A Raft-based Private/Consortium Blockchain Network

Yufeng Liu, Zhe Wang, Zhihao Huang, Zisheng Cheng
{yufeng2, zwwang4, zhihhuang, zishengc}@student.unimelb.edu.au

Abstract

Blockchain is an innovative technology that is a new shift in the Internet world, it has impacted on our lifestyles in the last decades. As a ditributed, transparent and immutable ledger, blockchain is facing many challenges. Consensus is one of the key problems in blockchains. In this paper, a comparative and analytical review on popular blockchain consensus algorithms is presented in terms of performance, use cases and fault tolerant. Enlightened by the strengths and constraints of each algorithm, we select to implement a Raft-based private/consortium blockchain application. Users are able to add transactions, retrieving value of certain keys in a consistent, crash tolerant manner.

Keywords: Blockchain, Consensus Algorithms, Raft

Contents

1 Introduction **3**

2 Background **3**
2.1 Blockchain 3
2.2 Blockchain Challenges 4
2.3 Consensus 4

3 Related Work **5**
3.1 Consensus Algorithms 5
3.2 Critical Analysis 7
3.3 Raft Related Work 8

4 Application **8**
4.1 Private/Consortium Blockchain 8
4.2 Application Details 9

5 Raft Algorithm **10**
5.1 Key Concepts 10
5.2 Working Mechanism 11
5.3 Discussion 12

6 Implementation **13**
6.1 Overview 13
6.2 Implementation Details 14

7 Future Directions **15**

8 Conclusion **16**

List of Figures

1 Private/Consortium Blockchain Network 9
2 Replicated state machine(Ongaro and Ousterhout, 2019) 10
3 Leader Election(Ongaro and Ousterhout, 2019) 11
4 Implementation Overview 14

List of Tables

1 BlockChain Types 4
2 Blockchain Consensus Algorithms (NF indicates not found) 8
3 Replication Arguments 12
4 Application functionalities 14

1 Introduction

Blockchain is a list of blocks that store committed transactions. A well-functioned and practical blockchain network is an open, immutable, decentralized ledger that can record transactions, makes itself replicated and synchronized across multiple nodes.(Ismail and Materwala, 2019) Current blockchain system could be categorized into three types: public blockchain, private blochain and consortium blockchain, based on how much the participants trust each other. (Zheng et al., 2017) Now the rise of private/consortium blockcahin applications spans across diverse range far beyond crytocurrencies, including insurance(Hess et al., 2017), medicine(Yue et al., 2016), economics(Huckle et al., 2016), Internet of things (Sun et al., 2016), supply chain(Saberi et al., 2019), software engineering (Stenum et al., 2015), etc.

Because the essence of blockchain is distributed system, *consensus* algorithm plays a critical role in blockchain. Consensus problem could be described as the coordination among nodes, e.g. make agreement on a single value with given a set of proposals.(Camargos et al., 2007) The core element of blockchain network is reaching consensus of information sharing, replicating state, and broadcasting transactions, among participants. (Viriyasitavat and Hoonsopon, 2019) Several consensus algorithms have been proposed, e.g., proof-of-work (PoW) (Nakamoto, 2019), proof-of-stake (PoS)(King and Nadal, 2012) practical Byzantine fault tolerant (PBFT)(Castro and Liskov, 1999), Paxos(Lamport, 1998), and Raft.(Ongaro and Ousterhout, 2019)

All of these algorithms have strengths and constraints. With comparative analysis in §3 , we proposed a Raft-based private/consortium blockchain system and implemented it on Java.

The rest of this paper is organized as follows: First, background knowledge of Blockchain and Consensus are introduced (Section §2). Then, a survey of Consensus algorithms in blockchain with critical analysis and related work of Raft is discussed (Section §3). This is followed by Application details including usage of Raft and functionalities (Section §4). In Section §5, we present the definition of Raft and discuss the reason of choosing it. Then we introduce the implementation in Section §6. Finally, we propose future directions of our application and make a conclusion in Section §7 and Section §8.

2 Background

2.1 Blockchain

Blockchain system utilises encryption technology stores data in the form of blocks and manages them as a connected chain in Peer-to-Peer network.(Kim et al., 2021) There are mainly three types of blockchain: *Public, Private and Consortium.* We provide a comparison of these types in Table 1.

The public blockchain system is a fully decentralized system where anyone could join and all records are visible to everyone, while consotium/private blockchain system only allow

Type	Configured for	Managed by	Centralised	Efficiency	Example
Public	Everyone	Anyone	No	Low	Bitcoin, Ethereum
Private	Organization	Organization	Yes	High	blockchain for accounting departments within a company
Consortium /Federated	Institution that have a partnership	Selected authority node	Partial	High	Hyperledger

Table 1: BlockChain Types

whitelisted participants to join. Since the consensus process of public chain is permissionless, i.e., everyone in the system could take part in the consensus process, while the consensus process of consortium/private blockchain is permissioned, i.e., only those pre-selected nodes would participate in the consensus process, people often treating private and consortium blockchain as a whole when comparing to the public blockchain. In addition, the key difference between consortium blockchain system and private blockchain system is that, in consortium blockchain system, participants in consensus process may be from several organizations while in private blockchain system, participants are from one organiazation.

