including resilience, performance and governance. While lots of improvements

have been made, they are nowhere near as ubiquitous as their traditional counterparts. We are at a crucial point in the evolution of the DLT, and its properties are deeply related to the consensus protocol, which maintains the consistent state of the whole distributed ledger system. Consensus protocol is the core component of the distributed ledger system, and we believe that this is where future work to improve the resilience, performance, and governance of DLT should be concentrated.

The earliest consensus protocol specifies how to make all the active nodes agree on a value in a distributed system. The consensus algorithm of this period has two key properties related to resilience [71]: (i) *Agreement*-all non-faulty processes decide on the same output value; (ii) *Termination*-all non-faulty processes eventually decide on some output value. Consensus algorithms of this period only take the benign fault into consideration, in which all nodes are considered honest and don't behave maliciously.

With the practical application of consensus algorithms, a new type of fault is considered in the consensus design-Byzantine fault [54], with which malicious behaviours of Byzantine nodes are dealt with. Consensus algorithms of this period have three key properties related to resilience based on which its applicability and efficacy can be determined [26]: (i) *safety*-if an honest node accepted (or rejected) a value, then all other honest nodes in the system make the same decision; (ii) *liveness*-requests from correct clients are eventually processed; (iii) *fault tolerance*-system can successfully recover from failure of the nodes which participates in consensus. While all the above three properties are crucial, a famous result by Fischer, Lynch and Paterson [32] known as FLP Impossibility Result, states that no deterministic consensus protocol can guarantee safety, liveness and fault tolerance simultaneously in an asynchronous system. While fault tolerance is crucial for operating globally distributed networks, distributed systems tend to choose between safety and liveness depending on their system requirements and assumptions. The distributed systems community has extensively studied it for decades and proposed robust and practical protocols (e.g., Paxos [51, 52], PBFT [12]) that can tolerate faulty and malicious nodes. However, these protocols were designed for limited and closed nodes.

Bitcoin's fundamental innovation was to enable consensus among an open, unlimited and decentralized group of nodes, which is achieved via a leader election based on proof-of-work (PoW). In PoW consensus, all nodes compete to find the solution to

a hash puzzle, and the winning node adds the next block to the blockchain. Since the PoW's probabilistic leader selection process is combined with performance fluctuations in decentralized networks, Bitcoin offers only weak safety. Additionally, Bitcoin suffers from poor performance, which cannot be remedied without fundamental redesign [19], and the used PoW consensus protocol consumes a large amount of energy, especially power energy. These evident flaws have led to a plethora of proposals for new consensus protocols [3], some of which replace PoW with more energy-efficient alternatives (e.g., Proof-of-Stake [62], Proof-of-Authority [43]), while others modify the original design of Bitcoin (e.g., Bitcoin-NG [28]) for better performance. To achieve strong consistency and similar performance as mainstream payment processing systems like UnionPay and Visa, a number of recent proposals seek to repurpose classical consensus protocols (e.g., Paxos [52], Raft [69] and PBFT [12]) for usage satisfying Byzantine fault tolerance in decentralized blockchain [86, 50, 11].

So far, although there exist a few surveys based on selected systems, which we will discuss in the Related Work Section, there has been no systematic and comprehensive study of the consensus protocols from the inherent properties and internal processes. This incurs two major challenges. Firstly, a comprehensive survey of blockchains would doubtless include a discussion of classical consensus protocols. However, the literature is vast and complicated, which makes it hard to tailor it to the DLT. Moreover, conducting a survey of consensus protocols in DLT has its own difficulties. Though young, the field is characterised by high-volume, fast-paced work. Secondly, consensus itself is a complicated issue, and its internal processes contain several subprocesses. Based on this, the classification and evaluation of consensus requires a high degree of refinement and common extraction.

To fill these gaps, firstly, based on the inherent properties and applied properties of the consensus, we proposed a novel quantitatively evaluative framework to comprehensively analyse the resilience, performance and governance of the consensus protocols with various consensus and network assumptions, such as timing model, failure model and network configurations and data organization. The evaluative framework mainly includes two models: the underlying and evaluative models. The underlying model specifies the most general environmental assumptions on which one specific consensus can run, and the evaluative model analyzes the properties of the consensus algorithms specifically under the specific attacking models. Secondly,

we conduct a comprehensive survey mapping how consensus protocols have evolved from the classical distributed systems use case to their application to blockchains, and based on the survey, we propose a general process structure of the DLT-based consensus algorithms, which mainly includes four subprocesses, namely leader election, block generation, data validation and chain updation. Moreover, according to the different leader election algorithms, we classified the consensus into three categories. Last but not least, we use the proposed evaluative framework to analyse the properties of some typical DLT-based consensus protocols from these three categories, respectively.

2.2 Extended Consensus Definition

As discussed in section 1.3, according to the extended failure model, compared with the traditional consensus definition, an algorithm achieves consensus if it satisfies the following two conditions:

Safety:

- `Agreement`: all non-faulty processes decide on the same output value.
- `Validity`: the decided value must be one of the non-Byzantine inputs.
- `Finality`: once a consensus is recorded, it should be immutable.

Liveness:

- `Termination`: all non-faulty nodes eventually decide on some output value.
- `Liveliness`: all non-faulty nodes are available to decide new values continuously.

Firstly, with the introduction of Byzantine nodes, we need to consider the validity of the consensus result, which means the decided value must be one of the non-Byzantine inputs. Meanwhile, with the birth of Bitcoin and the development of distributed ledger technology, consensus has been applied to many social society applications. So, we also need to consider the preservation and traceability of consensus results for future access, and processors must decide the value continuously

round after round. Therefore, once a decided value is recorded in a distributed ledger, it should be immutable (i.e., finality), and all non-faulty nodes are available to decide new values continuously (i.e., liveliness).

2.3 Overview of Evaluative Framework

Consensus algorithms have been studies and analyzed for many years, but there has been no systematic and comprehensive study of the consensus protocols from both inherent and applied properties. In this section, we creatively propose a novel generally evaluative framework that we can leverage to quantify the mainstream consensus algorithms used in distributed ledger systems. Reaching consensus in a distributed system is actually a game process, and designing a distributed consensus algorithm is always about trade-offs. From a game-theoretic point of view, if you want to overcome a limitation, you must make a sacrifice somewhere else. A completely perfect consensus algorithm which satisfies all requirements of designers has yet to emerge. Therefore, based on the proposed evaluative framework, we then quantitatively evaluate three typical series of consensus algorithms: consensus with BFT-style SMR (e.g., PBFT [12]), consensus with PoX-style SMR (e.g., PoW [66], PoS [76]) and consensus with Hybrid-style SMR (e.g., Tendermint [9], Casper FFG [11]). Specifically, we will systematically compare these consensus algorithms, and analyze the trade-offs of the internal factors of these algorithms. Our ultimate goal of the paper is to provide the distributed ledger system designers with a metric that allows them to choose the most appropriate and efficient consensus algothem for their new system by integrating our evaluative framework with their system capabilities, because different use cases naturally call for different designs.

Figure 2.1 shows the overview of our evaluative framework. Our framework mainly consists of two components, including underlying model and evaluative model. Firstly, consensus is a superstructure based on the system assumptions (i.e., underlying model), so they can vary much depending on the assumptions made about the system. In other words, for the same consensus algorithm, different system assumptions may result in different consensus results. System assumptions are not only premises but also purposes. But in all system assumptions, there always exists a most general one which fits the consensus algorithm and can make the consensus

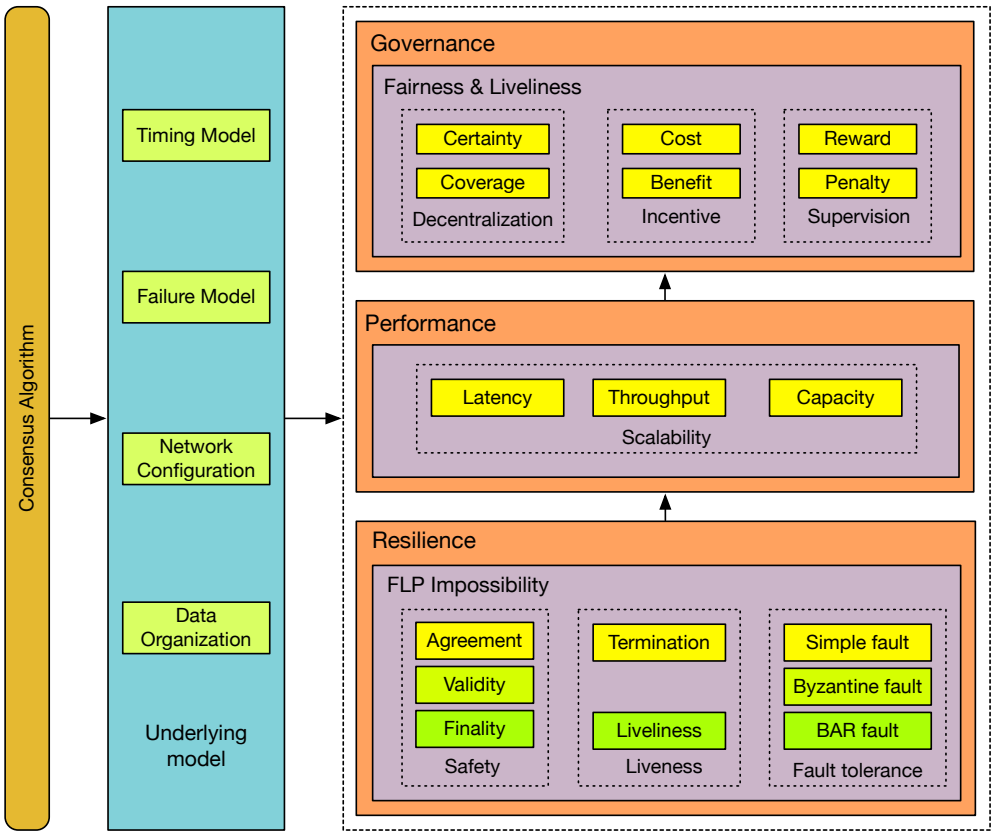


Figure 2.1: General consensus evaluative framework.

work best. Therefore, we firstly model the most general system assumptions for the specific consensus algorithm which the consensus will run on, specifically including the timing model, failure model, network configuration and data organization.

Next, based on modeling system assumptions, we will systematically evaluate the specific consensus algorithm from three aspects, including resilience, performance and governance. The traditional consensus algorithms mainly focus on distributed computing and are considered as a computational problem. So, the studies concentrate on resilience, mainly considering the safety, liveness and failure tolerance of the consensus algorithm because resilience is the fundamental problem of the distributed consensus. From the perspective of traditional distributed computing, resilience is mainly used to analyze the feasibility of the consensus algorithm. However, constraint to the “FLP impossibility result" [32], any consensus algorithm needs to strike a balance among the safety, liveness and failure tolerance. Meanwhile, for traditional distributed computing, once the consensus result is written to the database,

the consensus is considered finished and the result cannot be changed. But for DLT-based consensus, because of the probability of invertibility of the transactions in the distributed ledger (i.e., blockchain), we also need to consider the finality of consensus results. In other words, take the immutability of the distributed ledger into the consideration when we evaluate the resilience of the consensus algorithms.

Bitcoin creatively applied distributed consensus to a practical application for the first time and solved a commercial problem, which involves a new governance problem. In contrast to the traditional distributed consensus, governance is a unique property of the DLT-based consensus algorithms different from the traditional counterparts. The governance mainly focuses on the incentive mechanism that how to motivate more honest parties to participate in the consensus process to maintain maximum decentralization (i.e., rewards), and how to resist any form of attacking behaviours (i.e., penalties). The purpose of the governance is to coordinate the reward and penalty mechanisms to maximize the fairness of the consensus algorithms. Only in this way can the distributed ledger system develop smoothly and healthily.

At the same time, traditional consensus algorithms were designed for limited and closed nodes, therefore, researchers don't concern more about the performance of the consensus algorithms. But after creation of the Bitcoin, performance is significant for large-scale applications of emerging consensus algorithms in open-access network. Performance is mainly focused on the scalability of the consensus algorithm from the perspective of the large-scale industrial application. Scalability is mainly about two aspects, namely number of nodes N_N and number of the transactions N_{T_x}, and mainly to analyze the influence of these two factors on the system performance. Intuitively, we quantitatively use the latency and throughput to measure the scalability of every specific consensus algorithm by considering different network configurations and data organizations.

Last but not least, according different attacking models, we will analyze the anti-attacking of each consensus algorithms. Anti-attacking is the direct embodiment of the robustness of the consensus algorithm. Finally, through the analysis of the consensus algorithms, we can provide useful reference for the designers so that they can choose the most appropriate existing algorithm or design a new efficient consensus algorithm for the new distributed ledger system.

2.4 Underlying Model

The distributed ledger system is actually a distributed system, and the underlying model is a complex environment on which the consensus algorithm runs. Algorithms that solve consensus vary much depending on the assumptions that are made about the system. The more general the underlying running environment, the higher the applicability of the conensus algorithm. In this subsection, we will elaborate the underlying running enviroment in the DLT-based distributed system.

2.4.1 Timing Model

In a distributed system, all spatially independent nodes communicate and coordinate by "message passing" between one or more other nodes. So it is very important to determine the timing of the messages. However, even if the clocks on all network nodes in a DLT-based distributed system are set to the same initially, their clocks will eventually vary quitely significantly due to "clock drift" (a phenomenon in which clocks count time at slightly different rates) unless corrections are applied. In other words, there is no single global clock that determines the sequence of events happening across all nodes in the distributed P2P network.

In a distributed system, it is hard to set limits on the time that can be taken for message delivery across the network. Two opposing extreme positions provide a pair of single models - the first has a strong assumptions of time and the second makes no assumptions about time. Therefore, according to the time cost assumptions of message delivery in the distributed system, we can divide the message-passing evironments into three categories:

- **Synchrony.** Messages will be delivered within some fixed, known amount of time.
- **Patial synchrony.** A consensus protocol guarantees liveness if all non-faulty nodes participating in consensus eventually produce a value.
- **Asynchrony.** A consensus protocol guarantees liveness if all non-faulty nodes participating in consensus eventually produce a value.

Therefore, there are three types of timing models to consider when making a DLT-based distributed system: 1) **Synchronous model.** In a synchronous system, it is assumed that messages will be delivered within some fixed, known bound Δ of time. Synchromous message passing is conceptually less complex because users can guarantee that when they send a message, the receiving node will get it within a certain time frame (i.e., Δ), which allows users to model their consensus algorithm with a fixed upper bound of how long the message will take reach its destination. However, this type of environment is not very practical in a real-world distributed system where nodes can crash or go offline and messages can be dropped, duplicated, delayed or received out of order. 2) **Asynchronous model.** The asynchronous model allows no assumptions about the time intervals involved in any execution. In an asynchronous message-passing system, it is assumed that a network may delay messages infinitely, duplicate them, or deliver them out of order. In other words, there is no fixed upper bound on how long a message will take to be received. Any solution that is valid for an asynchronous distributed system is also valid for a synchronous one. 3) **Patially synchronous model.** Synchrony and asynchrony are the two extremes of the timing model. The work [26] creatively introduced the concept of partial synchrony, which lies between the cases of a synchronous system and an asynchronous system. Message passing with partial synchrony assumes that fixed bound Δ of time exists, but it is not known a priori. Regardless of the actual values of the bound Δ, consensus can work correctly in the partially synchronous system.

2.4.2 Failure Model

It's impossible to have a distributed system free of faults. Real systems are subject to a number of possible flaws or defects, such as processes crashing, messages being lost, distorted, or duplicated, or even a process going completely haywire and sending messages according to some malevolent plan6 These failures can be broadly classified into three categories:

- **Crash-fail.** The process stops working without warning (e.g., the computer crashes);

- **Omission.** The process sends a message but it is not received by the other nodes (e.g., the message is dropped);
- **Byzantine.** The process behaves arbitrarily. This type of fault is irrelevant in controlled environments (e.g., the permissioned blockchain system) where there is presumably no malicious behavior. Instead, these faults occur in what's known as an "adversarial context" (e.g., the permissionless blockchain system).

Basically, in a permissionless distributed ledger system, when a decentralized set of independent users serve as nodes in the distributed network, these users may choose to act in a "Byzantine" manner to maximize profits, which means they maliciously choose to alter, block, or not send messages at all. With this in mind, the aim is to design consensus algorithms that allow a system with faulty nodes to still achieve agreement and provide a useful service. Given that each distributed system has faults, a crucial we must make when designing a consensus algorithm is whether it can survive even when the nodes in the system deviate from normal behaviours, whether that's due to non-malicious behaviours (i.e., crash-fail or omission faults) or malicious behaviours (i.e., Byzantine faults). Therefore, there are two types of failure models to consider when making a DLT-based distributed system: 1) **simple fault-tolerance.** In a simple fault-tolerant system, we assume that all nodes in the system do one of two things: they either follow the protocol exactly or fail. This type of system should definitely be able to handle nodes going offline or failing. But it doesn't have to worry about nodes exhibiting arbitrary or malicious behaviours. 2) **Byzantine fault-tolerance.** A simple fault-tolerant distributed system is not very practical in an uncontrolled environment. In a decentralized system which has nodes controlled by independent users communicating on the open, permissionless network, we also need to design for nodes that intend to do malicious or "Byzantine [54]" behaviours. Overall, in a Byzantine fault-tolerant distributed system, we assume that nodes can fail by accident or do malicious behaviour intentionally.

For distributed ledger systems, many Byzantine nodes have developed various attacking models to gain benefits. Some of these attacks are even deadly, which leads directly to the failure of the system and the inability to provide services anymore. Among these attacking models, some are universal and can attack the majority of

the consensus algorithms for the distributed ledger systems, while some are targeted at the logic or design vulnerability of some kind of consensus protocol. Therefore, when we evaluate a specific consensus algorithm, we need to take the anti-attacking into consideration. In the context of the attacking models, then we can anylize the specific characteristics of the consensus algorithms.

Generic attacks. 1) *Double-spending:* using the same coins to issue two (or more) transactions effectively spending more coins that he possesses. Work in [44] has shown that accepting transactions without requiring blockchain confirmations is insecure. The more confirmations a transaction obtains, the less likely this transaction will be reversed in the future. 2) *Selfish-mining:* miners might attempt to perform selfish mining [29] attacks in order to increase their relative mining share in the blockchain by selectively withholding mined blocks and only gradually publishing them [29, 77]. Recent studies show that, as a result of these attacks, a selfish miner equipped with originally 33% mining power can effectively earn 50% of the mining power. 3) *Eclipse attacks:* an adversary creates a logical partition in the network, i.e., provides contradicting block and transaction information to different blockchain network nodes [37, 40]. Double-spending attacks and selfish mining can be alleviated if all nodes in the blockchain system are tightly synchronised. Note that, in addition to network latency, synchronisation delays can be aggravated due to eclipse attacks.

Targeted attacking models. In addition to the generic attacking models, attackers also use user logic and design vulnerabilities to develop targeted attacking models against specific consensus protocols. For instance, the *nothing-at-stake* and *long-range* attack of the PoS-based consensus algorithms. So before we evaluate a consensus protocol, we also need to consider the targeted attacking models of this consensus and then analyze the robustness of the consensus against these targeted attacks and the impact of these attacks on other characteristics of the consensus algorithm.

2.4.3 Network Configuration

In a Byzantine environment, the identity management mechanism plays a key role in determining the organization of the nodes in a blockchain network.

Overlay P2P protocols. The network is the medium of message passing, and the network protocols provide the means of P2P network organization, namely peer route discovery and maintenance as well as encrypted data transmission/synchronization over P2P connections. In the distributed ledger system, the identity management mechanism plays a key role in determining how the nodes in the network are organized and how they communicate. In an open-access (i.e., public/permissionless) DLT-based network, all nodes can freely join and leave the network and activate any available network functionalities. Notice that "node" in the distributed ledger system refers to the logical entity (i.e., the identity of a system user) rather than a physical device because multiple "nodes" associated with different network functionalities can be hosted on the same physical machine. Without any authentication scheme, the nodes are organized as overlay P2P networks. Comparatively, in a consortium or permissioned DLT-based network, only the authorized nodes are allowed to enable the core functionalities such as consensus participation or data propagation. The authorized nodes may be organized in a different topology (i.e., fully connected P2P networks). In this paper, we mainly focus on the network protocols in the permissionless cases (i.e., the partially connected P2P network).

