OptiShard: An Optimized & Secured Hierarchical Blockchain Architecture

Shyamkumar Rajesh Kantesariya

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science (Computer Science) at

Concordia University

Montreal, Quebec, Canada

December 2019

© Shyamkumar Rajesh Kantesariya, 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Shyamkumar Rajesh Kantesariya

Entitled: OptiShard: An Optimized & Secured Hierarchical Blockchain Architecture

and submitted in partial fulfillment of the requirements for the degree of

M. Comp. Sc.

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

Dr. A. Hanna

_____ Examiner

Dr. T. Eavis

_____ Examiner

Dr. H. Harutyunyan

_____ Supervisor

Dr. D. Goswami

Approved by _____

Dr. L. Kosseim
Graduate Program Director

_____

Dr. A. Asif
Dean, Gina Cody School of Engineering and Computer Science

Date _____

Abstract

OptiShard: An Optimized & Secured Hierarchical Blockchain Architecture

Shyam Kantesariya

Blockchain has become an emerging decentralized computing technology for transaction-based systems due to its peer-to-peer consensus protocol over an open network consisting of untrusted parties. Monolithic architecture supporting Bitcoin and other major alt-coins are inherently non-scalable. In recent past, some hierarchical approaches have been explored to shard the decentralized blockchain to improve scalability. However, there is no discussion in the literature about how to determine an optimal shard size to maximize performance and how the presence of malicious or faulty nodes can impact on choosing an optimal shard size. To address these issues, this thesis presents a sharding scheme and validation protocols for a hierarchical blockchain architecture named OptiShard. The hierarchy divides the network nodes into multiple disjoint shards and the majority of transactions are distributed among these shards in non-overlapped fashion. Optimal shard size is determined based on two parameters: performance and correctness of transaction validation in the presence of malicious or faulty nodes. OptiShard provides guaranteed majority of good shards, subject to a maximum allowable threshold of faulty nodes, by choosing the right shard size. It also provides a mechanism for identifying faulty shards, through the overlapping of a small fraction of transactions across all the shards, and discarding all their transactions. Experimental results performed on up to 800 Amazon EC2 nodes conform to the theoretical analysis and also exhibit the scaling characteristics of OptiShard.

Table of Contents

List of Figures

List of Flowcharts

List of Protocols

List of Algorithms

List of Tables

1 Introduction

Centralized model has been dominating the software applications for a long time, due to its direct control and governance over the information from a single central server. A central server is responsible to validate and fulfill every client request. It maintains a master copy of the database to persist the information. Thus, the authority maintaining central server is responsible to establish the digital trust among all the system users. Such a centrally organized system offers essential governance over confidential information; is easy to manage; and comparatively less complicated to implement.

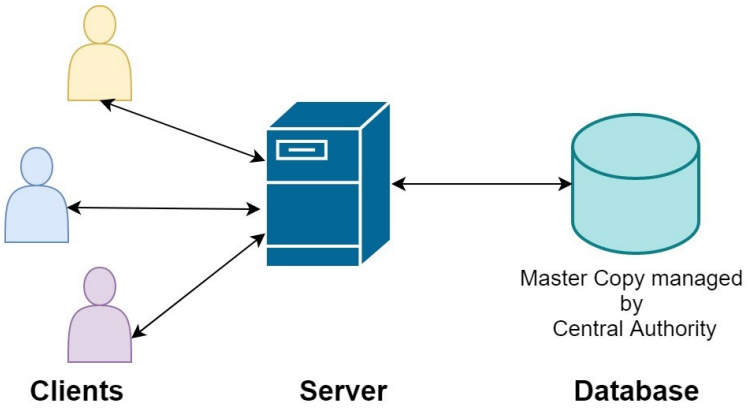


Figure 1.1 Architecture of a centralized system

Figure 1.1 depicts the architecture of a centralized system. Centralized systems offer better consistency and control over the information, however central server is the single point of failure. The whole system becomes inaccessible in the event of database or server crash. Secondly, an attacking adversary needs to target only one node to corrupt the whole system. Lastly, the architecture is not scalable as the overall performance degrades with the increasing number of clients accessing the system beyond a certain point. Hence, it became inevitable to replicate server and database for the reason of fault tolerance and workload distribution. Such an advanced system with multiple server instances, each responsible to maintain only a subset of the aggregated information, is formally known as distributed system [1].

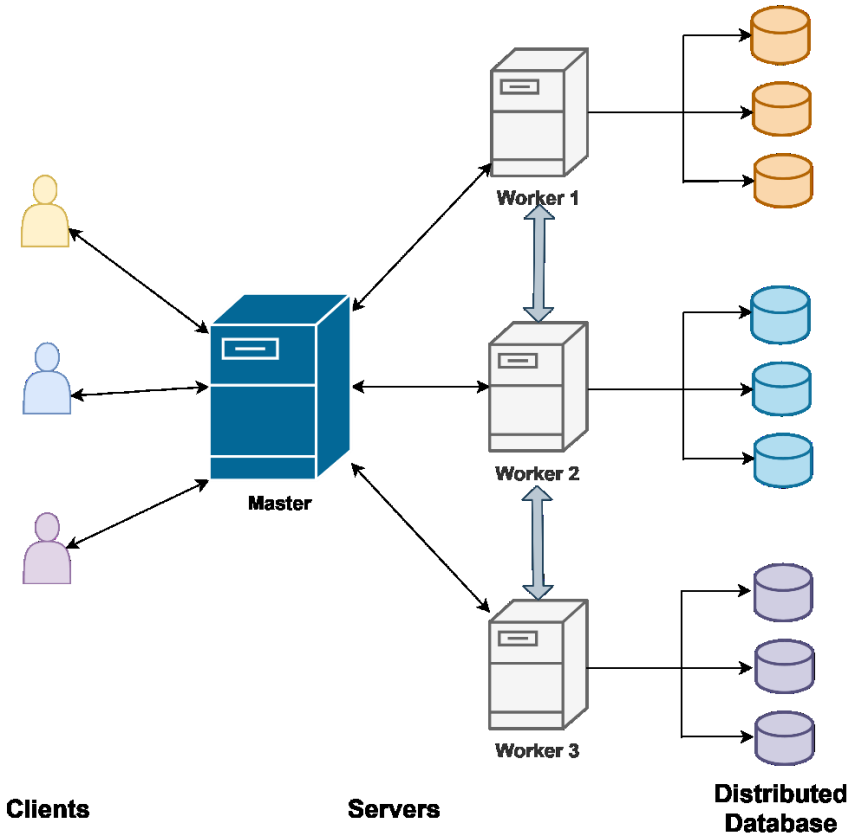


Figure 1.2 Architecture of a distributed system

Figure 1.2 shows the architecture of a distributed system with multiple server instances. Each server instance stores and replicates a subset of the information. System is horizontally scalable because computation and storage both are spread across the server instances. A master node is responsible for workload distribution and coordination. Replication of database instance offers reliability and fault tolerance. We may also replicate server instances to achieve similar fault tolerance at server level, however we have omitted it here to keep our discussion simple. Most of the distributed systems deploy multiple master nodes to overcome a single point of failure. Thus, distributed systems are horizontally scalable due to distribution of the workload; and do not have a single point of failure.

Distributed systems are favorable to deploy a software application in private environment consisting of all the trusted nodes. However, correctness of the system poses major concern, if the network has presence of public nodes (i.e. not governed by trusted authority). For example, Worker 1 in Figure 1.2 has ability to corrupt a subset of the information without the consent of any other nodes in the system.

In 2008, an anonymous person or a group of people called Satoshi Nakamoto released a paper [2] with title "Bitcoin: A Peer-to-Peer Electronic Cash System" and proposed a unique decentralized transaction processing architecture for a system of cashless digital currency, also known as cryptocurrency that offers reliability even in the presence of public nodes. It is unique because it completely eliminated the need of a central governing authority (i.e. bank) to approve/validate transactions. It is decentralized because clients can query or submit a transaction to any node; and each node is authorized to modify the information. Nodes participating in the network don't trust each other, still collectively they generate a single version of the truth by following a decentralized peer-to-peer consensus protocol. Additionally, the system maintains a complete history of all the transactions processed, which helps to backtrack the ownership of the digital currency. Such a decentralized network in the absence of a central trusted authority, made up of nodes without established trust, each storing a complete history of all the transaction and generating a single version of truth by following a consensus protocol, is known as Blockchain architecture. Thus, Blockchain can be defined as "*A peer-to-peer distributed ledger forged by consensus among the nodes participating in the network*"[3].

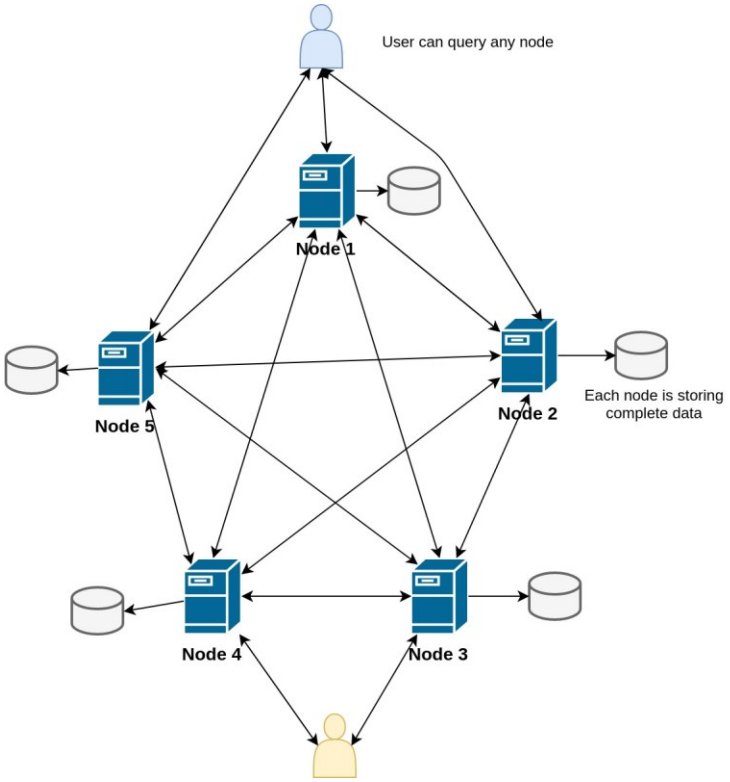


Figure 1.3 Architecture of a peer-to-peer decentralized system

Figure 1.3 shows architectural diagram of a peer-to-peer decentralized system. In traditional transaction system, the central trusted authority maintains a registry (i.e. central database) of cash belonging to each user. For the moment, let's consider cash as an asset. Hence, if a user wants to transfer the ownership of cash assets, it has to be approved by the central authority. Central authority validates the ownership of the asset and approves the transaction only if the asset belongs to the payer. Once a transaction is approved, ownership of the associated asset is transferred to the payee and updated in the registry; which effectively doesn't allow a payer to transfer the ownership of an asset to multiple payees. In a peer-to-peer decentralized system like Bitcoin [2], there is no central authority to maintain a registry, therefore a user can theoretically issue multiple transactions transferring ownership of an asset to multiple payees.

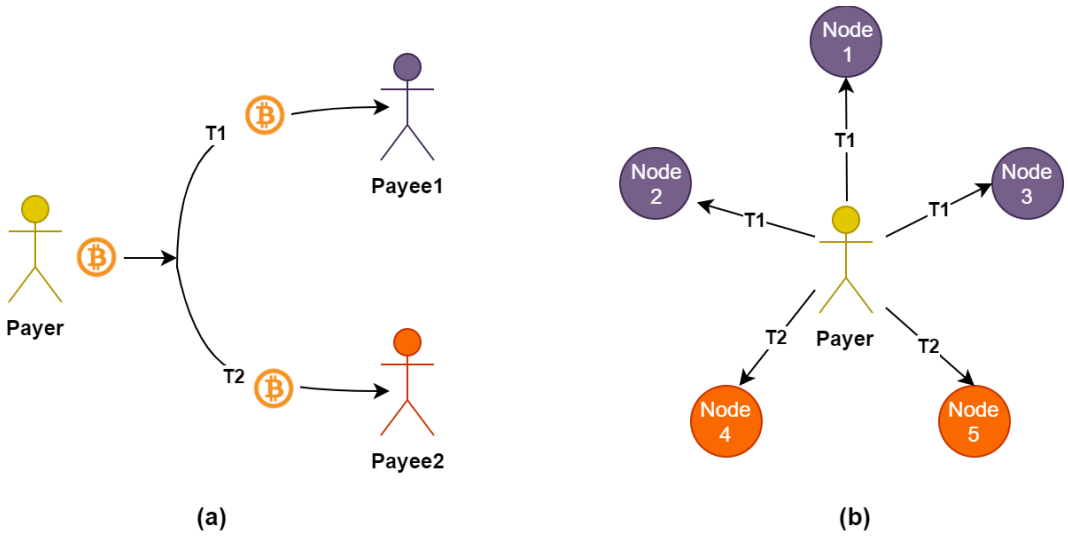


Figure 1.4 Double spend attack in peer-to-peer decentralized architecture

In Figure 1.4 (a), a payer initiates two transactions T1 and T2 to transfer the same digital asset to two different payees and smartly submits T1 to Node 1,2 and 3; T2 to Node 4 and 5 as shown in Figure 1.4 (b). In this situation, a part of the network approves T1 transaction and the other part approves T2 transaction; which is also referred as infamous *double spend attack* [2] in cryptocurrencies. Due to that reason, a transaction has to be approved by the consensus among majority of the nodes part of the peer-to-peer decentralized network. Consensus among majority of the nodes either approves T1 or T2 but both. Additionally, the network should offer the guarantee that a malicious node cannot alter the committed transactions. Therefore, transactions are committed in an immutable data structure called *Blockchain*.

1.1 Blockchain

Block is a single unit of storage in blockchain, that consists of information regarding a set of committed transactions. Size of a block can be up to predefined maximum limit, i.e. 1 MB for Bitcoin [2]. Each block has reference to its previous block, which collectively forms a chain of blocks, hence called as Blockchain. The very first block is called as genesis block [4], which is a special case as it omits the reference to previous block. Network adds a block to the chain only if it contains valid eligibility proof, approved by consensus among majority of the network nodes, to be the next block in the chain.

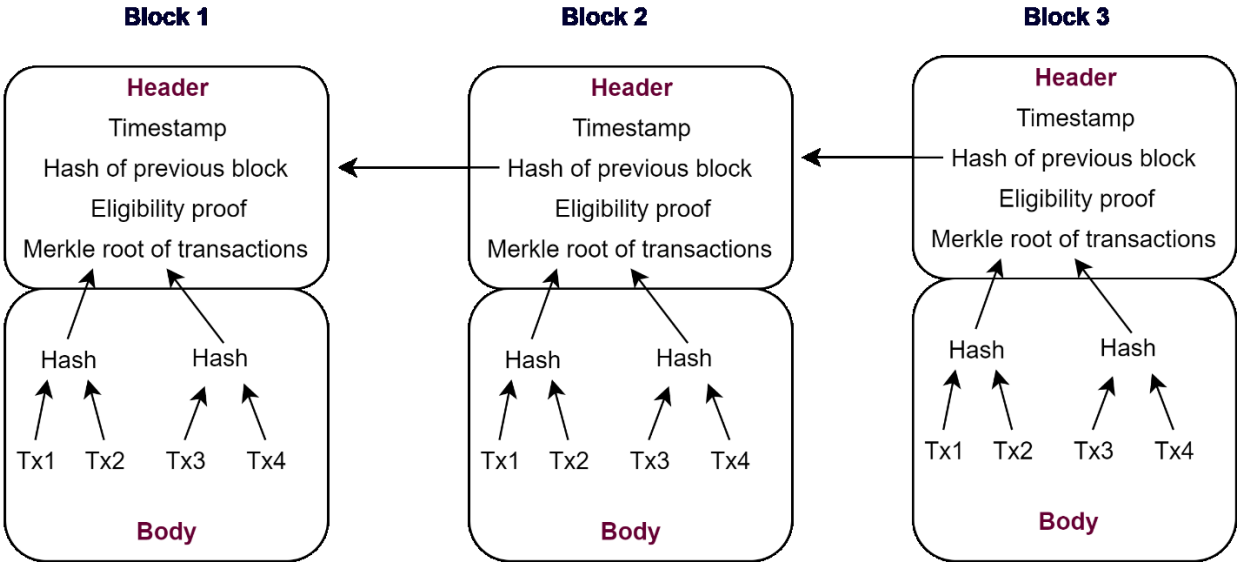


Figure 1.5 Blockchain data structure

Figure 1.5 represents a blockchain consisting of three blocks. Components within a block may vary based on the implementation (e.g. Bitcoin block header additionally contains version number and target hash value), therefore we have demonstrated minimal required fields in two sections as following:

- Header
    - Timestamp
    - Hash reference of previous block
    - Eligibility proof
    - Merkle Tree [5] (one kind of hash tree as shown in Figure 1.6) root of the transactions.
- Body
    - Details of the committed transactions

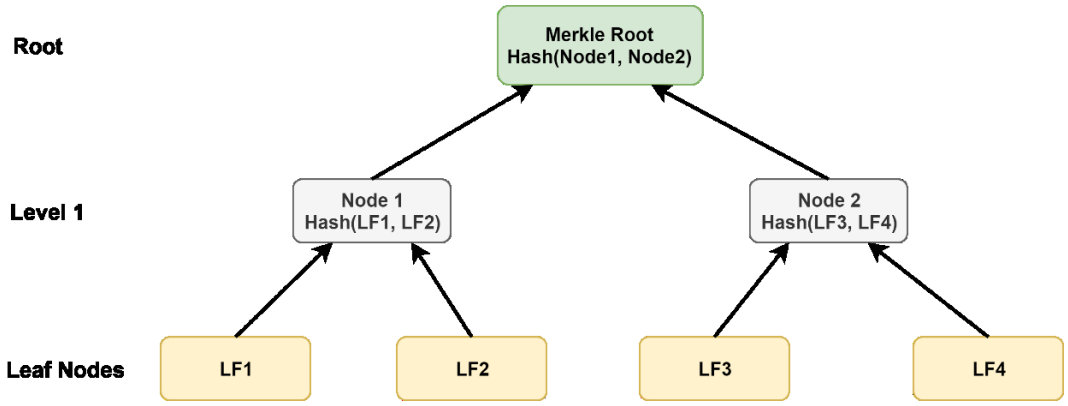


Figure 1.6 Merkle Tree composed of 4 leaf nodes. Each internal node is hash of both the children

There are multiple approaches to decide eligibility of a block to be the next block in chain. Bitcoin and other cryptocurrencies follow one such technique, Proof-Of-Work (PoW) [2], wherein nodes compete with each other to solve a mathematical problem and the solution is considered as a valid eligibility proof. Every node spends some CPU cycles to solve mentioned problem. Whosoever succeeds first to solve it, gets the privilege to create next block, hence called as leader node. PoW is further discussed in Chapter 2. Leader broadcasts the newly created block to the whole network. Receiving node validates transactions and eligibility proof of the block and add to the chain on successful validation. A block is considered to be committed when it is added to the chain by predefined majority (i.e. 51% in case of Bitcoin) of the nodes part of the network.

PoW uses a randomized approach to select the leader to generate the new block. It is possible that two nodes are selected as leaders simultaneously and they broadcast different versions of new block with different sets of committed transactions. Moreover, two nodes working on different states of the chain may also propose different versions of the next block due to hash reference of previous block. Such situations can lead to a *fork* [2].

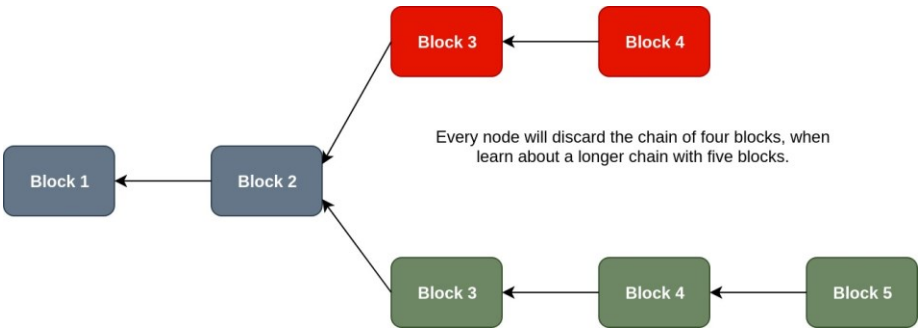


Figure 1.7 Fork in the chain

A fork in the chain happens when a blockchain ledger diverges into at least two potential paths. Nevertheless, this fork exists only for a short period of time; subsequently as shown in Figure 1.7, the longest chain is considered as the valid ledger and the other path is discarded by majority of the nodes.

Blockchain is an immutable data structure because once a transaction is committed in a block then it becomes nearly impossible for an adversary to alter it after a certain number of additional blocks are added to the chain. If a malicious node tries to alter a committed transaction then it needs to modify the entire subsequent blocks due to hash references.

