BiniBFT v1 - January 15, 2024

BiniBFT

- A Secure, Scalable BFT for Hyperledger Fabric

Anasuya Threse Innocent .A#, Riddhi Ravindra Katarki*, Ashna P S*, Ajitesh Kumar Soni*, Abhishek Ranjan*, Sahilsher Singh*, Siddhant Prateek Mahanayak*
#Founder & Director, BiniWorld Innovations Private Limited, Manakkarai West, Villukuri Post, Kanyakumari District, India, www.biniworld.com
#Mentor, Linux Foundation Hyperledger Collaborative Learning Program, 2023, bft@biniworld.com
*Mentee, Linux Foundation Hyperledger Collaborative Learning Program - An Optimized BFT on Fabric, 2023, riddhikatarki@gmail.com, psashna1984@gmail.com, ajiteshsoni.work@gmail.com, abhishekranjan6585@gmail.com, sandhu.sahil2002@gmail.com, siddhantprateek@gmail.com

Abstract

Blockchain technology has a wide range of day-to-day applications ranging from electronic voting to supply chain management to financial transactions to healthcare management, energy management etc in a distributed and a decentralized environment. The enterprise blockchain provides transparency and immutability in a premissioned setting. The consensus protocols play a crucial role in establishing trustworthiness within these systems, facilitating secure and reliable operations across various sectors. The seamless integration of Byzantine Fault Tolerance (BFT) further fortifies the trustworthiness of enterprise blockchain. To achieve high security and scalability this paper presents the BiniBFT consensus for Hyperledger Fabric. This paper highlights the pivotal role of blockchain and BiniBFT in revolutionizing the reliability and security of day-to-day operations across industries, offering a glimpse into the transformative potential of decentralized technologies in shaping the future of sustainable digital ecosystems.

I. Introduction

Blockchain is a digital ledger system that stores transactions in a chronological order of linked blocks. One distinguishing feature of blockchain is its transparency, which allows all network participants to view transaction data while providing immutability of the data. Blockchain is a distributed network that relies on nodes, or users, to validate transactions and maintain ledger integrity. Decentralization removes a single point of failure from the system, improving overall system security and reliability. The redundancy created by replicating the ledger across all nodes reduces the risk of data loss. Nodes equipped with authorized hardware play an important role in the network's long-term viability.

Each node in the network has a unique identifier, making it easier to distinguish across nodes. These nodes, which are critical to preserving decentralization, work together to support the consensus mechanism, ensuring transaction legitimacy. Control in a decentralized blockchain network is not centralized in specific organizations or authorities.

Blockchain can broadly be classified into Permissioned or Private Blockchain, Public Blockchain, Hybrid Blockchain and Consortium Blockchain. The public chains are mainly used in the crypto world, ensuring the anonymity of the individuals involved. The private chains are useful within a closed organization, whereas for enterprise applications we need permissioned public chains which can be used for day-to-day applications.

The fundamental function of a blockchain node is to check the legitimacy of succeeding blocks of network transactions verified using a consensus mechanism that allows the nodes to agree on a single state. This validation process ensures that established blockchain network rules and regulations are followed, contributing to overall security and integrity. The nature of the consensus mechanism used varies depending on the type of blockchain and the requirements.

The next Section talks about the different Byzantine Fault Tolerance (BFT) consensus and provides the comparative study of them focusing on the enterprise blockchain, Hyperledger Fabric. In Section III the BiniBFT Consensus algorithm is proposed with its detailed design and Section IV concludes.

II. BFT Consensus

Byzantine Fault Tolerance (BFT) is like a superhero power for computer systems, especially when lots of computers need to work together. Imagine you and your friends trying to decide on what game to play. Everyone suggests something, and we want to make sure that even if one friend gives a silly or wrong idea (like playing in the middle of the street), we still make a good choice.

Now, think of these friends as computers in a big network, working together to solve important tasks instead of just picking a game. The challenge is that sometimes, a computer might act weird, maybe due to a problem or even because someone is trying to mess things up on purpose.

BFT is the solution to this challenge. It's like having special rules that ensure the computers can still make the right decision, even if some of them are giving wrong information or acting strangely. It's like having a clever plan to outsmart any friend who might want to ruin the game.