In a permissionless distributed ledger system, each node in the network established connections with only a limited number of neighbours (i.e., partial connections) for free joining or leaving the P2P network and fast messaging propagation. All peers in the P2P distributed network are expected to receive all blocks, so a broadcast protocol is required. The choice of the underlying broadcast protocol clearly impacts the resilience and scalability of the network. The broadcast protocol defines how a new message spreads from the initial node to all other online nodes in the network. In permissionless DLT-based networks, the main network protocol is to induce a random topology among the nodes and propagate information efficiently for ledger replica synchronization. Most of the existing DLT-based networks employ the reay-to-use P2P protocols with slight modifications for topology formation and data communication.

To replicate the blockchain over all nodes in the network, the messages of transactions and blocks are "broadcast" by flooding the P2P links in a gossip-like [38] manner. Specifically, a P2P link in the DLT-based network is built on a persistent TCP connection after a protocol-level three-way handshake, which

exchanges the replica state and the protocol/hardware version of each node in the network [38, 33, 35, 88, 84]. Every peer in the P2P network actively tries to maintain a minimum C_{min} and maximum C_{max} connections in the overlay. That is, the peer tries to establish additional connections if this minimum number is underrun. The minimum number of connections can be significantly exceeded if incoming connections are accepted by a network peer, but it doesn't handle more than the maximum number of connections at a time, considering the overhead of the computation and network resources. After the connections to the peer nodes are established, another three-way handshake occurs for a node to exchange new transactions or blocks with its neighbours. The node first notifies its peers with the hashcode of the new transactions/blocks that it receives or generates. Then, the peers reply with the data transfer request specifying the hashcode of the information that they need. Upon request, the transfer of transactions or blocks is done via individual transfer messages [33]. The data transfer in the blockchain network is typically implemented based on the HTTP(s)-based Remote Procedure Call (RPC) protocol, where the message is serialized following the JSON protocol [33].

Propagation Protocols. The complexity of the broadcast propagation speed of the partially connected network is $O(log(N_N))$, where N_N is the number of active nodes in the network.

Proof. For a partially connected network, there a lower bound C_{min} and a upper bound C_{max} of number of connections for each node in the network. We can assume that there is an average number of connections C_{ave} for each node because of the □

Network Bandwidth. The network bandwidth, denoted by B_N, defines the maximum capability of each network connection, which indirectly reflects the speed of the data propagation. The network bandwidth can directly control the latency of the transaction confirmation. Large network bandwidth incurs faster data (i.e., transactions and blocks) propagation, which in turn slows down network synchronization and reduces the stale block rate (and enhances the security of blockchain).

2.4.4 Data Organization

In the distributed ledger system, the types of messages mainly include transaction data and block data. The size of the transaction data is fixed and does not have a significant impact on system performance. However, the size of block data will be significantly improved with the increase in the number of transactions, which greatly reduces the speed of block broadcast and thus affects the performance of network synchronization. Additionally, the bandwidth of network connections also has a significant impact on the speed of the network propagation. For blocks of the same sizes, the larger the bandwidth, the faster the propagation.

Data Ordering. In distributed ledger system, there are main three ways two order the data: linear (e.g., blockchain), nonlinear (e.g., Tree) and Directed Acyclic Graph (i.e., DAG).

Data Aggregation. In a distributed ledger system, there are two main ways to aggregate the data: block and transaction. the types of messages mainly include transaction data and block data. The size of the transaction data is fixed and does not have a significant impact on system performance. However, the size of block data will be significantly improved with the increase of the number of transactions, which greatly reduces the speed of block broadcast and thus affects the performance of network synchronization. Additionally, the bandwidth of network connections also has a significant impact on the speed of the network propagation. For blocks of the same sizes, the larger the bandwidth, the faster the propagation. The block size, denoted by S_B, defines the maximum capacity of each block, which indirectly reflects the maximum number of transactions carried within each block. The block size can directly control the throughput attained by the system. Large blocks incur slower broadcast speeds, which in turn increases the stale block rate (and weakens the security of the blockchain).

2.5 Evaluative Model

Based on the specific underlying model, we can systematically evaluate each consensus algorithm from three aspects, including resilience/robustness, performance and governance. Moreover, these evaluative properties are severely affected by the attacking models, such as double-spending, selfish mining and so on. Nextly, we

will firstly elaborate the common attacking models in distributed ledger system and then introduce above evaluative properties in details.

2.5.1 Resilience

A consensus algorithm is a process in computer science used to achieve agreement on a single data value (e.g., a new block in the blockchain system) among distributed system. Consensus is not a new problem and the community of the distributed system has extensively studied it for decades. Achieving consensus in a distributed system is challenging and consensus algorithms have to be resilient to various failures in the system as displayed in the failure model. For a blockchain network, achiveing consensus ensures that all nodes in the network agree upon a consistent global state of the blockchain. A consensus protocol has three key properties based upon which its applicability and efficacy can be determined [71, 88, 2].

- **Safety.** A consensus protocol is determined to be safe if all nodes produce the same output (i.e., *agreement*) and the output is valid according to the rules of the protocol (i.e., *validity*).
- **Liveness.** A consensus protocol guarantees liveness if all non-faulty nodes participating in consensus eventually produce a value (i.e., *termination*).
- **Fault Tolerance.** A consensus protocol provides fault tolerance if it can recover from failure of nodes participaring in the consensus.

In a distributed ledger system (e.g., blockchain system), if an honest node confirms a new block header, agreement means that any other honest node that updates its local blockchain view will updates with that block header; if all the honest nodes activated on a common state propose to expand the blockchain by the same block, validity means that any honest node transiting to a new local replica state adopts the blockchain headed by that block. Agreement and validity are both crucial because they can guarantee the DLT-based network can provide correct and robust services. Meanwhile, in a distributed ledger system, especially the blockchain system, liveness means that blockchain keeps growing by adding the valid new blocks. Liveness is important because it's the only way that the network can continue to be useful — otherwise, it will stall.

While all the above three properties are crucial, a famous result (i.e., FLP Impossibility result) proven in work [32] by Fischer, Lynch and Paterson states that no deterministic consensus protocol can guatantee safety, liveness and fault tolerance in an asynchronous system (an asynchronous and deterministic consensus protocol cannot tolerate any failure). While fault tolerance is crucial for globally distributed networks to operate, distributed systems tend to choose between safety and liveness depending on their system requirements and assumptions.

In order to guarantee fault tolerance, some redundancy is necessary in the distributed ledger system, and the amount of redundancy depends on the failure types. In the face of unreliable networks provisioning mainly for benign faults (i.e., crash-fail and omission), the number of nodes needed in such networks are $n >= 2f + 1$ to be able to tolerate f failures. Consider, for example, three entities holding a value. Assume that a single entity fails: it always returns the same, wrong value. By comparing the answers, it is easy to identify the failing entity and to decide on the true value. Two entities would not suffice, an analogous situation would result in a conflict.

However, failures in a broader sense can also be arbitrary or malicious (i.e., Byzantine faults after the famous Byzantine Generals problem [54]). The original problem description considers the case of n generals trying to mutually agree via messengers on a common battle plan. However, f of the generals are traitors and try to thwart the agreement. The situation is comparable to a distributed system which aims to reach consensus. The Byzantine Generals problem with synchronous and reliable communication reaches consensus as long as $n >= 3f + 1$ is satisfied [54]. In the case of asynchronous communication, constrained by the FLP Impossibility results, the tolerance of the Byzantine failures depends on the timing model (i.e., synchronous assumptions), message order and transmission method (i.e., peer-to-peer in full connections or broadcast in partial broadcast) [24].

2.5.2 Performance

Distributed consensus, infamous for its limited scalability, was for decades perceived as a synchronization primitive that is to be used only in applications in desperate need of consistency and only among a few nodes. However, Nakamoto's

Bitcoin cryptocurrency demonstrated the utility of decentralized consensus across thousands of nodes, changing the world of digital transactions forever. Two metrics are directly related to DLT-based network scalability [19]:

- **Throughput.** The maximum rate at which the DLT-based network can process transactions.
- **Latency.** Average time to confirm that a transaction has been included in the distributed ledger.

Scalability. Scalability, including transactions and nodes in the network, mainly focuses on the impacts on the throughput and latency when the number of transactions and nodes in the system increases significantly. Scalability directly reflects the application range of the consensus algorithm. Consensus with low scalability can only be applied to small-scale permissioned or consortium networks, while high scalability can be used to large-scale permissionless networks.

Data Organization. Original from the Bitcoin [66], blockchain is the mainstream framework of the immutable data organization of the distributed ledger system. However, the lower throughput and higher latency facilitate the rise of the new storage (e.g., DAG [21]), which effectively enhances the performance of the distributed ledger system through the parallel execution of blockchain transactions.

Specifically, in early DLT-based networks (e.g., Bitcoin [66], Bitcoin-NG [28]), the digitally signed transactional records are arbitrarily "packaged " into a cryptographically tamper-evident data structure known as "block". The blocks are then organized in a chronological order as a "chain of blocks", or more precisely, a linear list of blocks linked by tamper-evident hash pointers. Nevertheless, in order to improve the processing efficiency of transactions and network scalability, some researchers have made efforts to expand the linear data organization framework (i.e., blockchain) into the nonlinear structures such as trees [46] and graphs of blocks [80, 21]. Furthermore, block-less, non-linear data structure are also adopted in recent protocol design (e.g., Tangle [73]). Despite the different forms of storage structure organization, cryptographic data representation provides the fundamental protection of privacy and data integrity for DLT-based networks.

2.5.3 Governance

Before diving into the details of how governance works on blockchains, it's important to have a clear definition of what blockchain governance is. Every blockchain is an evolving system which needs to change to meet the needs of its users. If a blockchain isn't relevant and useful, then it won't survive, it needs to be able to evolve and adapt. To evolve, the blockchain needs to make changes and needs a way to make final decisions on what these changes should be. Organizations usually have a leadership team or a CEO who is the final authority for their organization. However, blockchain is designed to be decentralized in its nature, and not be under the control of any person or group. This means that blockchain needs another way to make decisions regarding the blockchain's roadmap.

So, blockchain governance, in order to be effective, needs to include both incentives and methods for members to coordinate. Without incentives, members won't participate in governance, and the blockchain will become less aligned with user needs over time. Without a method for members to coordinate, it will be impossible for a blockchain network to come to an agreement on future changes. Think about how many hard forks we could have.

Incentive Mechanism. Each group in the system has their own incentives. Those incentives are not always 100% aligned with all other groups in the system. Groups will propose changes that are advantageous for them over time. Organisms are biased towards their own survival. This commonly manifests in changes to the reward structure, monetary policy, or balances of power. Since it's unlikely all groups have 100% incentive alignment at all times, the ability of each group to coordinate around their common incentives is critical for them to affect change. If one group can coordinate better than another, it creates power imbalances in their favour. In practice, a major factor is how much coordination can be done on-chain vs. off-chain, where on-chain coordination makes coordinating easier. In some new blockchains, on-chain coordination allows the rules or the ledger history to be changed.

2.5.3.1 Anti-attacking

According to the evaluative model's results, we can analyze the consensus algorithm's anti-attacking, which intuitively reflects the robustness of each consensus algorithm. Based on the anti-attacking, we can analyze the consensus's internal

trade-offs and even come up with some improvement plans.

2.6 PoW Evaluation

For consensus evaluation, we select a typical consensus PoW of Bitcoin [65] to demonstrate the application of our evaluative framework. The main motivation of PoW is the scalability in the third development stage and the large-scale application of consensus algorithms.

Table 2.1: Functions of four sub-processes of PoW consensus

Leader election	Compute competition puzzle: $H(x\|nonce) \leq Difficulty$
Data package	The first miner finding the "nonce" can package the complete block
Data validation	Verify and broadcast blocks
Consensus record	Commit the block

Figure 2.2 shows the process of the PoW, including leader election, data package, data validation and consensus result record. Table 2.1 shows the core functions of these four sub-processes, and what I want to emphasize is the computing competition for the right of the next block's generation. The first miner who solves the puzzle will obtain the right and get the rewards from the system, but he also needs to spend massive resources. Next we will evaluate the PoW consensus with the proposed framework.

2.6.1 Underlying model

Table 2.2: Underlying model of PoW consensus

Timing model	**Failure model**	**Network configuration**	**Data organization**
Partially asynchronous	BAR faults	Partial connection	Blockchain

Table 2.2 shows the underlying model of the PoW, where the timing model is partially asynchronous because of the partial synchrony for network connectivity and mining difficulty adjustment, and the asynchrony for block generation, the failure model is BAR faults because there are no limits for the consensus attendance, the network configuration is a partial connection because any node only connects to the limited neighbours and the data organization is blockchain. Rational nodes can carry out attacking models in the PoW, which mainly include double-spending [45], Eclipse attacks [40], and selfish-ming [78].

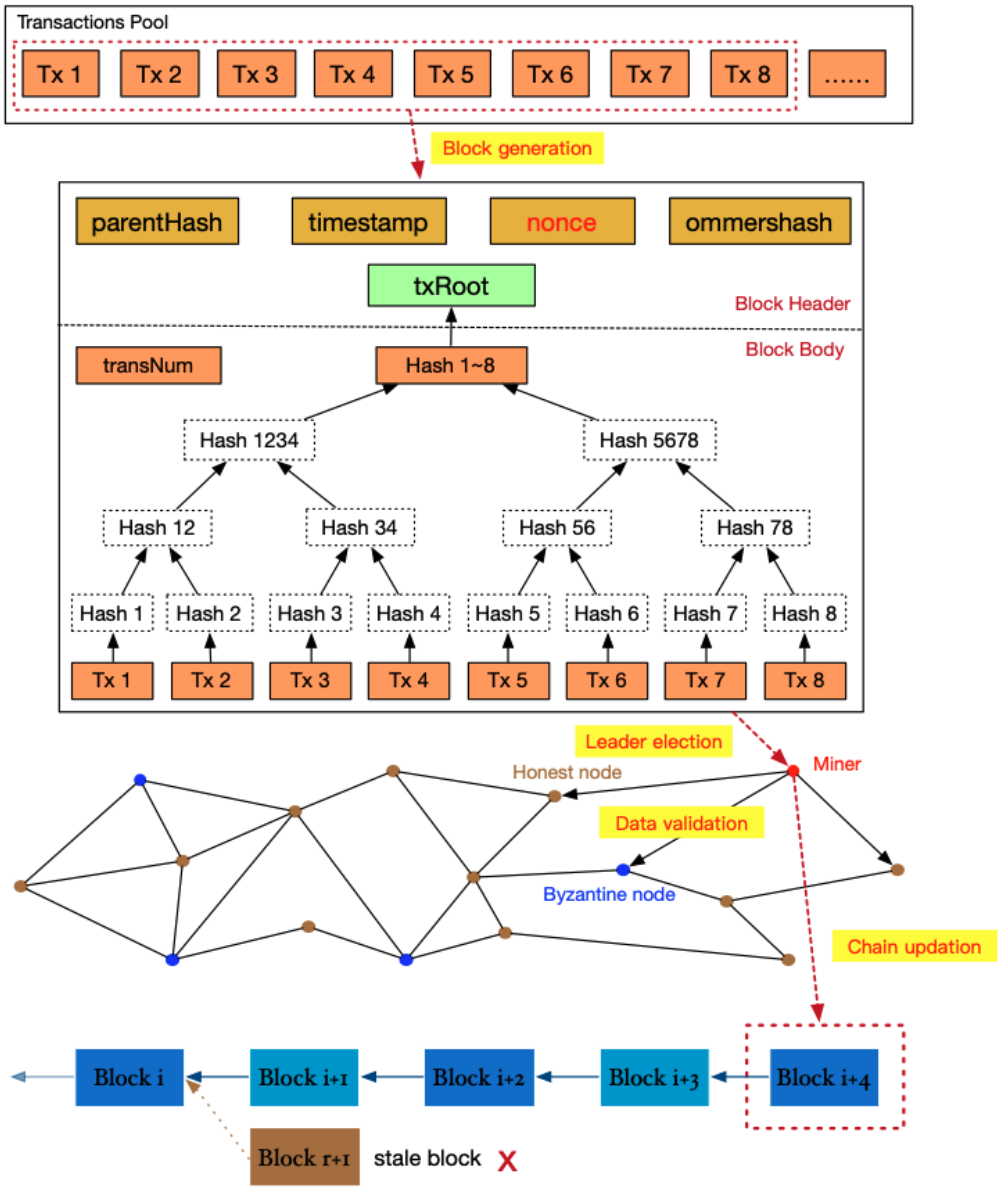


Figure 2.2: Process of PoW consensus

2.6.2 PoW Evaluation

For resilience, the fault-tolerance assumption is probabilistic. Firstly, the agreement and validity of PoW are both probabilistic, depending on the speed of message transmission in the P2P network and the computing power of Byzantine nodes, respectively. Because of the competitive protocol of recording right for the ledger, the first one who can broadcast the correct result to the whole network can record the result in the global ledger. If two miners mine the result simultaneously, there will

be brief forks across the entire system, so the agreement is probabilistic. Moreover, the higher computing power a miner has, the more likely he is to gain access to the ledger, so the validity is probabilistic. The finality of pow is temporary due to the PoW protocol design (i.e., fork protocol [66]). However, the termination is deterministic, which depends on the adjustment of puzzle difficulty. Moreover, we want to clarify PoW overcoming FLP impossibility results. PoW does not achieve the type of consensus as constrained by the FLP theorem and indeed includes non-determinism to mitigate attacks.

For governance, firstly, the PoW is a non-deterministic consensus algorithm, and the coverage is full. The cost of the PoW is mainly the consumption of electricity, and the benefit mainly depends on the mining rewards, including transaction fees and system rewards. There is no supervision in PoW, and the penalty is mainly the implicit useless mining of stale blocks. So, the randomness of the hash function guarantees fairness and both incentives from tokenomics and fairness guarantee liveliness.

Finally, we will evaluate the PoW's anti-attacking strategy. For double-spending, the anti-attacking is probabilistic, which mainly depends on the computing power of the attackers. Meanwhile, PoW cannot anti-attack the Eclipse attack because it can enable an adversary to carry out 51% attacks with less than 51% computing power. Moreover, For selfish mining, the anti-attacking is also probabilistic, which mainly depends on the computing power of the attackers.

2.7 Related Work

2.7.1 Surveys for Distributed Consensus

Work [88] provides a comprehensive literature review on the development of decentralized consensus mechanisms in blockchain networks. By emphasizing the unique characteristics of incentivized consensus in blockchain networks, our in-depth review of the state-of-the-art consensus protocols is focused on both the perspective of distributed consensus system design and the perspective of incentive mechanism design. Work [4] presents a comprehensive systematization of blockchain consensus protocols and evaluates their performance and security properties according to the

different categories.

2.7.2 Consensus Evaluation

Blockbench [23] is a framework for evaluating the security and performance of private blockchains. Their evaluation reveals that due to design gaps, popular blockchains lag far behind traditional database systems when processing traditional data processing workloads. Their recommendations include systematic benchmarking, improved usability, and revitalizing classical database design principles such as modularity, exploiting hardware primitives, sharding, and support for declarative languages. Work [36] presents a quantitative framework to evaluate the security and performance of PoW blockchains. They focus on optimal adversarial strategies for double-spending and selfish mining while accounting for network propagation, different block sizes, block generation intervals, information propagation mechanisms, and the impact of eclipse attacks. Work [19] presents metrics to evaluate the resource costs and performance of Bitcoin with a focus on scalability. They show that even with reparametrization, Bitcoin can only achieve a maximum throughput of 27 tx/s with a latency of 12 seconds.

2.8 Conclusion

This chapter mainly focuses on introducing the general evaluative framework for consensus protocols, which includes an underlying model and a three-layer evaluation model. Finally, we evaluate the PoW consensus to demonstrate the framework's feasibility.

Chapter 3

Proof-of-Data: A Consensus Protocol for Collaborative Intelligence

Existing research on federated learning has been focused on the setting where learning is coordinated by a centralized entity. Yet the greatest potential of future collaborative intelligence would be unleashed in a more open and democratized setting with no central entity in a dominant role, referred to as "decentralized federated learning". New challenges arise accordingly in achieving both correct model training and fair reward allocation with collective effort among all participating nodes, especially with the threat of the Byzantine node jeopardising both tasks.