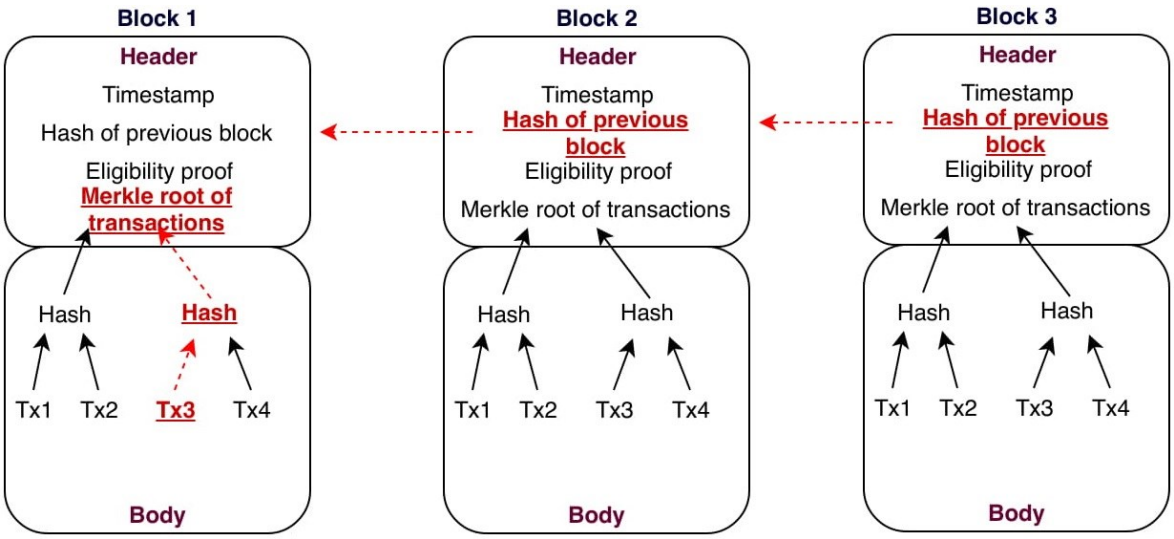


Figure 1.8 Alter a committed transaction in Blockchain

As shown in Figure 1.8, if an adversary tries to alter the committed transaction Tx3 in block 1, then it leads to change in the Merkle root and hash reference value of block 1, that consequently requires to modify all the consecutive blocks (i.e. block 2 and 3 in Figure 1.8). While the malicious node is catching up on the modified chain, remaining network keeps adding new blocks on the legitimate chain. Hence, in practice, a malicious node will not be able to make its chain longest and this makes the blockchain ledger immutable.

After discussing the blockchain data structure and architecture, we discuss the major challenges of blockchain architecture in the following section.

1.2 Problem Statement

Use cases of blockchain are found in various areas including cryptocurrencies, smart contracts, digital identity, supply chain, decentralized storage to name a few [6]. Blockchain has proved its correctness by hosting Bitcoin network. At present there are hundreds of decentralized applications built upon blockchain architecture; formally called with an acronym dApps [7]. However, Bitcoin is the largest such application, hence we discuss the major challenges of blockchain in the context of Bitcoin.

Firstly, every node is simultaneously burning CPU cycles to solve a mathematical problem required to add a new block; this consumes a considerable amount of electricity. When a node finds a solution (i.e. becomes the leader), it gets the privilege to create the next block in chain and the CPU cycles of all remaining nodes are wasted. It has been estimated that the entire bitcoin network consumes more energy than required to run a small country like Switzerland [8].

Secondly, if a node or group of nodes can manage 51% of computing power then collectively they have the ability to alter a committed block. As they have half of the computing power, they can quickly regenerate the hash references and catch up with the legitimate chain. While regenerating an altered chain, they keep it within their malicious network and the moment they have the longest chain, they broadcast it to the remaining nodes. The entire network will accept the modified chain, it being the longest, and discard their own copies. Such an attack is known as *51% attack* [9], discussed further in section 2.1. Bitcoin has other major security threats too (e.g. Sybil attack [10]), which are beyond the scope of our discussion.

Finally, the monolithic architecture itself is not scalable due to the considerable costs involved in computation, communication, and consensus. The current bitcoin network processes on an average 7 transactions [11] per second compared to mainstream transaction processing platforms like VISA, which has the ability to process up to 56,000 transactions per second [11] [12]. Hence, it is inevitable to reengineer the blockchain architecture and its consensus protocol so that it becomes scalable [11] [13] to compete with the market leaders and at the same time runs at economic operational cost; this is a necessary step for blockchain to be adopted for enterprise use cases and remain competitive with other market leaders.

1.3 Contribution

In this thesis we propose a hierarchical blockchain architecture and consensus protocol, named ***OptiShard***, which addresses the previous challenges in the adoption of blockchain and issues not addressed by other related works, more specifically on determining an optimal shard size while considering both performance and reliability in the presence of malicious and faulty nodes.

- The sharding scheme determines an optimal shard size based on two parameters: performance and correctness of transaction validation.
- The consensus protocol within each shard is a centralized variant of PBFT [14].
- Requirements of PBFT are used to calculate an allowable shard size that guarantees that majority of the shards are good, based on a predefined majority.
- Mathematical approach to calculate a shard size that offers optimal system performance, yet guarantees that majority of the shards are good at all the time.
- Majority of transactions are distributed among shards in non-overlapped fashion, i.e. one-to-one mapping of a transaction to a shard. However, a small fraction of transactions is mapped in overlapped fashion, i.e. one-to-all mapping of a transaction to shards; this is to identify malicious/faulty shards by comparing validation results of overlapped transactions across shards.
- OptiShard is a hybrid of decentralized and peer-to-peer. Peer-to-peer links across the network shards enable efficient exchange of information without broadcasting.
- Though the current protocols use PoS [15] instead of PoW [2] to handle energy inefficiency, they can be modified to incorporate any other technique, if needed, without affecting the sharding scheme.
- Experiments performed on up to 800 Amazon EC2 nodes to measure system performance across various dimensions to exhibit the scaling characteristics of OptiShard. We also compare the experimental performance with the calculated theoretical performance to exhibit the correctness of our theoretical analysis.

- Probabilistic experiments to simulate the probability of various randomly generated events on a scale of 1M iterations for each configuration.

1.4 Thesis Outline

The rest of this thesis is organized as follows: Chapter 2 and 3 discuss the background and related work respectively. Chapter 4 presents the proposed architecture and protocols of OptiShard in detail. Chapter 5 demonstrates our experimental results and compare them with the theoretical analysis. Finally, chapter 6 concludes the thesis by summarizing our contribution and discusses potential applications of OptiShard in addition of avenues for future research.

2 Background

We can refine the definition of Blockchain as an immutable data structure to represent an ordered chain of blocks which are time stamped and connected in such a way that each block has reference to its previous block. Every node part of the network maintains the complete chain which is called as the ledger. Thus, blockchain network consists of a distributed ledger agreed at all the time by consensus among majority of the nodes [16]. If admission to the network is governed by an authority then we categorize that as private, otherwise as public blockchain [17].

Based on the eligibility criteria to add a new block, there are multiple algorithms for the consensus. In the following, we discuss some of them.

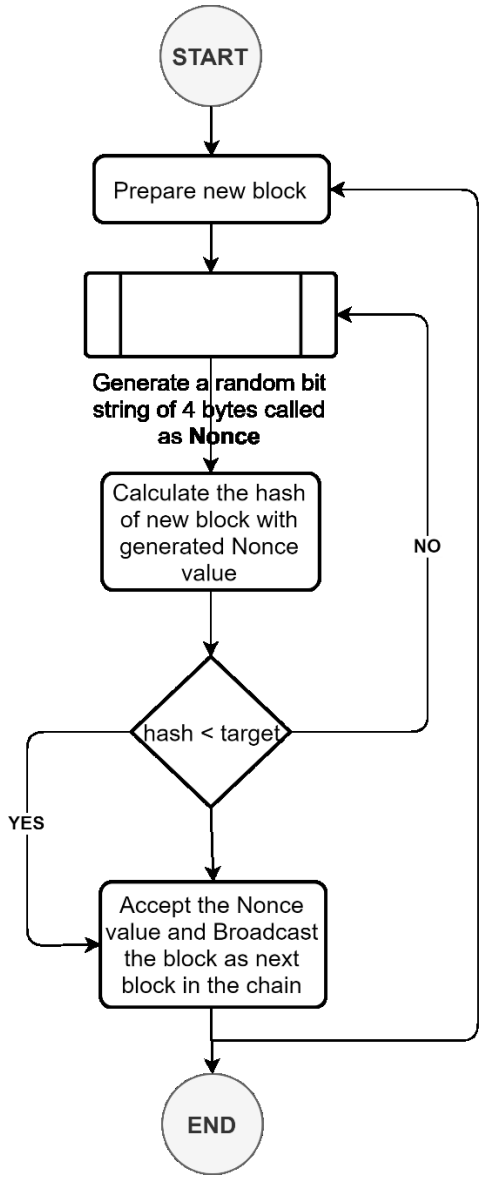
2.1 Bitcoin and Proof-of-Work (PoW)

Bitcoin was introduced to offer a purely peer-to-peer electronic cash system, wherein user can send and verify payments without the need of any intermediate trusted authority. It was the first ever system implemented based on cryptographic proof that allows two willing parties to transact directly without the presence of a trusted third party.

It follows the Nakamoto consensus protocol [2] for the consensus in a monolithic architecture. As per Nakamoto protocol, every node competes to generate a new block by consuming some CPU cycles required to solve a mathematical problem, which involves creating a valid hash value of the new block which is less than some predefined value; this is also called the Proof-of-Work (PoW). During our discussion, we refer PoW to Nakamoto PoW.

For a block to be valid, it must hash to a value smaller than a predefined value known as *target* [18], viz., hash value should contain certain number of leading zeros; this condition is known as *PoW*. Target value is set in such way that network can produce one block every 10 minutes. Lower the target value implies more difficult to produce the PoW. All the fields in a block are fixed except 4 bytes known as *nonce*, hence a node has to keep trying different values of nonce to produce PoW. Bitcoin uses SHA256 [19] as hashing algorithm. Efforts required to generate a valid hash is

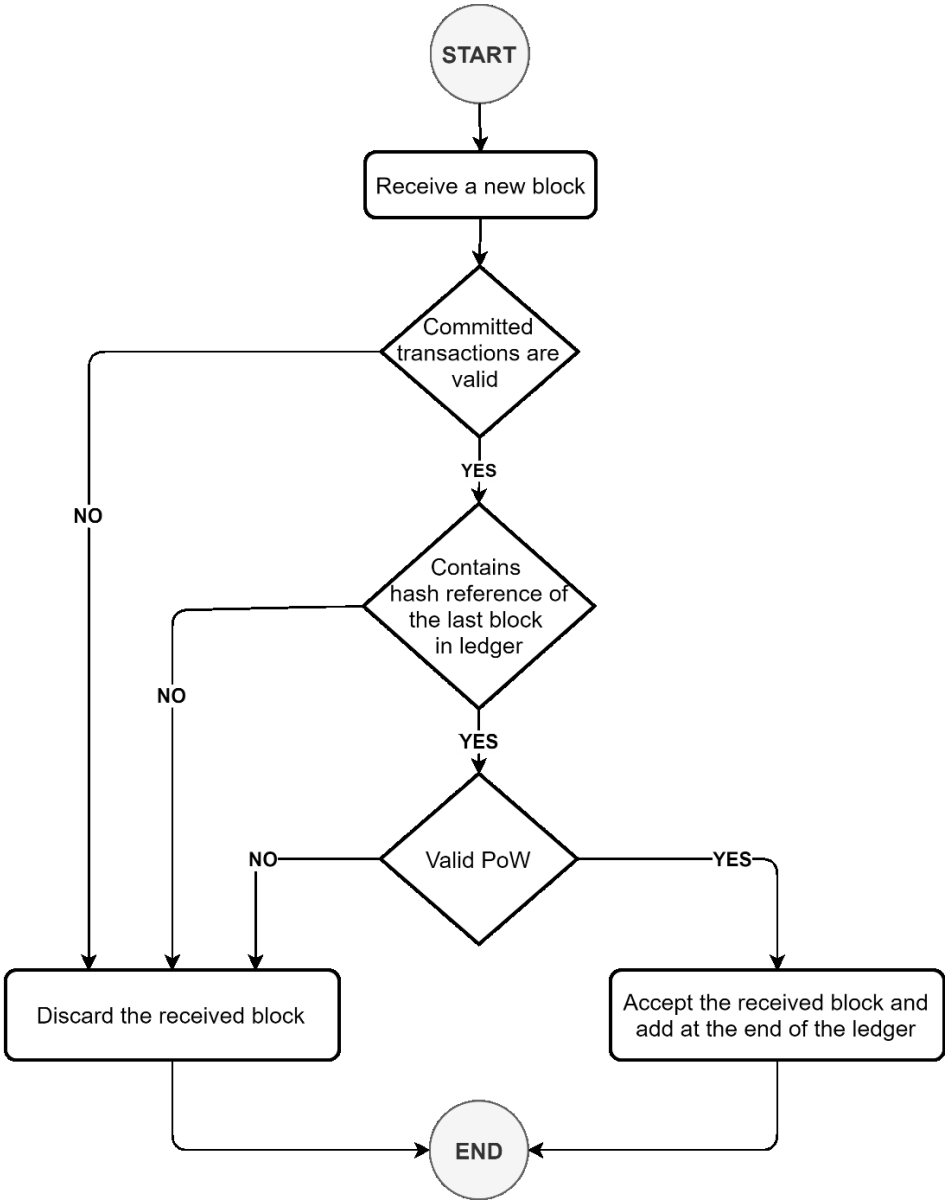
exponentially proportional to the number of leading zeros required in PoW. Ability of a node to generate new block depends on how quickly it can generate PoW. A node with higher computing power can try out difference nonce values at higher rate, therefore it has higher probability of finding PoW.



Flowchart 2.1 Bitcoin mining

This whole process of finding PoW is called as *mining process* [3] and node as a *miner* [3]. Flowchart 2.1 shows the steps of Bitcoin mining. A miner is rewarded some currencies as an incentive to generate new block. Therefore, the motive of each miner is to generate new block as quickly as possible to gain the incentives. A miner with higher computing power can quickly try

out different values of nonce, unless it finds the PoW. Hence, considering the obvious advantages of higher computing power in mining process, dedicated Graphics Processing Unit (GPU) [20] based hardware is developed by multiple leading organizations to increase the probability of generating PoW earlier than other competing miners. Once a miner generates new block, it broadcasts to the network. Each recipient node does some validation before adding a new block in the chain.



Flowchart 2.2 Validate received block

When a node receives new block from the network, it follows the steps mentioned in Flowchart 2.2 and does following validations before accepting it as the next block in the ledger:

- *Valid transactions*: all the transactions committed within the block should be valid
- *Contains hash reference of the last block*: "Hash reference of previous block" field in the received block should be equal to the hash reference of the last block in the ledger
- *PoW*: Hash value of the received block should be smaller than the target value

A block (and transactions within it), is considered committed only when at least 51% of the network nodes accept and commit the block. Once a new block is committed by the network, the leader node receives some Bitcoin(s) as incentives for investing CPU cycles to generate PoW. If multiple nodes broadcast different versions of new block then as shown in Figure 1.7, the block which is part of the longest chain, will be considered as committed.

Complexity of PoW makes whole chain secured. As demonstrated in Figure 1.5, if a node tries to alter a committed transaction, it must recalculate the PoW to regenerate hash references in all the subsequent blocks. While malicious node is recalculating PoW of altered chain, honest nodes will continue mining on the legitimate chain. Hence, honest nodes wouldn't be able to make altered chain longer than the legitimate, which secures Bitcoin blockchain from any malicious entity. However, PoW consensus also has certain major drawbacks as discussed in section 1.2.

2.2 Proof-of-Stake

Proof-of-Stake (PoS) [15] was first introduced by Peercoin [21] [22] as an alternative to energy inefficiency of PoW. Instead of miners burning CPU cycles to solve a mathematical problem, they invest some currency on stake which derives their chances of creating a new block; therefore, called as *validators* not miners. Any node who holds base currency of blockchain can become a validator. Validators send a special type of transaction to the network to reserve their stake values. Probability of a miner to become leader depends on its economic stake value, not computing power. The leader is chosen over a time duration (e.g. 10 seconds) in pseudo-random way based on its stake value. If the node, who invests highest stake currency, becomes eligible to create the new block all the time, then it centralizes the ability to create new block to the richest nodes. Hence, there are multiple variations adopted by various blockchain platforms to choose the leader.

Peercoin [21], the first cryptocurrency network to implement PoS along PoW, considers coin age to calculate the stake value. Coin is a unit of digital currency which carries some value and

can be transferred to others by executing cryptocurrency transactions [23]. In Peercoin PoS protocol, stake amount gets multiplied by coin age during the consideration of next candidate to generate the new block. Coins not spent for last certain period of time can only compete as stake amount. Additionally, once a node generates a new block, its stake coins' age is reset to zero, which demises any advantage to the richest nodes.

Advantages of PoS over PoW:

- Operational cost is very low as it avoids consumption of large amount of electricity in order to decide the leader node to create new block.
- Due to low cost for a validator to create new block, it requires less reward coins in order to motivate participants to keep participating in the network.
- As validators are reserving some currency as stake amount, it motivates them to secure the blockchain. If a validator is found to be malicious then it loses its stake value and ability to create new blocks, which makes the whole system secured.
- Its way more expensive for malicious validators to carry out a 51% attack than PoW blockchain, because it requires to control 51% of the currency on the platform.

As we have seen multiple advantages of PoS over PoW, many blockchain systems have started adopting it. However, there are few things to be taken care while implementing PoS based algorithms.

- To begin with, algorithm should be protected against centralization of power around wealthy nodes. There are different approaches to tackle this challenge. We discussed one of them to consider coin age and resetting it to zero, after it is utilized to calculate stake value.
- Secondly, a group of nodes may keep rotating the same currency among themselves to generate higher stake value. Due to higher stake value, they increase their probability of mining next block. Nxt [24] blockchain has implemented a PoS based algorithm, which allows only stationary currency with a node for at least last 1440 blocks to put on stake.
- Finally, validators should be kept motivated to remain active on the network and participate in the validation process.

2.3 Delegated Proof-of-Stake (DPoS)

Daniel Larimer developed DPoS [25] while looking for an alternate of the energy wasteful PoW. In DPoS algorithm anyone holding the base currency is considered as stakeholders. Validators are the nodes, who validate the transactions and create new blocks. It is not necessary for a validator to be a stakeholder, hence validators may also have no stake value in the system. Stakeholders vote for validators and weightage of stakeholders' vote is in proportion to their stake value. In other words, higher stake value has higher weightage of vote. Only top N validators, based on their votes, are allowed to create new blocks. If stakeholders find a validator malicious then they can withdraw their vote, hence validators may change based on their ability to earn stakeholders' trust. Validators receive rewards for validating transactions whereas stakeholders receive dividend for electing validators.

BitShares [26] is the first blockchain system to implement DPoS. In BitShares system, validators are called as *witnesses*. Every witness gets a chance to produce new block periodically, (i.e. every 2 seconds). If a witness node doesn't generate new block during its time slot, then it loses its chance and next witness generates the consecutive block. Stakeholders vote for witnesses once a day. Hence, witness nodes may change once a day during the normal operation.

Advantages:

- As we are allowing only a set of nodes (i.e. validators) to generate new block, consensus process is highly simplified. Due to that, new block generation rate is very high, which eventually increases the throughput, meaning we can validate more transactions within a given timeframe.
- As it avoids PoW, DPoS is more environment friendly and energy efficient; Secondly, operational cost is very less compared to PoW, hence system can hold validators' interest even with less incentives.
- Any node is eligible to be a validator even without holding any currency on stake.

Disadvantages:

- Essentially, we are limiting the ability to create new block to a set of validator nodes, which in turn creates more centralization than any other blockchain platform allowing equal

opportunity to any node at all the time to create new block. Theoretically, stakeholders may plot 51% attack by influencing the voting process, though practically its expensive.

- Stakeholders with higher stake has higher weightage and influence on voting result. Also, they receive higher dividend as it is in proportion to stake value. Hence, their voting power keeps increasing as compared to the one with lesser stake value.
- As in any democracy, it requires the participation of stakeholders during the voting process. If genuine stakeholders don't participate in the voting process, then there are high chances of malicious validators be chosen to validate the transactions.

2.4 Practical Byzantine Fault Tolerance (PBFT)

Byzantine fault is a category of faults when system produces arbitrary results, hence it is the most difficult task to handle byzantine faults in a distributed system. A very popular example to explain Byzantine faults is the Byzantine Generals problem [27]. In 1999, Miguel Castro and Barbara Liskov, from MIT laboratory, proposed a state-machine replication protocol that can tolerate Byzantine faults in asynchronous networks, and called it as Practical Byzantine Fault Tolerant (PBFT) protocol [14].

If the system consists of n nodes, each acting as a replicated state machine, then PBFT protocol offers guaranteed correctness, provided maximum faulty nodes are less than $\left\lfloor \frac{n-1}{3} \right\rfloor$ at all the time. A system is considered to be correct when it offers liveness (i.e. client request should be responded within a bounded time) and safety (i.e. though system is replicated, it should work as if a centralized system). In other words, for a system to tolerate f faulty nodes, it must host $3f + 1$ nodes (i.e. $n > 3f$). The reason being is, there could be f nodes, who are correct but their response is not received by client due to asynchronous network; and other f nodes, who responded are faulty; therefore, there must exist at least $f + 1$ additional correct responses to generate correct result.

