Blockchain-based Design for Performant Peer-to-Peer Energy Trading Systems

by

Caixiang Fan

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

With the emergence of renewable energy resources, such as solar panels, wind turbines and plug-in electric vehicles, there is an increasing need for peer-to-peer energy trading (P2P-ET) among small-scale energy producers, known as prosumers. Many recent studies have suggested blockchain for P2P-ET to provide many advantages over centralized solutions. These include but are not limited to, improved security, privacy, fast payment settlement and better fault tolerance. However, to leverage blockchain at scale, its well-known limitations, i.e., scalability and performance, should be adequately analyzed and addressed. Even though many promising consensus mechanisms have been proposed to tackle the blockchain performance issue, their performance has not been sufficiently evaluated and mathematically analyzed in production environments. Thus, in this proposed Ph.D. research, blockchain-based solutions for P2P-ET are investigated, with their performance being carefully evaluated, to show their feasibility and efficiency in energy trading applications.

First, we conducted a systematic survey on blockchain performance evaluation by categorizing all reviewed solutions into two main categories: empirical analysis and analytical modelling. The current empirical analysis methodologies include benchmarking, monitoring, experimental analysis and simulation. The analytical techniques include the Markov chains, queueing theory, and stochastic Petri nets. Through contrasting, comparing and grouping different methods, we extracted important criteria used to select the most suitable evaluation technique. We also identified a list of performance bottlenecks in various blockchain systems.

Then, we rigorously studied the performance of two promising and performant blockchain solutions: the DAG ledger IOTA and Hyperledger Besu, to examine their applicability for P2P-ET. In the IOTA performance study, we first extended a DAG simulator to support realistic IOTA simulations and investigated the impact of different design parameters on IOTA's performance. Then, we proposed a layered model to investigate the optimal waiting time to resend a previously submitted but not yet confirmed transaction. Our findings reveal the impact of the transaction arrival rate, tip selection algorithms (TSAs), weighted TSA randomness, and network delay on the throughput. Using the proposed layered model, we shed some light on the distribution of the confirmed transactions on different layers. The distribution is leveraged to calculate the optimal

time for resending an unconfirmed transaction to the distributed ledger. To study the performance of Hyperledger Besu, we designed and automated a benchmarking framework with an Nginx load balancer and judiciously selected a set of test parameters, including transaction send rate, network size, node flavour, load balancing, consensus, and block time. Then, we set up private blockchain networks to thoroughly test the impact of these parameters on the metrics such as transaction throughput, latency, scalability and resource utilization using the Hyperledger Caliper benchmark tool. Further performance data and log analyses reveal some interesting findings which shed some light on further performance improvement of Hyperledger Besu and building performant enterprise applications.

At last, we proposed and developed a unified Blockchain-based framework for P2P-ET called BPET, which combines blockchain with microservice architecture to achieve better reliability. Our proposed framework adopts a modular smart contract design, which enables P2P-ET at both the distribution level and end-user level by implementing different market rules. In particular, four smart contracts are developed: Registry for recording the registration information, EnergyToken for providing a stablecoin payment method, Market for defining all market rules, and Payment for processing the payment settlement step. We also developed a decentralized application (DApp) prototype on top of a private Hyperledger Besu blockchain network deployed on the cloud. A case study of P2P-ET in the Alberta electricity wholesale market was conducted with real data set from the Alberta Electric System Operator (AESO) to demonstrate the feasibility and efficiency of the proposed BPET system. The system evaluation results indicate that the implemented system can effectively and efficiently process energy trading transactions in the Alberta electricity wholesale market.

Preface

The research of this thesis has been conducted in the Performant and Available Computing Systems (PACS) Lab, led by Dr. Hamzeh Khazaei, and the ENergy digiTizAtIon Lab (ENTAIL), led by Dr. Petr Musilek. Some of the research conducted for this thesis forms part of a research collaboration with previous PACS lab members.

The main part of Chapter 2 and Chapter 4 of this thesis has been published as Fan, C., Ghaemi, S., Khazaei, H., Chen, Y., & Musilek, P. (2021), Performance analysis of the IOTA DAG-based distributed ledger, ACM Transactions on Modeling and Performance Evaluation of Computing Systems, 6(3), 1-20. I was responsible for the experiments, data collection and analysis, and manuscript composition. The previous PACS member Sara Ghaemi contributed to extending the DAG simulation tool and conducting the simulations.

Chapter 3 of this thesis has been published as C. Fan, S. Ghaemi, H. Khazaei and P. Musilek, "Performance Evaluation of Blockchain Systems: A Systematic Survey," in IEEE Access, vol. 8, pp. 126927-126950, 2020, doi: 10.1109/ACCESS.2020.3006078. I was responsible for literature articles collection, categorization, and analysis as well as manuscript composition. Sara Ghaemi contributed to the gathering, selecting, and summarizing related articles.

Chapter 5 of this thesis has been published as Fan, C., Lin, C., Khazaei, H., & Musilek, P. (2022, August), Performance Analysis of Hyperledger Besu in Private Blockchain, In 2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS) (pp. 64-73), IEEE. I was responsible for concept formation, experiments, data collection and analysis, and manuscript composition. The previous PACS member Changyuan Lin contributed to automating private blockchain network deployment on the cloud, running parts of benchmarking experiments, assisting with data visualization, and drafting subsection *5.3.4 Load Balancer*.

Chapters 6 and 7 of this thesis are my original work under the supervision of Dr. Hamzeh Khazaei and Dr. Petr Musilek. No part of these two chapters has been previously published at the time of this thesis submission.

Acknowledgements

Huawei-ECE Research Initiative has partially supported this work at the University of Alberta. Digital Research Alliance of Canada (alliancecan.ca) and Cybera (cybera.ca) partially supported this research work through their cloud computing resources.

I would like to thank my supervisors Dr. Hamzeh Khazaei and Dr. Petr Musilek for their professional direction and help. The support provided by the PACS lab members is also gratefully acknowledged. In particular, I would like to thank my dear wife for her encouragement and support for my research.

Contents

List of Tables

List of Figures

1 Introduction

Distributed ledger technologies (DLTs) have gained a lot of attention from both industry and academia because of their key advantages, such as decentralization, immutability, enhanced security, traceability, and lower costs. In the DLT world, participants no longer need a trusted third party (TTP) to complete a transaction, even without trust in each other. The DLTs are essentially distributed databases for storing and sharing data across all nodes in a network. Based on different usage contexts, various types of data, including transaction records (e.g., Bitcoin [91]), contracts [25], and even personal healthcare information [86], can be stored on a distributed ledger (DL) system. Within most DLTs, such as blockchain, the data is wrapped into a block structure. Each block includes the cryptographic hash of the prior block, a timestamp and transaction data to create a chain of linked blocks. This way, it is nearly impossible to tamper with data on the blockchain because any changes on a single block require changing all the history before it. According to a recent survey [23], there are about 58 industries (e.g., law enforcement, ride-hailing, and stock trading) that DLTs could transform in the future. Of course, as one of the emerging industries, peer-to-peer energy trading (P2P-ET) sits in the first-class cabin, which benefits from this cutting-edge technology.

Traditional energy trading requires a TTP to secure and enforce the trade execution. For example, the distribution system operators (DSO) or utility companies act as a TTP to complete the transaction between the buyer and the seller entities by transferring energy to the buying party after the selling party has confirmed the payment. This raises privacy and security issues, single points of failure, and performance bottlenecks. For example, if a DSO performs maliciously or the system is compromised, market participants' private information will be leaked, and energy transaction data will be muted. Smart contracts running on blockchain provide an alternative solution to resolve these issues by eliminating any TTP. When energy is delivered from producers to consumers, the corresponding data with value transfer will flow directly from one peer to another and be stored in all decentralized ledger nodes after the consensus process is completed. Market participants, including energy producers, distributors, consumers and prosumers, can interact with the blockchain network through smart meters and trade energy with each other using smart

contracts running on the blockchain, as shown in Figure 1.1.

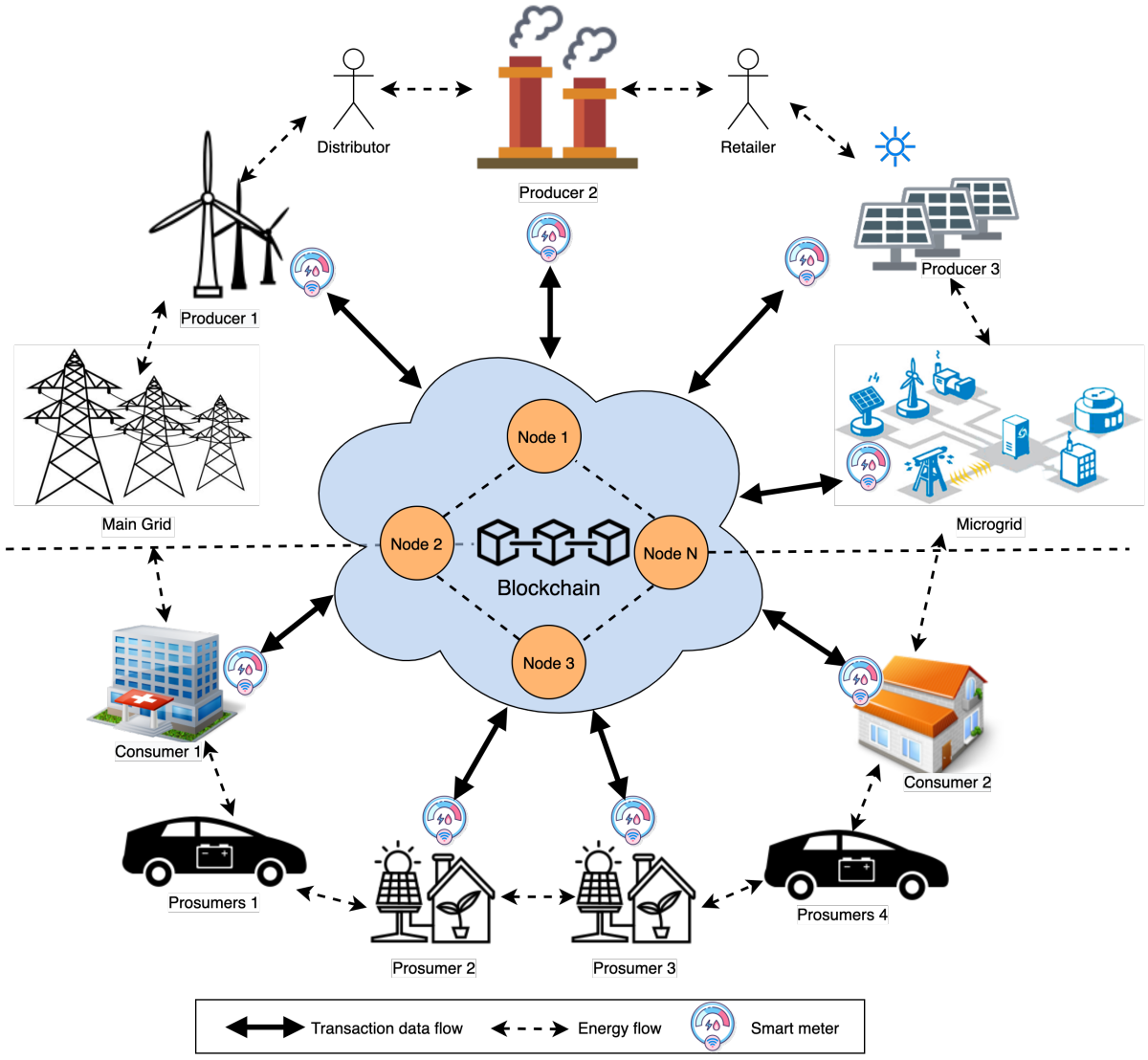


Figure 1.1: An overview of blockchain-based P2P energy trading

The upper part shows blockchain applications in distribution-level energy trading. At this level, peers are big companies such as energy producers (e.g., solar farms, power plants, wind turbine farms), energy distributors, energy retailers and microgrids. They can leverage blockchain and smart contracts to build a P2P-ET system for a wholesale market to record participants' information, define trading rules, and manage the electricity distribution process, as such to distribute energy to end users. While in the lower part, end users can leverage blockchain to build a platform for trading their excess energy with neighbours or with electric vehicles (EVs) in a P2P manner.

To this end, many research works have explored the feasibility of blockchain applications in P2P-ET and proposed different solutions to resolve particular issues. For example, in [3] [79] [76] [62] [31],

authors propose a blockchain-based P2P-ET system to tackle the security and privacy issues in energy trading. However, the performance of blockchain in the context of P2P-ET has not been well studied. Two recent studies [1] and [105] respectively propose a P2P-ET system based on the Hyperledger Besu with Istanbul Byzantine Fault Tolerance (IBFT) consensus and compare its performance with other blockchain networks and consensus algorithms through intensive experimental results. The first paper focuses on a distribution level P2P-ET that imitates the existing wholesale market, while the second focuses on renewable P2P-ET among household prosumers. Both of them agree that Besu is a performant solution for P2P-ET. But unfortunately, none of them has systematically analyzed and explained the root reasons.

1.1 Motivation

From the perspective of Quality of Service (QoS), service level agreement (SLA) and BaaS (Blockchain as a Service), the transaction confirmation rate (throughput), average waiting time (transaction delay) and system scalability are extremely important for a distributed ledger system and user experience. Some public blockchain systems with the Proof-of-Work (PoW) consensus, such as Bitcoin [92] and Litecoin [107], have very limited performance. For example, Bitcoin will, on average, take 10 minutes for a transaction to get initially confirmed; and it usually needs to wait for 6 blocks' confirmations for safety in a big transfer which means one hour of transaction settlement time. This makes them incapable of handling a high volume of energy trading transactions in a (near) real-time manner.

In practice, a blockchain-based P2P-ET system should not only serve as a payment system but also serve the whole energy trading process from end to end through smart contracts. This usually includes participant registration, bid/offer acceptance, auction processing, market rules execution, energy delivery control, payment settlement, etc. Each function in the smart contracts may trigger a transaction whenever called in an energy trading process. Moreover, an open auction mechanism in a pool market linearly increases the number of transactions. For example, in an energy market with five consumers and five producers participating, if every participant needs to submit two bids or offers to compete in the auction in a trading interval, e.g., every minute, it ends up with 20 transactions per minute in the bidding phase. Therefore, in the case of large-scale applications with hundreds of users, the total transaction requests can reach up to thousands of transactions per minute, which raises a performance challenge for the underlying energy trading system.

As an alternative solution to the traditional blockchain, the DAG-based IOTA [103] claims to

provide high performance and scalability by leveraging its innovative data structure and efficient consensus design. Another potential solution, Hyperledger Besu [45], also claims to be efficient and scalable by design with a voting-based consensus. However, both of their performance has rarely been well studied, especially in the application of P2P-ET. In this research work, we first would like to explore the application efficiency, including scalability and performance of DAG-based DL IOTA and Hyperledger Besu in the energy trading scenario. Then, we build a decentralized application (DApp) to investigate the feasibility of blockchain for P2P-ET.

1.2 Research Questions and Objectives

Even though many research works have pointed out that blockchain is a promising solution for P2P-ET, some questions remain concerning the system's efficiency, security, decentralization and scalability. In this research, we try to answer the following research questions:

- How to conduct a performance evaluation of a blockchain network?
- Which factors influence the performance of a DAG ledger (i.e., IOTA) system and how?
- Which factors influence the performance of a blockchain (i.e., Hyperledger Besu) system and how?
- How can we leverage blockchain for P2P-ET? How to choose the right blockchain solution for this application?

More research on various blockchain systems and their performance should be conducted to answer these crucial questions. From a system designer's perspective, it is vital to know about the underlying system's capacity to process generated transactions in the network. Also, from the end users' point of view, it is critical to know the transaction confirmation time in energy trading. In this Ph.D. research, we aim to:

1. Conduct a systematic survey on methodologies and techniques for evaluating the performance of various blockchain systems with different consensus algorithms;
2. Explore the performance and scalability of the IOTA ledger, which is one of the most promising decentralized solutions for P2P-ET, using both experimental and analytical approaches;
3. Explore the performance and scalability of the Hyperledger Besu, which is another promising blockchain solution for P2P-ET, using experimental approaches;

4. Design a unified blockchain-based microservice architecture with a case study to answer the question of how blockchain can be used for P2P-ET and to demonstrate the system's feasibility and efficiency.

1.3 Contributions

The contributions and expected impact of this research work are as follows:

- We conduct a systematic survey [37] on the blockchain performance evaluation; this covers all existing evaluation approaches (empirical and analytical) for evaluating the mainstream blockchain systems and compares their advantages and disadvantages.

- We study the performance of IOTA from three aspects [36]. First, we extend the DAGsim simulator [135], a DAG-based distributed ledger protocol simulator, to support the IOTA 1.0 protocol. Then we provide abundant experimental evidence on how different design parameters influence the IOTA performance and fill a gap in existing research on the optimal reattachment waiting time. At last, we propose a layered model to analyze the confirmed transactions' distribution in IOTA, providing a possible approach to investigate other DAG-based distributed ledgers.

- We rigorously design an evaluation architecture [39] to analyze the performance of the Hyperledger Besu in a private blockchain and provide abundant experimental results on the performance and scalability of its proof of authority (PoA) consensus algorithms.

- We propose and develop a unified framework for P2P-ET, which handles energy trading at distribution and end-user levels. The performance and scalability of the proposed solution are analyzed to show the feasibility and efficiency of blockchain for P2P-ET applications.

2 Background

Blockchain is a major type of distributed ledger technology (DLT). The relationship of blockchain to DLT is just like the car to the vehicle [2]. As such, the terms "blockchain" and "DLT" are used interchangeably throughout this research. Any ledger that is stored in a distributed fashion and shared among a set of nodes or participants can be referred to as a distributed ledger. For new information to be added to this ledger, all participating nodes must reach a consensus on whether the information is legitimate or not. The algorithm which determines how this decision is reached, called *consensus algorithm*, is an important part of the DLT.

2.1 Categorization of DLTs

A possible taxonomy of distributed ledger technologies is shown in Figure 2.1a. DLTs can be categorized based on their data architecture. Two main categories are blockchain and directed acyclic graph (DAG). In the blockchain, transactions are stored in containers called *blocks*, which are chained together using their hash values. This chain of information, similar to a linked list, is immutable. Examples of this category are Bitcoin [92], Ethereum [129], EOS [60], and Litecoin [107]. In DAG, on the other hand, transactions are connected to one another by a reference relationship, forming a directed graph rather than a linked list. Examples of DAG ledgers include IOTA [103], Byteball [26], and Nano [72]. In addition, there are distributed ledgers that have their unique data structure and do not fall into either of these categories, such as Radix [106] and Corda [57].

Based on the permissions of the ledger, DLTs can be classified as permissioned and permissionless, which usually makes one think of another taxonomy: private and public based on the ledger accessibility. In permissioned distributed ledgers, the identities of all the network participants are known. By contrast, everyone can participate anonymously in a permissionless open DLT network. Public and private DLTs can be distinguished by who can read the data on the ledger and verify its validity. Public ledgers are open and anyone can read the data on the ledger and host a node without the need to be approved. Private ledgers, by contrast, are only accessible to those who

are pre-approved.

Therefore, based on the permissions and accessibility of the ledger, DLTs can be divided into four groups, as shown in Figure 2.1a. Public permissionless ledgers, such as Bitcoin [91], Ethereum [129], and Litecoin [107], have no restrictions on the participating parties. In public permissioned ledgers, the identity of participants should be known but anyone can read and validate the ledger. EOS [60], Ripple [114], and Sovrin [64] are examples of this type. In private permissionless blockchains, the identities of the participants are not known but only pre-approved nodes validate the data. Examples of this category include LTO [82] and Holochain [58]. Finally, in private permissioned ledgers, such as Hyperledger [44] and Corda [57], access is restricted to pre-approved participants and the identities of the participants are known.

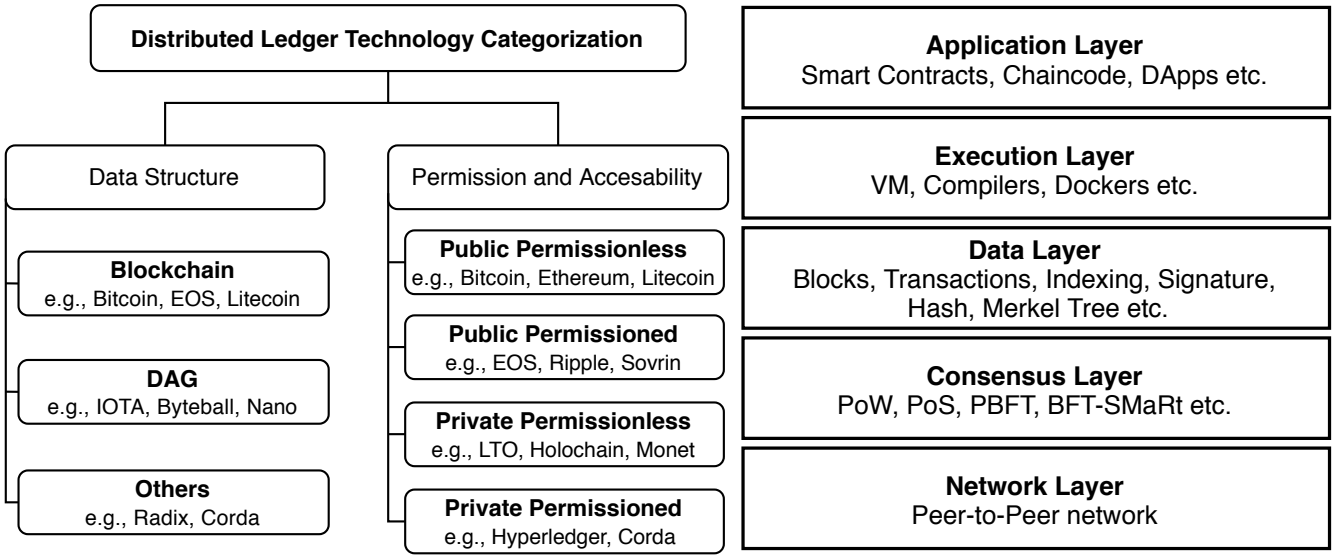


Figure 2.1: Basics of distributed ledger technologies. (a) Categories (b) Abstraction layers

2.2 DLT Abstraction Layers

Dinh *et al.* [29] introduced a blockchain design comprised of four identified abstraction layers, namely application, execution engine, data model and consensus. In the Oracle blockchain guidance book [2], the authors defined five layers to display the general architecture of blockchain, including the application and presentation layer, consensus layer, network layer, data layer and hardware/infrastructure layer. To better describe the architecture of DLT for the purpose of performance evaluation, we formulate an abstraction layer architecture following mainly Dinh's model [29] but extend it to five layers shown in Figure 2.1b.

- **Application layer** contains the applications that are mainly used by the end users such as cryptocurrency, wallet and DApps.
- **Execution layer** is in charge of executing a contract or low-level machine code (bytecode) in a run-time environment installed on DLT network nodes. Some examples are Ethereum Virtual Machine (EVM), Java Virtual Machine (JVM) and Docker container.
- **Data layer** defines a wide range of data-related topics such as transaction models, data structure, Merkel trees, hash function, encryption algorithms, etc.
- **Consensus layer** is the core of a DLT system. It sets the rules and forces all nodes to follow them to reach an agreement (e.g., transaction confirmation) on blockchain content.
- **Network layer** provides the fundamental peer-to-peer communication for a DLT system. It takes care of peer discovery, transactions, and block propagation.

2.3 DAG Distributed Ledgers

DAG-based distributed ledger, also called DAG distributed ledger or DAG ledger, is a promising DLT that stores transaction data as vertices of a directed acyclic graph. Different new transactions can be appended to different vertices in a DAG at the same time. Compared to the blockchain, which bundles transactions in blocks and stores blocks one by one to a chain, the DAG ledger naturally obtains better concurrency, as shown in Figure 2.2. Therefore, it has many advantages over blockchain in transaction throughput, network scalability and resource efficiency.

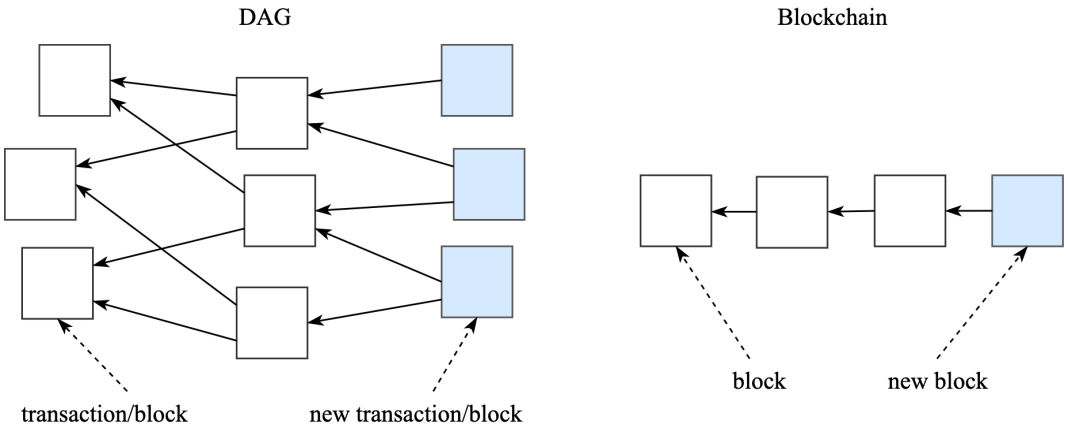


Figure 2.2: Structure comparison between DAG and blockchain.

DAG ledger and blockchain are the two most popular DLTs. According to the graph vertex granularity, DAG ledgers can be further divided into two categories: block-based (blockDAG)

and transaction-based (TDAG) [132]. The former first wraps transactions into blocks, and then appends the blocks onto a DAG. Some examples are CDAG [51], PHANTOM [120] and SPECTRE [119]. The latter attaches transactions to a DAG immediately and directly without waiting for block composition. Three notable examples are IOTA [103], Byteball [26] and Nano [73]. Since blockDAG is still in the conceptual stage, we focus our research on TDAG. For simplicity, DAG refers to TDAG throughout this report without further specification.

In the design of DAG ledgers, each end user issuing a transaction also needs to validate the previous transactions. All participants act as validators and contribute to maintaining the network. In other words, no miners are needed in DAG distributed ledger systems. For example, IOTA (see Section 2.4) requires the network participant or node to approve two previous transactions in order to issue a new transaction, while Byteball [26] encourages referencing all unapproved transactions. In addition, multiple transactions can be added to a DAG simultaneously and concurrently to increase system throughput and scalability.

2.4 IOTA

IOTA is a DAG-based open-source value transfer platform designed for the Internet of Things (IoT), with feeless microtransactions and data integrity for machines. It is currently the most popular representative of DAG ledgers. Since its first release in 2016, there have been three official versions: IOTA1.0 IRI (IOTA Reference Implementation, deprecated in 2020), IOTA1.5 Chrysalis, and IOTA2.0 Coordicide.

IOTA1.0 IRI

IOTA1.0 IRI was an early implementation of IOTA written in Java. In IOTA1.0, all transactions are linked together to form a directed acyclic graph (see Figure 2.3), called *Tangle*. A transaction is a fundamental operation unit that can stand alone. A *tip* is a newly issued but not approved/validated transaction. Each transaction has its weight and a cumulative weight. The *weight* reflects the computation resource that a sending node puts into this transaction. In order to issue a new transaction, a node must select two tips to validate. Once the validation is finished, this node will attach the newly issued transaction to the selected tips. This attachment is called a *reference*, through which all transactions are linked together. At the same time, the newly issued transaction adds its weight to all the predecessors' cumulative weight. For example, all the referenced transactions' (v_1 to v_9) cumulative weights in Figure 2.3 will add one after v_{10} is attached to the Tangle.

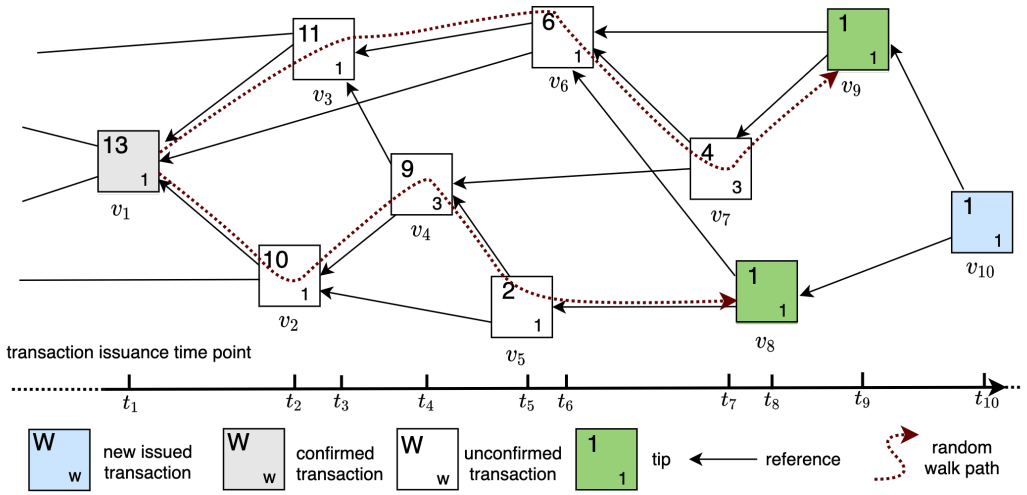


Figure 2.3: An example of DAG in IOTA. "w" stands for weight, and "W" stands for cumulative weight.

To find a good balance between punishing lazy behaviour and not leaving too many tips behind, IOTA1.0 recommends a sampling approach named *weighted Markov chain Monte Carlo (MCMC)* random walk or *weighted MCMC* to select two tips. For example, the selected tips in Figure 2.3 are v_9 and v_8, with the random walk paths $(v_1, v_3, v_6, v_7, v_9)$ and $(v_1, v_2, v_4, v_5, v_8)$, respectively. This approach leverages Equation (2.1) [103] to calculate the probability of choosing the next particular approver for each step in a random walk.

$$P_{\mathrm{xy}} = \frac{e^{\alpha H_{\mathrm{y}}}}{\sum_{z:z \rightsquigarrow x} e^{\alpha H_{\mathrm{z}}}} \tag{2.1}$$

where P_{xy} is the probability to walk from transaction x to y, H_{y} is the cumulative weight of y, and $z \rightsquigarrow x$ means "z directly approves x". Therefore, in the random walk step, a transaction with a higher cumulative weight has a much higher probability of being selected and approved. In other words, the probability of walking from x to y increases exponentially with the cumulative weight of y, multiplied by α. For example, using Equation (2.1), the probability walking from v_1 to v_3 in Figure 2.3 is calculated as $P_{v_1 v_3} = \frac{e^{11\alpha}}{e^{11\alpha}+e^{10\alpha}+e^{6\alpha}}$. Here, α is a parameter regulating the strength of the bias in the weighted random walk. A higher α value means that heavier tips are favoured when attaching new transactions to the DAG. If we set α to zero, the random walk is called *unweighted MCMC*. If we set α to a very high value, we get the super-weighted walk. We can also use a uniform random tip selection (URTS) approach to select two from all available tips for comparison.

At its early stage, IOTA relies on a centralized node, called Coordinator (COO), controlled by IOTA Foundation (IF) as the consensus mechanism to ensure network security. Coordinator is a special participant acting as a "finality device" to confirm transactions by periodically generating

zero-value transactions, called *milestones*. With Coordinator, IOTA leverages the above-discussed weighted (or biased) TSA to deal with conflicts and simply considers a transaction as confirmed if and only if a milestone references it. In this consensus, an issued transaction will continuously get approved/validated by peer nodes until it eventually gets confirmed. The more references a transaction gets, the more trustable and acceptable it becomes. As such, it has a higher probability of being referenced (indirectly) by the next coming milestone and thus confirmed.

IOTA1.5 Chrysalis

IOTA1.5 is an intermediate project before switching to IOTA2.0, where the centralized Coordinator will be completely removed. IOTA1.5 was rewritten from the ground up using Golang. Though this version kept the Coordinator, it implemented significant improvements in performance, stability, reliability, and security. First, controversial decisions such as ternary encoding and quantum-proof cryptography, namely the Winternitz One-Time Signature (W-OTS) scheme, were left behind and replaced with established standards, i.e., Ed25519. Second, for the transaction structures, IOTA1.5 replaced the concept of bundles with Atomic transactions to reduce network overhead, improve signature verification speed, and achieve better spam protection and overload control. Third, Chrysalis adopted the unspent transaction output (UTXO) transaction model instead of the old account-based scheme in the IRI. Last but not least, IOTA1.5 switched to the uniform random tip selection (URTS) algorithm for to achieve a much higher tip selection performance than the MCMC random walk mechanism.

Besides these improvements, IOTA1.5 also added new functionalities and features. For example, it introduced auto-peering to automatically connect a new Hornet node to several healthy neighbour network nodes. In addition, it utilized the white-flag approach to calculate credits and improve the speed and efficiency of tip selection while eliminating certain attacks (e.g., conflict spam) and significantly reducing the need for reattachments. Consequently, the milestone interarrival time was reduced to 10 seconds from 60 seconds in IOTA1.0.

IOTA1.5 was launched in April 2021. It is also the currently running version on the main network. Based on the significant improvements in performance and security, this version is ready for industrial applications. To bootstrap a private IOTA tangle for enterprise, we can simply generate a tangle snapshot as the genesis state and add nodes (e.g., COO, spammer, and extra nodes) to the network.

IOTA2.0 Coordicide

IOTA2.0 aims to build a completely decentralized DLT network without the Coordinator. With Coordicide, IOTA proactively resolves conflicts and reaches consensus through a voting mechanism called fast probabilistic consensus (FPC) rather than the weighted MCMC or URTS TSA [104]. To achieve this, IOTA2.0 adopts a reputation mechanism called Mana, which is a scarce resource in the network used to control transaction rate, provide a reference for FPC voting, determine the dRNG, and prevent Eclipse attacks in auto-peering. The FPC is a leaderless probabilistic binary consensus protocol with low communication complexity that is robust in a Byzantine infrastructure. The basic idea behind the FPC is majority voting, where every node queries k random nodes at each round and set its opinion equal to the majority of the queried nodes. In this way, all nodes can reach a consensus on the state of the ledger at any time without the involvement of the centralized Coordinator.

IOTA2.0 was deployed in late 2020. Many of its features, including smart contracts, are still in the development and test phase. As of the time of writing (January 2023), this latest version focuses on its functionality improvement and has not considered the performance improvement yet. So, it still has a long journey to go before becoming industry-ready, as claimed on its official site.

2.5 Hyperledger Besu and IBFT

Hyperledger Besu is an open-source Ethereum client developed for enterprise usage. It is written in Java and supports various consensus algorithms, including Proof of Work (e.g., Ethash) and Proof of Authority (e.g., Clique, IBFT 2.0 and QBFT). Besu provides enterprise features such as privacy and permissioning. Among all the consensus algorithms, IBFT 2.0 is recommended by Besu to ensure immediate finality and high robustness, which keeps a balance between security and transaction performance.

IBFT [89] was first implemented in Geth by Amis Technologies[1] around early 2017, and soon in Quorum[2], which is the first Enterprise Ethereum client. As a leader-based (or voting-based) consensus, IBFT is deterministic and can tolerate f faulty participants out of n, where $n \leqslant 3f+1$. It inherits from the original PBFT by achieving consensus in three phases, *pre-prepare*, *prepare* and *commit*, and has $O(n^2)$ total communication complexity. Before each round, the selected proposer will propose a new block proposal and broadcast it along with the *pre-prepare* message.

[1] https://www.amis.com/
[2] https://consensys.net/quorum/

Upon receiving the *pre-prepare* message, other validators enter the state of *pre-prepared* and then broadcast *prepare* message. When receiving $2f + 1$ of *prepare* messages, the validator enters the state of *prepared* and then broadcasts *commit* message. Finally, validators wait for $2f + 1$ of *commit* messages to enter *committed* state and then attach the proposed block to the chain.

Then, Saltini and Hyland-Wood [110] analyzed the correctness of this protocol and showed that it does not guarantee Byzantine-fault-tolerant consistency and liveness when operating in an eventually synchronous network. They further resolve the liveness and finality issues in IBFT 1.0 by proposing a new version, IBFT 2.0 [111]. Specifically, they modified the number of nodes for reaching consensus from $2f+1$ to super-majority and also improved the round change by removing the need for the block-locking mechanism in IBFT 1.0.

3 A Systematic Survey for Performance Evaluation of Blockchain Systems

In this work [37], we present a comprehensive and systematic survey on blockchain performance evaluation. The survey covers existing studies on evaluating the performance of various mainstream blockchains and compares the advantages and disadvantages of the evaluation approaches they use. It addresses the following research questions:

RQ1. What are the current mainstream techniques, main evaluation metrics and benchmark workloads for blockchain performance evaluation?

RQ2. How to comparatively evaluate the performance of two blockchain systems with different consensuses?

RQ3. What are the significant bottlenecks identified in various blockchain systems?

RQ4. What are the main challenges and opportunities in blockchain performance evaluation?

All the reviewed evaluation approaches can be classified into two high-level groups: empirical evaluation and analytical modelling, as shown in Figure 3.1.

Empirical evaluation includes benchmarking, monitoring, experimental analysis and simulation. Analytical modelling mainly covers three types of stochastic models: Markov chains, queueing models and stochastic Petri nets (SPNs). Through this classification, we aim to depict a big picture of the performance evaluation landscape, identify current challenges in this area, and provide suggestions for future research.

3.1 Empirical Analysis in Blockchain Performance Evaluation

In this section, we investigate existing approaches to blockchain performance evaluation from the perspective of empirical analysis. Specifically, different solutions, including benchmarking, monitoring measurements, self-designed experiments and simulation, are reviewed and compared.