In this chapter, we propose a blockchain-based decentralized Byzantine fault-tolerant federated learning framework based on a novel Proof-of-Data (PoD) consensus protocol to resolve both the "trust" and "incentive" components. By decoupling model training and contribution accounting, PoD is able to enjoy not only the benefit of learning efficiency and system liveliness from asynchronous societal-scale PoW-style learning but also the finality of consensus and reward allocation from epoch-based BFT-style voting. To mitigate false reward claims by data forgery from Byzantine attacks, a privacy-aware data verification and contribution-based reward allocation mechanism is designed to complete the framework. Our evaluation results show that PoD demonstrates performance in model training close to that of the centralized counterpart while achieving trust in consensus and fairness for reward allocation with a fault tolerance ratio of 1/3.

3.1 Introduction

The data economy today is becoming increasingly collaborative in nature. Take business intelligence, for example. To unleash the full potential of big data, it is essential to integrate multi-source data depicting entities from a multi-faceted and multi-modal perspective, which, not surprisingly, is not achievable by any company alone. It is mutually beneficial for companies to leverage each other's data for collective model training. On the other hand, however, privacy and security concerns have long been major roadblocks in cross-entity data exchange.

Among all the approaches proposed to resolve these data silo issues, federated learning [49] has gained growing popularity due to the fact that participating training nodes can retain all their data on-premise, train models locally and exchange only model parameters to cooperatively obtain a common model better than what each can individually train, maximally protecting their data privacy and security. This is the learning environment of collaborative intelligence we will focus on in this paper.

Unfortunately, existing federated learning models focus mostly on settings with a central entity to coordinate all other nodes in the training process, which we refer to as the "centralized" setting. While useful for some scenarios, we argue that the centralized setting will not be the most important and challenging collaborative intelligence mode in the future. Competing businesses would not participate if one is in a dominating position superior to the rest in the ecosystem. For all businesses to willingly contribute and collaborate sustainably, the system must be open to all and dominated by none, which we refer to as the "decentralized" setting.

While there have been researching efforts on decentralized federated learning [39, 47, 60, 58], the assumption is that all participating training nodes are cooperative and motivated for a common goal in good-will spirit, typical of a consortium with permissioned entry. The main task there is to maintain model consistency across various nodes in an amicable setting. Yet real-life application settings are never that rosy. A large-scale collaborative data intelligence ecosystem open to all must accommodate participants of all kinds, including those malicious nodes which are typically referred to as *Byzantine* nodes in distributed systems. Achieving correct consensus despite the existence of Byzantine nodes is called Byzantine fault tolerance.

Two main challenges arise in this decentralized setting with Byzantine fault

tolerance: (I) How to collectively train a common correct model with comparable results as in centralized federated learning; and (II) How to design fair incentivization to properly reward participating training nodes for their contribution in terms of data. The most difficult part of both challenges is to handle the "Byzantine" nodes that are largely ignored in existing studies for centralized settings. It is worth noting that in our case, Byzantine nodes not only refer to malicious nodes in traditional consensus research – those attacking the system by compromising the consensus, which is the main task of challenge I – but also to nodes harmful to the system by scheming for unwarranted reward allocation from their data contribution, the main task of challenge II.

To overcome these challenges, we propose a novel consensus protocol called *Proof-of-Data* (i.e., PoD) to achieve a decentralized Byzantine fault-tolerant federated learning framework. The contributions of this chapter can be summarized as follows:

- We propose a novel consensus protocol called Proof-of-Data tailored for decentralized federated learning with Byzantine fault tolerance. PoD combines Proof-of-Work (i.e., PoW) style asynchronous consensus with epoch-based consensus locking by a PBFT-style component, integrating the best of both by endowing, on the one hand, the missing consensus finality to the practically robust yet theoretically flawed PoW consensus and lending, on the other hand, the scalability necessary for societal-scale application setting to the otherwise sound PBFT consensus.

- We design a privacy-preserving data verification mechanism based on P4P [25], a zero-knowledge proof protocol, to prevent participating nodes from producing inconsistent data contributions in the training process without violating their data privacy.

- We devise an incentive mechanism to assess and allocate rewards based on nodes' data contribution, mitigating the risk of Byzantine attacks in terms of data contribution without sacrificing model performance economically.

- Finally, we comprehensively evaluate the performance of the framework as well as analyze the resilience, performance and governance of PoD. An analysis of the framework's anti-attacking capability is also provided. The framework

is empirically validated through a wide range of experiments on both time-invariant and time-varying datasets. The results of these experiments show that our framework performs closely with Centralized Federated Learning.

The remainder of this chapter is organized as follows. First, we formulate the decentralized federated learning (DFL) problem and present design ideas in Section 3.3 to better understand our PoD consensus protocol, which is detailed in Section 3.4. In section 3.5, the resilience, performance and governance of PoD are theoretically analyzed. Section 3.6 presents the experiments and evaluates the framework's performance. Section 3.7 discusses related work, and Section 3.8 concludes this paper.

3.2 Problem Formulation

We follow the standard notion to model the underlying environment of our problem with n training devices, among which several devices could be Byzantine [53] adversaries.

Definition 1. (Decentralized Byzantine Federated Learning): Given 1) a set $\mathcal{P}$ of n geo-distributed devices $\{\mathcal{P}_i\}_{i\in[n]}$ with private datasets $\{\mathcal{D}_i\}_{i\in[n]}$ connected by P2P network of which a set $\mathcal{A}$ of f devices are Byzantine nodes which conduct Byzantine attacks randomly, where $[n]$ is short for $\{1, 2, ..., |\mathcal{P}|\}$ through the paper; 2) a model with the objective function $F(\omega)$, where ω is the model weights, the Decentralized Byzantine Federated Learning problem, denoted as DBFL, aims to make all devices $\{\mathcal{P}_i\}_{i\in[n]}$ collectively train a common model weights set $\mathcal{W} = \{\omega_i\}_{i\in[n]}$ with private datasets $\{\mathcal{D}_i\}_{i\in[n]}$ through the exchange of information over an asynchronous network. Formally, the objective function of DBFL is to minimize the following function:

$$\min_{\mathcal{W}} G(\mathcal{W}) = \sum_i p_i F(\omega_i; \mathcal{D}_i). \tag{3.1}$$

Here, $\omega_i \in \mathbb{R}^d$ is the model weights of device $\mathcal{P}_i$, $p_i \geq 0$ specifies the relative impacts of each device to the whole network, and $\sum_i p_i = 1$.

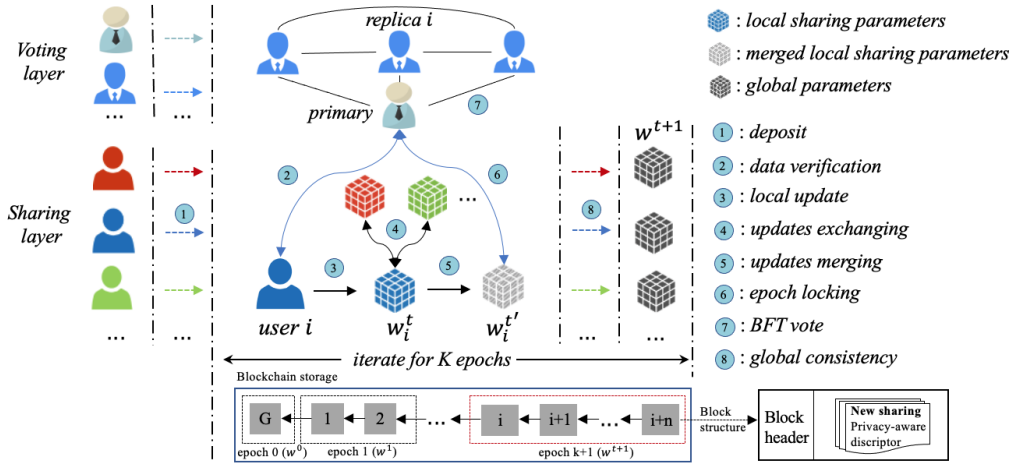


Figure 3.1: PoD: a consensus protocol for collaborative intelligence.

3.3 Design Ideas

To explain our design ideas, we start with our goal: To achieve decentralized Byzantine federated learning for societal-scale applications. This context entails three essential characteristics - (1)the large number of participating nodes, (2) the existence of Byzantine nodes and (3) the absence of a central coordinating entity. As a result, a consensus protocol is necessary to guarantee the consistency of model training and contribution accounting across different nodes with no "trust" assumption among them, lending the "trust" component to the solution. On the other hand, "incentive" component in terms of reward based on data contribution is equally indispensable to ensure the motivation of participation. The combination of both "trust" and "incentive" is the foundation of sustainable collaborative intelligence in a real-life setting.

Constrained by the FLP impossibility result [32], which states that a correct consensus algorithm is impossible if three properties are to be achieved simultaneously: (I)Asynchrony, (II)Determinism and (III)Fault tolerance, the design of any consensus protocol is essentially balancing the trade-off among the three properties.

In our setting, first of all, "fault tolerance" is indispensable as an open-access societal-scale application with neither simple faults nor Byzantine faults is simply unimaginable. "Determinism" is also deemed important because the consistency and the assurance of both model training result and reward distribution are crucial for

continual participation of data-contributing nodes and liveliness of the system. We are left with "asynchrony" as the only option to let go.

Suppose we relax the "asynchrony" property, can we use existing consensus protocol designed for synchronous or partially synchronous setting such as PBFT [12, 63, 10]? The answer seems to be negative because a quadratic time complexity in the number of nodes is infeasible for large-scale applications as we aim for in our case. Meanwhile, the nodes responsible for carrying out the protocol in PBFT are fixed, but in our setting, the nodes can join or quit at any time. More fundamentally, an asynchronous mode is much more desirable in our federated learning context as nodes do not need to wait for all others to complete training to benefit from the already partially-trained result.

In order to still enjoy the efficiency from asynchrony while keeping both fault tolerance and determinism, we draw inspiration from the design of PoW (i.e., Proof-of-Work [66]) as used in Bitcoin. Asynchrony (i.e., a node does not need to wait for any other node to proceed to mine independently) has played a critical role in the success of Bitcoin as the first application of consensus protocol in a societal-scale setting (18,000 public nodes as of February 2024). Unfortunately, however, PoW does not achieve the classic definition of consensus, as the finality is never secured. Specifically, it does not achieve the classic round-by-round consensus in terms of resilience, and the probability of eventually achieving global consensus increases over time, approaching infinitely close to but never reaching one.

To remedy the situation, we propose the idea of **decoupling of model training and contribution accounting** based on the following observation:

- Model training is the task performed by all the nodes most of the time with each data update. Contribution accounting, on the other hand, can be executed periodically at model training milestones when actual reward distribution is conducted.

- The task to benefit the most from asynchronous processing is the model training part – a globally consistent model training consensus is not necessary for nodes to benefit from partially trained results.

- The task that indeed requires finality for global consensus yet is to be performed

periodically is the contribution accounting part – a synchronous or partially synchronous protocol is possible as we do not necessarily need all the nodes to participate.

As shown in Fig. 3.1, we, therefore, propose a two-layer architecture for the decoupling design idea, with an underlying blockchain structure to immutably record the training result and contribution accounting. The sharing layer is responsible for the asynchronous model training and generating new blocks, while the voting layer is responsible for the locking of the training result and contribution periodically (i.e., by epoch) to secure consensus finality in a partially synchronous BFT voting manner.

The benefit of decoupling the two tasks is demonstrated from our experiments (see section 3.6), where the superiority of our solution over the centralized one can be seen in that nodes can already benefit from other nodes' data by adopting local partial results due to asynchronous model training, while each node, after submitting the parameters to the central server, would have to wait for the central server to finish processing the submission from all nodes before getting back the updated parameters to use.

3.4 Proof-of-Data Consensus Protocol

The two-layer consensus protocol as introduced in Section 3.3 is termed Proof-of-Data (i.e., PoD). In a DBFL system, all nodes $\{\mathcal{P}_i\}_{i\in[n]}$ join the network at random and start training the model $F(\boldsymbol{w}_i^j;\mathcal{D}_i)\rightarrow \boldsymbol{w}_i^{j+1}$ with their private datasets $\{\mathcal{D}_i\}_{i\in[n]}$, and can only exchange information employing two-party messages with P2P network with no central server to integrate information or manage distributed nodes. During the process of model training, nodes in the system generate a different sequence of model weights states $\{w_i^{j+1}\}_{i\in[n],j\geq 1}$, and some of these states may be spurious due to Byzantine nodes. Therefore, PoD takes trained model weights sequences $\{w_i^{j+1}\}_{i\in[n],j\geq 1}$ from nodes $\{\mathcal{P}_i\}_{i\in[n]}$ as inputs and aims to output a common model weights set $\mathcal{W}=w^{j+1}$. PoD guarantees the properties below except with negligible probability under the influence of any Byzantine attacks:

- `Safety`: 1) Agreement: if any two honest training nodes output $\boldsymbol{w}$ and $\boldsymbol{w'}$ for ids respectively, then $\boldsymbol{w} = \boldsymbol{w'}$; 2) Validity: if a training node outputs a model weight $\boldsymbol{w}$ for id, then it is an honest node; 3) Finality: if a model weight $\boldsymbol{w}$ is locked by voting nodes, it can not be revised any more.

- `Liveness`: 1) Termination: if every honest node $\mathcal{P}_i \in (\mathcal{P} - \mathcal{A})$ is activated on identification id, with taking as input a dataset $\mathcal{D}_i$ s.t. $F(\boldsymbol{w}_i^j; \mathcal{D}_i) \rightarrow \boldsymbol{w}_i^{j+1}$, then every honest node output a model weights $\boldsymbol{w}$ for id; 2) Liveliness: if a training node outputs a model weight $\boldsymbol{w^j}$ for id, then it is available to output $\boldsymbol{w^{j+1}}$ continuously.

Note that the algorithm need not reveal which nodes are faulty and that the outputs of faulty nodes may be arbitrary; it matters only that the non-faulty devices compute the same valid model weights vector for any given faulty node. Eventually, the non-faulty nodes come to a consistent view of the values of the model weight vector held by all the nodes, including the faulty ones. Fig. 3.1 shows the overview of the PoD consensus, and we will introduce important components in subsection 3.4.1 and processes in subsection 3.4.2 with two key proposed mechanisms: data verification in subsection 3.4.3 and measurement of data contribution in subsection 3.4.4.

3.4.1 Components of PoD Framework

As shown in Fig. 3.1, PoD can be viewed at a high level with three components: an underlying blockchain component, the sharing layer and the voting layer. We denote nodes in the sharing and voting layer as p_i^s and p_i^v, respectively. Meanwhile, We design two types of roles for a node in PoD: training nodes refer to nodes at the sharing layer, responsible for the training model, and voting nodes refer to nodes at the voting layer, responsible for BFT voting. The underlying blockchain maintains the model updates and settlement rewards.

3.4.1.1 Sharing layer

Nodes in the sharing layer are responsible for handling three tasks:

- Obtain the intermediate parameters with the latest private dataset, merge updates from other devices in the network to generate a new block, and then

broadcast the new block to other devices. To do that, the node must first deposit for epoch-sharing authorization.

- Listen for new blocks and epoch locking to trigger blockchain replacement and block merging events.
- Merge the updates generated by themselves and the listened blocks from the network

Furthermore, there are two main challenges during the whole process. The first is that the local data might be fabricated, and the second is that the devices are not always available. To address the first challenge, we let the voting layer verify the real existence of data with data summary information (see section 3.4.4). We design a local learning progress for nodes to address the second one. Nodes can actively obtain the initial global model of the working epoch at any time and perform local training. When eligible updates are packaged on the blockchain as a reward, tokens can be attached to them at the epoch settlement.

3.4.1.2 Voting layer

The finality of the consensus can not be resolved within the sharing layer. To do this, we have to overlay the voting layer on top of it, which will be a small set of nodes. The voting layer is mainly responsible for Epoch locking, data verification and value allocation. We maintain an active training device list around the system for epoch locking. Once the primary node of voting devices receives the signal of the epoch locking from the sharing layer, he will call for a vote. Then, all the voting devices will validate the block according to the active training list.

If more than the fault tolerance threshold τ per cent of nodes, the voting devices pass the validation and sign for the new block containing the locking signal. Then, the epoch will be locked. All other training devices will synchronize the locking epoch and start a new training round based on the new deterministic global model, and all the updates in the previous blocks cannot be tampered with anymore. In addition, before broadcasting the signed epoch, the voting devices will make an epoch settlement to distribute rewards according to the data contribution (see section 3.4.4). Moreover, we also listen for the data verification to prevent data forgery with privacy protection based on the P4P method (see section 3.4.3).

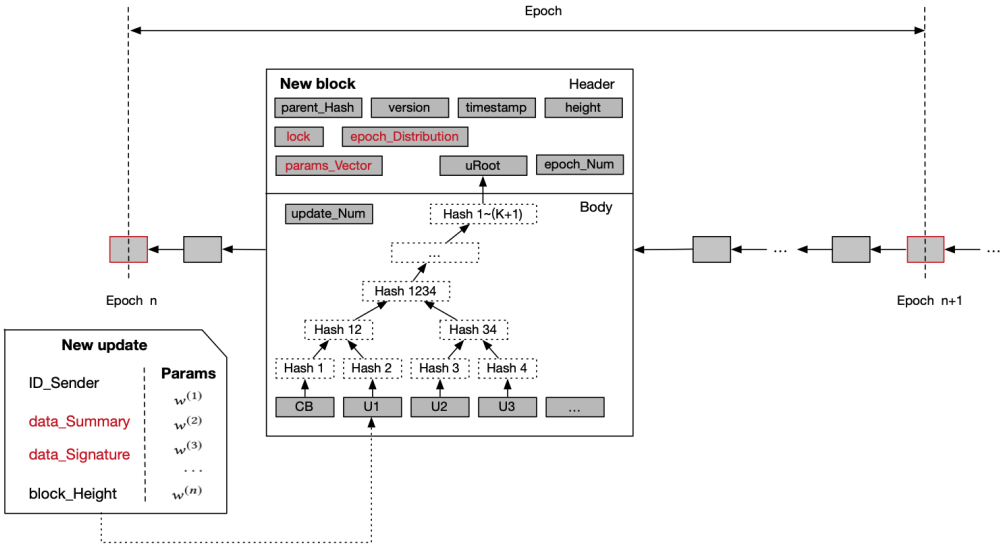


Figure 3.2: Block structure of PoD.

3.4.1.3 Block Structure

The block structure in PoD would contain most of the information in a typical blockchain structure (e.g., a block header and a block body) as shown in fig 3.2. We would focus on the information unique to PoD.

Definition 3. (Update): We define the training result of a device $\mathcal{P}_i$ with private dataset $\mathcal{D}_i$ as an update U_i^h, where h is the block height. An update mainly consists of the following fields:

- w: the latest model weights;
- ID_Sender: unique identity of the sender;
- $data\_Summary$: a summary that summarizes the characteristics of the data; In our work, we use the Gaussian Mixture Model (i.e., GMM [8]) to fit the private datasets of users and $data\_Summary$ mainly includes mean μ, covariance Σ and coefficient α.
- $data\_Signature$: a data falsification verification proving the reliability of the training results.

In PoD, nodes will pass local updates to other nodes, who will then decide whether to merge or not. We define the merging result of local updates of sharing

device p_i^s, $i \in [n]$ as a block B_i^h, where h is the block height. The block body mainly stores the specific data of all the updates that have been merged in the block, and the block header consists of some specific fields related to the storage management and system settlement.

Most noticeably, an important notion **epoch** is introduced to finalize, exactly to address the finality issue in the proof of our consensus, we use epoch to lock consensus on the blockchain permanently. We denote an epoch as E^H, where H is the epoch height. Each epoch contains a certain number of blocks (e.g.,100) with a fixed block size, and the number of blocks in an epoch depends mainly on the number of participating nodes in the system. At the end of an epoch, the voting layer must finalize the consensus by locking the current epoch. Once an epoch is locked, the information stored on the blockchain cannot be changed anymore.

3.4.2 Process of PoD Algorithm

As we have discussed in subsection 3.4.1, the sharing layer is mainly responsible for our asynchronous model training and generating new blocks with merging protocol, and the voting layer is mainly responsible for the partially synchronous BFT votes of epoch locking to finalize the global model. Firstly, before model training, each training node must pass the data verification to obtain the signature from the voting layer, as shown in steps 2 and 7 of Fig 3.1. Afterwards, the training node begins to train the model with the latest global model and merge other updates to generate new blocks, as shown in steps 3, 4, and 5. Finally, the voting layer will settle the epoch and allocate value according to each user's data contribution to this epoch, as shown in steps 6 and 7. Importantly, then, we have the device management system, which is responsible for token management. Devices need to deposit tokens for epoch sharing to prevent selfish attacks.