It follows hierarchical approach for state machine replication, because one of the nodes acts as primary (i.e. master) and rest all as backups (i.e. followers). A primary is chosen in round robin fashion. Any change of state event is initiated by the master. As consensus requirement is $f + 1$ in the presence of maximum up to f faulty nodes, a request is committed if approved and executed

by at least $f + 1$ nodes. As PBFT protocol operates in an asynchronous environment, consensus happens in five phases as explained in the following:

1. Request
    1.1. Client submits the request to the master node. A request could be of any type; read or write.
2. Pre-prepare
    2.1. The master node records the request in a log and assigns a unique id to maintain request ordering.
    2.2. Master broadcasts the request along its identity to all the follower nodes in a message tagged as pre-prepare message.
    2.3. Follower nodes evaluate the pre-prepare message and accept only if its valid.
3. Prepare
    3.1. If a node decides to accept the pre-prepare message then it informs the remaining network by broadcasting a new message with the same request identity generated by master. We call this message as prepare message.
    3.2. Receiving node considers a prepare message as the acceptance of the given request by respective sender node.
    3.3. A node waits for at least $2f$ prepare messages from other nodes, that match its own pre-prepare message, to start the commit phase.
4. Commit
    4.1. Each node multicasts a commit message to the network during commit phase.
    4.2. A node waits for at least $2f$ valid commit messages from other nodes, before executing (i.e. committing) the request.
    4.3. Each node executes the request after observing at least $2f + 1$ valid commit messages including its own.
5. Reply
    5.1. After executing the request, each node sends response to the client.
    5.2. Client waits for $f + 1$ similar responses from different nodes to confirm the execution of the request.

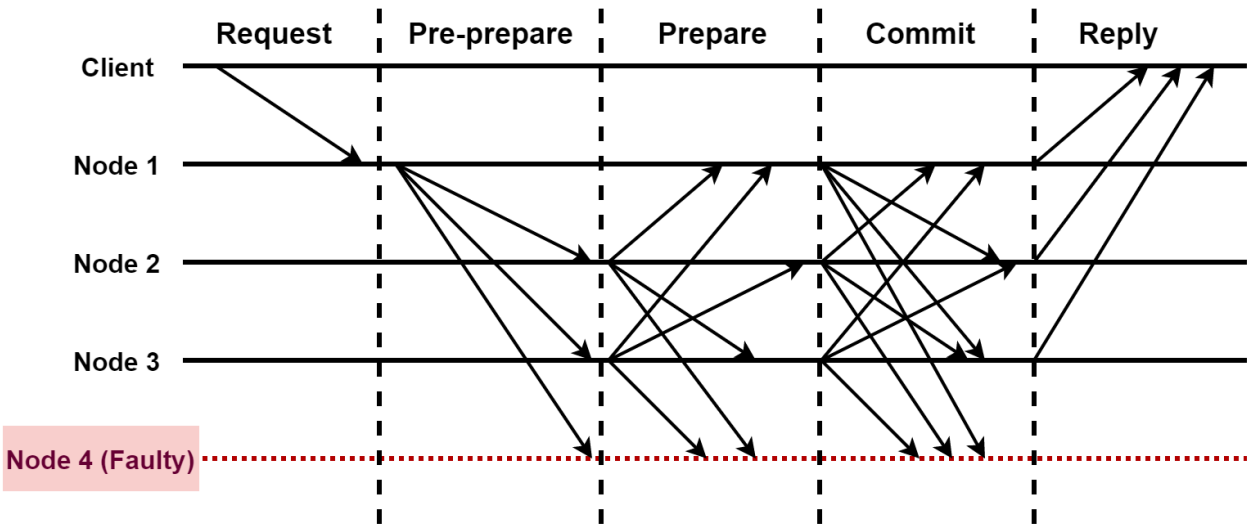


Figure 2.1 PBFT protocol message exchange for $n = 4$ and $f = 1$

Figure 2.1 shows the message flow diagram for above mentioned protocol. System is composed of four nodes; one of the nodes (i.e. Node 4) is faulty.

3 Literature Review

Scaling blockchain is one of the active topics among blockchain researchers' community. Multiple approaches are proposed to scale the blockchain architecture. Scalability has multiple dimensions e.g., storage, throughput, number of messages required for the consensus; however, in this thesis, we consider increasing the throughput of blockchain architecture. Most of the published research works are in reference to Bitcoin as the major application. In the following, we discuss major approaches to increase the throughput of the blockchain architecture.

3.1 Increasing Block Size

A very logical and straightforward step to increase the throughput of PoW based Bitcoin blockchain without any significant changes, is to accommodate more transactions in a block. More number of transactions implies bigger blocks; and a bigger block has greater block propagation latency (i.e. takes more time to populate across the network). Greater block propagation delay increases the probability of the fork in chain, because during the period of block propagation delay multiple nodes may find the PoW and broadcast different versions of the next block in chain. This is illustrated further in the following.

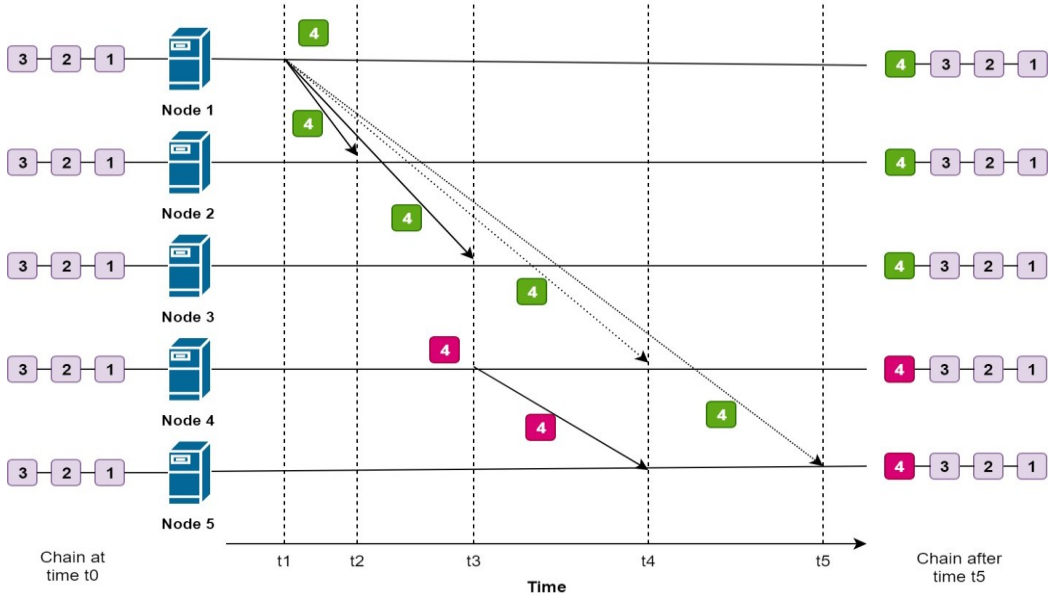


Figure 3.1 Fork of chain due to higher delay in block propagation

Figure 3.1 shows a blockchain network made up of five nodes. We can see the order of events take place as time proceeds as follows:

- Before t1: all the nodes are mining upon the same ledger state consisting of three blocks.
- t1: node 1 solves the PoW at the earliest at time t1, hence creates block 4 and starts broadcasting to the remaining network.
- t2 and t3: node 2 and node 3 accept and add block 4 sent by node 1 at time t2 and t3 respectively. However, due to network propagation delay, node 4 and node 5 haven't received block 4 created by node 1. Thus, they continue mining on the head of their local copy of ledger (i.e. block 3).
- t3: node 4 succeeds to solve the PoW and broadcasts a version of block 4 different (e.g. due to different nonce or transactions) than created by node 1.
- t4: node 4 receives the block 4 sent by node 1, however it rejects because its ledger has already mined and added a different version of block 4 by then. Simultaneously, node 5 also add block 4 received from node 4.
- t5: node 5 receives the block 4 sent by node 1, however it rejects because it has already committed a different version of block 4 received from node 4.

Thus, at time t5, we have two different versions of ledger (i.e. a fork). This fork exists in the network unless one of the chains becomes longer than the others and majority of the nodes learn about the longest chain. Hence, network will keep mining on both the branches, until one branch becomes longer than the others. When a node learns about presence of a longer chain, it accepts the longer chain as valid ledger and discards the alternative(s), which results in wastage of all the efforts invested to mine the blocks in discarded branches. Thus, to increase the throughput by accommodating more transactions in a block leads to frequent fork in the chain, that consecutively result in wastage of efforts due to fork(s).

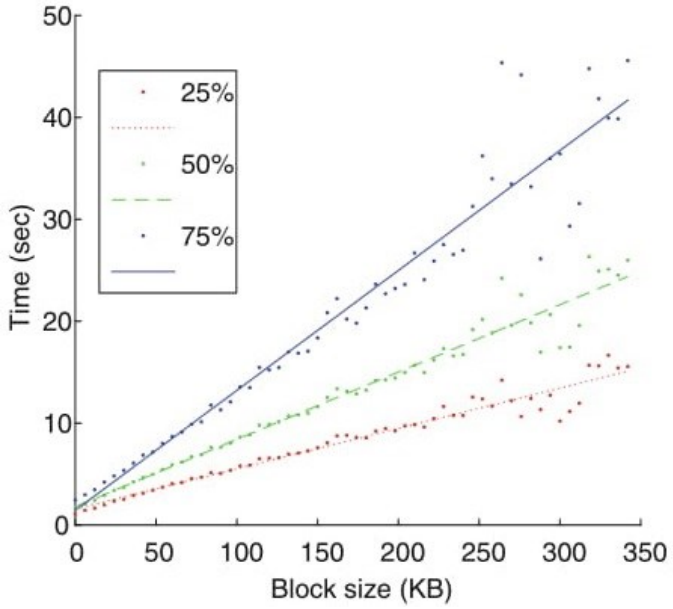


Figure 3.2 The relation between the block size and the time it took to reach 25% (red), 50% (green), and 75 % (blue) of monitored nodes

Figure 3.2 is an analysis done by Sompolinsky and Zohar from Microsoft research , to compare the block distribution delay associated with block size [28]. We can observe that block distribution time increases with the block size. This further creates centralization because nodes closer to the origin node of new block benefit with considerable higher head start to mine next block, thus higher probability of creating next block compared to the farther nodes.

Therefore, due to multiple reasons as explained before, increasing block size is not the solution to increase throughput.

3.2 Increasing the Rate of Block Creation

Second option to increase throughput is, by increasing the rate of block creation, viz., create more blocks of the same size within the same specified time period. The more time it takes to create a new block, the more difficult it is to find a valid PoW. Therefore, we need to reduce the complexity of PoW across the network to increase the rate of block creation. As the block size is intact, it takes the same duration to populate a new block across the network. However, due to reduced PoW complexity, it is more likely that multiple nodes simultaneously generate different

versions of new block; that again result in fork. Therefore, increasing the rate of block creation also increases the probability of fork and eventually wastage of mining efforts [28].

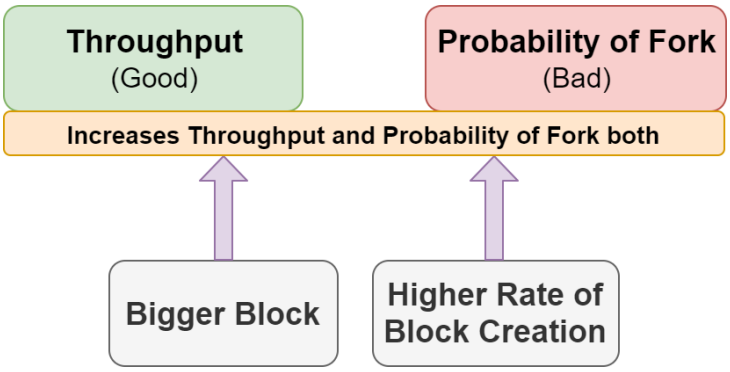


Figure 3.3 Impact of block size and rate of creation on throughput and probability of fork

Figure 3.3 illustrates the impact of increasing block size and/or the rate of block creation on throughput and probability of fork. Higher the block size and/or rate of block creation also mean higher throughput and probability of fork.

3.3 Bitcoin-NG

Reducing the complexity of PoW to increase the throughput is not a concrete approach as we discussed previously, hence Bitcoin-NG [29] proposes a different approach to increase throughput. It divides the block creation process in two phases as *leader election* and *transaction serialization.*

Every node tries to solve PoW challenge with equal probability similar to Nakamoto consensus protocol. Once a node solves the PoW, it becomes the leader and broadcasts a special block called *key block* to the network. A key block doesn't contain actual transaction. It only contains the hash reference to the previous key block, PoW and public key of leader. Unless the next key block is generated, the leader gets the privilege to validate and commit transactions in a predefined size of blocks, known as *microblocks.* A microblock is consisting of actual transactions and its Merkle root; and is valid only if signed by the private key of the active leader. The act of committing micro blocks is called as *transaction serialization* phase. The leader election and consecutive transaction serialization phases combined, is defined as an epoch.

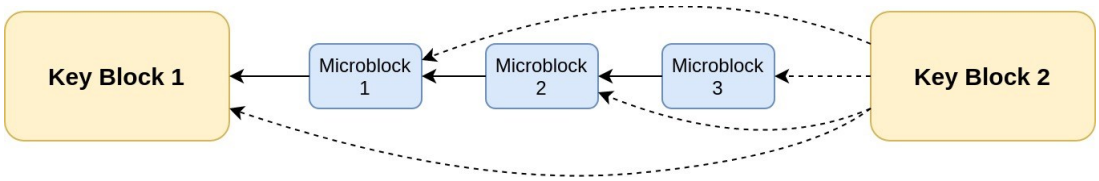


Figure 3.4 Structure of the Bitcoin-NG chain

As shown in Figure 3.4, miners may generate next key block referring to any of the key block or microblocks (usually the later one) of the latest epoch. Leader commits multiple micro blocks during an epoch, which eventually increases the throughput compared to Nakamoto consensus. However, it still uses PoW to choose the leader, which makes it energy inefficient. Secondly, it has the possibility of fork which leads to waste of CPU cycles. Finally, only leader is allowed to commit the transactions, which makes transaction validation process highly sequential.

3.4 Algorand

PoW based Bitcoin blockchain has latency on the order of hours to confirm a transaction due to the possibilities of fork in the ledger. Hence, Algorand [30] proposed a new blockchain architecture of the cryptocurrency that confirms transactions with latency on the order of a minute.

It uses a new Byzantine Agreement (BA) protocol [31] for the consensus among nodes to commit and confirm transactions in a new block within order of minute as its major contribution. Consensus in a large network has higher latency, therefore it limits the nodes participating in the consensus process by nominating a small set of random nodes to propose new block in the chain. It proposes a set of Verifiable Random Functions (VRF) [32] for the nodes to privately check whether they are selected to participate in the BA to agree on the next set of transactions to be committed in a new block.

Each node is assigned a pair of keys, secret key S_k and public key P_k. P_k is known to everyone else in the network. There exists a magic seed string Q_r, available to everyone in the system. A node computes $(S_k, Q_r) \rightarrow (Y, \rho)$ using VRF to check its eligibility to be a part of the committee to run Byzantine Agreement on a new block. Range $[0, P]$ is computed based on the stake the user holds in the system. If Y falls within the range $[0, P]$ then respective node is eligible to be a part of the committee that generates new block. For a given (Y, ρ) and the respective public key P_k,

anyone can verify that Y is a valid unique output and it falls within a desired range $[0, P]$. Hence, a node uses (Y, ρ) as the proof to validate its membership to commit next block.

Instead of PoW, it is motivated by an approach similar to PoS as explained previously for the reason of reduced confirmation time and energy efficiency. It requires 2/3 of the monetary owned by honest nodes to fulfill security criteria.

3.5 Byzantine Fault Tolerant (BFT)

BFT protocol has remained active research topic among consensus protocols for replicated state machine as it offers consensus in asynchronous network consisting of byzantine nodes up to predefined maximum. Research on BFT protocol convinces its ability to offer great throughput with minimal latency, however it often receives criticism regarding performance while scaling the network beyond 20 nodes. Contrarily, Nakamoto PoW offers good scalability of network but considerably less throughput compared to BFT protocol. In the following, we discuss the comparison between BFT and PoW over various attributes as presented by the research team of IBM blockchain framework, Fabric [33] [34].

3.5.1 Node Identity Management

PoW offers completely anonymous identity of miners, thus anybody can download existing chain along required mining software package and become a miner. Network is completely decentralized as it doesn't require to offer a central identity management module to issue a unique identity to each miner, therefore such blockchain is categorized as *public* or *permissionless blockchain* [17]. On the other hand, BFT based consensus protocol requires each node to know the identities of peers, which in turn enforces the need of central identity management authority to issue unique identity to each node participating in the network. Unlike PoW based blockchain network, a node has to be issued valid identity to participate in network, therefore categorized as *private* or *permissioned blockchain* [17].

In some of the literatures, presence of an identity management authority is considered as a disadvantage of decentralized architecture of blockchain. However, in major business domains

(e.g. banking, finance and real-estate) identity management is essential, due to legal and other compliances imposed on business model.

3.5.2 Commit Time

As we discussed in section 2.1, PoW follows randomized leader election to create a new block. Even if all the nodes are legitimate, there are chances of fork in the chain due to the network propagation delay and concurrency among all mining nodes trying to create the next block. The longest chain will be accepted by network in the event of fork. Therefore, a transaction committed in a block, will only be considered as committed when respective block becomes candidate of the longest chain. Current Bitcoin network takes some minutes to hours to commit a transaction [35]. In contrast, a valid transaction processed in BFT network is considered as committed instantaneously, without any delay between processed to committed states. This gives advantage to BFT over PoW as most of the business applications are aiming to offer nearly real time transaction processing.

3.5.3 Network Size

Bitcoin network is hosting thousands of miners which proves the ability of PoW to host a large network. On the other hand, BFT and state machine replication are tested for smaller scale (i.e. 1- to 20 nodes) blockchain networks only. Due to quadratic communication cost (i.e.$O(n^2)$), BFT requires modifications to deploy on a large scale blockchain network.

3.5.4 Performance

Bitcoin network is processing 7 transactions per second [11] [33] due to inherent challenges associated with PoW protocol. We may increase throughput by increasing block size or the rate of block creation, however that introduce other challenges as discussed in sections 3.1 and 3.2 respectively**.** Wherein BFT offers throughput up to processing tens of thousands of transactions per second in a smaller network.

3.5.5 Fault Tolerance

PoW based blockchain can tolerate up to 49% of the computing power occupied by faulty nodes. If 51% of the computing power comes from the malicious nodes then PoW network becomes victim of infamous 51% attack [9]. BFT based network follows a different fault model. An asynchronous BFT network consisting of n nodes can tolerate up to $\frac{n}{3}$ faulty nodes, as mentioned in section 2.4 while discussing PBFT.

To summarize this discussion, BFT is highly energy efficient compared to POW; it has quadratic communication cost, therefore it offers great performance in a relatively smaller network. PoW can accommodate thousands of participating nodes yet limited by relatively very low throughput and high transaction commit latency; also, the operational cost is very high due to the extensive energy consumption required to generate correct proof-of-work.

3.6 RSCoin

George Danezis et al proposed a blockchain framework named as *RSCoin* [36], to scale the blockchain architecture. The framework is motivated to develop a cryptocurrency system with traditional banking compliances imposed, thus presented as Centrally Banked Cryptocurrencies. It has relaxed decentralization property of blockchain by introducing a central authority to approve any block to add to the ledger.

RSCoin blockchain is a permissioned network consisting of two types of nodes; *monetary* nodes deployed by central authority and general nodes known as *mintette*, which are also approved to participate in network by governing body. Mintette nodes can accept transactions from users and validate it locally. Each user is mapped to certain mintettes by deploying a deterministic hash function, which takes user identity as input and returns a set of mintettes responsible to process respective user's transactions; mintettes may coordinate among themselves to avoid any double spend attack. Over a predefined period of time, every mintette node submits locally validated transactions to the monetary nodes. Monetary nodes aggregate the results from all mintettes and generate a final block consisting of all valid transactions proposed by mintettes. This approved block is then sent to the network to commit in the ledger.

As every mintette node maintains complete ledger and operates independently to process transactions locally, transaction validation and blockchain read operations scales well with the size of network. However, it has major drawback of centralization due to trusted monetary nodes are required to approve each block. Additionally, it assumes that a set of mintettes, responsible to process a user's transactions, is composed of majority of the non-malicious nodes.

3.7 Chainspace

Chainspace [37] improves RSCoin by introducing a more general distributed ledger specifically for smart contract processing in an open environment. The network is divided into multiple shards and users are mapped to a specific shard using a public hash function. Sharded Byzantine Atomic Commit (S-BAC) protocol is used for consensus within the shards (i.e. intra-shard consensus). S-BAC uses MOD-SMART [38] implementation of PBFT protocol as the base line for the state machine replication.

Intra-shard consensus takes place in parallel to improve the consensus performance. For the transaction involving multiple shards, it uses a simple two phase commit protocol [39] to achieve distributed commit and replicated consensus; such transaction is committed only if all the shards are willing to commit the transaction.

Shards publish evidence (i.e. signatures of the consenting nodes) of consensus on the transactions committed in blocks for the reason of transparency and auditability. Chainspace threat model detects malicious shards by following a major audit and replaying all transactions processed by the shard, which has linear time complexity (i.e. $O(n)$) for n transactions.