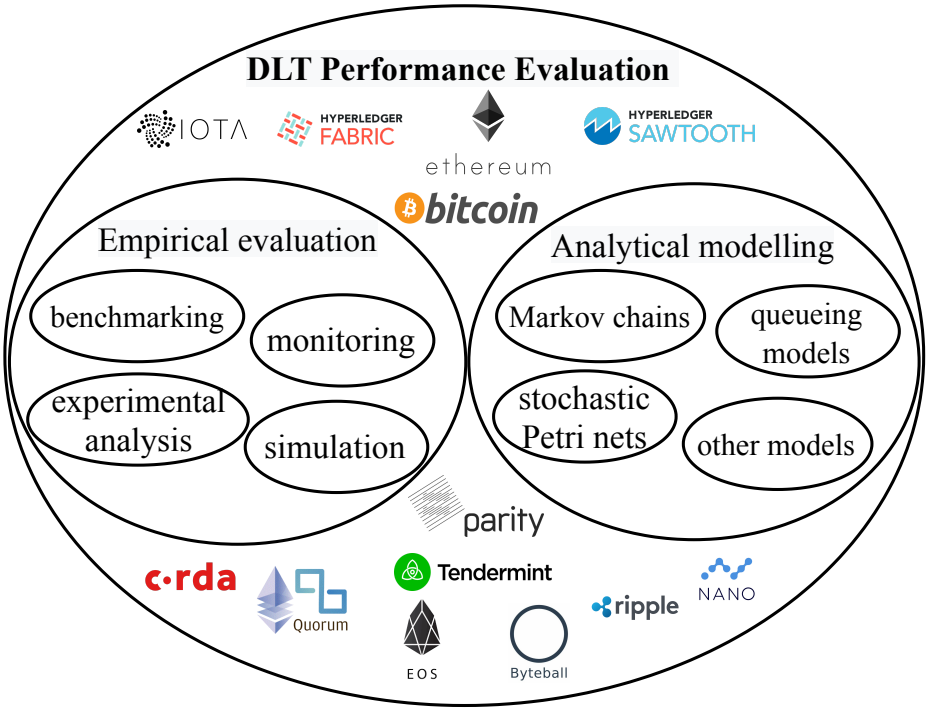


Figure 3.1: A landscape of DLT performance evaluation approaches and evaluated ledgers.

In practice, these approaches are usually used together to provide more evidence for blockchain performance evaluation.

3.1.1 Benchmarking

Up to date, there are three popular performance benchmarking tools dedicated to evaluating blockchain systems, as listed in Table 3.1.

Blockbench [29] is the first benchmark framework designed for evaluating private blockchains in terms of performance metrics on throughput, latency, scalability and fault-tolerance. Presently, it supports measurement on four major private blockchain platforms, namely Ethereum, Parity, HLF and Quorum. However, it claims to support the evaluation of any private blockchain by accordingly extending the workload and blockchain adaptors.

In the design of Blockbench, four abstraction layers in blockchain are identified: consensus, data model, execution engine and application, from the bottom (low level) to the top (high level). The consensus layer is in charge of setting the rule of agreement and getting all network participants to agree on the block content so that it can be appended to the blockchain. The data model defines the data structure, content and operations of the blockchain data. The execution engine contains resources of the runtime environment, such as the EVM and Docker, which support the execution operations of blockchain codes. The application layer includes all kinds of blockchain applications,

Table 3.1: A comparison of three popular blockchain benchmarking tools

Tools	Supported DLTs	Workloads Used	Evaluated Metrics	Pros & Cons
Blockbench [29]	Ethereum, Hyperledger Fabric, Parity and Quorum.	• macro: YCSB(k-v store), Smallbank(OLTP), EtherId, Doubler, and WavesPresale • micro: DoNothing, Analytics, IOHeavy, and CPUHeavy	throughput, latency, scalability and fault tolerance.	adaptor-based framework, scalable; carefully designed workloads; but they are constant.
DAGbench [30]	IOTA, Nano and Byteball	• value/data transfer • transaction query: input/output transaction numbers and balance for a given account	throughput, latency, scalability, success indicator, resource consumption, transaction size and fee	adaptor-based framework, scalable; specific for DAG DLT; workloads are not representatives.
Hyperledger Caliper [98]	Hyperledger Fabric, Hyperledger Besu, Ethereum, FISCO BCOS	Self-defined in the configuration file	throughput, latency, resource consumption, success rate	adaptor-based framework, scalable; no pre-defined workload design, but support more DLT systems.

such as smart contracts and different types of DApps.

Hyperledger Caliper [98] is a performance evaluation framework mainly focusing on benchmarking Hyperledger blockchains such as Hyperledger Fabric, and Besu. In the system architecture, there are two main components: Caliper core and Caliper adaptors. The former defines system workflow, while the latter is used to extend evaluation for other blockchains such as Ethereum and FISCO BCOS. Before running a test, benchmark workloads and necessary information interfacing adaptor to the system under test (SUT) need to be predefined in configuration files. During the test, a resource monitor runs to collect resource utilization information (e.g., CPU, RAM, network and IO), and all clients publish their transaction rate control information to a performance analyzer. When a test is finished, a detailed test report is generated by a report generator.

DAGbench [30] is a relatively recent framework dedicated to benchmarking DAG distributed ledgers such as IOTA, Nano and Byteball. The currently supported indicators are throughput, latency, scalability, success indicator, resource consumption, transaction data size and transaction fee. From the system design perspective, DAGbench shares the same approach with Blockbench and Caliper which adopt a modular adaptor-based architecture. Users just need to choose (or

develop if not available) associated adaptors for different workloads and blockchain systems under evaluation.

Besides the general performance metrics evaluation, there are also studies focusing on specific metrics for particular blockchains. For example, OpBench [5] and another benchmark framework [6] are proposed to evaluate if a miner's award is proportional against to the CPU execution time or consumed computation power for Ethereum smart contracts.

3.1.2 Monitoring

Blockchain benchmarking usually requires a standardized environment and a well-documented workload as input. However, for public blockchain systems, it is impossible to have good control against the real workload and consensus participants, which makes benchmarking more challenging. In terms of evaluating public blockchains, there are two potential solutions.

The first solution is to build a private version of the associated test network and leverage the existing benchmarks mentioned above to evaluate blockchain under artificially designed workloads. This may require new adapter development for either workload or blockchain network. In addition, this approach should take into consideration the scalability problem of blockchain because the tested private version of blockchain may encounter scaling issues when implemented publicly. Therefore, the tested result may show better values of performance metrics compared to the real public network.

The second solution is to monitor and evaluate the live public system's performance under a realistic workload [95]. Zheng *et al.* [140] proposed a detailed, real-time performance monitoring framework using a log-based approach. It has lower overhead, more details, and better scalability compared to its counterpart solution via remote procedure call (RPC).

3.1.3 Experimental Analysis

Even though experimental analysis can hardly provide standardized test results like benchmarking, this approach is very flexible in parameterization. It can be used to identify potential bottlenecks and pave the way to further performance improvements.

Experiment-based approaches have been widely employed to evaluate distributed ledger systems such as Hyperledger, Ethereum and DAG-based ledger. Various private blockchain platforms and different versions of a certain blockchain can be compared on performance by running tests under a well-controlled test environment. In addition, some studies examined the detailed performance, for example, the performance of different encryption and hash algorithms, from the data

Table 3.2: Overview of different DLT performance under various test environments

DLT	Consensus	Throughput (TPS)	Latency (Secs)	Workload	Network Size	Node Config
HLF v0.6 [29]	PBFT	1273	38	YCSB	8 nodes	E5-1650 3.5GHz CPU, 32GB RAM, 2TB HD
	PBFT	1122	51	Smallbank	8 nodes	
Geth v1.4.18 [29]	PoW	284	92	YCSB	8 nodes	
	PoW	255	114	Smallbank	8 nodes	
Parity v1.6.0 [29]	PoA	45	3	YCSB	8 nodes	
	PoA	46	4	Smallbank	8 nodes	
Quorum 2.0 [12]	Raft	2,000+	1.5	write-only/null	3 nodes	8 vCPUs
	IBFT	1,900	3.2	null	4 nodes	4 cores 3.6 GHz, 16GB RAM
	IBFT	1,800	3.5	write-only	4 nodes	
HL Sawtooth v1.1.2 [14]	Proof of Elapsed Time (PoET)	3	-	Smallbank	6 nodes	Dockers share VM on Intel Xeon X7350 CPU 16 Cores, 2.93GHz, 64GB RAM
EOS v1.5.3 [14]	Delegated Proof of Stake (DPoS)	21	-	Smallbank	6 nodes	
Ethereum Geth v1.8.21 [14]	PoW	10	-	Smallbank	6 nodes	
HLF v1.0 [53]	BFT-SMaRt	1,700	-	Payment transaction	16 nodes	E5-2630 CPU 8 cores 2.4GHz, 64GB RAM
HLF v0.6 [53]	PBFT	2600	1.8	Invoking chaincode	16 nodes	
Ripple v0.60.0 [53]	XRP	1450	6	Payment transaction	16 nodes	
Tendermint v0.22.4 [54]	PBFT and Casper	6,000	0.15	Invoke Payment transaction	16 nodes	E5-2630 CPU 4 cores 2.4GHz, 12GB RAM
Tendermint v0.22.4 [54]	PBFT and Casper	5,600	0.05	Query Payment transaction	16 nodes	
R3 Corda v3.2 [54]	BFT-SMaRt	50	8	Query Payment transaction	4 nodes	
Geth v1.7.3 [55]	PoW	130	1,297	YCSB(N=10,000)	4 nodes	8GB RAM, 128GB SSD
Geth v1.7.3 [55]	-	235	569	YCSB(N=10,000)	4 nodes	
HLF v1.0 [55]	BFT-SMaRt	535	78	YCSB(N=10,000)	4 nodes	
HLF v1.0 [55]	-	1,033	40	YCSB(N=10,000)	4 nodes	
Geth [109]	PoW	-	0.199	Payment transaction	1 node	Core i7-6700 CPU, 24GB RAM
Parity [109]	PoW	-	0.105	Payment transaction	1 node	
Geth 1.5.8 [100]	-	21	361	TransferMoney (N=10,000)	1 node	AWS EC2 Intel E5-1650 8 core CPU, 15GB RAM, 128GB SSD
HLF v0.6 [100]	-	160	4	TransferMoney (N=10,000)	1 node	

layer in the blockchain abstraction model. Table 3.2 lists an overview of various DLTs' performance extracted from the reviewed experimental analysis studies.

3.1.4 Simulation

All the evaluation solutions mentioned above (i.e., benchmarking, monitoring and experimental analysis) require the availability of the systems, no matter private or public blockchains. However,

the system under evaluation is not always available. For instance, when a company needs to make a selection between two blockchain platforms under development according to their performance, none of the previously discussed solutions is feasible. Moreover, it is usually costly in both time and resources to construct a real blockchain network for testing. This brings us to explore another evaluation approach, namely, *simulation*. A blockchain simulator can mimic the behaviours of network nodes in reaching a consensus, providing performance similar to a real system. Besides, a blockchain simulator usually provides a convenient way for users to tune the system parameters to run different settings for the sake of comparison. In this subsection, we will take a look at the role of simulation in the blockchain world.

BlockSIM In 2019, there were three similar simulators with the same name, *BlockSim* (or *BlockSIM*), proposed for simulating blockchain systems. Alharby and Moorsel [7] proposed and implemented a framework called BlockSim to build discrete-event dynamic system models for PoW-based blockchain systems. This framework was organized in three layers: incentive layer, connector layer and system layer. Using the proposed simulation tool, the authors explored the block creation performance under the PoW consensus algorithm. This simulator helped to understand the details of the block generation process in PoW. The predefined test cases were validated and verified in their extension study, where the simulation outcomes were compared with results of real-life systems such as Bitcoin and Ethereum to show the feasibility of this approach. However, the extensibility of this simulator is still a problem for future research.

To help architects better understand, evaluate and plan for the system performance, Pandey *et al.* [96] proposed and developed a comprehensive open-source simulation tool called BlockSIM for simulating private blockchain systems. This tool is designed to evaluate system stability and transaction throughput (TPS) for private blockchain networks by running scenarios and then deciding on the optimal system parameters suited for the purposes of architects. The comparison results between BlockSIM and a real-world Ethereum private network running PoA consensus show that BlockSIM can be used effectively.

More recently, Faria and Correia [40] presented a flexible discrete-event simulator (also called BlockSim) to evaluate different blockchain implementations. With a good design of APIs, new blockchains can be easily modelled and simulated by extending the models. Running this simulator for Bitcoin and Ethereum, the authors got some interesting performance conclusions. For instance, doubling the block size (number of transactions) had a small impact on the block propagation delay (10ms), while encrypting communication had a higher impact on that delay (more than 25%).

DAGsim Similarly, Zander *et al.* [136] presented a continuous-time, multi-agent simulation framework called *DAGsim*, for DAG-based distributed ledgers. Specifically, the performance of

IOTA in terms of the transaction attachment probability was analyzed using this tool. The results indicate that agents with low latency and high connection degrees have a higher probability of having their transactions accepted in the network. Another multi-agent tangle simulator [17] built with NetLogo simulates both random uniform and MCMC tip selection in a visualized and interactive way.

3.1.5 Comparison of Different Empirical Evaluation Approaches

From the perspectives of each approach's general characteristics and its suitability in evaluating different types of blockchains, we comparatively discuss the advantages and disadvantages of the above-mentioned solutions. The compared items are divided into two categories: solution requirements and solution efficiency, see Table 3.3. Solution requirements describe the network specifications for evaluating blockchain systems in terms of the node, network and workload. Solution efficiency provides three dimensions, namely parameterization, extensibility and difficulty, to compare the efficiency and effectiveness of different solutions.

Table 3.3: A comparison of different empirical performance evaluation solutions for blockchain

Solutions	**Characteristics**			**Efficiency**		
	Node	**Network**	**Workload**	**Parameterization**	**Extensibility**	**Difficulty**
Monitoring	Real	Real	Real	Low	Low	Easy
Benchmarking	Real	Test	Artificial	Low	High	Easy
Experimental Analysis	Real	Test	Artificial	High	Low	Medium
Simulation	Virtual	Virtual	Virtual	Very High	Very High	Hard

3.2 Analytical Modelling in Blockchain Performance Evaluation

In this section, we survey the performance evaluation solutions of distributed ledger systems based on analytical modelling. We aim to summarize the mainstream techniques, explore how and why these models are employed for certain distributed ledgers, and then identify the current challenges in blockchain performance modelling. In particular, we focus on surveying the stochastic models, which have been used to successfully model many cloud systems.

In Table 3.4, the existing popular solutions for blockchain performance modelling are classified into four categories: Markov chains, queueing models, stochastic Petri nets and other models.

Table 3.4: A summary of blockchain performance modelling studies

Model Types	Models	Consensus	DLs	Model Outputs
Markov chains	Absorbing Discrete Time Markov chain (DTMC) [59]	Raft	private blockchains	network split probability as a function of packet loss rate, election timeout, and network size
	Discrete Time Markov chain (DTMC) [19]	the tangle	IOTA	cumulative weight and transaction confirmation delay
queueing models	$M/G^B/1$ queue variant [63]	PoW	Bitcoin	tx confirmation time
	CTMC GI/M/1 queue [77]	PoW	Bitcoin	mean number of txs in the queue and in a block; average tx-confirmation time.
	CTMC GI/M/1 queue [78]	PoW	Bitcoin	mean stationary number of txs in the queue and in the block
	M/G/1 queue variant [108]	PoW	Bitcoin	confirmation time and tx delay
	non-exhaustive queue [139]	PoW	NA	mean number of txs and mean confirmation time of txs in the system
	Discrete-time $GI/GI^N/1$ queue [47]	Proof-of-Authority	Ethereum	system queue size and tx waiting time
	$M/M^B/1$ queue [48]	BFT-SMaRt	HLF	tx latency
	M/G/1 and M/M/1 queue [4]	PBFT	NA	system delay
	(n,k) fork-join queue [66]	vote-based consensus	permissioned blockchain	blocks commitment delay, block validation response time and synchronization processes among mining nodes.
	Fluid queue [41]	the tangle	IOTA	conflicting txs cannot coexist when a random tip-selection algorithm is employed
stochastic Petri nets	Generalized stochastic Petri nets (GSPN) [133]	PBFT	HLF v1.2	latency and throughput of each phase
	Stochastic Reward Nets(SRN) [122]	PBFT	HLF v1.0	throughput, utilization and mean queue length at each peer
other models	World State Prediction model [138]	PoW	Ethereum	transaction time cost
	Stochastic network model [97]	PoW	Ethereum	tx processing rate
	Random Graph model [117]	PoW	Bitcoin	block propagation delay and traffic overhead

3.3 Findings and Future Research Directions

We summarize the main findings from previous evaluation sections. First, we discuss the findings from the empirical and analytical evaluations. We then take a look at the performance bottlenecks identified from all reviewed solutions. Finally, we point out some open issues and provide suggestions for future research.

3.3.1 Findings for Empirical Analysis

Performance metrics and workloads: The evaluated performance metrics can be divided into two categories: macro (or overall) metrics and micro (or detailed) metrics. *Macro metrics* provide an overview of the system's performance for users from the application level, such as transaction throughput, latency, scalability, fault tolerance, transactions per CPU/memory second/disk IO/network data. The first two metrics (transaction throughput and latency) are evaluated most frequently in overall blockchains. *Micro metrics* depict the performance of different subprocesses of transactions or specific layers in the blockchain abstract model for developers, such as peer discovery rate, RPC response rate, transaction propagating rate, contract execution time, state updating time, consensus-cost time, encryption and hash function efficiency. Both macro and micro metrics are evaluated under well-designed workloads.

Evaluated blockchains: HLF (v0.6 with PBFT and v1.0+ with BFT-SMaRt), private Ethereum (Geth with PoW and Parity with PoA/PoW) and Ripple with XRP consensus are the most often comparatively evaluated blockchain platforms [29], [55], [100], [109]. Among them, HLF and Ripple can reach 1,000+ TPS within a small network and outperform the Ethereum platforms in terms of throughput and latency under both macro and micro benchmark workloads. However, because of the underlying consensus algorithms they use, both HLF and Ripple fail to scale beyond a certain number of nodes in the network (e.g., 16 [29] for HLF v0.6).

Consensus finality: *Consensus finality* refers to the deterministic property of a blockchain where a block is considered confirmed once it is appended to the ledger. BFT-based blockchains are all with consensus finality, while PoW-based are usually not. This property has a direct impact on the transaction latency. For example, Bitcoin usually requires six successive confirmations as a secure finality that a transaction will not end up being pruned and removed from the blockchain, which makes the latency reach an unacceptable time of almost one hour. In contrast, HLF with BFT-based consensuses can finalize a transaction within seconds, right after it is appended to the ledger. Therefore, BFT-based blockchains have an obvious advantage over PoW-based blockchain systems in terms of performance.

3.3.2 Findings for Analytical Modelling

Most models neglect information propagation delays in the network and simply collapse the whole network into a single node that provides service to process and confirm transactions. These models are usually queue systems that provide bulk services such as $M/M^B/1$ and $M/G^B/1$ queue. Only a small portion of models consider the system as separate disjoint nodes and take the network latency among network nodes into consideration. They aim to calculate system end-to-end output (e.g., delays) using queue networks or to cascade different queues, such as M/G/1 and M/M/1, together to model the blockchain network.

An (n, k) fork-join queue combines both modelling strategies. It first regards the system as a single server when the system receives a job request. Then, it splits the job into several sub-tasks for independent and simultaneous processes on different network nodes. In the joint phase, process results are collected from different nodes to finish the original job (e.g., block validation).

3.3.3 Identified Performance Bottlenecks

All bottlenecks identified in the reviewed papers are listed in Table 3.5.

As we can see, most bottlenecks are still unresolved. This means that corresponding effective solutions to solve the performance problems have not yet been found. Another observation is that most bottlenecks are identified by empirical analysis, which can be attributed to two reasons. First, there are more empirical analyses conducted than performance modelling. Second, due to the involved mathematical expressions, analytical modelling is much more difficult than experimental solutions in exploring the impact of design parameters. In blockchain performance modelling, even one simple extra parameter can significantly increase the model complexity. Therefore, empirical analysis becomes more efficient and popular in bottleneck identification than its modelling counterpart.

3.3.4 Future Research Directions

Here, we identify some open issues and suggest potential directions for future research in this area.

- For empirical analysis, difficulties lie in comparative evaluation among different blockchain platforms, especially for those with very different consensus algorithms and data structures. The main reason is the lack of interface standards in running workloads. One solution is to design an equivalent workload, such as transferring a unity amount from account A to another account B [53]. However, this approach has limited extensibility and requires deploying a dedicated workload for each blockchain under evaluation. Thereby, there is

Table 3.5: Identified performance bottlenecks for different blockchain systems

Blockchain	**Bottlenecks Identified**	**Evaluation Approaches**	**Latest State**
Ethereum v1.5.9	peer discovery, transactions propagation, consensus-cost	Monitoring [140]	Unresolved
Geth v1.4.18	consensus protocols	Benchmarking [29]	Unresolved
HLF v0.6.0	consensus protocols	Benchmarking [29]	Unresolved
Parity v1.6.0	transaction signing	Benchmarking [29]	Unresolved
HLF v1.0	endorsement policy verification, sequential policy validation of transactions in a block, and state validation and commit (with CouchDB)	Experimental analysis [124]	Resolved (HLF v1.1)
Byteball	data storage which is a relational database	Benchmarking [30]	Unresolved
HLF v1.0	no parallel transaction processing on the committing peer	Experimental analysis [30]	Unresolved
HLF	ordering service	Experimental analysis [48]	Unresolved
Private Ethereum	module responsible for reading and writing data	Experimental analysis [24]	Unresolved
Private Ethereum	consensus mechanism	Experimental analysis [55]	Unresolved
HLF	consensus mechanism	Experimental analysis [55]	Unresolved
HLF v1.0+	transmission from client to the ordering service and ledger write	Analytical modelling [122]	Resolved
HLF v1.2	committing phase if the number of transactions in a block is small and endorsement phase if it is large	Analytical modelling [133]	Unresolved

great potential for future research to develop more extensible tools for the comparative evaluation of blockchain platforms.

- Many methods of experimental analysis rely on RPCs to communicate with blockchain consensus nodes and collect transaction statistic data (e.g., the total number of confirmed transactions of a certain duration). Although the RPC API protocols (e.g., gRPC and JSON-RPC) claim to be efficient, they still induce extra overhead onto the consensus peers [140], which is counted as the peer consumption and, in turn, makes the evaluation results inaccurate. Therefore, a more lightweight and low-overhead data collection approach, such as a log-based approach [140], deserves more attention in the future.

- RPC methods are widely used for data collection in empirical performance evaluation of blockchain systems. For micro metrics and micro workload design, it is challenging to de-

couple the impact from other layers. For example, two queries on transaction values are designed to evaluate the *data model* performance. For Ethereum, both queries can be easily implemented via invoking JSON-RPC APIs. However, for HLF, a chaincode (VersionKVStore) must be implemented, as there are no APIs to query historical states in the system. Inevitably, this involves the execution of a smart contract making the evaluation inaccurate by adding extra overhead. Therefore, for a detailed evaluation of performance metrics in specific blockchain abstraction layers, it is an open issue to design a reasonable workload that alleviates the impact of other layers and improves accuracy.

- Besides the classic blockchain systems such as HLF and Ethereum, there is an urgent need to evaluate the performance of their proposed improvements. For example, *sharding* claims to be a promising solution and has been implemented in many blockchains. However, there is no evaluation work for comparing different shard-based blockchain systems. Different solutions, such as sharding v.s. DAG and off-chain v.s. side-chain also needs to be comparatively evaluated. In addition, it would be beneficial to combine empirical and analytical approaches in blockchain performance evaluation in the future.

3.4 Summary

In this work, we present a systematic survey covering existing blockchain performance evaluation approaches. From the high-level perspective, they can be categorized into *empirical* and *analytical* evaluation methods. The empirical analysis can be further divided into four groups: performance benchmarking, monitoring, experimental analysis and simulation. Three popular benchmark frameworks (i.e., Blockbench, DAGbench and Hyperledger Caliper) are introduced and comparatively analyzed. Performance monitoring is recognized as the best solution for the performance evaluation of public blockchain. Analytical modelling approaches are more powerful than empirical solutions, especially for analyzing the consensus layer of blockchain systems. There are three main types of modelling approaches compared in this survey: Markov chains, queueing models and stochastic Petri nets. This comparison can provide directions for selecting a blockchain evaluation approach suitable for a given purpose. We also summarized the results of surveyed performance evaluation studies and identified the bottlenecks of major blockchain platforms. This survey concludes with the identification of open issues and ascertainment of future research directions in this important area.

4 Performance Analysis of the IOTA DAG-based Distributed Ledger

In this work [36], we investigated the performance of IOTA using simulation, experimental and analytical approaches. For a distributed ledger (DL) system, the transaction throughput, average waiting time (transaction delay) and system scalability are essential [69], as they reflect the transaction processing capacity, usability and extensibility of the system. As an alternative solution to the traditional blockchain, the DAG-based IOTA claims to be scalable due to its innovative design. In our previous work [38], we designed an energy transactive system for smart communities using IOTA and explored the system's scalability. Our experimental results show that the proposed system is scalable and effective for IoT use cases. In this work, we focus on the system performance analysis, including throughput and transaction reattachment waiting time, of a DAG-based DL for P2P-ET.

It is critical for an end user to know the optimal waiting time before reattaching the submitted but not yet confirmed transactions. If the waiting time is too short, the premature redundant transactions cause network congestion; if the waiting time is too long, the user experience declines, as does the system throughput. Either way leads to poor system efficiency. Therefore, in this research work, we strive to answer two vital research questions about the performance of a private IOTA network:

RQ1. Which factors influence the throughput of an IOTA system? And how, quantitatively, do they impact the throughput?

RQ2. What is the optimal waiting time for confirmation before resubmitting the same original transaction?

4.1 IOTA Performance Simulation

To conduct the simulation, we leverage and extend the DAGsim simulator [135], which is an asynchronous, continuous time, and multiagent simulation framework for DAG-based distributed

ledgers. In addition to the analysis of simulation results, we run experiments in a private IOTA network to validate the results. For better understanding, all symbols and acronyms used for performance analysis are summarized in Table 4.1.

Table 4.1: Descriptions of all used symbols and acronyms

Notation	Description
λ	average transaction arrival rate in transactions per second
λ_{M}	average milestone arrival rate in transactions per second
α	randomness factor for weighted random walk, $\alpha \geq 0$
$agents$	the total number of nodes in a simulated network
d	distance between simulation agents, reflects network delay
D	distances matrix for all agents
$CTPS$	confirmed transactions per second, reflects transaction throughput
RWT	reattachment waiting time
DL	distributed ledger, e.g., blockchain and DAG ledger
DAG	directed acyclic graph, a finite directed graph with no cycles
$Tangle$	DAG structure in IOTA, formed by all connected transactions
COO	coordinator, generates milestones to confirm transactions, also refers to IOTA consensus based on the coordinator
$MCMC$	Markov chain Monte Carlo, a sampling approach for IOTA tip selection
$URTS$	uniform random tip selection
TSA	tip selection algorithm, e.g., weighted/unweighted MCMC and URTS
$IREA$	indirect references extraction algorithm
PoW	proof of work, a protection mechanism asking client to solve a cryptographic puzzle before issuing transactions

To explore the influence of some impact factors on IOTA's transaction throughput, an empirical study is conducted through simulations. These factors include transaction arrival rate λ, TSAs, network latency d and randomness parameter α of the weighted MCMC TSA. For other tested parameters such as network size, PoW difficulty, and the number of Coordinators, refer to our previous work [38] for details. Before presenting the simulations, we first talk about the simulation tool.

4.1.1 IOTA Simulator Extension

COO consensus, based on the DAGsim simulator [135], has been developed by the authors to perform the simulations. As the original DAGsim only supports consensus without a COO, no milestones are generated in between regular transactions. We extended this simulator to support COO consensus by introducing milestones. Technically, we configured the extended DAGsim to generate and broadcast milestones to the network every 60 seconds, just like the currently running

IOTA mainnet. The generated milestones are acting exactly like regular transactions, but with the ability to confirm transactions. Our extended version of DAGsim is available publicly[1].

When we want to find out all the confirmed transactions in the simulation data, according to the definition of the Coordinator consensus mechanism, we simply check if a transaction is directly or indirectly referenced by a milestone. In the original DAGsim simulator, for each transaction, we only have access to the transactions that are directly referenced by this transaction in the simulation data. However, we also need indirect references when using COO consensus. To fetch this information, we propose a recursive solution named *Indirect References Extraction Algorithm (IREA)* as shown in Algorithm 1.

Algorithm 1 Indirect References Extraction Algorithm

```
1: indirect_references = empty
2: function FIND_REFERENCES(tx, direct_references)
3:     APVD_1 = The 1st transaction approved by tx
4:     APVD_2 = The 2nd transaction approved by tx
5:     if tx is genesis then
6:         End
7:     else if tx is in indirect_references then
8:         Append the new TXs to indirect_references
9:         End
10:    else
11:        if APVD_1 is not in indirect_references then
12:            Append APVD_1 to indirect_references
13:        find_references(APVD_1, direct_references)
14:        if APVD_2 exists then
15:            if APVD_2 is not in indirect_references then
16:                Append APVD_2 to indirect_references
17:            find_references(APVD_2, direct_references)
```

It utilizes a recursive function to find the transactions directly referenced by the input transaction. According to the random walks, we know that there are always two (or at least one of the walks overlap) transactions directly referenced by any issued transaction. These two transactions are added to a list, and the recursive function is run again for each of them. This goes on until the genesis,i.e., the first transaction, is reached or a transaction that is already in the list is encountered. By running this algorithm, all transactions confirmed by a milestone can be found from the simulation data and, subsequently, the throughput (CTPS) can be calculated.

[1]https://github.com/pacslab/iota_simulation

4.1.2 Simulation Setup and Results

To collect the simulation data, we run a group of 10 simulations with 6000 transactions, 20 agents, $d=1, \alpha=0.001$ and λ varying from 1 to 10 with $step=1$, see Table 4.2. This is a baseline configuration which we use to explore the influence in comparison with other configurations. Then, we run 5 independent simulations by only changing λ varying from 10 to 30 with $step = 5$ and transactions from 3,000 to 9,000 with $step=1,500$ to explore higher rate scenarios. In total, over 90,000 transactions are simulated. In all simulations, λ_{M} is set to be 1/60, i.e. one *Milestone* is issued to the Tangle every minute, because this is the setting in the running IOTA mainnet as of August 2020; for each simulation, *transactions* is set to 6,000, so that at least 10 *Milestones* (namely 10 replicas) are ensured for each λ configuration. The simulations are conducted on a DELL PC with Windows 10 OS, 8th Generation Intel Core$^{\mathrm{TM}}$ i7-8700 12-Core Processor, and 16GB RAM.

Table 4.2: Default parameter configurations

Parameter	Simulation	Configurable	Experiment	Configurable
λ[a]	1∼10	✓	1∼10	✓
α[b]	0.001	✓	0.001	✓
TSA[c]	MCMC	✓	MCMC	✗
d[d]	1	✓	≈1 ms	✗
agents	20	✓	20	✓
transactions	6000	✓	NA	✗
COOTick	60 s	✓	60 s	✓

[a] 1∼10 are explored in both; 15, 20, 25, and 30 are explored in simulation.
[b] 0.001, 0.01 and 0.1 are explored in both simulation and experiment.
[c] weighted MCMC, unweighted MCMC and URTS are explored in simulation.
[d] 1, 5 and 10 are explored in simulation.

After simulation, the proposed IREA is used to extract the transaction confirmation data and conduct a statistical analysis of the data. The result provides an almost linear relationship between CTPS and λ, as shown in Figure 4.1a, in which all CTPS values are obtained by averaging over all confirmations of 10 replicas.

To examine the impact of different TSAs on CTPS, we conduct two more groups of simulations (10 simulations in each group) on the unweighted MCMC and URTS strategies, respectively. In the weighted MCMC random walk, to explore the influence of the randomness factor α on CTPS, we conduct 2 groups of simulations with α=0.01 and 0.1, respectively. In addition, different network delays are indicated as distances are examined, i.e., d=1, 5, and 10, respectively. For each group of simulations, the value of λ is varying from 1 to 10 with $step = 1$ (refer to Figure 4.1b, Figure 4.1c

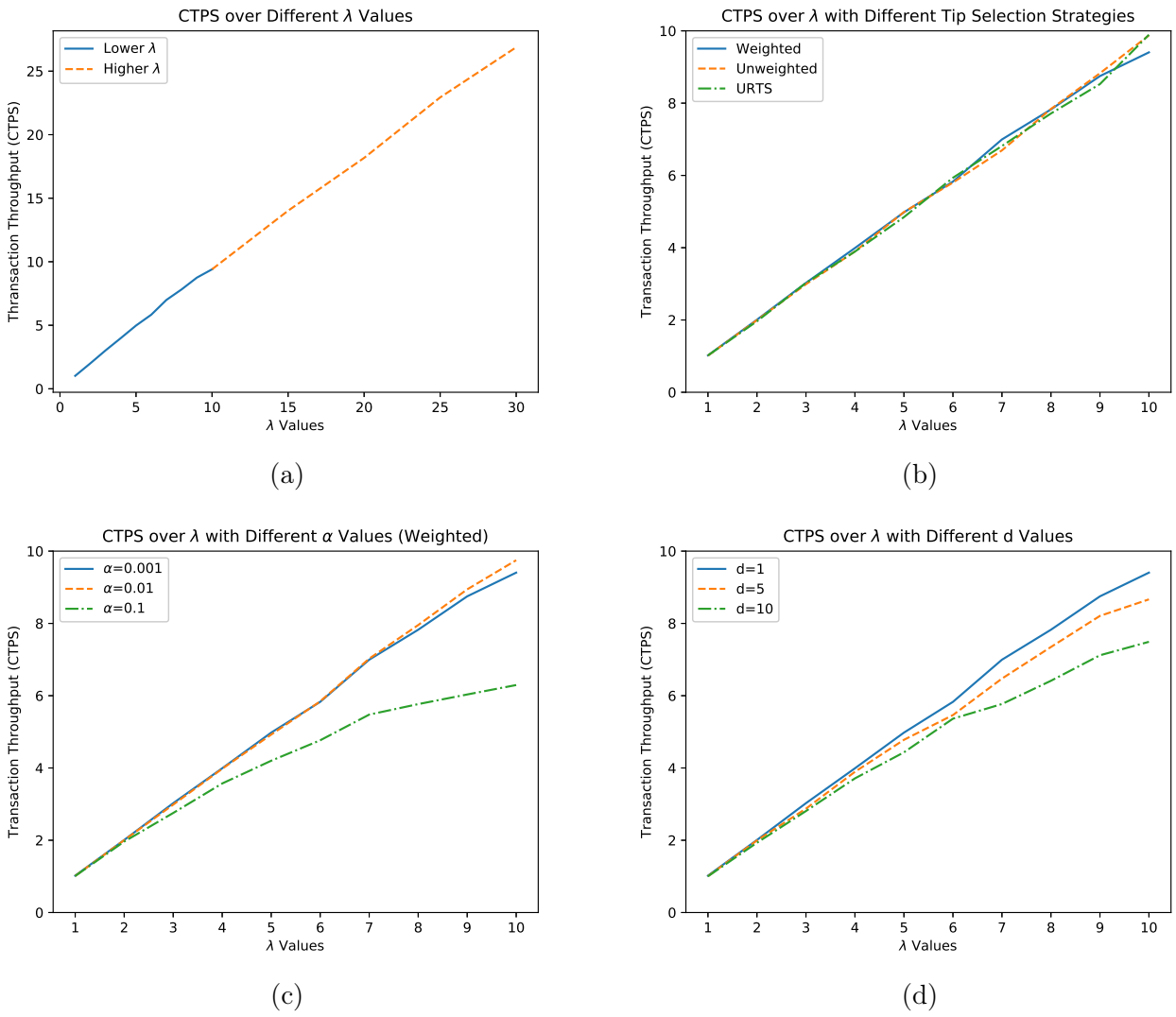


Figure 4.1: Simulation results: CTPS over λ under different influence factors (a) λ only (b) TSAs (c) randomness α (d) distance d.

and Figure 4.1d).

As can be seen in Figure 4.1b, there are almost no differences for CTPS under different TSAs, i.e., *weighted*, *unweighted* and $URTS$. This is because when α is set to the default value 0.001, there is sufficient randomness in the tip selection random walk. Nevertheless, the α values have an obvious impact on CTPS in a weighted random walk. According to the MCMC random walk, larger α will increase the probability to select heavier tips so that most new coming transactions will always be attached to the heaviest path. It shows that as α increases from 0.001 to 0.01, there is no big difference on CTPS; but when α raises to 0.1, both CTPS and its increase rate decrease dramatically. In a more extreme case, CTPS even keeps flat when α is set to a big value, e.g., 10 [67], which implies a linked chain rather than a DAG in IOTA.

However, the different distances which simulate the network delay of IOTA do not show much

difference in general as shown in Figure 4.1d, which means that network delays have a limited influence on the throughput under the examined situations. This can be explained as follows. On the one hand, compared to the time spent on PoW and transaction validation, network delay only takes a very small portion of the whole transaction life from being sent to getting confirmed. On the other hand, all examined d values remain quite low, which further weakens their influence on the performance.

4.1.3 Parameter Analysis

To discuss how λ influences IOTA's performance, we model the IOTA system as a single-server network with a first-come-first-served (FCFS) servicing policy. The service rate of this network refers to the maximum transaction processing capacity μ, which can be obtained from the load test in practice. In most simulation settings, the actual transaction throughput (denoted as X) keeps a near-linear growth against the transaction arrival rate λ, i.e., $X = k \cdot \lambda$. This trend continues until λ is larger than μ. As transactions are continuously issued and confirmed, the throughput should be the same as the arrival rate when $\lambda < \mu$ in this single-server network model. However, since there is a probability that an issued transaction will be eventually abandoned or orphaned under the random walk TSA, the actual transaction throughput becomes less than λ. In addition, more transactions get abandoned in high load (see Figure 4.1a). Therefore, λ dominates the growth of the transaction throughput when $\lambda < \mu$, while there is an obvious difference between λ and the actual transaction throughput due to the abandoned transactions. This difference is determined by the transaction attachment mechanism, namely TSA in the Tangle.

Different TSAs affect the growth of Tangle by impacting its properties, such as the number of tips, the transaction's cumulative weight, and the time until the first approval [68]. The simplest TSA is the URTS, which selects a tip from the set of all available tips in a uniform random manner. Another strategy is MCMC, which is a sampling approach based on a random walk starting from an entry point in the Tangle towards a tip. If the random walk is biased to a more weighted transaction in each step, we call it *weighted MCMC*; if there is no bias in random walks, we call it *unweighted MCMC*. Since the URTS and unweighted MCMC do not encourage validating more weighted or honest transactions to build a main DAG, they leave vulnerabilities for attackers to build a parasite chain where double spending attacks can be launched (see [68] for details). Therefore, they are not secure against *parasite chain attacks* [68], and should be only treated as tools to study the Tangle rather than applied in production. To address this problem, IOTA employs the weighted MCMC as its TSA to encourage new transactions to verify honest tips and to achieve consensus. The bias level is controlled by a randomness parameter α. Theoretically,

the greater the value of α, the greater the probability that the random walk will choose a more weighted tip in each step. Thus, the DAG is more secure by always expanding on the main chain. However, there are more abandoned honest transactions in this case, which will impact the transaction throughput.