Supposing we have four training nodes, A, B, C, and D and one primary voting node, V, as shown in Fig. 3.3. In the process, they train parameters locally and independently. Suppose A is the first one to finish the training with the local private dataset and get the updated parameter vector w_a; then, A will broadcast the w_a to training nodes B, C, and D and apply it to the voting layer for the epoch settlement. Because the merging list in the voting layer only contains A's update less than the

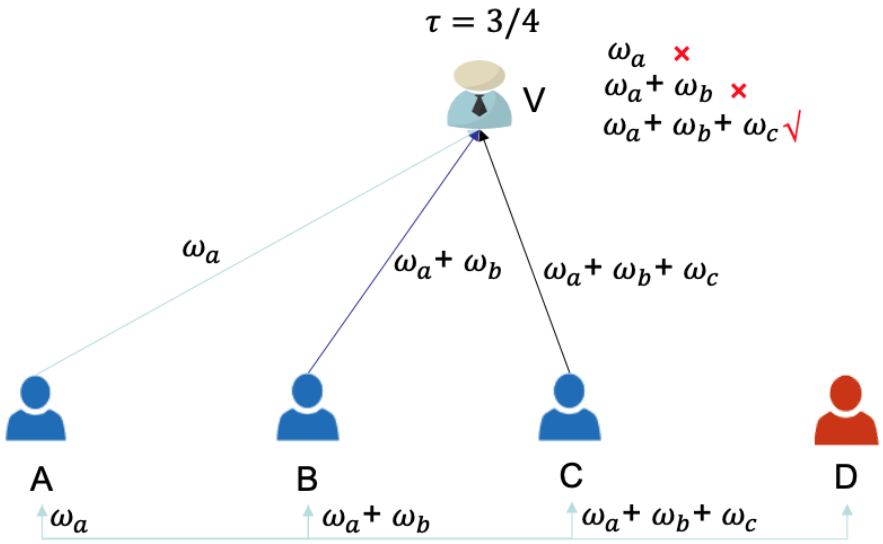


Figure 3.3: Example of PoD process

voting threshold τ, the voting layer passes the request directly. After that, if B gets the parameters update w_b and is the second one to finish training, B will merge the parameter update w_a from training node A with his own update, then broadcast the merged result $w_a + w_b$ to A, C and D and request to voting layer for the epoch settlement too, similarly, this request is passed until C finishes the training and the request with merged $w_a + w_b + w_c$, and the epoch settlement will be agreed because the merging list is large than voting threshold τ. Then, the voting layer will broadcast the locking flag to all training nodes with the latest merged result $w_a + w_b + w_c$ and value settlement result. Once each training node receives the locking signal, it first updates the merging result with the latest one and then stores the result in the blockchain, and then continues to train the next w continuously.

At the same time, because training node D is slow and will be left out in this epoch, the value settlement will only happen in A, B, and C. But if D catch up in the next epoch, D will also benefit from the value allocation. Moreover, suppose D now is dead or malicious on purposely hold the result, the whole system will not be affected because A, B and C will be able to proceed.

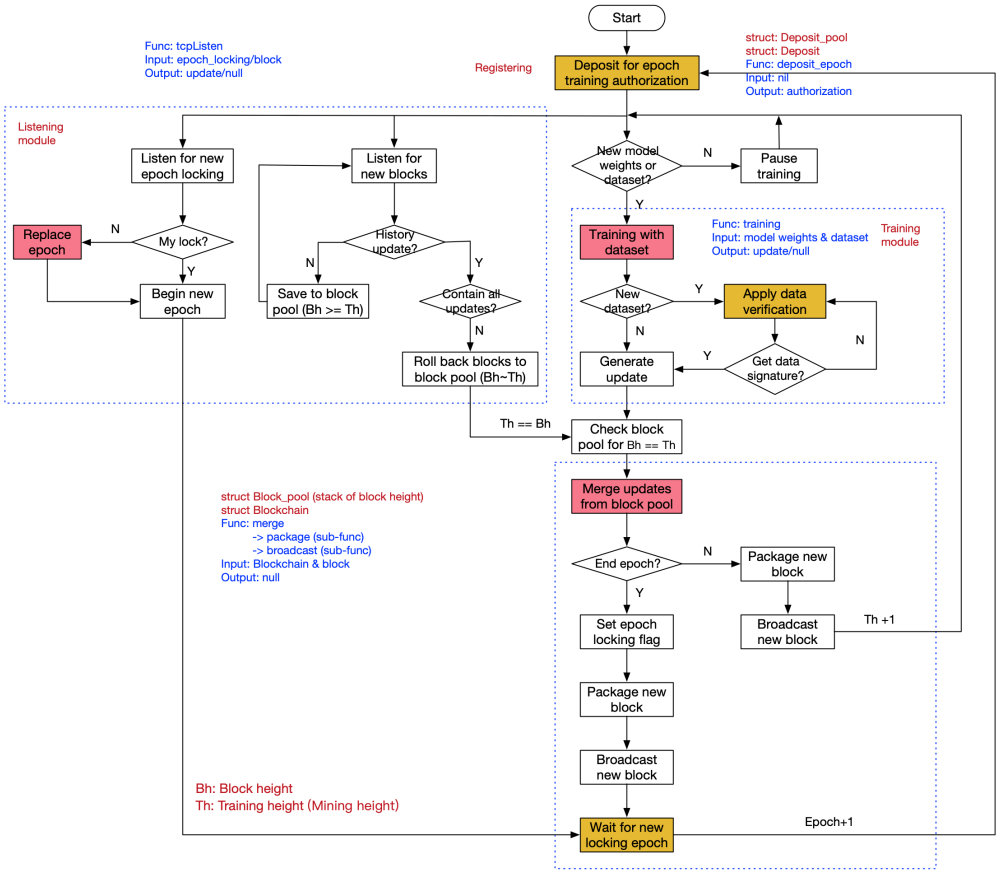


Figure 3.4: Block generation process of training nodes.

3.4.2.1 Block generation

The main task of the sharing layer is to train new updates with the latest private dataset and merge updates from other devices in the network to generate a new block, which is then broadcast to other devices in the network. Moreover, if the block height is at the end of the epoch, the training device needs to wait for the epoch-locking signal from the BFT voting level. Generally, there are three modules of training devices as shown in Fig. 3.4: 1) training module: trains the model with the private dataset to generate updates; 2) listening module: listens for new blocks and listens for new epoch locking; 3) merging module: generate new block and broadcast.

Training nodes perform local training each round. In FL, for security and privacy, raw data will be kept in nodes locally, and these nodes only upload the gradients to the blockchain. Furthermore, there are two main challenges: 1) the local data might

be fabricated; 2) the devices are not always available. To address the first challenge, we let the BFT voting level indirectly verify the real existence of data with the P4P solution [25]. To address the second one, we design an initiative for local learning progress for nodes. Nodes can actively obtain the current global model and perform local training. When eligible updates are packaged on the blockchain, tokens can be attached as a reward. We will discuss the data verification in the next section.

3.4.2.2 Block merging.

For each epoch, each device, including newly entered devices, starts training the model from the same initial state w_0, which is finalized in the last block of the previous epoch. After generating a new update U_i for device $P_i (i \in [n])$ with the private dataset $\mathcal{D}_i$, P_i packages new block and broadcasts the block B_i^h to the whole network and also listens to Blocks B from other devices in the network. Once a block is received from another device, the device P_i integrates the received update according to the *block merging protocol* as shown in fig 3.5. Each block in the network is the merging result of the device and has the merging list or update list. If the update list has contain the received list, the block will not need to be merged, otherwise, we continue to merge the new block and broadcast to the network. Meanwhile, if the block is the history block and we do not merge, we need to roll back the training process and retrain from the height of the history block.

3.4.2.3 Epoch voting

In traditional Proof-of-X (e.g., PoW [65], PoS [62]), there is no deterministic finality for all transactions stored in the blockchain ledger. It's just that over time, the chance of the ledger being tampered with decreases, going to zero indefinitely, which doesn‘t mean tampering doesn't happen. However, for the PoD, tampering with the consensus result stored in the distributed ledger is easy. Therefore, we propose a BFT voting (i.e., PBFT [12]) level to solve the problem. We divide the block height into several epochs, and at the end of the epoch, we will lock the epoch through the voting level. Once the epoch is locked, all the updates stored in the previous blocks can no longer be tampered with. Specifically, we maintain an active device list in the whole system. Once the primary node of voting devices receives the signal of the epoch locking from the training level, he will call for a vote. Then,

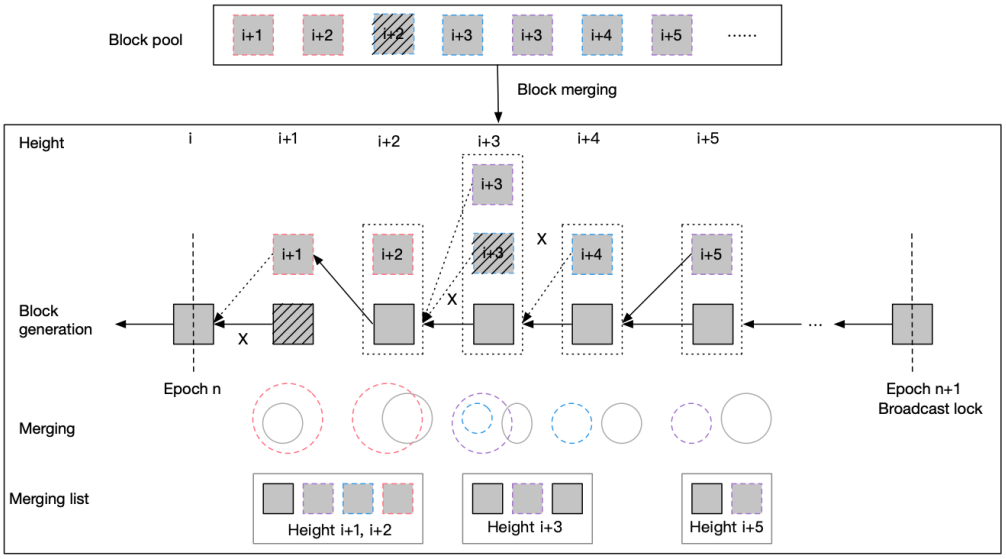


Figure 3.5: Block merging protocol.

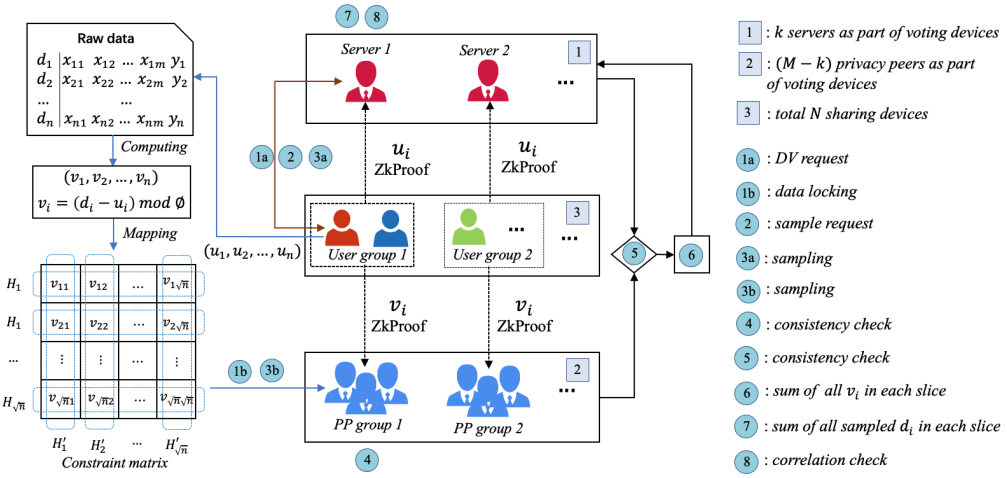


Figure 3.6: Structure of data verification with privacy protection.

all the voting devices will validate the block. If more than 2/3 of the voting devices pass the validation and signature of the new block, then the epoch will be locked, and all the updates in the previous blocks cannot be tampered with anymore.

3.4.3 Data Verification

One important task of the voting level is data verification. If a training device violates a rule, the voting layer should detect the violation and know which device

violated the rule. Accountability allows us to penalize malfeasant devices. For example, a training device has private data D=$\{d_1, d_2, \cdots, d_n\}$, and claims the Gaussian distributions G=$\{g_1, g_2, \cdots, g_q\}$ for all features F=$\{f_1, f_2, \cdots, f_q\}$ in its data. Our data verification objective is to check whether the training devices have data that match the distributions they claim. Meanwhile, we can access devices' private data.

Concretely, we need to verify the matching for all features. Then, the system splits f_j's value range($[Min(f_j), Max(f_j)]$) into some intervals, and the server calculates each interval's average value according to the corresponding Gaussian distribution g_j. If the user has data that matches the Gaussian distribution, then the same interval's average sampled by the user should be similar to the average calculated by the server. To do this, our data verification consists of two parts, namely *Consistency Check* and *Distribution Matching Check*

3.4.3.1 Consistency Check

To validate the 'data-distribution matching' without accessing data, we need three independent entities: the training device, privacy peer, and server. The training device has to send data-related information to both privacy peer and server. To validate the existence of private data and verify its consistency with server and privacy peer, we launch the P4P [92] for each training device.

The system requires the user to fetch some data points in each interval and transmit their u and v to the system. The system verifies the consistency of the data transmitted by the user, as shown in Algo. 1. Under the ZK-Proof scheme, the system will broadcast several challenge vectors for corresponding independent checks. We need the verifications for all challenge vectors to be successful. For each independent check, the training device calculates the corresponding u_i and v_i for datum d_i where $i \in \{1, n\}$ and send them to sever and peer respectively. Only if all Pair-Consist-Checks are successful is this independent check verified.

3.4.3.2 Distribution Matching Check

To verify whether D matches G without data breaching, take Feature f_j as an example. First, the user gives the Gaussian distribution g_j of Feature f_j in her data. To prevent the user from tampering with the information provided later, the user

Algorithm 1: Training Device Consistency Check

input : Datum d_i; system parameter ϕ; system challenge vectors C.
output : Consistency check result $Pass\text{-}Flag$

$Pass\text{-}Flag \leftarrow True$;
for $c_k \in C$ **do**
    $u_i \leftarrow$ Generate Random Vector as d_i;
    $v_i \leftarrow (d_i - u_i) \bmod \phi$;
    $CM_k \leftarrow X_k, Y_k, S_k, B_k, Z_k \leftarrow \text{Commit}(u_i, v_i, c_k, \phi)$;
    $Send(CM_k) \rightarrow Server, Peer$;
    $Pass\text{-}Flag \mathrel{*}=$ Pair-Consist-Check(Server, Peer, CM_k);
    $Pass\text{-}Flag \mathrel{*}=$ Pair-Consist-Check(User, Server, X_k);
    $Pass\text{-}Flag \mathrel{*}=$ Pair-Consist-Check(User, Peer, Y_k);
    if *Not* $Pass\text{-}Flag$ **then**
        Break;
return $Pass\text{-}Flag$;

needs to put all v_i into a recording matrix (under the shape of $\sqrt{n} * \sqrt{n}$), calculate the hash value of each row and column and transmit them to the system. Therefore, the system only needs $2 * \sqrt{n}$ hash values, significantly reducing the information required for verification.

The system first verifies the consistency of the data transmitted by the user, as mentioned before. After passing the *Consistency Check*, the server calculates its average value through Algo. 2. Both Algo. 1 and Algo. 2 are same to the processes in P4P [92] scheme. F is a specific nonlinear function.

At the same time, for each v in a specific interval, the system also requires the user to give all values of the same row and the same column in the recording matrix where v is located. This is to verify whether v is consistent with the original data. For f_j, if the verification of all intervals passes, then the verification of feature f_j is successful. For a user, if the verification of all features passes, the user's data verification is successful.

3.4.4 Measurement of data contribution

To facilitate our proposed framework, we need first to design an algorithm for the voting nodes to calculate the contribution of each user's dataset $\{\mathcal{D}_i\}_{i\in[n]}$ to the model update under the premise of privacy protection at each epoch. For this

Algorithm 2: Training Device Data Summation

input : u set of all data U; v set of all data V; system parameter ϕ; initial summation A.

output : Updated summation A'

2: $\mu, \nu \leftarrow 0$; $A \leftarrow 0$;

5: **for** $u_i \in U$ **do**

7: $\quad \mu \mathrel{+}= u_i \bmod \phi$;

9: **for** $v_i \in V$ **do**

11: $\quad \nu \mathrel{+}= v_i \bmod \phi$;

13: $s \mathrel{+}= (\mu + \nu) \bmod \phi$;

15: $A' = F(s, A)$;

17: **return** A'

purpose, we design an algorithm to indirectly calculate the data contribution of each training node based on data summary (e.g., Gaussian fitting). For each training node, Gaussian fitting is performed locally first, and then the fitting result $G_i, i \in n$ is sent to the voting layer. At each epoch settlement, the voting layer will calculate the contribution $r_i, i \in n$ of each training node according to the Gaussian fitting results of all nodes participating in the settlement at this epoch.

3.4.4.1 Sharing layer: Gaussian fitting

The Gaussian fitting G_i for each user's private dataset is defined as a set of Gaussian functions $\{G_{ik}\}_{k\in[1,q]}$, where q is the number of features after the feature extraction from raw dataset $\mathcal{D}_i$. Specifically, after feature extraction, features are independent, so we carry out Gaussian fitting $G_{ik}, k \in [1, q]$ for each feature, respectively. Therefore, the result of Gaussian fitting of original data G_i is the combination of a series of feature Gaussian fitting result $\{G_{ik}\}_{k\in[1,q]}$. The Gaussian fitting for the raw dataset is shown in Algorithm 3.

3.4.4.2 Voting layer: contribution calculation

In the settlement stage, the voting layer needs to update the global model and allocate benefits according to the proportion of user data contribution. Since the voting layer only has the result of each user's Gaussian fitting result $G_i = \{G_{i1}, G_{i2}, ..., G_{iq}\}$ and the number of user volumes $count_i, i \in [n]$, we adopt random sampling method to approximate the real data distribution according to the all Gaussian fitting results

Algorithm 3: Training: Gaussian fitting of private dataset

input : User private data $\mathcal{D}_i, i \in [n]$
output : Result of Gaussian fitting $G_i = \{G_{i1}, G_{i2}, ..., G_{iq}\}$, G_{iq} is the Gaussian distribution of feature q

F_i = feature_extraction($\mathcal{D}_i$); /*GNN*/
q = columns of F_i;
for $k = 1$ *to* q **do**
    G_{ik} = Gaussian_fitting(F_{ik});
return $G_i = \{G_{i1}, G_{i2}, ..., G_{iq}\}$;

$G^k = \{G_{mk}\}, m \in [n]$ on specific feature $k, k \in [1, q]$ to indirectly calculate the proportion of each user's data contribution $\{r_{1k}, r_{2k}, ..., r_{nk}\}$ on the feature k. Therefore, the proportion of user data contribution $r = \{r_1, r_2, ..., r_n\}$, $\sum_{i\in[1,n]} r_i = 1$ and $r_i = \sum_{k\in[1,q]} r_{ik}/q$.The data contribution calculation method is shown in algorithm 4.

Algorithm 4: Voting: a privacy protection method for data contribution calculation

input : Gaussian fitting results set $\{G_i\}$ of all users' data $\{\mathcal{D}_i\}_{i\in[n]}$; data volumes of each user $\{count_i\}$; sample rate r_s; slices number n_s; Gaussian range g_r
output : Percentage of all users' data contribution $r = \{r_1, r_2, ..., r_n\}$

q = columns of G_i; /*Number of features*/
for $k = 1$ *to* q **do**
    $G^k = \{G_{mk}\}$, $m \in [n]$;
    min, max = get_max_min(G^k);
    $range = (max - min)/n_s$; /*Range of each slice*/
    for $i = 1$ *to* n **do**
        $n_s = count_i * r_s$; /*Number of samples for user i */
        $p_i = \{s_j : n_j\}$ = random_samples(G_{ik}, g_r,n_s);
        /*s_j: number of slice; */
        /*n_j: number of samples in slice s_j;*/
    $\{r_{1k}, r_{2k}, ..., r_{nk}\}$ = feature_contribution($\{p_1, p_2, ..., p_n\}$);
for $i = 1$ *to* n **do**
    $r_i = \sum_{k\in[1,q]} r_{ik}/q$;
return *Proportion of all users' data contribution* $r = \{r_1, r_2, ..., r_n\}$;

3.5 PoD Evaluation

In this section, we evaluate the PoD consensus with the proposed consensus, mainly including resilience (i.e., liveness, safety, and fault tolerance), performance and governance. Furthermore, we analyze the PoD's anti-attacking, aiming at specific attacking modes.