So, why do we need BFT? Well, when computers work together, we want to be sure they can trust each other. Imagine if your computer had to make important decisions, like sending money or running a crucial program. We wouldn't want one mischievous computer to mess everything up. BFT is like having a trusty team where even if a few members try to misbehave, the majority can still make sure everything goes smoothly. In simple terms, BFT is like having a group of friends who always figure out the best game to play, no matter if some friends are being silly. It helps computers in a network make reliable decisions, even when some of them are acting strange or being a bit naughty.

Features of BFT

Agreement on Shared State: In a distributed system, multiple participants need to agree on the state of the blockchain. A consensus mechanism ensures that all honest nodes in the network agree on a single source of truth, even if some nodes are faulty or malicious.

Security: Consensus mechanisms are designed to withstand various types of failures and attacks.

Fault Tolerance: Systems need to be able to handle faults or malicious behavior from a subset of participants. BFT mechanisms specifically are designed to handle a certain threshold of Byzantine faults.

Coordination and Synchronization: In any distributed system, nodes may need to operate concurrently and often asynchronously. Consensus mechanisms help in coordinating system actions and synchronizing state across all nodes.

Robustness: Nodes may join or leave the network, or they might become temporarily unavailable. A robust consensus mechanism can adapt to changes in the network's composition, maintaining the system's overall integrity and availability.

Existing BFT Mechanisms

This section provides a brief of available consensus mechanisms used in permissioned blockchains like Hyperledger Fabric.

pBFT

Practical Byzantine Fault Tolerance (pBFT) [1] stands as a prominent and influential consensus algorithm in distributed systems, owing to its effective Byzantine fault tolerance, operational efficiency, and relative simplicity. Among various Byzantine Fault Tolerance (BFT) algorithms, pBFT distinguishes itself with its practicality and efficiency. The algorithm employs a three-phase message exchange protocol, which includes the *Pre-prepare, Prepare*, and *Commit phases*.

In the *Pre-prepare phase*, the primary node disseminates a proposal to all other nodes in the network. Subsequently, in the *Prepare phase*, upon receiving the pre-prepare message, other nodes cast their votes on the proposal and transmit their prepared messages back to the primary node. Finally, in the *Commit phase*, if the primary node accumulates enough prepared messages, surpassing two-thirds of the total nodes, it broadcasts a commit message, thereby finalizing the proposal.

Despite its effectiveness, the standard pBFT algorithm has limits when applied to networks with many nodes, often more than 100. This constraint limits scalability, limiting its applicability in large networks. To address this problem, a multi-layer pBFT algorithm was developed to improve network scalability and reduce the obstacles posed by an increased number of consensus nodes. A multi-center pBFT algorithm, on the other hand, has been presented, with the goal of improving fault-tolerant capabilities to assure the system's smooth functioning even in increasingly demanding conditions.

Scalability of the pBFT method has been examined primarily through the lenses of communication complexity and scalability. Scalability appears to be a substantial obstacle, necessitating the development of a number of techniques to address it. Strategies such as network sharding have been implemented to increase the network's capability for expansion. However, it is critical to note that, while these approaches improve network scalability, they usually weaken system security by ignoring the consensus network's fault tolerance.

The pBFT algorithm, lauded for breaching the original Proof of Work (POW) algorithm's performance constraint, achieves improved throughput and reduced transaction confirmation wait. The algorithm, however, confronts problems such as high communication complexity, poor scalability, and limited fault tolerance. These difficulties have an impact on the performance of blockchain-related projects, making it impossible for them to meet the strict standards of real-world business scenarios.

While pBFT remains a strong consensus algorithm, its limits in scalability and fault tolerance have prompted the development of alternate and improved versions, such as multi-layer and multi-center pBFT algorithms. The ongoing investigation and development in this sector strives to achieve a balance between scalability and security, guaranteeing that distributed systems, particularly in blockchain applications, can fulfill the demands of real-world business requirements.

Zyzzyva

The research paper titled "Zyzzyva: Speculative Byzantine Fault Tolerance" [2] introduces a novel approach to achieving Byzantine fault tolerance in distributed systems. The primary goal of the protocol is to provide a high-throughput and low-latency solution for replicated state machines, even in the presence of Byzantine faults where some nodes may exhibit arbitrary and potentially malicious behavior.

Zyzzyva is a Byzantine Fault Tolerance (BFT) protocol designed for distributed systems. It ensures that a network can reach a consensus on the order of tasks, even when some nodes are malicious or faulty. The core of Zyzzyva's functionality lies in three sub-protocols: Agreement, View Change, and Checkpoint.

