Master's Thesis

Blockchain Algorithms with Random Committee Selection

Yusuke Ichiki

23M30508

Graduate Major in Computer Science
School of Computing
Tokyo Institute of Technology

Supervisor: Xavier Défago

January, 2025

Abstract

This research tackles the significant challenges faced by distributed energy trading in microgrids, specifically the scalability and efficiency limitations inherent in traditional Byzantine Fault Tolerance (BFT) algorithms. These algorithms struggle with high-communication overhead when achieving consensus on blocks in large-scale systems. To address these issues, this study introduces a new approach that integrates moving target defense mechanisms into BFT algorithms through the implementation of random committee selection. This innovative process involves the dynamic and unpredictable formation of committees, ensuring both scalability and efficiency such as throughput and message complexity.

Central to the proposed solution is the development of a method for generating consensus-based random numbers within the committees. Utilizing block commitment and threshold signatures, the system ensures a robust agreement process. Comprehensive simulations involving the large number of participants validate the performance improvements brought about by this approach. The results demonstrate significant gains in throughput and latency, particularly under low-latency network conditions.

This research explores the broader implications of committee-based consensus mechanisms. It highlights their potential to improve scalability and security while maintaining decentralization. By analyzing existing BFT algorithms, identifying their limitations, and introducing application of committee selection, this study contributes a foundational framework for efficient decentralized systems and aim not to be limited to a specific algorithm or situation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The emergence of blockchain technology has fundamentally transformed digital systems by enabling decentralized, secure, and transparent platforms for data exchange. Through distributed ledger technology, blockchain eliminates the need for a centralized authority, allowing peer-to-peer transactions to occur with trust and security. At the core of blockchain's functionality lies the consensus mechanism, a process by which network participants agree on the validity of transactions. Traditionally, consensus has been achieved through mechanisms such as Proof of Work (PoW) [15] and Proof of Stake (PoS) [11]. While effective, these methods face challenges regarding scalability, energy efficiency, and security, particularly as blockchain networks grow in size and complexity.

1.1 Problems

In recent years, committee-based consensus mechanisms have gained traction as a promising alternative, offering improved scalability and reduced energy consumption. Unlike PoW or PoS, committee-based consensus relies on a smaller, randomly selected subset of network participants to validate transactions and add blocks to the chain. This approach reduces computational demands and promotes faster transaction speeds, making it suitable for high-throughput applications. A critical component in this approach is the committee selection algorithm, which determines the subset of participants involved in consensus at any given time. Ensuring that committee selection is unanimous, random, and unpredictable is essential for maintaining the security, decentralization, and integrity of the blockchain network.

Despite its promise, the random selection of committee members introduces challenges related to security and efficiency. Ensuring true randomness while preventing adversarial influence is crucial, as an insecure selection process could compromise the blockchain's integrity. Additionally, achieving efficient random selection without sacrificing performance or scalability is a significant technical challenge. Existing algorithms in this area, such as Algorand [4] and Ouroboros [10], have made strides toward balancing randomness, security, and efficiency. However, there remains room for innovation and improvement, particularly in designing algorithms that are resilient to manipulation, capable of handling large-scale networks, and compatible with various blockchain architectures.

1.2 Contributions

This thesis explores the design and implementation of blockchain algorithms utilizing random committee selection. The objectives are to analyze existing methods, identify their strengths and limitations, and propose improvements that enhance security and scalability. By investigating the cryptographic techniques and randomization methods

used for committee selections, this study aims to contribute to the ongoing development of efficient, secure, and decentralized blockchain systems.

The proposed method reduces message complexity and need for view changes of the consensus algorithms used in blockchain systems and improves throughput and latency.

1.3 Road Map

The structure of this thesis is as follows. Chapter 2 provides an overview of the essential concepts and terminologies of this research. Chapter 3 introduces the proposed method for random committee selection using random beacon. In Chapter 4, we address scalability challenges in consensus algorithms, focusing on message complexity and committee corruption probability. Then, we compare the proposed solution with existing algorithms. Chapter 5 evaluates the performance of the proposed solution through extensive simulations. Chapter 6 summarizes the contributions of this thesis.

Chapter 2

Background and Definitions

We describe the terminology used for the proofs and the committee selection protocol in the framework of blockchain.

2.1 Background

2.1.1 Consensus Problem

The consensus problem requires all correct nodes to agree on the same value v, which has to be one of the proposed values. They use two main functions: propose and decide. Initially, each node i selects a value v_i and calls $Propose(v_i)$. After doing some executions, they complete their participation in the consensus by executing $Decide(v)$, meaning they finalize the protocol. An algorithm solves the consensus problem if it satisfies these properties:

- *Termination*: Every correct node eventually makes a decision.
- *Agreement*: No two correct nodes decide on different values.
- *Validity*: If a value v is decided, it must have been proposed by some nodes.

Consensus algorithms are a foundational element in the field of distributed systems, enabling a group of independent nodes or processes to agree on a single state or value. This agreement is crucial in distributed systems, as it ensures consistency, reliability, and coordination among multiple nodes operating without a central authority. Consensus algorithms were developed to tackle various challenges, such as network partition, node failures, and malicious behavior, all of which can compromise the stability and security of a distributed system.

Nodes in a distributed network can experience failures at any time. There are several types of failures such as crash faults, i.e., nodes simply stop functioning, and Byzantine faults [12], i.e., nodes behave maliciously or unpredictably. Consensus algorithms must be designed to tolerate these failures, typically by achieving agreement among a majority or a quorum of nodes.

Consensus mechanisms are integral to blockchain technology, allowing distributed networks to agree on the validity of transactions and the state of the ledger without a central authority.

2.1.2 Byzantine Fault Tolerance

Byzantine Fault Tolerance (BFT) algorithms form a critical area of study within consensus algorithms, addressing the challenge of achieving reliable consensus in the presence of Byzantine faults, where nodes may behave unpredictably, fail, or act maliciously [3].

BFT algorithms were developed to provide fault tolerance and consistency in distributed systems, making them foundational for applications requiring high reliability and security, such as distributed databases, distributed computing systems, and, more recently, blockchain networks.

Recently, BFT consensus has been widely adopted in blockchain technology, where the decentralized and open nature of these networks makes them susceptible to various security threats. BFT algorithms are commonly used in permissioned blockchain platforms like Hyperledger Fabric [17] and Quorum [9], where network participants are known and trusted, enabling efficient Byzantine consensus without sacrificing security.

2.1.3 Blockchain Technology

Blockchain technology leverages BFT algorithms to achieve secure, decentralized consensus across distributed networks. Originally conceived as a means of establishing trust and transparency without a central authority [15], blockchain relies on consensus algorithms to validate transactions, secure the ledger, and ensure that all participants maintain a consistent view of the network. BFT algorithms, designed to handle malicious or faulty nodes, have proven to be highly effective in blockchain networks, where participants are often untrusted and spread across diverse locations.

Despite their advantages, BFT algorithms face scalability challenges. Traditional BFT requires communication among all nodes in each consensus round, which limits performance as the network grows. Additionally, some BFT-based blockchains must balance security with decentralization, as overly centralized decision-making can undermine the openness and fairness that blockchain technology aims to achieve.

2.2 Nodes and Network

In this paper, we focus on a permissioned blockchain system, which assigns ID to each participant and knows the set of participants ahead of time. This is because energy trade involves sensitive data, such as pricing and consumption patterns and permissions make it easier to ensure compliance with laws and industry standards. In addition, we name a node with i-th ID as i for convenience.

The model has n nodes in the blockchain network, of which the adversary controls at most f Byzantine nodes. The BFT models tolerate even if f nodes are faulty. In this thesis, we assume $f = \lfloor \frac{n-1}{3} \rfloor$ [2]. Byzantine nodes act intentionally to undermine the system, such as lying, sending conflicting messages, or colluding with other Byzantine nodes.

We also focus on a partially synchronous model [6], where the network has a worst-case bound Δ[1] on message delivery known by all participants and they are not aware of an actual message delivery interval δ. The model assumes that the messages sent after the *global stabilization time* (GST) by honest nodes take less time than the worst-case bound, i.e., $\delta < \Delta$ holds for messages sent after GST.

2.3 Blockchain

This research supposes that every honest node in the system will put transactions into a block and store the block to a blockchain, so we describe the elements of the blockchain in advance.

[1]The model does not ensure that it always takes less than Δ with regard to the messages sent before GST.

A block proposed in round r B_r is a tuple $(r, H(B_{r-1}), \mathsf{txs})$, where $H(\cdot)$ is a public hash function modeled as a random oracle and txs is a series of transactions which are yet to be added to a blockchain. In addition to a normal block B_r, we define a genesis block $B_0 \coloneqq (0, \emptyset, \perp)$ and an empty block $\perp_r \coloneqq (r, H(B_{r-1}), \perp)$ to specify each node adds these blocks in specific occasions[2]. It is still possible that normal blocks have an empty transaction $\perp$; however, an empty block $\perp_r$ implies that the system did not agree on any proposed block and move on to round $r + 1$ to select a new committee.