3.5.1 Undelying model

Firstly, PoD doesn't explicitly state the synchrony model in which the protocol operates but dictates that training devices need to wait for the verification result and epoch-locking result from the voting level. Thus, we consider the protocol partially synchronous because of timeouts for data verification and epoch locking. Moreover, PoD is designed for the permissionless network, which is open-access to everyone, including dishonest nodes who have a great probability of behaving maliciously. Therefore, PoD is Byzantine tolerant. Meanwhile, PoD leverages Gossip to broadcast information through the partially connected P2P network. And the data is organized by Blockchain.

3.5.2 Resilience

Firstly, in the training layer, all nodes train the model with a private dataset and then broadcast the blocks to the whole network, so the timing model is asynchronous for the training layer. Moreover, the PoD uses the PBFT to realize the verification work for the voting layer, so the timing model is partially synchrony. Last but not least, PoD dictates that training devices must wait for the verification and epoch-locking results from the voting layer. Thus, we consider the protocol partially synchronous because of timeouts for data verification and epoch locking. So, the PoD can ensure the system's liveliness.

BFT vote only happens every epoch, so it can not prevent the temporary forks of the asynchronous training layer. So the agreement is probabilistic, and the finality is temporary, but the probability increases with epoch length between two consecutive epochs due to the "longest chain". However, after epoch locking, the agreement is deterministic, and the finality is permanent. The validity is deterministic due to data verification and signature. Specifically, data verification can guarantee the

correctness and consistency of raw data, and a signature from the sender's private key can block the possibility of block tampering during propagation. The termination of the PoD is deterministic because of the repeated submission of each node during the merging process and voting threshold. Repeated submission guarantees that if the node doesn't receive the epoch-locking flag from the voting network, he/she will keep submitting the locking requests to the voting network. Moreover, because of the partially synchronous assumption of communication between training and voting layers, the number of nodes is fixed at a certain time. Once the merging list reaches the threshold, the locking works. All of these key designs can guarantee the safety of the PoD consensus algorithm.

In the PoD, nodes in both layers participate in the consensus-reaching process. The voting layer is a classic PBFT model that tolerates no more than $\lfloor m \rfloor$ faulty nodes based on the conclusion $Z \geq 3f + 1$ [13]. In the sharing layer, if it is a Proof-based consensus model, we need to analyze the threshold of consensus-reached sharing nodes required to ensure the security and liveness of the whole system. During the consensus-reaching process, as there is a voting threshold τ for the voting layer to validate the epoch-locking application, the $\tau \leq 1 - f$.

3.5.3 Performance: communication complexity

The PoD model is proposed in Fig. 3.1, where the sharing layer has n sharing nodes, and the voting layer has m voting nodes. In the voting layer, for peer-to-peer communication, the communication complexity is the square of the node number. Thus, the required complexity C_v to reach consensus is $C_v = m^2$. Moreover, we investigate the merging and locking process with minimum communication complexity for a proof-based system $C_t = n$. But based on the design of PoD, we can get that m is much less than n, so the communication complexity of the PoD is n.

3.5.4 Governance

For the governance of the PoD consensus, we mainly focus on decentralization, incentives, and supervision. For decentralization, it depends on PoD; the certainty is probabilistic, and the coverage is full, but depending on the BFT vote, the certainty

is deterministic, and the coverage is partial. The cost of the PoD consensus is mainly the deposit for training authorization of the training device. Moreover, for the training device, the benefit is a token reward from data contribution. For the voting device, the benefit is the token reward from epoch voting and data verification. For the supervision, the reward is the "Finder's fee" of the submitter of the slashing condition of the BFT vote, and the penalty is the forfeit of the deposit for malicious behaviours of training devices. Therefore, the data verification, deposit, and reward allocation guarantee the fairness of the PoD, and the incentive from tokenomics and fairness guarantees the liveliness of the PoD.

3.5.5 Proof of Correctness

Table 3.1: Fault tolerance of PoD

Challenges	**Attacks**	**Defense**
Challenge I (Model training)	Leave out of epoch settlement	Fault-tolerance ratio f; Voting threshold $\tau \leq 1 - f$
	Delay epoch (liveliness)	Voting threshold $\tau \leq 1 - f$
	DoS attack	The priority is inversely proportional to the number of requests
	Eclipse attack	Voting threshold $\tau \leq 1 - f$
	Finality overturn	Finality locking at epoch
Challenge II (Value settlement)	Data falsification	Data verification
	Data domination / shadowing	Data contribution calculation tolerating overlap
	Tamper with block data	Asynchronous encryption; Update pool to be processed for public key propagation delay
	Value settlement overturn	Value settlement locking at epoch

To prove the correctness of PoD, we must first prove that we have successfully addressed the two challenges which we have analyzed in the section 3.3. For challenge one, we need to prove that our model training is correct. That means our decentralized version can achieve what a centralized version can do. Then, we are going to examine possible ways that a malicious node can tamper with the system. Similarly, with the training nodes A, B, C and D as examples, as shown in Fig. 3.3, suppose the D is the malicious node. What could be the possible ways D can ruin the system.

Firstly, D can train fast to rush into the epoch locking and leave out all others in the system to achieve the most benefits. Our solution is that we have the Voting threshold $\tau \leq 1-f$ in the voting layer to make sure that less than fault-tolerant ratio malicious nodes will not be able to dominate the system. Secondly, D can run slow intentionally to delay epoch locking, which ruins the system's liveliness. Similarly, we have the voting threshold in the voting layer to ensure that malicious nodes with a lower fault-tolerant ratio cannot affect the settlement of the epoch. Next, D can construct multiple transactions to submit epoch settlement requests redundantly to block the primary node in the voting layer (i.e., DoS attack). D may continuously broadcast his demand to the voting layer and not perform any subsequent operations. Our solution sets a request priority, which is inversely proportional to the number of locking requests. Also, D can collude other training nodes to launch Eclipse attack, our voting threshold can protect the system from this attack. At last, depending on the epoch locking at epoch settlement, the finality can not be overturned.

For challenge two, we need to prove that our value settlement is correct. That means our value allocation is fair and cannot be overturned. Similarly, D can launch data falsification attacks, and our solution is to have data verification to ensure data consistency during model training. In order to encourage training nodes to attend the system, especially those with small datasets, our data contribution calculation can tolerate overlap to protect fairness from data domination or data shadowing. Moreover, our solution uses asynchronous encryption to protect the block data consistency from tampering. At last, depending on the epoch locking at epoch settlement, the value settlement can not be overturned.

3.6 Experiments and Evaluation

In this section, we first declare the experimental setup and then evaluate the performance and fairness of PoD on dataset ImageNet [20] with varied data allocation for all training nodes.

3.6.1 Experimental setup

3.6.1.1 Dataset

The publicly accessible dataset ImageNet [20], widely used in the image classification field, is used for our performance and fairness validation in this chapter. We extracted a subset from ImageNet, which comprises about 1,500,000 images and a total of 1,000 categories, with a training set of 1,400,000 examples and a test set of 100,000 examples, respectively.

3.6.1.2 Dataset allocation

In our experiments, we aim to verify system performance and system fairness in different data distribution scenarios. We mainly simulate the practical application scenarios from three aspects, i.e., data variation (i.e., static vs. dynamic), data overlap (i.e., data intersection ratio) and data volume (i.e., small vs. large dataset). (1) Data variation: We mainly consider static and dynamic data cases and design data that increase the rate of dynamic situations. (2) Data overlap: We also consider the intersection of user data. *Intersection ratio* represents data overlap, and in the previous discussion, the greater the data overlap, the lower the data contribution. (3) Data volumes: Moreover, the amount of user data can greatly influence consensus results, so we also take data volumes into account during the validation process. Specifically, We use the *uneven rate* to measure the unbalanced volume distribution of user datasets.

Therefore, we designed two ways for data allocation for each data variation scenario in our experiments, i.e., data volume imbalance and overlap rate difference allocations. (1) data volume imbalance: each user is assigned a different number of training samples $\{D_i\}_{i\in[n]}$ with a uniformly random distribution over 1,000 classes, and there is no overlap of any two datasets, i.e., $D_i \cap D_j = \emptyset, 1 \leq i < j \leq n$. We use the uneven rate $r_{uneven} = 1/n\sqrt{\sum_{i\in[n]}(count_i - avg)^2}$, where $avg = 1/n \sum_{i\in[n]} count_i$. (2) Overlap rate difference: each user is assigned the same number of training samples with a uniformly random distribution over 1,000 classes, but there exists an overlap between any two datasets, i.e., $D_i \cap D_j \neq \emptyset, 1 \leq i < j \leq n$. We use the overlap rate $r_{overlap} = n_{overlap}/\sum_{i\in[n]} count_i$, where $n_{overlap}$ represents the data number of all intersections.

It should be emphasized here that, for practical application purposes, we only consider independent identical distribution (i.e., i.i.d) data in our experiment. Moreover, In actual applications, user data basically remains unchanged, so our experiments are mainly focused on static scenarios.

3.6.1.3 Neural network structure

For the training models used to perform the image classification tasks, we use the convolutional neural network (CNN) for ImageNet, which contains two 5×5 convolutional layers (each layer is followed with a batch normalization and 2×2 max pooling), a fully connected layer with ReLu activation and a final softmax output layer. The CNN model is mended from [61]. Unless otherwise specified, some important hyperparameters in our experiments are set as Table 3.2.

Table 3.2: Experimental parameters setting of PoD

Layer	**Parameter**	**Numerical value**
Sharing layer	#Training nodes	4000
	#Byzantine nodes	0; 1000 (1/4); 2000 (1/2)
Voting layer	#Voting nodes	10
	#Byzantine nodes	≤ 3 (PBFT fault tolerance)

3.6.1.4 Baseline and Metric

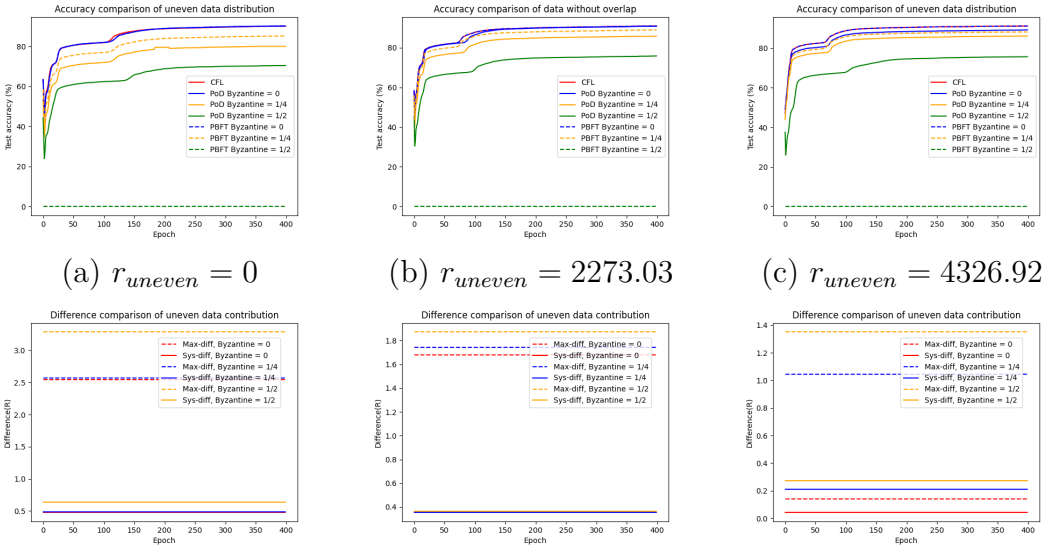
In this work, we compare our PoD with a conventional centralized federated learning method (e.g., Fedavg [68]) and PBFT. For the FedAvg and PBFT implementation in this chapter, a centralized topology with the same number of training nodes as the decentralized topology is considered, where more than 80% of all training nodes are ensured to participate in each training round. To make the result clear, we design two metrics, including *Acc* and *Diff*, to indicate the performances of different methods and the fairness of the proposed framework. Specifically, for performance, we respectively test the global model of each method and get the test accuracy (i.e., *Acc*) of the global model after each epoch. Generally, a superior federated learning method is expected to obtain a higher *Acc*. For fairness, we calculate theoretical and actual rewards for each training node respectively and get the maximum reward difference (i.e., *Max-diff*) among all training nodes and the total reward difference (i.e., *Sys-diff*) of the system. Generally, a fairness federated learning system is expected to obtain a smaller *Max-diff* and *Sys-diff*). Moreover, we

consider the quantitive metric f-resilient to characterize the resilience PoD protocol. A consensus protocol is said to be f-resilient if it can tolerate an (adaptive) adversary that corrupts up to f devices. If $3f + 1 = n$, the consensus protocol is said to be optimally resilient. Through the paper, we focus on the optimally resilient DFL framework against an adaptive adversary.

3.6.2 Performance and fairness on static data

Fig. 3.7 and Fig. 3.8 respectively show the performances and fairness of PoD with static data allocation under different data volume distribution and data overlap rates. The hyper-parameters in this experiment follow Table 3.2.

3.6.2.1 Uneven data without overlap



(a) $r_{uneven} = 0$ (b) $r_{uneven} = 2273.03$ (c) $r_{uneven} = 4326.92$

(d) $r_{uneven} = 0$ (e) $r_{uneven} = 2273.03$ (f) $r_{uneven} = 4326.92$

Figure 3.7: The result of performance and fairness under different data volume distribution

Fig. 3.7 shows the result with static data and different data volume distributions. From the following three perspectives, we can draw different conclusions.

First, the proposed PoD consensus protocol has a close performance to conventional centralized federated learning (i.e., CFL) and PBFT in terms of *Acc* under almost uniform data volume distribution without Byzantine nodes, albeit slightly inferior under extremely uneven data distribution. Specifically, from Fig. 3.7(a),

Fig. 3.7(b), the PoD finally harvests 92.02% and 92.01% accuracy, respectively, under uniform and slightly uneven data distribution. This is closer to the result of CFL with 92.03% and 92.03% accuracy and PBFT with 92.03% and 92.03% accuracy under the same data distribution. However, from Fig. 3.7(c), PoD harvests 87.01% accuracy under extremely uneven data distribution, which is inferior to the result of CLF with 92.02% under the same data distribution. Since under almost uniform data distribution, all training nodes have a closer training speed. Therefore, regardless of the voting threshold, all users' updates can participate in the settlement at each epoch and be written to the blockchain. As a result, PoD training results are close to those of conventional CFL. Contrarily, extremely different training speeds exist under extreme data volume distribution, which may severely rely on the threshold of voting limits. If the threshold is very high, the waiting time of the system becomes longer, and more training time is given to the big data training node. If the threshold is very low, the big data node does not have enough time to complete training before the system epoch settlement and cannot participate in the system settlement. This is exactly why PoD underperforms conventional CFL under extreme data volume distribution in this experiment.

Second, the PoD consensus protocol is more fair in the case of almost uniform data distribution but less fair in the case of extremely uneven data. That is, the *Max-diff* is close to the *Sys-diff* under the almost uniform data distribution, and both are significantly smaller than extremely uneven data distribution. Specifically, from Fig. 3.7(c) and Fig. 3.7(d), the PoD has 0.042 and 0.045 system differences, and the corresponding 0.071 and 0.073 maximum differences under the two almost uniform data distribution, which are close to the system differences and much less than 1, while has 1.4 system difference and 1.8 maximum difference under the extremely uneven data distribution as shown in Fig. 3.7(e). On the one hand, because the PoD consensus uses Gaussian fitting to approximate the calculation of user contribution, there is a small system difference. However, since the maximum difference is similar to the system difference, the contribution measurement deviation of each user is small, and the system is relatively fair. On the other hand, as the data distribution is extremely uneven, big data users cannot participate in settlement under the pledge condition, so there will be significant inequity.

Third, the attacks from Byzantine nodes can significantly affect the system's

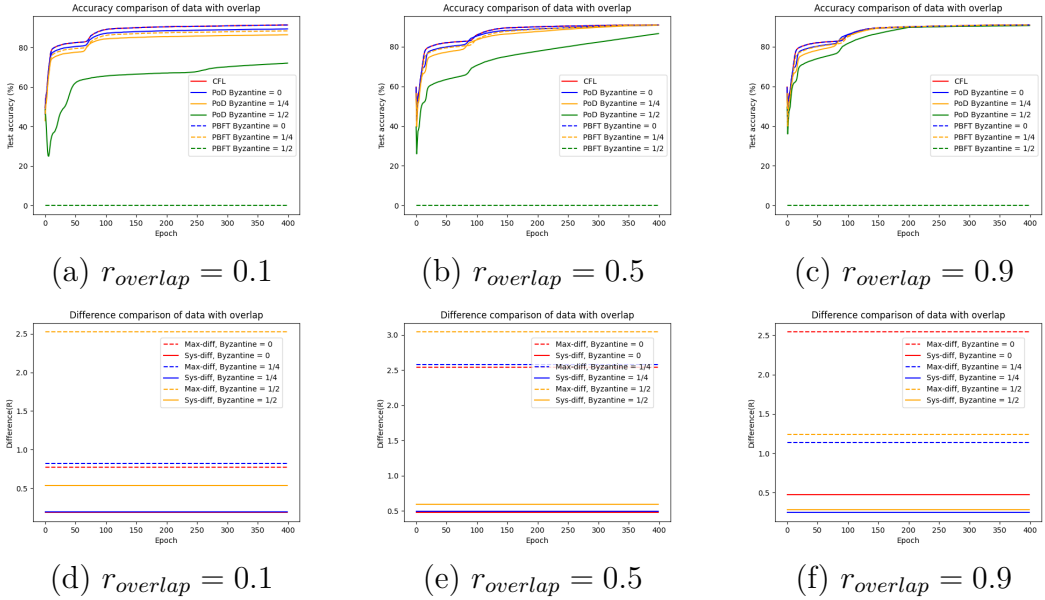
performance and can result in significant unfairness. Rather, these Byzantine nodes will suffer profit losses. Specifically, from Fig. 3.7(a), Fig. 3.7(b) and Fig. 3.7(c), it can be seen that the test accuracy of PoD-Byzantine is significantly inferior to that of PoD without Byzantine nodes. This slight discrepancy can be interpreted as a loss of the Byzantine nodes' own data that does not participate in the computation and the lower threshold τ that honest nodes cannot participate in the settlement, but the attacking behaviour of Byzantine nodes can significantly affect the system. Moreover, from Fig. 3.7(d), Fig. 3.7(e) and Fig. 3.7(f), it can be seen that the system difference and max difference of PoD-Byzantine are larger than those of PoD without Byzantine nodes. In order to resist attacks of Byzantine nodes and guarantee the liveliness of the system, PoD-Byzantine sets a lower threshold τ than honest PoD that causes several honest training nodes to be unable to participate in epoch settlement, which results in significant unfairness. Moreover, the deposits of Byzantine nodes are also awarded to honest nodes participating in the epoch settlement.

In summary, the PoD consensus protocol performs closely to conventional CFL and PBFT under almost uniform data contribution and has excellent fairness without Byzantine nodes. PoD consensus protocol is Byzantine tolerance, and the tolerant ability is relevant to the voting threshold τ. The fact is, there is a game between fairness (i.e., higher threshold τ) and liveliness (i.e., lower threshold τ) of the Byzantine system, and we need to look for a balance to maximize the performance of the system.

3.6.2.2 Uniform data with overlap

In this section, we investigate how the data overlap affects the performance and fairness of the PoD consensus protocol. Fig. 3.8 presents the result under the uniform data with overlap.

From the perspective of *Acc*, the proposed PoD consensus protocol has a close performance to the conventional CFL and PBFT under uniform data volume distribution with overlap. Take the result on overlap = 50% as an example(Fig. 3.8(d)). The PoD finally reaches 92.03% accuracy, which is close to the CFL with 92.07% and PBFT with 92.02%. Byzantine nodes can only affect the speed of convergence but cannot affect the training result. Intuitively, the PoD consensus protocol has a novel



(a) $r_{overlap} = 0.1$ (b) $r_{overlap} = 0.5$ (c) $r_{overlap} = 0.9$

(d) $r_{overlap} = 0.1$ (e) $r_{overlap} = 0.5$ (f) $r_{overlap} = 0.9$

Figure 3.8: The result of performance and fairness under different overlap rate

contribution measurement system to support the *global aggregation* phase. User data contribution measurement system measures user aggregating factors from data volume and data uniqueness comprehensively, unlike the conventional CFL, which only considers the user data volume. Therefore, the PoD consensus can effectively eliminate the impact of data redundancy on system performance.