Though malicious shards can be detected by following a major audit, it is not explained how we can recover corrupted transactions. Secondly, details about sharding the network are kept abstract in the proposal; and there is no consideration of performance and security while sharding the network.

3.8 ELASTICO

ELASTICO [40] proposes a PoW based network sharding approach to scale transaction validation for public blockchain. For a network made up of n nodes each with equal computing abilities, it assumes up to $f < \frac{n}{4}$ byzantine adversaries. Based on network size n and maximum allowable byzantine adversaries, it follows probabilistic model to decide the network shard size. It divides the network into multiple shards in a permission-less environment, each processing a disjoint set of transactions. By sharding the network and transactions, it achieves parallelism for transaction validation.

Each network shard is called as a *committee*. Committee size C_n is fixed up to a minimum before starting the transaction validation process. Every node solves PoW puzzle at the first step of transaction validation process. Initial C_n nodes, who solve PoW puzzle at the earliest, form the first committee known as *directory committee*. Rest of the nodes send their PoW result along the identity to directory committee members. Once directory committee members hear enough responses required to form the remaining committees, every individual member of directory committee multicast the membership details to the remaining network nodes. Network nodes consider the majority as valid response from directory committee.

It requires some mapping algorithm to map a transaction to a specific committee, such that transaction shards are disjoint. Each transaction shard is assigned to only one committee (i.e. one-to-one mapping) for the validation. Each node individually validates all the transactions within its assigned transaction shard. Committee members follow PBFT algorithm for the intra-committee consensus on transaction validation result. After the consensus among committee members, each committee sends the validation result to directory committee members. Directory committee aggregates the validation results from all the committees and broadcast a final aggregated validation result to the network. Additionally, they also broadcast a unique set of random values; which must be used as the seed for the next PoW. This is required to avoid any node to reuse the PoW solution in the future epoch processing.

Thus, by sharding the network and creating a hierarchy in a public blockchain, ELASTICO achieves better throughput. It reduces the time to commit a transaction by following consensus

within smaller committees. Transaction shards are disjoint and mapped to only one committee; and each committee validates transactions in parallel, that improves the efficiency of the transaction validation phase.

Both ELASTICO and Bitcoin-NG (discussed in section 3.3) use PoW, which does not address the energy consumption aspect and other challenges associated with PoW as discussed in section 2.1. Though our approach follows a sharding structure similar to ELASTICO, there are some major differences. Firstly, ELASTICO chooses nodes on random basis while sharding the network and offers only probabilistic correctness with no mechanism to identify malicious shards. In comparison, our approach provides a mathematical model to decide on the optimal shard size based on both performance and correctness of transaction validation in the presence of malicious and faulty nodes. In addition, our approach provides a mechanism to identify malicious/faulty nodes by overlapping transactions across shards. Secondly, broadcasting transactions during commit phase is inefficient in the hierarchical structure of ELASTICO, hence it broadcasts and commits only the block header. Thirdly, there is no peer-to-peer communication across the shards to exchange information and synchronize the state. Lastly, our approach uses PoS instead of PoW, which removes the high energy consumption burden.

3.9 Omniledger

OmniLedger [41] proposed a sharding protocol to address some of the limitations of ELASTICO. It shards the network using a probabilistic model that is based on distributed randomness generation protocol RandHound [42], combined with VRF leader election algorithm [32]. The goal is to shard the network in such a way that each shard is composed of the majority of non-faulty nodes. The network of n nodes can consist maximum up to $\frac{n}{4}$ malicious nodes.

RandHound is a client-server randomness scavenging protocol enabling a client to gather fresh randomness on demand from a potentially large set of nearly-stateless randomness servers, preferably run by independent parties. It relies on a leader to orchestrate the protocol run, hence requires an appropriate mechanism to select one of the honest nodes as leader. Any deterministic approach to select the leader may easily enable an adversary to be the leader, hence the leader selection mechanism must not be either predictable or biased. To fulfil this criterion, Omniledger

uses VRF based leader election algorithm to choose an unpredictable and unbiased leader for the RandHound execution.

In addition to sharding the network nodes, it also shards the ledger, which requires each node to store only a portion of the ledger. As the ledger is also sharded, a two-phase commit protocol is used to handle cross shard transactions. The client (i.e. user) sends a cross shard transaction to all the shards required to process the transaction; shards evaluate the request and respond to the client with the status either accept or reject by following a variant of PBFT consensus protocol; if all the shards respond with the accept status then client asks them to commit; otherwise, it asks to abort the transaction even if at least one of the shards has responded with the reject status.

Omniledger presented a proof of concept to address scalability issue of blockchain. By sharding the network without using PoW, it solves the energy consumption aspect of traditional blockchain. Sharding the network and ledger makes the system scalable. However, it requires client's involvement to process cross shard transactions. Also, there is a probability of the shard being malicious (i.e. composed of majority of the malicious nodes), though very small, it doesn't propose any technique to detect a malicious shard. Secondly, if majority of the transactions are touching most of the shards then the system is better off being monolithic (i.e. the whole network as just one shard).

3.10 RapidChain

RapidChain [43] partitions the network into multiple smaller groups of nodes (i.e. network shards), each processing a disjoint set of transactions and maintaining their own ledger. Thus, by sharding the network and workload, it enables the parallelization of transaction processing and distribution of storage.

It shards the network by randomly assigning nodes to the committees to distribute malicious nodes evenly among all the committees. Network shards are also called as *committees*. There exists a *reference committee* responsible to drive the network sharding. Committee members follow a peer-discovery algorithm to discover the other nodes belonging to the same committee.

For a network made up of n nodes, it allows maximum up to $\frac{n}{3}$ nodes to be malicious or byzantine combined. It uses the hypergeometric distribution model [44] to calculate the failure probability of each epoch. Therefore, the committee size m is decided using formula $m = c\ logn$, where c is a constant (roughly 20) derived based on hypergeometric distribution model of probability.

User submits the transactions to a small set of random nodes in the network. On receiving a transaction from the users, a non-malicious and non-byzantine node applies a hash function to redirect it to the respective committee, which is also known as *inter-committee routing protocol*. Inter-committee routing protocol is also used to coordinate cross shard transactions. The receiving committee is responsible to coordinate cross shard transactions. Network follows a gossip protocol built based on the Information Dispersal Algorithm (IDA) [45] to efficient disperse large messages.

An individual committee periodically batches a set of transactions into new block (usually of 2MB size) by following byzantine consensus protocol for the intra-committee consensus protocol. Committee members choose a leader to follow byzantine consensus protocol for the intra-committee consensus. The leader proposes a new block consisting of valid transactions and consequently the whole committee commits it, if majority of the nodes agree during intra-committee consensus.

It doesn't rely on any public-key infrastructure, instead requires each node to solve a PoW puzzle to participate in the next epoch. However, nodes can generate PoW solution offline in parallel to epoch processing for better throughput. Thus, it doesn't solve energy inefficient characteristic of PoW, albeit the overall throughput is improved due to parallelly generating PoW.

While sharding the network into committees and distributing the work among committees, it considers that each committee is consisting of majority of the non-malicious nodes. Therefore, it doesn't deploy any mechanism to detect a malicious committee. Also, sharding scheme emphasizes only on the security aspect; it doesn't consider the overall system performance while sharding the network.

3.11 Monoxide

Monoxide [46] is a PoW based blockchain sharding system. It proposes to divide the network into multiple zones called as *asynchronous consensus zones*, each working as an individual PoW blockchain. For a given sharding scale k, it creates 2^k zones. In addition to sharding the network, it also distributes the ledger such that each zone is responsible to maintain a ledger that is specific to the zone. Each zone works as an individual PoW blockchain; in other words, mining competition, chain growth, and transaction confirmation happen asynchronously in parallel.

PoW based blockchain is vulnerable to 51% attack, therefore it is implicit that it relies on the majority of the computing power coming from honest nodes to outpace the malicious nodes. It becomes more feasible for the malicious nodes to corrupt an individual zone by dominating at least 51% of the computing power in Monoxide architecture, because the overall computing power is divided among zones. To address this issue, it does effective mining power amplification and call it as Chu-ko-nu mining; wherein a node once solves the PoW puzzle, can create multiple blocks in different zones. Such technique amplifies the power of honest nodes and forces malicious nodes to distribute their mining power evenly among all the zones.

It is crucial to handle cross zone transactions involving multiple zones because each zone is an independent ledger. The protocol must offer atomicity for any cross zone transactions. A two-phase commit protocol is the usual choice to achieve atomicity of cross shard transactions. However, instead of using any two-phase commit protocol, Monoxide offers eventual atomicity to ensure that any cross zone transaction will be committed by all the relevant zones eventually. It is called eventual atomicity because each zone is incentivized to process cross zone transactions and relay to the next one; therefore, incentive model motivates the nodes within a zone to eventually commit a cross zone transaction.

By sharding the network into multiple independent zones, Monoxide demonstrates higher throughput in the blockchain network, however it does not explain a sharding scheme that offers optimal performance. Also, miners still solve PoW puzzle to commit a new block, which makes it energy inefficient. ■

In this chapter, we discussed multiple approaches to scale the blockchain network, mostly by sharding the network. Following table summarize our discussion by comparing key characteristics of discussed proposals.

	Network Type	Uses PoW	Shards the network	Sharding model	Shards the ledger	Considers performance while sharding the network	Proposes a technique to detect malicious shard
Bitcoin-NG	Public	✔	✗	NA	NA	NA	NA
Algorand	Public	✗	✗	NA	NA	NA	NA
RSCoin	Private	✗	✔	Probabilistic	✗	✗	✗
Chainspace	Public	✗	✔	Kept abstract in proposal	✗	✗	✔
ELASTICO	Public	✔	✔	Probabilistic	✗	✗	✗
Omniledger	Public	✗	✔	Probabilistic	✔	✗	✗
RapidChain	Public	✔	✔	Probabilistic	✔	✗	✗
Monoxide	Public	✔	✔	Fixed & Deterministic	✔	✗	✗
OptiShard	Hybrid	✗	✔	Fixed & Non-Deterministic	✗	✔	✔

Table 3.1 Comparison of different blockchain scaling models

4 Proposed Architecture

4.1 Introduction

In chapter 3, we discussed various approaches to scale blockchain architecture to compete with other mainstream transaction processing platforms, e.g. VISA, Mastercard. Albeit scalability has multiple dimensions (e.g. storage, throughput, communication bandwidth), our focus is mainly in increasing the throughput of the blockchain architecture as the network size grows.

We discussed the possibilities of fork in the chain by increasing the block size or the rate of block creation in PoW blockchain. Given the fact that fork in the chain negatively impacts the transaction commit latency and throughput, Bitcoin-NG proposed an architecture that allows a node to commit multiple micro blocks of predefined size followed by producing a valid solution of the PoW puzzle. Bitcoin-NG offers comparatively better yet limited improvement in the throughput, because each node competes to create new block without distributing the workload.

It is implicit that a large-scale system doesn't scale the throughput, if the nodes do not divide the workload. Therefore, sharding the network and distributing the workload (i.e. pending transactions) among shards in such a way that each shard validates a disjoint set of transactions, promises scalable blockchain. While sharded system offers better scalability, it also requires to deploy special measures to hold security and fault tolerance in the presence of malicious and faulty nodes in the network. Many approaches are proposed to shard the blockchain with different security and fault tolerance models, as we discussed in chapter 3. However, none of the literature derives sharding scheme considering performance, security and fault tolerance altogether.

In this chapter we propose a hierarchical blockchain architecture, named OptiShard, which is a hybrid of decentralization and peer-to-peer. The goal is three-fold: performance, security (in the presence of malicious nodes), and fault tolerance. The hierarchy arises due to sharding of the network nodes into disjoint partitions (we call them committees) and distribution of workloads among these committees. Transactions are distributed among committees in such a way that most of the transactions are non-overlapped (i.e. one-to-one mapping), however certain predefined

number of transactions are overlapped across all the committees (i.e. one-to-all mapping), in the following discussion we call them as *overlapped transactions*. Validation result of overlapped transactions across all the committees is compared to identify corrupted committee(s), which is explained further in section 4.3.4.

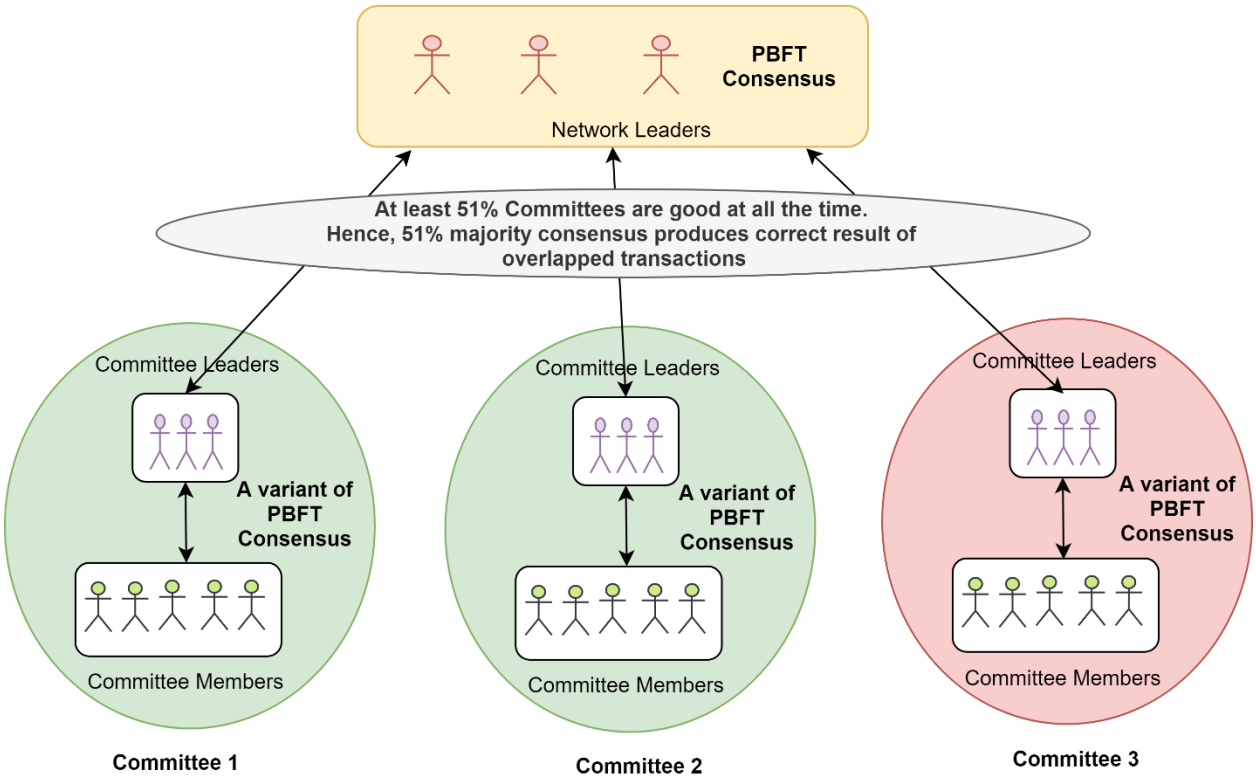


Figure 4.1 Network hierarchy and consensus protocols

Each committee is headed by committee leaders, while committee leaders are headed by network leaders as shown in Figure 4.1. For the purpose of majority consensus and fault tolerance, there is an odd number, greater than 1, of network and committee leaders.

In the hierarchy, there are two levels of consensus: intra-committee consensus among all committee members of a committee and inter-committee consensus among all network leaders. The Practical Byzantine Fault Tolerant protocol (PBFT) and its variant are used at both the levels of consensus. Network leaders follow PBFT protocol, whereas committees follow an optimized variant of PBFT protocol that has communication complexity in linear order compared to polynomial in standard PBFT protocol. This is further explained while discussing intra-committee consensus protocol (section 4.3.3). Committee leaders do not execute any consensus among

themselves and this consensus is handled by the network leaders. Committee members can also communicate with their selected peers in other committees using peer-to-peer links; this is required during the transaction commit phase as explained in commit protocol (section 4.3.5).

4.2 Terminologies and Assumptions

We assume a semi-private blockchain network consisting of N physical nodes. Our network is an asynchronous environment where each node is represented as a state machine responsible to maintain the complete ledger. A change of state event, say X→Y, is called an epoch. In the following discussion, we assume a financial domain and hence financial transactions (e.g. in a banking system) for illustration purposes; however, the proposed approach is also applicable to other blockchain applications involving transactions.

4.2.1 Network Nodes

4.2.1.1 Network Leaders:

Network Leaders are deployed by a trusted party in the semi-private domain (e.g. a governing authority in a financial institution) and hence they are non-malicious; however, the network leaders can be faulty (e.g. byzantine failure). Identity of network leaders are known to the entire network and any node can establish a point-to-point communication with a network leader. Majority of the network leaders keep their ledger up-to-date.

4.2.1.2 Committee Leaders & Members:

Committee Leaders and Members are any nodes from the untrusted (e.g. public) parties and they can be malicious or faulty. They can leave and join the network at any time. A node can receive updated copy of the ledger from the network leaders before participating in the epoch processing.

Every node participating in the epoch processing reserves some currency as a bet to the network leaders, which is called as reserved stake for the current epoch. A higher stake usually implies a higher probability that a node is non-malicious. PoS protocol is used to select leader(s) of a committee, which also ensures a higher probability that a committee leader is non-malicious.

4.2.1.3 Peer Node:

Every committee member and leader are assigned at least one node across the committees to exchange the transactions required to commit the epoch (explained in section 4.3.5). We call such nodes as *peers*.

4.2.2 Fault Model

4.2.2.1 Malicious Node:

In addition to producing incorrect results, a malicious node may also involve in other undesired activities, for example: not participating in the protocols, or altering the contents or signatures in messages received from other nodes.

4.2.2.2 Byzantine Node:

A node can become byzantine at any time and produce corrupted result. However, maximum allowable malicious and byzantine nodes together are fixed.

4.2.2.3 Faulty Node:

A node can be either malicious or faulty. For our discussion, the end result of transaction validation by a malicious node and a faulty node is similar; hence we will treat and call them equally as faulty nodes, unless otherwise stated explicitly.

4.2.2.4 Good Node:

A non-faulty node is a good node. Any good node can become byzantine faulty during an epoch processing by producing a succession of incorrect results till the end of the epoch; however, the total number of faulty nodes remains within the upper bound set during the start of the epoch.

4.2.2.5 Faulty Committee:

The Practical Byzantine Fault Tolerant (PBFT) protocol and its variant are used for intra-committee (among committee members) and inter-committee (among network leaders) consensus as shown in Figure 4.1. As per PBFT, the maximum allowable faulty nodes f_c in a committee of C_n

nodes are such that, $f_c \leq \lfloor \frac{C_n - 1}{3} \rfloor$. By assumption of the protocol, at the most another $\lfloor \frac{C_n - 1}{3} \rfloor$ good nodes are controlled by adversaries and may be unreachable in an asynchronous network up to a defined duration. So, the protocol requires a $\lfloor \frac{C_n - 1}{3} \rfloor + 1$ majority for reaching consensus within a committee. If f_c is not within the allowable maximum, then we define such committee as a *faulty committee*.

4.2.2.6 Good Committee:

A committee is *good* if f_c (as defined above) is within the allowable maximum. A good committee reaches the consensus within a definite time. Network leaders also need consensus among themselves, e.g. during inter-committee consensus, and so it is assumed that less than $\frac{1}{3}$ of them can be faulty, as required by PBFT.

Nodes can communicate only with their assigned peers across other committees and not among nodes within the same committee. ■

In determining an optimal shard size, another required input parameter is the predefined majority of good shards. The sharding scheme guarantees that the number of good shards is at least the predefined majority. This majority is required for reaching a valid consensus during inter-committee consensus (performed by each network leader). In our following discussion, a 51% majority is considered.

Assuming a network of N worker nodes and only one committee in the extreme scenario, PBFT requires that at most $\lfloor \frac{N - 1}{3} \rfloor$ nodes can be faulty so that a valid consensus can be reached. Hence the requirement for validly committing an epoch is that there is an upper bound $f = \lfloor \frac{N - 1}{3} \rfloor$ on the total number of faulty nodes in the network.

Each epoch is timestamped with an increasing unique identifier generated and agreed by the network leaders. The same identifier is used by all the nodes to uniquely identify communication messages specific to an epoch. Hence every message contains an epoch id; and a good recipient node discards the message if the epoch id is smaller than the latest epoch the node has committed.

4.2.3 Transaction Validation

An example of a transaction T in the context of a banking system is of the form T(x):A→B representing transfer of x amount from account A to B. Every node applies a deterministic function β to a transaction T; β(T) returns 0 or 1 depending on if the transaction is invalid or valid respectively. β(T) on a malicious node or on a non-crash byzantine node produces inconsistent results, i.e., for a valid transaction it can return 0 (invalid) or 1 (valid), and vice versa for an invalid transaction. It is assumed that from the instant a malicious or faulty node produces incorrect result during an epoch, it will produce incorrect results for all the remaining transactions till the end of the epoch. β(T) on a node that has stopped due to crash (fail-stop failure) produces no output, hence the end result is the same as an unreachable node. Transactions remain anonymous to worker nodes; hence they do not have any prior information about the transactions. Every node maintains complete ledger; and all the transactions from a user are assigned to a single shard (explained further in section 4.3.2); hence, our system does not have any cross shard transaction.