Honest nodes do not need to specify all the blocks stored on blockchain when they make a block and choose which branch to add to; they only include the hash of the previous block $H(B_{r-1})$ and can track a chain from the latest block B_r to the genesis block B_0.

2.4 Random Beacon

Random beacon is a mechanism for generating random numbers especially utilized in decentralized applications (dApps), such as decentralized finance (DeFi) and non-fungible tokens (NFTs). This ensures that no single node can manipulate or predict the generated values [13]. The key technique making this algorithm practical is a *distributed verifiable random function* (DVRF), where nodes evaluate some input and publicly verify the output for correctness [7]. DVRFs consist of a setup function and three other functions which make signatures. In addition, a random beacon defines a function used for updating random numbers in each round and works as a protocol which generates unpredictable numbers at the end of the rounds, ensuring that every honest node has the same signatures as well as numbers.

Random beacon and DVRF satisfy four properties.

- *Pseudo-randomness*: guarantees that the output is indistinguishable from a value chosen uniformly and randomly even when the adversary is present.
- *Consistency*: ensures that the function computes the same random value σ_r on an input st_r, regardless of which set of partial signatures to use.
- *Robustness*: guarantees the ability to compute an output for any input even in the presence of adversary. Specifically, it requires that if the combine function does not return $\perp$, its output must pass the verification test, even when the adversary provides partial signatures to the combine function.
- *Uniqueness*: ensures that for every input st_r, there is a single unique value σ_r that passes the verification test.

We formalize these functions to clarify how the protocol behaves. We mainly adopt the BFTRAND [13] functions and add an additional one for committee selection described in Chapter 3. The protocol provides $Setup(1^\lambda, t, n)$, which takes inputs as a security parameter 1^λ [7, 8], a threshold parameter t, and the number of the nodes n and outputs as a global public key gpk, a list of public keys $\{pk_0, \ldots, pk_{n-1}\}$, and a list of secret keys $\{sk_0, \ldots, sk_{n-1}\}$. λ in the security parameter 1^λ is the parameter of the key size. A larger key size provides stronger security, but it increases computational and storage overhead. The size also measures collision resistance, which means how hard it is to find two different inputs such that the DVRF produces the same output. For instance, the collision resistance of a 256-bit DVRF output is around 2^{128} operations, which is virtually

[2]We assume the random oracle used in this model is SHA-256 [5], but the system can use another hash function as long as it always reacts to the same input with the same output that is random and uniformly distributed.

infeasible. The protocol ensures that every node can compute a threshold signature from any of $t + 1$ distinct nodes' correct partial signatures. In this thesis, we set $t = 2f$.

Each node of the protocol uses $Partial(st_r, sk_i, gpk)$, where st_r is a random state number of round r, receiving a proposed block to make their own partial signature $\sigma_{r,i}$ and proof $\pi_{r,i}$; the output of this function is $(i, \sigma_{r,i}, \pi_{r,i})$. Once the nodes receive $2f + 1$ corresponding sets of partial signatures and proofs $\{(\sigma_{r,i}, \pi_{r,i})\}_{i \in I}$ from a set of the sending nodes I, they generate a threshold signature with $Combine(st_r, \{\sigma_{r,i}, \pi_{r,i}\}_{i \in I}, gpk)$. This function produces a threshold signature σ_r and its corresponding proof π_r when each node has collected sufficient correct partial signatures and proofs; otherwise, the function outputs $\perp$. As soon as producing (σ_r, π_r), they use $Verify(st_r, \sigma_r, \pi_r, gpk)$, which serves to verify σ_r is the signature of st_r, using the proof π_r. This whole process of the functions is a DVRF.

In addition to the DVRF process, we apply $UpdateState(\sigma_r)$ to generate a random number, which is identical to the next round's state number st_{r+1}. The output of this function is σ_r. Therefore, it is possible $st_{r+1} = \perp$ when honest nodes cannot collect enough partial signatures in time.

The proofs for the properties of DVRF and random beacon used in this thesis remain the same as the original DVRF paper [7].

Chapter 3

Solution

3.1 Existing Method: Committee Selection

Committee selection is a critical process in this paper because our objective is to optimize scalability, security, and energy efficiency using this technique. In blockchain systems using committee-based consensus, a subset of nodes is dynamically selected to perform transaction validation or propose new blocks; however, the process of committee selection introduces its own challenges in simultaneously achieving unanimity, randomness, and unpredictability. In this thesis, we divide committee selection into two types: verifying committee selection and proposing committee selection.

3.1.1 Verifying Committee Selection

Verifying committee selection functions as a limitation of members who can agree with the proposed block or deny it. One of the examples of the consensus algorithm using this selection is Algorand [4]. This selects a verifying committee by using signatures of a proposed block, which we call cryptographic self-selection. Each node generates their own signature and compares it with a threshold, and if the hash of the signature is smaller than the threshold, the node becomes a verifier, i.e., i is a verifier if $H(\sigma_{r,i}) < \mathit{Threshold}$.[1]

Although this algorithm achieves selecting a verifying committee in a simple way, it is difficult to set the size of the committee for every round due to randomness of the hash function. When the size is too small to reach consensus, the protocol consumes redundant rounds and worsens throughput and latency even if the number of Byzantine nodes is small.

If the protocol sets the maximum size of the committee as n_V and the n_V nodes with the smallest $H(\sigma_{r,i})$ become verifiers, it cannot ensure unanimity of the committee in this case. This is because every node does not know about other signatures in advance and becoming a verifier depends on whether they send a message and it arrives in time.

3.1.2 Proposing Committee Selection

Proposing committee selection offers a fair and straightforward approach to assigning responsibilities of proposing a block in each round. We assume the protocol is round-based and randomly selects a leader in every round. On top of this, we extend the protocol to a variant of multiple proposers. The objective of the multiple proposers is avoiding redundant rounds and improving throughput and latency.

Most round-based algorithms adopt round-robin selection [3, 10]. It is a deterministic method and involves the sequential selection of nodes from a predefined list, where the

[1] Algorand decides state number st_r, which the original paper calls *seed*, based on the latest block B_{r-1}.

selection starts from the first node and cycles back to the beginning after reaching the end. This method ensures that every node, including a Byzantine node, has an equal opportunity to propose a block and assume a role in a predictable order. Due to its deterministic manner and predictability, the protocol is vulnerable to attacks and tends to waste rounds, which causes the deterioration of throughput and latency.

3.2 Proposed Solution: Random Beacon for Committee Selection

In this research, we propose the method that the protocol produces the committee's set and its priority, using random beacon. Random beacon generates a random number agreed among nodes in each round, namely st_r in this thesis. We introduce $Committee(st_r, n_P, n_V)$, where n_P, n_V are the size of the proposing and verifying committees respectively. This function outputs a proposing committee P_r and verifying committees $V_{r,1}, V_{r,2}$. This thesis mainly focuses on applying committee selection to two-phase BFT algorithms and assumes the protocol selects committees in each phase.[2] These three committees mean

$$P_r = \{c_0 \bmod n, \ldots, (c_0 + n_P - 1) \bmod n\}, \tag{3.1}$$

$$V_{r,1} = \{c_1 \bmod n, \ldots, (c_1 + n_V - 1) \bmod n\}, \text{ and} \tag{3.2}$$

$$V_{r,2} = \{c_2 \bmod n, \ldots, (c_2 + n_V - 1) \bmod n\} \ (c_k = H(r, st_r, k) \text{ for } k \in \mathbb{N}). \tag{3.3}$$

In Chapter 5, we use the above fashion to select committees in the simulation.

In addition to this method, we propose another selection manner which is more shuffled but complex. The protocol selects

$$P_r = \bigcup_{k=0}^{n'_P-1} \{c_k \bmod n\}, \text{ such that } c_k \not\equiv c_l \ \forall c_l \in \bigcup_{m=0}^{k-1} \{c_m\}, |P_r| = n_P, \tag{3.4}$$

$$V_{r,1} = \bigcup_{k=n'_P}^{n'_P+n'_V-1} \{c_k \bmod n\}, \text{ such that } c_k \not\equiv c_l \ \forall c_l \in \bigcup_{m=n'_P}^{k-1} \{c_m\}, |V_{r,1}| = n_V, \text{ and} \tag{3.5}$$

$$V_{r,2} = \bigcup_{k=n'_P+n'_V}^{n'_P+n'_V+n''_V-1} \{c_k \bmod n\}, \text{ such that } c_k \not\equiv c_l \ \forall c_l \in \bigcup_{m=n'_P+n'_V}^{k-1} \{c_m\}, |V_{r,2}| = n_V. \tag{3.6}$$

Equations (3.4), (3.5), and (3.6) describe an alternate method for committee selection. These equations aim to create shuffled and distinct committees in a more randomized manner compared to Equations (3.1), (3.2), and (3.3). This method selects each committee member with $H(r, st_r, k)$. Simply executing this way, it would not guarantee that P_r has n_P members and $V_{r,1}$ and $V_{r,2}$ have n_V members respectively. Therefore, this method checks if there is no duplication in the committee lists.