We can observe that the Tangle performs almost the same on throughput under URTS and MCMC with $\alpha = 0, 0.01$ in Figure 4.1b, as well as the weighted MCMC with $\alpha = 0.001$ in Figure 4.1c. They all have sufficient randomness to ensure that every tip has a chance to be selected, leaving a smaller number of abandoned transactions in the Tangle. However, the throughput drops significantly when $\alpha = 0.1$ in Figure 4.1c, where more honest transactions are abandoned because of the biased tip selection strategy. Therefore, it is an interesting research topic to find the optimal α, or propose more TSAs (e.g., hybrid TSA [42] and first-order biased random walk [27]) to tackle the trade-off problem between security and efficiency of the Tangle. In addition, the time complexity of URTS is $\mathcal{O}(n)$, while both weighted and unweighted random walk TSAs have $\mathcal{O}(n^2)$ time complexity.

Even though the network delay has a very limited influence on throughput compared to on transaction latency, we can observe a drop in the throughput as the delay increases in Figure 4.1d. In the IOTA network, a node relies on TCP/IP to broadcast transactions and synchronize the local ledger with its neighbours. High network delay will cause an asynchronization problem among different local ledgers on the nodes. This asynchronization will further decrease the number of new tips from a delayed node's view. Since the weighted MCMC prefers new tips, more delayed transactions will get older and eventually abandoned as the network delay increases. Thus, the transaction throughput decreases.

4.1.4 Experimental Validation for Simulation

To validate the simulation, we conduct three groups of experiments on a medium-sized network with the α configurations of 0.001, 0.01, and 0.1. For each α, the total λ values vary from 1 to 10 with $step = 1$, with other parameters remaining default as shown in Table 4.2. We employ IOTA Implementation Reference[2] (IRI 1.6.1) with Docker to deploy a private network of 20 nodes on the SAVI OpenStack cloud platform[3]. Each node is a virtual machine with the flavour of medium size, see Table 4.3 for configuration details. The open-source Compass[4] provided by the IOTA Foundation is used as the COO to generate milestones and confirm transactions in the Tangle.

[2]https://hub.docker.com/r/iotaledger/iri
[3]https://www.savinetwork.ca
[4]https://github.com/iotaledger/compass

The COO is set to generate a milestone every 60 seconds, just as in the simulations.

Table 4.3: Experimental environment

Network Nodes	Number	CPU	RAM	Disk
IRI Node	20	2 VCPU	4GB	40GB HD
Client Node	1	4-Core i5	8GB	256GB SSD

To better control the transaction arrival rate λ and avoid the impact of PoW to λ, we bring all PoW to a PC client with the configurations shown in Table 4.3. We run all experiments with the transaction requests in a Poisson process, i.e., the transaction interarrival time follows an exponential distribution. The socket ZeroMQ[5] is employed to listen and receive transaction data. To simplify the problem, we only send zero-value transactions.

In total, 151,104 confirmed out of around 169,701 generated transactions are collected from the three groups of experiments. More details on the experimental and simulation results can be found in [35]. Then, we compare the experimental data with the previous simulation data on λ and α to examine the validity and accuracy of the simulation.

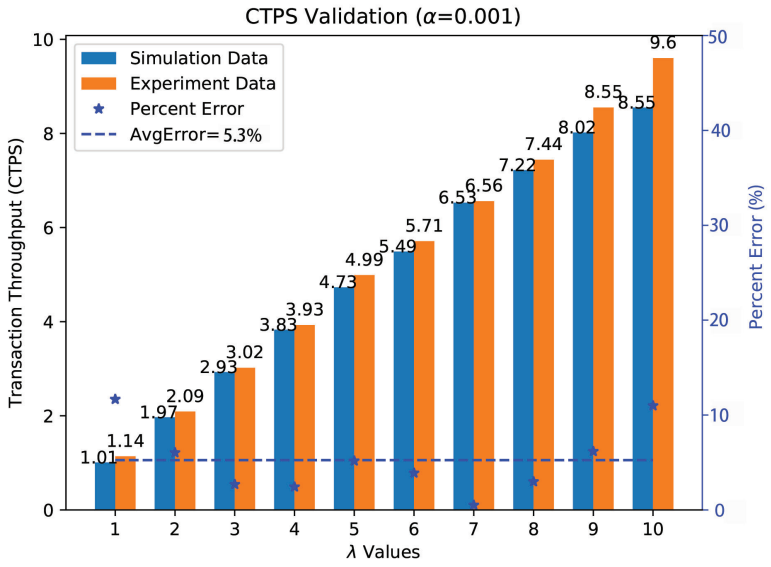


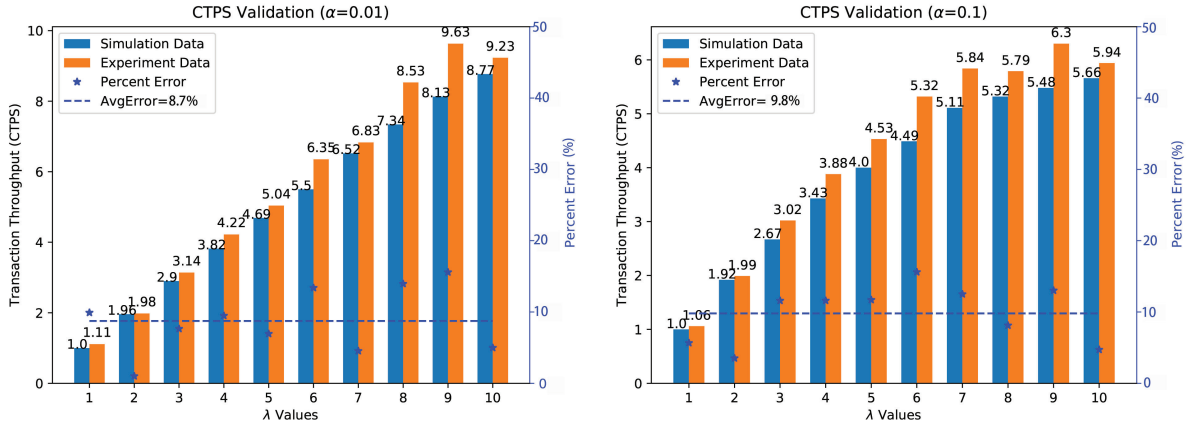
Figure 4.2: Experimental and simulation results comparison: λ only.

First, we compare the CTPS values of the simulation and experiment under varying λ 1∼10 by keeping other parameters as default (see Table 4.2) to calculate the percent error. As we can see in Figure 4.2, (1) both the simulation and experimental CTPS results increase almost linearly as λ values increase, implying good scalability; (2) they match well to each other with average percent error of 5.3%. Here, the percent error is calculated from the following equation,

[5]https://docs.iota.org/docs/client-libraries/0.1/how-to-guides/python/listen-for-transactions

$$Percent\ Error = \frac{|CTPS_{\text{exp}} - CTPS_{\text{simu}}|}{CTPS_{\text{exp}}} * 100\% \tag{4.1}$$

Second, we compare the simulation and experimental results in terms of different α values. As can be seen in Figure 4.3, simulation and experiment are well in tune for the two values of α. The average errors are 8.7% and 9.8%, respectively. Having average errors below 10%, we assume that the simulation data can effectively and accurately capture the Tangle behaviour in the real world.



(a) (b)

Figure 4.3: Experimental and simulation results comparison under different α values: (a) α=0.01 (b) α=0.1.

4.2 Analytical Layered Model

The previous simulation explored some CTPS influential factors, identified the most important one, and provided a quantitative relationship between CTPS and λ (i.e., linear relationship). However, it is difficult to describe more details such as how these confirmations are distributed in the Tangle, which keeps the second question about reattachment waiting time still unanswered. Therefore, we propose a layered model to explore the confirmation distribution in each single graph layer.

4.2.1 Model Description and Solution

In the IOTA Tangle, we define a layer as all confirmed transactions with the same depth from a Milestone in a hierarchical architecture, as shown in Figure 4.4. In the case of two different transactions referencing the same transaction with different depths, we take the minimum layer index as the layer depth for this transaction. For example, in Figure 4.4, transaction 5 holds references from both 1 and 4, which are from different layers, $Layer_1$ and $Layer_2$. In this case, we

assume that 5 is located in $Layer_2$ rather than $Layer_3$. With respect to this layering decomposition and using the previously mentioned simulation data set with over 90,000 transactions, we extract the transaction confirmation number in each layer of the DAG, as shown in Figure 4.4.

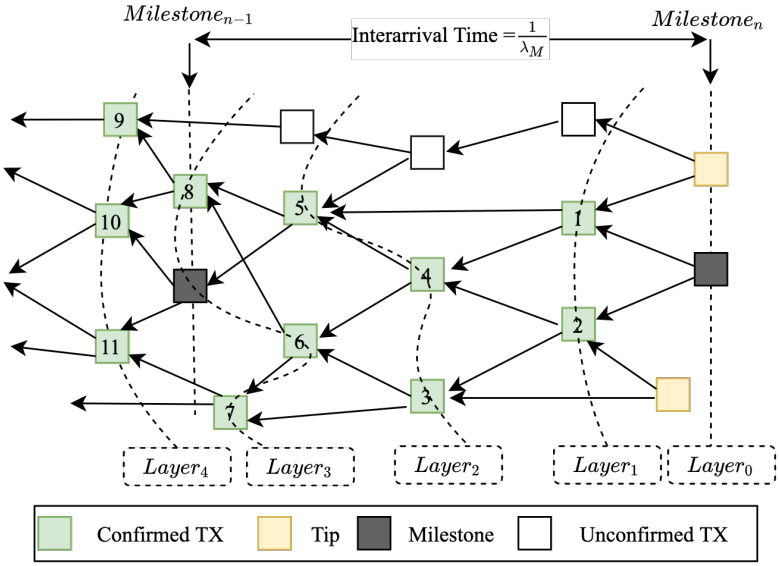


Figure 4.4: Layered model for transaction confirmations.

After plotting the confirmed transactions over the layer number for each λ, we observe a lot of bell-shaped curves, such as the blue dot "Data" curve shown in Figure 4.5, which points us to the nonlinear model fitting, e.g., the Gaussian model. Therefore, after taking the average confirmations of all milestones for each λ, we strive to fit our simulation data as a nonlinear model to characterize the relationship.

In total, for each λ we use 45 nonlinear models to fit our data in CurveExpert[6]. The results show that under all λ values except for λ=1, the Gaussian Model outperforms others and is always listed in top 3 models, as we can see from the example of λ=10 in Figure 4.5. In our fitting, we use the target data set under various λ values by taking the mean layered transactions of milestones 5, 6, 7, 8, and 9, by getting rid of the potential warming-up and cooling-down phases. The fitting results of the Gaussian Model are listed in Table 4.4.

By checking the values of *Correlation Coefficient*, we carefully claim that the mean confirmed transactions located at different layers can be fitted as a Gaussian Model. Therefore, we have the number of confirmed transactions to be a Gaussian function of layer x,

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{4.2}$$

Here, b has an almost linear increase trend as λ increases, while c almost remains the same from our simulation data in Table 4.4. This indicates that all examined Gaussian models have a

[6]https://www.curveexpert.net

Table 4.4: Gaussian model fitting results for different λ values

λ	1	2	3	4	5	6	7	8	9	10
Correlation Coefficient	0.934	0.988	0.987	0.982	0.992	0.997	0.984	0.996	0.990	0.991
Standard Error	0.864	0.909	1.274	2.271	1.911	1.476	3.469	2.242	3.733	4.146
a	7.598	16.561	22.439	32.087	40.232	49.378	51.804	66.691	71.021	81.009
b	7.944	8.360	8.976	9.172	9.726	9.390	10.169	9.745	10.298	9.722
c	4.246	3.517	3.742	3.656	3.296	3.298	3.859	3.296	3.536	3.283
AMUB*	13.600	15.000	15.600	16.800	16.400	16.200	17.600	17.000	17.400	16.600
CIUB+	16.265	15.252	16.310	16.337	16.187	15.854	17.733	16.205	17.229	16.156

*Actual Mean Upper Bound, +Confidence Interval Upper Bound

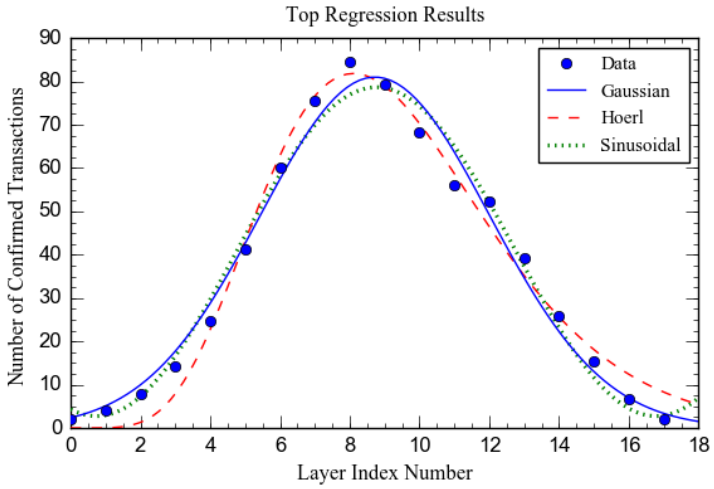


Figure 4.5: Simulation data and fitted models for λ=10.

very similar shape, and the next randomly confirmed transaction is expected to be located at a deeper layer in the Tangle with higher λ values. Our main finding is summarized as below.

Finding 1: *If a transaction remains unconfirmed for more than $2/\lambda_M$ seconds after being submitted, the probability that it gets confirmed afterward is sufficiently low.*

Proof. Let layer index be the discrete-time dimension, then the *Confidence Interval (CI)* of Gaussian models can be used to estimate the length of time to wait before reattaching transactions. First, let us look at the *Upper Bound* layers in our model. If we take a $confidencelevel$ of 95%, the *critical value (Z-value)* for this CI is 1.96, where $(1-0.95)/2 = 0.025$. In our case, this means that there is a very small probability (2.5%) for a confirmation to happen after a specific *Upper Bound* layer. The layer of *CIUB* shown in Table 4.4 is calculated by the following estimation

formula.

Given that

$$Z = \frac{X - b}{c} \tag{4.3}$$

we have

$$X_{\text{CIUB}} = Zc + b \tag{4.4}$$

Then, we translate the *Upper Bound* layer to the time dimension by analyzing the layered model. As we notice from Figure 4.5, the decrease happens just after a specific layer. From Figure 4.4, we know that when the confirmation layers of $Milestone_{\text{n}}$ cross the arrival time of $Milestone_{\text{n-1}}$, there will be a decrease in the number of transactions confirmed by $Milestone_{\text{n}}$ because of the overlap. For example, as shown in Figure 4.4, transactions 10 and 11 will not be counted as confirmations by $Milestone_{\text{n}}$ since they had already been confirmed by $Milestone_{\text{n-1}}$. Therefore, we empirically notice that the peak CTPS layer in the confirmation Gaussian Model refers to the previous $Milestone$ arrival time, which is $1/\lambda_M$ seconds ago from the latest one. Thus, the optimal waiting time before reattaching for users is estimated to be around $\mathbf{2}/\lambda_M$ seconds.

4.2.2 Model Validation

To validate the deductive results of our layered model, we use three groups of experimental data with different α values to conduct a statistical analysis. First, we determine all identical transactions by matching their hash values with the sent and confirmed records, respectively. Then, we leverage the $TimeStamp$ property to calculate the time difference from sending to getting confirmed for each transaction. Since some confirmed transactions recorded in a time period are generated from the previous time segment and cannot be matched within the corresponding sent records, the number of matched transactions is usually less than all actually confirmed transactions. For example, there are 40,023 sent, 39856 confirmed, and 39462 matched transactions when $\alpha = 0.001$.

Table 4.5: Statistics of confirmations over 2 minutes ($\mathbf{2}/\lambda_M$ seconds)

α	Conf txs	Conf txs(>2 mins)	Percentage
0.001	39,462	2,235	5.7%
0.01	39,355	2,523	6.4%
0.1	31,909	190	0.6%

In total, there are 110,726 confirmed transactions with confirmation waiting time (i.e., they are matched and have the sent-confirmed time difference). Within these transactions, we find that only 4,948 are confirmed after 2 minutes, see Table 4.5.

The detailed transaction confirmation distribution over time under different α values can be observed from Figure 4.6. As we know that λ_M is set to be 1/60, i.e., a milestone is issued every minute, so $\mathbf{2}/\lambda_M$ seconds are exactly 2 minutes in our experiments. Therefore, we have only 5.7%, 6.4% and 0.6% of the transactions confirmed after the time of $\mathbf{2}/\lambda_M$ for α=0.001, 0.01 and 0.1, respectively, which is relatively low and well matched with the prediction of our proposed layered model.

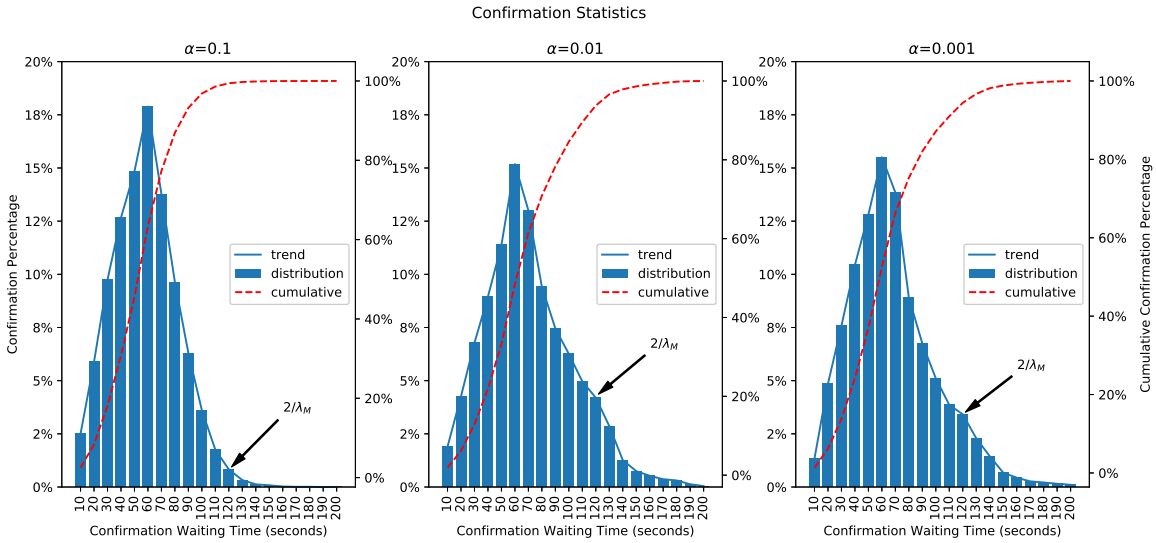


Figure 4.6: Experimental confirmation statistics in different time segments; the confirmation ratios are sufficiently low after the times of $2/\lambda_M$.

4.3 Summary

In this work, we have studied the performance of a private IOTA network through experiments, simulations, and a layered model. We leverage these approaches to answer two research questions on throughput and RWT, with a high level of confidence. In particular, we extended the DAGsim simulator and used it to empirically analyze the influence of transaction arrival rate λ, *TSAs*, randomness α in weighted random walk algorithm and network delay d on CTPS. Among all these impact factors, λ is the most important one, which has a near-linear relationship with the CTPS. Moreover, we leveraged the proposed analytical layered model to explore the confirmation distribution and found that the confirmations are normally distributed in DAG layers, which led to characterizing the Gaussian Model. Using this model, we estimated the RWT for a private IOTA network, which was validated by our experimental results. In conclusion, our extended DAG ledger simulator and the proposed layered model provide valuable insights into the performance of the IOTA distributed ledger in private network scenarios.

As for future work, on one hand, we would like to study how other factors such as encryption algorithms and ledger databases influence the throughput. It would be also interesting to evaluate IOTA's performance under consensus without COO in further studies. On the other hand, our extended simulator did not resolve the efficiency problem in large-scale simulations, which would be another direction of our future research.

5 Performance Analysis of Hyperledger Besu System

In this work, we present a set of comprehensive experimental studies on Hyperledger Besu in a private blockchain. We aim to exhibit its performance characteristics in terms of transaction throughput, latency, resource utilization, and scalability, from the application perspective by adding a load balancer middleware. We have carefully designed a set of comparative experiments and judiciously selected typical parameters, including transaction send rate, network size, node flavour, load balancing, consensus, and block time. In particular, three proof of authority consensus algorithms, Clique, IBFT 2.0, and QBFT, are investigated. Through extensive experimental evaluations using the Hyperledger Caliper benchmark tool, we analyze how these parameters impact the performance of a private Besu blockchain. Our studies reveal several interesting findings: 1) Blockchain parameters, e.g., block time and block size, are the most significant factors in determining Besu's performance; 2) The performance of Besu is bottlenecked by transaction execution and blockchain state updates, which are determined by node computation power, transaction complexity, and load balancing; 3) A Besu network with QBFT consensus can scale up to 14 validators without noticeable performance loss. Our findings shed some light on further performance improvement of Hyperledger Besu. The identified bottlenecks and root cause analysis provide insightful suggestions for blockchain practitioners to build performant enterprise applications.

5.1 Introduction

Beyond cryptocurrency, blockchain technology has been adopted by many enterprise applications, such as supply chain[20], [116], [131], healthcare[71], [123], and peer-to-peer transactions[32], [115], [126], to overcome security and privacy issues in traditional centralized solutions and build trust between participants by removing the trusted third party. Decentralization, immutability, and enhanced security make blockchain a promising platform for enterprises to store and share data among participants in a network. From the perspective of network accessibility, blockchain can be divided into two main categories: public and private. Unlike a public blockchain, where the

participant does not need permission to join the network, a private blockchain usually requires permission and assumes a certain degree of trust among entities. Due to the benefits of verified participants, enterprises can build decentralized applications on a private blockchain with an efficient consensus to achieve immediate finality, high performance, and improved privacy.

Recently, Ethereum has become one of the mainstream blockchain platforms for developing enterprise applications. An Ethereum network consists of nodes, called clients, running software that executes transactions, verifies, and creates blocks. All nodes in the network maintain a state machine, namely the Ethereum Virtual Machine (EVM), which is responsible for executing transactions and smart contracts to compute the network's state from block to block. A smart contract is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain and automatically enforces the pre-defined rules to execute as programmed.

There exist many Ethereum clients implemented in various programming languages, such as Go, JavaScript, Python, and Java, by following a formal specification. Although enterprises can build a private blockchain by initiating a genesis file using regular clients, such as Geth, Nethermind, and OpenEthereum, this network can hardly meet specific enterprise needs. To address this problem, Enterprise Ethereum emerges to extend public Ethereum by providing enterprise specifications, including the capability to enforce membership (permissioning), provide high performance, and perform private transactions, which allow only participants of those transactions to access the metadata or payload. As one of the most popular Enterprise Ethereum clients, Hyperledger Besu has recently gained much attention in constructing enterprise decentralized applications [1], [105] due to its versatility, improved privacy, and high performance.

However, the performance and scalability of Besu, e.g., the impact of system configurations and chain parameters on the performance metrics of throughput and latency, has not been thoroughly studied. In this paper, we propose a load balance-based performance evaluation architecture, as shown in Figure 5.1, and analyze the performance of Hyperledger Besu from the application perspective in private blockchains, where proof of authority (PoA) consensus is preferred over proof of work (PoW) and proof of stake (PoS). In particular, we analyze three PoA consensus algorithms supported by Besu, namely Clique, Istanbul Byzantine Fault Tolerance (IBFT) 2.0, and QBFT, through extensive benchmark experiments and log analysis. The main contributions of this work are as follows:

- We present an in-depth experimental evaluation of Hyperledger Besu for its PoA consensus algorithms, showing its performance characteristics under various configurations.
- Through extensive benchmarks, we identify critical parameters that impact the performance

and scalability of Besu and analyze to what extent the influence is.

- We analyze the root causes of the identified bottlenecks through log analysis.

The remainder of this study is organized as follows: In Section 5.2, we elaborate on the basics of Hyperledger Besu. In Section 5.3, we present the experimental evaluation of the proposed benchmark framework. Section 5.4 outlines the experimental results achieved. In Section 5.5, we analyze the root causes of how the identified parameters impact the performance of Besu. In Section 5.6, we review the latest related work on private blockchain performance analysis. Section 5.7 concludes the paper and discusses potential future works.

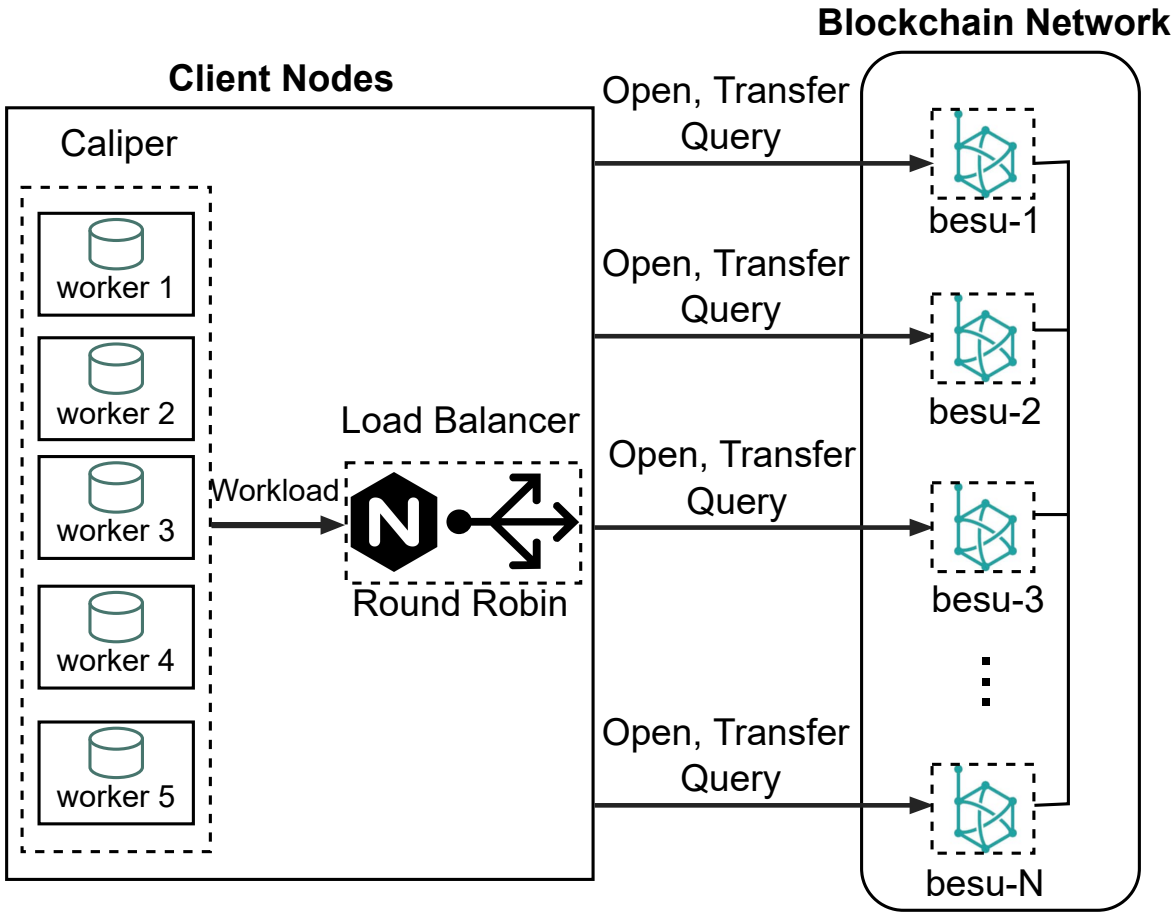


Figure 5.1: Hyperledger Besu performance evaluation architecture.

5.2 Hyperledger Besu Overview

Hyperledger Besu is an Ethereum client implementing the Enterprise Ethereum specifications, which are maintained by the Enterprise Ethereum Alliance (EEA), a membership of blockchain and incumbent businesses from around the world. It was initially known as Pegasys Pantheon,

then joined Hyperledger Foundation and was renamed Hyperledger Besu in 2019. Besu becomes popular because of its versatility. Its latest version (v22.4, May 2022) supports three types of consensus protocols: PoW (Ethash), PoS, and PoA (Clique, IBFT 2.0, QBFT). A summary of all consensus protocols that Besu supports is listed in Table 5.1. We will focus on proof of authority consensus protocols used in private networks.

Table 5.1: A comparison of Besu's consensus protocols.

Property	**Clique**	**IBFT2**	**QBFT**	**Ethash**	**PoS**
Type	PoA	PoA	PoA	PoW	PoS
Finality	No	Yes	Yes	No	No
Liveness	1/2	1/3	1/3	1/2	1/2
Throughput	High	High	High	Low	Medium
Quorum	1/2	2/3	2/3	1/2	1/2
BFT	No	Yes	Yes	Yes	Yes
Usecases	Private, Rinkeby, Goerli	Private	Private	Public	Public

5.2.1 Clique

Clique is a PoA consensus protocol presented in the Ethereum Improvement Proposal (EIP) 225 [99]. In this protocol, only a fixed set of authorized nodes, called signers, can collect and execute the transactions from the transaction pool into a block and update the world state. It maintains a signer list through a voting mechanism, in which existing signers propose and vote to add or remove signers. The process of creating or mining a block is called *sealing*. Signers create the next block in a round-robin fashion at a fixed interval determined by *blockperiodseconds*. The order will be determined by the block number and the signer count. In order to prevent a malicious signer from sealing fraudulent transactions, a signer is only allowed to seal a block every $\lfloor N/2 \rfloor + 1$ blocks. Therefore, at any point in time, there are only $N - \lfloor (N/2) + 1 \rfloor$ signers who are allowed to seal a block. In Clique, forks occasionally happen due to concurrent blocks. In addition, if a designated signer cannot seal a block in time, any other signers may propose a block after waiting for the block period, which is called out-of-order sealing. Out-of-order sealing can happen quite frequently in case of short block period times and significant network delays between nodes, which will negatively impact the network performance.

Therefore, Clique does not have immediate finality. Implementations using Clique must be aware of forks and chain reorganizations occurring. Besu implements Clique, which can be used

in a private network or test networks such as Rinkeby and Goerli. But, it is not recommended for production. Other Ethereum clients implementing Clique consensus include Geth and Quorum.

5.2.2 IBFT 2.0

Istanbul Byzantine Fault Tolerance (IBFT) is a variant of PBFT [21] suitable for blockchain networks. It was first proposed informally in EIP 650 [8] and was implemented in Geth by AMIS Technologies in 2017, and soon in Quorum. As a leader-based (or voting-based) consensus, IBFT is deterministic and can tolerate f faulty participants out of n, where $n \leqslant 3f+1$. It inherits from the original PBFT by achieving consensus in three phases, *pre-prepare*, *prepare* and *commit*, and has $O(n^2)$ total communication complexity. Before each round, the selected proposer will propose a new block proposal and broadcast it along with the *pre-prepare* message. Upon receiving the *pre-prepare* message, other validators enter the state of *pre-prepared* and then broadcast *prepare* message. When receiving $2f+1$ *prepare* messages, the validator enters *prepared* state and then broadcasts *commit* message. Finally, validators wait for $2f+1$ *commit* messages to enter the *committed* state and then attach the proposed block to the chain.

However, Saltini and Hyland-Wood [110] from ConsenSys analyzed the correctness of the initial version of IBFT and pointed out that it does not guarantee Byzantine-fault-tolerant consistency and liveness when operating in an eventually synchronous network. They further resolved the liveness and finality issues in IBFT 1.0 by proposing a new version, called IBFT 2.0 [111]. Specifically, they modified the number of nodes for reaching consensus from $2f+1$ to super-majority and also improved the round change by removing the block-locking mechanism. Besu quickly deprecated IBFT 1.0 and replaced it with IBFT 2.0 after the new version was developed in 2020. So, unless otherwise specified, IBFT or IBFT2 refers to IBFT 2.0 throughout this paper.

5.2.3 QBFT

To further resolve the safety and liveness issues [61] in IBFT where the whole system can get stuck because of two honest nodes locking on different blocks, Quarum blockchain proposed and developed a variant of IBFT, called QBFT [89]. Like IBFT and PBFT, QBFT employs the famous three-phase commit scheme: *pre-prepare*, *prepare* and *commit*. For each round, a block proposal is created and broadcast with a *pre-prepare* message to other validators. Other validators broadcast a *prepare* message upon receiving the *pre-prepare* message. Then, on receiving the *prepare* message, a validator sends a *commit* message. Finally, a new block is inserted into the chain by the proposer after $2N/3$ *commit* messages. If a consensus is not achieved among all validators before the predefined time, a round change will happen with all clock time being reset. To ensure safety across

round changes, QBFT employs a justification mechanism of proposed values that is critical in achieving quadratic communication complexity [89]. Besu fully implemented QBFT in the release of v22.1.1 and recommended it as the enterprise-grade consensus for production in place of IBFT 2.0.

5.3 Experimental Evaluation

In this section, we introduce our evaluation methodology, experimental setup, and configurations. We extensively tested varying configurations, including network size, node flavour, load balancing, consensus algorithm, and block period seconds. In particular, we used the control variate method to explore the impact of each parameter on the performance of Besu. All tested parameters are summarized in Table 5.2.

Table 5.2: Tested parameters in the experiments.

Test Parameters	**Varying**	**Fixed**
Send rate	100 → 10,000 with step 100	8-LB-2C7.5G-QBFT-1S
Network size(N)	4 → 36 with step 2	⋆-LB-2C7.5G-QBFT-1S
Load balance(LB)	LB, NLB	8-⋆-2C7.5G-QBFT-1S
Node flavor(F)	1C7.5G, 2C7.5G, 2C15G, 4C15G, 4C30G, 8C30G, 16C60G	8-LB-⋆-QBFT-1S
Consensus(C)	Clique, IBFT2, QBFT	8-LB-2C7.5G-⋆-1S
Block time(BT)	1S, 2S, 3S, 4S, 5S	8-LB-2C7.5G-QBFT-⋆

Here, we organize our configurations into the formation of "N-LB-F-C-BT". If a parameter is missing, it indicates that this parameter is under test with varying values. By default, the transaction send rate (i.e., the designated total number of transaction requests sent from all workers per second, measured in req/s) varies from 100 to 1,000 with a step of 100, which is configured by Caliper [98] at each test. According to the minimum system requirements (a VM with 6GB RAM and 20GB HDD) for a private Besu network, we define the configurations in the first line in Table 5.2 as the baseline, which has 8 nodes, each with 2vCPUs, 7.5GB RAM, and 36GB SSD. Other flavours have at least 36GB of SSD storage. All artifacts (i.e., source codes, configuration files, logs, and experimental results) of this article are publicly available in our reproducible repository[1].

[1] `https://github.com/pacslab/besu`

5.3.1 Experimental Setup

We performed our experiments on an OpenStack-based cloud with varying configurations shown in Table 5.2. For each configuration, for example, "8-LB-2C7.5G-QBFT-1S", we deployed a private Hyperledger Besu network with the latest version v22.4 at the time of writing on an OpenStack-based cloud. Each Besu node runs in a Docker (20.10.14) container on a virtual machine with Ubuntu 20.04 OS installed. The cross-node communication uses host networking, which has a bandwidth of 1,000Mb/s and a network delay of less than 1 ms. We also ran a network time protocol (NTP) service on each node to synchronize its time with a configured server, ensuring the time offset between any two nodes is less than 1 ms. Besu DEBUG logs were collected through Docker logging for further analysis.

To avoid the limitation of gas consumption in smart contract deployment and transaction execution, we configured all Besu blockchain networks to be gas-free by setting "gasLimit" to the maximum "0x1fffffffffffff", "contractSizeLimit" to the maximum 2147483647, and the node startup option "min-gas-price" to 0. Also, we set the difficulty to the minimum "0x1" and the transaction pool queue size to the default 4096. For simplicity, all nodes were configured as validators (IBFT and QBFT) or signers (Clique).

Then, we used the benchmark tool Hyperledger Caliper v0.4.2 with Ethereum SDK v1.4 to connect and test the deployed Besu blockchain networks. Three types of performance metrics are collected: transaction/query throughput, transaction/query latency, and node resource utilization (CPU and memory). In each test, Caliper was running in a container on a client VM with rich resources (e.g., 16vCPU and 60GB RAM). A Docker monitor was configured to collect the resource utilization of all the remote Besu node containers. The results were stored in performance reports.

5.3.2 Workloads

To comprehensively test the key parameters listed in Table 5.2, we leverage the popular simple contract [18] with three types of transactions (i.e., open, query and transfer) to generate workloads, as shown in the code snippets listed below. Open transaction performs one single write operation to the blockchain by assigning a value to an account through mapping; query performs a single read operation; transfer performs two write operations by adding to the receiver account and subtracting from the sender account. For each type of transaction, we set up the workload to 1,000 transactions, which will be sent at a fixed rate in a single test. The transaction send rate changes from 100 to 1,000 in 10 tests for all configurations except for the baseline heavy load test, where the transaction rate ranges from 100 to 10,000. From Table 5.2, we have 30 different

configurations, and for a single configuration under a specific transaction send rate, we run 25 replicas to generate 25 performance reports. Therefore, over 22,500,000 transactions have been generated and sent to the deployed Besu networks in our experiments.

```
contract simple {
    mapping(string => int) private accounts;

    function open(string memory acc_id, int amount) public {
        accounts[acc_id] = amount;
    }

    function query(string memory acc_id) public view returns
        (int amount) {
        amount = accounts[acc_id];
    }

    function transfer(string memory acc_from, string memory
        acc_to, int amount) public {
        accounts[acc_from] -= amount;
        accounts[acc_to] += amount;
    }
}
```

5.3.3 Performance Metrics

In this section, we introduce four performance metrics, i.e., throughput, latency, resource consumption, and scalability, which are evaluated in this paper. The first three are the basic metrics supported by Caliper, while scalability is an extended metric indicated by the basic metrics in varying sizes of Besu networks. Here, we define all the evaluated performance metrics as follows:

- **Throughput** is the number of successfully committed transactions or query operations per second. The transaction throughput is expressed as transactions per second (TPS) at a network size. Query throughput is expressed as transactions per second (TPS) from a single node.