We'll further discuss the details of blockchain and our application in §4.

2.2 Blockchain Challenges

The challenges or performance issues faced by Blockchain Technologies can be categorized as (Bamakan et al., 2020):

i ***Security:*** There is a probability of a 51% attack.

A single entity or organization is able to control 51% of the network power would potentially causing a network disruption, the blockchain may be subtaged by the attack in this scenario.

ii ***Speed:*** There is a significant delay in transaction execution.

iii ***Energy:*** Amounts of energy are wasted during the mining process in Blockchain.

2.3 Consensus

- **The Consensus Problem**

  Consensus problem could be described as a group of nodes agree on a single value, given a set of proposals.(Asaro Camargos et al.,) It is used to implement a fault-tolerant distributed system, for example, managing replicated state machines to handle faulty processes and deceptive nodes.

  There are many types of consensus problem including the *Byzantine generals problem* (Bach et al., 2018), which demonstrates that some nodes become malicious and break the properties of a consensus protocol deliberately, and *crash failure*, which demonstrates that some nodes stop to process temporarily or permanently.

- **The Consensus Algorithms**

The studies on consensus algorithm in distributed systems propose the tolerance of the partition of network, node failures , message delays and messages being out-of-order, lost or tampered. However, in the context of blockchain, consensus algorithm also needs to concern about malicious nodes and make sure all nodes agree on a consistent global state.

Blockchain consensus algorithms tries to address three key properties based upon which its applicability and efficacy can be determined(Viriyasitavat and Hoonsopon, 2019):

i ***Liveness*** which guarantees that something good will happen eventually.

ii ***Safety*** which property guarantees that something bad will never happen.

iii ***Fault-tolerant*** which means that a system could operate correctly from failures of some nodes at any given time.

3 Related Work

In this Section, we introduced several popular consensus algorithms on the Blockchain technology, a critical analysis is then conducted in terms of use cases, performance and incentives. Finally, we discuss the related work of Raft.

3.1 Consensus Algorithms

To make the Blockchain more trustworthy by meeting the challenges addressed in §2.2, consensus algorithms must be designed to serve as a resolution process. These algorithms are of two types: proof-based and voting based.(Pahlajani et al., 2019)

3.1.1 Proof-based Consensus Algorithm

In Proof-based Consensus Algorithms, a cryptographic problem need to be figured out for nodes trying to get the right of appending the block to join.

i ***Proof of Work(PoW)***

PoW is the first version of proof-based consensus strategy used in the Bitcoin Network proposed by (Nakamoto, 2019). If participants joining in the bitcoin network want to append a new block in the blockchain, they need to solve a mathematical puzzle, i.e., prove their work. And theses participants are call *miners*. Specifically, they need to tune a parameter called *nounce* of the latest block header on the blockchain to get different hash value until the calculated value is less than or equal a given value. The PoW procedure is called *mining* in Bitcoin.

Although very expensive, the *Security* issue may occur when the attacker controls 50% of the computing power of the Blockchain network. The *Speed* and *Energy* are still challenges as it consuming a large amount of electricity and time to compute, and becomes severe when the length of blockchain increase.

ii ***Proof of Stake(PoS)***

PoS is proposed for Peercoin cryptpcurrency(King and Nadal, 2012) The algorithm is based on the idea that the various combination of random selection is used to choose

the creator of next block.(Bamakan et al., 2020) It is believed that participants with more digital currencies in blockchain network are less likely to attack the network but to maintain the stability of network, the node must own the required amount of currencies for a specific while to participant.(Bamakan et al., 2020) PoS is more energy-saving than PoW. Additionally, PoS is immune to 51% attack as false verification is penalized and attacker needs to hold enough currencies before attacking.

3.1.2 Voting-based Consensus Algorithm

It is more preferable in private/consortium blockchain where nodes are known.(Pahlajani et al., 2019) In voting-based consensus algorithm, if a participant wants to append a block on blockchain, at least n[1] participants should agree on it. Before appending, state exchange among participants is required. If there are f failed nodes, then $f+1$ nodes should function well for a decision.

As nodes need communication to decide, a well-known classification of Voting-based Consensus Algorithms is classifying them into two categories: 1) Byzantine tolerance based, where nodes are crashed and unsettle and 2) Crash tolerance based, where nodes are crashed.