In many of the blockchain sharding networks [36][37][40][41][43][46], each shard stores a partial copy of the ledger or works as an independent blockchain. Such technique scales system in storage dimension, because the storage requirement from each node is inversely proportional to the number of shards within the network. However, validation of a cross shard transaction requires a protocol for the communication and coordination among all the involved shards; secondly, if most of the transactions are cross shard transactions then system better off being monolithic; lastly, ledger replicas in the network are also limited up to only the number of nodes within a shard. Due to these reasons, we preferred each node in OptiShard to maintain the complete ledger.

4.2.4 Message Authentication and Validation

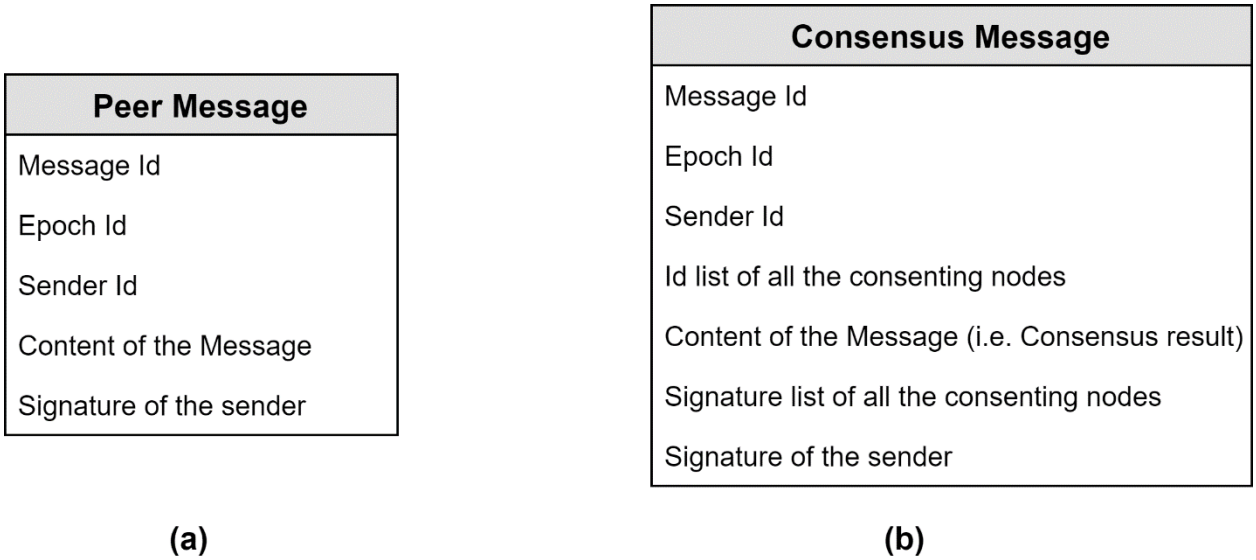


Figure 4.2 Data structure of Peer and Consensus messages in (a) and (b) respectively

In the presence of a key issuing authority, each physical node is assigned a unique key pair (secure key S_k, public key P_k) and a unique node id as an identity required for message authentication. Key issuing authority maintains the dictionary of node id to key pair mapping, therefore any node can lookup the public key corresponding to a node id from key issuing authority.

4.2.4.1 Sign a Message

Each message contains a unique message id generated by the sender node to discard any duplicate messages in the asynchronous network. Epoch id is required to uniquely identify the message corresponding to a specific epoch. It allows a node to disregard any delayed messages. Also, a malicious node can't reuse the messages from previous epoch to malfunction the current epoch processing. The originator of a message signs the message with its signature, generated by its secure key S_k, message content, and the epoch id.

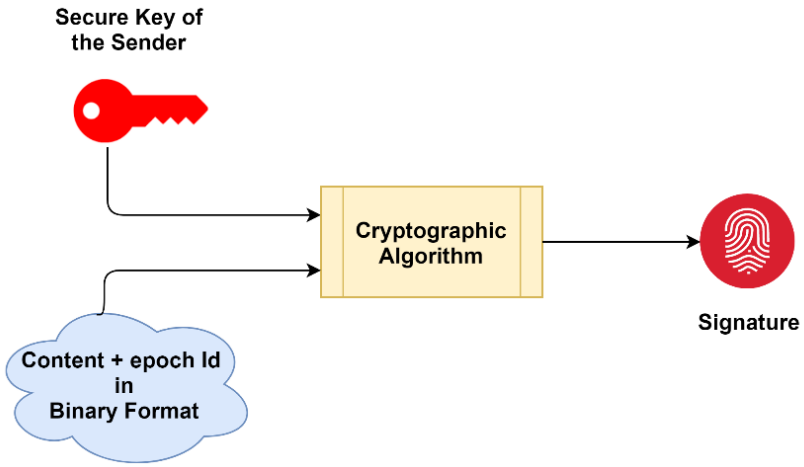


Figure 4.3 Generate signature using secure key of the sender

Cryptographic algorithm signs the binary content using secure key of the sender as shown in Figure 4.3. We assume the existence of a deterministic cryptographic algorithm that produces a unique signature for the combination of binary content and secure key; and signature differs even if there is a difference of single bit in the combination of binary content and secure key. Sender id is required to authenticate the message on the receiving end, which is discussed in the following section.

4.2.4.2 Authenticate a Message

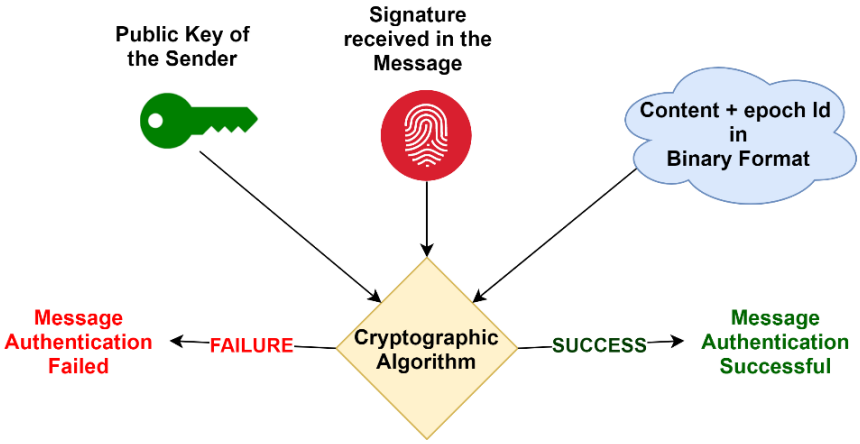


Figure 4.4 Message authentication using public key of the sender

A receiver node applies function ϕ to authenticate a message M. Cryptographic function ϕ takes public key, signature and binary content (i.e. message content and epoch id) to authenticate the message as shown in Figure 4.4. Recall that a node can lookup the key issuing authority to get the

public key associated with the sender id mentioned in the message. ϕ authenticates a message M only if the message content and epoch id are securely signed using secure key of the sender. A peer message is considered valid, if it can be authenticated by ϕ.

In the case of a consensus message, the message also contains the identities of all the consenting nodes along with their signatures as shown in Figure 4.2 (b). In addition to authenticating the signature of the sender, ϕ also authenticates the signature of each consenting nodes; also, checks for the total number of authentic signatures of the consenting nodes should be at least as many as required to prove the consensus majority of the respective message. We call this extra step as the validation of consensus message. Therefore, a consensus message has to be authenticated as well as validated at receiving end.

Though it is implicit for any cryptographic algorithm, we mention that no adversary can alter the content of a message and regenerate the signature unless it has access to the secure key.

A good recipient node discards an invalid message M and reports to the network leaders that the sender could be faulty. We cannot surely say that the sender is faulty because there are multiple possibilities of a message being invalid (e.g. the message is corrupted over the network).

4.3 Protocols

We divide our epoch processing protocol into five sections as following:

- *Network Sharding*: Shard the network into multiple committees
- *Workload Sharding*: Distribute the workload among committees
- *Intra-Committee Consensus*: Consensus on the transaction validation result within the committee
- *Inter-Committee Consensus*: Consensus among committees at the network level to finalize the valid transactions to be committed by the network.
- *Commit*: Commit the transactions as the last phase of epoch processing. We follow lazy commit protocol that offers eventual consistency [47].

In the following, we discuss these protocols in detail.

4.3.1 Network Sharding Protocol

The blockchain network of *N* nodes, that excludes the network leaders, is partitioned into C disjoint shards called *committees*, each of size $C_n = \lfloor \frac{N}{C} \rfloor$. Based on our previous discussion, PBFT requires $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ majority for reaching a consensus within a committee, and hence it requires that the number of faulty nodes within the committee $f_c \leq \lfloor \frac{C_n - 1}{3} \rfloor$ for the validity of the consensus. A committee produces valid intra-committee consensus if, $f_c \leq \lfloor \frac{C_n - 1}{3} \rfloor$.

Each committee sends intra-committee consensus result (including overlapped transactions) to network leaders. Network leaders compare the validation result of overlapped transactions among all the committees during inter-committee consensus. They consider a 51% majority as the valid comparison. Therefore, we require at least 51% of the good committees in order to reach a valid consensus. This establishes *Lemma 1* for determining the shard size.

Lemma 1 (Majority of committees are non-malicious): Let us consider a blockchain network of N nodes in the presence of f faulty nodes such that $f \leq \lfloor \frac{N-1}{3} \rfloor$. The network is sharded into C committees, each of size $C_n = \lfloor \frac{N}{C} \rfloor$, where C is an odd number to facilitate determining 51% majority consensus. In such a case, the number of good committees remains in majority as long as $f < \left(\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1 \right) \times (\frac{C+1}{2})$.

Proof:

As per PBFT, the majority required for consensus within a committee is $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$. Therefore, a committee becomes faulty when it contains at least $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ faulty nodes. In the worst-case scenario, the f faulty nodes are distributed in such a way that every faulty committee contains exactly $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ faulty nodes. This is the worst-case scenario because such a distribution of faulty nodes forms the maximum number of faulty committees. Considering that C is an odd number, the number of faulty committees must be at the most $\frac{C-1}{2}$ so that faulty committees remain in minority in a 51% consensus. Thus, in the worst case, the total number of faulty nodes required

to generate $\frac{C-1}{2}$ faulty committees, is $(\left\lfloor\frac{C_n-1}{3}\right\rfloor + 1) \times (\frac{C-1}{2})$. The remaining faulty nodes will be distributed in the good committees, however their number must be less than $\left\lfloor\frac{C_n-1}{3}\right\rfloor + 1$; otherwise, if these faulty nodes populate just one of the remaining committees then the number of faulty committees will be in majority. This leads to the following relationship:

$$f - \left(\left\lfloor\frac{C_n-1}{3}\right\rfloor + 1\right) \times (\frac{C-1}{2}) < (\left\lfloor\frac{C_n-1}{3}\right\rfloor + 1).$$

When simplified, it gives:

$$f < \left(\left\lfloor\frac{C_n-1}{3}\right\rfloor + 1\right) \times (\frac{C+1}{2}) \quad (1)$$

This concludes the proof.■

Referring to (1), with an increase in C, initially the term $\left\lfloor\frac{C_n-1}{3}\right\rfloor + 1$ decreases at a more rapid rate than the rate of increase of the term $\frac{C+1}{2}$. But as we keep increasing C, the right-hand side term of the inequality in (1) fluctuates as shown in following Figure 4.5. The goal is to choose a C such that number of good committees remains a majority.

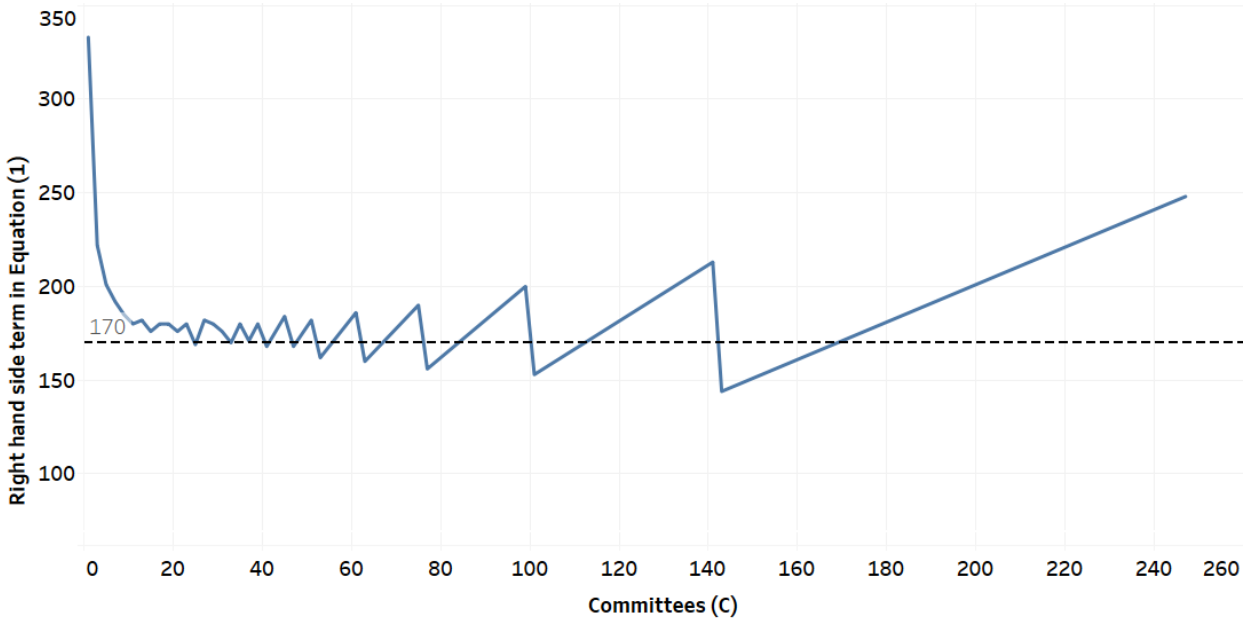


Figure 4.5 Right hand term of equation (1) for different values of C, given $N = 999$ and $f = 170$

As we discussed in section 4.1, hierarchy requires consensus at multiple levels (i.e. intra-committee and inter-committee). Intra-committee consensus time is in proportion to the committee size, whereas inter-committee consensus time is in proportion to number of committees. Nevertheless, transaction validation time is in inverse proportion to number of committees. Therefore, another important factor while sharding the network is to choose the right value of C, that offers optimal performance. Theoretical performance analysis is discussed in section 4.5 in the following discussion, because it requires us to first discuss the complete protocol. By combining results of Lemma 1 with the right value of C for performance measure, an optimal value of C can be determined.

As the network begins state transition, network leaders determine a suitable value of C based on the previous discussion. The network nodes are sorted in non-increasing order of their stake values; the topmost m nodes are nominated to become committee leaders, where m is a multiple of C such that each committee is headed by an odd number, greater than one, of committee leaders. The remaining nodes are divided into C disjoint shards, called committees. Nodes are assigned to committees in a round robin fashion. Committee members and leaders of each committee are sorted based on their stake values, and nodes across the committees at the same rank in the sorted order are defined as peers of each other. Thus, each member node is associated with at least $C - 1$ peers, at least one peer from each other committee. This is to enable peer-to-peer inter-committee communication, which is required to commit the epoch.

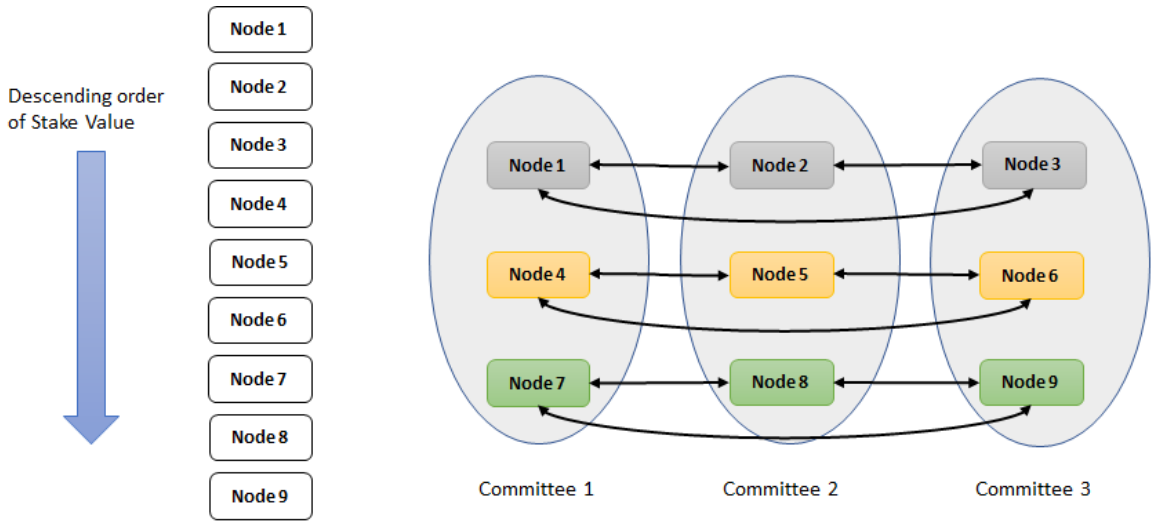


Figure 4.6 Node distribution and peer assignment among committees

Figure 4.6 shows node distribution and peer assignment while sharding a network of nine nodes into three committees. In addition to sharding the network and peer assignment, network leaders also generate new epoch id. Each network leader increments the previous epoch id by one to generate new epoch id. Subsequently, they follow PBFT among themselves for reaching consensus on the committee details discussed in the previous paragraph and the epoch id. Once consensus is reached, each network leader sends the following information to each committee leader:

- Committee member details of its assigned committee
- Epoch id
- Peer-to-peer links of each committee member

Thereafter, the epoch id and peer details are forwarded to member nodes by respective committee leaders. The previous discussion is summarized in the following steps:

Network Sharding Protocol

1. Each node registers its stake value to network leaders.
2. Each network leader does the following:
   2.1. Calculate C by following Lemma 1
   2.2. Sort all nodes in non-increasing order of the stack values
   2.3. Nominate a multiple of C top ranked nodes as committee leaders
   2.4. Shard remaining nodes into C committees
   2.5. Assign an odd number of committee leaders to a committee
   2.6. For each node, set at least $C - 1$ peers across the committees
   2.7. Generate epoch identity
   2.8. Follow PBFT with other network leaders to reach consensus on the results of steps 2.1 to 2.7
   2.9. Multicast to each committee leader about: committee member details, epoch id, and peer information for each committee member
3. Each committee leader multicasts epoch id and peer details to its committee members

Protocol 1. Network sharding protocol

4.3.2 Workload Sharding

Every transaction gets a timestamp when it is generated. Users can submit their transactions to any node in the network. Every node forwards each transaction to network leaders, who are responsible to decide on the transactions to be processed in each epoch. A transaction, when submitted to a network leader, remains in pending (i.e. yet to be processed) transaction pool of the corresponding network leader node until processed.

Network leaders generate a unique transaction id for each transaction by appending its timestamp value to the user id and then linearize all pending transactions based on non-decreasing order of transaction ids. *Epoch workload* (W) is decided by considering all pending transactions up to a predefined limit. Epoch workload is divided into C transaction shards in such a way that a predefined small fraction of transactions is common across all the shards. All the remaining transactions are distributed among the C transaction shards in a round robin fashion so that one transaction is mapped to exactly one shard. Common transactions across all the shards are overlapped transactions. Overlapped transactions are used to identify any faulty shards (discussed in section 4.3.4). All the transactions of the same user are batched together in one shard to avoid any double spend attack [2].

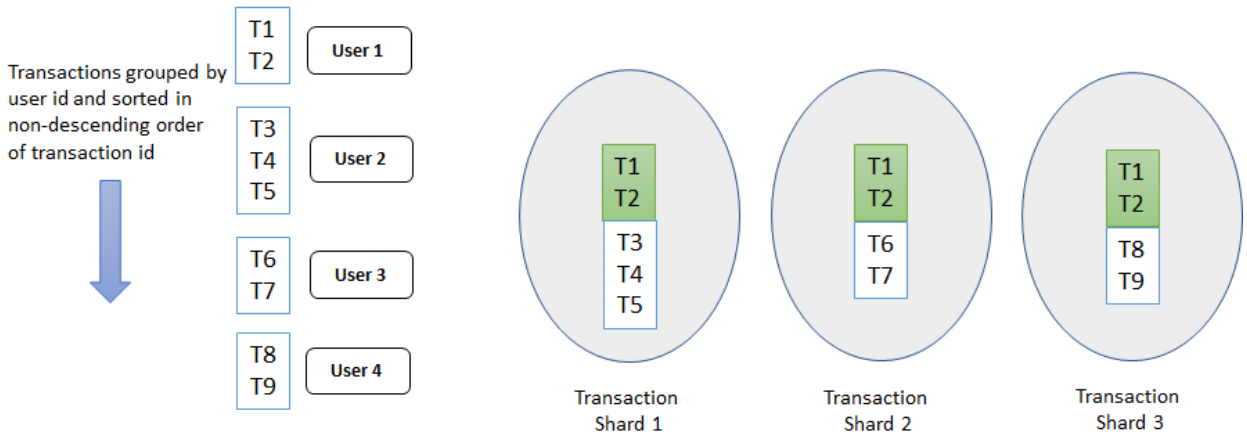


Figure 4.7. Workload distribution for $C = 3$, transactions T1 and T2 are overlapped transactions

Figure 4.7 shows a workload of nine transactions; generated by four different users; distributed among three transaction shards. Transactions T1 and T2 are overlapped transactions, therefore

common across all the transaction shards. Remaining transactions are distributed in round robin fashion such that, all the transactions of the same user are assigned to a single shard.