- **Latency** is the amount of time taken for a transaction's effect to be usable across the network. The measurement includes the propagation time and the consensus time. Query latency is the time between when the read request is submitted and when the reply is received.

- **Resource consumption** is the CPU and memory (RAM) utilization in each node container.

- **Scalability** is the ability to support increasing network participants or computation resources of nodes. This measurement is indicated by the throughput and latency when the network or node size increases.

5.3.4 Load Balancer

The load balancer is a crucial component for distributed systems deployed in the production environment. In our setting, we add a middleware layer between Caliper and the Besu network, where an NGINX-based load balancer leverages the round-robin algorithm to distribute all incoming Remote Procedure Call (RPC) transaction requests evenly across the Besu nodes. The load balancer can optimize the resource utilization of upstream nodes, pass requests to other nodes if a node fails or has high tail latency, limit the number of connections under heavy load, and place the request in a queue for further processing. Therefore, the load balancer can generally increase the throughput, reduce the latency, and improve the availability and fault tolerance of the Besu network. In addition, it makes the performance metrics derived from our experiments more aligned with those in the production environment.

5.4 Experimental Results

In this section, we will go through our experimental results and discuss them in detail.

5.4.1 Baseline

To explore when the network with baseline configuration becomes saturated, we tested Besu by varying the transaction send rate from 100 req/s up to 10,000 req/s. Figure 5.2 shows the performance of Besu under heavy load. We observe that the client could generate up to 1,485 open, 1,570 transfer, and 3,028 query transactions per second, respectively. But, the Besu network could only deal with up to 442 open, 414 transfer, and 3,007 query transactions per second, respectively. The query throughput is limited by the actual transaction arrival rate, which is upper-bounded by the maximum query request rate that the Caliper client node could generate. When handling open and transfer transactions, the network was saturated at the 1,000 req/s send rate. So, we select 1,000 req/s as the representative send rate in the following experiments.

Figure 5.3 shows the detailed performance of Besu with the baseline configuration under varying send rates from 100 to 1,000. We can see that the average throughput of open transactions keeps increasing to over 400 TPS with an average latency of fewer than 1.5 seconds when the send rate reaches 1,000. The performance of open transactions consistently outperforms that of transfer

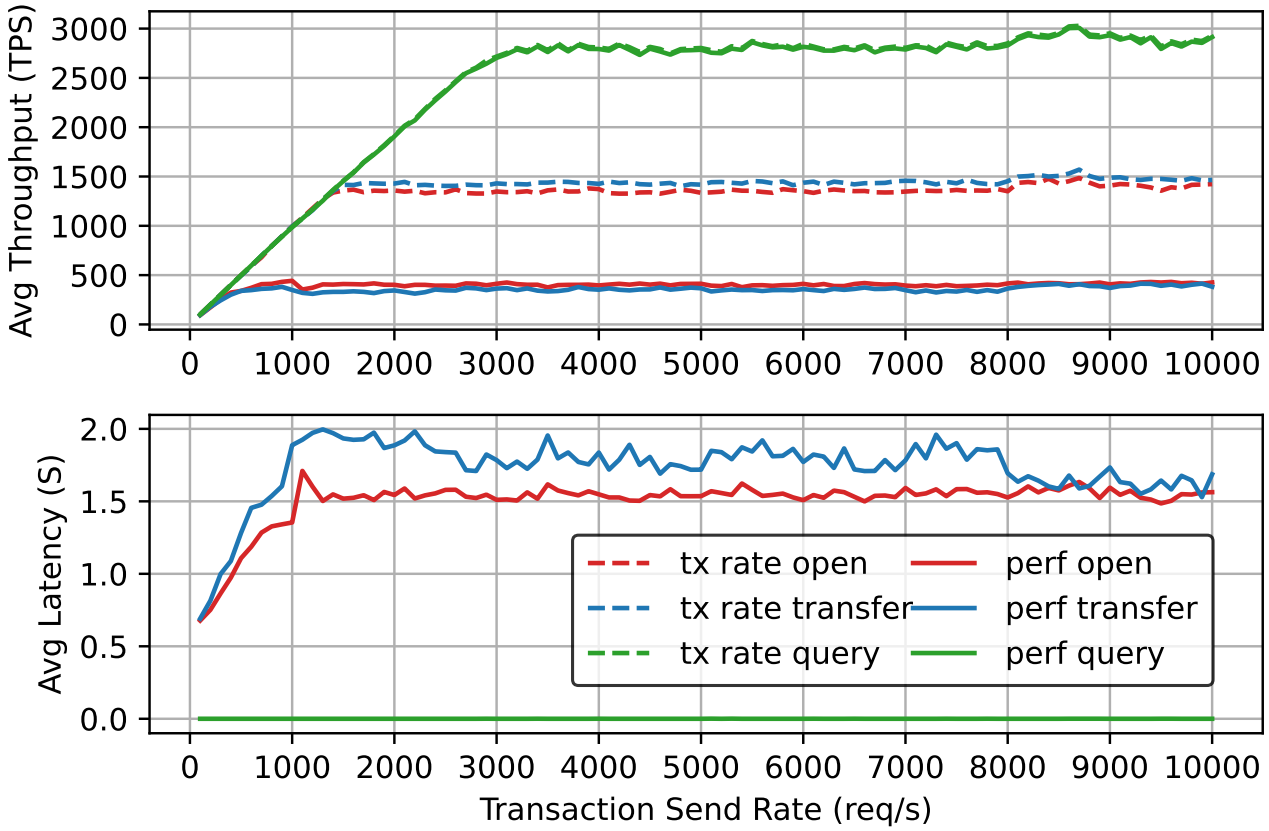


Figure 5.2: Besu performance under heavy load with the baseline configurations. Solid lines indicate the average throughput and latency, while dashed lines represent the actual number of requests per second sent to the Besu network.

transactions, especially under a heavier workload of more than 500 req/s, where the average throughput is around 10% higher with latency around 10% lower. This subtle difference could be caused by the different complexities of transaction operations. The result also indicates that the query operation is very efficient in Besu, reflected by its close-to-zero latency and linearly increasing throughput.

For the performance trend, the open and transfer throughput increases in a near-linear manner under a low send rate (e.g., from 100 req/s to 500 req/s), where a very small variation can be seen from the box plot, which implies a stable performance. However, the average throughput increases very slowly or keeps unchanged after the Besu network is saturated at a high send rate (e.g., from 600 req/s to 1,000 req/s), where the result variation becomes larger than light workloads.

5.4.2 Load Balancing

To explore the effect of load balancing on the Besu network performance, we tested a Besu network with the same configurations as the baseline, except for the load balancing (LB) setup

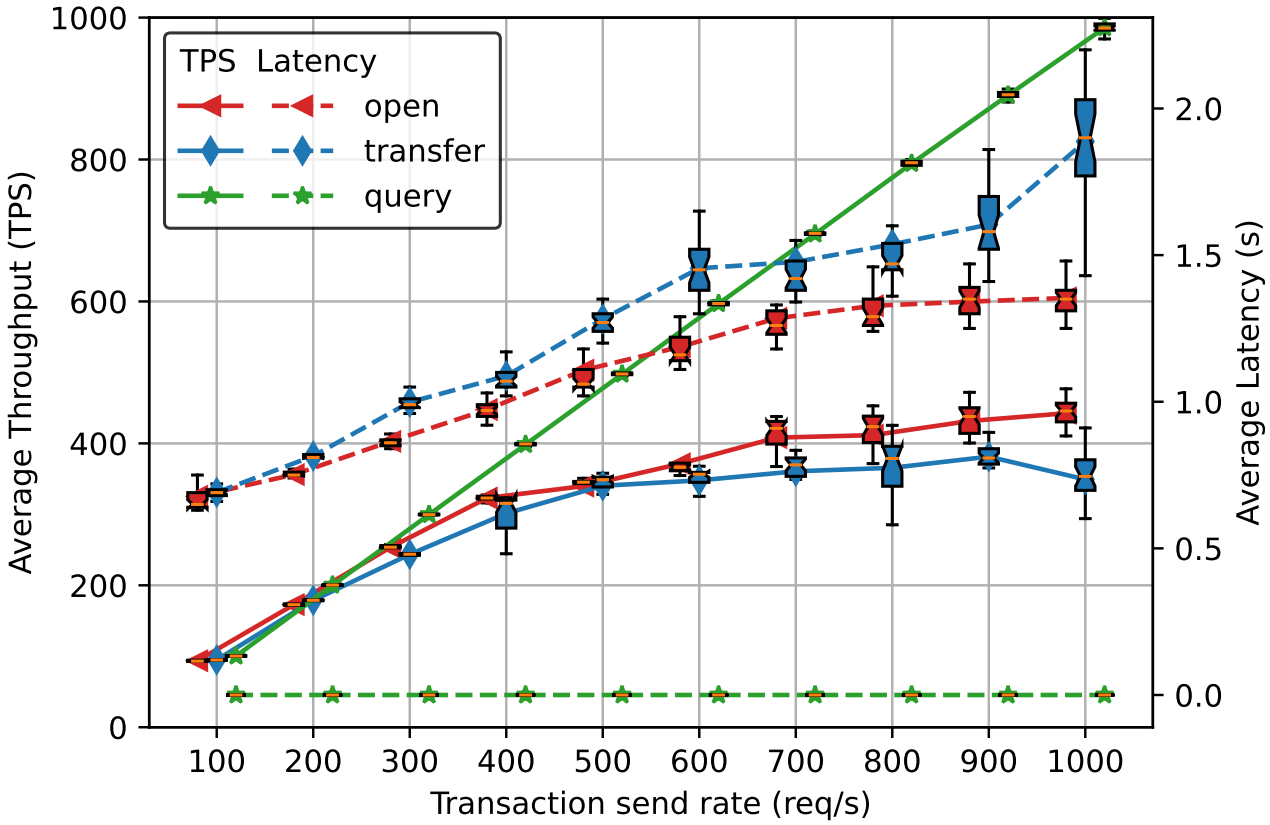


Figure 5.3: Besu performance under the baseline configurations. Solid lines indicate throughput; dashed lines represent latency.

Table 5.3: Besu node resource utilization statistics and comparison between LB and NLB modes at 1,000 req/s send rate.

	Statistics	besu-1	besu-2	besu-3	besu-4	besu-5	besu-6	besu-7	besu-8
NLB	CPU%(max)	**162.05**	5.57	4.49	3.52	5.43	4.25	3.50	4.00
	CPU%(avg)	**58.24**	4.16	3.24	2.78	4.23	3.39	2.76	3.21
	RAM(max GB)	**2.39**	0.88	0.86	0.93	0.84	0.86	0.86	0.84
	RAM(avg GB)	**2.39**	0.88	0.86	0.93	0.84	0.86	0.86	0.84
LB	CPU%(max)	4.91	5.76	6.23	6.24	6.52	7.37	8.21	5.81
	CPU%(avg)	3.62	3.92	4.29	4.31	4.63	5.39	5.33	4.04
	RAM(max GB)	0.94	0.91	0.98	0.92	1.00	0.96	1.02	0.90
	RAM(avg GB)	0.93	0.90	0.96	0.91	0.99	0.95	1.01	0.90

(Section 5.3.4). Instead of connecting to the load balancer, we directly connected Caliper to a Besu node, which is responsible for both receiving RPC requests and processing consensus tasks. From Figure 5.4, we can see that the transaction latency of open and transfer at a high request rate is unstable, reflected by large variations in the box plots. By comparing Figure 5.4 with Figure 5.3,

it is obvious that Besu's performance with LB is much better than that without load balancing (NLB). The throughput of open and transfer transactions with LB (i.e., 400TPS) is around twice as high as NLB (i.e., 200TPS). Accordingly, the latency of open/transfer transactions with LB is around half of NLB's. But, the query performance is not impacted by the setting of a load balancer. This is because query operation only reads from one Besu node's ledger and 1,000 req/s is still far away from its maximum RPC capability. In contrast, write (open/transfer) operations involve block creation, consensus message process and transit, transaction validation, etc., which require much more computation power and memory than query. So, a single validator as the RPC node could be easily saturated by an overwhelming number of transaction requests.

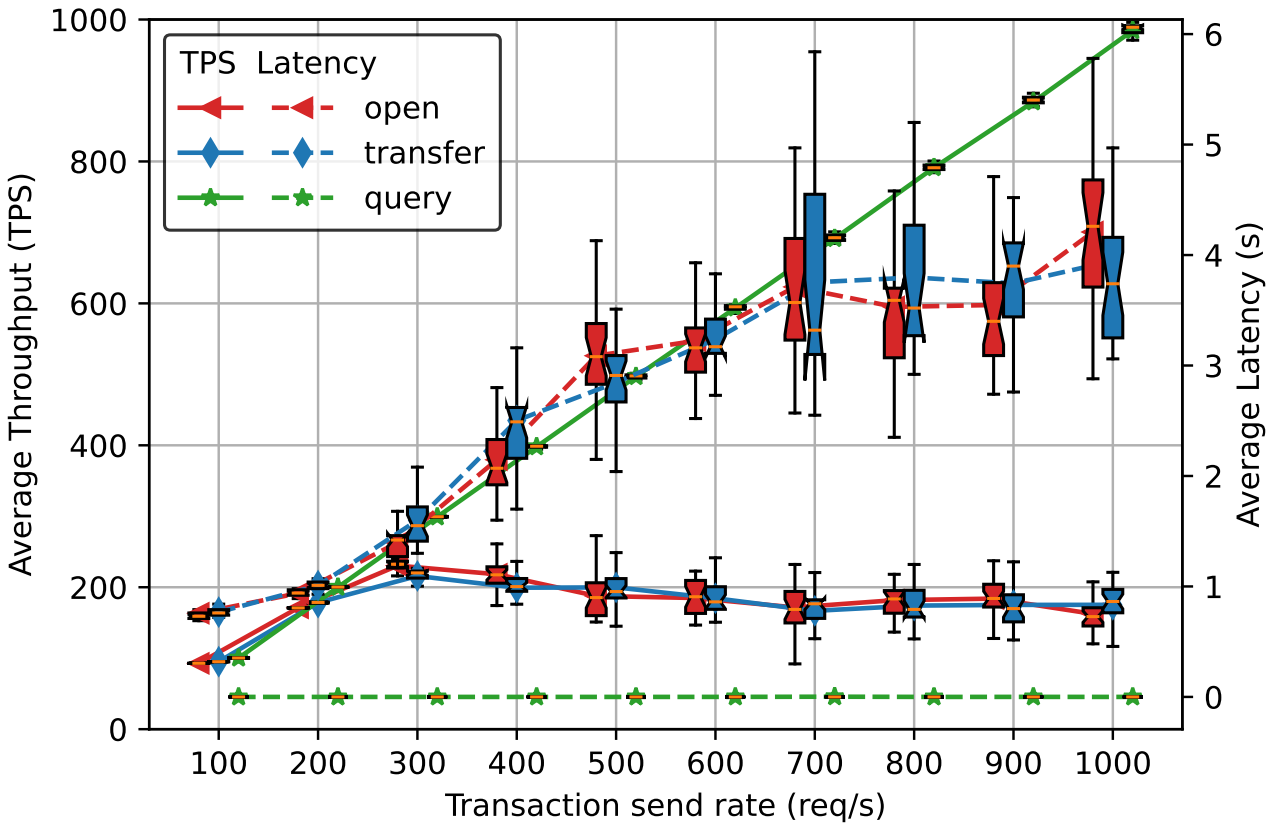


Figure 5.4: Besu performance under the baseline configuration without load balancing. Solid lines indicate throughput; dashed lines represent latency.

As we can see in Table 5.3, in the NLB mode, *besu-1* is obviously saturated in terms of CPU and memory utilization at 1,000 req/s send rate, while other nodes have relatively low utilization. The unbalanced load indicates the RPC node resource as a bottleneck of decentralized applications built on top of a Besu network. To make full use of all resources in the network, we add a load balancer to redirect the WebSocket RPC requests from Caliper to all Besu nodes in a round-robin manner. As seen in Table 5.3, the load is balanced among nodes in the LB mode, and the

performance of Besu increases as expected by comparing Figure 5.3 with Figure 5.4.

5.4.3 Scalability

We explore the scalability of the Besu network from two perspectives, scale-out (or horizontal scalability) and scale-up (or vertical scalability). Scale-out is evaluated by varying the network size in a fixed node flavour, i.e., 2C-7.5GB, while scale-up is evaluated by changing the node flavour in a fixed network size (i.e., 8 nodes).

Figure 5.5 and Figure 5.6 show the scalability of different consensus algorithms at 1,000 req/s open transaction rate. We observe that QBFT performs slightly better than the other two consensus algorithms in both horizontal and vertical scalability.

Figure 5.7 shows the detailed horizontal scalability of QBFT. The average performance is getting worse as the network size increases in general. But, it is interesting to see that the open throughput has a small improvement when the network size increases from 4 to 8 before dropping back to less than 400 TPS. Accordingly, the open latency has a subtle decrease at 8 nodes configuration. This is attributed to the increased network size, which distributes the transaction load to more nodes and reduces the queue time on each receiving node. However, as the network size increases to a certain extent that the communication complexity dominates the processing delay and becomes the new bottleneck, the performance gradually drops. We can also observe that open transaction has a consistently better performance than transfer.

Figure 5.8 shows the detailed vertical scalability of QBFT. We observe that Besu achieves better performance as the network nodes are equipped with more CPU and/or memory resources. The average throughput increases rapidly from 1C7.5G to 4C15G and then increases slowly as more resources are added to nodes. This is because after reaching a big enough node configuration, the hardware limitation is no longer the main performance bottleneck. Hence, keeping adding resources will have very limited help in the performance improvement of the Besu network.

5.4.4 Block Time

In Besu networks with proof of authority consensus algorithms, block time ("blockperiodseconds" in configuration) determines the frequency of block creation. The timer starts when the protocol receives a new chain head, and when it expires, the protocol proposes a new block. Figure 5.9 shows how the block time influences the performance of Besu with QBFT consensus. We observe a significant performance drop when the block time increases from 1 to 5 seconds. This is expected since QBFT spends more time waiting for the next block proposal when the configured block time is longer.

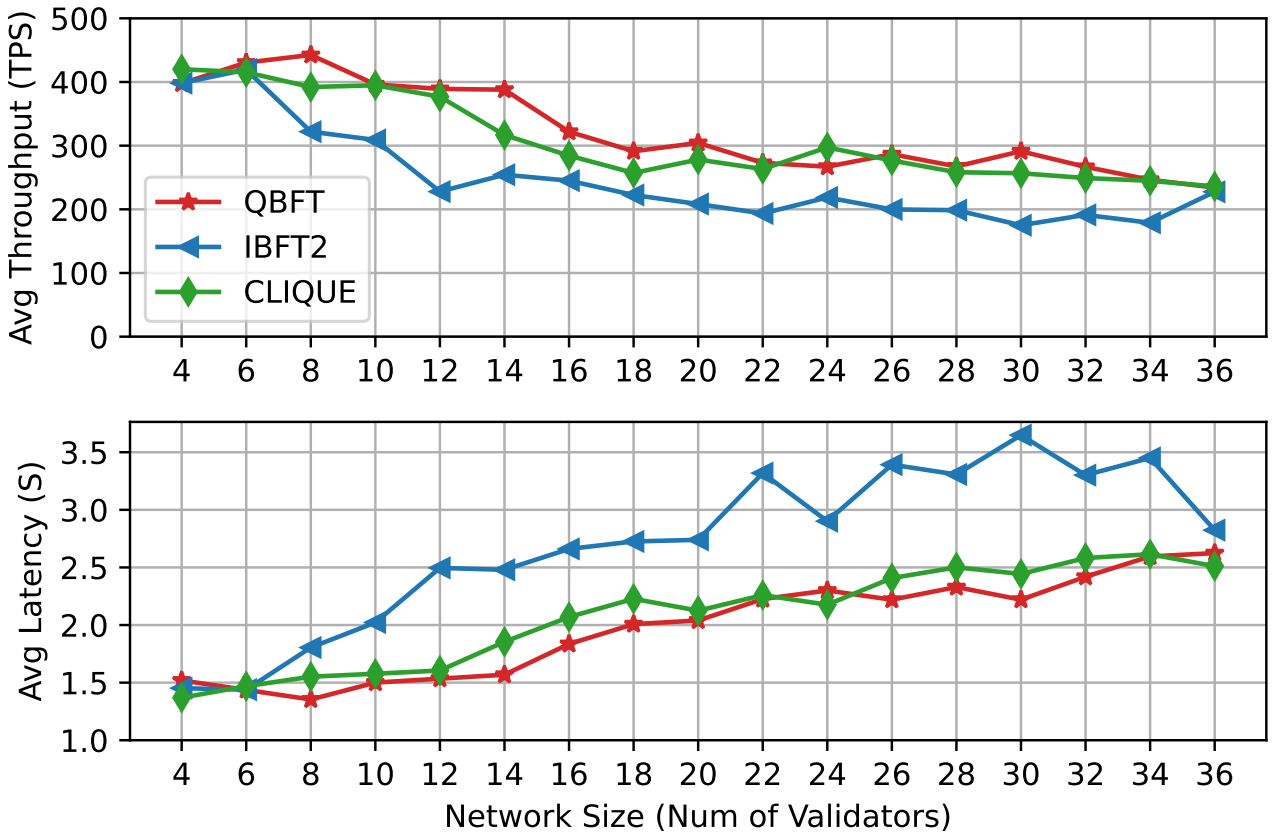


Figure 5.5: Horizontal scalability against different PoA consensus algorithms at 1,000 req/s open transaction send rate.

5.4.5 Consensus

Figure 5.10 shows the transaction performance of Besu against three proof of authority algorithms, QBFT, IBFT2, and Clique, under varying send rates. We observe that these three consensus algorithms have very close throughput under low workloads, e.g., 400 req/s. Then, QBFT has subtly better performance under high send rates, and Clique has a slightly lower throughput than the other two. However, Clique has a relatively lower transfer latency at the same time. Also, the open transaction has a better performance in terms of both throughput and latency in all three consensus algorithms.

5.5 Analysis and Discussions

In previous sections, we presented the performance benchmark results of the deployed private Hyperledger Besu networks under various configurations. From these results, we have identified some critical parameters which impact the blockchain performance. In this section, we aim to understand and present how these parameters impact Besu performance through log analysis and

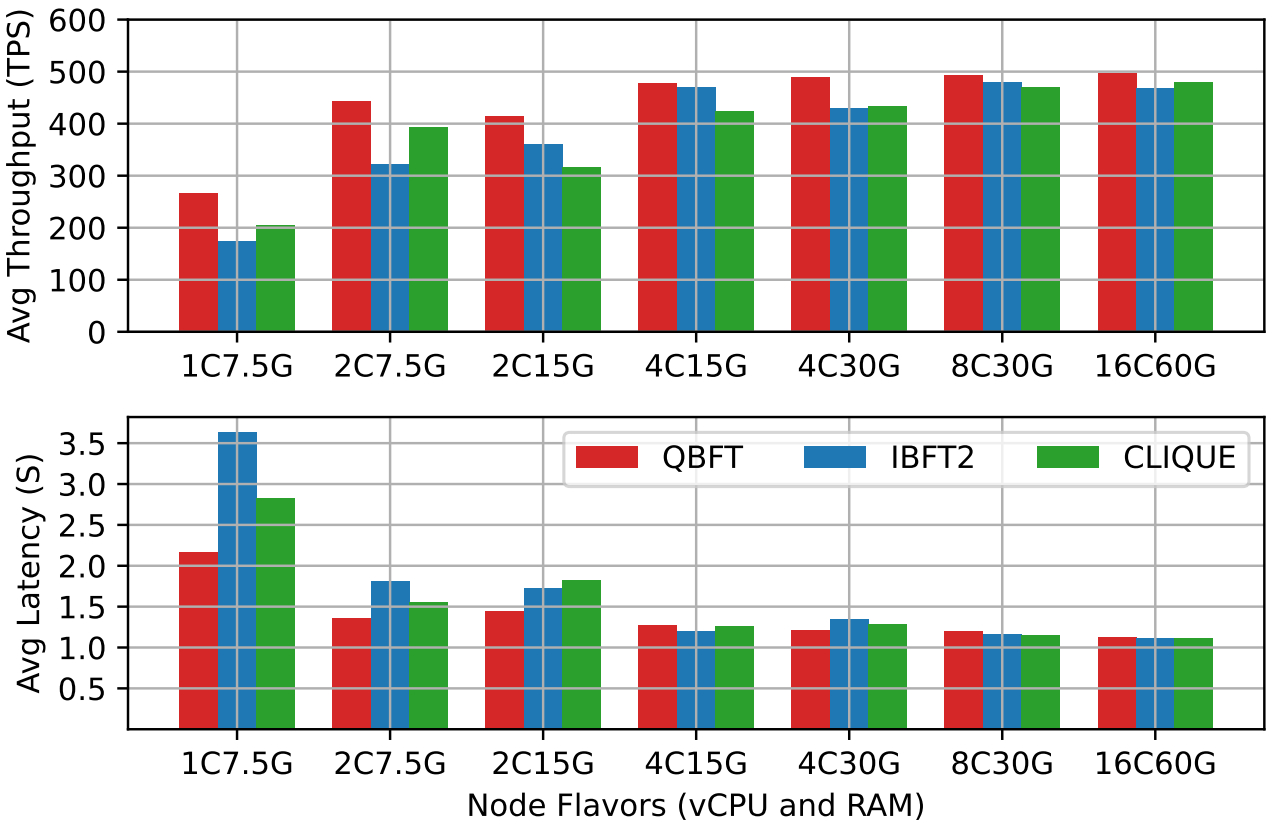


Figure 5.6: Vertical scalability against different PoA consensus algorithms at 1,000 req/s open transaction send rate.

discuss the limitations.

Load Balancing: First, a load balancer is significant to improve the system performance for decentralized applications built on a private Besu blockchain. In a public blockchain network, workloads are assumed to be balanced among all the connected nodes because of their fully decentralized nature. But, in a private blockchain network in enterprise applications, a load balancer needs to be considered in system design.

Scalability: The scalability of a vote-based blockchain system has been studied in [112], [113]. Similar to the conclusion in previous works, we found that Besu has limited vertical scalability in terms of node size, while it does not horizontally scale well in terms of an increasing number of validators. When a validator is more powerful, it takes less time to execute transactions, create blocks, and process consensus tasks. By analyzing the logs, we found that a Besu network validator with more computational resources generally has more transactions packed into a single block, namely a bigger block size, at the same block creation frequency and transaction send rate. However, this improvement becomes less and less significant as the node size grows to a high level (e.g., 16C60G), in which the hardware resource is no longer the bottleneck, and it can hardly

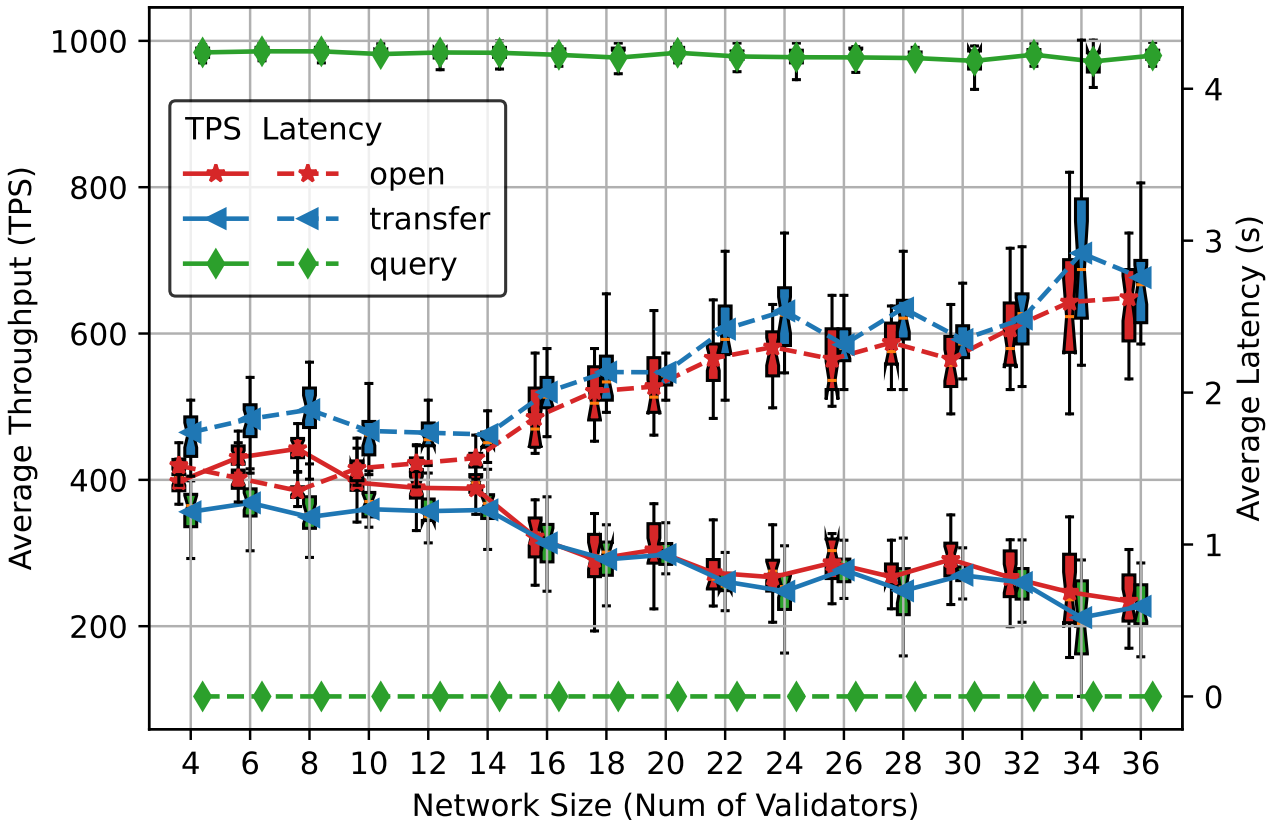


Figure 5.7: Horizontal scalability of QBFT at 1,000 req/s send rate. Solid lines indicate throughput; dashed lines represent latency.

reduce the computation time any further. For the network size, increasing the number of validators will increase the block propagation and consensus time in a proof of authority blockchain. Since the message transmission is very efficient in a private network, Besu QBFT performance does not have an obvious drop until the network size reaches 16. The tested scalability result of QBFT is lower than the official scalability of IBFT 2.0, which claims that IBFT 2.0 handles up to 30 validators with no loss of performance.

To further understand how the consensus time contributes to latency as network size increases, we extracted the consensus time of all blocks from logs. The consensus time of each non-empty block is calculated by the following equation:

$$T_{\text{consensus}} = TS_{\text{import}} - TS_{\text{create}} \tag{5.1}$$

where TS_{import} is the timestamp of the last node importing the block, and TS_{create} is the timestamp of block creation. Note that only open and transfer transactions contribute to the consensus time as query operation is not involved in any consensus process. Figure 5.12 shows the

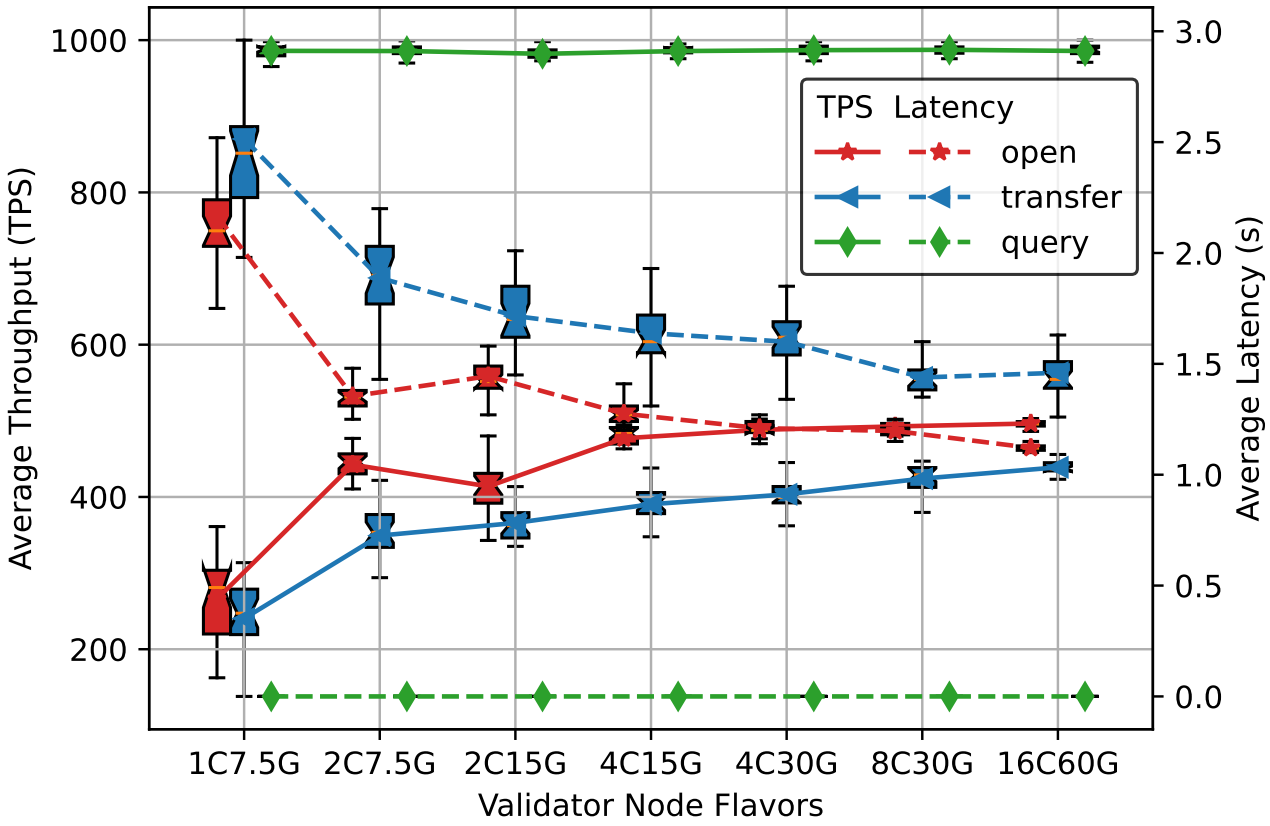


Figure 5.8: Vertical scalability of QBFT at 1,000 req/s send rate. Solid lines indicate throughput; dashed lines represent latency.

box plots of QBFT consensus time and the quorum size when the number of validators increases from 4 to 36. We observe that the consensus time increases slowly as the network size increases before reaching 30. Then, it has a significant growth from 30 to 34. The mean consensus time is even over one second at 34 nodes configuration. By comparing this figure with Figure 5.7, we can see that as the number of nodes increases, the consensus time occupies a higher and higher proportion of the entire transaction latency. In contrast, non-consensus time almost remains unchanged, as shown in Figure 5.11. This indicates that consensus time is the most significant contributor to increasing transaction latency. However, the non-consensus time dominates the latency in all the tested network sizes, even though we see a clear trend that the consensus time will become the new latency contributor as the network scales.

Figure 5.13 shows the consensus time of Besu QBFT under various node flavour configurations. We can observe that the consensus time decreases when node flavour changes from a small size to the baseline configuration, then it has a subtle improvement even increased to an extra large instance, i.e., 16 vCPU and 60GB RAM. This indicates that node computation resources have a limited impact on consensus time in a Besu network with a fixed number of validators, e.g., 8. A

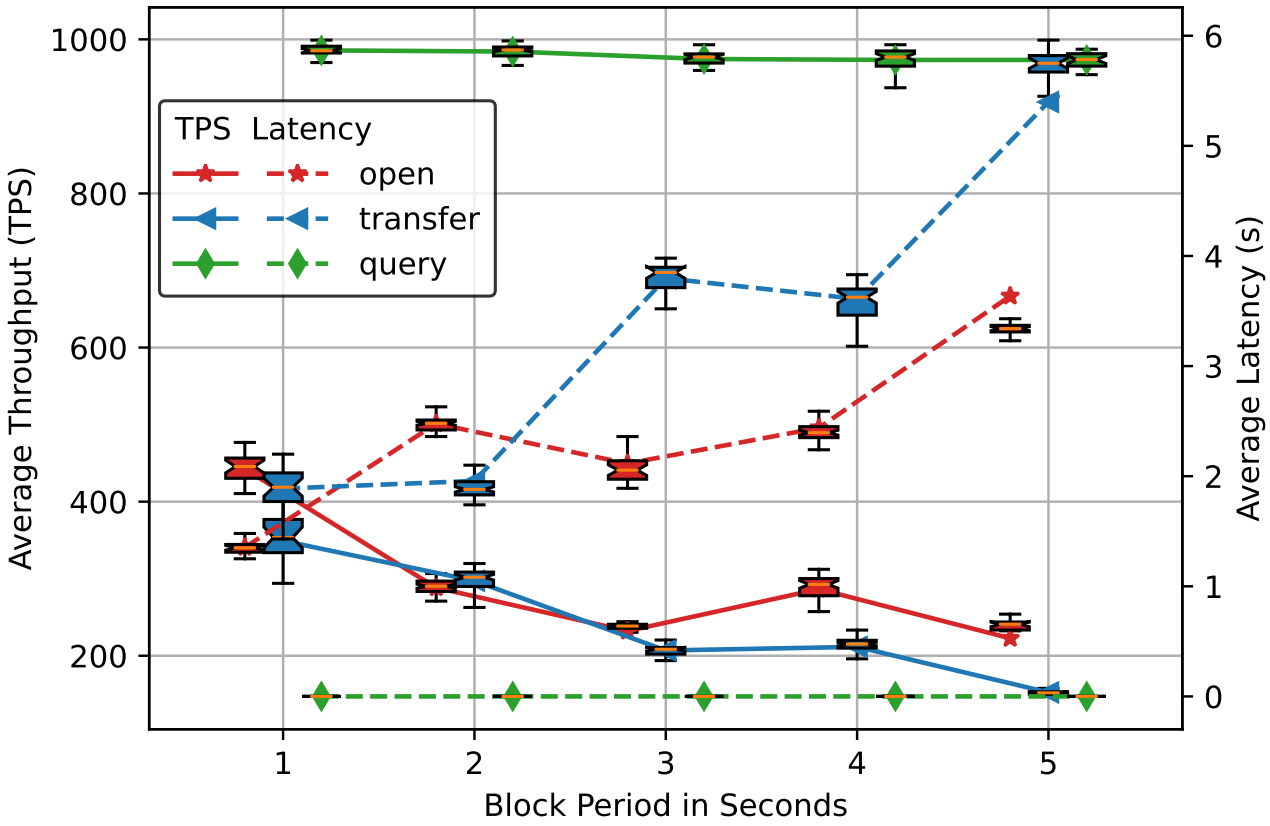


Figure 5.9: Performance against varying block times at 1,000 req/s send rate.