1. ***Byzantine tolerance-based***

   i ***Practical Byzantine Fault Tolerance (PBFT)***

   PBFT is designed to solve byzantine general problem. A new block is genetared at each round, a round is consisted of three phases *pre-prepare*, *prepared*, and *commit*. At the beginning of a round, a leader is elected. Then it decides the orders of transactions and broadcasts transactions to each nodes. In each phase, a node executes the transactions and broadcast its result to its peers. The node could enter next phase if it receive more then 2/3 agree votes from its peers. Therefore, the PBFT could tolerate up to 1/3 malicious nodes. Energy efficiency and high throughput are main advantages of PBFT, though transaction requests are processed in a high speed, few or no parameters available for being scalable and possible delays are its disadvantages.

   ii ***Delegated Byzantine Fault Tolerance(dBFT)***

   The algorithm follows the rules of the PBFT, but all nodes of voting is not required to append a new block. (Salimitari and Chatterjee, 2018) Some professional nodes are voted to record transactiona for all in dBFT. Compared to PBFT, dBFT is less likely to face delays, but the incomplete voters can threaten the decentralization of the network.

2. ***Crash tolerance-based***

   i ***Raft***

[1] (n is a defined threshold)

The most important and most prominent algorithms were *Paxos* (Lamport, 1998) and *Viewstamped Replication* (Liskov and Cowling, 2012). *Raft*, proposed in 2014 (Ongaro and Ousterhout, 2019) is a recent prominent members in this family. All protocols in this family progress in a sequence of views or epochs, with single-leader-multiple-followers pattern. Once the leader fails, or some followers suspect that the leader is down, they could start an election and replace the current leader, then move to next view.(Cachin and Vukolić, 2017) This family works correctly as well as more than a half of the network node functions normally. Raft cannot tolerate Byzantine failure, but achieved up to 50% of crash fault tolerance.

ii ***Chain with Federal consensus***

In this protocol, blocks are accepted only when they have been signed by specific quorum of blocked signers.(Pahlajani et al., 2019) It reduced the complexity to reach the consensus since each node has a set of trusted peers for block signing, consensus is reached only when m out of n block signers.

3.2 Critical Analysis

To conclude, we described two types consensus algorithms in blockchain technology: 1)Proof-based algorithms including PoW and PoS, and 2)Vote-based algorithms including PBFT, dBFT(Byzantine tolerance-based) and Raft, Federal consensus(Crash tolerance-based).

All of these algorithms have their pros and cons, As shown in Table 2, we compared these consensus algorithms in terms of:

i ***Suitable use cases (Blockchain types)***

As discussed in §3.1.2, voting-based algorithms are more preferable in private/consortium blockchain as nodes are known in this scenario, while proof-based algorithms are mainly for public blockchain.

ii ***Performance (Challenges)***

In §2.2 we drawed three challenges or performance issues blockchain may face: Energy, Speed and Security(51% attack). Although PoW and PoS ensures security to some extend, i.e. very expensive for attackers to control 50% of the computing power to sabotage blockchain and supports Scalability, these proof-based algorithms suffer from energy consuming issue. Voted-based algorithms such as PBFT and Raft on the other hand, are more efficient comparing to PoW and PoS, but doesn't ensure Security.(Huang et al., 2020)

iii ***Incentive***

Incentive is the reward for mining in PoW and PoS. Blockchain need to offer reward to those miners encouraging them to participate in the mining processes. However in private/consortium blockchain, the mining operations depend on enterprise resources.(Alsunaidi and Alhaidari, 2019) Incentive is not necessary.

Algorithms	Proof-based		Vote-based			
			Byzantine tolerance		Crash tolerance	
	PoW	PoS	PBFT	dBFT	Raft	Federal consensus
Use case(type)	Public		Private/consortium			
Energy saving	No	Partial	Yes			
Block creation speed	Low	High			NF	
51% attack	Yes	No			NF	
Scalability	Strong		Weak			
Incentive	Yes	No	No			
Application	Bitcoin	Peercoin	Hyperledger	Antshares	smart contract	NF

Table 2: Blockchain Consensus Algorithms (NF indicates not found)

Therefore, we need to consider the tradeoffs when selecting consensus algorithm for blockchain. Both satisfying our needs and achieving better performance should be taken into consideration. The reason why we choosed to embed Raft algorithm into our private/consortium blockchain application will be further clarified in §5.3.

3.3 Raft Related Work

The author of Raft(Ongaro and Ousterhout, 2019) states that Raft is more understandable to help to build practical systems than Paxos. There are also some publications that proposed Raft shows better performance than other algorithms in some applications, for example, (Fazlali et al., 2019) indicates Raft is an efficient substitute for Paxos in Cassandra. Some extension of Raft are also proposed recently: (Christopher Copeland, 2016) introduced Byzantine tolerant version of Raft that can tolerant Byzantine faults, (Zhang et al., 2017) proposed a Network-Assisted Raft Consensus Algorithm that reduces consensus latency.