Besides, for the *Sys-diff* and *Max-diff* of honest PoD shown by Fig. 3.8(d), Fig. 3.8(e) and Fig. 3.8(f), results similar to that under uniform data volume distribution as shown in Fig. 3.7(d). As shown in the figures, the honest PoD also has similar system differences and maximum differences under the uniform data distribution, and both differences are much less than 3.0. This result indicates that the system's fairness is closely related to the distribution of data volume but not to the data overlap. Finally, Pod consensus can not defend against *data redundancy attack*. System performance and fairness are significantly reduced.

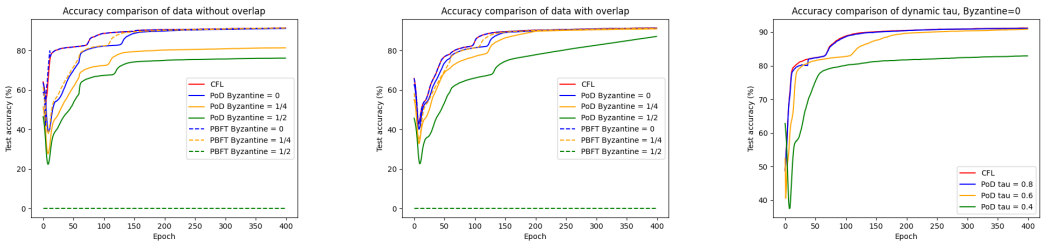
In summary, with the support of a user data contribution measurement system, data overlap does not significantly affect the performance and fairness of PoD. However, because the PoD consensus does not consider the quality of model updates, the system cannot resist *data redundancy attack*.

3.6.3 Influences of dynamic changes

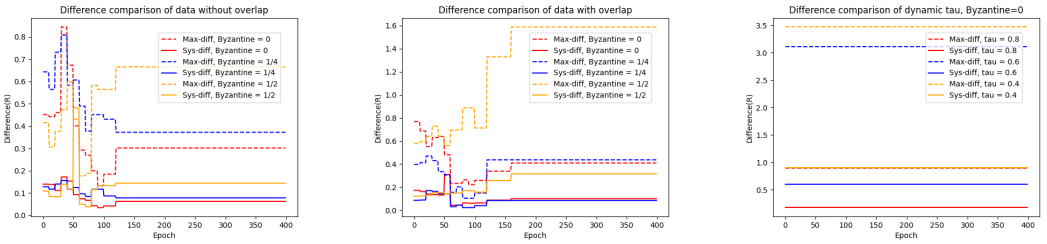
The section 3.6.2 has shown the result of PoD consensus on static datasets and voting threshold τ and declared its practicality under both data volumes and data overlap distribution. In this section, we test the performance and fairness on dynamic datasets and voting threshold τ and show the result in Fig. 3.9. The hyper-parameters in this experiment follow Table 3.2.

3.6.3.1 Dynamic data volume without overlap

Fig. 3.9(a) and Fig. 3.9(d) present the experimental result under the dynamically uneven data distribution without overlap. All users start training with the same amount of data but will increase at different exponential rates, such as $2^n, 3^n$, and eventually form an extremely uneven distribution.



(a) Without overlap (b) With overlap (c) Dynamic threshold τ



(d) Without overlap (e) With overlap (f) Dynamic threshold τ

Figure 3.9: The result of performance and fairness under dynamic data distribution and threshold

First, for the *Acc*, at the beginning, PoD, CFL and PBFT converge at the same rate, but inflexion points of convergence occur as data volume differentiates, as shown in Fig. 3.9(a). This is because the threshold plays a role, and some nodes with a large amount of data are too late to participate in the settlement, thus significantly reducing the convergence speed. Subsequently, these big data nodes will not have the opportunity to participate in the settlement. Hence, the accuracy of the final training

result is slightly lower than the traditional CFL, which gets a final test accuracy of 92.34%, while PoD only reaches 90.34%. Meanwhile, as discussed in section 3.7, the differentiation of data volume will lead to system unfairness. Therefore, as the number of training epochs increases, user data volume differentiation and system fairness decrease, as shown in Fig. 3.9(d).

Second, the attacks from Byzantine nodes can not significantly affect the system's performance but can result in significant unfairness in the dynamic environment. From Fig. 3.9(a), it can be seen that the test accuracy of PoD-Byzantine is close to that of PoD without Byzantine nodes. This slight discrepancy can be interpreted as a loss of the Byzantine nodes' own data that does not participate in the computation and the lower threshold τ that honest nodes cannot participate in the settlement, and the attacking behaviour of Byzantine nodes does not affect the system at all. Moreover, from Fig. 3.9(d), it can be seen that the system difference and max difference of PoD-Byzantine are larger than those of PoD without Byzantine nodes. In order to resist attacks of Byzantine nodes and guarantee the liveliness of the system, PoD-Byzantine sets a lower threshold τ than honest PoD that causes several honest training nodes to be unable to participate in epoch settlement, which results in significant unfairness. Moreover, the deposits of Byzantine nodes are also awarded to honest nodes participating in the epoch settlement.

In summary, the PoD consensus protocol had a close performance to conventional CFL before the huge divergence in data volume and had excellent fairness without Byzantine nodes. However, the performance and fairness of the system decrease obviously after the user data volume is greatly differentiated. Meanwhile, the PoD consensus protocol is Byzantine tolerant, and the tolerant ability is relevant to the voting threshold τ.

3.6.3.2 Dynamic data volume with overlap

Fig. 3.9(b) and Fig. 3.9(e) present the experimental result under the dynamically uneven data distribution with overlap. All users start training with the same amount of data without overlap but will increase at different exponential rates with 50% overlap, and eventually form an extremely uneven distribution with a large amount of overlap.

From the perspective of *Acc*, the proposed PoD consensus protocol also outper-

forms the conventional CFL before huge user data volume differentiation, as shown in Fig. 3.9(b). The main reason is that the novel contribution measurement system greatly supports the *global aggregation* phase. User data contribution measurement system measures user aggregating factors from data volume and data uniqueness comprehensively, unlike the conventional CFL, which only considers the user data volume. Therefore, the PoD consensus can effectively eliminate the impact of data redundancy on system performance. However, when the amount of data is greatly differentiated and affected by the voting threshold τ, the advantage of the PoD consensus protocol is not obvious.

Besides, for the *Sys-diff* and *Max-diff* of honest PoD shown by Fig. 3.9(e), results similar to that under uniform data volume distribution as shown in Fig. 3.9(d). As shown in the figure, the honest PoD also has similar system differences and maximum differences under the uniform data distribution. This result indicates that the fairness of the system is closely related to the distribution of data volume but not to the data overlap. Finally, Pod consensus can not defend against *data redundancy attack*. System performance and fairness are significantly reduced.

To sum up, with the support of a user data contribution measurement system, data overlap does not significantly affect the performance and fairness of PoD. However, because the PoD consensus does not consider the quality of model updates, the system cannot resist *data redundancy attack*.

3.6.3.3 Dynamic threshold τ

From the above experimental results, we can obtain that there is a game between fairness (i.e., higher threshold τ) and liveliness (i.e., lower threshold τ) of the Byzantine system, and we need to look for a balance to maximize the performance of the system. To figure out how the voting threshold τ affects our solution, we also test the *Acc*, *Sys-diff* and *Max-diff* under dynamic extremely uneven data distribution with overlap, and the experimental results are shown in Fig. 3.9(c) and Fig. 3.9(f).

From the perspective of *Acc*, the proposed PoD consensus protocol also outperforms the conventional CFL before huge user data volume differentiation, as shown in Fig. 3.9(c). The main reason is that the novel contribution measurement system greatly supports the *global aggregation* phase. User data contribution measurement

system measures user aggregating factors from data volume and data uniqueness comprehensively, unlike the conventional CFL, which only considers the user data volume. Therefore, the PoD consensus can effectively eliminate the impact of data redundancy on system performance. However, when the amount of data is greatly differentiated and affected by the voting threshold τ, the advantage of the PoD consensus protocol is not obvious.

Besides, for the *Sys-diff* and *Max-diff* of honest PoD shown by Fig. 3.9(f), results similar to that under uniform data volume distribution as shown in Fig. 3.9(f). As shown in the figure, the honest PoD also has similar system differences and maximum differences under the uniform data distribution, and both differences are much less than 1. This result indicates that the system's fairness is closely related to the distribution of data volume but not to the data overlap.

In summary, the dynamic threshold can effectively reduce the inequity caused by uneven data distribution and improve the accuracy of the training result. But there will be temporary inequities.

3.7 Related work

In this section, we first provide a brief introduction to the development of federated learning, and then summarize some related works about consensus algorithms in decentralized federated learning.

3.7.1 Blockchain-enabled Federated Learning

Konecn et al. proposed Federated Learning whose goal is to train a high-quality centralized model while training data remains distributed over a large number of clients [49]. Subsequently, FL is applied in many scenarios like video analysis, information inspection, and classification, and credit card fraud detection and so on [14, 31, 18] while keeping personal data sensitivity safe. Besides, the theoretical studies of convergence [22, 90], network latency [94], or malicious attacks [93, 83] on FL are also active fields.

Meanwhile, The centralized federated server has been challenged and questioned growly in these years. It is a natural thought that keeps the concept of server at a minimum or even avoids it completely. The study of [39] assumed that the data

remains at the edge devices, but it requires no aggregation server or any central component. Hu et al. [41] proposed a segmented gossip approach, which fully utilizes node-to-node bandwidth and then can achieve a convergence efficiently.

Moreover, decentralization may be the most direct way to avoid the risks in centralized federated learning. Blockchain, a distributed ledger technique, can store historical operations and keep them tamper-resistant. With the aim of the blockchain, collaborative machine learning methods can eliminate the centralized server and improve security. It is reasonable to assume that the clients in FL might be malicious. Therefore, the local updates from all clients should be recorded under blockchain-based FL settings. Nguyen [67] presented an overview of the fundamental concepts and explores the opportunities of FLchain in MEC networks which systematically analyzes the challenges and opportunities when Federated learning meets blockchain, and studies [47, 60, 72, 74, 59] proposed blockchain-based federated learning architecture to solve these challenges, such as focus on convergence speed, stability, attacks and so on. These blockchain-based learning methods can effectively record the nodes' performance to reduce malicious attacks. However, there are still several main challenges, such as consensus efficiency, model security, framework scalability and so on.

3.7.2 Consensus for Blockchain-enabled FL

With the development of blockchain-enabled federated learning, a series of new consensus algorithms have been proposed to support decentralized federated learning systems.

Bao et al. [5] proposed a public blockchain-based FL architecture, which provides trusty consensus based on nodes' data amount and historical performance. Yuzheng et al. [58] proposed a blockchain-based decentralized federated learning framework called BFLC with committee consensus which uses blockchain for the global model storage and the local model update exchange. Zhikun et al. [15] proposed a new DFL implementation DACFL with a first-order dynamic average consensus FODAC method to track the average model in the absence of the PS. Xidi et al. [75] proposed an energy-recycling consensus algorithm PoFL reinvest the energy wasted in PoW puzzles computing to federated learning problems. Wang et al. [89] proposed

a energy-recycling consensus mechanism named platform-free proof of federated learning (PF-PoFL) to leverages the computing power originally wasted in solving hard but meaningless PoW puzzles to conduct practical federated learning (FL) tasks. However, these works leave the Byzantine nodes untouched.

3.8 Conclusion

Over-reliance on the central PS possibly paralyses the federated learning when the server breaks down. To alleviate this single-point failure in conventional FL, in this paper, we devise a novel decentralized federated learning framework coined as Proof-of-Data (i.e., PoD) consensus protocol to solve the consistency and liveliness problem in decentralized and open-access systems with Byzantine nodes. To confirm the feasibility of PoD, we also deliver a theoretical analysis on the premise of some assumptions, which offers a liveliness and safety guarantee of our solution. Besides, we design specific experiments on MNIST under static and dynamic allocations and analyze the performance and fairness of the PoD. The results verify the effectiveness and fairness of PoD under various data distributions and declare that PoD can maintain outstanding performance and fairness in most cases.

Chapter 4

Evaluative Framework for DAO Governance

Decentralized Autonomous Organizations (DAOs) have gained widespread attention in academia and industry as potential future models for decentralized governance and organization. This chapter addresses a significant gap in the current understanding of decentralization within Decentralized Autonomous Organizations (DAOs). While many theoretical discussions generalize governance models, there is a lack of in-depth analysis of the unique governance structures and their inherent decentralization mechanisms in individual DAOs. To bridge this gap, the chapter presents two primary contributions. First, it offers a systematic investigation of the degree of decentralization for DAO based on its entire transaction, voting and delegation history. Second, the study introduces a "decentralization degree-based governance analysis", a new framework aimed at guiding future quantitative analyses of governance. This system is based on two pillars: a critical evaluation of the degree of decentralization and a stratification approach to analyze governance. The chapter further demonstrates the application of this framework by analyzing the governance models of the MakerDAO. Through these analyses, the chapter provides insights into the strengths and vulnerabilities of DAO's decentralization mechanisms, offering a comprehensive perspective on DAO governance.

4.1 Introduction

DAOs represent organizational structures designed to offer an alternative, decentralized form of governance for decentralized applications (dApps) operating on

Distributed Ledger Technologies (DLTs). The intention of DAOs is to circumvent central authorities and hierarchical structures that are prevalent in traditional organizations and democratize the decision-making process by distributing voting rights through so-called governance tokens to community members [17].

It is well known that governance tokens are distributed primarily to team members, early investors, or protocol treasuries [6], and decision-making power can be concentrated in the hands of a few [7]. Earlier research has provided preliminary evidence on the involvement of DAO team members and developers in DAOs decision-making processes [81, 42, 34, 48]. However, there is a surprising gap in studies that systematically investigate the role of users in the governance of DAOs and how they determine their trajectories. Meanwhile, due to the early stages of development of Decentralized Finance (DeFi) and their DAOs, there are some pitfalls to their current decentralization status. Many of the DeFi protocols, such as Automated Market Makers, are indeed operating automatically in an on-chain smart decentralized environment. However, even with their decentralized operation, it does not necessarily mean that their governance power is equally distributed [81, 48]. The relevant work reveals a notable gap in the quantitative assessment of decentralization within these entities.

In this chapter, we focus on the state of the prominent DAO governance systems on the Ethereum blockchain: MKRDAO. In particular, we examine how the voting power is distributed in this system. Using a comprehensive dataset of all governance token holders, delegates, proposals and votes, we analyze who holds the voting power and how this power is being used to influence governance decisions. We identify the following contributions to our work:

- We proposed a novel general DAO governance evaluative framework with four layers: 1) generated networks, which are built networks with raw governance data such as voting data, delegation data, transaction data and so on; 2) network metrics, which are general network measurement methods; 3) decentralization metrics, which are measurements of the distribution of network metrics; and 4) Governance evaluation, which is evaluation on voting power in governance system based on decentralization degree.

- We demonstrate our evaluation framework on the entire transaction, voting

and delegating history of MakerDAO with daily granularity for an insight into the fullest picture of MakerDAO. Specifically, We have examined the decentralization of MakerDAO from both static and dynamic perspectives. The analysis is given both to capture the degree of decentralization of voting tokens at important snapshots along the timeline and to understand how the tokens have flown through the transaction network to connect observations at static snapshots.

- All our analyses focused on the quantitative measurement of the decentralization of MakerDAO. Instead of indiscriminately adopting network measures on the transaction, voting and delegating networks, we proposed carefully chosen metrics based on their effectiveness in reflecting the decentralization of MakerDAO governance. We further illustrate these findings by detailed case studies to align with real-life events in the MakerDAO governance community, lending interpretability to our data analytical results.

- we reveal that the majority of voting power is concentrated in the hands of a small number of addresses, we rarely observe these powerful entities overturning a vote by choosing a different outcome than that of the overall community and less influential voters.

The rest of this chapter is organized as follows. We first introduce our proposed evaluative framework in section 4.2, and then we describe the MakerDAO transaction, voting and delegating data in Section 4.3. In Section 4.4, we introduce two types of decentralization degree computing methods and we conduct governance evaluation from three perspectives based on the fourth of the proposed evaluative framework in section 4.5. Finally, we conclude the chapter in Section 4.6.

4.2 Overview of Evaluative Framework

The novel designs of DAO governance systems and the large amount of power they hold make it compelling and relevant to study how these systems work and how decisions are reached. Who holds the voting power in these systems, and how do the actors influence the decisions? To systematically evaluate the DAO governance

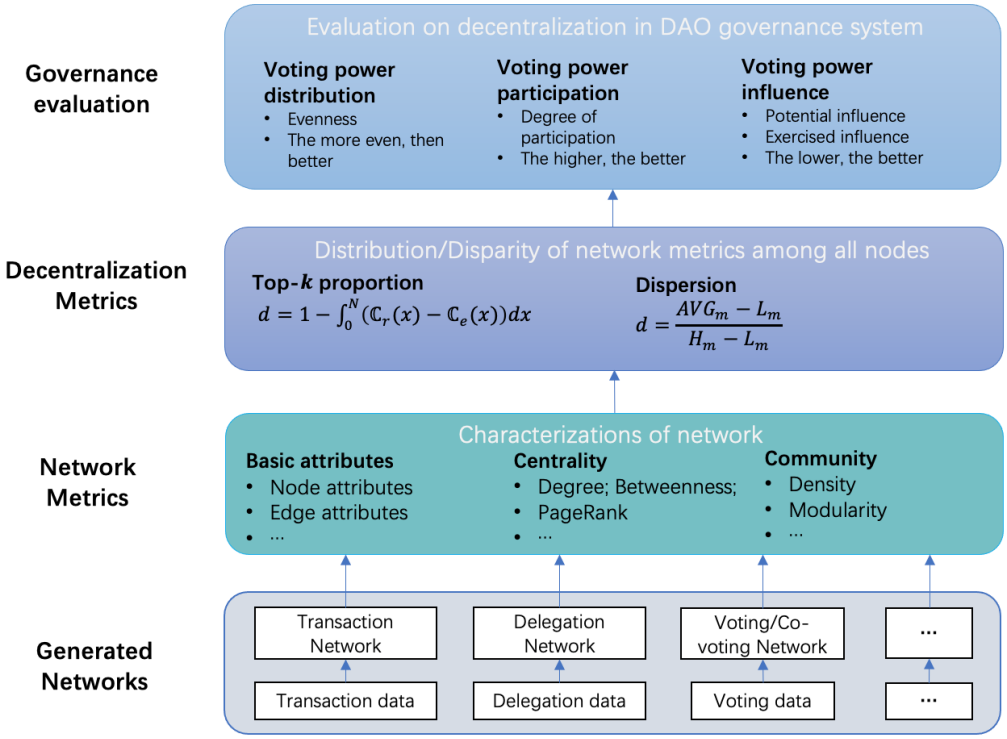


Figure 4.1: General evaluative framework for DAO governance.

system, in this chapter, we propose a four-layer evaluative framework based on the degree of quantitative decentralization.

Firstly, we will build analysis networks from the underlying data. Our aim is to try to apply network analysis methods to explore the distribution of voting power in the governance system. Specifically, we will build a transaction network for transaction history data. Due to the historical timeliness of token transactions, we mainly analyze token transactions in the network during the poll voting process. So for each poll, we set up a separate transaction network for the lifetime of the poll. Assume that the life cycle of the poll is $[pt_s, pt_e]$; considering the timeliness of transactions, we will take all the token transaction records of all voters participating in this cycle $[pt_s - 7, pt_e]$ as shown in the Figure 4.2(a), where $pt_s - 7$ represents the 7 days before the poll begins voting.

Delegation is an important trust transfer behaviour in the DAO governance system and generally speaking, the delegation also has a life cycle, so for the delegated record, we can get a historical accumulated state in days. So, for each

poll, we only need to consider the cumulative state until the end of the poll lifecycle, and Figure 4.2(b) shows the delegating network of the poll 1000.

Finally, we try to identify potential voting groups from similar voting behaviour patterns of co-voting where co-voting means the number of polls on which two voters vote together. If there is a relatively concentrated cluster of similar behaviours in a governance system, then we have reason to believe that there are potential vote manipulators. Figure 4.2(c) shows the whole historical voting records of all votes with the number of co-voting polls greater than 50.