In the first case, each node calculates c_0, c_1, and c_2 by r and st_r and automatically selects committees because of no duplication in the committee.[3] However, once another round r' uses the same c_k again, the protocol selects the same committee members. It is ignorable when Byzantine nodes do not have close IDs; if not, the protocol needs the second method. This method continues calculating c_k until the committee has the sufficient number of the members whose IDs are distinct.

In both methods, the primary proposer (with ID $c_0 \bmod n$) has the best priority to propose a block, and the order of priority is the same as that of P_r. Each node uses this priority list to select one of the proposed blocks when there are multiple nodes.

[2]The protocol can apply $V_{r,3}$ for three-phase algorithms such as HotStuff [18].

[3]It is possible that a node become a member of the multiple committees in the same round.

3.3 Application for BFT Algorithms

3.3.1 Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (PBFT) [3] is one of the most well-known BFT algorithms. The authors design this algorithm for permissioned networks with a known set of nodes. It operates in one proposing phase and two verifying phases: pre-prepare, prepare, and commit, requiring a majority of nodes (at least $2f + 1$) to agree on each phase before moving to the next. PBFT is efficient in smaller networks but suffers from scalability limitations, as its communication complexity increases quadratically with the number of nodes.

Algorithm 1 Pseudocode for PBFT with committee selection

Require: $(gpk, pk_i, sk_i) \leftarrow Setup(1^\lambda, t, n)$

timer starts
$st_r \leftarrow UpdateState(\sigma_{r-1})$
$(P_r, V_{r,1}, V_{r,2}) \leftarrow Committee(st_r, n_P, n_V)$

if $i \in P_r$ **then**
  $(i, \sigma_{r,i}, \pi_{r,i}) \leftarrow Partial(B_{r,i}, sk_i, gpk)$
  broadcast $(\mathsf{Preprepare}, B_{r,i}, (i, \sigma_{r,i}, \pi_{r,i}))$
end if

if receive $(\mathsf{Preprepare}, B_{r,c_0}, (c_0, \sigma_{r,c_0}, \pi_{r,c_0}))$ **or** 2Δ passes **then**
  $B_r \leftarrow B_{r,j}$ with the biggest priority based on the P_r order
  if $i \in V_{r,1}$ **then**
    $(i, \sigma_{r,i}, \pi_{r,i}) \leftarrow Partial(B_r, sk_i, gpk)$
    send $(\mathsf{Prepare}, (i, \sigma_{r,i}, \pi_{r,i}))$ to $V_{r,2}$
  end if
end if

if receive $\lfloor \frac{2}{3} n_V \rfloor + 1$ $(\mathsf{Prepare}, (j, \sigma_{r,j}, \pi_{r,j}))$ **and** $i \in V_{r,2}$ **then**
  $(\sigma_r, \pi_r) \leftarrow Combine(B_r, \{\sigma_{r,j}, \pi_{r,j}\}, gpk)$
  if $Verify(B_r, \sigma_r, \pi_r, gpk) = 0$ **then**
    broadcast $(\mathsf{Commit}, (\sigma_r, \pi_r))$ and never broadcast another block at round r
  end if
end if

if receive $\lfloor \frac{2}{3} n_V \rfloor + 1$ $(\mathsf{Commit}, (\sigma_r, \pi_r))$ by 4Δ **then**
  add B_r to a *blockchain*
else
  $\sigma_r \leftarrow \perp_r$
end if
$r \leftarrow r + 1$

We apply random beacon to PBFT with no additional phases according to Algorithm 1. As soon as proposer commits B_{r-1}, it broadcasts its own block candidate B_r. Once the $V_{r,1}$ members get B_r, they create their partial signature $\sigma_{r,i}$ and multicast it to $V_{r,2}$. When the $V_{r,2}$ members get at least $\lfloor \frac{2}{3} n_V \rfloor + 1$ partial signatures, they create a threshold signature σ_r and use $Verify(st_r, \sigma_r, \pi_r, gpk)$. If they can verify it, they broadcast σ_r. Finally, each node collects $\lfloor \frac{2}{3} n_V \rfloor + 1$ threshold signatures, adds B_r to a blockchain, and starts round $r + 1$.

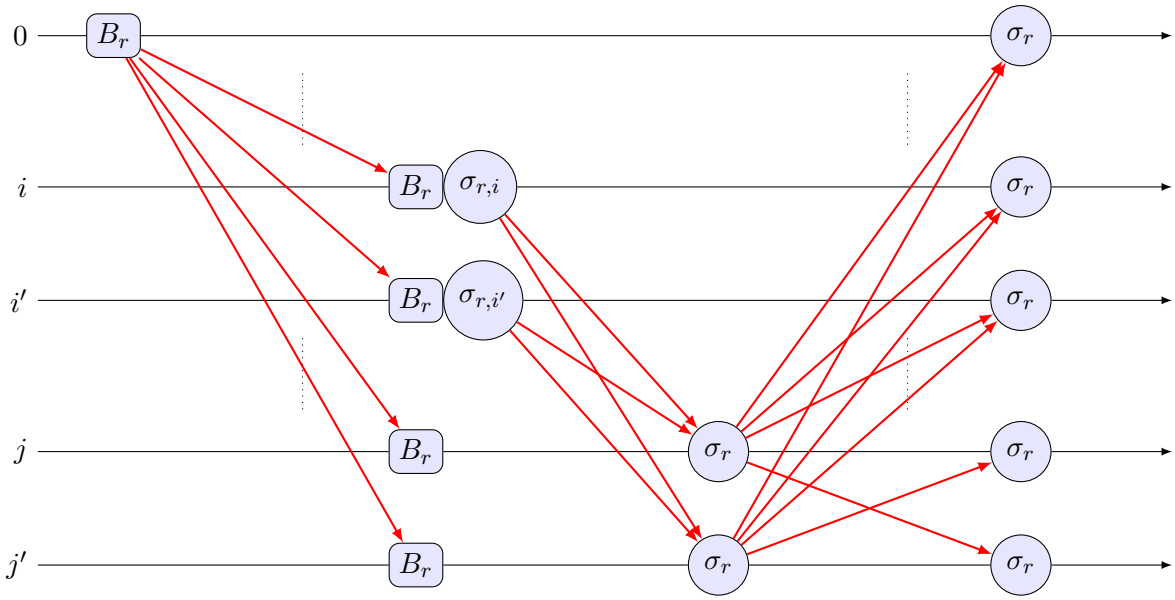

Figure 3.1: Illustration of applying random beacon to PBFT when node 0 is the proposer, $i, i' \in V_{r,1}$, and $j, j' \in V_{r,2}$. $V_{r,1}$ members do not broadcast but send to $V_{r,2}$.

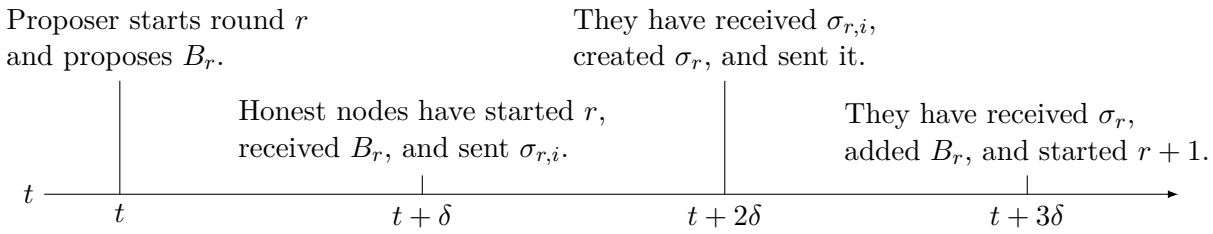

Figure 3.2: Timeline of PBFT when the proposer is honest ($t >$ GST). Every honest node sends messages which take δ.

On the condition that the protocol sets the timer 3Δ, every honest node at least creates a threshold signature in every round after GST, and it is efficient to commit B_r if there is no committee selection. However, when the protocol applies committee selection and the timer has fired, only at most n_V nodes have threshold signatures; it is difficult to use this as a checkpoint. In this thesis, we do not select committees for view changes, and each node broadcasts $\perp_r$ as a *view-change* message.

Figure 3.1 illustrates the application of a random beacon in the PBFT algorithm, focusing on the scenario where node 0 acts as the proposer. The diagram demonstrates the flow of messages and the sequence of actions performed by the nodes in the consensus process. Node 0 begins by committing the block from the previous round and proposing a new block B_r. This node then broadcasts the proposal to all nodes. Upon receiving the proposed block, i and i', the members of $V_{r,1}$, create their partial signatures $\sigma_{r,i}$ and $\sigma_{r,i'}$ for the block and multicast it to the second verifying committee $V_{r,2}$. j and j', members of $V_{r,2}$, collect at least $\frac{2}{3}$ of the partial signatures and combine them into a threshold signature σ_r. Then, they use the random beacon process to verify the signature; if it is valid, broadcast it to all nodes in the network. Once all honest nodes verify the threshold signature, they add B_r to the blockchain and proceed to the next round.