Besu node with adequate resources, for example, 2vCPU and 7.5GB RAM, can efficiently create blocks, process consensus messages, and update states to the database in the QBFT consensus.

Block Time: Another critical parameter, block time, determines the block frequency, which directly impacts the latency and throughput of Besu. This parameter should be carefully configured according to the application's needs in practice. On the one hand, if a block period is very short (e.g., 1s), Besu will generate blocks very frequently. This results in high throughput and low latency for consensus algorithms with finality properties such as QBFT and IBFT 2.0. However, a short block period will cause more forks in a Clique network, which in turn degrades the performance. In addition, a lot of empty blocks will be created in the case of low transaction loads, resulting in low storage utilization. On the other hand, if a block period is very long, transactions need to wait for a long time to be packed into the next block, which significantly reduces Besu performance. In a small private network, block creation and processing times are the prominent components of transaction latency. Unlike Geth [113], the performance of Besu is not inversely linear against the increasing block period seconds. This indicates that Besu is not as efficient as Geth in creating and processing blocks so the validator is not capable of packing all the received

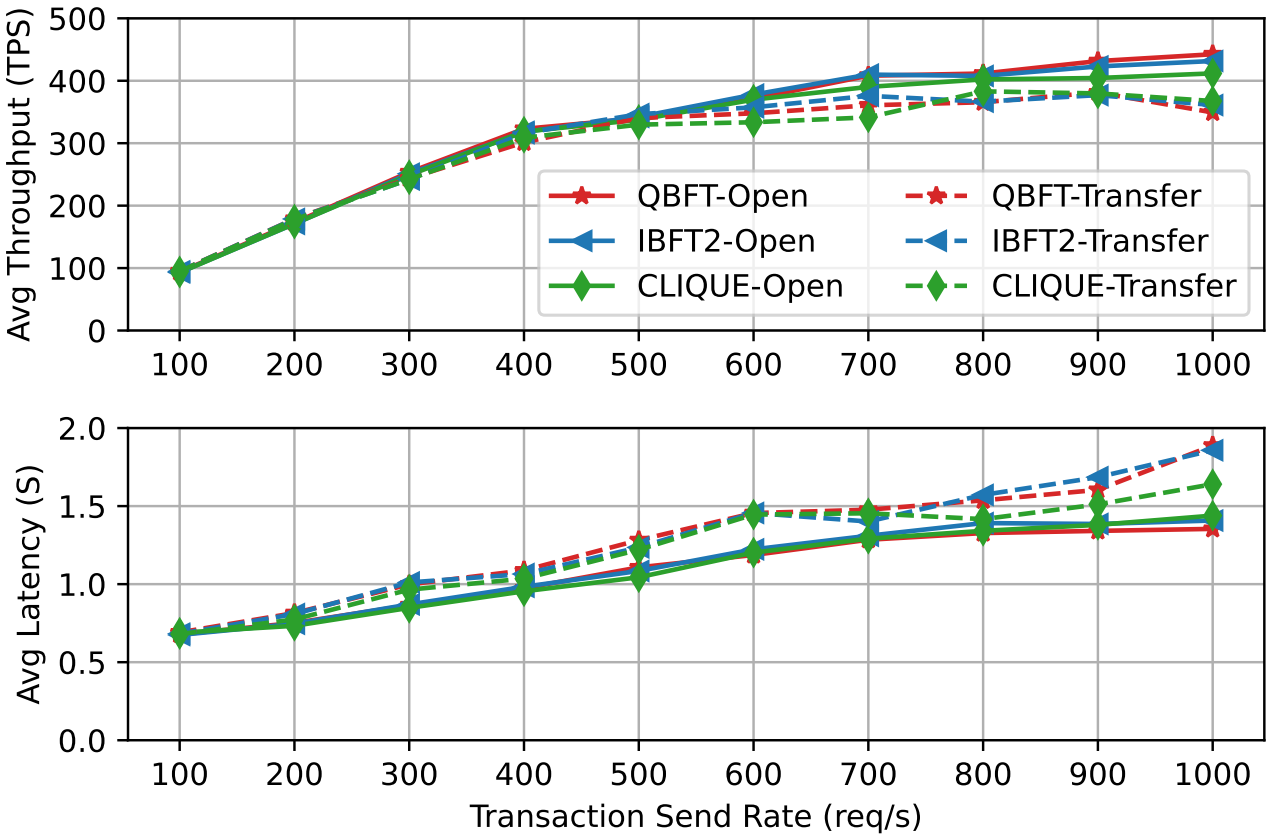


Figure 5.10: Performance against different proof of authority consensus algorithms under varying transaction send rates.

workload (1,000 transactions) into a single block within a short period (e.g., one second). Instead, the rest are pending and waiting for the next block.

Block Size has a direct impact on the throughput of the Besu network by determining the number of transactions in a block. Figure 5.14 shows the box plots of block size against varying transaction send rates under the baseline configurations, where Caliper sent 100,000 open and transfer transactions in total; out of them, 99,432 were executed successfully and wrapped into 383 blocks. The mean block size increases linearly from 100 to 400 req/s send rate, then grows slowly as the transaction send rate increases. This trend keeps aligned with the baseline throughput in Figure 5.3. We can also see larger size variations and many outliers starting from 500 req/s transaction send rate. This is because proposers can not process a large number of transactions and wrap them all into the current block in one second, leaving a small portion to be processed in the next block.

Consensus: Different types of PoA consensus algorithms (i.e., Clique, IBFT 2.0, and QBFT) in Besu have a very similar performance profile in our experiments. Theoretically, Clique should reach consensus and add blocks faster than IBFT 2.0 and QBFT under the same configuration,

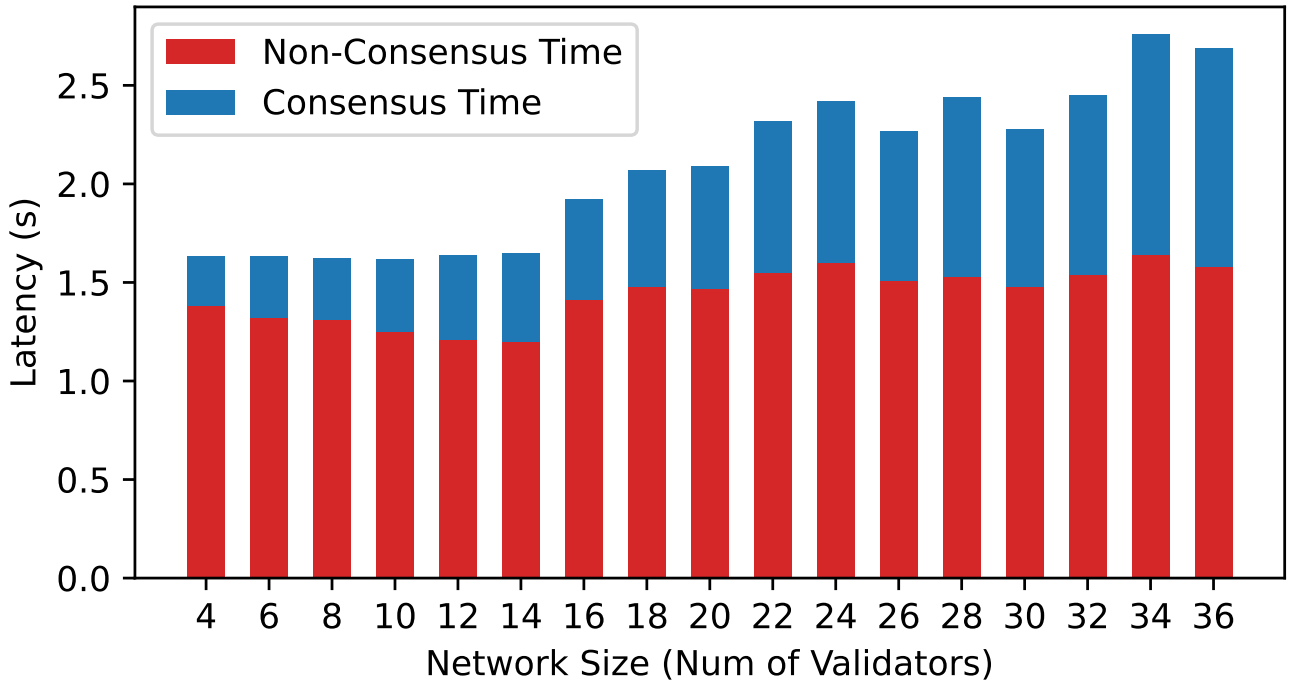


Figure 5.11: Besu consensus time v.s. non-consensus time in latency against varying network size.

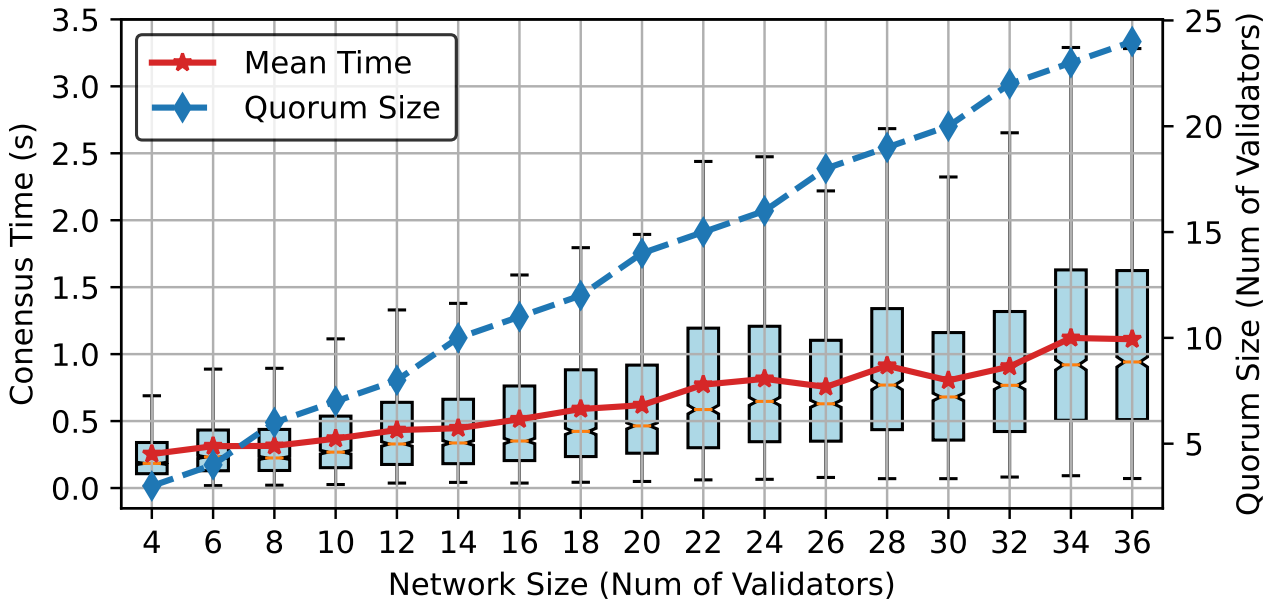


Figure 5.12: Besu QBFT consensus time and quorum size against varying network size. Quorum size is the minimum number of validators voting on a block in a stage so that the consensus can proceed to the next stage.

since the latter two require three extra communication stages before adding blocks to the chain. However, for Clique, the probability of a fork and out-of-order sealing increases as the number of validators increases, especially at a small block period. From our experiments, three PoA con-

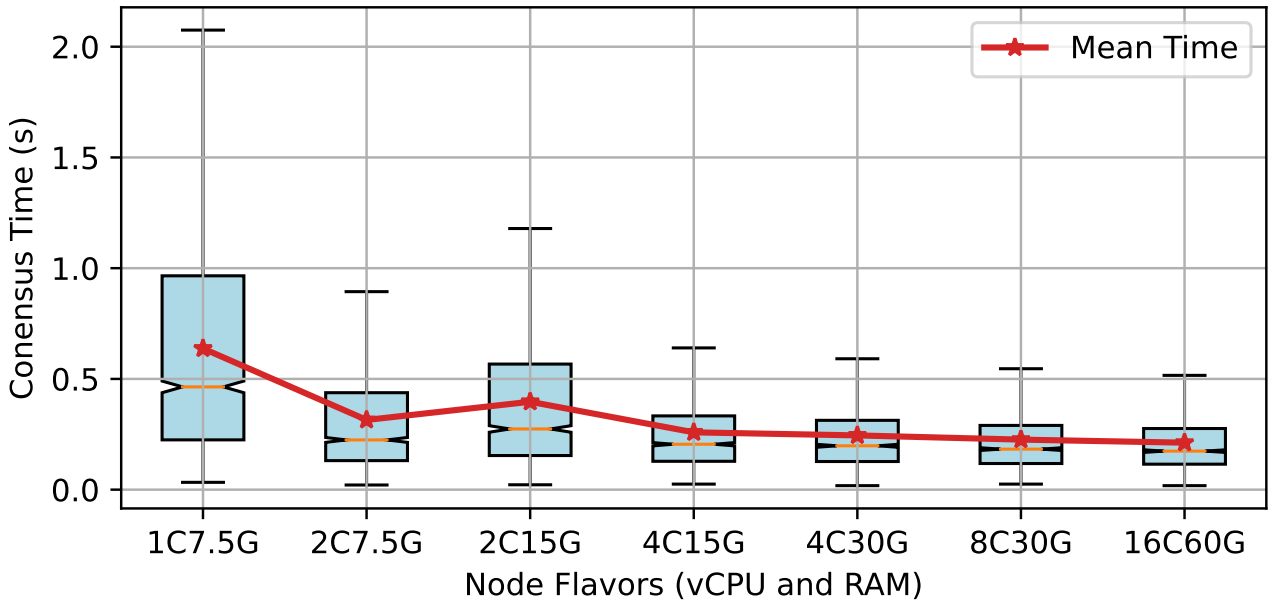


Figure 5.13: QBFT consensus time against varying node flavours.

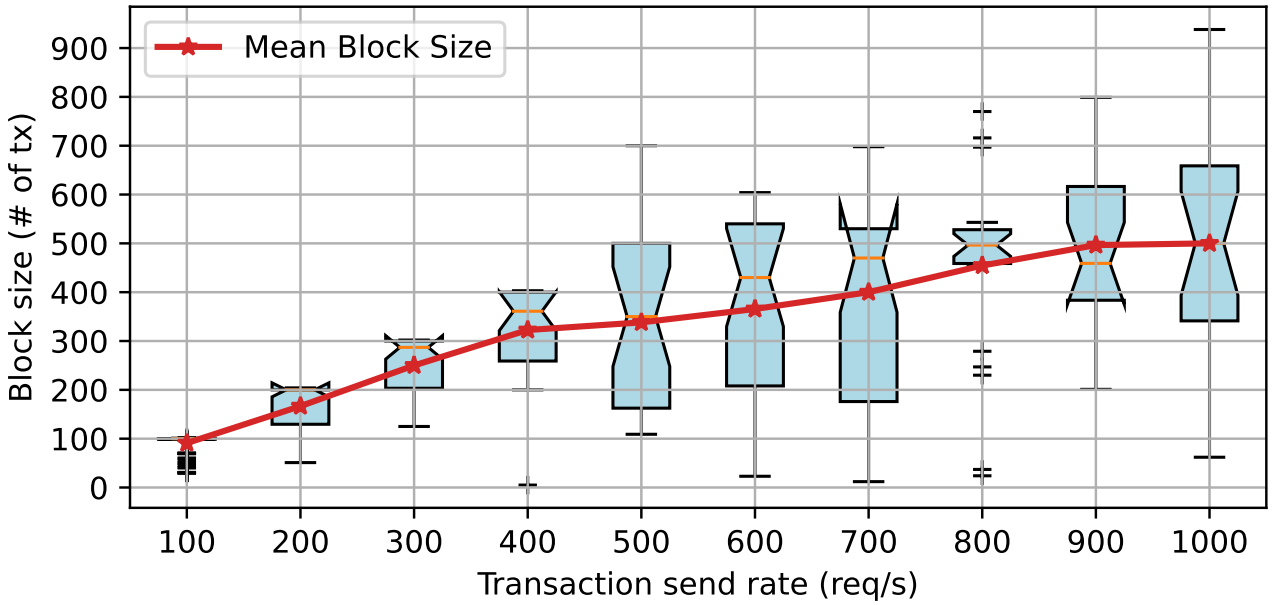


Figure 5.14: Block size against varying transaction send rates.

sensus algorithms have similar performance, which indicates that consensus time cost is negligible compared to other operations such as transaction execution and blockchain state updates in a small private blockchain network with 8 nodes.

Transaction Types: At last, workload type is one of the most important factors influencing the Besu network performance. This can be seen in all our experiments that open transaction

consistently has a better performance than transfer. The root cause is that transfer transactions have two write operations, while open transactions only have one. Therefore, under the same node configuration, Besu can execute more open transactions than transfer and pack them into a block, which increases the block size and improves the performance.

Limitations: All the findings and conclusions are constrained by specific experimental setups on the OpenStack cloud, which limits the generalization. For example, since all VMs are deployed on the same server cluster with high-speed communication among nodes, the network delay is negligible. In contrast, a real production deployment may experience higher latency and lower transaction throughput under the same node configurations. In addition, the load balancer may incur additional development concerns about session persistence and node affinity in practice.

5.6 Related Work

Many studies have conducted performance analyses on various private blockchain networks [37], mainly focusing on Hyperledger Fabric (HLF) and private Ethereum. Most of these studies use Hyperledger Caliper [98] to benchmark and evaluate the performance.

Performance analysis of Hyperledger Fabric was discussed in [10], [12], [48], [69], [94], [124], [127]. Baliga *et al.* [12] studied the throughput and latency of HLF v1.0 using Caliper by tuning transaction and chaincode configuration parameters, such as the number of chaincodes, channels, and peers. Thakkar *et al.* [124] explored the performance bottlenecks of the HLF v1.0 under different block sizes, endorsement policies, number of channels, resource allocation, and state database choices (GoLevelDB v.s. CouchDB). Based on the identified bottlenecks, the authors suggested three optimization solutions, i.e., parallel VSCC validation, the cache for a membership service provider (MSP), and bulk read/write for CouchDB, which were implemented in HFL v1.1. Androulaki *et al.* [10] explored the impact of block size, peer CPU, and SSD v.s. RAM disk on blockchain latency, throughput, and scalability under different numbers of peers in HLF v1.1. Nguyen *et al.* [94] conducted an experimental study to explore the impact of large network delays on the performance of HLF v1.2.1 over an area network between France and Germany. In [69], Kuzlu *et al.* analyzed throughput, latency, and scalability of Hyperledger Fabric v1.4, the first long-term support release, under varying transaction send rates and participants. Meanwhile, Geyer *et al.* [48] used Caliper to benchmark HLF by tuning network parameters, e.g., latency and packet loss, and to explore the influence of transaction rate, chaincode, network properties, local network impairment, and block size. Wang [127] researched the performance of HLF under abnormal behaviours by designing multiple malicious behaviour patterns and testing the transaction

throughput and latency in such contexts. The results show that delay attacks, along with keeping some replicas out of working, dramatically decrease the system's performance.

Performance analysis of the private Ethereum platform was discussed in[16], [109], [112], [113]. In [109], Rouhani and Deters analyzed the two most popular Ethereum clients, Geth and Parity, in a private network. The results indicate that, compared to PoW-based Geth, PoA-based Parity is 89.82% faster in transaction processing, on average, under different workloads. In [113], Schaffer *et al.* introduced a concept for measuring the performance and scalability of private Ethereum smart contract platforms to examine the impact of various parameters on Ethereum performance. In [16], Bez *et al.* analyzed the scalability of Ethereum under an extensible test environment with synthetic benchmarks. In a recent work [112], Samuel *et al.* proposed a performance test tool by integrating a load-balancer middleware, to comparatively evaluate three mainstream Ethereum clients with their proof of authority consensus algorithms, namely Geth 1.9.18 with Clique, OpenEthereum(Parity) V3.0.1, and Hyperledger Besu 1.5.0 with Clique and IBFT 2.0, for private blockchain. Unfortunately, they only evaluated the throughput and scalability in several configurations. In addition, due to the limitations of their queue and nonce management design, the tested throughput in load balancing was even worse than the non-balanced results.

Some other performance evaluation of private blockchain was discussed in [13], [118]. In [13], Arati *et al.* comparatively studied the throughput and latency characteristics of Quorum, a permissioned blockchain platform built from the Ethereum codebase, under Raft and IBFT consensus, respectively. Shi *et al.* [118] empirically studied the performance on consistency, stability, and scalability of Sawtooth, another well-known permissioned blockchain platform from Hyperledger.

Performance evaluation of private blockchain in the context of peer-to-peer transaction applications was discussed in [1], [80], [105]. Lohachab *et al.*[80] proposed a secure peer-to-peer energy trading framework, called HFPET, based on Hyperledger Fabric. They evaluated two different versions (i.e., v1.4 and v1.4.1) of Fabric through extensive experimentation using Caliper. In [105], Pradhan *et al.* proposed a flexible permission ascription (FPA) based Hyperledger Besu IBFT 2.0 for peer-to-peer energy trading, and conducted comparative performance evaluation between Besu and other blockchain platforms, including Hyperledger Fabric Raft and Kafka, and Ethereum Ethash. In another recent work [1], Abdella *et al.* compared the performance of a proposed Hyperledger Besu IBFT-based system, called UBETA, with other three blockchain systems (i.e., Ethereum Clique, Ethereum PoW, and Hyperledger Fabric Raft) using specific performance metrics. However, the authors used their own test metrics and tool rather than the popularly adopted Caliper benchmarking method, which might impact the confidence in the results.

Different from all the above studies, our work is the first one to research the performance of

QBFT consensus, and systematically study the performance of Hyperledger Besu for enterprise applications.

5.7 Summary

In this work, we introduced a load balance-based blockchain performance evaluation architecture and leveraged it to benchmark the Hyperledger Besu private blockchain. We designed and conducted extensive experiments to evaluate the impact of selected parameters that may affect the network performance, such as transaction send rate, network size, node flavour, load balancing, consensus, and block time. Our results indicate that blockchain parameters such as block time and block size have the most significant impact. In addition, Besu's performance is limited by block creation and transaction execution, which are determined by various factors, such as node configuration, transaction complexity, and load balancing. We also find that QBFT has slightly better scalability than IBFT 2.0 and Clique in Besu. It can scale up to 14 validators without obvious performance loss. All these findings, along with root cause analysis, have paved the way for further analysis and performance improvement of Hyperledger Besu. For future work, it will be interesting to investigate the private transaction performance of the Enterprise Ethereum blockchain and compare the performance of different clients implementing the same consensus. In addition, we would like to build an analytical model to characterize the performance of Besu.

6 Performant Blockchain for Peer-to-Peer Energy Trading

In this chapter, we first propose a unified performant blockchain-based microservice architecture for peer-to-peer energy trading (P2P-ET). Then, we introduce more technical details, including microservice design, smart contracts design, blockchain network deployment, algorithms used to determine market prices, and the system implementation. At last, we conduct a case study to demonstrate our proposed solution's feasibility, scalability, and effectiveness, using the real trading data from the Alberta Energy System Operator (AESO) [1] on our implemented blockchain-based peer-to-peer energy trading (BPET) system. Our modular design enables different types of blockchain networks, and consensus mechanisms to be plugged in and used to provide services for energy trading.

6.1 Introduction

Humans are experiencing an energy revolution in responding to global warming and environmental pollution issues. Renewable energy, also called low-carbon energy, plays a vital role in this revolution to meet our increasing daily energy needs. In 2012 in New York City, Hurricane Sandy destroyed the century-old concept of utility power supplies and heralded a new era of distributed energy supplies that value resilience over tradition. With the expansion of distributed renewable energy sources, such as solar panels, wind turbines, geothermal energy, plug-in electric vehicles, and bioenergy, there is an urgent demand for a more decentralized and open market to trade energy among those small-scale energy producers called prosumers and consumers. The direct energy trading between peers, such as energy from small-scale distributed energy resources (DERs) in dwellings, offices, factories, etc., is defined as peer-to-peer energy trading (P2P-ET) [137], which can improve resource utilization and consequently contribute to addressing the impending electricity crisis and environmental pollution issues.

State-of-the-art energy trading systems mostly rely upon a centralized platform controlled by

[1] https://www.aeso.ca/

a single authority to process, store and manage a vast amount of transaction data. However, these centralized systems may encounter many vulnerabilities from both inner malicious actions and outside network attacks. On the one hand, electricity consumers have a potential motivation to tamper with the data on their meters to benefit themselves, which is usually called "stealing electricity". An immoral system operator with administrator permission can also easily tamper with households' electricity usage records. On the other hand, attackers may distort the energy data stored on the centralized system once compromised, leading to privacy leakage and data integrity issues.

A blockchain-based system provides a tamper-proof and decentralized solution to ensure the originality and authenticity of the energy data in a mathematical way. Unlike the centralized system, blockchain stores data copies into a large number of decentralized ledger nodes geographically distributed worldwide. For example, as the first and largest blockchain project, the Bitcoin network has 15,150 nodes across 95 countries (as of September 2022), including the United States, Germany, France, Netherlands, and Canada [2]. In the application of P2P-ET, traders use a popular hashing algorithm such as SHA-256 or Keccak-256 to generate a hash value from each transaction and then make a Bitcoin transaction by writing the hash value into this transaction. In such a way, once this transaction is processed and stored in all Bitcoin nodes, the input hash value will act as a tamper-proof digital signature of the energy trading transaction. The traders can prove the originality and integrity of this transaction anytime, and any slight tampering with that transaction will be easily detected because Bitcoin stores data in a structure called a block, and links each block to the previous one in a cryptographic way. It uses the proof of work (PoW) consensus algorithm to determine the ledger record right. Only an entity with a computation power of more than 51% of the whole network can possibly launch a successful attack to tamper with the historical data, which is almost impossible to achieve with the latest technologies as Bitcoin has a large number of nodes distributed worldwide. Therefore, blockchain technologies developed from Bitcoin have attracted a lot of attention from researchers and industrial practitioners to resolve trust and information security issues in many industries including energy trading.

However, a public blockchain running the PoW consensus can hardly meet the transaction efficiency and system scalability requirements of enterprise applications. For example, the bitcoin network takes around 10 minutes to generate a block and validate the transactions inside a block. This is unacceptable for most trading applications where a near real-time transaction process is usually required. Moreover, PoW consumes a lot of power in resolving a mathematical puzzle by calculating a meaningless hash value. In practice, a blockchain-based P2P-ET system is more

[2]https://bitnodes.io/nodes/

than a payment system. It also serves the whole energy trading process from end to end through smart contracts, such as bid/offer acceptance, auction rules execution, market price calculation, energy delivery control and payment settlement. In other words, it serves as a secure data storing and sharing platform among all market participants. More generally, it can serve as a machine-to-machine (M2M) communication platform in a smart grid, where smart meters can communicate with each other to automatically trade energy at any time. Such an Internet of Things (IoT) use case requires the P2P-ET system to provide significantly high throughput, e.g., hundreds or even thousands of transactions per second in a large-scale application and low latency within seconds.

To this end, permissioned and consortium blockchains were developed to meet the requirements of enterprises. Among all permissioned blockchain projects, the Enterprise Ethereum client Hyperledger Besu wins a lot of attention due to its high performance and versatility. In addition, DAG-based distributed ledger technologies (DLTs) were proposed to facilitate blockchain applications in IoT by providing higher performance and better scalability. IOTA is the most successful project among this type of DLT. Based on our previous introductions in Chapter 2 and performance analysis of these two types of DLTs in Chapter 4 and Chapter 5, we briefly summarize the main characteristics of Hyperledger Besu and different versions of IOTA in Table 6.1.

Table 6.1: Comparison between Hyperledger Besu and different versions of IOTA.

Items	**Hyperledger Besu**	**IOTA1.0 (IRI)**	**IOTA1.5 (Chrysalis)**	**IOTA2.0 (Coordicide)**
Network Type	Private, Public	Public, Private	Public, Private	Public, Private
Smart contract	Yes	No	No	Yes
Throughput	400+ TPS	1000+ TPS	1000+ TPS	NA
Latency	∼4 seconds	60 seconds	10 seconds	NA
Scalability	Medium	High	High	High
Decentralization	High	Low	Low	High
Consensus	PoA(QBFT, IBFT2, Clique) PoW(Ethash)	COO, PoW	COO, PoW	Voting-based (FPC)
Permissioning	Smart contract, PermissioningService	NA	NA	NA
Privacy	Tessera	MAM	MAM	NA
Energy Consumption	Medium	Low	Low	Low
Dev Stage	Industry ready	Deprecated	Industry ready	In Development
Community Support	Good	NA	Good	Fair

6.2 A Unified Architecture

In reality, energy trading happens at different levels among corresponding stakeholders in various existing energy markets. In an energy distribution network, big generators are transferring from traditional fossil-based energy, e.g., coal electricity, to cleaner energy, e.g., heat and gas electricity, and renewable energy resources, e.g., solar panels and wind turbines. At this level, generators can directly sell energy to big consumers like factories in a well-designed wholesale market. Any suppliers with aggregated energy generation and buyers with a significant consumption need can trade energy with each other in a peer-to-peer way.

At the end-user level, prosumers and consumers can trade energy in a local market within a microgrid. For example, in a smart community, residents first self-suffice their household electricity needs through the installed solar panels on their roofs in day times. Then, if surplus electricity is generated, they can sell it to their neighbours. All the suppliers and consumers in a community connect to form a smart microgrid. A microgrid connects to the main grid and participates in the wholesale market as a single entity. In the same way, a microgrid first self-suffices its local residents, then sells the aggregated surplus energy to the wholesale market if the microgrid generates more energy than it needs. When the self-generated electricity cannot meet the community needs in a microgrid, this microgrid buys energy from the market, which is called a community-based peer-to-peer energy trading market [81], [121], [128].

When all these market participants want to join a single market and get the most significant benefit, energy management becomes more complex. The market needs a redesign, and data needs more security while maintaining high system efficiency and reliability. For example, when adding a new solar farm to an existing electricity wholesale market, the electricity should be delivered as reliably and securely as before. The traditional trusted centralized management system can hardly meet the requirements of security, privacy and high performance and scalability at the same time.

Managing the distribution wholesale market, end-user retail market, and the emerging decentralized energy resources (DERs) in a secure and efficient way becomes an urgent challenge for system designers and market governors. To this end, a performant blockchain-based system is a promising solution to resolve the issues mentioned above by providing a decentralized paradigm, as shown in Figure 1.1.

6.2.1 Markets Analysis

From the perspective of energy trading business models, there are three typical market models: bilateral market, pool market, and balancing market [1]. The first two markets can operate

independently with their market mechanisms and payment systems. All markets can also depend on each other and work together to achieve better system performance and transaction efficiency.

- **Bilateral market**: a pair of buyers and sellers negotiate to agree on the final energy price and amount in a periodic manner, which can be long-term. Buyers and sellers usually negotiate price and quantity in an off-chain manner and then leverage blockchain to record the transactions. This market mechanism with a long-term negotiation process can also be transited to on-chain.

- **Pool market**: many buyers and sellers put bids in a pool. The market determines the price and transaction volume based on the demand and supply curves. The **merit order effect** trading model is usually used as the pricing strategy in the pool market, which can be easily implemented in a smart contract [1] and executed on the blockchain.

- **Balancing market**: as its name indicates, a balancing market exists to supplement the bilateral and pool markets and ensure total power demand and supply are balanced at all times in physical energy networks. This market relies on the information received from the pool and bilateral markets to launch real-time energy trading.

From the perspective of market structures, all P2P-ET markets can be divided into three types: full decentralized market, community-based market, and composite market [85].

1. **Full decentralized market**: A fully decentralized P2P-ET market provides a platform for participating prosumers to independently and directly negotiate with one another to decide on the energy trading parameters, such as amount, price, and energy type, without any centralized supervision. Such decentralization of a P2P market usually relies on bilateral contracts between individual prosumers and consumers.

2. **Community-based market**: A community-based P2P-ET market relies on a community manager to enable each member to trade its energy within the community. When a peer chooses to trade their energy with someone outside the community, the community manager acts as an intermediary between the community and the outside world. Thus, a community manager manages the energy trading activities within the community by imitating the role of an auctioneer and also acts as an aggregator to bridge the community with the rest of the system.

3. **Composite market**: A composite market is a combination of fully decentralized and community-based markets in which each community and every single prosumer can interact

with one another while maintaining their market properties. The manager buys energy from a regulated market when not enough power is generated from prosumers in a community.

In recent years, a lot of research has been working on P2P-ET markets, including solutions with or without blockchain technologies. We briefly list some representatives with their market types, pricing mechanisms, and smart contracts used in Table 6.2.

Table 6.2: Comparison of Different Markets for P2P Energy Trading

Ref	**Market Type**	**Pricing Mechanism**	**Smart Contracts**
[79]	Pool	Stackelberg game	NA
[3]	Bilateral	Off-chain negotiation	NA
[62]	Pool	iterative double auction	NA
[46]	Bilateral	off-chain negotiation	NA
[1]	Pool, Bilateral, Balancing	merit-order effect	PM, BiM, BM, PS, RC
[34]	Bilateral, Pool	bilateral auction, merit-order effect	OLICoin, OLIDetail, ChildAuction, ParentAuction, OLIBilateral
[52]	Bilateral, Pool	Vickrey auction	VickreyAuction
[126]	Bilateral	Negotiated pricing	CES Chaincode
[80]	Pool	Balance service operator	AO, RIT, SIT

PM: pool market, BiM: bilateral market, BM: balancing market, PS: payment settlement OLI: an energy-blockchain startup in Germany, RC: registry contract, AO: account opening, RIT: receiver-initiated trading, SIT: sender-initiated trading, CES: crowdsourced energy systems

In [79], Li *et al.* devised a unified blockchain-based framework for peer-to-peer energy trading for three typical ET scenarios in IIoT: microgrids, energy harvesting networks, and vehicle-to-grid networks. In addition, they proposed an optimal pricing strategy using the Stackelberg game for credit-based loans to maximize the utility of the credit bank. In [3], authors implemented a proof-of-concept for a decentralized energy trading system using blockchain technology, multi-signatures, and anonymous encrypted messaging streams, enabling peers to negotiate energy prices anonymously and perform trading transactions securely. In [62], Kang *et al.* proposed a localized peer-to-peer (P2P) electricity trading system with consortium blockchain, called PETCON, for locally buying and selling electricity among plug-in hybrid electric vehicles (PHEVs) in smart grids. They utilized an iterative double auction, in which the auctioneer solves an optimal energy allocation problem to achieve effective market equilibrium after receiving all bid prices from buyers

and sellers. In such a way, the auctioneer determines the final trading prices and the amount of traded electricity, avoiding revealing private information about PHEVs during electricity trading directly. In another work, Gai *et al.* [46] presented a consortium blockchain-oriented approach to resolve the privacy leakage problem without restricting trading functions. Specifically, they created noises to prevent malicious activities and leveraged an account mapping mechanism to entirely change the distributions/trends so privacy attacks using data mining algorithms could be defended. However, these solutions remained off-chain, and no smart contracts were developed to execute the market rules. In [52], Hahn *et al.* designed a decentralized architecture for a transactive energy auction on campus. They developed a Solidity smart contract to execute the Vickrey auction, in which bidders submit written bids without knowing others' bids in the auction. The highest bidder wins but pays the second-highest bid price. In [34], Faizan *et al.* devised a blockchain-based microgrid energy market model and developed smart contracts to enable P2P-ET in both bilateral and pool markets. An energy token called OLICoin was designed to address the price volatility of cryptocurrency for transaction payment. In [126], Wang *et al.* developed an optimization model and blockchain-based architecture to manage the operation of crowdsourced energy systems (CESs) with P2P-ET transactions. A prototype was implemented using Hyperledger Fabric, and a case study was conducted to show the feasibility of the proposed model.

In a more recent work [1], Abdella *et al.* proposed a unified blockchain-based energy trading architecture, called UBETA, on top of a permissioned Hyperledger Besu network with the Istanbul Byzantine Fault Tolerance (IBFT) consensus. Three typical markets were considered: bilateral market, pool market, and balancing market. They implemented smart contracts in Solidity and Golang according to the *merit order effect* trading model for the pool market to provide an on-chain pricing mechanism for energy trading. In addition, they comparatively evaluated the performance of blockchain under Hyperledger Besu and Hyperledger Fabric with different consensus mechanisms. In [80], Lohachab *et al.* proposed a Hyperledger Fabric-based framework for Peer-to-Peer Energy Trading (HFPET) among PHEVs and conducted performance evaluation under different system parameters to develop the proof of concept for PET applications. In this framework, buyer smart energy nodes (SENs) will initiate trading if their energy buffer is less than the minimum requirement or if necessary; sellers will submit a proposal to sell energy if their energy level exceeds the maximum requirement or is necessary. However, it is unclear how the balance service operator maps the amount and price from buyer SENs to the individual seller SENs.

From these market designs, we see that most works focus on defining market rules and imple-

menting rules using smart contracts, needing a comprehensive solution taking systematic requirements such as scalability, dependability, and resiliency into account. To this end, a service-oriented and modular P2P-ET system based on blockchain as the decentralized data storing and sharing is promising to tackle the increasing complexity of energy trading market models.

6.2.2 Participants Analysis

Even though there are more and more rules and roles emerging in the energy market, there are essentially four major types of market participants in a complex electricity market.

1. Producers/Prosumers: The producers, also called suppliers, offer energy for sale. In its simplest form, they have three functions: to register or update basic information such as the asset identification, the supply capacity, and energy type; to place an offer to the market, including the price of energy on offer and quantity available for selling; and finally, to sell energy over a time interval or in real-time, from storage, electric vehicle or directly from generators to consumers. In a microgrid, a producer can also be a prosumer who sells the surplus energy to neighbours in a smart community.