4 Application

In this section, the details of our private/consortium blockchain application will be illustrated. We'll then discuss how the application works, where algorithm plays a role and the main functionalities or goals of our Raft-based Blockchain application.

4.1 Private/Consortium Blockchain

As described in §2, Blockchain is a incorruptible distributed ledger of transactions that can be used to record not just financial transactions, but virtually everything of value. Figure 1 present a simple structure of BlockChain network. It contains "Block" which records a number of transactions, and "chain" since the has of each transaction is generated to include information from the current and past transactions. The chain effect makes transactions immutable once they have been added. The body of block contains a checklist of transactions which includes transaction details such as amounts, addresses of the parties involved. The Header connects the current block to the previous block and carries the previous block hash, this creates a chain effect so that transactions are immutable once they've been added. The ledger has copies in all the peers. In private/consortium blockchain, only verified nodes can join.

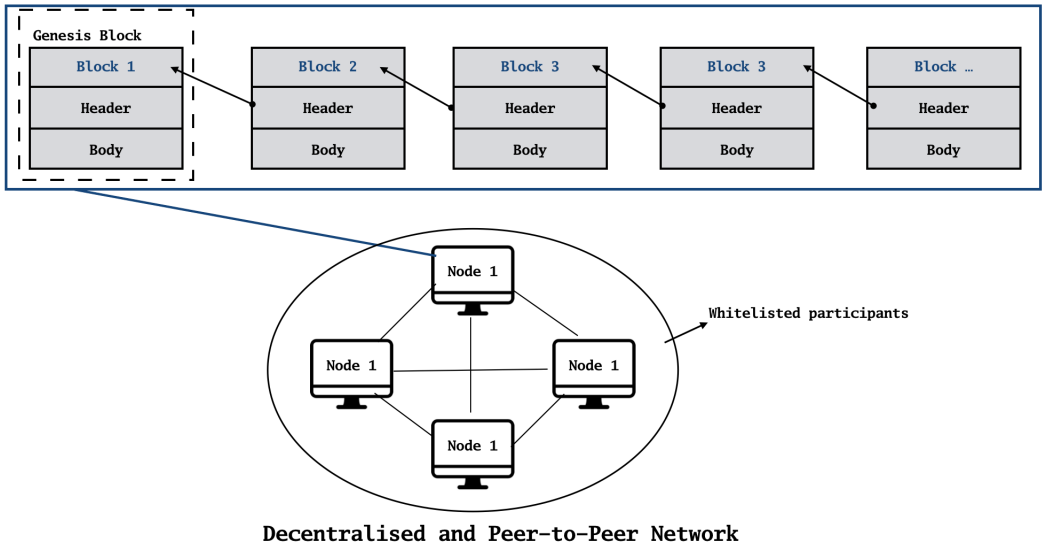

Figure 1: Private/Consortium Blockchain Network

4.2 Application Details

- *How Blockchain works?*:

  The *Genesis Block* contains the initial transaction, the following transactions must be verified before appended to the end, where is realized by two steps:(Alsunaidi and Alhaidari, 2019)

  i Some other nodes in the same chain conduct verification.

  ii Digital signature of the sender

  The verified transaction will be accepted to the block and added to the end of the chain. The block requires a replication process that broadcast to all the peers in the network.

- *Where Distributed Algorithms play a role?*:

  Performing transaction with the replication process may cause some confusion that lead the system inconsistent, as each node tries to broadcast its block updates. The *distributed algorithms* plays a role to avoid this situation: it brings agreements among nodes to determine which block will append and via which node.

- *Functionalities*:

  Two main functionalities is introduced in our Raft-based Private/Consortium Blockchain in a consistent manner:

  i Search the latest value of a certain key

  ii Append transactions by adding new blocks

  The concrete examples of functionalities will be given in §6.1.

5 Raft Algorithm

In this section, we provide a formal description of Raft Algorithm, we start with key concept description, then describe the details of working mechanism. Finally, we discuss the reason of picking Raft.

5.1 Key Concepts

Raft is a partition tolerant consensus algorithm proposed in 2014 by Ongaro et al (Ongaro and Ousterhout, 2019). Compared with the Paxos algorithm, Raft is more implementable and understandable.

- ***State Machine***

  The distributed system will get a consistent state if the state of node is consistent and each node executes same command sequence.(Bolosky et al., 2011) Therefore, the job of Raft is to ensure consistency of log replication. As illustrated in Figure 2, one node received commands from client, then write the command to the log and copy to other nodes. When the logs copied correctly, the State Machine of each node will execute the commands in same sequence to get a consistent state.