Figure 3.2 depicts the timeline of PBFT operations during a single round when the proposer is honest after GST. A node enters a round r, which means this node has received σ_r and verified it. As the process described in Figure 3.1, the messages take δ, so it overall

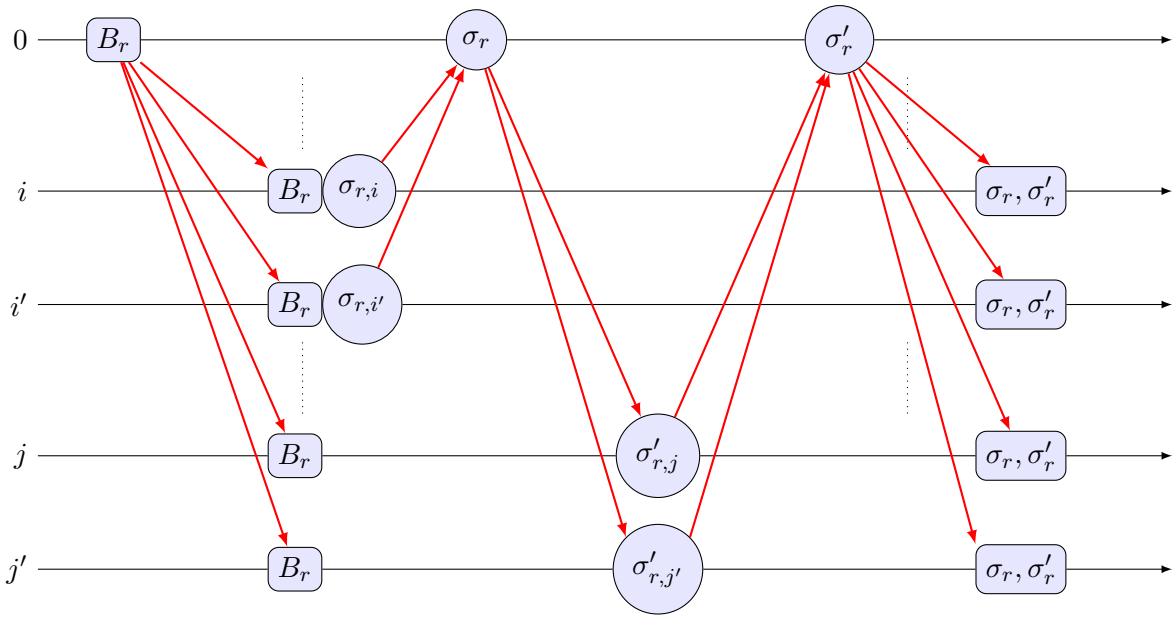

Figure 3.3: Illustration of applying random beacon to HotStuff-2 when node 0 is the proposer. HotStuff-2 uses threshold signatures twice a round.

takes 3δ for all honest nodes to commit the block.

3.3.2 HotStuff-2

HotStuff-2 [14] is a more recent BFT algorithm designed to improve throughput of BFT consensus. Unlike PBFT, HotStuff-2 uses a linear communication model, reducing communication overhead as the number of nodes increases. HotStuff-2's efficiency has made it a popular choice for high-throughput distributed systems and blockchain platforms.

Figure 3.3 illustrates the integration of a random beacon mechanism into the HotStuff-2 algorithm, which is Algorithm 2. The original paper [14] also uses threshold signatures, and there is no additional phase for applying committee selection as well as PBFT. In the first phase, the proposer broadcasts B_r. As soon as receiving B_r, each $V_{r,1}$ member creates their partial signature $\sigma_{r,i}$ and sends it to the proposer. The proposer collects $\lfloor \frac{2}{3} n_V \rfloor + 1$ partial signatures and uses $Combine(st_r, \{\sigma_{r,i}, \pi_{r,i}\}_{i \in I}, gpk)$ to make a threshold signature σ_r. Then, the proposer multicasts σ_r to $V_{r,2}$. Once they get σ_r, the $V_{r,2}$ members send it back. The proposer creates a threshold signature again and broadcasts it, and consequently each node adds B_r to a blockchain and enters $r + 1$.

Not using a threshold signature for view changes in PBFT, each node just broadcasts $\perp_r$ as a view change for HotStuff-2 in this thesis to make r and a view number equal.

3.4 Multiple Proposers

In BFTRAND [13], the random beacon's functions use st_r as a random number and a seed for st_{r+1}. However, we can use B_r as a seed if the outputs of $Partial(B_r, sk_i, gpk)$ and $Combine(B_r, \{\sigma_{r,i}, \pi_{r,i}\}_{i \in I}, gpk)$ are unpredictable. Pseudo-randomness of the random beacon is a sufficient condition and is proven in the papers about the random beacon protocol [7, 13]. Therefore, we change this part to uniquely choose one out of the multiple proposers' proposed blocks. Figure 3.4 is the illustration of applying multiple proposers to PBFT.

In addition, the protocol practices a priority list described in Section 3.2. Figure

Algorithm 2 Pseudocode for HotStuff-2 with committee selection

Require: $(gpk, pk_i, sk_i) \leftarrow Setup(1^\lambda, t, n)$

timer starts
$st_r \leftarrow UpdateState(\sigma_{r-1})$
$(P_r, V_{r,1}, V_{r,2}) \leftarrow Committee(st_r, n_P, n_V)$

if $i \in P_r$ **then**
  $(i, \sigma_{r,i}, \pi_{r,i}) \leftarrow Partial(B_{r,i}, sk_i, gpk)$
  broadcast $(\mathsf{Preprepare}, B_{r,i}, (i, \sigma_{r,i}, \pi_{r,i}))$
end if

if receive $(\mathsf{Preprepare}, B_{r,c_0}, (c_0, \sigma_{r,c_0}, \pi_{r,c_0}))$ **or** 2Δ passes **then**
  $B_r \leftarrow B_{r,j}$ with the biggest priority based on the P_r order
  if $i \in V_{r,1}$ **then**
    $(i, \sigma_{r,i}, \pi_{r,i}) \leftarrow Partial(B_r, sk_i, gpk)$
    send $(\mathsf{Preprepare}, (i, \sigma_{r,i}, \pi_{r,i}))$ to *leader* (the proposer of B_r)
  end if
end if

if receive $\lfloor \frac{2}{3} n_V \rfloor + 1$ $(\mathsf{Preprepare}, (j, \sigma_{r,j}, \pi_{r,j}))$ **and** i is *leader* **then**
  $(\sigma_r, \pi_r) \leftarrow Combine(B_r, \{\sigma_{r,j}, \pi_{r,j}\}, gpk)$
  if $Verify(B_r, \sigma_r, \pi_r, gpk) = 0$ **then**
    $(i, \sigma'_{r,i}, \pi'_{r,i}) \leftarrow Partial(\sigma_r, sk_i, gpk)$
    send $(\mathsf{Prepare}, \sigma_r, (i, \sigma'_r, \pi'_r))$ to $V_{r,2}$
  end if
end if

if receive $\left(\mathsf{Prepare}, \sigma_r, (j, \sigma'_{r,j}, \pi'_{r,j})\right)$ from *leader* **and** $i \in V_{r,2}$ **then**
  $(i, \sigma'_{r,i}, \pi'_{r,i}) \leftarrow Partial(\sigma_r, sk_i, gpk)$
  send $(\mathsf{Prepare}, (i, \sigma'_r, \pi'_r))$ to *leader* and never send another at round r
end if

if receive $\lfloor \frac{2}{3} n_V \rfloor + 1$ $\left(\mathsf{Prepare}, (j, \sigma'_{r,j}, \pi'_{r,j})\right)$ **and** i is *leader* **then**
  $(\sigma'_r, \pi'_r) \leftarrow Combine(\sigma_r, \{\sigma'_{r,j}, \pi'_{r,j}\}, gpk)$
  if $Verify(\sigma_r, \sigma'_r, \pi'_r, gpk) = 0$ **then**
    broadcast $(\mathsf{Commit}, (\sigma_r, \pi_r), (\sigma'_r, \pi'_r))$
  end if
end if

if receive $(\mathsf{Commit}, (\sigma_r, \pi_r), (\sigma'_r, \pi'_r))$ by 4Δ **then**
  if $Verify(B_r, \sigma_r, \pi_r, gpk) = 0$ **and** $Verify(\sigma_r, \sigma'_r, \pi'_r, gpk) = 0$ **then**
    add B_r to a *blockchain*
  end if
else
  $\sigma_r \leftarrow \perp_r$
end if
$r \leftarrow r + 1$