2. Consumers: The consumers place bids on the quantity of energy on the platform. They also have three functions: to register and update their basic information on the platform, such as the asset identification and the maximum load; to place a bid including the type and amount of energy they want; and finally, buy and consume energy from the platform.

3. Market administrator: The market administrator designs the market rules by writing rules in smart contracts, deploying the smart contracts to the blockchain, and managing payment after the trading process completes. This role is usually taken by a selected and experienced individual or a group of skilled individuals such as system controllers and energy coordinators. Even though they are in charge of administration work, it does not mean that this is a centralized role because the administrator's permission is limited. Every transaction issued by the administrator still needs to go through the consensus among blockchain network nodes before getting confirmed.

4. Blockchain nodes: Different types of participants can run blockchain nodes in a particular local energy market. Typically, entities with specific authority or reputation, such as distribution system operators (DSO), transmission system operators (TSO), market operators (MO), regulators and utility companies, take the role of validator nodes to run a permissioned

blockchain network. Sometimes, prosumers and consumers also run normal blockchain nodes to achieve better decentralization, especially in a microgrid.

Among all participants, the market administrator is responsible for implementing rules for smart contracts, network nodes are responsible for executing rules, and providers and consumers follow the rules to complete energy trading transactions. If a new participant wants to join the quorum of validators, one of the existing validators first needs to send a transaction proposing this participant as a new validator. Then, at least a super-majority of the existing validator nodes must agree on this proposal through consensus to accept the newcomer's application. To become a legitimate new trader, a user only needs to connect to the blockchain network through the front-end application or RPC APIs and register as a supplier or consumer by filling in some basic information.

6.2.3 A Blockchain-based Microservice Architecture

An energy trading system should be as resilient as a distributed energy network. When transaction data flows from one peer to another on the trading system, the corresponding electricity supply flows reversely on the underlying energy distribution network. Both data and energy supplies can be designed as services, which can be further combined to become digital energy services. To achieve these goals, we propose a unified Blockchain-based P2P-ET system called BPET, combining microservice architecture with blockchain technology. In summary, our proposed BPET system has the following design requirements:

- Reliable: it should eliminate any single point of failure and be tolerant to any one or more data storage nodes failure or off-line. System faults should be isolated in individual modules, meaning that a single module failure should not affect the functionality of another module.
- Secure: all transaction data should be cryptographically protected and stored in the system with the consensus of all participants. No attackers can tamper with the stored information.
- Decentralized: transaction data should be stored in the system in a decentralized manner, which means that all nodes store identical copies of the ledger. No central authority can control the stored information or tamper with a single transaction record of the system.
- Performant: it should provide high throughput in transactions per second (TPS) and a low transaction latency in seconds. When users interact with the system, they should have a smooth experience.

- Scalable: it should process a large scale of user requests and transactions. When more market participants join the system with an increasing number of blockchain network nodes, the system should increase the throughput accordingly by processing more transactions per second to meet the performance requirements.

To reach these requirements, we employ a microservice and blockchain-based architecture to build the proposed system, as shown in Figure 6.1.

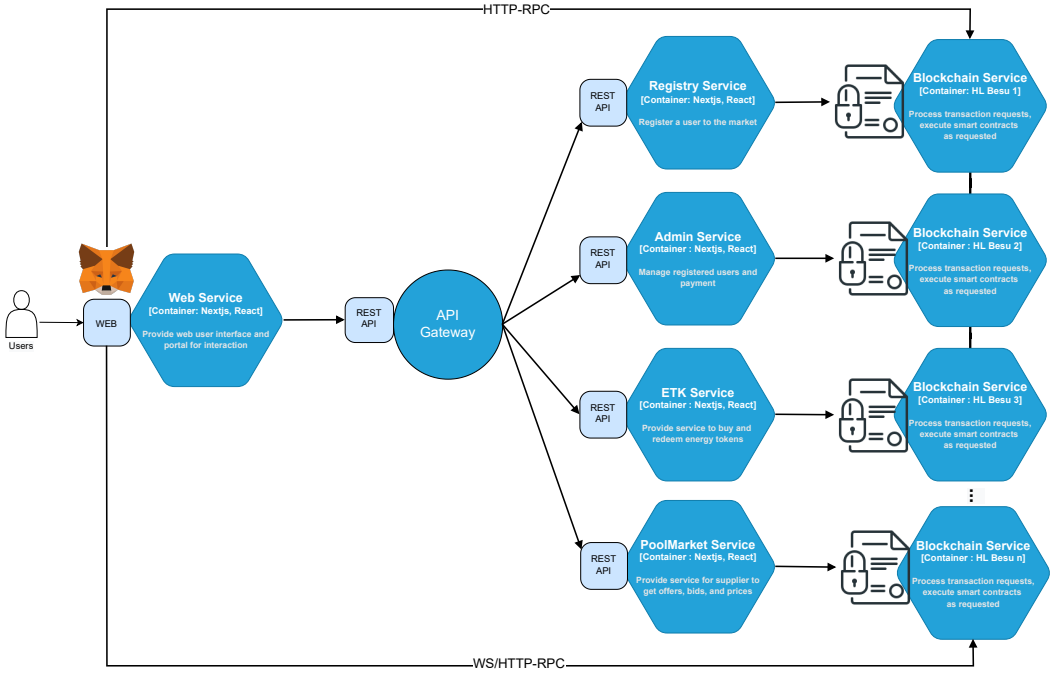


Figure 6.1: BPET architecture.

We employ the separation of concerns (SoC) design principle to construct our system by separating the front-end and back-end systems. In our proposed unified BPET architecture, all data is stored on-chain, and users interact with the back-end system through a front-end web application with a wallet plugin (e.g., MetaMask) installed. The wallet is in charge of managing accounts and signing transactions. Thus, the wallet must first connect to the blockchain network through the HTTP-RPC protocol. Then, the front-end web communicates with the blockchain in two different ways depending on the operation types, read or write. When users want to write data to the blockchain and change blockchain states, i.e., on-chain data write, the web application directly sends corresponding transactions to the blockchain through the WebSocket or HTTP RPC protocol. When users want to read data from the blockchain or query the state, the web service/application calls REST API through the API gateway, aggregating all microservices interacting with the blockchain.

In summary, the proposed system can be described in four layers:

1. Data storage: all data is stored in a performant private Hyperledger Besu blockchain running the proof of authority consensus, i.e., QBFT.

2. Architecture: front-end web is separated from the back-end system. All write operations are directly sent to the blockchain through smart contracts and the WebSocket or HTTP RPC protocol. All read operations are designed to call REST API through the API gateway. We adopt a microservice architecture to construct the back-end system.

3. Consensus: proof of authority is used as the underlying consensus algorithm, where the authority comes from different types of traditional market participants with good reputations or facilities such as distribution system operator (DSO), transmission system operator (TSO), market operator (MO), regulators and utility companies.

4. Application: a decentralized application is built on top of the blockchain, which is secure, tamper-proof, scalable and extensible.

Table 6.3 lists the brief comparison between our work and the previous similar solutions.

Table 6.3: A comparison of our solution with literature.

Items	**UBETA [1]**	**HFPET [80]**	**BPET(our work)**
Network	HL Besu (Eth)	HL Fabric	HL Besu (Eth)
Consensus	IBFT	Raft	IBFT, QBFT, Clique
SCs	PM[a], BiM[b], BaM[c], PS[d], RC[e]	AO[f], RIET[g], SIET[h]	PM, Payment, RC, ETK[i]
Benchmark tool	Customized Java client	Caliper 0.30	Caliper 0.42
Parameters	consensus	DB, endorse policy, batch size/timeout, tx rate, clients #	Consensus, network/node size, transaction type, send rate, LB
Metrics	read/write latency & throughput	Throughput, latency, execution time, CPU/RAM	SC performance, system resources, scalability
Implementation	No	No	Yes
Energy Token	No	No	Yes
Microservice	No	No	Yes

[a] PoolMarket, [b] BilateralMarket, [c] BalancingMarket, [d] PaymentSettlement, [e] RegisterContract

[f] AccountOpening, [g] Receiver-initiated energy trading, [h] Sender-initiated energy trading, [i] EnergyToken

6.3 System Design

In this section, we introduce more details on the system design of our proposed blockchain-based peer-to-peer energy trading solution.

6.3.1 Energy Trading Models

Because of various economic, policy, and social regulations, the wholesale market rule details can be different even though they all adopt the same auction mechanism, e.g., the most commonly used merit order effect. For example, the wholesale electricity markets in Australia [3] and Alberta [4] in Canada both employ the merit order effect to decide the pool price in a trading interval. However, their pool market rules use the merit order in different ways in reality. The former strictly follows the merit order to find a balance point with the total trading volume and pool price, or called market clearance price (MCP), in which the sellers with a higher bidding price and the buyers with a lower bidding price than the pool price fail to bid energy in this market. So, the Australian Energy Market Operator (AEMO) employs other markets (e.g., bilateral and balancing markets) to meet the needs of those bidding losers in the pool market. While in Alberta, Canada, sellers must supply enough energy to meet the Alberta internal load (AIL) demand at any time. The Alberta Electricity System Operator(AESO) uses the merit order effect to find a system marginal price (SMP) each minute by matching the AIL to the aggregated energy supplies. Then, AESO calculates the hourly pool price using a weighted average approach. Figure 6.2 shows the demand-supply curves of Australia (Figure 6.2a) and Alberta (Figure 6.2b) for the same original sample bidding data in Table 6.4 that consists of offers and bids from 5 suppliers and 5 consumers.

In the Australian electricity pool market, both the consumers and producers compete with each other to win the auction. The pool price (i.e., MCP) is computed as the intersection point of the supply and demand curves, which present the aggregated supply or demand on the horizontal axis and the price rate at which the participants wish to sell or buy on the vertical axis. Then, the individual bids and offers are compared with the MCP as it determines the highest bidders or winners. Consumers who offer a price rate greater than the MCP will be winners, while the opposite is true for producers. As shown in Figure 6.2a, the pool price is 70 \$/MWh, and the corresponding total volume of traded energy is 1800 MWh. The winners are the users on the left side of the intersection point between the supply and the demand curves, namely suppliers $1 - 3$ and consumers $8 - 10$ in this case. Bidding losers are the users that fall on the right side

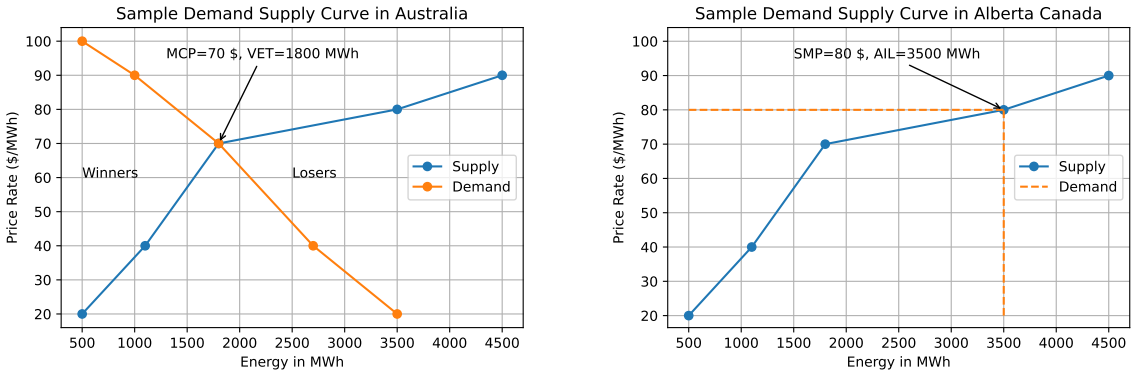
[3] https://aemo.com.au/
[4] https://www.aeso.ca/

of the intersection point between the supply and the demand curves, namely suppliers 4 - 5 and consumers 6 - 7.

In the Alberta electricity pool market, market participants who wish to buy or sell electricity submit several supply offers and demand bids to the market on a day-ahead basis for every hour, 24 hours a day. These offers and bids are sorted from the lowest to the highest price for each hour of the day into a list called a merit order. System Controllers use the merit order to balance electricity supply, starting at the lowest-priced supply offers and moving up to the highest. In this way, the AESO ensures Alberta's overall electricity needs (i.e., AIL) are met by the most competitively priced electricity. As shown in Figure 6.2b, the current system marginal price is 80 $/MWh, and the total traded energy volume is 3500 MWh. There are no bidding losers as all demands are met in this case.

Table 6.4: Original sample demand and supply data.

ID	User Type	Price($/MWh)	Amount(MWh)
1	Supplier	20	500
2	Supplier	40	600
3	Supplier	70	700
4	Supplier	80	1700
5	Supplier	90	1000
6	Consumer	20	800
7	Consumer	40	900
8	Consumer	70	800
9	Consumer	90	500
10	Consumer	100	500



(a) (b)

Figure 6.2: Sample demand-supply curves: (a) Australia (b) Alberta Canada.

Typically, the merit order mechanism consists of two steps. The first step is to calculate the aggregated demand-supply curves. Supply offers are sorted by price in ascending order, and demand bids are sorted in descending order. Then, the cumulative summation operation is applied to each list separately to get the aggregated supply and demand amounts in MWh. Both markets in Australia and Alberta involve the same calculation for the first step. The second step is determining the pool price and trading volume. This step differs from each other according to different market price mechanisms. In Australia, the intersection point of demand and supply curves is determined by matching price and aggregated energy volume. The pool price, or market clearance price (MCP) and volume of energy to be traded (VET) in a trading interval are defined as:

$$MCP = \{a | a = b \wedge ESA_a = EDA_b\} \tag{6.1}$$

$$VET = \{ESA_a | a = b \wedge ESA_a = EDA_b\} \tag{6.2}$$

where ESA_a is energy supply aggregate at price a, EDA_b is energy demand aggregate at price b.

In Alberta, the intersection point is determined by matching AIL demand and the aggregated supply, which does not consider the bidding price of buyers. The system marginal price (SMP) and VET for a single minute and the pool price (MCP) for each trading interval, i.e., one hour, are defined as:

$$SMP = \{a | ESA_a \geq AIL\} \tag{6.3}$$

$$VET = AIL = \sum_{i=1}^{m} LAB_i \tag{6.4}$$

$$MCP = \frac{\sum_{i=j}^{n} SMP_j * d_j}{60} \tag{6.5}$$

where AIL is the total Alberta internal load, LAB represents the load amount of bid i, and there are m bids at the current minute, d_j represents the duration (in minutes) of $SMP_j (1 \leq j \leq n)$, and there are n different marginal prices at current hour. Table 6.5 shows the aggregated amounts of supply and demand at each price rate for the sample data in Table 6.4. Two intersection points, Australia (70, 1800) and Alberta (80, 3500), are highlighted in bold.

Table 6.5: Merit order effect.

ID	User Type	Price($/MWh)	Aggregated Amount(MWh)
1	Supplier	20	500
2	Supplier	40	1100
3	Supplier	**70**	**1800**
4	Supplier	**80**	**3500**
5	Supplier	90	4500
10	Consumer	100	500
9	Consumer	90	1000
8	Consumer	70	**1800**
7	Consumer	40	2700
6	Consumer	20	**3500**

6.3.2 Smart Contracts Design

In this system, smart contracts play a critical role in implementing the back-end business logic, such as user registration, transaction management, market rules, and payment settlement. To make the design more general and extensible, we extract the core smart contracts and their interactions as shown in Figure 6.3.

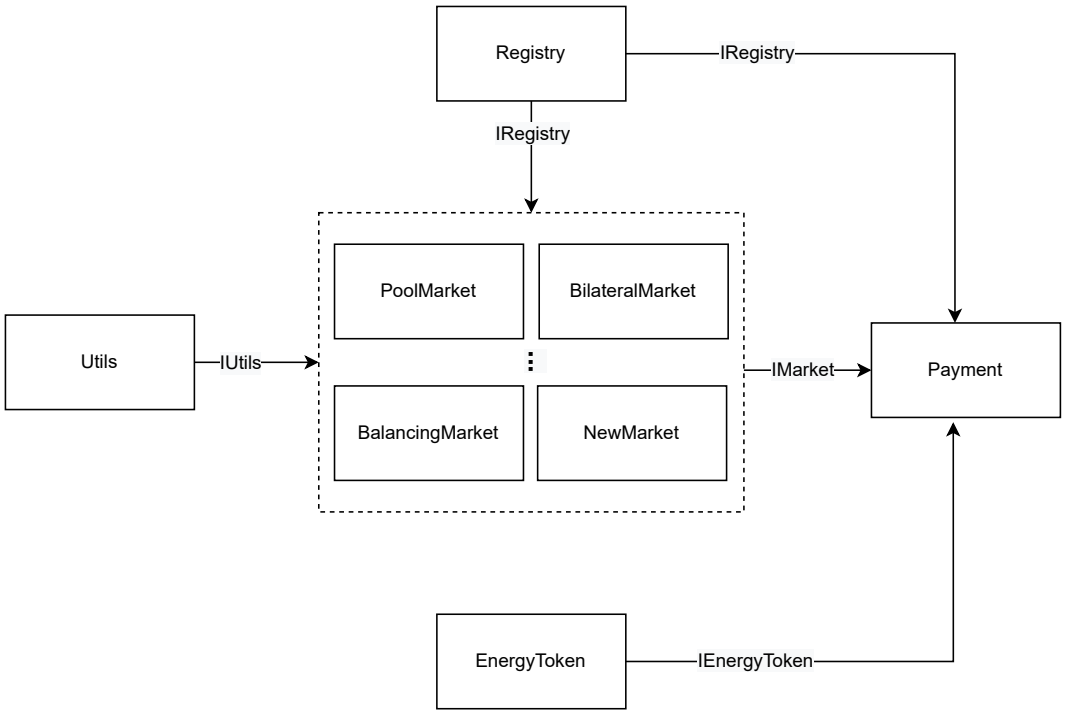


Figure 6.3: BPET smart contracts design.

Each smart contract running in blockchain provides specific functions to manage the data

storage and sharing. Meanwhile, different contracts interact with each other by exchanging information. Their functionalities and interactions are described as follows:

- Registry: provides all functions regarding user management, such as supplier/consumer registration, information updating, and deleting.

- Market: defines the market rules and records energy transactions into the ledger. It determines the basic information required to submit an offer/bid and, more importantly, how the price is calculated according to various market rules and auction models. This contract relies on the Registry smart contract to check user registration status before submission. More market smart contracts can be easily integrated into the system by properly adding adaptors to the interfaces, i.e., IUtils and IMarket.

- Utils (optional): provides all the common market contract states, events, and tooling functions such as converting data types and preparing a merit order snapshot.

- EnergyToken: provides a unified ERC20 token as the payment method to exchange energy. We design the token as one type of stablecoin similar to the widely used Tether (USDT), USD Coin (USDC), and Binance USD (BUSD) in the current cryptocurrency market. In short, one Energy Token (ETK) value is always equal to one US dollar. To achieve this stability, one can connect the ETK contract to the existing bank system or run it as a public stablecoin to reach the liquidity requirement as a functioning payment system. Either way needs further development to be ready for production.

- Payment: executes the payment process after each transaction. It relies on the final energy price rate, the total dispatched energy quantity recorded in the market contract, and the payer's registration status in the Registry smart contract.

In the above design, the market rules are the core components, which can be customized based on different trading scenarios and localized for various locations, e.g., countries, provinces, districts, or cities. For example, for the electricity wholesale market in a distribution network, a pool market rule with the merit order effect is usually adopted to determine the market price for each single trading interval; while for the peer-to-peer energy trading among end users within a microgrid or smart community, a bilateral market rule is usually necessary for market participants to negotiate the final price through either on-chain or off-chain strategies.

6.3.3 System Sequence UML

In this subsection, we introduce the sequence UML diagrams for the core modules of our proposed BPET system. These system sequence diagrams (SSD) show process interactions arranged in time sequence and depict the sequence of messages exchanged between processes involved to realize specific functionalities.

The first step for a user to start using this system is to register as a participant: a supplier, a consumer, or both. Figure 6.4 shows the UML SSD for user registration. A user inputs all required information, such as *asset identification*, *energy capacity or load*, *block amount*, and *offer control* of this asset. The front end will validate the information before sending it to the back end for further processing. If one account has been registered, the front end queries the blockchain based on the registered account address and displays all recorded information to the user. Otherwise, the front end sends a registration transaction to the underlying blockchain to store the user's input information on-chain. Then, the user gets a confirmation message after a successful registration transaction has been confirmed by blockchain validators.

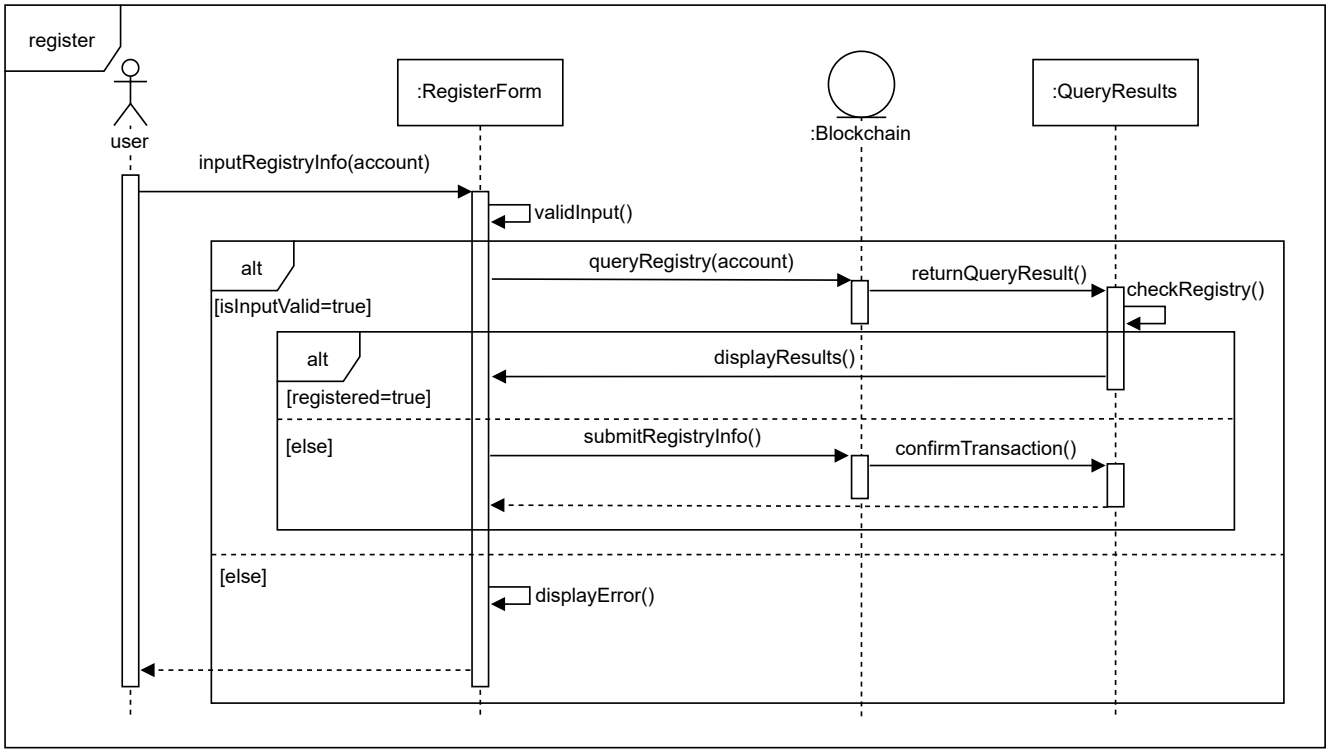


Figure 6.4: BPET registration sequence diagram.

Figure 6.5 shows the UML SSD for offer submission. First, the user inputs offer information in the front-end web form. BPET only allows registered participants to submit offers or bids to control the transaction rate and prevent denial-of-service (DoS) attacks. So, the front end

validates the input information for every offer submission by querying supplier registration data from the blockchain and checking if the account has already been registered. It also validates other information, such as if the submitted amount of energy exceeds the capacity of this asset, the price rate is legitimate, the energy block number is out of range, etc. Only after all validations pass will the front end send a transaction request to the blockchain, which records the offer data on-chain. Finally, all blockchain validators reach a consensus to confirm the transaction and send a successful confirmation message to the end user. The front end displays a corresponding error message to the user if any item fails validation.

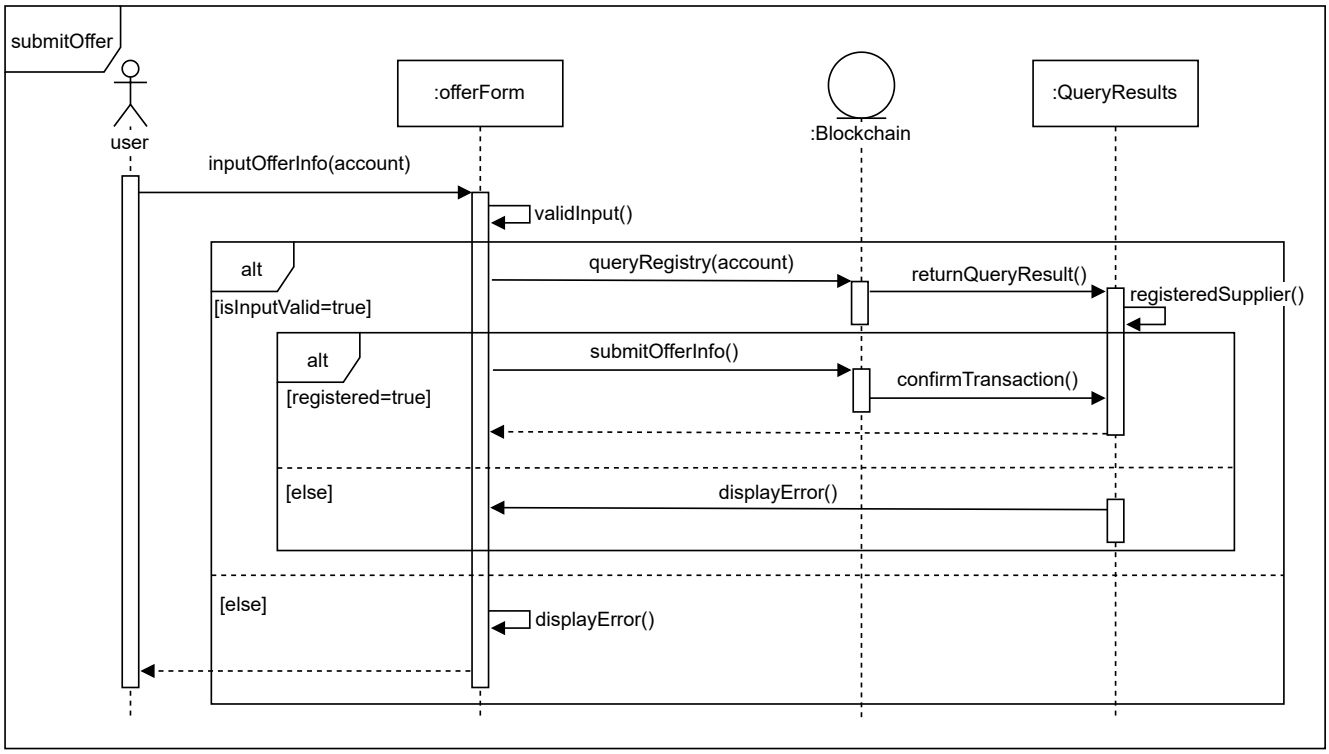


Figure 6.5: BPET submitOffer sequence diagram.

The process of submitting a bid is very similar to submitting an offer, as shown in Figure 6.6. The difference exists in the validation items, and the bid submission front end validates *account registration*, *submitted energy load*, and *price rate*. Like the offer submission process, this whole bid submission process involves both data-read and data-write from/to the blockchain. Meanwhile, it involves modules and services on multiple levels, from the front-end web and the back-end APIs to the underlying blockchain operations.

Other processes include buying or redeeming energy tokens, administrator displaying registered participants, and portal showing current and historical offers and bids. They all follow the similar sequence depicted in Figure 6.4 to Figure 6.6, in which the front end first validates the account and responds by sending a transaction or displaying an error message according to the validation

result. All account access control and value validity rules are defined in the blockchain state declarations in the smart contracts, which are stored in a decentralized way to be tamper-proof.

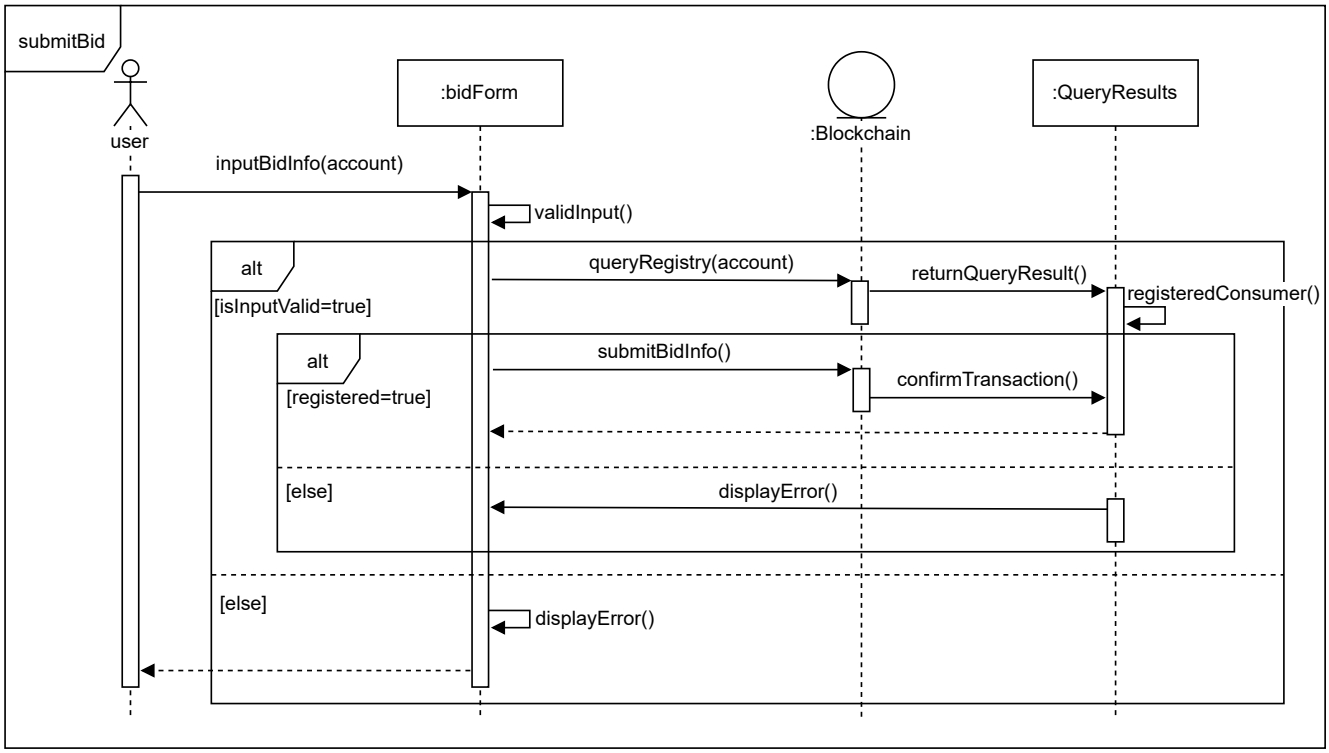


Figure 6.6: BPET submitBid sequence diagram.

6.4 System Implementation

This section introduces more technical details, such as algorithms and techniques used in system implementations.

6.4.1 Smart Contracts

A decentralized application in Web3 differs from a traditional Web2 application in where to process the business logic and how the back-end data storage is managed. Web2 applications hosted on centralized servers mainly depend on intermediaries to perform tasks where sensitive information is involved. In contrast, Web3 applications are hosted on blockchain networks and employ self-executing smart contracts to define the core business logic.

In the pool market smart contract, the core business logic determines the market clearance price, called the pool price in Alberta. Here, we take the price mechanism in the Alberta energy market as an example to design our smart contracts. According to the official guidance documentation on the AESO website, the pool price is determined in the following way. First, the market

collects all near-future electricity supplies and demands in terms of price, amount, etc. Pool market participants who wish to buy or sell electricity submit several supply offers or demand bids to the market on a day-ahead basis for every hour, 24 hours a day. These supply offers and demand bids are sorted from the lowest to the highest price for each hour of the day into a list called a merit order, which is used to balance the supply of electricity, starting at the lowest-priced supply offers and moving up to the highest. In this way, the AESO ensures the most competitively priced electricity meets Alberta's overall electricity needs.

Then, the pool price rate, i.e., the dollar cost of a megawatt hour of electricity at the end of a given hour that retailers pay to electricity generators for supplying electricity, is calculated from the merit order. Setting the pool price is highly detailed and can be summarized into two steps. The **first** step is to calculate the system marginal price (SMP), as described in Algorithm 2. The algorithm gets the latest total demand and energy supply offers as inputs every minute, and it gets a merit order of all valid offers at this minute. Then, the highest-priced supply offer is calculated by comparing the aggregated supply with the total demand in a loop, where the algorithm automatically dispatches all lower-priced offers. Here, for simplicity, we assume that there are no import offers or export demands, and all valid offers have no dispatch flexibility, which means that either none or all amount of an offered capacity is dispatched, but not partially. The highest-priced offer submitted to the market, and dispatched by the algorithm, is designated as the system marginal offer (SMO) for this minute, with its price designated as the SMP. All SMOs are stored on-chain and indexed by their timestamps in minutes. In reality, the demands across the province may change frequently (e.g., every minute or minutes), which causes fluctuations in the supply/demand curves and rapid changes to the merit order. In the system implementation, Algorithm 2 constantly monitors these fluctuations in demand and only calculates the SMP to match the supply from generators with electricity consumers when the total demands or energy offers have changed from the last minute.

The **second** step is to leverage the SMPs from the first step to calculate the pool price, as described in Algorithm 3. At the begging of each hour, this algorithm calculates the SMP for the following market clearance hour, i.e., the first SMP to calculate the next pool price at the beginning of the next hour. Then, it calculates the pool price by averaging all 60 of these one-minute SMPs in a loop. In the simplest terms, the pool price is the average of 60 one-minute system marginal prices accumulated over an hour. But, in most cases, the pool price is calculated in a weighted average manner, namely the summation of the multiplication results of each SMP and its duration minutes divided by 60 minutes, as shown in Equation (6.5). All pool prices are stored on-chain and indexed by their timestamps in hours. The SMP is posted to the front-end

Algorithm 2 System Marginal Price Calculation Algorithm

Input: $bidUpdated$, $energyOffers$ ▷ read from states
Output: systemMarginalOffer (smp at a particular minute timestamp) ▷ update to states

1: **if** $bidsNotEmpty$ & $offersNotEmpty$ **then**
2: $totalDispatched \leftarrow 0$
3: $latestTotalDemand \leftarrow getLatestTotalDemand()$ ▷ gets current total demand
4: $meritOrderOffers \leftarrow getMeritOrderSnapshot()$ ▷ gets a merit order snapshot
5: $currHour \leftarrow block.timestamp$ in hour
6: $N \leftarrow meritOrderOffers.length$
7: **for** $k \leftarrow 0$ to N **do**
8: $id \leftarrow meritOrderOffers_k$
9: $offer \leftarrow energyOffers[id]$ ▷ gets an offer
10: $amount \leftarrow offer.amount$
11: $account \leftarrow offer.supplierAccount$
12: $totalDispatched \leftarrow totalDispatched + amount$
13: $dispatchedOffers[currHour].push(DispatchedOffer(account, amount, blocktime))$
14: **if** $totalDispatched \geq latestTotalDemand$ **then** ▷ total demands are met
15: $currMinute \leftarrow block.timestamp$ in minute
16: $smpOfferIDs[currMinute] \leftarrow id$ ▷ stores system marginal offer ID
17: $smMinutes.push(currMinute)$
18: **break**

web application in real-time, and the pool price is posted after the hour's end. Then, it is used in financial settlement to calculate payments to suppliers and charges to wholesale consumers.

6.4.2 Microservices

A microservice architecture is an architectural pattern that arranges an application as a collection of loosely-coupled, fine-grained services communicating through lightweight protocols. It is designed for immense flexibility, scalability, and fault tolerance. In this architecture, services can be implemented using different programming languages, databases, hardware and software environments, depending on what fits best. In our implementation, we choose NestJS, a framework for building efficient and scalable Node.js server-side applications, and TypeScript as the programming language to develop our back-end microservices such as Registry, Admin, ETK and PoolMarket services. One of the advantages of NestJS is that it naively supports the microservice architectural style of development. A microservice is fundamentally a Nest application that uses a transport layer different from HTTP. Nest supports several built-in transport layer implementations, called transporters, e.g., TCP, Redis, MQTT, RabbitMQ and Kafka, responsible for transmitting messages between different microservice instances. For simplicity, we choose the default transporter TCP as the communication protocol. Most transporters support two types of message patterns: request-response and event-based message styles. The request-response method

Algorithm 3 Pool Price Calculation Algorithm

Input: $hour$, $energyOffers$
Output: poolPrice (pool price at a particular hour timestamp)

$H \leftarrow 3600$ ▷ 1 hour = 3600 seconds
$M \leftarrow 60$ ▷ 1 hour = 60 minutes
$cumPrice \leftarrow 0$
$N \leftarrow smMinutes.length$
calculateSMP() ▷ calculates smp for the first minute of next hour
for $k \leftarrow 0$ to N **do**
    $T \leftarrow smMinutes_k$
    if $T \geq hour$ and $T < hour + H$ **then**
        $id \leftarrow smOfferIDs_T$
        $offer \leftarrow energyOffers_{id}$
        $price \leftarrow offer.price$
        $duration \leftarrow 0$
        if $k < N - 1$ and $smMinutes_{k+1} < hour + H$ **then**
            $duration \leftarrow (smMinutes_{k+1} - smMinutes_k)/M$
        else
            $duration \leftarrow M - (smMinutes_k - hour)/M$ ▷ the last offer duration
        $cumPrice \leftarrow cumPrice + price \times duration$
$pooPrice \leftarrow cumPrice/M$ ▷ weighted average system marginal price
$poolPrices[hour] \leftarrow poolPrice$
$poolPriceHours.push(hour)$
return $poolPrice$

is ideal for exchanging messages between services. In contrast, the event-based message pattern is more suitable when a service just wants to publish events without waiting for a response.