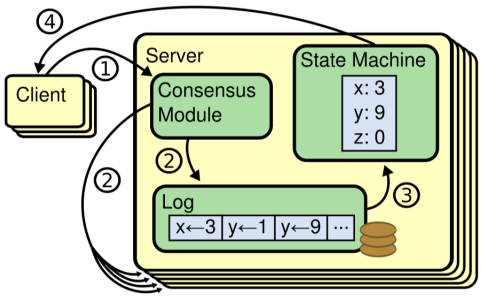

Figure 2: Replicated state machine(Ongaro and Ousterhout, 2019)

- ***Roles***

  In Raft, there are three possible roles for each node: *Follower*, *Candidate* or *Leader*. The leader is fully responsible for managing the replicated log to achieve consensus. Followers will become candidate if it has not received the heartbeat request sent by the leader after *Election Timeout*[2].

- ***Heartbeat***

  It is a timed task sent by the leader to the follower at regular intervals. The purpose is to ensure its own authority and check the term in the cluster to prevent multiple leaders.

- ***Term***

  Time is divided into Term in Raft, in each term, the first task that occurs is leader election. When a candidate wins a election, the leader will complete some operations in this term

[2]random time to reduce collision probability

and then end the term. If leader not be selected, then another term will be started with an election started immediately.

5.2 Working Mechanism

The entire protocol can be clearly divided into leader election and log replication tasks.

5.2.1 Leader Election

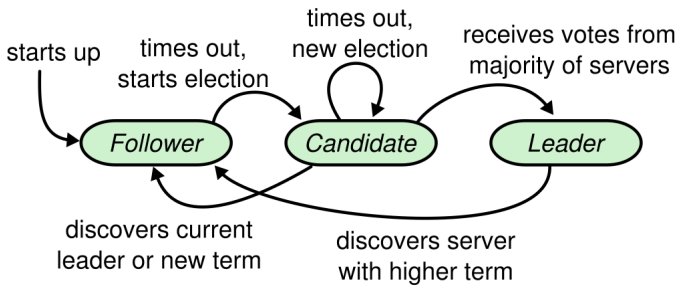

Figure 3: Leader Election(Ongaro and Ousterhout, 2019)

Raft uses *heartbeat mechanism* to trigger leader election. When the nodes start, they start as Followers. The leader regularly sends heartbeat message to all followers to maintain its authority. If the follower does not receive any message in a period of time, it will assume no leader and become a Candidate, then start election with increased term. It votes itself and issues RequestVoteRPCs to all.

After becoming a candidate, there are three possible situations may occurs:

i *Become a leader:* The candidate receives votes from majority of servers and wins the election. Then, it sends a heartbeat message to all other nodes to establish its authority and block new elections.

ii *Become a follower:* After the candidate sends RequestVoteRPC to other nodes, the candidate will have a waiting period for voting. During this time, it may also receive *heartbeat* message from another node claiming to be the leader. If the leader's term (included in its RPC) is larger or equal to the candidate's current term, the candidate recognizes the node to be a legal leader and returns to the follower state. Otherwise it will reject and remain to be a candidate.

iii *Maintain the candidate status:* If many followers become candidates at the same time, since each follower can only vote for one candidate, there will be a situation where no one candidate gets more than half of the votes. When this happens, each candidate will time out and start a new election by increasing its term and sending another round of RequestVoteRPC.

5.2.2 Log Replication

Log replication is the key of achieving raft consistency. The elected leader will received client requests, it appends command to local log and replicates to all nodes. The leader will write

Arguments	Explanation
lastApplied	index of highest log entry applied to state machine (initialized to 0, increases monotonically)
commitIndex	index of highest log entry known to be committed (initialized to 0, increases monotonically)
nextIndex	for each server, index of the next log entry to send to that server (initialized to leader last log index+1)

Table 3: Replication Arguments

to its own state machine(commit) after replicated it on a majority of nodes. Once follower learns that a log entry is commited, it applies the entry to local state machine in order.

However, the client requests may not correctly executed in real world because of follower reconnecting, data missing and other reasons. We'll then describe how the raft algorithm ensures cluster consistency, the arguments needed in shown is Table 3.

The task done by leader can be categorized as:

- Encapsulates the user's data into a log structure, including term, index and command, then pre-submits it to the local.
- Send data to other nodes in parallel, which known as log replication.
- Within specified time, if majority of the nodes return success, the log is committed by leader and return success to client.
- When the follower informs that the index of the current transmitted log is too large, reduce the index of the transmitted log and copy more logs to the other party. That is to help followers fill in missing logs. This step is also the key to ensuring consistency.
- Update leader's commitIndex, lastApplied and other information.

The task done by follower can be categorized as:

- Check the term received from the entry. If the term in entry cannot match local term, an error will be returned.
- If the log index does not match local commitIndex, return message to the leader and tell it to decrease nextIndex.
- If the locally existing log conflicts with the leader's log, the leader's log will prevail and its own will be deleted.
- Apply the log to the state machine, update the local commitIndex and return leader success.