Network leaders assign each transaction shard to exactly one committee and multicast the shard details to the respective committee leaders. Each committee leader subsequently broadcasts its transaction shard details to the rest of the committee members.

By requirement, committee leaders and committee members have no way of identifying overlapped versus non-overlapped transactions; this information is solely maintained by the network leaders to keep overlapped transactions anonymous to all the other untrusted nodes. We keep overlapped transactions anonymous for the security purpose, which is discussed further in section 4.3.4. The previous discussion is summarized in the following:

Workload Sharding Protocol

4. Each node sends its pending transactions to network leaders
5. Network leaders collectively do the following:
   5.1. Assign a transaction id to each transaction
   5.2. Linearize all pending transactions
   5.3. Decide on a predefined number of overlapped transactions common across all transaction shards
   5.4. Distribute the remaining transactions among C shards in a round robin fashion so that each transaction goes to exactly one shard and all the transactions of same user are assigned to a single shard.
   5.5. Assign each transaction shard to exactly one committee
   5.6. Multicast transaction shard details to respective committee leaders
6. Each committee leader broadcasts the message to committee members.

Protocol 2. Workload sharding protocol

4.3.3 Intra-Committee Consensus

In the original PBFT protocol, each node sends prepare message to all the remaining nodes as we discussed in section 2.4. Therefore, it requires quadratic order of communication messages (i.e. $O(n^2)$) to produce consensus result in a network consisting of n nodes. Having said that, peer-to-peer message complexity of standard PBFT protocol is $O({C_n}^2)$ for a committee of size C_n, which is quite expensive for the purposes of performance and scalability. Hence, a centralized variant of PBFT protocol is used for intra-committee consensus.

In a modified variant of PBFT protocol, nodes within the committee do not communicate among themselves. A node produces its validation result and sends it to all the committee leaders. Committee leaders consider $\left\lfloor \frac{C_n-1}{3} \right\rfloor + 1$ majority for the consensus. If the committee is good then number of faulty nodes $f_c \leq \left\lfloor \frac{C_n-1}{3} \right\rfloor$. Therefore, $\left\lfloor \frac{C_n-1}{3} \right\rfloor + 1$ majority guarantees the consensus result produced by at least one good node.

Each committee member and leader apply the deterministic function β (discussed in section 4.2.3) on each of its assigned transactions; sorts its list of valid transactions in non-decreasing order of transaction ids; and calculates the Merkle root based on the sorted list. Committee members send only the Merkle root to the committee leaders. Note that valid transactions are sorted in non-decreasing order of transaction ids to generate a consistent value of the Merkle root across all the good committee members.

A committee leader accepts the Merkle root sent by committee member only if it matches with the Merkle root computed by itself. When a committee leader accepts Markel root value in a majority from at least $\left\lfloor \frac{C_n-1}{3} \right\rfloor + 1$ committee members, it marks consensus fulfilled; sends transaction ids of valid transactions and the agreed Merkle root to network leaders. Otherwise, committee leader considers consensus unfulfilled and reports to network leaders on a predefined timeout. It is important to note that all the consenting nodes must have generated the same validation result, because Merkle root value differs even if there is a mismatch of a single transaction in validation result. Considering exact matching result provides highest level of safety of consensus result. Recall that a *good* committee can contain up to a maximum of $\left\lfloor \frac{C_n-1}{3} \right\rfloor$ faulty

nodes, because of that $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ majority in a good committee assures the consensus result is produced by at least one good committee member. Also, a node is chosen as committee leader based on higher stake value, hence it is highly likely that a committee leader is not malicious due to the trust model followed by PoS. Therefore, committee leaders also validate the transactions and compare its Merkle root with the committee members.

Each node sends the Merkle root generated on the sorted list of valid transactions to only committee leaders, hence modified PBFT consensus protocol has the message complexity linear in committee size C_n, i.e., consensus complexity is $O(C_n)$ within a committee. Also, all committees can work in parallel to achieve intra-committee consensus, therefore the message complexity of the intra-committee consensus protocol is $O(C_n)$ in committee size C_n, for the whole network. This is one of the major advantages of the hierarchical architecture and consensus. The previous discussion is summarized in the following:

Intra-Committee Consensus Protocol

7. Each committee member and each committee leader do the following:
   7.1. Sort all transactions in non-decreasing order of transaction ids to linearize the transactions
   7.2. Apply β to validate each transaction (discussed in section 4.2.3)
   7.3. Calculate Merkle root on the sorted list of valid transactions.
8. Each committee member sends its Merkle root to each committee leader
9. Each committee leader does the following:
   9.1. If at least $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ identical Merkle roots are received from committee members that match with its own Merkle root then
      9.1.1. Mark consensus successful and send the following information to each network leader in a consensus message: transaction ids of valid transactions, the agreed Merkle root.
   9.2. If not or predefined timeout, then
      9.2.1. Consider intra-committee consensus unsuccessful and report to the network leaders in a message containing the received and its own Merkle roots.

Protocol 3. Intra-committee consensus protocol

4.3.4 Inter-Committee Consensus

Each network leader receives the consensus message from all committee leaders. The message includes: transaction ids of the valid transactions in the committee and the agreed Merkle root. A network leader accepts a consensus message M only if the message is validated by ϕ (discussed in section 4.2.4.2), that is the Merkle root is successfully authenticated using public key of at least $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ consenting nodes within the respective committee. Once the message is authenticated, the network leader also validates the consensus result by recalculating the Merkle root against the transaction ids. A faulty committee leader may send corrupted or invalid Merkle root (e.g. consensus majority is unfulfilled or Merkle root doesn't match with the list of valid transactions), therefore above validation steps are required to rule out any faulty committee leaders. A committee leader is marked as faulty if its message cannot be successfully authenticated or validated. Finally, a network leader accepts the consensus result from a committee only if there is at least one successfully authenticated and validated consensus message from a committee leader of the committee. If all committee leaders of a committee are marked faulty, then the entire committee is also marked as faulty.

Subsequently, the network leader checks the validation results of all the overlapped transactions. Recall that overlapped transactions are common to all the committees. The overlapped transactions are used to identify any faulty committees as follows: if the network is sharded based on Lemma 1, then it is guaranteed that more than half of the committees are good. Hence, a 51% majority consensus among committees is required on the result of each overlapped transaction. If a committee does not agree with the majority consensus on an overlapped transaction then the committee is marked as faulty. If a 51% majority could not be achieved on overlapped transactions due to such faulty committees then a network leader decides to discard the whole epoch.

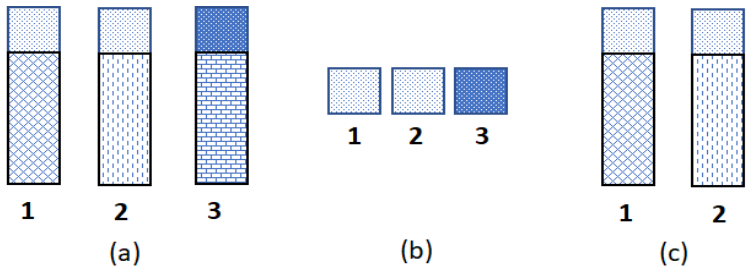


Figure 4.8. (a) Intra-committee consensus result for $C = 3$. (b) Comparison of overlapped transactions during inter-committee consensus. (c) Discard committee no 3

Figure 4.8 (a) shows intra-committee result for three committees. As we can see that validation result of overlapped transactions produced by third committee is different (i.e. validation result of at least one transaction differs) than the other two committees, which is identified by network leaders in Figure 4.8 (b). Network leaders discard the whole validation result (i.e. overlapped and non-overlapped transactions) of third committee, considering the chances of corrupted validation result produced for non-overlapped transactions, which is shown in Figure 4.8 (c).

Each network leader prepares the list of the Merkle roots from all the good committees. Subsequently, the network leaders execute PBFT protocol among themselves for reaching consensus on the results of the Merkle roots and the good committees or a decision to discard the epoch. If a consensus is not reached or consensus reached on to discard the epoch, then the results from the entire epoch are discarded; otherwise the agreed list of Merkle roots of all the good committees is broadcasted to all good committee leaders. This information is subsequently forwarded to committee members by the respective committee leaders.

All transaction results from a faulty committee are discarded and these discarded transactions are processed in the subsequent epochs. In the presence of a faulty committee, the network needs to be resharded at the beginning of the next epoch based on the network sharding protocol. The previous discussion is summarized in the following:

Inter-Committee Consensus Protocol

10. Each network leader does the following:
    10.1. Receive intra-committee consensus message from committee leaders
    10.2. Authenticate and validate the consensus result of each message by additionally recalculating the Merkle root against the transaction ids

10.3. Mark a committee leader as faulty if the consensus message from the committee leader cannot be authenticated or the Merkle root in the message cannot be validated within a predefined timeout.

10.4. If there is a valid intra-committee consensus then

10.4.1. Accept the consensus result from the committee leader

10.4.2. Else, mark a committee as faulty if all its committee leaders are marked as faulty

10.5. Compare validation result of each overlapped transaction among all committees and consider 51% majority (i.e. $\frac{C+1}{2}$) as the valid consensus on the result of an overlapped transaction.

10.6. If a majority consensus cannot be reached on an overlapped transaction then

10.6.1. Mark the epoch as *Aborted*

10.6.2. Else

10.6.2.1. Mark a committee as faulty if its result of an overlapped transaction does not match with the majority consensus in step 10.5.

10.6.2.2. Prepare the list of the Merkle roots from all the good committees.

10.7. Follow the PBFT protocol among all network leaders on the results of step 10.6 (i.e. step 10.6.1 or 10.6.2)

10.8. If a consensus is not reached or a consensus is reached on as to abort epoch then

10.8.1. Abort the current epoch and discard the results of all transactions. These transactions will be processed in the subsequent epoch(s).

10.8.2. Else

10.8.2.1. Discard all transaction results assigned to faulty committees. These transactions will be processed in the subsequent epoch(s).

10.8.2.2. Broadcast agreed list of (Merkle root, committee id) to committee leaders of all committees

11. Each committee leader forwards the message from step 10.8.2.2 above to the committee members

Protocol 4. Inter-committee consensus protocol

4.3.5 (Lazy) Commit

Each node maintains the complete copy of the ledger for the benefits explained in section 4.2.3. Therefore, it needs all transactions from all the committees that are validated during the epoch to commit the final block(s) into its ledger. As transactions are sharded among committees, each committee member possesses only its own list of transactions. However, a message containing all the valid transactions can be quite large, hence for the performance reasons, network leaders send only the Merkle roots from the good committees to the committee leaders (protocol step 10.8.2.2) and members (protocol step 11). Each committee member subsequently receives the list of all valid transactions from all its *peer*s (a *peer* is defined in section 4.2.1.3) via peer-to-peer exchange. If the peer is faulty then it may not participate in peer-to-peer exchange protocol or send incorrect list of valid transactions. Therefore, receiving node validates the received list of valid transactions by calculating the Merkle root on it and compare against the Merkle root received from the network leaders. If the Merkle root cannot be validated or the list of valid transactions is not received from a peer then a committee member can request this information from the committee leaders. Committee leader may get this information from its peers or other committee members. At worst case, if all the committee leaders are also faulty then a node can connect with network leaders to get this information.

Network leaders need not wait for all good committees to acknowledge before committing the epoch; the epoch can be committed by the network leaders on a timed-out event. Hence, committing of the epoch by the committee members can proceed lazily for performance reasons. This is safe because the network leaders already have the valid ledger. Moreover, the parallelism involved in peer-to-peer exchange can relieve the inefficiency of one-to-all broadcast of all valid transactions by a network leader to all committee members. Any node that does not have the up-to-date ledger can request and receive the required data from network leaders. The steps of the protocol are summarized in the following:

(Lazy)Epoch-Commit Protocol

12. Each committee member does the following:
    12.1. Receive the list of Merkle roots from at least one committee leader (protocol step 11)

12.2. Peer-to-peer exchange its own list of valid transactions with all its peers. Recall that there is at least one peer from each other committee (protocol step 2.6)

12.3. Repeat steps 12.3.1 to 12.3.3 until lists of valid transactions are received from all the peers from good committees

12.3.1. Calculate the Merkle root of transactions received from a peer and compare with the one approved by network leaders (received in step 12.1). If Merkle root matches, then

12.3.1.1. Consider those transactions as candidates to commit. Go back to step 12.2.

12.3.2. If Merkle root does not match or list of valid transactions not received from a peer then ask committee leader(s) to share valid transactions for the specific committee(s)

12.3.3. If none of the committee leaders responded within a predefined timeout or if the received Merkle root(s) could not be validated, then report to network leaders and collect details of all valid transactions for specific committee(s) from network leaders.

12.4. After receiving transactions from all good committees, sort all valid transactions, including its own, in increasing order of transaction ids

12.5. Prepare block(s), each consisting of a predefined maximum number of transactions, chosen from the sorted sequence

12.6. Commit transactions by appending block(s) to the blockchain

12.7. Acknowledge to committee leader(s)

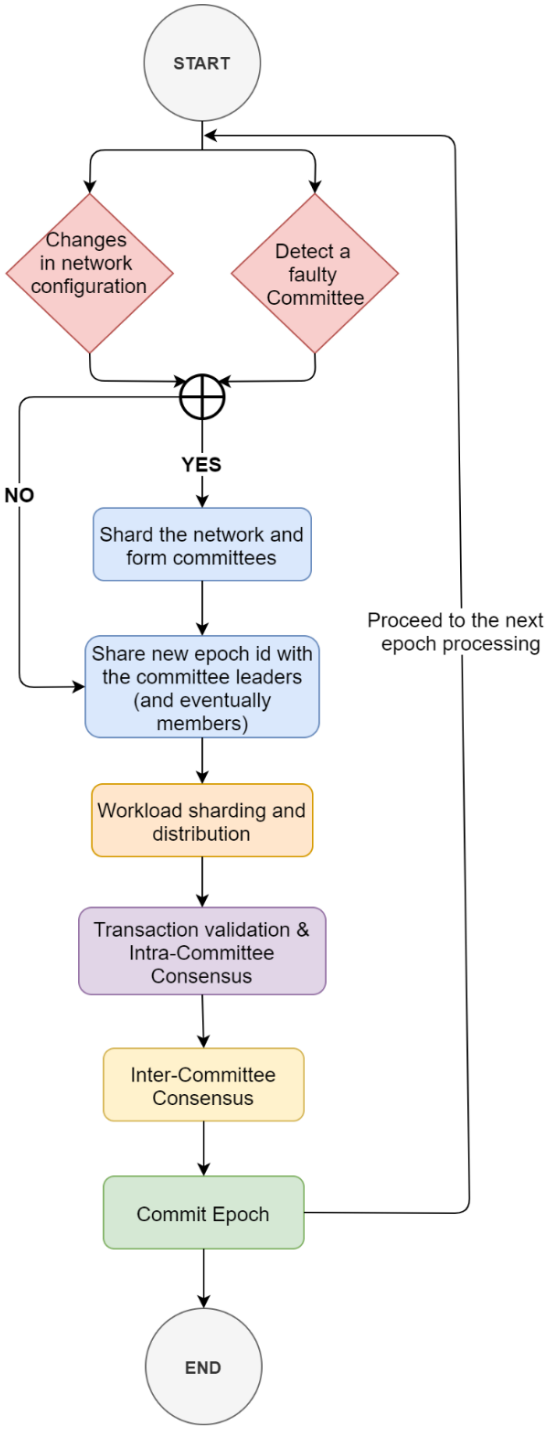
13. Committee leaders acknowledge to network leaders after receiving commit acknowledgement from at least $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ committee members

14. Network leaders mark the epoch as committed on receiving at least one valid acknowledgement message from each good committee or on a predefined *timed out*, whichever comes first.

Protocol 5 (Lazy)Epoch-commit protocol

4.3.6 The Epoch Processing in OptiShard

After discussing all the individual protocols, we summarize the complete protocol of epoch processing in OptiShard.



Flowchart 4.1. Epoch processing in OptiShard

Flowchart 4.1 depicts steps of epoch processing in OptiShard. It is important to note that network doesn't need to be re-sharded in the beginning of every epoch. After committing an epoch, network leaders re-shard the network only if they have observed any faulty committee or changes in the network configurations (e.g. new nodes joined the network). Otherwise, they share new epoch id with the existing committee leaders and proceed with the workload distribution step.

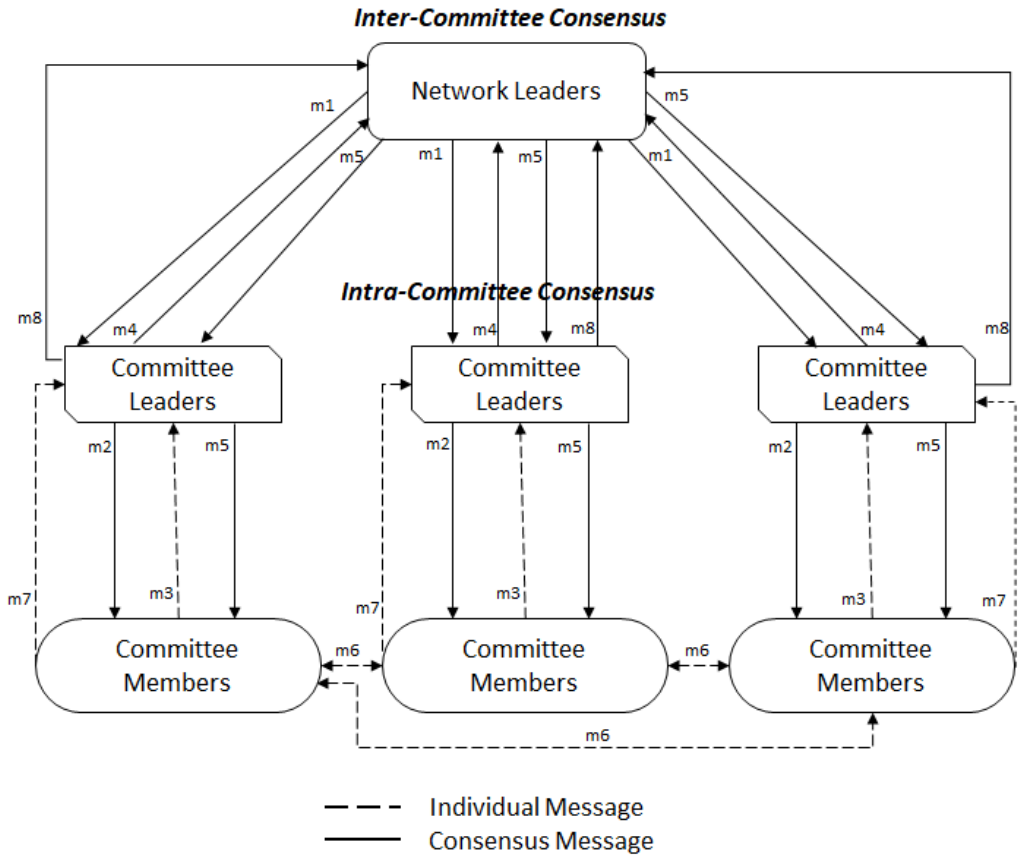


Figure 4.9. Communication diagram of an epoch processing

Figure 4.9 explains communication diagram of an epoch processing. Details regarding messages m1-m8 are explained as following:

- **m1**: Committee details, Epoch identity, Peer-to-Peer links, Transactions
- **m2**: Epoch identity, Peer-to-Peer links, Transactions
- **m3**: Merkle root of approved transactions
- **m4**: Merkle root approved by Intra-committee consensus
- **m5**: A list of Merkle roots approved by Inter-committee consensus

- **m6**: Approved transactions
- **m7-8**: Epoch commit acknowledgement

4.4 Correctness Proof

The following lemma is the consequence of the protocols discussed.

Lemma 2 (detect a faulty committee): A faulty committee that produces an incorrect consensus result is identified either during intra- or inter-committee consensus and all its transactions are discarded.

Proof:

A committee can be faulty due to the following reasons:

Case 1**:** *All its committee leaders are marked as faulty*.

Network leaders mark a committee leader as faulty if it either sends invalid intra-committee consensus result or is unresponsive due to crash or network failure (protocol step 10.3). Note that network leaders validate the consensus result by checking the Merkle root and signatures of the consenting nodes (protocol step 10.2) and hence can determine for certain if the consensus result has been altered. If all committee leaders of a committee are marked as faulty then the committee is marked as faulty (protocol step 10.4.2) and its transactions are discarded (protocol step 10.8.2.1).

Case 2**:** *At least* $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ *committee members are faulty*.

Case 2.1**:** *At least* $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ *of these faulty committee members produce incorrect result from the start of the epoch.*

By assumption (mentioned in section 4.2.2.44.2), once a node produces an incorrect result, it will produce incorrect results for all the remaining transactions for the rest of the epoch. As a result, there are three possibilities:

1. An intra-committee consensus cannot be reached. In that case, network leaders are informed (protocol step 9.2.1) and eventually all its transactions are discarded (protocol steps 10.4.2 and 10.8.1).