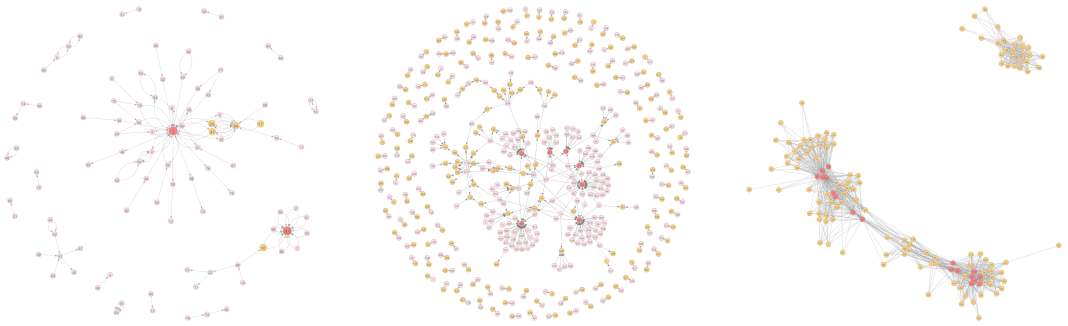
4.3 Dataset

According to the analytical framework in Figure 4.1, We collect comprehensive data on the token holders, delegates, votes and Polls for governance systems from the Ethereum blockchain and public interfaces for specific projects to generate the transaction, delegation and co-voting networks. In our work, we focus on the Ethereum-based DeFi platform, Maker protocol, developed and managed by MakerDAO, as a case study to illustrate the feasibility and effectiveness of our evaluative framework. The rationale behind this choice is simple. MakerDAO is one of the most influential-based DAOs [82]. Since 2017, when the DeFi universe expanded exponentially, the Maker protocol has emerged as the leading lending protocol, which conceptually replicates the operation of a bank in cryptocurrency markets. In the Maker protocol, Maker (MKR) is the governance token. In terms of its value, one token equals one vote in the proposed polls. Aside from this tokenized value, Maker protocol issues the DAI, which is a stablecoin soft-pegged to the US Dollar (MakerDAO, 2020). Currently, DAI is one of the most traded stablecoins, with a daily number of transactions exceeding ten thousand.

Table 4.1: Key figures of the dataset

	Polls	**Voters**	**Delegates**	**Total Votes**
MakerDAO	1,076	2,703	340	34,440

Table 4.1 provides an overview of the state of MakerDAO governance systems as of 06 June 2024. The analysis encompasses a more than four-year period between Ethereum blocks 8,292,000 (15 August 2019) and 20,012,044 (06 June 2024). In our evaluative framework, we mainly consider the influence of voter behaviour on



(a) Transaction network (b) Delegation network (c) Co-voting network

Figure 4.2: A visualization of the transaction, delegation networks for poll 1000 and co-voting network with the co-voting number > 50

the voting result, so we generate transaction (e.g., Figure 4.2(a)) and delegation (e.g., Figure 4.2(b)) networks for each poll voting stage, respectively, according to the historical transaction and delegation records. Meanwhile, we also generate a co-voting network for all voters according to the historical voting results as shown in Figure 4.2(c).

4.4 Measurement of decentralization degree

To find the answer to the question "Who controls DAO governance?" we take the degree of decentralization of the system as a measure and use social network analysis methods to measure the distribution of each network metric across all voting nodes. In the most ideal state, all voting nodes have the same network measurement value, so the voting weights in the system are evenly distributed and completely decentralized. In practice, due to the differences of individual factors, the distribution of voting weights often deviates from the optimal state. Therefore, in our work, we use this degree of deviation to define the degree of decentralization.

In this section, we first introduce several network metrics which we can use to measure the decentralization degree (i.e., dd) in our framework, and next, we introduce two decentralization degree metrics used in work [16] and what they mean and work in our evaluative framework.

4.4.1 Network metrics

When token transactions and delegations, and voting behaviours are represented in the format of a network, it is natural to ask how network properties can be used to offer the most insight in measuring the decentralization from transactions, delegations and votes. To that end, we adopt a two-step approach as follows, referring to the work [16].

Firstly, we identify network measures that strongly indicate the node's importance in controlling votes. In network analysis, the following three network metrics - Degree Centrality (In-degree and Out-degree) and PageRank - come the closest to what we aim to characterize as shown in Figure 4.1.

Degree Centrality. In a network, in-degree and out-degree carry different meanings and significance, just like in the case of hyperlinks between webpages. In-degree means asset transferring into the target node and, therefore, naturally indicates greater trust and endorsement for the target node than out-degree. However, as we observed that the in-degree and out-degree have very similar distributions in our study, we used degree centrality to summarize both. In general, the greater the Degree Centrality, the more important the node due to involvement in more votes.

PageRank. The reason why PageRank, the classic network measure for ranking the authority of web pages, is relevant in our setting is that – Transactions or delegations can be viewed as trust endorsement. The higher the PageRank score of a node, the more trust endorsement for the node as manifested by token transactions transferring the voting power to this node [16, 82]. In general, a node with a high PageRank score in a transaction or delegation network is considered to serve as an intermediary of three kinds: (I) financial intermediary, serving as middleman for financial transactions to create efficient markets and to lower business costs; (II) credibility intermediary, offering trust to other nodes to enable transactions; and (III) information intermediary, providing information for asset pricing, bidding, matching and others based on data from large transaction volume.

Secondly, it should be noted that the notion of decentralization does not necessarily depend on the importance of transaction control for any particular node as measured by any one network property; it is more related, rather, to the disparity among different nodes in their values for any chosen network property. As such, it

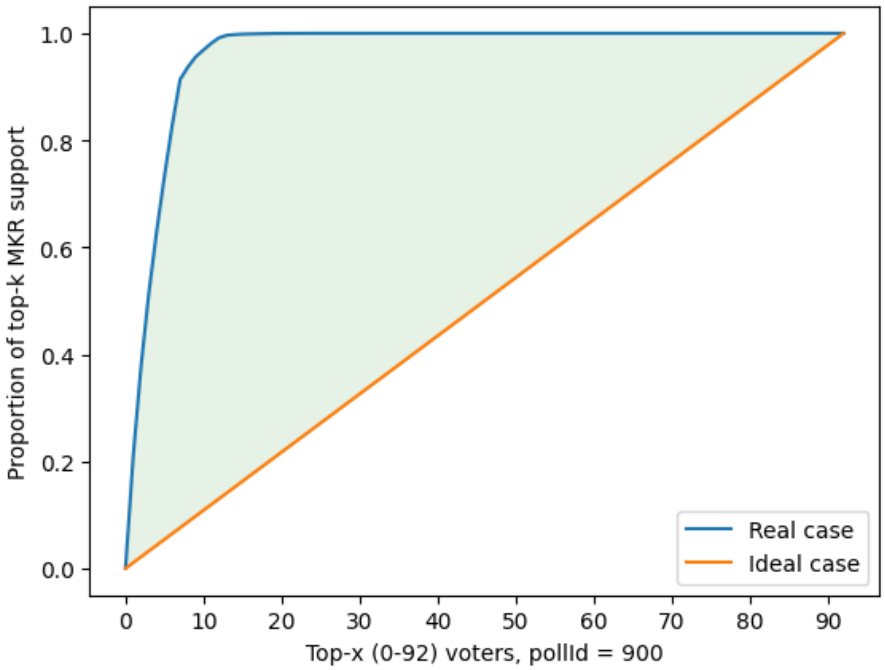


Figure 4.3: Measurement of top-k decentralization degree

is not sufficient simply to identify the network properties, more importantly, we use two notions of **top-k proportion** and **dispersion** and use them on each metric to evaluate the decentralization of the networks.

4.4.2 Top-k proportion decentralization degree

For the distribution of the basic attributes, including node and edge attributes, we use the static decentralization degree referring to work [16] with the idea that maximum decentralization in terms of static token distribution is when everyone owns the same amount of token in the system. This situation is depicted in Figure 4.3 by the yellow straight line, which represents the cumulative proportion of tokens owned by top-N addresses as N increases for the poll 900. The blue curve representing the real situation would be a convex curve deviating upward from this green straight line by definition. The more bulging the curve, the more uneven the distribution of tokens among the top-N addresses, and accordingly, the lower the decentralization. The same is true for other network metrics. We, therefore, define the Top-k proportion decentralization degree as follows.

We measure the area between the curve of the cumulative proportion of the top-N

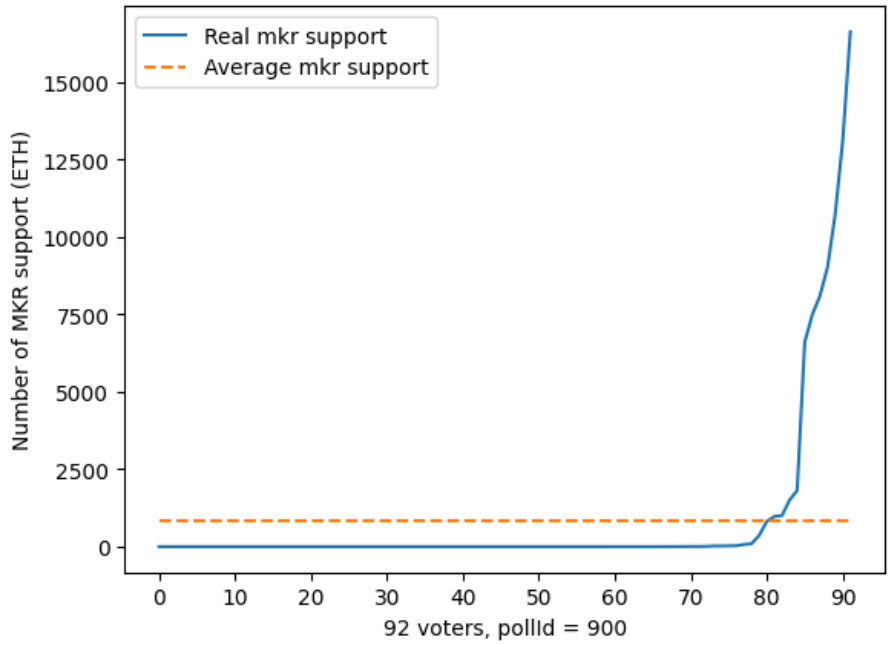


Figure 4.4: Measurement of dispersion decentralization degree

addresses and the straight line of equal distribution, and the static decentralization degree, denoted by D_{static}, is defined by 1 minus this area. Specifically, we have

$$D_{top-k} = 1 - \int_1^N (\mathbb{C}_r(x) - \mathbb{C}_e(x))dx \tag{4.1}$$

where $\mathbb{C}_r(x)$ and $\mathbb{C}_e(x)$ represent respectively the cumulative proportion of tokens owned by top-N addresses in the real case and in the ideal case where every address owns the same amount, the blue curve in Figure 4.3 represents the data of poll 900. The sharp rise of the curve before $x = 15$ and the flatness thereafter, as well as the corresponding bulging of the curve toward $x < 15$ indicate the high centralization of MKR in the top-15 addresses by that time, echoing the results in ranking analysis in the section 4.5.

4.4.3 Dispersion decentralization degree

For voting power in DAO governance, we are interested not only in its distribution among all voters for any poll voting period but also in its flow through transactions and delegations of tokens from a static timestamp point of view. In this section, we will continue to introduce its level of decentralization from a dynamic perspective

by studying transfer and delegate behaviours according to the transaction and delegation network, i.e., whether and to what extent the poll votes are monopolized by a voter or a certain subgroup of voters. As such, we use the notion of *dispersion* and use it on each metric to evaluate the decentralization of the networks. We use dispersion [16] D_m for a chosen metric m as follows:

$$D_{dispersion} = \frac{AVG_m - L_m}{H_m - L_m} \tag{4.2}$$

Where H_m,L_m and AVG_m stand for the greatest, the least and the average value for metric m. The higher the dispersion, the more pronounced the control of transactions by a subgroup within the whole graph, and the less the decentralization of token transactions or delegation network as shown in Figure 4.4 for poll 900.

To better understand dispersion in our setting, we explore a few cases for which the dispersion takes some specific values in section 4.5. Note that, by definition, for a chosen metric, if the largest value is much greater than the second largest value, the dispersion would take a value close to 1. If the largest and the second largest values are close, and these two values are much greater than the third largest value, the dispersion would take a value close to 0.5. It is not hard to extend a similar analysis to other cases. It is, therefore, possible to use dispersion to estimate the number of addresses that form the top-ranking group such that their metric values are close to one another and, as a whole, much greater than the rest.

4.5 Governance evaluation

In the following, based on our proposed evaluative framework, we study who holds the voting power or even controls the decision results in the governance systems (e.g., MakerDAO [27]). In our work, We focus on the Ethereum-based DeFi platform, Maker protocol, developed and managed by MakerDAO, as a case study. The rationale behind this choice is simple. MakerDAO is one of the most influential-based DAOs [27]. Specifically, We examine this in three steps [34] on the basis of the fourth layer of our proposed framework: 1) How are the voting rights distributed? 2) How frequently are voting rights executed? 3) How much do the executed votes influence the final governance decision?

4.5.1 Distribution of voting power

Firstly, we examine how voting rights are distributed in the governance systems, which can reveal much information about the centralization trend in the decision-making process. work [64] has shown that token distribution is usually centralized in DeFi, and we suspect such centralization also exists in the DAO system, which includes two types of voters, i.e., single voters and token delegates.

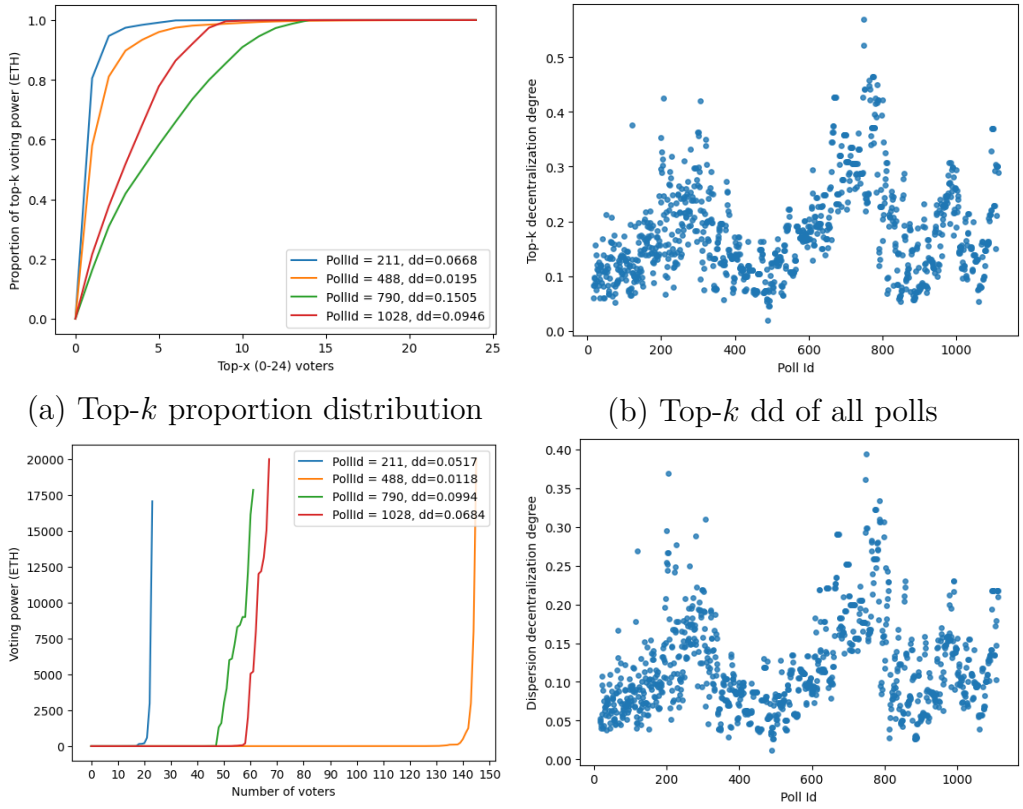
For each single voter, the voting power TS_i is represented by the token holding through the token transaction. And we calculate the token balance controlled by a delegate TD_i during the delegation lifecycle $[dt_s, dt_e]$, which equals the sum of tokens owned by delegate TD_i and represented by delegate TD_i. Assuming that there are n token holders have delegated the tokens to TD_i until the date d, the token balance controlled by this delegate TD until the date d can be calculated as follows:

$$TD_{i,d} = \sum_{j=1}^{n} TH_{j,d} \tag{4.3}$$

where $TH_{j,d}$ is the token balance controlled by the token holder j on the date d. So, the whole voting power of the system is the collection of two types of users $V = \{TS, TD\}$, which can be obtained from the transaction and delegation network, respectively. A higher TS or TD means more voting power is controlled by the delegate, implying governance centralized caused by these influential voters.

Based on the generated transaction and delegation networks, we can get the sets TS and TD by calculating the degree value of two networks, respectively, during each poll lifecycle. Therefore, firstly, for each poll, we can calculate a top-k voting power proportion distribution graph as shown in Figure 4.5(a) as the example of polls 211, 488, 790 and 1028. We compute the proportion of MKR controlled by large MKR holders with the top-k decentralization degree with the calculation formula 4.1. From Figure 4.5(a), we can easily see that within a poll, a very small number of voters hold most of the voting power. For instance, for poll 488, only six voters control more than 98% of the voting power of the whole system. So, based on the top-k proportion distribution for each poll, we can calculate the top-k proportion decentralization degree using the formula 4.1, and the results for all polls have been shown in the Figure 4.5(b). In the figure, each blue point represents the value of the decentralization degree for a specific poll. The column "Top-k

dd/Cumulation" of Table 4.2 shows the proportion of the number of polls contained in each decentralization degree value interval. We can see that the decentralization degrees of 94.79% polls are less than 0.4 << 1. As discussed in the subsection 4.4.2, for the decentralization degree, the higher, the better. The closer it is to 1, the more evenly the voting weights of the system are distributed.



(a) Top-k proportion distribution

(b) Top-k dd of all polls

(c) Dispersion distribution

(d) Dispersion dd of all polls

Figure 4.5: Voting power distribution analysis

Secondly, for each poll, we can get the dispersion distribution graph of voting power as shown in Figure 4.5(c) as the same example of polls 211, 488, 790 and 1028. From Figure 4.5(c), we can easily see that within a poll, only one voter holds the absolute voting weight of the vote. For instance, for poll 488, only one voter controls more than 20,000 ETH voting power, and the second has only 8,000 ETH voting power. So, based on the dispersion distribution for each poll, we can calculate the dispersion decentralization degree using the formula 4.2, and the results for all polls have been shown in the Figure 4.5(d). In the figure, each blue point represents the value of the decentralization degree for a specific poll. From the two figures

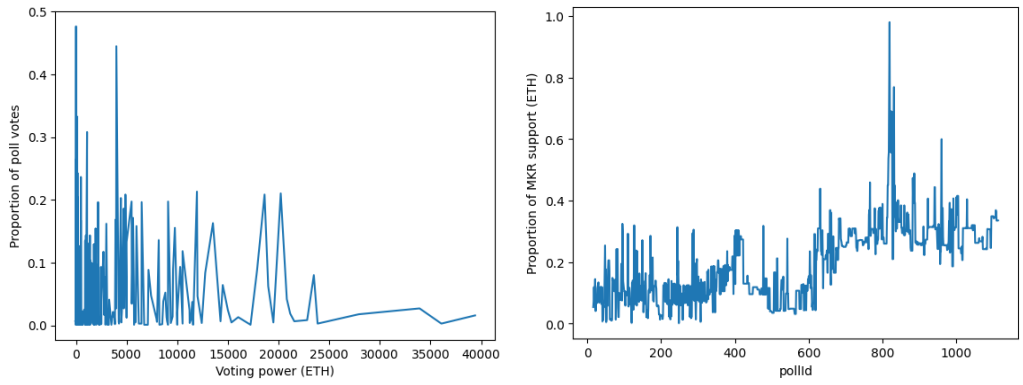
(b) and (d), it is not difficult to find that they have the same distribution, which also verifies the feasibility of our method from the side. The column "Dispersion dd/Cumulation" of Table 4.2 shows the proportion of the number of polls contained in each decentralization degree value interval. We can see that the decentralization degrees of 98.88% polls are less than $0.3 << 1$. As discussed in the subsection 4.4.3, for the decentralization degree, the higher, the better. The closer it is to 1, the more evenly the voting weights of the system are distributed.