5.3 Discussion

To conclude, we list three properties of Raft Algorithm:

i ***Crash Fault tolerant:*** It guarantees the consistency of a cluster when (N+1)/2 (rounded up) nodes are working normally in a cluster composed of N nodes. For example, in a five-node cluster, non-Byzantine errors are allowed for two nodes, such as node downtime, network differentiation, and message delay.

ii ***Safety:*** Raft's safety guarantees: Correctness and availability of the system remains guaranteed as long as a majority of the servers remain up, this realized by adding restrictions during the Leader election phase.

iii ***Liveness:*** Raft, as described in the original paper, does not guarantee liveness in face of network failures.(Viriyasitavat and Hoonsopon, 2019) However, Raft with PreVote and CheckQuorum can guarantee liveness.

The ***Safety*** property of Raft is one of the reason we choosing it. Additionally, as private/consortium only allow verified members to join the network, there is no need to emphasize security, it values crash faults more than Byzantine faults. Du et al. (2017), the ***Crash Fault tolerant*** property of Raft satisfies our needs.

In terms of Performance, as discussed in §3, Raft is an efficient consensus algorithm compared to PoW, PoS and PBFT (Huang et al., 2020). It has some efficient settings to avoid disruptions from isolated nodes, eg. those followers who receive heartbeat within the minimum election timeout cannot vote for candidates, this setting extends the leader working time.

Compare to PoW and PoS that are widely used in public blockchain network to ensure Security, the private/consortium blockchain can benefit from the high efficiency and simplicity of Raft. Additionally, each transaction is processed in consensus of all nodes in order to hold data consistency guarantees. In this scenario, Raft could perfectly support the functionality of private/consortium blockchain network, because of the replication of ledger it has and strong consistency it provides.(Huang et al., 2020)

6 Implementation

In this section, we first gives an overview of our system and where Raft plays a role, then detailed implementation is introduced with the explanation of technologies (RPC, Redis etc.)

6.1 Overview

The structure of our system in shown in Figure 4. As we mentioned in §4.2, the two main functionalities is realized by PUT and GET request sent by clients (shown in Table 4), clients can also view the current blockchain. All of these functionalities are achieved in consistent manner. i.e. Same returns for requesting different nodes.

Simplified working process can be summarised as: Leader received request, append to its local log module in Redis and replicate to all peers. Followers receive entry and stores in their log module, leader commit entry by adding it to state machine in radis when majority of followers logged. Finally, all peers writes to their state machine. Communication is conducted by RPC framework.

As mentioned in §4.2, the Raft algorithm plays a role in the PUT operation. The transactions put by clients are encapsulated in a new block, appending the new block to the end of the current blockchain need the help of Raft, ensuring all the node contains same version

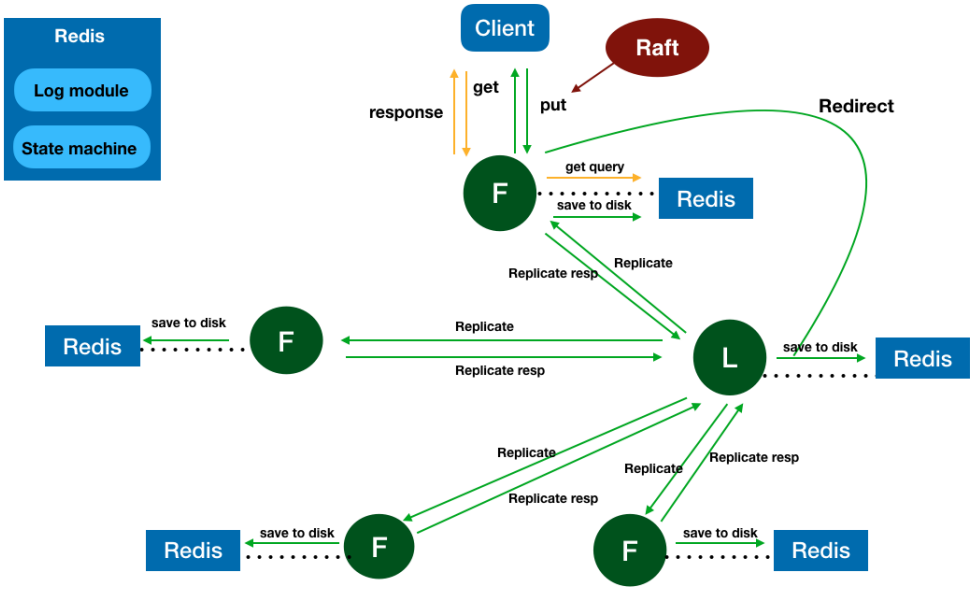

Figure 4: Implementation Overview

operation	functionality	example
GET	Get the value of certain keys	get("bob"), clients can get the latest value of "bob"
PUT	Put new transactions by adding a new block	put({"bob": 4, "alice":6}), a new block contains the transactions created and add to blockchain
View blockchain	View current blockchain in the system	(the current blockchain is same among all nodes)

Table 4: Application functionalities

of blockchain. Moreover, if leader crashes, the cluster will elect a leader immediately to maintain its high availability. For $N \geq 5$ cluster, it can tolerate $N/2$ nodes failure.