2. Intra-committee consensus is reached with correct consensus result. In that case, the consensus result is equivalent of a good committee and its transaction results are accepted.

3. An incorrect intra-committee consensus is reached with majority of faulty nodes as consenting members. In that case, the committee will be identified as a faulty committee during inter-committee consensus due to its non-agreement of the overlapped transactions with the majority of good committees (protocol steps 10.5 and 10.6).

Case 2.2**:** *At most* $\left\lfloor \frac{C_n - 1}{3} \right\rfloor$ *of the faulty members produce incorrect results from the start of the epoch*.

This case can be treated the same way as case 2.3 below.

Case 2.3**:** *The committee was a good committee to start with. However, after processing at least one transaction, it turns faulty due to some of the good nodes turning faulty*.

In the worst case scenario, if at least $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$ good nodes turn faulty exactly after processing the same number of transactions, which include all the overlapped transactions, then it is possible that an incorrect consensus is reached which can go undetected through the overlapped transactions because the consensus produced correct results for all the overlapped transactions. In any other scenario, one of the three possibilities of case 2.1 arises and can be handled in a similar way to case 2.1. We have simulated probabilistic experiments to measure the experimental probability of majority (i.e. at least $\left\lfloor \frac{C_n - 1}{3} \right\rfloor + 1$) of the good nodes turn faulty exactly after processing the same number of transactions, which is discussed further in section 5.2.2.

This concludes the proof. ■

4.5 Theoretical Performance Analysis

Sharding of the network nodes into multiple committees and dividing transactions among committees transform the traditional monolithic blockchain architecture into a hierarchical one. Since the committees can work independently and in parallel on their own set of transactions, OptiShard offers an obvious performance gain over the traditional monolithic blockchain architecture. There is also performance gain in consensus due to its hierarchy: first intra-committee consensus which can run in parallel among committees and subsequently inter-committee consensus. OptiShard follows a modified variant of PBFT protocol for intra-committee consensus, which optimizes communication among nodes.

Performance of an epoch processing is dominated by the following: transaction validation, intra-committee consensus, and inter-committee communication to exchange valid transactions required to commit an epoch. As the workload is divided among all committees and nodes within a committee work in parallel, transaction validation time complexity is proportional to transaction shard size, i.e. bounded by $O(\frac{W}{C})$. As committee leaders collect the results during intra-committee consensus or forwards a message to committee members, our protocol does not require peer-to-peer communication among committee members; therefore, intra-committee communication and consensus time is linear in proportion to the committee size, i.e., bounded by $O(C_n)$. Nodes exchange transactions with their peers across the committees during the commit phase, which requires at least a quadratic number of messages in terms of total committees, C, i.e., it is of the order of $O(C^2)$.

Let t_t be the time to process a single transaction, t_m be the average time to send one message during intra-committee consensus, and t_g be the average time to send transactions validated by a committee during the commit phase, then the estimated dominating theoretical runtime can be expressed as

$$F(C) = t_m C_n + t_g C^2 + t_t \frac{W}{C} \quad (2)$$

We can rewrite (2) as a function of total committees, C, as:

$$F(C) = t_m \frac{N}{C} + t_g C^2 + t_t \frac{W}{C} \quad (3)$$

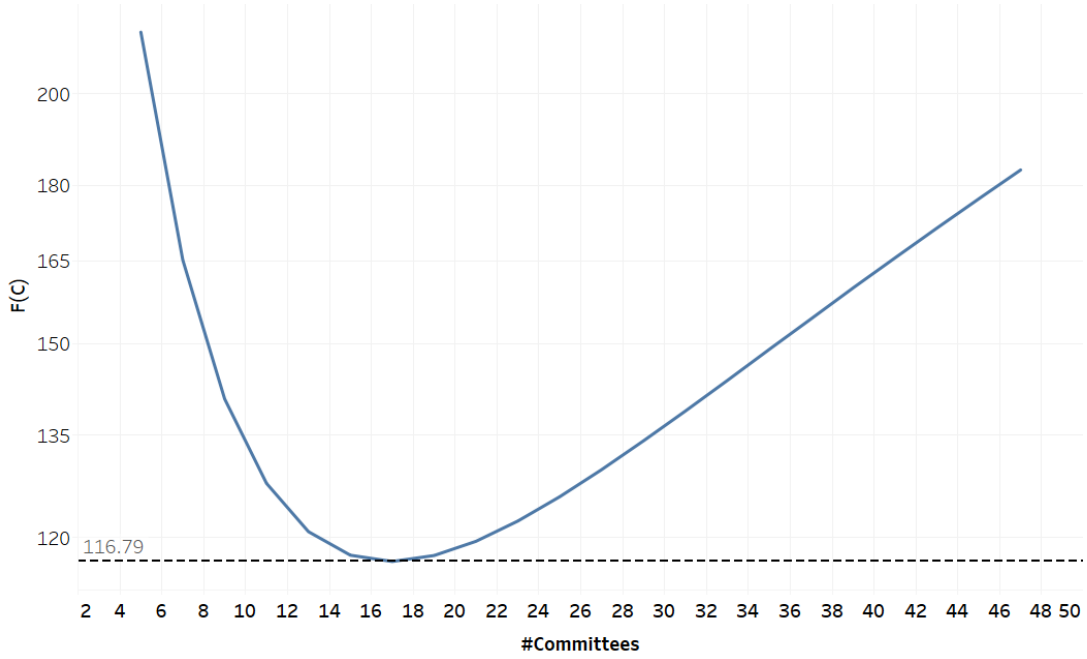


Figure 4.10. $F(C)$ versus C for W=98.6k, t_m= 0.00035, t_t= 0.01, and N = 400

Figure 4.10 plots the graph of $F(C)$ versus C, keeping network size and workload constant. We can observe that $F(C)$ decreases up to a certain point and then starts increasing as C increases. The C value for the optimal performance is given at the point where the gradient is 0, i.e., $\frac{\mathrm{d}F(c)}{\mathrm{d}c} = 0$. We call this curve as *performance curve* in our discussion.

Combining the results of equation (3) with Lemma 1, we can design an optimal sharding scheme by choosing a suitable value of C from the two important perspectives: performance and security. There is a possibility that C value for optimal performance does not fulfill the security criteria defined in Lemma 1 (i.e. more than half of the committees are good). Nevertheless, Lemma 1 offers multiple values satisfying the security condition. Therefore, we choose a C value on performance curve that satisfies Lemma 1 and closest to the optimal point (i.e. zero gradient) on either side (i.e. left or right side to the optimal point on the performance curve).

4.5.1 How to Choose The C Value

4.5.1.1 Optimal Value of c on Performance Curve Fulfills Lemma 1

In that case we consider the optimal value on performance curve as the number of network shards (i.e. committees).

1. c = The optimal point (i.e. $\frac{\mathrm{d}F(c)}{\mathrm{d}c} = 0$) on performance curve
2. **if** c satisfies Lemma 1:
3. **return** c;

Algorithm 4.1 Optimal value of c on performance curve fulfills Lemma 1

4.5.1.2 Optimal Value of c on Performance Curve doesn't Fulfill Lemma 1

If optimal c doesn't fulfill Lemma 1, we need to look for a c value, which fulfills Lemma 1 and also offers better performance comparatively. Our performance curve is inverted bell-shaped curve; hence we get two different x-values (i.e. c) for a single y-value (i.e. system performance $F(c)$) as shown in Figure 4.11.

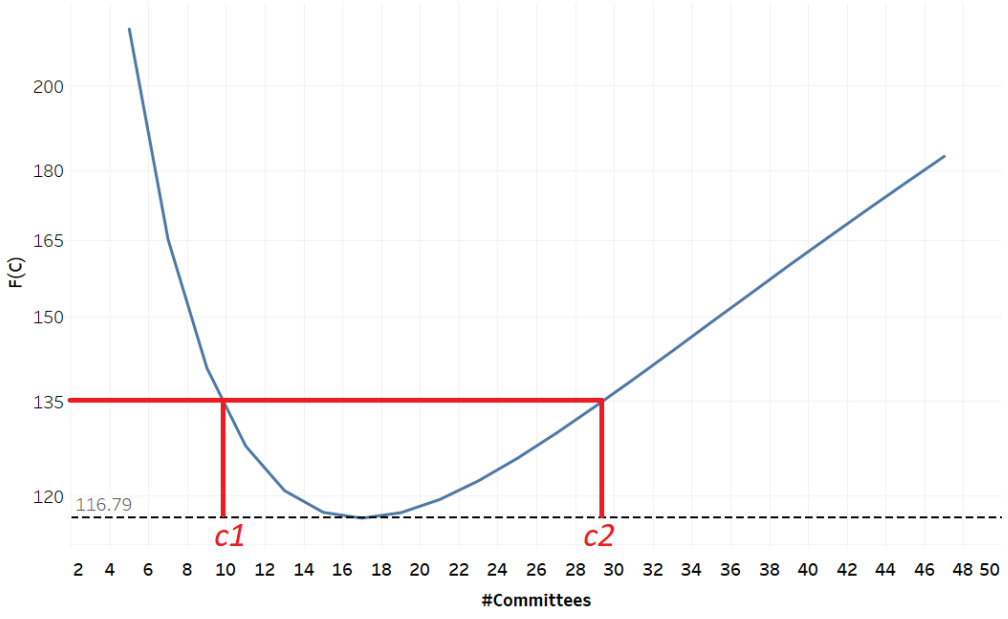


Figure 4.11 Multiple c values offer similar performance

Also, it is important to note that y-value increases, viz. system performance degrades, as we go farther on either side of the optimal point on the x-axis. Therefore, we traverse on both the sides of optimal point to look for valid c values that satisfy Lemma 1; calculate and compare system performance for both the values; and choose the one with better performance (i.e. smaller y-value).

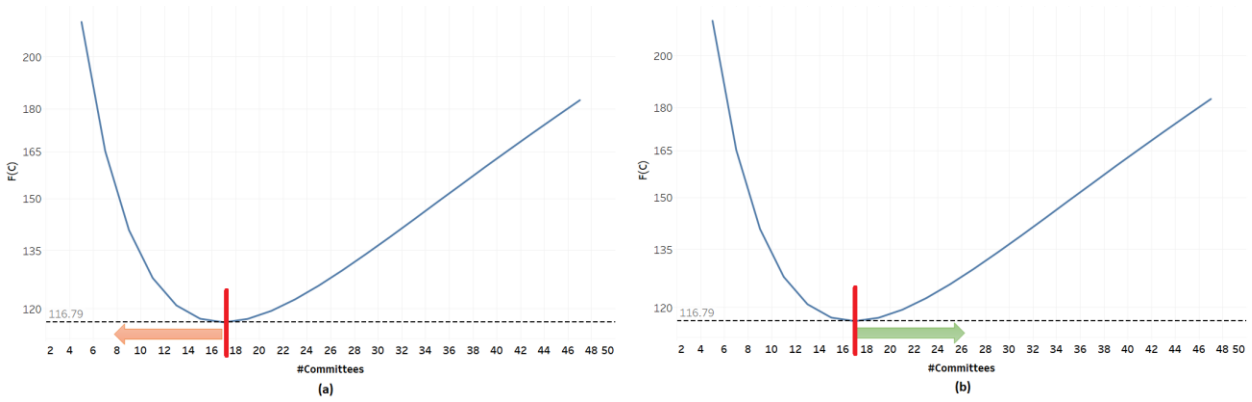


Figure 4.12 (a) Traverse to the left of optimal c. (b) Traverse to the right of optimal c

To start with, traverse on the left side of the optimal point as shown in Figure 4.12 (a), until we find a positive odd value of c that fulfills Lemma 1. We denote that as *left_c* for our discussion.

```
1.  for each odd number v in range [c,1]:
2.      if v satisfies Lemma 1:
3.          left_c = v
4.          break;
```

Algorithm 4.2 Choose the most appropriate c to the left of optimal point

Also traverse on the right side of the optimal point as shown in Figure 4.12 (b), until we find a positive odd number that satisfies Lemma 1. We denote that as *right_c* for our discussion.

```
1.  for each odd number v in range [c,N]:
2.      if v satisfies Lemma 1:
3.          right_c = v
4.          break;
```

Algorithm 4.3 Choose the most appropriate c to the right of optimal point

Both right and left values as calculated above fulfill Lemma 1, hence shard the network by choosing the one that offers better performance.

```
1. if F(left_c) > F(right_c):
2.     return right_c
3. else:
4.     return left_c
```

Algorithm 4.4 Choose the most appropriate c between left and right c values

4.6 Applications

The hybrid model of OptiShard offers network governance in completely decentralized environment. The network is consisting of a small fraction of trusted nodes deployed by the governing body and the rest are public nodes. This setup offers reliability of the critical data in the presence of public nodes. Applications of OptiShard are not just limited to finance domain. It can also be deployed in any hybrid and private blockchain environments (e.g. use cases in health care, identity management and voting).

Health care records of a patient are once committed in the ledger, become immutable and accessible forever. The documents are secured in public environment because they are encrypted using public private key cryptography. A physician or medical authority may easily access the details by the consent of respective patient. An enterprise blockchain service provider company Guardtime, deployed a health care blockchain in Estonia to store about 1 million health records [48]. Similarly, government bodies may persist national identity and other contract documents on a public ledger. Recently, The State of West Virginia used blockchain based mobile voting application called Voatz, for their overseas voters to vote in the 2018 U.S. midterm election [49]. The Government of Canada also launched first-ever live trial of public blockchain technology for the transparent administration of government contracts in January 2018 [50].

5 Experimental Analysis

We used up to 800 Amazon Web Services (AWS) ec2 micro instances with 1 GB RAM and 1 CPU core, across different regions to execute our test scenarios. We programmed our code in Scala, considering a scalable JVM based language that supports heterogeneous platforms. For intra-committee consensus, a variant of PBFT was used as explained in section 4.3.3. We used individual MySql instances on every node for structured storage. Instead of creating random dummy transactions on runtime, we kept a set of transactions pre-populated for valid comparison across different runs.

Every node follows peer-to-peer communication to exchange information with its peers across different committees. Every message is signed by the secured key of the sender and validated by its public key at the receiving end. We used Java BouncyCastle implementation of SHA256 with ECDSA encryption algorithm for the simplicity. Each node is assigned a secure-public key pair. Participating nodes exchange their public keys during initial setup. The following experiments are mainly for measuring the throughput and validate theoretical analysis of OptiShard.

5.1 Throughput Comparison

5.1.1 TCP vs UDP Protocol for The Communication

We performed this experiment to measure and compare the communication costs for TCP and UDP protocols in our network. As part of this experiment, each node sends a message of fixed size to rest of the network nodes. Nodes do not implement any special broadcasting technique, instead execute send operation sequentially. We executed this experiment for different network sizes, each with TCP and UDP protocols for peer-to-peer communication.

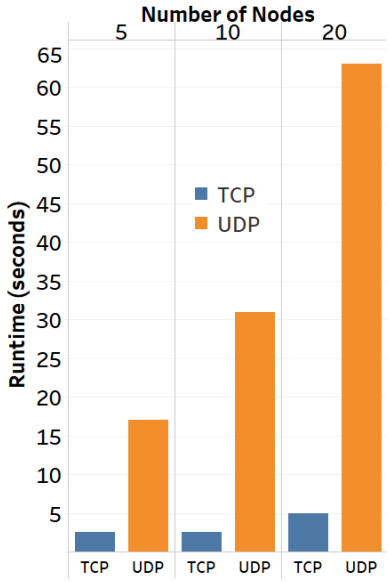


Figure 5.1 Communication costs comparison of TCP and UDP protocols across different network sizes

We can observe in Figure 5.1 that TCP runtime is much better compared to UDP across all the network sizes. Therefore, we follow TCP protocol to establish peer-to-peer communication in OptiShard.

5.1.2 Runtime vs Number of Committees

Our theoretical analysis indicates that system performance improves (i.e. runtime decreases) up to certain point as we increase committees (i.e. generating more network shards), while keeping rest of the parameters constants. Increasing number of committees further degrades the overall system performance. Therefore, plotting a graph of total number of committees against theoretical runtime gives an inverted bell shape curve as shown in Figure 4.10. To validate this theoretical analysis, we measured the runtime for various number of committees in the same network with identical workload across all the experiments.

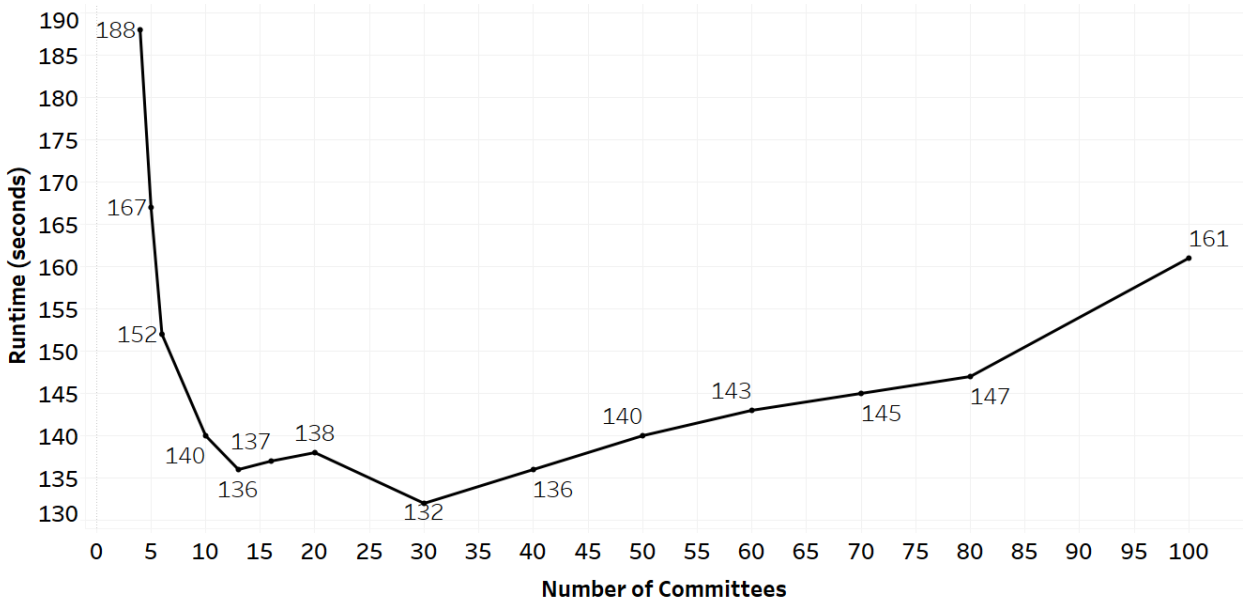


Figure 5.2. Number of committees vs Runtime

Figure 5.2 shows experimental runtime, while increasing number of committees from 3 to 99, keeping number of transactions as 98.6k constant in a network consisting of 600 nodes. We can observe that runtime decreases up to 30 committees as we increase number of committees; and then starts increasing if we continue increasing number of committees. Therefore, system offers optimal performance with 30 committees for the given network configurations. We can validate our theoretical analysis by comparing graphs in Figure 4.10 and Figure 5.2.

5.1.3 Runtime Is Linearly Proportional to Committee Size

With a fixed total number of transactions and committees, the number of transactions assigned to a committee remains intact. Thus, the workload of a committee member remains unchanged with an increased committee size, because each member validates all the transactions assigned to the committee. However, our modified variant of PBFT protocol for intra-committee consensus is in linear proportion of committee size C_n, therefore theoretically, runtime should only increase linearly with the increase in the committee size, keeping total number of transactions and committees fixed. We did this experiment to conform linear increase of runtime with increased committee size.

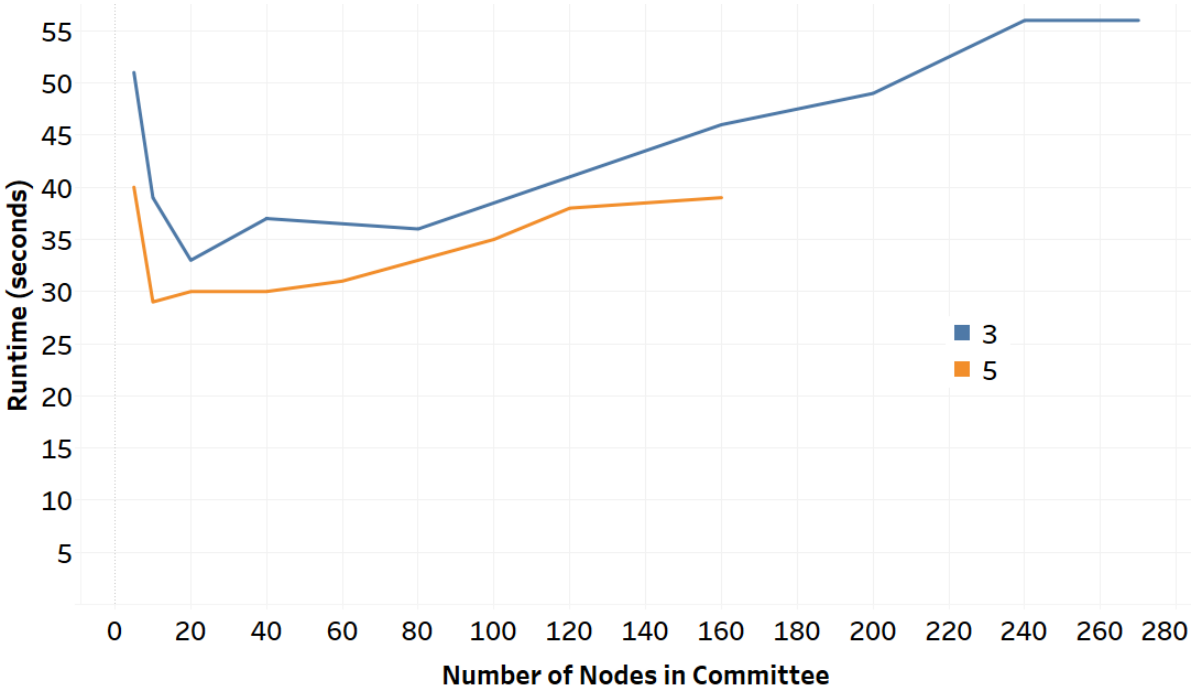


Figure 5.3 Scaling committee size while network processes 98.6k transactions with 3 and 5 committees.