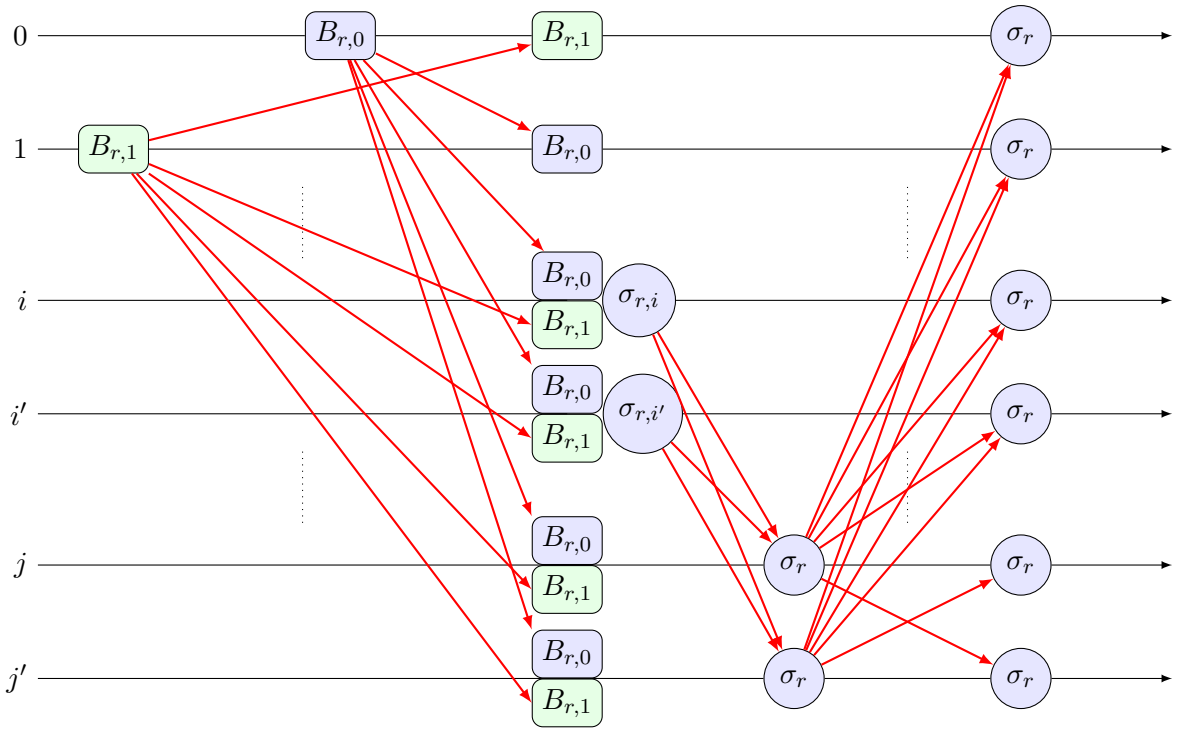

Figure 3.4: Illustration of applying multiple proposers to BFT algorithms when nodes 0 and 1 are the proposers, and 0 has the bigger priority.

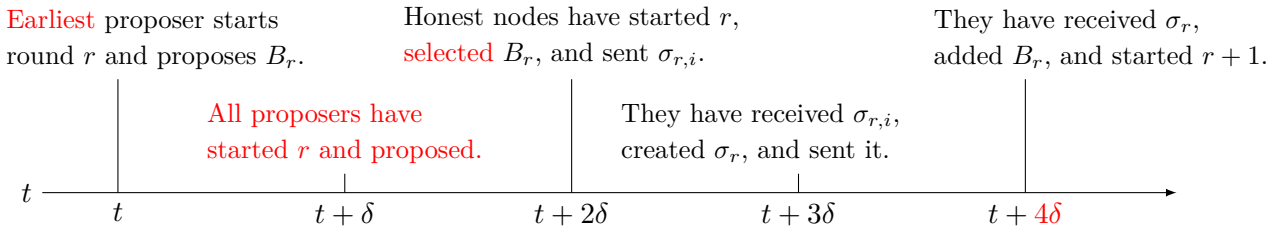

Figure 3.5: Timeline of PBFT with multiple proposers ($t > \mathsf{GST}$). The timeout is set to 4Δ to wait for all proposed blocks.

3.5 shows the timeline of PBFT with multiple proposers during a single round when the proposer is honest and there are more than $\frac{2}{3}$ honest verifiers out of n_V after GST. Until 2Δ has elapsed, each $V_{r,1}$ member sends $\sigma_{r,i}$ as soon as they get the block proposed from the first node of P_r (primary block). If they do not receive the primary block until 2Δ, they select a block with the highest priority from the blocks they have received in round r and send it, regardless of whether they get the primary block later. If they do not receive anything until 2Δ, they multicast a block they receive the earliest in r.

It takes at most 2Δ for $V_{r,1}$ members to get all honest proposers' blocks since the earliest proposer started the round at $t > \mathsf{GST}$. Both BFT algorithms cannot guarantee that all P_r members propose blocks; thus, they need to set an upper bound for proposing, which is 2Δ. They also adjust the limit for the round per se from 3Δ to 4Δ to accommodate the change to multiple proposers.

Chapter 4

Scalability

In blockchain systems, communication overhead plays a crucial role in determining scalability. This chapter examines the corruption of committees and the complexity of the messages associated with committee selection.

4.1 Message Complexity

Table 4.1 provides a comparative overview of the message complexity of PBFT and HotStuff-2 under different configurations.

PBFT needs $O(n^2)$ message exchanges to reach consensus if the primary node is honest. Moreover, it has $O(n^4)$ complexity for view changes [18] because all $O(n)$ non-faulty nodes broadcast $O(n)$ view-change messages which have the latest $2f + 1 (= O(n))$ quorum certificates. Worst case scenario, Byzantine nodes become primary $f (= O(n))$ times consecutively; thus, the worst case view changes have $O(n^4)$ complexity. With committee selection, PBFT reduces its complexity for consensus to $O(nn_V)$ and that for view changes to $O(n^3)$ because of not using $2f + 1$ quorum certificates.[1]

HotStuff-2 keeps a two-phase commit algorithm as PBFT and reduces message complexity to $O(n)$ when the primary node is honest. Furthermore, the worst-case scenario has only $O(n^2)$ complexity because all honest nodes send a view-change message only to the next primary node and this may continues f times. When applying committee selection, every honest proposer in HotStuff-2 sends B_r to $V_{r,1}$, so the message complexity is $O(nn_P)$. The worst-case complexity also increases to $O(n^2 n_P)$ in the same way.

4.2 Corruption of Committee

We express a committee is corrupt when all proposing committee members are faulty or when $\frac{1}{3}$ or more verifying committee members are faulty. The necessary condition for committing blocks is that all committees are not corrupt. The minimum number of the honest P_r members for completing the round is one because it is possible that verifiers

[1] We assume $n_P \ll n_V$.

Table 4.1: Comparison of message complexity

BFT algorithm	Correct proposers	Single view change	Worst case
PBFT	$O(n^2)$	$O(n^3)$	$O(n^4)$
PBFT (committee)	$O(nn_V)$	$O(n^2)$	$O(n^3)$
HotStuff-2	$O(n)$	$O(n)$	$O(n^2)$
HotStuff-2 (committee)	$O(nn_P)$	$O(nn_P)$	$O(n^2 n_P)$

select one out of the blocks proposed by honest P_r members. However, $V_{r,1}$ and $V_{r,2}$ need $\frac{2}{3}$ majority of honest nodes to make a quorum certificate mentioned in the PBFT paper [3].

To calculate the probability of committee's corruption, we admit that the adversary cannot expect which nodes become committee members in advance. This assumption is practical because of the pseudo-randomness of random beacon [7, 13]. In addition, this calculation assumes that the IDs of the Byzantine nodes uniformly spread out and are not excessively numerous near a particular ID. In this chapter, we also consider the worst case; the adversary manipulates all of the f Byzantine nodes to corrupt the system.

Probability that P_r is corrupt (all proposers are faulty) is

$$\frac{\binom{f}{n_P}}{\binom{n}{n_P}}. \tag{4.1}$$

The numerator $\binom{f}{n_P}$ represents the number of ways to select n_P faulty nodes from f, while the denominator $\binom{n}{n_P}$ accounts for all possible committee selections. This probability monotonically decreases as n_P increases. In this case, the protocol needs to change views (move to the next round).

Probability that more than $\frac{2}{3}$ verifiers are honest for both verifying committees is

$$\left(\sum_{i=0}^{\lfloor \frac{1}{3} n_V \rfloor} \frac{\binom{f}{i}\binom{n-f}{n_V-i}}{\binom{n}{n_V}} \right)^2. \tag{4.2}$$

This equation assesses the likelihood of more than two-thirds of the verifying committee being honest. The summation accounts for all possible configurations where i faulty nodes are present, up to the threshold of $\frac{1}{3} n_V$. The complement of this probability represents scenarios where at least one of the verifying committees is corrupt, which is

$$1 - \left(\sum_{i=0}^{\lfloor \frac{1}{3} n_V \rfloor} \frac{\binom{f}{i}\binom{n-f}{n_V-i}}{\binom{n}{n_V}} \right)^2. \tag{4.3}$$

When only one of the verifying committees is corrupt, the protocol cannot commit B_r, but it still keeps safety.