6.2 Implementation Details

6.2.1 RPC

Ongaro and Ousterhout (2019) used remote procedure calls (RPCs) to communicate between each node. RPC is a request–response protocol devised for the distributed computing.

Remote method invocation (RMI) extends RPC into the world of distributed objects. It enables us to pass parameters not only by value (input or output) but also by object reference. But implementing RMI require us to handling complex interfaces. In our application, we select SOFARPC framework as our RPCs module to speed up the development and reduce workload. SOFARPC is an open source high-performance, high-extensibility, production-level Java RPC framework [3]. We encapsulated the RpcServer and RpcClient object from SOFARPC, define our own send request method for client and handle request method for server respectively. Each node has two RPC components: 1)RPC client to send requests and 2) RPC server to receive and handle requests.

[3]https://github.com/sofastack/sofa-rpc

6.2.2 Redis

Redis[4] is a high-performance, key-value, in-memory data structure database. We can benefit from the small and sane Redis java client-Jedis[5], and the nature of single thread features of Redis can help us to deal with concurrent problem.

Each node in our system is connecting to its Redis server storing two types of data:

- ***Log Module:*** Entries are put into a Redis list via *lpush* command with the uuid as the key, logs are stored as sequences.
- ***State Machine:*** There are two types of key-value pairs stored in state machine:

  i address - blockchain pair

  As the key remains same, everytime a new block added, the blockchain value is updated. We can get the current blockchain from any node in a consistent manner.

  ii key - transaction value pair

  When new transaction coming in to the system, the key would be updated if existed, or appended if new. So that we can obtain the newest version value of certain key.

To read the key-value from persistent storage file *dump.rdb*, we use a parser from third-party library[6] to parse the RDB file.

6.2.3 Task Implementation

As discussed in the Leader Election task (§5.2.1), the leader has to broadcast messages to all the other nodes in parallel. As a result, asynchronous method should be applied in sending messages. Java native mthod `CompletableFuture` is used to implement asynchronous operations, so that leader can waits for response after sending the message concurrently.

As Ongaro and Ousterhout (2019) recommended to set election timeout to 150-300ms, we used `150 + random number in (0-200)` as the election timeout. The heartbeat is sent every 125 ms from leader to followers.

As for Replication task (§5.2.2), we use `ScheduledExecutorService` to create scheduled tasks and `ExecutorService` to execute replication tasks, both of which are provided by Java concurrent package[7]. When leader receives a client request, log replication is executed by Thread Pool, the leader then wait for majority success response for commmit.

7 Future Directions

i **Algorithm Extension**

Christopher Copeland (2016) proposed a Byzantine tolerant variant of Raft algorithm BFTRAaft, which combines the advantage of PBFT and Raft, it allows Raft to maintain

[4]https://redis.io/
[5]https://github.com/redis/jedis
[6]https://github.com/jwhitbeck/java-rdb-parser
[7]https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/package-summary.html

Safty, Liveness and Fault Tolerance in the presence of Byzantine faults. This algorithm is worth implementing in future work.

ii **Implementation Extension**

In the development stage of our application,we (developers) know which nodes are alive. We manually connect to alive nodes for clients and try get or put data to show the functionalities of our blockchain network in testing. Connecting to failed node will throw an *RPC RemotingException*. However, in the real world, there is no way for real clients to know which node is down and which node they should not connect to. The group of nodes in blockchain network should be transparent to clients. Therefore, to develop the system further, we should introduce reverse proxy, health check and load balance functions into our system. Both Nginx[8] and Redis Sentinel[9] are good tools to provide these functions.

We decide to store the blockchain and state into Redis due to its light-weight property. However, if the amount of data becomes too large, we should consider to use some disk-based database to store the data, e.g., MongoDB[10].

8 Conclusion

Consensus is the basis of blockchains. In this paper, we conduct a survey of popular consensus algorithms and give a comparative analysis to enlighten the strengths and constraints of them. After describing the detail of our blockchain application and formly defining Raft algorithm, we discuss the reason of implementing Raft in our private/consortium blockchain. The details of implementation is given. Finally, a future direction is proposed in the perspective of algorithm and implementation extension.

References

Shikah J. Alsunaidi and Fahd A. Alhaidari. 2019. A survey of consensus algorithms for blockchain technology. *2019 International Conference on Computer and Information Sciences, ICCIS 2019*.