Figure 5.3 shows the experimental runtime while increasing committee size from 3 to 270, keeping identical workload across each experiments and total number of committees as 3. Additionally, we performed similar experiment for total 5 committees in the network, which is also plotted in Figure 5.3. We can observe that runtime increases near linearly to the committee size in our modified variant of PBFT protocol for intra-committee consensus.

5.1.4 Scaling Factor

In a hierarchical sharding network, in addition to transaction validation cost, additional fixed costs are associated to shard the network; distribute the work; and consensus. Therefore, increasing the workload and network size linearly, doesn't offer constant throughput in a sharded network. Hence, we measured the runtime for different workloads and network sizes and compared them using *scaling factor* to observe the impact of increased network size and workload on throughput. We define *scaling factor* as the ratio of the runtime of a network consisting of $2N$ nodes sharded in $2C$ committees to the runtime of a network consisting of N nodes sharded in C committees, keeping all other parameters unchanged.

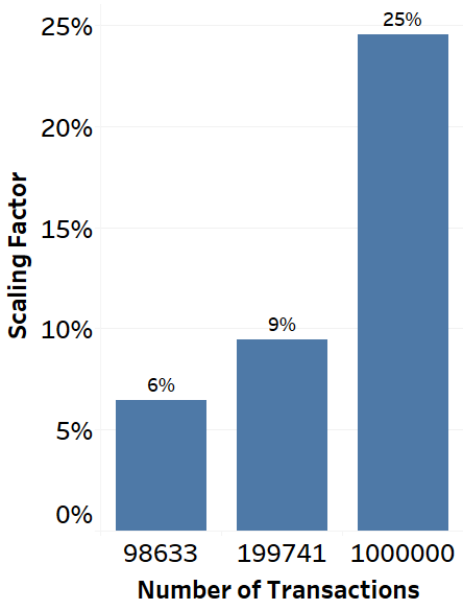


Figure 5.4. Scaling factor for different workloads

Experimental results in Figure 5.4 shows a scaling factor of 25% for 1M transactions; the results also demonstrate that scaling factor improves as the workload is increased, which implies that the system offers better throughput with increased workload, while doubling the size of the network and keeping committee size constant (i.e. doubling total number of committees). Therefore, we can conclude that system offers better speed up (i.e. hierarchical to monolithic network performance) with increased workload in the same network.

5.2 Simulation for The Probabilistic Experiments

We form committees by choosing network nodes in round robin fashion from a list of nodes sorted based on non-increasing order of their stake values. As our network is composed of certain percentage of malicious nodes and their identities are anonymous, there is a probability that such random distribution produces a committee consisting of majority of the malicious nodes.

Additionally, our fault model allows a node to become faulty at any time during epoch processing. Hence, a good committee at the beginning of epoch, turns faulty during epoch processing, if majority of the committee members turn faulty. If a good committee turns faulty due to majority of the nodes turning faulty exactly after processing the same number of transactions; positioned after all the overlapped transactions in the list of transactions sorted in non-decreasing

order of transaction id; then such faulty committee may go undetected and populate incorrect result. This scenario is also discussed in Lemma 2 (section 4.4).

Therefore, we discuss and demonstrate the probability of such randomly generated events in the remaining part of this chapter. We programmed a Python simulator to measure the experimental probability. To make our experiments more general, we used generic *random* library provided with python3 distribution to generate the randomness. In the following, we discuss our experiments along results.

5.2.1 Faulty Committees from The Beginning of The Epoch

For these experiments, network is consisting of a fixed number of nodes; whereas total number of malicious nodes and committees in the network are variables. We assume all the nodes are carrying equal stake value for these experiments, hence we do not consider stake value while sharding the network. A predefined number (i.e. $f\%$ of the network size) of randomly chosen nodes are marked as faulty in the list consisting of all the network nodes sorted based on their node id. The nodes are distributed in round robin fashion among network shards to form committees of equal size, C_n. Sorting nodes based on id and distributing among committees in round robin fashion works, because we consider each node with equal stake value. We record total number of faulty committees (i.e. more than $\left\lfloor \frac{C_n-1}{3} \right\rfloor$ nodes are faulty) formed by such random distribution of nodes among committees.

We experimented above discussed simulation with a network consisting of 999 nodes (i.e. N=999) and various combinations of total number of faulty nodes in the network and the committees. Following tables indicate the instances of faulty committees resulted in various configurations; each configuration experimented 1M times consecutively.

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments	
		n=1	n=2
3	333	0	0
5	199	0	0
7	142	0	0
9	111	41	0
11	90	595	0
13	76	1435	0
15	66	9197	13
17	58	11421	28
19	52	22767	120

Table 5.1 $N = 999, f = 17\%$ (i.e. 170 faulty nodes)

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments		
		n=1	n=2	n=3
3	333	0	0	0
5	199	0	0	0
7	142	7	0	0
9	111	209	0	0
11	90	1861	0	0
13	76	4141	3	0
15	66	21462	0	0
17	58	24977	116	0
19	52	46116	520	2

Table 5.2 $N = 999, f = 18\%$ (i.e. 180 faulty nodes)

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments			
		n=1	n=2	n=3	n=4
3	333	0	0	0	0
5	199	0	0	0	0
7	142	25	0	0	0
9	111	753	0	0	0
11	90	5567	1	0	0
13	76	10253	13	0	0
15	66	44961	395	0	0
17	58	49988	587	0	0
19	52	83989	1989	18	1

Table 5.3 $N = 999, f = 19\%$ (i.e. 190 faulty nodes)

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments			
		n=1	n=2	n=3	n=4
3	333	0	0	0	0
5	199	2	0	0	0
7	142	150	0	0	0
9	111	2495	0	0	0
11	90	14265	20	0	0
13	76	23298	98	1	0
15	66	85916	1832	10	0
17	58	91620	2292	20	0
19	52	143239	6954	101	1

Table 5.4 $N = 999, f = 20\%$ (i.e. 200 faulty nodes)

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments			
		n=1	n=2	n=3	n=4
3	333	0	0	0	0
5	199	9	0	0	0
7	142	602	0	0	0
9	111	7261	2	0	0
11	90	32381	170	0	0
13	76	49255	480	0	0
15	66	151395	6840	80	0
17	58	157185	7860	117	0
19	52	224300	19529	625	6

Table 5.5 $N = 999, f = 21\%$ (i.e. 210 faulty nodes)

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments				
		n=1	n=2	n=3	n=4	n=5
3	333	0	0	0	0	0
5	199	59	0	0	0	0
7	142	2205	0	0	0	0
9	111	18909	0	0	0	0
11	90	68745	862	0	0	0
13	76	94586	2144	17	0	0
15	66	242472	21213	585	5	0
17	58	242825	23082	803	12	0
19	52	318010	49708	3071	78	2

Table 5.6 $N = 999, f = 22\%$ (i.e. 220 faulty nodes)

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments				
		n=1	n=2	n=3	n=4	n=5
3	333	0	0	0	0	0
5	199	402	0	0	0	0
7	142	7173	3	0	0	0
9	111	44033	226	0	0	0
11	90	130829	3762	42	0	0
13	76	166421	7841	115	0	0
15	66	347009	58212	3323	69	0
17	58	340797	58891	3842	113	1
19	52	400276	109304	12149	573	12

Table 5.7 $N = 999, f = 23\%$ (i.e. 230 faulty nodes)

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments				
		n=1	n=2	n=3	n=4	n=5
3	333	0	0	0	0	0
5	199	1819	0	0	0	0
7	142	19784	17	0	0	0
9	111	92869	1292	3	0	0
11	90	226679	14474	221	0	0
13	76	264140	24994	653	7	0
15	66	430543	132231	15157	692	9
17	58	420281	127065	15595	772	13
19	52	436883	201242	39798	3567	152

Table 5.8 $N = 999, f = 24\%$ (i.e. 240 faulty nodes)

#Committees C	Committee Size C_n	Occurrence(s) of n faulty committees in total 1M experiments					
		n=1	n=2	n=3	n=4	n=5	n=6
3	333	45	0	0	0	0	0
5	199	7045	2	0	0	0	0
7	142	49856	205	0	0	0	0
9	111	174550	6480	40	0	0	0
11	90	343789	46464	1727	18	0	0
13	76	372708	67551	3880	74	0	0
15	66	445046	245048	53040	4689	149	3
17	58	440795	229400	49686	4866	205	1
19	52	396367	304209	103468	16978	1341	49

Table 5.9 $N = 999, f = 25\%$ (i.e. 250 faulty nodes)

5.2.2 A Committee Turning Faulty During Epoch Processing

If majority of the nodes turn faulty right at the same transaction, after processing all the overlapped transaction, then such faulty committee generates a corrupted yet valid intra-committee consensus result, that will not be detected during inter-committee consensus. Theoretically, the probability of m nodes within a committee of size C_n, randomly choosing the same transaction from the sorted list of total t transactions can be derived as $\left(\frac{1}{t}\right)^m$.

We executed multiple experiments to observe the experimental probability of majority of the good nodes turning faulty from the same transaction during transaction validation phase. Every node picks a random transaction from a list of assigned transactions, sorted in non-decreasing order of transaction id. In these experiments, we didn't notice a single instance, when at least majority (i.e. $\left\lfloor\frac{C_n-1}{3}\right\rfloor + 1$) of the nodes within a committee have randomly chosen the same transaction.

We kept total number of transactions as 1000 fixed and experimented above explained set up 1M times for each committee size as mentioned in following Table 5.10.

Committee Size C_n	Majority	Occurrences of majority of the nodes turn faulty from the same randomly chosen transaction in 1M experiments for each Cn
51	17	0
101	34	0
151	51	0
201	67	0
251	84	0
301	101	0
351	117	0
401	134	0
451	151	0
501	167	0
551	184	0
601	201	0
651	217	0
701	234	0
751	251	0
801	267	0
851	284	0
901	301	0
951	317	0
1001	334	0

Table 5.10 Occurrences of majority of the nodes turn faulty from the same randomly chosen transaction in 1M experiments for each C_n value

6 Conclusion & Future Works

We propose a hierarchical blockchain architecture OptiShard, which addresses the issues of performance, security, and fault tolerance in traditional blockchain architecture. The hierarchy divides the network into multiple disjoint shards called committees and workload is distributed among the committees. Optimal shard size is determined based on three parameters: performance, security, and fault tolerance. OptiShard provides improved transaction validation time with both increasing number of committees and increasing number of transactions, and guarantees a majority of good committees subject to a maximum allowable number of faulty nodes. Overlapping of transactions across committees is one of the mechanisms of identifying faulty committees. Experimental results demonstrate encouraging performance improvement and conform to the theoretical analysis.

There are certain assumptions on the fault model which become necessary to address the correctness of the discussed protocols. We assumed a faulty node always produces corrupted result. Also, a majority of the nodes within a committee do not turn faulty from the same transaction. These assumptions can be avoided by having a more restricted model where the committee leaders are more trustworthy or network leaders do additional validations.

A committee leader produces intra-committee consensus only if majority of the nodes have produced and sent the same validation result as produced by committee leader. Therefore, if committee leader(s) are also trust worthy then, we can guarantee the correctness of the intra-committee consensus and eventually the correctness of the transaction validation.

Second approach could be to assign all the responsibilities of committee leaders to network leaders and completely decommission the role of committee leader. In addition to taking care of all the communication with committee members, network leaders also validate the transactions and accept the transaction validation result from committee members if its matching with their own validation result. Network leaders are the trusted nodes; hence correctness of the transaction

validation holds. We need to make sure only a subset composed of majority of the good network leaders leads a committee; otherwise network leaders have to validate all transactions, that violates the fundamental work distribution principle. This approach completely centralizes the trust around network leaders, because only network leaders have authority to validate a transaction, that.

As we discussed previously, any of the mentioned approaches makes the model more centralized and may add performance overhead. Therefore, there is a compromise between centralization and freedom of peer-to-peer validation as in a traditional blockchain. These issues will be addressed in our future works.

Lastly, workload imbalance is very common and well-known problem in any sharded system. Therefore, we also consider dynamic load balancing across the committees in our future work. ■

References

[1] C. George, D. Jean, K. Tim, and B. Gordon, *Distributed Systems Concepts and Design*, 5th ed. Pearson, ch. 1 pp. 1-33.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system." Bitcoin.org, 2009.

[3] The Linux Foundation, "Blockchain for Business," *edx.org*. [Online]. Available: https://www.edx.org/professional-certificate/linuxfoundationx-blockchain-for-business. [Accessed: 13-Nov-2019].

[4] Bitcoin Wiki, "Genesis block." [Online]. Available: https://en.bitcoin.it/wiki/Genesis_block. [Accessed: 13-Nov-2019].

[5] M. Ralph C, "A Digital Signature Based on a Conventional Encryption Function," presented at the CRYPTO'87, 1988, vol. 293, pp. 369–378.

[6] M. H. Miraz and M. Ali, "Applications of Blockchain Technology beyond Cryptocurrency," *AETiC*, vol. 2, pp. 1–6, Jan. 2018.

[7] S. Raval, *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*, 1st ed. O'Reilly Media, Inc., 2016.

[8] S. Lee, "Bitcoin's Energy Consumption Can Power An Entire Country But EOS Is Trying To Fix That," *forbes.com*. [Online]. Available: https://www.forbes.com/sites/shermanlee/2018/04/19/bitcoins-energy-consumption-can-power-an-entire-country-but-eos-is-trying-to-fix-that/#41e6a36c1bc8. [Accessed: 19-Aug-2019].

[9] "Bitcoin 51% attack." [Online]. Available: https://bitcoin.org/en/glossary/51-percent-attack. [Accessed: 18-Aug-2019].

[10] J. R. Douceur, "The Sybil Attack," presented at the The First International Workshop on Peer-to-Peer Systems, 2002, vol. 2429, pp. 251–260.

[11] K. Croman *et al.*, "On scaling decentralized blockchains (a position paper)," presented at the Financial Cryptography and Data Security, Berlin, Heidelberg, 2016, vol. 9604, pp. 106–125.

[12] "Visa's transactions per second," *Visa Inc*. [Online]. Available: https://usa.visa.com/content_library/modal/benefits-accepting-visa.html. [Accessed: 18-Aug-2019].

[13] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where Is Current Research on Blockchain Technology?—A Systematic Review," *PLOS ONE*, 2016.

[14] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," presented at the OSDI'99, New Orleans, Louisiana, 1999.

[15] "Proof-Of-Stake." [Online]. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ. [Accessed: 18-Aug-2019].

[16] X. Xu *et al.*, "A Taxonomy of Blockchain-Based Systems for Architecture Design," presented at the ICSA, Gothenburg, Sweden, 2017.

[17] "Difference between Public and Private Blockchain." [Online]. Available: https://www.ibm.com/blogs/blockchain/2017/05/the-difference-between-public-and-private-blockchain/.

[18] "Bitcoin - ELI5: what is a target? and how it is calculated?," *reddit*. [Online]. Available: https://www.reddit.com/r/Bitcoin/comments/1rerhx/eli5_what_is_a_target_and_how_it_is_calculated/. [Accessed: 28-Mar-2019].

[19] "SHA256 Hashing Algorithm." [Online]. Available: https://en.bitcoinwiki.org/wiki/SHA-256. [Accessed: 13-Nov-2019].

[20] “Difference between CPU and GPU.” [Online]. Available: https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/. [Accessed: 13-Nov-2019].
[21] “Peercoin and Proof-of-Stake.” [Online]. Available: https://university.peercoin.net/#/9-peercoin-proof-of-stake-consensus. [Accessed: 13-Nov-2019].
[22] P. Vasin, “BlackCoin’s Proof-of-Stake Protocol v2,” 2014.
[23] “Token vs Coin.” [Online]. Available: https://blog.chronobank.io/token-vs-coin-whats-the-difference-5ef7580d1199. [Accessed: 18-Aug-2019].
[24] “Nxt.” [Online]. Available: https://nxtplatform.org/what-is-nxt/. [Accessed: 18-Aug-2019].
[25] “Delegated Proof of Stake.” [Online]. Available: https://en.bitcoinwiki.org/wiki/DPoS. [Accessed: 18-Aug-2019].
[26] BitShares, “BitShares DPoS,” *bitshares.org*. [Online]. Available: https://bitshares.org/technology/delegated-proof-of-stake-consensus/.
[27] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” in *ACM*, 1982, vol. 4, pp. 382–401.
[28] Y. Sompolinsky and A. Zohar, “Secure High-Rate Transaction Processing in Bitcoin,” in *FC 2015*, 2015, pp. 507–527.
[29] I. Eyal, A. Efe Gencer, E. Gün Sirer, and R. van Renesse, “Bitcoin-NG: A Scalable Blockchain Protocol,” in *USENIX*, Santa Clara, CA, 2016, pp. 45–59.
[30] 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*, Shanghai, China, 2017, pp. 51–68.
[31] C. George, D. Jean, K. Tim, and B. Gordon, *Distributed Systems Concepts and Design*, 5th ed. Pearson, pp. 659-671.
[32] S. Micali, M. Rabin, and S. Vadhan, “Verifiable Random Functions,” in *40th Annual Symposium on Foundations of Computer Science*, New York City, NY, USA, 1999, pp. 120–130.
[33] M. Vukolić, “The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication,” presented at the Open Problems in Network Security – iNetSec’15, 2016, vol. 9591, pp. 112–125.
[34] “IBM Fabric.” [Online]. Available: https://www.ibm.com/blogs/blockchain/category/blockchain-development/hyperledger-fabric/. [Accessed: 14-Nov-2019].
[35] “Bitcoin transaction commit latency.” [Online]. Available: https://www.blockchain.com/charts/avg-confirmation-time. [Accessed: 14-Nov-2019].
[36] G. Danezis and S. Meiklejohn, “Centrally banked cryptocurrencies,” presented at the NDSS’16, 2016, pp. 21–24.
[37] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, “Chainspace: A Sharded Smart Contracts Platform,” presented at the arXiv:1708.03778, 2017.
[38] J. Sousa and A. Bessani, “From Byzantine Consensus to BFT State Machine Replication: A Latency-Optimal Transformation,” presented at the 2012 Ninth European Dependable Computing Conference, Sibiu, Romania, 2012, pp. 37–48.
[39] C. George, D. Jean, K. Tim, and B. Gordon, *Distributed Systems Concepts and Design*, Fifth. Pearson.

[40] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, “A Secure Sharding Protocol For Open Blockchains,” presented at the CCS’16, Vienna, Austria, 2016, pp. 17–30.
[41] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Fordy, “OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding,” in *2018 IEEE Symposium on Security and Privacy*, San Fransisco, USA, 2018.
[42] E. Syta *et al.*, “Scalable Bias-Resistant Distributed Randomness,” in *38th IEEE Symposium on Security and Privacy*, 2017.
[43] M. Zamani, M. Movahedi, and M. Raykova, “Rapidchain: Scaling blockchain via full sharding,” in *The 2018 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2018, pp. 931–948.
[44] J. A. Rice, *Mathematical Statistics and Data Analysis*, 3rd ed. Duxbury Press, 2007.
[45] M. O. Rabin, “The Information Dispersal Algorithm and its Applications,” in *Capocelli R.M. (eds) Sequences*, 1990, pp. 406–419.
[46] J. Wang and H. Wang, “Monoxide: Scale out Blockchains with Asynchronous Consensus Zones,” in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’19).*, Boston, MA, USA, 2019.
[47] W. Vogels, “Eventually Consistent,” *Commun. ACM*, vol. 52, no. 1, pp. 40–44, Jan. 2009.
[48] K. Tsung-Ting, K. Hyeon-Eui, and O. Lucila, “Blockchain distributed ledger technologies for biomedical and health care applications,” *Journal of the American Medical Informatics Association*, vol. 24, no. 6, pp. 1211–1220, Nov. 2017.
[49] L. Moore and N. Sawhney, “The West Virginia Mobile Voting Pilot.” [Online]. Available: https://sos.wv.gov/FormSearch/Elections/Informational/West-Virginia-Mobile-Voting-White-Paper-NASS-Submission.pdf. [Accessed: 14-Nov-2019].
[50] “Exploring blockchain for better business,” 20-Aug-2018. [Online]. Available: https://nrc.canada.ca/en/stories/exploring-blockchain-better-business. [Accessed: 14-Nov-2019].