Probability that two verifying committees are both corrupt is

$$\left(\sum_{i=\lceil \frac{1}{3} n_V \rceil}^{n_V} \frac{\binom{f}{i}\binom{n-f}{n_V-i}}{\binom{n}{n_V}} \right)^2. \tag{4.4}$$

This equation calculates the probability of both verifying committees being corrupted simultaneously, leading to safety violations. Safety does not hold if both verifying committees are corrupt. Therefore, the protocol needs to set appropriate n_V to avoid the corruption of the protocol itself.

Chapter 5

Evaluation of Throughput and Latency

5.1 Simulation Environment

To evaluate the proposed blockchain algorithms utilizing random committee selection, we establish a simulation environment using OMNeT++ [16], a discrete event simulation framework widely applied for network and distributed system analysis.

We configure the simulation environment to replicate a decentralized microgrid infrastructure where participants interact within a blockchain framework. Key components of the setup include nodes with a smart meter and data concentrators. A smart meter is a digital meter which records the consumption of electric power and transfers the consumption information to utilities. A data concentrator is a combination of the software and hardware units that collects information from smart meters and forwards the information [1]. In the simulation, one data concentrator handles data from a hundred smart meters. The system demonstrates a hierarchical architecture where multiple smart meters communicate with data concentrators on LTE, which convey information and electric power over the blockchain network. Exchanges between data concentrators rely on fiber-optic communication. There are a thousand nodes (smart meters) in the simulation.

We set the upper bounds for block timeout and for round timeout as 10 and 20 seconds respectively. Block timeout is the maximum interval to insert transactions into a block, and round timeout is the duration from the start of rounds until the timer fires. The transaction size is 512 bytes, and we change the default block size, showed in Table 5.1. The participants of the microgrid system always generate 1000 transactions per second (TPS).

Under this configuration, we simulate the decentralized microgrid system using blockchain and measure throughput and latency.

- *Throughput*: refers to the number of transactions a blockchain network can process in a given amount of time, typically measured in transactions per second (TPS).
- *Latency*: is the time it takes for the system to create a block and confirm and add it to the blockchain.

Byzantine nodes in the simulations behave as followings. If selected as proposers, they propose the blocks the adversary wants to commit. If they are selected as verifiers and receive the adversary's block candidate, they verify as soon as possible. If not receiving any adversarial blocks, they send an arbitrary message at 2Δ after the round starts.

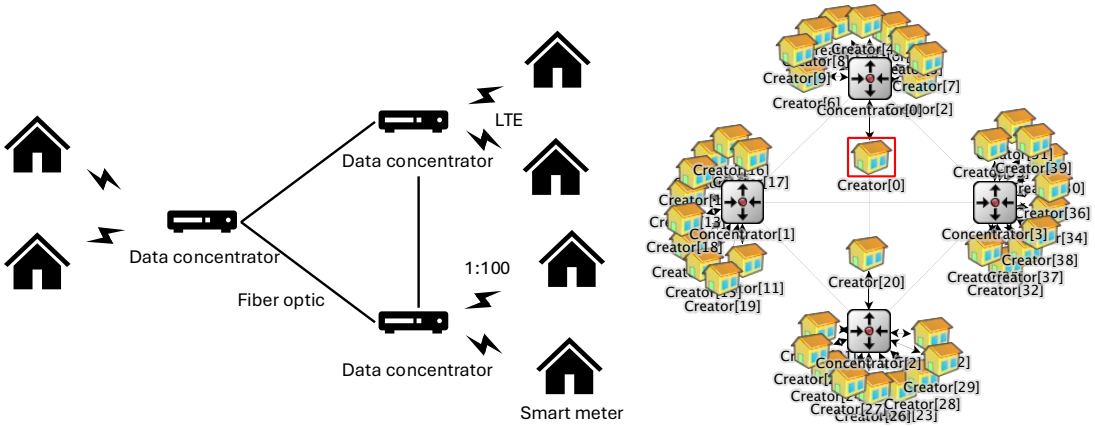

Figure 5.1: Network structure of the microgrid system simulated on OMNeT++. Each data concentrator takes charge of 100 smart meters.

Table 5.1: Simulation Settings

Number of Data concentrators	10
Number of Smart meters	1000
Block timeout	10 s
Round timeout	20 s
Size of each transaction	512 bytes
Transaction feed rate	1000 TPS
Data rate (LTE)	100 Mbps (normal condition) 5 Mbps (poor-network condition)
Data rate (fiber optic)	50 Gbps (normal condition) 200 Mbps (poor-network condtion)
Network latency (LTE)	50 ms (normal condition) 300 ms (poor-network condition)
Network latency (fiber optic)	5 ms (normal condition) 30 ms (poor-network condition)
Signature size	64 bytes
Block size	64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384 KB = 125, 250, 500, 1000, 2000, 4000, 8000, 16000, 32000 txs

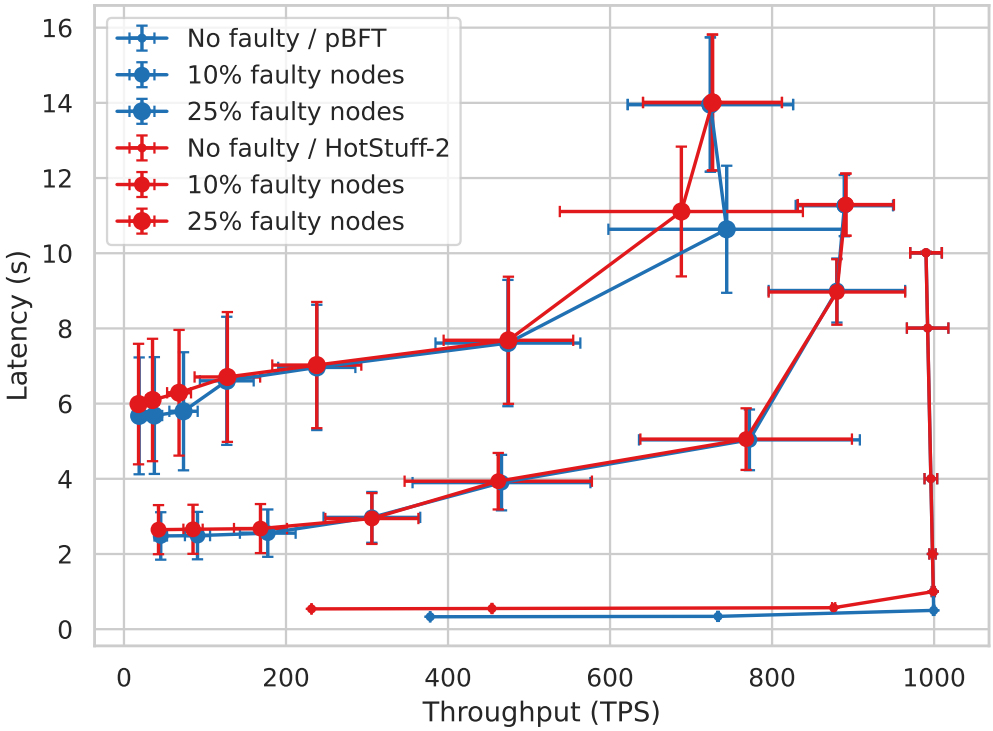

Figure 5.2: Throughput latency relationship on BFT algorithms with 1000 nodes. Each plot means a result with different block sizes. The plots overlap in the cases of the two biggest block sizes.

5.2 Baseline

In this section, we analyze the performance of PBFT and HotStuff-2, under varying block sizes and fault conditions. The simulation uses two key metrics of throughput (TPS) and latency (s) to evaluate system performance, with a network size of 1000 nodes.

Each plot in Figure 5.2 represents a different block size, showing how the system behaves as the block size changes, particularly under the following fault conditions: no faulty nodes, 10% faulty nodes, and 25% faulty nodes. When the block size is small (e.g., up to 512 KB), both PBFT and HotStuff-2 demonstrate relatively low latency but low throughput. For no faulty system, larger block sizes allow higher transaction throughput but also introduce a notable increase in latency, which indicates that the system's maximum throughput is 1000 TPS. In the case there are faulty nodes, throughput gradually increases and latency also continuously increases. In all simulations, the plots of 8192 KB and 16 384 KB overlap, which means 8 MB is too large for this system and never full of transactions. Therefore, the system with over 10% faulty nodes does not reach the throughput saturation.

5.3 Committee Selection under Low-Latency Network

We apply committee selection for verifying committees first. When $n = 1000$ and $f = 100$, the minimum size of two verifying committees with one in a million chances of admitting corruption in the same round, is 22, calculated based on Equation (4.4). In the same way, the minimum size of either of the verifying committees with one in a million chances of admitting corruption, is 51 by Equation (4.3).