Table 4.2: Voting power distribution statistical analysis

Value interval	**Top-k dd/Cumulation**	**Dispersion dd/Cumulation**
[0.0, 0.1)	0.1702/0.1702	0.4800/0.4800
[0.1, 0.2)	0.4902/0.6605	0.4149/0.8949
[0.2, 0.3)	0.2335/0.8939	0.0940/0.9888
[0.3, 0.4)	0.0809/0.9749	0.0112/1.000
[0.4, 0.5)	0.0232/0.9981	
[0.5, 0.6)	0.0019/1.000	

4.5.2 Participation of voting power

A second crucial aspect of analyzing voting power is considering how often MKR holders participate in voting, i.e., how often they use their voting power—the higher the frequency, the better.



(a) Voting rate (b) Supporting rate

Figure 4.6: Voting participation analysis based on voting network

We can obtain the participation frequency from the voting network, which has been shown in Figure 4.1(d). In this network, each blue node represents the voting node with the attribute "voting power", each red node represents the poll node with

Table 4.3: Voting participation analysis

Nodes	**Top-k dd**	**Dispersion dd**
Vote nodes (voting rate)	0.2302	0.1334
Poll nodes (supporting rate)	0.3402	0.2567

the attribute "total MKR support", and the directed edge represents the voting behaviour with the weight of MKR support. In our work, we use the "voting rate" and "supporting rate" to simultaneously represent the voting power participation.

We can obtain the voting rate by calculating the out-degree of each voting node in the voting network because each edge connected to it represents a vote, and the result of the MakerDAO has been shown in Figure 4.6(a). Specifically, for some polls, no more than ten voters will participate in the decision-making process. The finding implies that not all polls have large voting participation. Compared to the rapid growth of Maker users, voters are a small group. Our analysis extracts a total of 2703 unique voters in our dataset. For each voter, the number of polls that they participate in can be surprisingly different.

Moreover, We can obtain the supporting rate by calculating the weighted in-degree of each poll node in the voting network because each edge connected to it represents a support, and the result of the MakerDAO has been shown in Figure 4.6(b). Specifically, for most of the polls, no more than 0.4 total voting power of the system will participate in the decision-making process. The finding implies that most of the polls have small voting participation. Compared to the rapid growth of MKR supply, voting power is a small part of it.

Finally, by calculating both top-k and dispersion decentralization degree as shown in Table 4.3 and examining the total votes and the highest votes that a voter has in a single poll, it is implied that the voting power is not equally distributed across voters. This could be an early sign of voting centralization. However, to make this claim clearer, we need to delve deeper into the composition of the voters and their characteristics. To that end, we identify the voters whose identity is publicly available, the top ten voters who participate in most polls, the top ten voters who have the largest total votes and the top ten voters who have the largest single vote.

4.5.3 Influence of voting power

As the final step, we examine how the votes that are cast actually influence the outcome of the governance decisions. It is one thing for delegates to have the possibility to change votes. A completely different matter is whether they actually choose to do so. To measure this, we use the following two metrics inspired in the work [34], which are designed to quantify the potential and exercised voting power of single voters and delegates.

Potential voting power. We measure the potential voting power of a voter by counting how often the voter could have changed the outcome of a proposal vote by changing their own vote. Formally, the potential power $P_{pot} \in \{0, 1\}$ of a voter who participated in a poll is defined as

$$P_u^{pot} = \begin{cases} 1, if\ V^y - v_u < v^n + v_u \\ 0, else \end{cases} \tag{4.4}$$

where v^y and v^n are the numbers of Yes- and No-votes on the poll, respectively. Furthermore, v_u denotes the voter's number of votes. The overall potential voting power of a poll is then defined as the sum of the potential voting power of all voters.

Exercised voting power. The exercised voting power measures how often did the vote of a voter actually change the outcome of a proposal vote, i.e. how often the result would have been different without the voter's votes. It is defined as

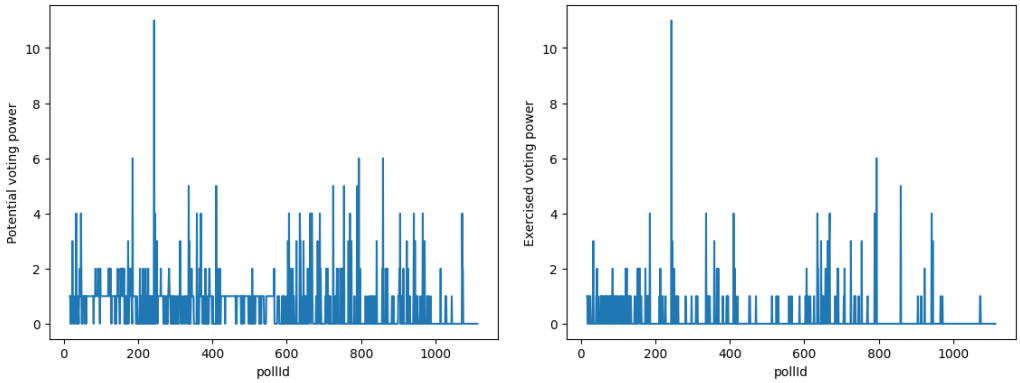
$$P_u^{ex} = \begin{cases} 1, if\ V^y - v_u < v^n \\ 0, else \end{cases} \tag{4.5}$$

Again, where v^y and v^n are the numbers of Yes- and No-votes on the poll, respectively. Furthermore, v_u denotes the voter's number of votes. The overall exercised voting power of a poll is then defined as the sum of the exercised voting power of all voters.

Table 4.4: Voting power influence analysis

Influence	**Top-k dd**	**Dispersion dd**
Potential voting power	0.4128	0.3791
Exercised voting power	0.3765	0.2763

Figure 4.7 plots the potential and exercised voting power of voters in the MakerDAO governance system. The chart reveals that a substantial portion of the



(a) Potential voting power (b) Exercised voting power

Figure 4.7: Voting influence analysis based on voting network

potential and voting power is shared among several polls; actually, close to 46.88% of the polls have the potential or exercised voting power, which can change the voting results. This observation is once again evidence of how much power lies within the hands of a select few individuals. However, we also find that these individuals do not exercise their power often; their actual voting activity falls significantly short of their potential influence. Large voters only exercise their voting power rarely.

4.6 Conclusion

In this chapter, we propose a general evaluative framework for DAO governance analysis. Our analysis of MakerDAO with the framework reveals a significant concentration of voting power in the DAO governance systems, with a majority controlled by a limited number of addresses. Additionally, the often low participation in voting leads to even more influence of those with significant voting power who actively participate. We state these observations without judgment. Nonetheless, they illustrate the challenges of building a genuinely decentralized governance system.

Chapter 5

Conclusion and Future works

In this thesis, we first propose a general consensus evaluative framework from three perspectives: resilience, performance, and governance. By analyzing the consensus algorithms, we can provide a useful reference for the designers so that they can choose the most appropriate existing algorithm or design a new efficient consensus algorithm for the new distributed ledger system.

Secondly, We propose a novel partially asynchronous consensus protocol called Proof-of-Data (i.e., PoD) tailored for decentralized federated learning settings with Byzantine fault tolerance. PoD combines Proof-of-Work (i.e., PoW) style asynchronous consensus with epoch-based consensus locking enabled by a BFT-style component, taking the best of both by endowing, on the one hand, the missing consensus finality to the practically robust yet theoretically flawed PoW consensus and lending, on the other hand, the scalability necessary for societal-scale application setting to the otherwise sound PBFT consensus. Through our proposed evaluation framework and a large number of experiments, we verify the feasibility and safety of PoD.

Finally, we propose a general evaluative framework for DAO governance of tokenization based on decentralization degree distribution. Through the core decentralized analysis, we aimed to illustrate the challenges of building a genuinely decentralized governance system and prepare for the design of a relatively perfect tokenization governance system.

For future works, firstly, We want to comprehensively collate existing consensus algorithms in four stages of development and evaluate them categorically with the proposed framework. For PoD we want to do comparative experiments with more

datasets and baseline algorithms to evaluate PoD's performance fully. Finally, we want to comprehensively research existing DAO governance systems and evaluate them with the proposed evaluative framework.

Bibliography

[1] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth, "Bar fault tolerance for cooperative services", in *Proceedings of the twentieth ACM symposium on Operating systems principles*, 2005, pp. 45–58.

[2] A. Baliga, "Understanding blockchain consensus models", in *Persistent*, 2017.

[3] S. Bano, M. Al-Bassam, and G. Danezis, "The road to scalable blockchain designs", *USENIX; login: magazine*, 2017.

[4] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Sok: Consensus in the age of blockchains", *arXiv preprint arXiv:1711.03936*, 2017.

[5] X. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, "Flchain: A blockchain for auditable federated learning with trust and incentive", in *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, IEEE, 2019, pp. 151–159.

[6] T. Barbereau, R. Smethurst, O. Papageorgiou, A. Rieger, and G. Fridgen, "Defi, not so decentralized: The measured distribution of voting rights",, 2022.

[7] T. Barbereau, R. Smethurst, O. Papageorgiou, J. Sedlmeir, and G. Fridgen, "Decentralised finance's timocratic governance: The distribution and exercise of tokenised voting rights", *Technology in Society*, vol. 73, p. 102 251, 2023.

[8] S. R. Bond, A. Hoeffler, and J. R. Temple, "Gmm estimation of empirical growth models", *Available at SSRN 290522*, 2001.

[9] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains", Ph.D. dissertation, 2016.

[10] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on bft consensus", *arXiv preprint arXiv:1807.04938*, 2018.

[11] V. Buterin and V. Griffith, "Casper the friendly finality gadget", *arXiv preprint arXiv:1710.09437*, 2017.

[12] M. Castro and B. Liskov, "Practical byzantine fault tolerance", in *OSDI*, vol. 99, 1999, pp. 173–186.

[13] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery", *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.

[14] M. Chahoud, S. Otoum, and A. Mourad, "On the feasibility of federated learning towards on-demand client deployment at the edge", *Information Processing & Management*, vol. 60, no. 1, p. 103 150, 2023.

[15] Z. Chen, D. Li, J. Zhu, and S. Zhang, "Dacfl: Dynamic average consensus based federated learning in decentralized topology", *arXiv preprint arXiv:2111.05505*, 2021.

[16] L. Cheng, F. Zhu, H. Liu, and C. Miao, "On decentralization of bitcoin: An asset perspective", *arXiv preprint arXiv:2105.07646*, 2021.

[17] U. W. Chohan, "The decentralized autonomous organization and governance issues", *Available at SSRN 3082055*, 2017.

[18] D. Chowdhury, S. Banerjee, M. Sannigrahi, A. Chakraborty, A. Das, A. Dey, and A. D. Dwivedi, "Federated learning based covid-19 detection", *Expert Systems*, vol. 40, no. 5, e13173, 2023.

[19] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, *et al.*, "On scaling decentralized blockchains", in *International Conference on Financial Cryptography and Data Security*, Springer, 2016, pp. 106–125.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[21] D. E. Dillenberger and G. Su, "Parallel execution of blockchain transactions", US Patent 10,255,108, Apr. 2019.

[22] C. T. Dinh, N. H. Tran, M. N. Nguyen, C. S. Hong, W. Bao, A. Y. Zomaya, and V. Gramoli, "Federated learning over wireless networks: Convergence analysis and resource allocation", *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 398–409, 2020.

[23] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "Blockbench: A framework for analyzing private blockchains", in *Proceedings of the 2017 ACM International Conference on Management of Data*, ACM, 2017, pp. 1085–1100.

[24] D. Dolev, C. Dwork, and L. Stockmeyer, "On the minimal synchronism needed for distributed consensus", *Journal of the ACM (JACM)*, vol. 34, no. 1, pp. 77–97, 1987.

[25] Y. Duan and J. F. Canny, "Zero-knowledge test of vector equivalence granulation of user data with privacy." In *GrC*, Citeseer, 2006, pp. 720–725.

[26] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony", *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 1988.

[27] E. W. Ellinger, T. Mini, R. W. Gregory, and A. Dietz, "Decentralized autonomous organization (dao): The case of makerdao", *Journal of Information Technology Teaching Cases*, p. 20 438 869 231 181 151, 2023.

[28] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol", in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 45–59.

[29] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable", *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[30] E. F. Fama and M. C. Jensen, "Separation of ownership and control", *The journal of law and Economics*, vol. 26, no. 2, pp. 301–325, 1983.

[31] M. S. Farooq, R. Tehseen, J. N. Qureshi, U. Omer, R. Yaqoob, H. A. Tanweer, and Z. Atal, "Ffm: Flood forecasting model using federated learning", *IEEE Access*, vol. 11, pp. 24 472–24 483, 2023.

[32] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process." MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, Tech. Rep., 1982.

[33] E. Foundation, "Ethereum wire protocol", 2017, `https://github.com/ethereum/devp2p/blob/master/caps/eth.md`, accessed: 2019-10.

[34] R. Fritsch, M. Müller, and R. Wattenhofer, "Analyzing voting power in decentralized governance: Who controls daos?" *Blockchain: Research and Applications*, p. 100 208, 2024.

[35] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, and E. G. Sirer, "Decentralization in bitcoin and ethereum networks", *arXiv preprint arXiv:1801.03998*, 2018.

[36] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains", in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, ACM, 2016, pp. 3–16.

[37] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin", in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2015, pp. 692–705.

[38] J. B. Haviland, "Gossip, reputation, and knowledge in zinacantan", University of Chicago Press Chicago, Tech. Rep., 1977.

[39] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning", in *IFIP International Conference on Distributed Applications and Interoperable Systems*, Springer, 2019, pp. 74–90.

[40] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network", in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, 2015, pp. 129–144.

[41] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach", *arXiv preprint arXiv:1908.07782*, 2019.

[42] J. R. Jensen, V. von Wachter, and O. Ross, "How decentralized is the governance of blockchain-based finance: Empirical evidence from four governance token distributions", *arXiv preprint arXiv:2102.10096*, 2021.

[43] S. Joshi, "Feasibility of proof of authority as a consensus protocol model", *arXiv preprint arXiv:2109.02480*, 2021.

[44] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin", in *Proceedings of the 2012 ACM conference on Computer and communications security*, ACM, 2012, pp. 906–917.

[45] G. O. Karame, E. Androulaki, and S. Capkun, "Double-spending fast payments in bitcoin", in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 906–917.

[46] A. Kiayias and G. Panagiotakos, "On trees, chains and fast transactions in the blockchain", in *International Conference on Cryptology and Information Security in Latin America*, Springer, 2017, pp. 327–351.

[47] Y. J. Kim and C. S. Hong, "Blockchain-based node-aware dynamic weighting methods for improving federated learning performance", in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2019, pp. 1–4.

[48] S. Kitzler, S. Balietti, P. Saggese, B. Haslhofer, and M. Strohmaier, "The governance of distributed autonomous organizations: A study of contributors' influence, networks, and shifts in voting power", *arXiv preprint arXiv:2309.14232*, 2023.

[49] J. Konečnỳ, H. B. McMahan, X. Y. Felix, A. T. Suresh, D. Bacon, and P. Richtárik, "Federated learning: Strategies for improving communication efficiency",, 2018.

[50] J. Kwon, "Tendermint: Consensus without mining", *Draft v. 0.6, fall*, vol. 1, p. 11, 2014.

[51] L. Lamport *et al.*, "The part-time parliament", *ACM Transactions on Computer systems*, vol. 16, no. 2, pp. 133–169, 1998.

[52] L. Lamport *et al.*, "Paxos made simple", *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[53] L. LAMPORT, R. SHOSTAK, and M. PEASE, "The byzantine generals problem", *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.

[54] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem", *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[55] C.-Y. Lee, "Decentralized allocation of emission permits by nash data envelopment analysis in the coal-fired power market", *Journal of environmental management*, vol. 241, pp. 353–362, 2019.

[56] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation", in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, 2020, pp. 515–528.

[57] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus", *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[58] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A blockchain-based decentralized federated learning framework with committee consensus", *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2020.

[59] H. Lin, K. Chen, D. Jiang, L. Shou, and G. Chen, "Refiner: A reliable and efficient incentive-driven federated learning system powered by blockchain", *The VLDB Journal*, pp. 1–25, 2024.

[60] U. Majeed and C. S. Hong, "Flchain: Federated learning via mec-enabled blockchain network", in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 2019, pp. 1–4.

[61] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data", in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.

[62] Q. Mechanic, "Proof of stake", Website, `https://en.bitcoin.it/wiki/ProofofStake`, 2011.

[63] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols", in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 31–42.

[64] M. Nadler and F. Schär, "Decentralized finance, centralized ownership? an iterative mapping process to measure protocol token distribution", *arXiv preprint arXiv:2012.09306*, 2020.

[65] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system",, 2008.

[66] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system",, 2008.

[67] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges", *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 806–12 825, 2021.

[68] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms", in *Proceedings of the second workshop on distributed infrastructures for deep learning*, 2018, pp. 1–8.

[69] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm", in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 305–319.

[70] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults", *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 228–234, 1980.

[71] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults", *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 228–234, 1980.

[72] B. Podgorelec, M. Turkanović, and S. Karakatič, "A machine learning-based method for automated blockchain transaction signing including personalized anomaly detection", *Sensors*, vol. 20, no. 1, p. 147, 2020.

[73] S. Popov, "The tangle", *cit. on*, p. 131, 2016.

[74] A. Qammar, A. Karim, H. Ning, and J. Ding, "Securing federated learning with blockchain: A systematic literature review", *Artificial Intelligence Review*, vol. 56, no. 5, pp. 3951–3985, 2023.

[75] X. Qu, S. Wang, Q. Hu, and X. Cheng, "Proof of federated learning: A novel energy-recycling consensus algorithm", *IEEE Transactions on Parallel & Distributed Systems*, vol. 32, no. 08, pp. 2074–2085, 2021.

[76] F. Saleh, "Blockchain without waste: Proof-of-stake", *Available at SSRN 3183935*, 2019.

[77] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin", in *International Conference on Financial Cryptography and Data Security*, Springer, 2016, pp. 515–532.

[78] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin", in *Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, February 22–26, 2016, Revised Selected Papers 20*, Springer, 2017, pp. 515–532.

[79] E. Schleiger, C. Mason, C. Naughtin, A. Reeson, and C. Paris, "Collaborative intelligence: A scoping review of current applications", *Applied Artificial Intelligence*, vol. 38, no. 1, p. 2 327 890, 2024.

[80] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol." *IACR Cryptology ePrint Archive*, vol. 2016, p. 1159, 2016.

[81] K. Stroponiati, I. Abugov, Y. Varelas, K. Stroponiatis, M. Jurgeleviciene, and Y. Savanth, "Decentralized governance in defi: Examples and pitfalls", *Tech. rep*, 2020.

[82] X. Sun, C. Stasinakis, and G. Sermpinis, "Decentralization illusion in decentralized finance: Evidence from tokenized voting in makerdao polls", *Journal of Financial Stability*, p. 101 286, 2024.

[83] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems", in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*, Springer, 2020, pp. 480–501.

[84] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies", *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[85] G. Tsoukalas and B. H. Falk, "Token-weighted crowdsourcing", *Management Science*, vol. 66, no. 9, pp. 3843–3859, 2020.

[86] M. Vukolic, "Eventually returning to strong consistency." *IEEE Data Eng. Bull.*, vol. 39, no. 1, pp. 39–44, 2016.

[87] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones", in *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, 2019, pp. 95–112.

[88] W. Wang, D. T. Hoang, Z. Xiong, D. Niyato, P. Wang, P. Hu, and Y. Wen, "A survey on consensus mechanisms and mining management in blockchain networks", *arXiv preprint arXiv:1805.02707*, pp. 1–33, 2018.

[89] Y. Wang, H. Peng, Z. Su, T. H. Luan, A. Benslimane, and Y. Wu, "A platform-free proof of federated learning consensus mechanism for sustainable blockchains", *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3305–3324, 2022.

[90] X. Wei and C. Shen, "Federated learning over noisy channels: Convergence analysis and design examples", *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 2, pp. 1253–1268, 2022.

[91] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger", *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[92] N. Youdao, "P4p: Practical large-scale privacy-preserving distributed computation robust against malicious users", *Proc USENEX*, 2010.

[93] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients", in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2545–2555.

[94] Z. Zhao, J. Xia, L. Fan, X. Lei, G. K. Karagiannidis, and A. Nallanathan, "System optimization of federated learning networks with a constrained latency", *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 1095–1100, 2021.