L ´ Asaro Camargos, Lasaro@unicamp Br, Rodrigo Schmidt, and Fernando Pedone. *Brief Announcement: Multicoordinated Paxos **.

L. M. Bach, B. Mihaljevic, and M. Zagar. 2018. Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, pages 1545–1550.

[8]https://www.nginx.com/
[9]https://redis.io/topics/sentinel
[10]https://www.mongodb.com/

Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. 2020. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, 154.

William J. Bolosky, Dexter Bradshaw, Randolph B. Haagens, Norbert P. Kusters, and Peng Li. 2011. Paxos replicated state machines as the basis of a high-performance data store. *Proceedings of NSDI 2011: 8th USENIX Symposium on Networked Systems Design and Implementation*, (January 2011):141–154.

Christian Cachin and Marko Vukolić. 2017. Blockchain consensus protocols in thewild. *arXiv*.

Lásaro Camargos, Rodrigo Schmidt, and Fernando Pedone. 2007. Brief announcement: Multicoordinated Paxos. *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, pages 316–317.

Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. Technical report.

Hongxia Zhong Christopher Copeland. 2016. A Byzantine Fault Tolerant Raft.

Mingxiao Du, Xiaofeng Ma, Zhe Zhang, Xiangwei Wang, and Qijun Chen. 2017. A review on consensus algorithm of blockchain. *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, 2017-Janua:2567–2572.

Mahmood Fazlali, Sara Moazezi-Eftekhar, Mohammad Mahdi Dehshibi, Hadi Tabatabaee Malazi, and Masoud Nosrati. 2019. Raft Consensus Algorithm: an Effective Substitute for Paxos in High Throughput P2P-based Systems. *arXiv*, pages 1–19.

Zackary Hess, Yanislav Malahov, and Jack Pettersson. 2017. AEternity blockchain The trustless, decentralized and purely functional oracle machine. Technical report.

Dongyan Huang, Xiaoli Ma, and Shengli Zhang. 2020. Performance Analysis of the Raft Consensus Algorithm for Private Blockchains. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):172–181.

Steve Huckle, Rituparna Bhattacharya, Martin White, and Natalia Beloff. 2016. Internet of Things, Blockchain and Shared Economy Applications. In *Procedia Computer Science*, volume 58, pages 461–466. Elsevier B.V., 1.

Leila Ismail and Huned Materwala. 2019. A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions. *Symmetry*, 11(10).

Donghee Kim, Inshil Doh, and Kijoon Chae. 2021. Improved Raft Algorithm exploiting Federated Learning for Private Blockchain performance enhancement. *International Conference on Information Networking*, 2021-Janua:828–832.

Sunny King and Scott Nadal. 2012. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Technical report.

Leslie Lamport. 1998. The Part-Time Parliament. Technical report.

Barbara Liskov and James Cowling. 2012. Viewstamped replication revisited.

Satoshi Nakamoto. 2019. Bitcoin: A peer-to-peer electronic cash system. Technical report.

Diego Ongaro and John Ousterhout. 2019. In search of an understandable consensus al-

gorithm. *Proceedings of the 2014 USENIX Annual Technical Conference, USENIX ATC 2014*, pages 305–319.

Sunny Pahlajani, Avinash Kshirsagar, and Vinod Pachghare. 2019. Survey on Private Blockchain Consensus Algorithms. *Proceedings of 1st International Conference on Innovations in Information and Communication Technology, ICIICT 2019*, (July):1–6.

Sara Saberi, Mahtab Kouhizadeh, Joseph Sarkis, and Lejia Shen. 2019. Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, 57(7):2117–2135, 4.

Mehrdad Salimitari and Mainak Chatterjee. 2018. A survey on consensus protocols in blockchain for IoT networks. *arXiv*, pages 1–15.

Jacob Stenum, Czepluch Jstc@itu Dk, Nikolaj Zangenberg, Lollike Nlol@itu Dk, Simon Oliver, and Malone Soma@itu Dk. 2015. The Use of Block Chain Technology in Different Application Domains Bachelor Project in Software Development. Technical report.

Jianjun Sun, Jiaqi Yan, and Kem Z.K. Zhang. 2016. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financial Innovation*, 2(1):1–9, 12.

Wattana Viriyasitavat and Danupol Hoonsopon. 2019. Blockchain characteristics and consensus in modern business processes. *Journal of Industrial Information Integration*, 13(July 2018):32–39.

Xiao Yue, Huiju Wang, Dawei Jin, Mingqiang Li, and Wei Jiang. 2016. Healthcare Data Gateways: Found Healthcare Intelligence on Blockchain with Novel Privacy Risk Control. *Journal of Medical Systems*, 40(10), 10.

Yang Zhang, Bo Han, Zhi-Li Zhang, and Vijay Gopalakrishnan. 2017. Network-Assisted Raft Consensus Algorithm.

Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. 2017. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, pages 557–564.