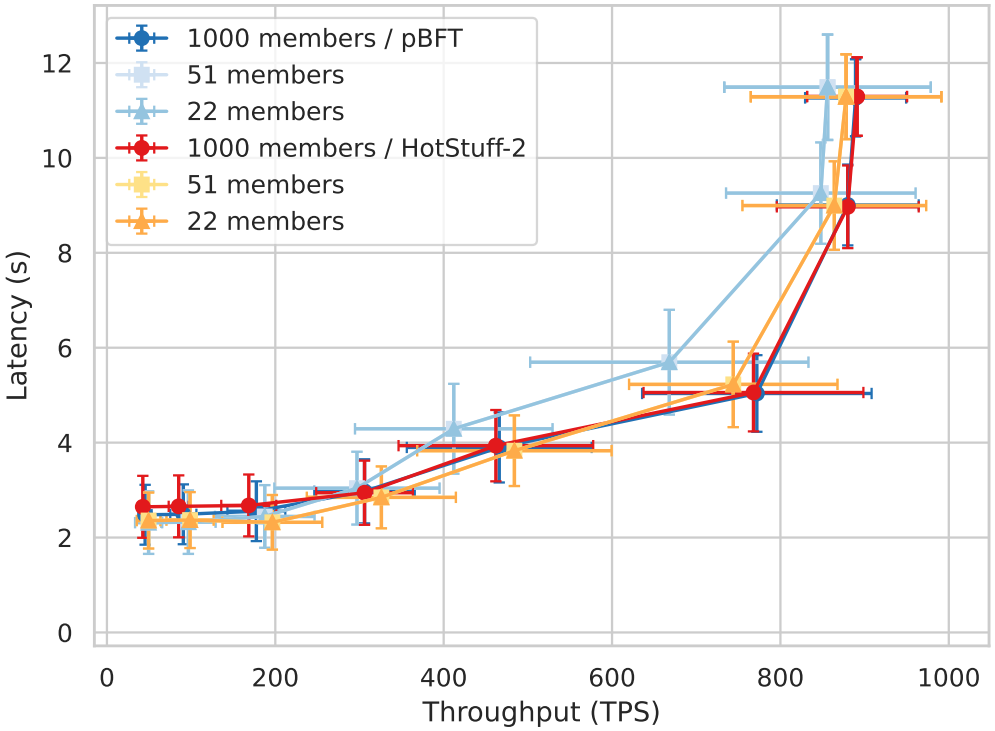

Figure 5.3: Verifying committee selection on BFT algorithms among 1000 nodes in which 10% are faulty

We analyze the impact of committee selection on the performance, varying the committee size as 1000 (all nodes), 51, and 22. Figure 5.3 shows different committee sizes do not significantly change the performance under the low-latency network. Throughput and latency gently increase as the block gets larger. The number of messages needed for consensus is smaller when the system apply committee selection to verifying committees, but the probability of a Byzantine node becoming the proposer does not fundamentally differ. Thus, the performance also does not change significantly. Regardless of the block size which is big enough to prevent corruption, view changes occur and the system adds empty blocks about once every ten times, which is roughly the same as $\frac{f}{n}$.

On top of the committee selection for verifying committees, we examine the impact of the proposing committee. Figure 5.4 is the result of applying committee selection for both types of the committees. The result shows significant improvements on both throughput and latency in case of multiple proposers. As the block size is 1 MB or more, throughout reaches saturation. Furthermore, the throughput of PBFT gets slightly worse than that of HotStuff-2 when the block size is more than 8 MB. This is because both algorithms consume much time to propose blocks and have little time to verify them by the timeout. In this case, HotStuff-2 is more efficient due to the less messages. Especially, the efficiency of PBFT with 3 proposers is almost the same as the algorithm with a single proposer if the blocks are too big.

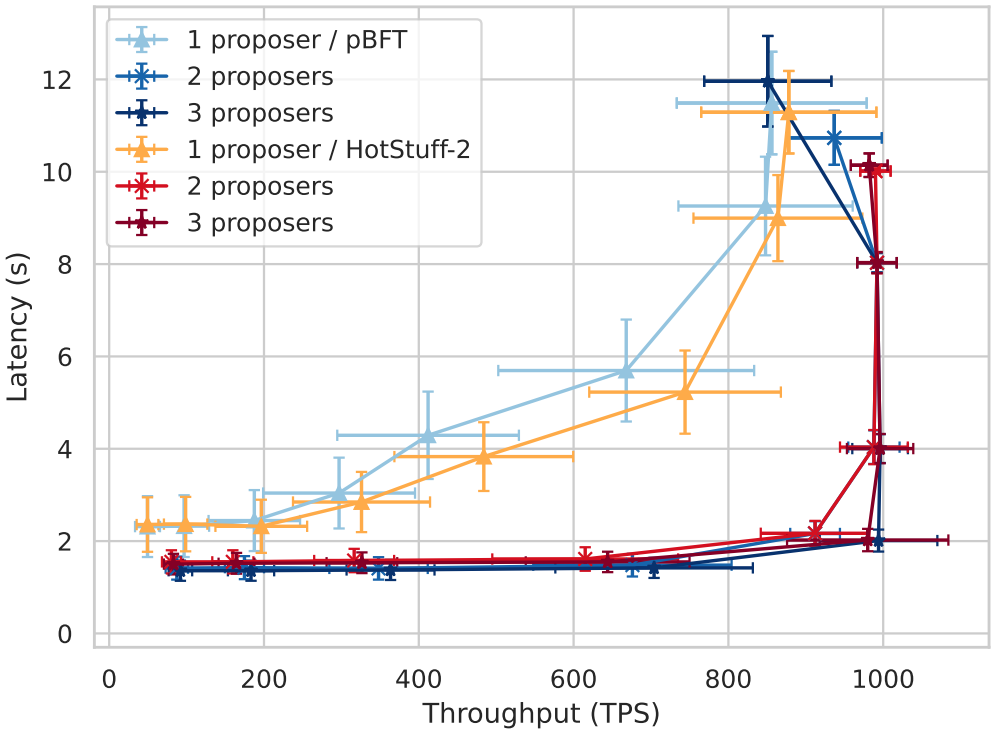

Figure 5.4: Proposing and verifying committee selections on BFT algorithms among 1000 nodes in which 10% are faulty. The size of the verifying committees $V_{r,1}$ and $V_{r,2}$ is 22.

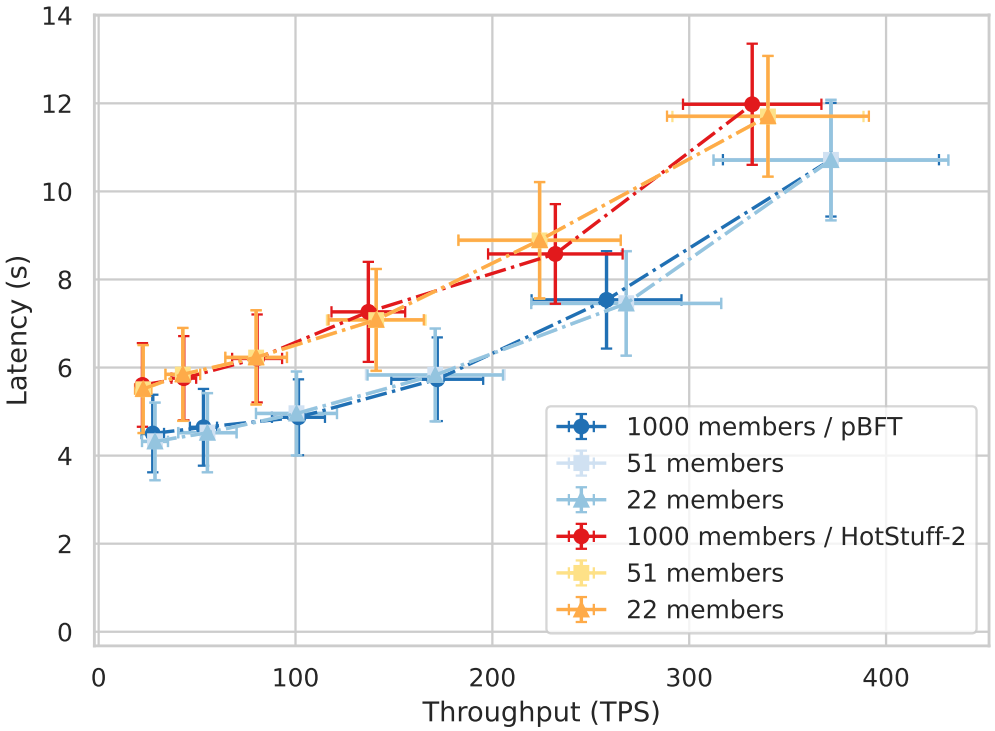

Figure 5.5: Verifying committee selection under poor network condition among 1000 nodes in which 10% are faulty

5.4 Committee Selection under High-Latency Network

We examine the case when the network speed is low, which Figure 5.5 displays. When the block size is 4096 KB or more, the systems based on both algorithms never reach consensus. Throughput and latency steadily rise as the block size; however, the performance of both algorithms is worse than those under low-latency network. PBFT relatively shows a better performance than HotStuff-2 but not a significant difference. This may be because HotStuff-2 has more time-based phases. The effect of verifying committee selection is still not considerable. Therefore, committee selection for verifiers reduces message overhead under any network condition, not harming the performance of BFT algorithms.