To balance the management complexity and the service granularity, we implement several related functionalities in one microservice. The implemented microservices and functionalities of our system are listed as follows:

- **Admin Service**: *getSuppliers()* gets all registered suppliers, including basic information and allowance; *getConsumers()* gets all registered consumers, including basic information and allowance; *approve(approveRequest)* approves allowance for a spender to spend from a token owner account based on the request; *requestToken(tokenRequest)* manages to buy or burn a certain amount of administrator's tokens as requested.

- **Registry Service**: *isRegisteredSupplier(account)* checks if the provided account is a registered supplier; *isRegisteredConsumer(account)* checks if the provided account is a registered consumer; *getSupplier(account)* gets a registered supplier based on the provided account; *getConsumer(account)* gets a registered consumer based on the provided account; *getOwnerAddress()* gets the address of the contract owner; *getAllSuppliers()* gets a account

list of all registered suppliers; *getAllConsumers()* gets a account list of all registered consumers; *registerSupplier(supplierRegistryRequest)* registers a supplier as requested; *registerConsumer(consumerRegistryRequest)* registers a consumer as requested.

- **PoolMarket Service**: *getsmp* calculates and returns the SMP for the current minute; *getProjectedPoolPrice()* calculates and returns the projected pool price based on the SMP list of the current hour; *getValidOffers()* gets all valid offers; *getValidBids()* gets all valid bids; *getDispatchedOffers()* gets all dispatched offers at current hour; *getTotalDemandMinutes()* gets a list of minute timestamps at which the total demand has changed; *getMarginalOffer(timestamp)* gets the marginal offer based on the given timestamp; *getTotalDemand(timestamp)* gets the total demand based on the given timestamp; *getMinMaxPrices()* gets the minimum and maximum allowed prices for offers/bids.

- **ETK Service**: *getETCOwnerAddress()* gets the owner account of the ETK contract; *getBalance(account)* gets the token balance of the given account; *allowance(owner, spender)* gets the allowance of given owner account to spender account.

- **Gateway Service**: This gateway acts as a client proxy to aggregate all functionalities of the above microservices and provide a unified REST API portal to the front end or users. This gateway can be integrated with some OpenAPI platforms, such as Swagger, to provide a GUI-enabled API service.

Given all developed microservices, we containerize each using Docker and deploy the whole back-end system on the server using Docker compose. Each service serves on a dedicated port number: Gateway 3000, Admin 3001, Registry 3002, PoolMarket 3003, and ETK 3004. For better scalability in an extensive application, we can also deploy the microservices using Kubernetes, which manages containers from a large number of container runtimes.

6.4.3 Web Application

We implement the front end using the popular open-source web development framework Next.js, which enables React-based web applications with server-side rendering and generating static websites. We use JavaScript as the programming language to develop the processing logic of all web pages, where Material UI (MUI), an open-source React component library implementing Google's Material Design, is employed to develop the user interface components such as button, select form, dialogue, text field, and data grid etc.

To make the front end a Web3 application, we integrate a Web3 (or Ethereum) provider solution for all Wallets such as MetaMask, Brave Wallet, Dapper, Frame, Gnosis Safe, Tally, Web3 Browsers, and WalletConnect etc. MetaMask is installed as a plugin to the browser and connected to the blockchain network through the HTTP-RPC protocol. Meanwhile, the Ethers.js library interacts with smart contracts on the blockchain through the WebSocket-RPC or HTTP-RPC protocol.

Unlike the read operations, a write operation to blockchain needs to be signed first by a signer with the private key of a certain account. It consumes Gas from the transaction sender account, as of transaction fees to execute the smart contracts. In addition, many transactions such as *submitOffer* and *submitBid* require the account address of the transaction sender as the input to complete this operation successfully. So, it is quite challenging to deploy these functionalities of sending transactions as a microservice, like query operations. Even though we achieve this by signing the transaction using the MetaMask wallet in the front end and then sending the signed transaction to the back end for further processing, it will compromise the performance and significantly impact user experience. Therefore, we implement all write transactions in the front end to enable them to interact with the blockchain directly.

6.5 Case Study

In this section, we take the case of the electricity wholesale market in Alberta to study the effectiveness and efficiency of our proposed BPET system. As stated on its website, a fundamental principle of Alberta's electricity system is that supply (electricity produced by generators) and demand (electricity consumed) must be perfectly matched at all times. An energy management system continually collects data from every generator connected to the transmission system, enabling System Controllers to match the supply of electricity with demand and monitor the health of the provincial electric system.

When switching the management process from the traditional operators to a novel blockchain-based system, i.e., BPET, the fundamental principle remains while some detailed rules may be simplified. To this end, we simplify a complete electricity trading system's workflow into four main stages: initiation, buying ETK, trading, and payment, as shown in Figure 6.7.

In the initiation stage, we register all suppliers and consumers to the system. According to local policies and laws, different markets may require different information to register a participant. For example, in some countries with a strict privacy protection policy, such as Germany, it is hard to require sensitive personal information such as name and ID number. However, some basic

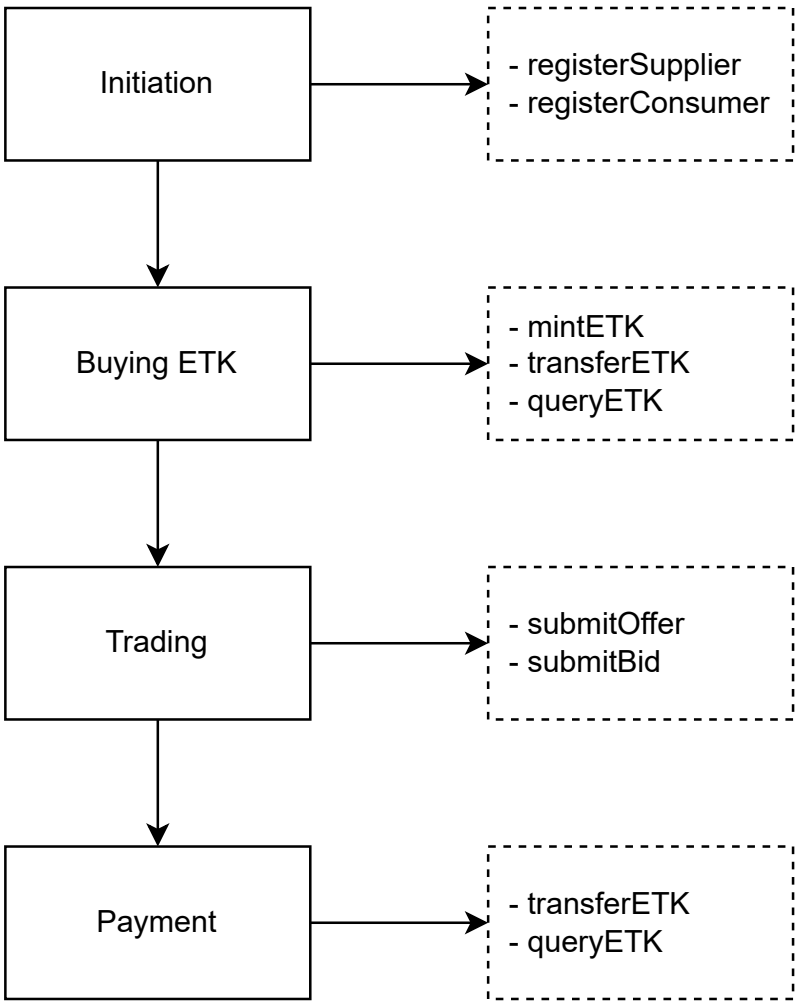


Figure 6.7: BPET workflow.

information, including generation capacity (for a supplier or seller), load (for a consumer or buyer), and a unique asset identity, is necessary to finish the rest of the trading steps.

In the buying ETK stage, market participants first send a request to buy ETK tokens. Then, the token contract deployer or owner with minting authority mints ETK tokens and transfers them to participants as requested.

In the trading stage, suppliers submit offers, and consumers submit bids every minute according to their needs. Since the policy requires generators to provide stable electricity supplies, the electricity supplies are expected not to fluctuate much, implying that suppliers will not frequently submit or update offers after the initial submissions. However, electricity demands usually change rapidly, even in a short period, reflecting a high volume of bid submissions in each trading interval. Either offer or bid submissions will break the old balance of supply and demand, causing new pool price calculations.

In the payment stage, buyers transfer a certain amount of ETK tokens to sellers. The token

amount is calculated as the market clearance price (pool price) multiplied by the actual consumption amount recorded by a smart meter.

6.5.1 Historical Data Analysis

AESO is a market-based platform that has been running for decades, with many historical and real-time data sets produced and publicly available such as merit order, pool price, and marginal price data sets. To keep the seasonality property of the real data, we select a whole year's historical data of merit orders and marginal prices from September 2021 to August 2022 to analyze and try to answer the following questions:

- How many participants in total need to be registered in the initiation stage?
- How many energy tokens need to be minted and transferred to participants?
- How often do the suppliers submit offers? Or what is the estimated offer submission TPS?
- How often do the consumers submit bids? Or what is the estimated bid submission TPS?

The historical merit order and marginal price data sets are downloaded from the AESO website. From the historical merit order data set, we find 201 unique assets: energy suppliers or producers. Unfortunately, no consumers are recorded in this data set. Since electricity retailers compose the main body of consumers in a wholesale market, we estimate the number of consumers as the number of retailers. There is a total of 60 companies [5] providing electricity services to end users in Alberta. In the initiation stage, all market participants must register their information to the BPET, which verifies the account and authenticates users with related operation rights. In the case of the Alberta market, there are 261 participants registered at this stage. Only registered users can participate in any market activities.

After registration, participants may buy energy tokens for further trading. Energy token (ETK) is a type of stablecoin which pegs its market value to some external reference such as US dollars or the amount of electricity generated. Some well-known stablecoins pegged to US dollars, such as USDT and USDC, are in the marketplace. In our implementations, we only deployed a simple ERC20 smart contract to prototype the energy token, which is treated as a stablecoin. We discuss more details about stablecoins in Chapter 7. We assume every participant initially buys a certain amount of ETKs after registration. Therefore, there were 261 token transfer transactions before trading.

[5] https://ucahelps.alberta.ca/retailers.aspx

In the trading stage, participants submit several supply offers and demand bids to the market on a day-ahead basis for every hour, 24 hours a day, respectively. We analyze the merit order data set and find that actual offers do not have frequent updates after initial submissions. Here, we define an offer as updated when it does not appear in the previous hour's offer list. The comparison factors are listed in a tuple $(AssetId, BlockNumber, Price, From, To, Size, AvailableMW)$. For simplicity, we do not resubmit an offer if it does not change in the next hour. Therefore, we can approximate the offer submission rate as the offer update rate, which also applies to bid submission. Throughout the whole year, the most frequent offer updates happen at the first hour of the day, with a maximum of 245, minimum of 52, and mean of 106. Other daily hours have a much lower mean transaction request rate, as shown in Table 6.6.

Each time the total demand changes, the market calculates a system marginal price (SMP). Thus, we estimate the demand update rate as the number of SMP calculations each hour. It is worth noting that the total demand changes are measured in a large electricity unit scale, i.e., MWh. Analyzing the historical SMP data set from AESO, we find that all hours have a close demand update rate, with a maximum of 17, a minimum of 1, and a mean of around 4. This indicates that the market system calculates a maximum of 17 SMPs in an hour.

A basic statistical analysis of the historical data outlines the performance requirements for our proposed system. In an extreme situation where all the supply and demand updates happen in the exact second, the system needs to process a large number of transactions quickly, resulting in 245 transactions per second (TPS) of offer submission and 17 TPS of bid submission. In the next section, we intensively evaluate the implemented system using AESO's data sets to see how effectively it can support the current Alberta wholesale electricity trading business.

6.5.2 System Evaluation

We leveraged Hyperledger Caliper v0.4.2 with Ethereum SDK v1.4 as the benchmark tool to evaluate our blockchain system. In each test, Caliper was running in a container on a client VM with rich resources (e.g., 16vCPU and 60GB RAM). The results were stored in performance reports and then processed using Pandas, Numpy, and Matplotlib.

Experimental Setup

We performed our experiments on the same OpenStack-based cloud environment as in Section 5.3.1. All tests ran in the baseline configuration, i.e., "8-LB-2C7.5G-QBFT-1S", which had 8 nodes each with 2 vCPU, 7.5GB RAM, and 36GB SSD. All requests were balanced through an Nginx load balancer on the client. The blockchain network ran in the QBFT proof of authority

Table 6.6: Statistics of hourly supply and demand updates in Alberta.

Hour	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Supply																								
mean	106	29	26	27	26	33	31	73	43	41	41	41	40	38	36	35	37	35	34	37	38	39	37	68
std	53	15	10	14	10	10	10	13	12	13	12	12	12	11	11	11	10	9	9	10	10	15	14	12
min	52	3	1	4	5	7	10	26	19	10	16	12	7	8	10	7	16	14	11	10	11	15	9	28
25%	76	21	19	19	19	26	24	65	33	30	33	33	32	31	29	27	30	29	28	31	31	31	27	60
50%	87	27	25	25	25	33	30	72	41	39	39	39	38	36	35	33	36	35	33	35	36	38	34	68
75%	98	35	33	32	32	39	36	80	51	48	47	48	46	44	42	40	43	40	39	42	42	45	44	75
max	245	225	60	216	69	62	77	110	82	87	81	84	80	71	75	85	73	69	64	72	73	216	82	104
Demand																								
mean	4	4	4	3	3	4	5	5	4	4	4	4	4	4	4	4	5	4	5	4	4	4	5	5
std	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
min	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25%	3	2	2	2	2	2	3	3	3	3	3	2	2	2	2	2	3	3	3	3	2	3	3	3
50%	4	3	3	3	3	3	4	4	4	4	4	3	4	4	3	4	4	4	4	4	4	4	4	5
75%	5	5	4	4	4	5	6	6	6	5	5	5	5	5	5	5	6	6	6	5	5	6	6	6
max	13	13	10	10	16	12	15	15	13	12	11	17	11	13	11	14	12	17	13	13	13	13	12	13

consensus with one second of block time.

We kept all blockchain parameters the same as in Chapter 5, where"gasLimit" was "0x1fffffffffffff", "contractSizeLimit" was 2147483647, the node startup option "min-gas-price" was 0, the difficulty was "0x1", the transaction pool queue size was 4096, and all nodes were configured as validators.

Workloads

Table 6.7 shows an overview of the workloads used in our experimental studies. As can be seen, we have experimented on all the core tasks except for minting tokens in an energy trading system as depicted in Figure 6.7. In actual energy trading, the administrator can mint a large number of tokens at the initial setup. In subsequent steps, these tokens will be transferred to buyers and sellers through energy transactions. In addition, from the system's perspective, token minting is an asynchronous task which does not require high performance. The performance of minting normal ERC20 tokens is assumed to be adequate for our application. Therefore, we did not benchmark *mintETK* transactions throughout the experimental study.

To comprehensively benchmark the smart contracts listed in Table 6.7, we selected their core public functions to generate workloads. Workload modules were customized by creating a specific JavaScript file and defining transaction workloads with necessary parameters to make Caliper fit the energy trading smart contracts. The transaction rate for each operation varied from 10 TPS to 80 TPS. For each type of transaction, we set up the total number of transactions to be the same as the number of workers, which will be sent at a fixed rate in a single test and result in one transaction per worker. We ran five replicas for a single test under a specific transaction rate to

Table 6.7: Overview of workloads used to benchmark the BPET smart contracts.

Contracts	**Operations**	**Send Rates**	**txNum**
EnergyToken.sol	transfer	10 → 80 TPS	1 tx/worker
	query		
PoolMarket.sol	submitOffer		
	submitBid		
Registry.sol	registerSupplier		
	registerConsumer		

generate five performance reports. After deploying all smart contracts to the Hyperledger Besu network, we ran tests in the order of Registry, EnergyToken, and PoolMarket.

In the *Registry* contract, *registerSupplier* and *registerConsumer* perform write operations by storing the provided registration information of suppliers and consumers, respectively, to the blockchain. The system requires every account to be unique by checking the registered record to avoid duplicated registration. So, we chose the "fromAddressSeed" approach to generate unique test accounts for each test in the Caliper network configuration. This approach uses a fixed seed to derive test addresses via BIP-44 key derivation, which can derive a new set of accounts when we manually modify the seed as needed. For example, we simply increased the last digit of the seed by one when switching from 40 TPS to 50 TPS in testing *registerSupplier*. Then, the workload module in the Caliper generated 50 unique test accounts different from the 40 old ones in the last test. The generated 50 unique test accounts were reused in the subsequent offer/bid submission and transfer tests by setting the seed back to the original value. In short, we need a unique seed for each transaction rate.

In the *EnergyToken* contract, *transfer* performs an ERC20 token transfer transaction defined by the OpenZeppelin [6], which is the premier crypto cybersecurity technology and services company founded in 2015. Before tests, the administrator minted 1,000 tokens for each account. Then, in each test, these accounts with 1,000 tokens transferred one token to the pre-defined account. *Query* performs a single read operation by calling the *balanceOf* function.

In the *PoolMarket* contract, *submitOffer* and *submitBid* perform data-write operations by storing the provided offer and bid information, respectively, to the blockchain. These two tasks played a vital role in the whole energy trading process by recording the user's original data. Besides checking basic requirements, e.g., if the sender is a registered user, the amount and price are valid etc., they both contain a loop over the list of identifications to check if each submitted offer/bid already exists on-chain. If it exists, the contract updates the corresponding offer/bid information

[6]https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/ERC20.sol

but keeps the *offerId* or *bidId* unchanged. Otherwise, the contract updates the offer/bid state in the mapping and pushes a new *offerId* or *bidId* to the identification list.

Performance Metrics

In the experimental study, we evaluated two basic performance metrics, Throughput and Latency. They are defined as follows:

- **Throughput** is the number of successfully committed transactions or query operations per second. The transaction throughput is expressed as transactions per second (TPS) at a network size. Query throughput is expressed as transactions per second (TPS) from a single node.

- **Latency** is the amount of time taken for a transaction's effect to be usable across the network. The measurement includes the propagation time and the consensus time. Query latency is the time between when the read request is submitted and when the reply is received.

6.5.3 Evaluation Results

Figure 6.8 shows the benchmark results of the *Registry* contract under the Besu baseline configuration. We observe that the throughput of both registration operations increases near-linearly and then drops to a steady level of around 40 TPS as the transaction send rate increases from 10 TPS to 80 TPS. The throughput of registering consumers is slightly higher (around 10% in most cases) than registering suppliers, which is expected because in the smart contract *registerSupplier* stores one more parameter *blockAmount* than *registerConsumer*, and it includes an additional step of updating the total supply capacity for each registered supplier. Accordingly, the latency of *registerConsumer* is slightly lower than that of *registerSupplier*. But both remain at a low level of around 0.75 seconds.

It is worth noting that each worker only sent one registration transaction in the tests, miming the production scenario where producers and consumers only register themselves. Our proposed BPET system only takes 6-7 seconds for the Alberta energy market to register all 201 suppliers and 60 consumers, as analyzed in Section 6.5.1. The experimental results show that users will experience a latency of less than one second.

Figure 6.9 shows the benchmark results of the *EnergyToken* contract under the Besu baseline configuration. As we can see, the query throughput dramatically decreases as the transaction send rate increases from 10 TPS to 80 TPS. The blockchain has a relatively high throughput of

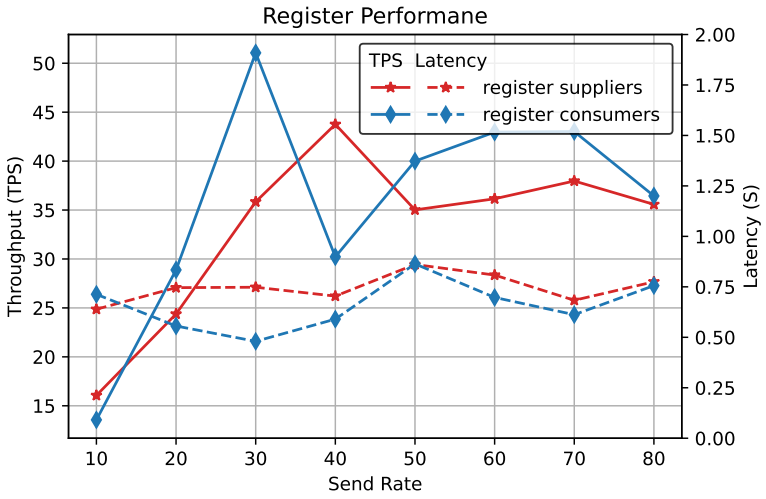


Figure 6.8: Registry contract performance under the Besu baseline configuration. Solid lines indicate throughput; dashed lines represent latency.

over 250 TPS under a low query request rate of 10 TPS. When the query request rate increases to an extent, e.g., 80 TPS, network congestion happens, causing a significant performance drop with a throughput of 50 TPS. However, the query latency always maintains at a negligible level. The transfer throughput increases to around 48 TPS when the send rate reaches 40 TPS, then fluctuates between 30 TPS and 50 TPS. The transfer latency is relatively low and steady, around 0.7 seconds. The observed experimental results indicate that up to 50 users can send transfer transactions to pay for their energy simultaneously and will get a response from the blockchain system within one second.

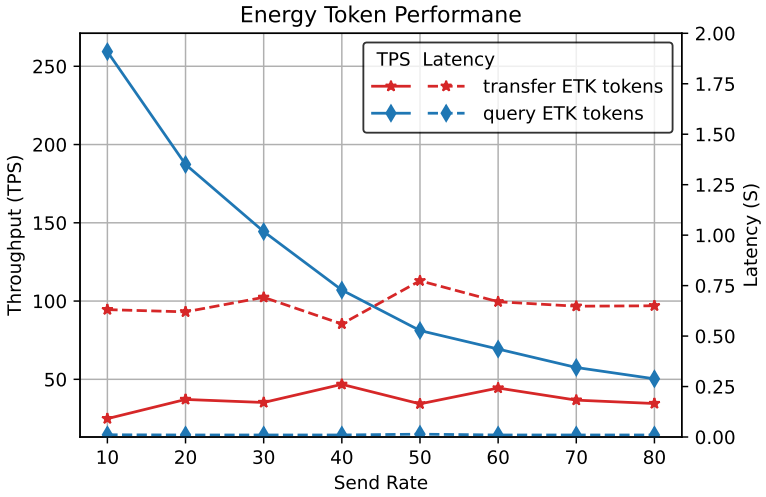


Figure 6.9: EnergyToken contract performance under the Besu baseline configuration. Solid lines indicate throughput; dashed lines represent latency.

Figure 6.10 shows the benchmark results of the *PoolMarket* contract under the Besu baseline

configuration. As can be seen, both transactions have similar performance trends. The throughput increases linearly from 10 TPS to the maximum of 43 TPS for *submitBid* and 46 TPS for *submitOffer* when the send rates reach 30 TPS and 40 TPS, respectively. Then, the throughput drops significantly, followed by a slow and slight increase when the send rate increases to 80 TPS. Accordingly, the latency of both transactions drops from 0.75 seconds to 0.5 and 0.6 seconds for *submitBid* and *submitOffer*, respectively, when the send rate reaches 40 TPS. Then, it increases to around one second as the send rate increases to 80 TPS.

The observed experimental results indicate that the blockchain system can process up to 43 *submitBid* or 46 *submitOffer* transactions per second. In other words, if 40 users submit their offers or bids simultaneously, they will get a successful response from the blockchain system within one second. In the Alberta energy market, with a maximum of 245, a minimum of 52, and a mean of 106 offer updates in the first daily hour, our proposed BPET system takes up to 6 seconds to process all the offer submission transactions.

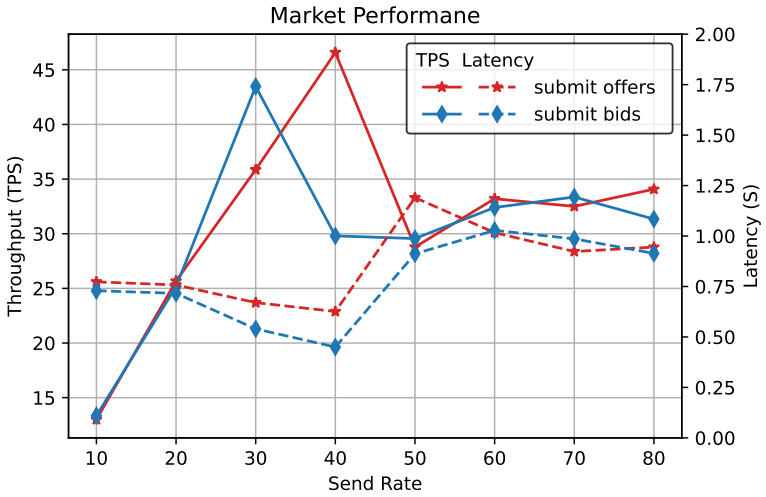


Figure 6.10: PoolMarket contract performance under the Besu baseline configuration. Solid lines indicate throughput; dashed lines represent latency.

6.5.4 Summary

Blockchain technology has drawn a lot of attention to resolve the security, reliability, and efficiency issues in managing decentralized energy resources. However, the system design challenges, implementation difficulties, and blockchain performance have not been studied from the engineering perspective. In this study, we first proposed a unified architecture of a blockchain-based peer-to-peer energy trading system called BPET, combining it with a microservice design. Then, we designed and implemented a prototype of BPET by developing four smart contracts, deploying

them to a private Hyperledger Besu blockchain network, and building the backend using NestJs and a web application frontend using NextJs, to demonstrate the feasibility of our proposed solution. We also conducted a case study using the real data set from the AESO and evaluated the performance of smart contracts using the Hyperledger Caliper benchmarking tool. Our experimental results indicate that the implemented system can effectively and efficiently process energy trading transactions in the Alberta energy market.

7 Discussions

This section will discuss challenges and potential solutions concerning blockchain performance and blockchain-based P2P-ET. These topics include blockchain scalability, consensus algorithms, and security. We also discuss the fairness issues related to market rules design in energy trading. From the perspective of web3 applications, we discuss stablecoins and why we use stablecoins in energy trading applications.

7.1 Scalability

The system scalability of a DApp depends on high-level system architecture and the scalability of the underlying blockchain network. The microservice architecture in our design provides a scalable infrastructure to meet any fast-growing business performance requirements. One can easily scale the microservices using containerization platforms like Kubernetes and Docker Swarm. In contrast, the scalability of the Hyperledger Besu blockchain network with the QBFT consensus mechanism deserves more discussion. Blockchain scalability is usually the shortest wood in a decentralized application "barrel" due to the blockchain trilemma problem. Similar to the CAP theory [50] in a traditional distributed system, the blockchain trilemma points out that it is hard for blockchains to achieve optimal levels of all three properties, including decentralization, security, and scalability simultaneously. For example, adding a coordinator to the IOTA or shortening the block interval of Bitcoin can significantly increase the transaction throughput. But, the former loses the decentralization, and the latter affects the security due to the increasing probability of fork. Therefore, blockchain scalability solutions must consider a good balance between performance and the other properties, i.e., security and decentralization.

7.1.1 Blockchain Sharding

The mainstream blockchain scalability solutions can be divided into two main categories: Layer1 on-chain and Layer2 off-chain [141]. Layer1 focuses on the network, consensus, and data structure to improve the efficiency of on-chain execution using DAG (e.g., IOTA [103], Byteball [26],

Nano [72], Dagcoin [74], and Conflux [75]), sharding (e.g., Elastico [83], OmniLedger [65], RapidChain [134], and TEE-based [28] solution), and performant consensus (e.g., Bitcoin-NG [33], Algorand [49], and Snow white [15]). In contrast, Layer2 seeks to scale out the blockchain using off-chain methods such as off-chain channels (e.g., Lightning Network [102], Sprites [87], and Raiden Network [93]), side-chain (e.g., Plasma [101] and Pegged Sidechain [11]), and cross-chain protocols (e.g., Cosmos [70] and Polkadot [130]).

However, the off-chain solutions are more subject to forks, and the transactions in the DAG layout are not organized in a chain structure and a total block/transaction order. Sharding schemes are among the most effective solutions for overcoming scalability problems at the fundamental consensus level. The basic idea of sharding is "divide-and-conquer", which divides a blockchain network into several smaller networks, called shards, to parallelly process transactions and boost the system concurrency. Compared to sharding in a classic database system with crash-failure models, sharding a decentralized blockchain network with a Byzantine failure model is more challenging. At first, all participating nodes, including the malicious ones, should be dynamically allocated to committees in a randomized process, aiming to prevent an adversary from concentrating its presence on one committee and conquering that committee when exceeding the Byzantine tolerance threshold. It is a known hard problem to generate a random number in a distributed manner. Second, the committee size must be carefully designed to ensure the probability of a faulty committee is negligible in all circumstances. Third, some transactions, called cross-shard transactions, with their inputs and outputs allocated in different shards, should be executed by other shards. Those cross-shard transactions must be carefully handled to achieve global consistency among different shards. At last, a shared blockchain system should periodically reconfigure the partitions carefully to guarantee the system's security. Examples of epoch reconfiguration strategies are hash + Final Committee, DRG + PoW + Cuckoo Rule, and VRF + Global Reconfiguration [125].

7.1.2 Rollups

Another strategy for scaling blockchain, e.g., Ethereum, is to build the second chain (Layer 2) and move computation and state storage off-chain. Rollups are a popular development of this strategy, which performs calculations off-chain, rolls many transactions up into a single batch, and sends it to the Ethereum Mainnet or its compatible chains (Layer 1) in a single action. Instead of submitting each transaction separately, rollup operators submit a summary of the required changes representing all transactions in a batch.

To be able to work on a rollup, funds need to be locked on a smart contract on the Layer 1 blockchain. This allows transactions to be processed without the overhead of all the data

associated with performing a transaction on the main chain. Rollups scale the Ethereum blockchain by significantly decreasing associated transaction processing times and gas fees and increasing the system throughput simultaneously. Currently, there are two major types of rollups used for scaling Ethereum: Zero-Knowledge Rollups (ZK Rollups), e.g., zkSync, StarkNet, loopring, and Mina Protocol etc, and Optimistic Rollups, e.g., Arbitrum, Optimism, Metis Andromeda, and Boba Network etc. The main difference between ZK and Optimistic rollups lies in how to finalize this batch of transactions.

In ZK rollups, the Ethereum network verifies the correctness of the batch of transactions. After verification, the batch of transactions is considered final, like any other Ethereum transaction. This is achieved through cryptographic validity proofs (commonly called zero-knowledge proofs). With any batch of off-chain transactions, the ZK rollup operator generates a validity proof for this batch. Once the proof is generated, it is submitted to Ethereum to verify and finalize the roll-up batch. In zkSync, this is achieved via a SNARK, succinct non-interactive argument of knowledge. In StarkNet, this is done via another type of cryptographic proof technology STARK, scalable transparent argument of knowledge.

Optimistic rollups, on the other hand, have no mechanism to prove the validity of the off-chain transactions. Instead, they are considered "optimistic" because they assume off-chain transactions are valid unless proven otherwise. Hence, they rely on fraud proofs, a challenge to the submitted state to Ethereum. If such a challenge is submitted, the Optimistic rollup operator must show that the state and transactions in question are valid. This cumbersome process requires watchers to ensure that the Optimistic rollup operator is always honest.

7.1.3 BPET Scalability

Based on the experimental results in Chapter 5, a private Hyperledger Besu blockchain network with QBFT consensus can scale up to 14 validators without noticeable performance loss. Other performance factors include block frequency, block size, workload type, node configuration, and consensus algorithms. For a local or provincial energy market with a limited number of authorities, 14 validators are sufficient to form a small-scale blockchain network and run a BFT-based consensus. More blockchain participants can join the network by running a normal Besu node to receive remote API calls, submit transactions, and store the ledger.

There are at least three potential methods to improve the performance of the BPET system. First, validators add more resources, e.g., CPU, and RAM, to their blockchain nodes and increase the network speed. This method works from the perspective of the vertical scalability of the Hyperledger Besu network. However, adding more resources to nodes can only increase the scalability

to a very limited level. Second, the smart contract developer optimizes the registration and energy trading contracts, especially the loop operations, to improve the on-chain execution efficiency. This step is based on the fact that workload significantly influences blockchain performance. For example, as observed from the comparative benchmark results in Chapter 5 and Chapter 6, the throughput of "transfer" in the Simple contract is much higher than in the Registry smart contract because the latter has more complex calculation steps, e.g., checking zero address and requiring sufficient balance. Third, smart contracts offload as many on-chain operations as possible to off-chain microservices. Such operations include registering participants, calculating the system marginal price, and calculating the hourly pool price. This will significantly improve the specific operation's performance but loses decentralization.

7.2 Consensus

According to a systematic survey study on 140 blockchain initiatives in the energy sector being pursued by a large number of companies, startups and research institutions in 2019 [9], the mostly used blockchain consensus algorithms in the energy sector are PoW (e.g., Ethash), PoA (e.g., Clique, QBFT), and PBFT. As an alternative to PoW, PoS (e.g., Tendermint, Casper) is becoming increasingly popular. The underlying consensus mechanisms are essential to determine a blockchain network's performance, security, and decentralization properties. Therefore, discussing the advantages and disadvantages of mainstream blockchain technologies in building a P2P energy trading system is meaningful.

Proof of work (PoW) is the most widely-used consensus in public blockchains such as Bitcoin. It requires consensus participants, called miners, to resolve an arbitrary mathematical puzzle, such as calculating a hash value with a specific number of leading zeros, to prevent anybody from gaming the system. The first puzzle resolver wins the current round of competition and has the right to form a block and consequently get rewards. The puzzle's complexity increases when more miners participate in the block-forming contest. PoW is a relatively decentralized and secure consensus, primarily in an open public blockchain network. However, the main criticism of PoW is that it consumes too much electricity to run the consensus, which is opposite to the ultimate goal of a blockchain-based P2P energy trading system.

As an alternative to PoW, proof of stake (PoS) was proposed to solve the power consumption problem. In PoW, miners buy hardware equipment and power to compete for generating blocks. In PoS, validators stake their digital assets, e.g., cryptocurrency and tokens, to be selected for creating the next block in the random validator selection process. This change improves the

blockchain's scalability, flexibility, and performance and saves a lot of energy. However, PoS has the potential for centralization because it relies on delegates chosen to validate transactions, and it's always possible for larger nodes to overpower smaller ones, which prevents smaller ones from participating, eventually making the PoS less decentralized. Another problem of PoS is "Nothing at Stake", in which validators can effectively break safety by voting for multiple conflicting blocks at a given block height without incurring a cost for doing so. To resolve this problem, the major PoS consensus protocols, e.g., Casper the Friendly Ghost, a chain-based PoS design led by Vlad Zamfir, Tendermint, a BFT-based PoS design led by Jae Kwon, and Casper the Friendly Finality Gadget, a hybrid of the chain-based PoS and BFT-based PoS led by Vitalik Buterin, employ a punishment mechanism called slashing, whereby other network participants forcibly eject an offending validator while continuously draining their balance. As a relatively new type of consensus, PoS needs time to prove its ability to tolerate Byzantine faults and tackle cartel formation centralization.

Proof of authority (PoA) is a modified form of PoS where a validator's identity performs the role of stake instead of a stake with a monetary value. Identity here refers to the certainty that validators are who they claim they are, as shown by the correspondence between their personal identity on the platform and any officially issued documents presented by them. It should be difficult and equal for anyone to obtain the eligibility for staking identity so that the right to be a validator becomes earned, valued, and unpleasant to lose. Establishing the authority should have the same procedure for all validators to ensure the PoA's fairness and integrity. The advantage of PoA is that it is more secure than PoS because the nodes are known and can be held accountable for their actions. The disadvantage is that it is less decentralized than PoS because the set of nodes is predetermined. Therefore, PoA is more suitable for use in a permissioned or private blockchain network with a certain level of trust among participants.

Another famous consensus algorithm used in permissioned blockchain is Practical Byzantine Fault Tolerance (PBFT) [22], which proves that a distributed system can reach a consensus through complex communications in a network with at most m adversaries among a total of $3m + 1$ participants. PBFT uses all-to-all messages that create $O(n^2)$ communication complexity during the normal-case leader commit phase. Thus, the naive PBFT lacks scalability due to network communication congestion as network size increases.

Our proposed BPET system adopts PoA QBFT consensus for the Hyperledger Besu blockchain network. First, blockchain participants provide their identities to be selected as validators. Then, when receiving transactions from RPC APIs, QBFT requires validators to take turns to propose blocks. QBFT employs the classic three phases commit scheme, *pre-prepare*, *prepare* and *commit*, to validate and commit a block to the chain. Each round requires all-to-all messages and over

$2N/3$ votes of all validators for each message to reach a consensus and keep it consistent. Thus, QBFT has a quadratic communication complexity which limits its network scalability. From the perspective of the CAP theorem [50], both PBFT and QBFT are consensus mechanisms preferring consistency over availability. In a wholesale market, as in our case study, validators are entities with high reputations, such as distribution system operators (DSO), transmission system operators (TSO), market operators (MO), regulators and utility companies. They have the motivation to collaborate with other validators and reserve the security of blockchain without incentives from validating transactions. Therefore, it is possible to set the gas price to zero and eliminate transaction fees in a permissioned blockchain network. However, when BPET is used in a microgrid where an individual household acts as a validator, there should be non-zero transaction fees and gas prices to incentivize validators.

7.3 Security

This section introduces common cyber attacks on a blockchain system, smart contract vulnerabilities, and how the proposed solution overcomes these security challenges. Several types of cyber attacks can be launched against a blockchain network. Some of the most common include:

1. 51% Attack: When a single entity controls more than 50% of the network's computation power or stacked cryptocurrency, a 51% attack can occur, allowing this malicious participant to manipulate which transactions are added to the blockchain and potentially tamper with previous transactions.

2. Double Spending: A malicious participant spends the same cryptocurrency multiple times by creating copies of a transaction. For example, in a PoW-based blockchain network, attackers can take advantage of the time window between a transaction's initiation and confirmation to launch an attack quickly. Before the second transaction can be mined and marked as invalid, the attacker has already obtained the output of the first transaction, which results in the double spending of the same funds.

3. Sybil Attack: Within a blockchain peer-to-peer network, an attacker subverts the network service's reputation system by creating a large number of active fake identities to control a significant portion of the network's peers, which allows them to manipulate the consensus process.