Figure 5.6 exhibits the depreciation of performance caused by proposing committee selection under poor-network condition. The performance gets worse according to the number of proposers. Specifically, PBFT with multiple proposers cannot commit blocks whose size is 2048 KB or more, and PBFT and HotStuff-2 in both situations cannot decide when the block is 4096 KB or more. The more proposed blocks the nodes send in poor network, the more time they spend on the first phase, and it is nearly impossible to accomplish the block verification in the round. Hence, proposing committee selection is practical when the system sets the appropriate timeouts and network situations are good, yet it is possible the selection deteriorates the performance in bad conditions.

The maximum executable block size is different between PBFT and HotStuff-2 because PBFT has more congestion in verifying phases. When blocks are large, it takes long time for each node to send proposed blocks, which makes the room for verification too small to finish. PBFT has a larger message complexity for verifying blocks; thus it cannot commit in time and drastically degrades its performance as a result.

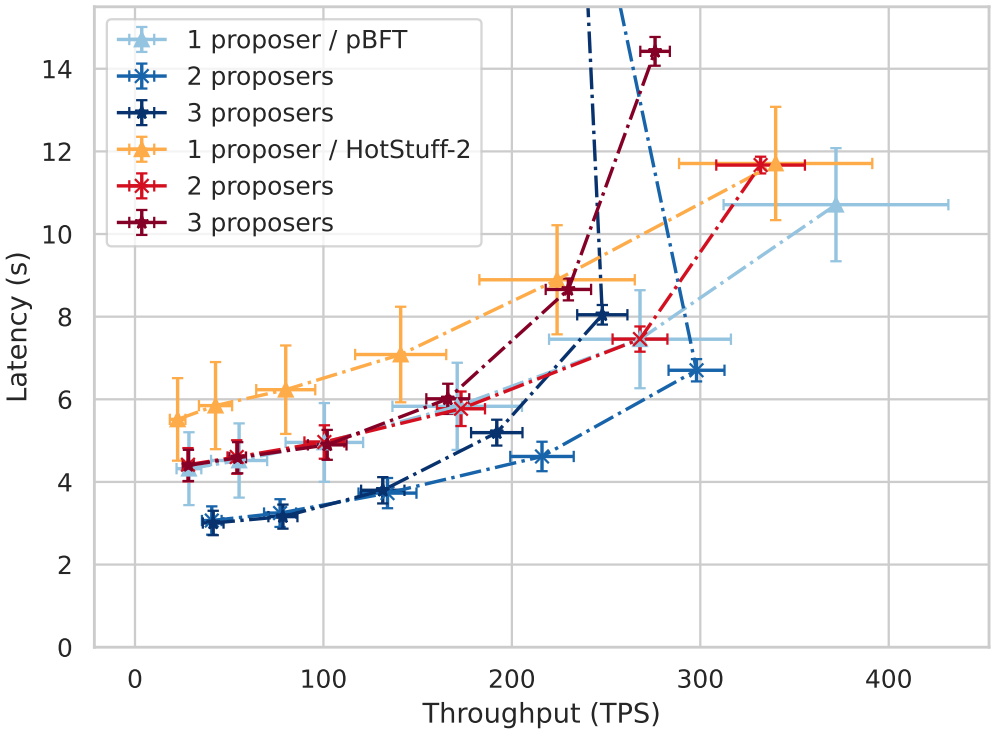

Figure 5.6: Proposing and verifying committee selections under poor network condition among 1000 nodes in which 10% are faulty. The size of the verifying committees $V_{r,1}$ and $V_{r,2}$ is 22.

Chapter 6

Conclusion

This thesis introduces a new approach to improving the scalability of blockchain systems by incorporating random committee selection into Byzantine Fault Tolerance (BFT) algorithms. The proposed method addresses the challenges of traditional consensus mechanisms, such as high communication overhead and vulnerability of traditional selection to adversarial manipulation, by integrating moving target defense principles with distributed verifiable random functions.

6.1 Summary and Assessment

The findings of this research highlight several key advancements. First, the use of committee-based consensus effectively reduces message complexity especially for PBFT while maintaining security, enabling the system to scale efficiently to larger networks. This approach delivers better transaction throughput and latency compared to conventional BFT algorithms. Additionally, the randomized and unpredictable nature of the committee selection process enhances security by minimizing the risk of adversarial manipulation, ensuring robust fault tolerance. Performance analysis shows that under low-latency network conditions, the proposed algorithms achieve higher transaction throughput and reduced delays while maintaining resilience. However, in high-latency environments, the benefits of proposing committees are diminished, underscoring the importance of tailoring configurations to specific network conditions.

These contributions prepare for more scalable and decentralized blockchain systems, making them suitable for diverse applications such as decentralized finance (DeFi), energy trading in microgrids, and other high-throughput systems.

6.2 Open Questions

Despite these advancements, an important open question remains: how to effectively apply committee selection to view-change processes. While this research focuses on enhancing consensus and reduces complexity in normal operation, view changes still rely on all network participants, which could be optimized further. Future work could explore adaptive mechanisms for dynamic committee formation during view changes, balancing the trade-offs between scalability and security. Addressing this challenge would further advance the practicality of committee-based consensus in real-world blockchain applications.

Overall, this work contributes to the scalability of BFT-based blockchain technology by providing a framework that balances efficiency and security, establishing a foundation for further research in decentralized systems.

Acknowledgment

This thesis is a summary of research conducted by the author in Dependable Distributed Systems Laboratory during my two years of master's study in Department of Computer Science, School of Computing, Institute of Science Tokyo (Tokyo Institute of Technology). I would like to express my sincere gratitude to many people who gave me guidance and advice during this research.

Firstly, I would like to express my deepest gratitude to my supervisor, Professor Xavier Défago, for his support during the years. I am interested in distributed systems, cooperative behavior of robots using distributed algorithms, and blockchain technology. Above all, I am very grateful to him for giving me the opportunity to study this field. After that, I was able to learn not only about the content of my research but also many other things, such as what to keep in mind when conducting research and the significance of the research itself. I could not have completed this research without his guidance. In addition to the research activities, he gave me opportunities to present and discuss my research in English. I am confident that this experience will be useful in the future.

I would also like to thank Associate Professor François Bonnet for his guidance on various aspects of algorithms and discussion of simulation results. In particular, he was very active in discussing the differences between logically expected results and actual results. In our weekly meetings, he kindly helped me have my English understood and created an environment in which I could make the most of opportunities to communicate and conduct research, which would have been difficult on my own.

To the students of this laboratory, all your questions and advice during my research, especially the presentation rehearsals, helped me improve and concentrate on my own research until the end. Thank you very much.

Finally, I would like to thank my family for their financial and mental support during my two years in the master's program. Thank you very much.

References

[1] Desong Bian, Murat Kuzlu, Manisa Pipattanasomporn, Saifur Rahman, and Di Shi. Performance evaluation of communication technologies and network structure for smart grid applications. *IET Communications*, 13(8):1025–1033, 2019.

[2] Gabriel Bracha. Asynchronous Byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[3] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, November 2002.

[4] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theoretical Computer Science*, 777:155–183, 2019.

[5] IETF Datatracker. IPR Details: The United States of America as represented by the National Security Agency's general license statement. `https://datatracker.ietf.org/ipr/858/`, 2024. Accessed on November 18th, 2024.

[6] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, April 1988.

[7] David Galindo, Jia Liu, Mihair Ordean, and Jin-Mann Wong. Fully Distributed Verifiable Random Functions and their Application to Decentralised Random Beacons. *2021 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 88–102, 2021.

[8] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20:51–83, 2007.

[9] Kaleido.io. Quorum Blockchain. `https://www.kaleido.io/blockchain-platform/quorum`, 2024. Accessed on November 4th, 2024.

[10] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. *Cryptology ePrint Archive, Paper 2016/889*, 2016.

[11] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *https://www.peercoin.net/read/papers/peercoin-paper.pdf*, 2012.

[12] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[13] Jinghui Liao, Borui Gong, Wenhai Sun, Fengwei Zhang, Zhenyu Ning, Man Ho Au, and Weisong Shi. BFTRAND: Low-latency Random Number Provider for BFT Smart Contracts. In *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 389–402. IEEE, 2024.

[14] Dahlia Malkhi and Kartik Nayak. HotStuff-2: Optimal Two-Phase Responsive BFT. *Cryptology ePrint Archive*, 2023.

[15] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *https://bitcoin.org/bitcoin.pdf*, 2008.

[16] OMNeT++. OMNeT++ Discrete Event Simulator. `https://omnetpp.org`, 2024. Accessed on December 14th, 2024.

[17] LF Decentralized Trust. Hyperledger Fabric. `https://www.lfdecentralizedtrust.org/projects/fabric`, 2024. Accessed on November 4th, 2024.

[18] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. HotStuff: BFT Consensus with Linearity and Responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, page 347–356. Association for Computing Machinery, 2019.