4. Distributed Denial of Service (DDoS) Attack: DDoS attack is a type of attack in which an attacker floods a network service with a large number of malicious attempts, causing it to

become overwhelmed and unable to process legitimate requests.

5. Eclipse Attack: In an eclipse attack, an attacker isolates a specific node or group of nodes from the rest of the network by controlling a portion of the network's peers. By doing this, the attacker can manipulate the information the isolated nodes receive, causing them to make decisions based on false or misleading information. This can be used to disrupt the network's consensus mechanism, prevent transactions from being confirmed, and ultimately compromise the security and integrity of the blockchain.

6. Smart Contract Vulnerabilities: A malicious actor can exploit vulnerabilities in a smart contract to steal funds stored on the contract. For example, in the notable security incident "The DAO attack" that happened on the Ethereum blockchain in 2016, an attacker exploited a reentrancy vulnerability in the smart contract to repeatedly drain funds from The DAO, resulting in the loss of 3.6 million Ether. The incident caused a significant amount of controversy within the Ethereum community and led to a hard fork in the Ethereum blockchain.

Among all the above security challenges, smart contract vulnerabilities have caught the most attention from blockchain practitioners and researchers because of their significant threat to the security and integrity of a blockchain network. Once a contract's vulnerability is exploited, a lot of digital assets will be lost. Furthermore, it damages the network's reputation and makes it less attractive and trustable to potential users. Some typical smart contract vulnerabilities and their corresponding potential solutions are listed below:

1. Reentrancy attack: Reentrancy is a type of attack that can occur in smart contracts that allow untrusted external code to be executed within the contract, potentially leading to a depletion of resources or other unintended consequences. One solution to prevent this is to use a mutex (mutual exclusion) mechanism to ensure that the function is not re-entered while still executing.

2. Unchecked send: If a smart contract fails to check the return value of a transfer operation, the execution may resume, leading to a potential loss of funds. So, it is significant to check the transfer operation's return value and have a mechanism to handle the transfer failures.

3. Uninitialized storage pointer: This vulnerability occurs when a smart contract uses uninitialized storage pointers, allowing an attacker to read or write to arbitrary memory locations. One quick solution is to initialize all storage pointers before use.

4. Timestamp dependence: The timestamp dependence vulnerability happens when the smart contract relies on the block timestamp value to execute an operation. The value of the timestamp is generated by the node executing the smart contract which makes it manipulable and vulnerable to attacks. It will be more secure to use a robust source of time, such as the number of blocks since a specific event.

5. Lack of access control: If a smart contract does not have proper access controls for certain functions, it will be very easy for attackers to withdraw all the funds from the contract. So, implementing excellent access control mechanisms, like roles, modifiers, or libraries that provide access control functionalities, is critical in smart contract development.

6. Inadequate exception handling: A smart contract can enter an unrecoverable state if it fails to properly handle exceptions. One solution to prevent this is to have a proper exception-handling mechanism that can handle any unexpected situations and ensure that the contract remains stable.

7. Unsafe external calls: When a smart contract calls an external or third-party contract with a vulnerability that the attacker can exploit, an unsafe external call attack happens. It is important to use a secure library or a safe contract proxy pattern that acts as an intermediary between the contract and the external contracts, allowing us to perform some security checks before making the call.

Blockchain technology and smart contracts are still relatively new, and new vulnerabilities may emerge as the technology and the threat landscape change. Therefore, it is essential to have regular security audits, conduct intensive testing before deployment, and use best practices and libraries in contract development to mitigate these vulnerabilities and ensure smart contracts' security.

In BPET, all validators are entities with their identities verified. A malicious validator voting on false transactions will be detected and voted out of the validators' quorum by all other honest validators. Similarly, if a normal node wants to become a validator, all existing validators must vote to reach a consensus. Unfortunately, the process of detecting malicious actions of validators is not automatic in the current version of QBFT implementation. Moreover, the immediate finality property of QBFT eliminates the threats of long-range attacks aiming to revise the transaction history of the ledger and selfish mining attacks in which the attacker holds discovered blocks privately and then attempts to fork a private chain.

During development, all BPET smart contracts were fully tested using the Mocha test framework with the Chai assertion library before being deployed to the blockchain to eliminate vulnera-

bilities and ensure security. It is worth noting that timestamp dependence vulnerabilities exist in the PoolMarket smart contract to record the timestamp of the offer, bid, system marginal price, and pool price. Since this is only used to record basic information but not used for significant decision-making, its threat should be limited. One possible solution is to offload the price calculation process to microservice. We also leveraged the most popular Openzeppelin's ERC20 token libraries to implement EnergyToken. For the PoolMarket contract, we employed administrator roles, valid offer/bid modifiers, and Openzeppelin's libraries, providing access control functionalities to manage permissions of functions and operations. In the security field, it is always too soon to say it is secure enough. Therefore, more reviews, verifications, and audit tools will be used to ensure the security of the BPET smart contracts.

7.4 Fairness

The fairness issue in blockchain refers to the potential for some participants in a blockchain network to have an unfair advantage over others. It can occur in several ways, including centralization, incorrect protocol implementation, inadequate incentives, and access to information. Suppose a small group of participants controls a large portion of the network's resources, such as mining power or stake. In that case, they may be able to influence the consensus process and make decisions that benefit themselves at the expense of other participants, causing the centralization issue. Incorrect or malicious implementation of the protocol refers to a situation where a node that does not follow the protocol or has a malicious implementation can gain an unfair advantage over the other nodes. Inadequate incentives are a type of unfairness issue where the network's incentive structure may not be designed in a way that ensures fair participation. For example, suppose the rewards for participating in the network are not proportional to the resources invested. In that case, some participants may be able to earn more than others without making a fair contribution. Access to information refers to an issue in which some participants may have access to more information or resources than others, giving them an unfair advantage when making decisions on the network.

PoA QBFT consensus has some potential advantages over other consensus algorithms. Because validators are chosen based on their reputation, it can be more difficult for a small group of participants to control the network and make decisions that benefit themselves at the expense of others. Compared to the PoS, where stakes vary with the enormous differences among validators, in the design of the QBFT, each validator only has one vote, which is fair in the consensus process without any manipulation potential. Additionally, because the BFT algorithm is designed to

tolerate a certain number of malicious actors, it can be more resistant to Sybil attacks.

However, the PoA QBFT consensus also has some potential fairness issues. For example, if the process for determining validators' reputations is not transparent or is subject to manipulation, this can lead to a small group of participants having an unfair advantage. Additionally, suppose the network's incentive structure is not designed in a way that ensures fair participation. In that case, some validators may be able to earn more than others without making a fair contribution. Therefore, to ensure fairness, it is crucial to have a transparent and fair process for determining validators' reputations and to design an appropriate incentive mechanism for validators.

Another type of fairness issue that the BPET may have is trading price fairness, which is highly related to the design of the energy market rules. In the case of the Alberta electricity wholesale market, the price is determined by the merit order effect. In particular, the *system marginal prices* are offer prices proposed by energy suppliers or generators. This can cause a price fairness issue because the energy buyers or consumers lose their right to propose a bid price they would like to pay. In an oligopoly market, if all suppliers or generators collude on raising the energy offer prices, consumers have to buy energy at a higher price than they want. This is where regulators come into play to break the oligarchy and force all suppliers fully compete with each other, resulting in a reasonable offer price. In addition, it is possible to take the bid price into consideration in the merit order effect implementation, such as calculating a weighted price between the offer and bid prices.

7.5 Stablecoins

Even though digital assets and cryptocurrencies, such as Bitcoin(BTC) and Ethereum(ETH), have seen explosive growth for several years, they are not widely accepted as payment methods in our daily lives. One of the main reasons is their high volatility, which makes the price of cryptocurrencies much more unstable compared to fiat currencies. For example, the price of Ethereum in January 2023 is 68.03% below the all-time high of $4,891.70 in November 2021. The high volatility of typical cryptocurrency makes it a risky asset, which is unsuitable for online payment in energy trading applications. In addition, the value of a kWh of electricity should be steady. Therefore, energy trading needs a convenient, decentralized, secure, and price-stable digital asset to support its payment procedure. This is where the stablecoins come into play.

Stablecoins are cryptocurrencies whose value is pegged to that of another currency, commodity, or financial instrument [56]. Compared to typical cryptocurrencies, stablecoins are specifically designed to offer a steady price. Since their emergence, different types of stablecoins have

been proposed and developed, mainly consisting of traditional asset-backed stablecoins, crypto-collateralized stablecoins, and algorithmic stablecoins [43].

Traditional asset-backed stablecoins peg their prices with traditional collaterals, including fiat currencies (e.g., USD) and commodity (e.g., gold). For example, Tether USD (USDT), Circle's USD coin (USDC), Binance USD (BUSD), and True USD (TUSD) pegged their price to the U.S. Dollar. Stablecoins pegging their price to gold include Tether Gold (XAUT), DigixGlobal (DGX), PAX Gold (PAXG), and Gold Coin (GLC) etc. This type of stablecoin is easy to implement by maintaining the same value as fiat or commodity reserves, namely fully-reserved or fully-collateralized. The major problem with this form of stablecoins is that it requires some central organization to process all transactions, which is against the principle of decentralization for blockchain.

Crypto-collateralized stablecoins rely on a pool of other cryptocurrencies, such as Ethereum and Bitcoin, to protect the stablecoin from speculative attacks. Some examples are MakerDAO's Dai (DAI) and minimalist USD (USM) [1]. However, due to the fact that cryptocurrencies are volatile, this type of stablecoin is often "overcollateralized", meaning that they must have a pool of cryptocurrencies in reserves over the value of all the issued stablecoins. For example, USM employs a high-risk funding coin, called FUM, to deposit additional ETH into the pool to prevent the market from going "underwater" ($pool\_value < usm\_outstanding$) and make profits by taking all risks of Ethereum price fluctuation. DAI is backed by Ethereum (ETH) and other cryptocurrencies worth 150% of the DAI stablecoin in circulation [84].

Algorithmic stablecoins are non-collateralized [88] but typically rely on two tokens – one stablecoin and another cryptocurrency that backs the stablecoins – and so the algorithm (or the smart contract) regulates the relationship between the two. Some examples are TerraUSD (UST), magic internet money (MIM), frax (FRAX) and neutrino USD (USDN). Algorithmic stablecoins are typically "undercollateralized", meaning that they don't have independent assets in reserves to back the value of their stablecoins. However, because algorithmic stablecoins don't have one-to-one collateral in reserve, they are at risk of a catastrophic price collapse, as occurred to TerraUSD in May 2022.

An ideal stablecoin would be both fully collateralized and fully decentralized at the same time, leading to improved price stability and requiring no trust in centralized custodians. In a recent study, Murialdo [90] proposed a fully decentralized and fully collateralized stablecoin, called e-stablecoin, by pegging to the price of one kWh of electricity. Different from Bitcoin and Meter.io, which set their value by consuming electricity to do proof-of-work, e-stablecoin pegs its value to the

[1] https://github.com/usmfum/USM

price of one kWh of electricity that is generated and put into the system. The reserved electricity acts as a commodity, which is not lost and can later be extracted by redeeming the token.

In the BPET system, the proposed energy token (ETK) is a prototyping ERC20 stablecoin, which any other type of stablecoin can easily replace. The price stabilization mechanism implementation is not the focus of this study, so we logically define the value of an ETK as equal to one USD. We design a general interface for developers to deploy their stablecoins or plug existing solutions, such as USDT, USDC, or e-stablecoin, into the BPET system. However, it remains an interesting future research topic about the liquidity of stablecoin between a consortium or private blockchain and the open cryptocurrency market.

8 Conclusion and Future Work

In conclusion, this Ph.D. research explores the potential of performant blockchain-based solutions for peer-to-peer energy trading (P2P-ET). Through a systematic survey, the study first summarizes the mainstream blockchain performance evaluation solutions into two categories: empirical analysis and analytical modelling. The former includes including benchmarking, monitoring, experimental analysis and simulation; the latter focuses on stochastic models, such as Markov chains, queueing models and stochastic Petri nets (SPNs). Then, this study extensively evaluates the performance of two promising blockchain solutions: IOTA and Hyperledger Besu. Evaluation results reveal some interesting findings concerned with key parameters in determining the performance and scalability of both blockchain solutions and their suitability in building a P2P-ET system. In particular, IOTA has better network scalability, while its performance (e.g., latency) is significantly limited by the centralized Coordinator design. In contrast, a private Hyperledger Besu can scale up to a middle-size network, i.e., 14 nodes, without a noticeable performance drop. Based on the comprehensive comparison results, Hyperledger Besu is selected to design and develop a unified blockchain-based P2P-ET framework. The case study in the Alberta electricity wholesale market demonstrates the feasibility and efficiency of the proposed BPET system. These experimental results and research findings provide important insights for the design and implementation of blockchain-based solutions for P2P-ET and pave the way for further research in this area.

Future research topics include functional and non-functional improvements to the BPET system. For functionalities, it is interesting to design incentive mechanisms to encourage renewable energy suppliers and users and add compliance and regulations to enforce electricity delivery. For performance improvement, it would be interesting to capture the performance bottlenecks through a stochastic model, and improve the BPET's performance by optimizing smart contracts or moving price calculation off-chain. It is also our interest to investigate the possibility of bringing cutting-edge zero-knowledge-proof technologies into P2P-ET to roll up transactions and achieve better performance and privacy.

References

[1] J. Abdella, Z. Tari, A. Anwar, A. Mahmood, and F. Han, "An architecture and performance evaluation of blockchain-based peer-to-peer energy trading," *IEEE Transactions on Smart Grid*, 2021. 3, 41, 62, 6

[2] V. Acharya, A. E. Yerrapati, and N. Prakash, *Oracle Blockchain Quick Start Guide: A practical approach to implementing blockchain in your enterprise*. Packt Publishing Ltd, 2019. 6, 7

[3] N. Z. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2016. 2, 69

[4] M. Alaslani, F. Nawab, and B. Shihada, "Blockchain in iot systems: End-to-end delay evaluation," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8332–8344, 2019. 21

[5] A. Aldweesh, M. Alharby, M. Mehrnezhad, and A. Van Moorsel, "Opbench: A cpu performance benchmark for ethereum smart contract operation code," in *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2019, pp. 274–281. 17

[6] A. Aldweesh, M. Alharby, E. Solaiman, and A. van Moorsel, "Performance benchmarking of smart contracts to assess miner incentives in ethereum," in *2018 14th European Dependable Computing Conference (EDCC)*, IEEE, 2018, pp. 144–149. 17

[7] M. Alharby and A. van Moorsel, "Blocksim: A simulation framework for blockchain systems," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 135–138, 2019. 19

[8] AMIS Team, *Istanbul Byzantine Fault Tolerance*, `https://github.com/ethereum/EIPs/issues/650`, no. 650, 2017. [Online]. Available: `https://github.com/ethereum/EIPs/issues/650`. 44

[9] M. Andoni, V. Robu, D. Flynn, *et al.*, "Blockchain technology in the energy sector: A systematic review of challenges and opportunities," *Renewable and sustainable energy reviews*, vol. 100, pp. 143–174, 2019. 100

[10] E. Androulaki, A. Barger, V. Bortnikov, *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18, Porto, Portugal: Association for Computing Machinery, 2018, ISBN: 9781450355841. DOI: `10.1145/3190508.3190538`. [Online]. Available: `https://doi.org/10.1145/3190508.3190538`. 61

[11] A. Back, M. Corallo, L. Dashjr, *et al.*, "Enabling blockchain innovations with pegged sidechains," *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, vol. 72, pp. 201–224, 2014. 98

[12] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of hyperledger fabric," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, IEEE, 2018, pp. 65–74. 18, 61

[13] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Performance evaluation of the quorum blockchain platform," *arXiv preprint arXiv:1809.03421*, 2018. 62

[14] S. Benahmed, I. Pidikseev, R. Hussain, *et al.*, "A comparative analysis of distributed ledger technologies for smart contract development," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, IEEE, 2019, pp. 1–6. 18

[15] I. Bentov, R. Pass, and E. Shi, "Snow white: Provably secure proofs of stake.," *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 919, 2016. 98

[16] M. Bez, G. Fornari, and T. Vardanega, "The scalability challenge of ethereum: An initial quantitative analysis," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, IEEE, 2019, pp. 167–176. 62

[17] M. Bottone, F. Raimondi, and G. Primiero, "Multi-agent based simulations of block-free distributed ledgers," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, IEEE, 2018, pp. 585–590. 20

[18] Caliper Benchmarks Maintenance Team, *Simple Contract*, `https://github.com/hyperledger/caliper-benchmarks/blob/main/src/ethereum/simple/simple.sol`, 2019. [Online]. Available: `https://github.com/hyperledger/caliper-benchmarks/blob/main/src/ethereum/simple/simple.sol`. 46

[19] B. Cao, S. Huang, D. Feng, L. Zhang, S. Zhang, and M. Peng, "Impact of network load on direct acyclic graph based blockchain for internet of things," in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, IEEE, 2019, pp. 215–218. 21

[20] R. Casado-Vara, J. Prieto, F. De la Prieta, and J. M. Corchado, "How blockchain improves the supply chain: Case study alimentary supply chain," *Procedia computer science*, vol. 134, pp. 393–398, 2018. 40

[21] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186. 44

[22] 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. 101

[23] CBinsights, "Banking is only the beginning: 58 big industries blockchain could transform," CB Information Services, Inc., Tech. Rep., Mar. 2021. [Online]. Available: `https://www.cbinsights.com/research/industries-disrupted-blockchain/`. 1

[24] S. Chen, J. Zhang, R. Shi, J. Yan, and Q. Ke, "A comparative testing on performance of blockchain and relational database: Foundation for applying smart technology into current business systems," in *International Conference on Distributed, Ambient, and Pervasive Interactions*, Springer, 2018, pp. 21–34. 24

[25] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *Ieee Access*, vol. 4, pp. 2292–2303, 2016. 1

[26] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," *White Paper*, pp. 1–49, 2016. [Online]. Available: `https://obyte.org/Byteball.pdf`. 6, 9, 97

[27] A. Cullen, P. Ferraro, C. King, and R. Shorten, "On the resilience of dag-based distributed ledgers in iot applications," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7112–7122, 2020. 32

[28] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proceedings of the 2019 international conference on management of data*, 2019, pp. 123–140. 98

[29] 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*, 2017, pp. 1085–1100. 7, 15, 16, 1

[30] Z. Dong, E. Zheng, Y. Choon, and A. Y. Zomaya, "Dagbench: A performance evaluation framework for dag distributed ledgers," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, IEEE, 2019, pp. 264–271. 16, 24

[31] A. Dorri, F. Luo, S. S. Kanhere, R. Jurdak, and Z. Y. Dong, "Spb: A secure private blockchain-based solution for distributed energy trading," *IEEE Communications Magazine*, vol. 57, no. 7, pp. 120–126, 2019. 2

[32] A. Esmat, M. de Vos, Y. Ghiassi-Farrokhfal, P. Palensky, and D. Epema, "A novel decentralized platform for peer-to-peer energy trading market with blockchain technology," *Applied Energy*, vol. 282, p. 116 123, 2021. 40

[33] 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. 98

[34] M. Faizan, T. Brenner, F. Foerster, C. Wittwer, and B. Koch, "Decentralized bottom-up energy trading using ethereum as a platform," *Journal of Energy Markets*, vol. 12, no. 2, 2019. 69, 70

[35] C. Fan, "Performance analysis and design of an iot-friendly dag-based distributed ledger system," Master of Science Thesis, University of Alberta, 2019. 33

[36] C. Fan, S. Ghaemi, H. Khazaei, Y. Chen, and P. Musilek, "Performance analysis of the iota dag-based distributed ledger," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 6, no. 3, pp. 1–20, 2021. [Online]. Available: `https://doi.org/10.1145/3485188`. 5, 26

[37] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126 927–126 950, 2020. 5, 14, 61

[38] C. Fan, H. Khazaei, Y. Chen, and P. Musilek, "Towards a scalable dag-based distributed ledger for smart communities," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, Limerick, Ireland, Ireland: IEEE, Apr. 2019, pp. 177–182. DOI: `10.1109/WF-IoT.2019.8767342`. 26, 27

[39] C. Fan, C. Lin, H. Khazaei, and P. Musilek, "Performance analysis of hyperledger besu in private blockchain," in *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, IEEE, 2022, pp. 64–73. 5

[40] C. Faria and M. Correia, "Blocksim: Blockchain simulator," in *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, 2019, pp. 439–446. 19

[41] P. Ferraro, C. King, and R. Shorten, "Distributed ledger technology for smart cities, the sharing economy, and social compliance," *IEEE Access*, vol. 6, pp. 62 728–62 746, 2018. 21

[42] P. Ferraro, C. King, and R. Shorten, "Iota-based directed acyclic graphs without orphans," *arXiv preprint arXiv:1901.07302*, pp. 1–11, 2018. 32

[43] I. Fiedler and L. Ante, "Stablecoins," in *The Emerald Handbook on Cryptoassets: Investment Opportunities and Challenges*, Emerald Publishing Limited, 2023, pp. 93–105. 107

[44] H. Foundation, "An introduction to hyperledger," Hyperledger Foundation, Tech. Rep., Aug. 2018. [Online]. Available: `https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf`. 7

[45] H. Foundation, *Hyperledger besu ethereum client*, `https://besu.hyperledger.org/en/stable/`, Last accessed 2023-01-31, 2022. [Online]. Available: `https://besu.hyperledger.org/en/stable/`. 4

[46] K. Gai, Y. Wu, L. Zhu, M. Qiu, and M. Shen, "Privacy-preserving energy trading using consortium blockchain in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3548–3558, 2019. 69, 70

[47] S. Geissler, T. Prantl, S. Lange, F. Wamser, and T. Hossfeld, "Discrete-time analysis of the blockchain distributed ledger technology," in *2019 31st International Teletraffic Congress (ITC 31)*, IEEE, 2019, pp. 130–137. 21

[48] F. Geyer, H. Kinkelin, H. Leppelsack, *et al.*, "Performance perspective on private distributed ledger technologies for industrial networks," in *2019 International Conference on Networked Systems (NetSys)*, IEEE, 2019, pp. 1–8. 21, 24, 61

[49] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68. 98

[50] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *Acm Sigact News*, vol. 33, no. 2, pp. 51–59, 2002. 97, 102

[51] H. Gupta and D. Janakiram, "Cdag: A serialized blockdag for permissioned blockchain," *arXiv preprint arXiv:1910.08547*, pp. 1–13, 2019. 9

[52] A. Hahn, R. Singh, C.-C. Liu, and S. Chen, "Smart contract-based campus demonstration of decentralized transactive energy auctions," in *2017 IEEE Power & energy society innovative smart grid technologies conference (ISGT)*, IEEE, 2017, pp. 1–5. 69, 70

[53] R. Han, V. Gramoli, and X. Xu, "Evaluating blockchains for iot," in *2018 9Th IFIP international conference on new technologies, mobility and security (NTMS)*, IEEE, 2018, pp. 1–5. 18, 23

[54] R. Han, G. Shapiro, V. Gramoli, and X. Xu, "On the performance of distributed ledgers for internet of things," *Internet of Things*, p. 100 087, 2019. 18

[55] Y. Hao, Y. Li, X. Dong, L. Fang, and P. Chen, "Performance analysis of consensus algorithm in private blockchain," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 280–285. 18, 22, 24

[56] A. Hayes, *Stablecoins: Definition, how they work, and types*, `https://www.investopedia.com/terms/s/stablecoin.asp`, Last accessed 2023-01-12, 2022. [Online]. Available: `https://www.investopedia.com/terms/s/stablecoin.asp`. 106

[57] M. Hearn and R. G. Brown, "Corda: A distributed ledger," *Corda Technical White Paper*, vol. 2016, 2016. 6, 7

[58] Holochain, *What is holochain?* https://www.holochain.org/what-holochain/, Last accessed 2023-01-31, 2022. [Online]. Available: https://www.holochain.org/what-holochain/. 7

[59] D. Huang, X. Ma, and S. Zhang, "Performance analysis of the raft consensus algorithm for private blockchains," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019. 21

[60] E. IO, "Eos. io technical white paper," *EOS. IO (accessed 18 December 2017) https://github.com/EOSIO/Documentation*, 2017. 6, 7

[61] *Istanbul BFT's design cannot successfully tolerate fail-stop failures*, https://github.com/ConsenSys/quorum/issues/305, 2018. [Online]. Available: https://github.com/ConsenSys/quorum/issues/305. 44

[62] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang, and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium blockchains," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3154–3164, 2017. 2, 69

[63] Y. Kawase and S. Kasahara, "Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism," in *International Conference on Queueing Theory and Network Applications*, Springer, 2017, pp. 75–88. 21

[64] D. Khovratovich and J. Law, "Sovrin: Digital identities in the blockchain era," *Github Commit by jasonalaw October*, vol. 17, pp. 38–99, 2017. 7

[65] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2018, pp. 583–598. 98

[66] U. R. Krieger, M. H. Ziegler, and H. L. Cech, "Performance modeling of the consensus mechanism in a permissioned blockchain," in *International Conference on Computer Networks*, Springer, 2019, pp. 3–17. 21

[67] B. Kusmierz and A. Gal, "Probability of being left behind and probability of becoming permanent tip in the tangle v0.2," *White Paper*, pp. 1–15, 2018. 30

[68] B. Kusmierz, W. Sanders, A. Penzkofer, A. Capossele, and A. Gal, "Properties of the tangle for uniform random and random walk tip selection," in *2019 IEEE International Conference on Blockchain (Blockchain)*, IEEE, New York, NY, USA: IEEE, 2019, pp. 228–236. 31

[69] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability," in *2019 IEEE international conference on blockchain (Blockchain)*, IEEE, 2019, pp. 536–540. 26, 61

[70] J. Kwon and E. Buchman, "Cosmos whitepaper," *A Netw. Distrib. Ledgers*, 2019. 98

[71] G. Leeming, J. Cunningham, and J. Ainsworth, "A ledger of me: Personalizing healthcare using blockchain technology," *Frontiers in medicine*, vol. 6, p. 171, 2019. 40

[72] C. LeMahieu, "Nano: A Feeless Distributed Cryptocurrency Network," *Whitepaper*, pp. 1–8, 2014. [Online]. Available: https://nano.org/en/whitepaper. 6, 98

[73] C. LeMahieu, "Nano: A feeless distributed cryptocurrency network," *White Paper*, pp. 1–8, 2018. [Online]. Available: https://content.nano.org/whitepaper/Nano%5C%5FWhitepaper%5C%5Fen.pdf. 9

[74] S. D. Lerner, "Dagcoin: A cryptocurrency without blocks," *White paper*, 2015. 98

[75] C. Li, P. Li, D. Zhou, W. Xu, F. Long, and A. Yao, "Scaling nakamoto consensus to thousands of transactions per second," *arXiv preprint arXiv:1805.03870*, 2018. 98

[76] M. Li, D. Hu, C. Lal, M. Conti, and Z. Zhang, "Blockchain-enabled secure energy trading with verifiable fairness in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6564–6574, 2020. 2

[77] Q.-L. Li, J.-Y. Ma, and Y.-X. Chang, "Blockchain queue theory," in *International Conference on Computational Social Networks*, Springer, 2018, pp. 25–40. 21

[78] Q.-L. Li, J.-Y. Ma, Y.-X. Chang, F.-Q. Ma, and H.-B. Yu, "Markov processes in blockchain systems," *Computational Social Networks*, vol. 6, no. 1, pp. 1–28, 2019. 21

[79] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial internet of things," *IEEE transactions on industrial informatics*, vol. 14, no. 8, pp. 3690–3700, 2017. 2, 69

[80] A. Lohachab, S. Garg, B. H. Kang, and M. B. Amin, "Performance evaluation of hyperledger fabric-enabled framework for pervasive peer-to-peer energy trading in smart cyber–physical systems," *Future Generation Computer Systems*, vol. 118, pp. 392–416, 2021. 62, 69, 70,

[81] C. Long, J. Wu, C. Zhang, L. Thomas, M. Cheng, and N. Jenkins, "Peer-to-peer energy trading in a community microgrid," in *2017 IEEE power & energy society general meeting*, IEEE, 2017, pp. 1–5. 67

[82] LTO, *What is lto network?* `https://www.ltonetwork.com/`, Last accessed 2023-01-31, 2022. [Online]. Available: `https://www.ltonetwork.com/`. 7

[83] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 17–30. 98

[84] MakerDAO, *A guide to dai stats*, `https://blog.makerdao.com/a-guide-to-dai-stats/`, Last accessed 2023-01-19, 2020. [Online]. Available: `https://blog.makerdao.com/a-guide-to-dai-stats/`. 107

[85] E. Mengelkamp, J. Gärttner, K. Rock, S. Kessler, L. Orsini, and C. Weinhardt, "Designing microgrid energy markets: A case study: The brooklyn microgrid," *Applied Energy*, vol. 210, pp. 870–880, 2018. 68

[86] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, IEEE, New York, NY, USA: IEEE, 2016, pp. 1–3. 1

[87] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, "Sprites and state channels: Payment networks that go faster than lightning," in *International Conference on Financial Cryptography and Data Security*, Springer, 2019, pp. 508–526. 98

[88] M. Mita, K. Ito, S. Ohsawa, and H. Tanaka, "What is stablecoin?: A survey on price stabilization mechanisms for decentralized payment systems," in *2019 8th International Congress on Advanced Applied Informatics (IIAI-AAI)*, IEEE, 2019, pp. 60–66. 107

[89] H. Moniz, "The istanbul bft consensus algorithm," *arXiv preprint arXiv:2002.03613*, 2020. 12, 44, 45

[90] M. Murialdo, "E-stablecoin: Separating fact from fiction," *Cryptoeconomic Systems*, 2022. 107

[91] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *White Paper*, pp. 1–9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf. 1, 7

[92] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *J. Gen. Philos. Sci.*, vol. 39, no. 1, pp. 53–67, 2008, ISSN: 09254560. DOI: 10.1007/s10838-008-9062-0. [Online]. Available: https://bitcoin.org/bitcoin.pdf. 3, 6

[93] R. Network, *What is the raiden network*, 2019. 98

[94] T. S. L. Nguyen, G. Jourjon, M. Potop-Butucaru, and K. L. Thai, "Impact of network delays on hyperledger fabric," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2019, pp. 222–227. 61

[95] M. T. Oliveira, G. R. Carrara, N. C. Fernandes, *et al.*, "Towards a performance evaluation of private blockchain frameworks using a realistic workload," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, IEEE, 2019, pp. 180–187. 17

[96] S. Pandey, G. Ojha, and B. Shrestha, "Blocksim: A practical simulation tool for optimal network design, stability and planning.," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2019, pp. 133–137. 19

[97] N. Papadis, S. Borst, A. Walid, M. Grissa, and L. Tassiulas, "Stochastic models and wide-area network measurements for blockchain design and analysis," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 2546–2554. 21

[98] Performance and S. W. Group, "Hyperledger blockchain performance metrics(white paper v1.01)," HyperLedger Found., Tech. Rep., 2018. 16, 45, 61

[99] Péter Szilágyi, *EIP-225: Clique proof-of-authority consensus protocol*, https://eips.ethereum.org/EIPS/eip-225, no. 225, Mar. 2017. [Online]. Available: https://eips.ethereum.org/EIPS/eip-225. 43

[100] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2017, pp. 1–6. 18, 22

[101] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017. 98

[102] J. Poon and T. Dryja, *The bitcoin lightning network: Scalable off-chain instant payments*, 2016. 98

[103] S. Popov, "The tangle," *White Paper*, pp. 1–28, 2016. [Online]. Available: http://www.descryptions.com/Iota.pdf. 3, 6, 9, 10,

[104] S. Popov, H. Moog, D. Camargo, *et al.*, "The coordicide," *White Paper*, pp. 1–55, 2020. 12

[105] N. R. Pradhan, A. P. Singh, N. Kumar, M. Hassan, and D. Roy, "A flexible permission ascription (fpa) based blockchain framework for peer-to-peer energy trading with performance evaluation," *IEEE Transactions on Industrial Informatics*, 2021. 3, 41, 62

[106] Radix, *What is radix?* https://learn.radixdlt.com/article/what-is-radix, Last accessed 2023-01-31, 2022. [Online]. Available: https://learn.radixdlt.com/article/what-is-radix. 6

[107] J. Reed, *Litecoin: An Introduction to Litecoin Cryptocurrency and Litecoin Mining*. CreateSpace Independent Publishing Platform, 2017. 3, 6, 7

[108] S. Ricci, E. Ferreira, D. S. Menasche, A. Ziviani, J. E. Souza, and A. B. Vieira, "Learning blockchain delays: A queueing theory approach," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 122–125, 2019. 21

[109] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, 2017, pp. 70–74. 18, 22, 62

[110] R. Saltini and D. Hyland-Wood, "Correctness analysis of ibft," *arXiv preprint arXiv:1901.07160*, 2019. 13, 44

[111] R. Saltini and D. Hyland-Wood, "Ibft 2.0: A safe and live variation of the ibft blockchain consensus protocol for eventually synchronous networks," *arXiv preprint arXiv:1909.10194*, 2019. 13, 44

[112] C. N. Samuel, S. Glock, F. Verdier, and P. Guitton-Ouhamou, "Choice of ethereum clients for private blockchain: Assessment from proof of authority perspective," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, 2021, pp. 1–5. 54, 62

[113] M. Schäffer, M. d. Angelo, and G. Salzer, "Performance and scalability of private ethereum blockchains," in *International Conference on Business Process Management*, Springer, 2019, pp. 103–118. 54, 57, 62

[114] D. Schwartz, N. Youngs, and A. Britto, "The Ripple protocol consensus algorithm," *Ripple Labs Inc White Pap.*, pp. 1–8, 2014. [Online]. Available: `http://www.naation.com/ripple-consensus-whitepaper.pdf`. 7

[115] S. Seven, G. Yao, A. Soran, A. Onen, and S. Muyeen, "Peer-to-peer energy trading in virtual power plant based on blockchain smart contracts," *IEEE Access*, vol. 8, pp. 175 713–175 726, 2020. 40

[116] A. Shahid, A. Almogren, N. Javaid, F. A. Al-Zahrani, M. Zuair, and M. Alam, "Blockchain-based agri-food supply chain: A complete solution," *IEEE Access*, vol. 8, pp. 69 230–69 243, 2020. 40

[117] Y. Shahsavari, K. Zhang, and C. Talhi, "Performance modeling and analysis of the bitcoin inventory protocol," in *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, IEEE, 2019, pp. 79–88. 21

[118] Z. Shi, H. Zhou, Y. Hu, S. Jayachander, C. de Laat, and Z. Zhao, "Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth," in *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, IEEE, 2019, pp. 50–57. 62

[119] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: Serialization of proof-of-work events: Confirming transactions via recursive elections," *White Paper*, vol. 2016, pp. 1–66, 2016. [Online]. Available: `https://eprint.iacr.org/2016/1159.pdf` (visited on 07/23/2020). 9

[120] Y. Sompolinsky and A. Zohar, "Phantom and ghostdag: A scalable generalization of nakamoto consensus," *White Paper*, vol. 2018, pp. 1–14, 2018. [Online]. Available: `https://eprint.iacr.org/2018/104.pdf` (visited on 07/23/2020). 9

[121] T. Sousa, T. Soares, P. Pinson, F. Moret, T. Baroche, and E. Sorin, "Peer-to-peer and community-based markets: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 104, pp. 367–378, 2019. 67

[122] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, IEEE, 2018, pp. 1–8. 21, 24

[123] S. Tanwar, K. Parekh, and R. Evans, "Blockchain-based electronic healthcare record system for healthcare 4.0 applications," *Journal of Information Security and Applications*, vol. 50, p. 102 407, 2020. 40

[124] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing hyperledger fabric blockchain platform," in *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, IEEE, 2018, pp. 264–276. 24, 61

[125] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Sok: Sharding on blockchain," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 41–61. 98

[126] S. Wang, A. F. Taha, J. Wang, K. Kvaternik, and A. Hahn, "Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 8, pp. 1612–1623, 2019. 40, 69, 70

[127] S. Wang, "Performance evaluation of hyperledger fabric with malicious behavior," in *International Conference on Blockchain*, Springer, 2019, pp. 211–219. 61

[128] D. J. Wilkins, R. Chitchyan, and M. Levine, "Peer-to-peer energy markets: Understanding the values of collective and community trading," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–14. 67

[129] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014. 6, 7

[130] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White Paper*, vol. 21, pp. 2327–4662, 2016. 98

[131] H. Wu, J. Cao, Y. Yang, *et al.*, "Data management in supply chain using blockchain: Challenges and a case study," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, IEEE, 2019, pp. 1–8. 40

[132] K. Yeow, A. Gani, R. W. Ahmad, J. J. Rodrigues, and K. Ko, "Decentralized consensus for edge-centric internet of things: A review, taxonomy, and research issues," *IEEE Access*, vol. 6, pp. 1513–1524, 2017, ISSN: 21693536. DOI: 10.1109/ACCESS.2017.2779263. 9

[133] P. Yuan, K. Zheng, X. Xiong, K. Zhang, and L. Lei, "Performance modeling and analysis of a hyperledger-based system using gspn," *Computer Communications*, 2020. 21, 24

[134] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 931–948. 98

[135] M. Zander, T. Waite, and D. Harz, "Dagsim : Simulation of dag-based distributed ledger protocols," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 118–121, 2018, ISSN: 01635999. DOI: 10.1145/3308897.3308951. 5, 26, 27

[136] M. Zander, T. Waite, and D. Harz, "Dagsim: Simulation of dag-based distributed ledger protocols," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 118–121, 2019. 19

[137] C. Zhang, J. Wu, Y. Zhou, M. Cheng, and C. Long, "Peer-to-peer energy trading in a microgrid," *Applied Energy*, vol. 220, pp. 1–12, 2018. 64

[138] H. Zhang, C. Jin, and H. Cui, "A method to predict the performance and storage of executing contract for ethereum consortium-blockchain," in *International Conference on Blockchain*, Springer, 2018, pp. 63–74. 21

[139] W. Zhao, S. Jin, and W. Yue, "Analysis of the average confirmation time of transactions in a blockchain system," in *International Conference on Queueing Theory and Network Applications*, Springer, 2019, pp. 379–388. 21

[140] P. Zheng, Z. Zheng, X. Luo, X. Chen, and X. Liu, "A detailed and real-time performance monitoring framework for blockchain systems," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, IEEE, 2018, pp. 134–143. 17, 24

[141] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16 440–16 455, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2967218. 97