Here is the broad view of Zyzzyva's transaction flow:

- **Request Initiation:** A client initiates a request to a primary replica.
- **Primary's Leadership:** The primary, acting as a leader, coordinates the agreement process, assigning sequence numbers to maintain order.
- **Replicas Execute the Task:** All replicas, including the primary, independently execute the task.
- **Replicas Respond:** Each replica sends a response to the client.
- **Client's Verification:** The client collects responses to ensure agreement.
- **Ensuring Safety:** Zyzzyva guarantees task execution in the agreed order, even with faulty replicas.
- **Ensuring Liveness:** Tasks initiated by a correct client will eventually be completed.

The Zyzzyva, employs a two-phase approach, distinguishing between an optimized fast path and a slower, but more robust, fallback mechanism. The fast path is designed to achieve high throughput under normal operating conditions, leveraging speculative execution to process client requests with a single round of communication. The idea is to allow the client to quickly obtain a response from a majority of replicas, assuming that a consensus will be reached.

In the fast path, a client issues a request to all replicas, and upon receiving a sufficient number of matching speculative responses, it assembles a commit certificate. This certificate serves as cryptographic proof that a majority of correct replicas agree on the ordering of requests up to and including the client's request. The protocol then completes the second phase of agreement, ensuring that enough servers possess this proof.

The paper also addresses scenarios in which the network, primary, or some replicas are slow or faulty, potentially leading to mismatches in responses. In such cases, the two-phase mechanism applies when the client receives between $2f + 1$ and $3f$ matching responses, where f is the maximum number of Byzantine faulty replicas the system can tolerate.

To overcome challenges related to faulty replicas, Zyzzyva introduces a commit certificate that includes signatures from a sufficient number of replicas, providing a reliable mechanism for clients to trust the responses. The paper emphasizes the importance of stability in responses and outlines the steps taken by the client to ensure the consistency of the received responses.

The research evaluates the performance of Zyzzyva through experiments, comparing it with existing Byzantine fault-tolerant protocols such as pBFT and HQ. Throughput, latency, and performance during failures are considered key metrics. Zyzzyva demonstrates impressive throughput, exceeding 50,000 requests per second without batching and achieving competitive latency compared to other protocols.

In conclusion, "Zyzzyva: Speculative Byzantine Fault Tolerance" presents an innovative approach to Byzantine fault tolerance, combining a fast path for optimal performance under normal conditions with a robust fallback mechanism for fault scenarios. The protocol's design choices and optimizations contribute to its ability to achieve high throughput and low latency, making it a promising solution for replicated state machines in distributed systems. The research findings offer valuable insights into the practicality and efficiency of Zyzzyva in real-world scenarios.

Mir-BFT

Mir-BFT protocol [3], Mir-BFT protocol aims to improve network communication reliability, particularly in decentralized systems such as blockchain. Unlike prior protocols, Mir-BFT enables many leaders to suggest batches of requests simultaneously, eliminating malicious attacks that attempt to duplicate requests. Spreading the burden across multiple leaders minimizes the likelihood of one leader creating a bottleneck while increasing the network's capacity to process transactions.

One of its main advantages is how it handles client verification, which is sometimes a bottleneck in wide-area networks. Mir-BFT accelerates this process using a technique known as client signature verification sharding, which increases its efficiency even further.

Mir-BFT outperformed current protocols in tests, handling over 60,000 transactions per second, each of which is equal in size to Bitcoin transactions, across a huge network of 100 nodes connected by a 1 Gbps WAN. Processing these transactions typically takes only a few seconds, which is impressive for such a large network.

Mir is a protocol based on pBFT (Practical Byzantine Fault Tolerance), with a few major differences that improve its efficiency and security. Its key features are:

- **Authenticate client requests**: Unlike pBFT, Mir uses signatures instead of MACs to authenticate client requests. This update eliminates some sorts of attacks connected with MACs, however it may cause a CPU bottleneck owing to the additional burden in verifying client signatures.

- **Batching and Watermarks**: Mir uses batch processing, similar to pBFT, to boost throughput. It also keeps the request and batch watermarks used by pBFT to improve efficiency. Unlike some modern BFT protocols, Mir continues to employ watermarks to allow different leaders to submit batches at the same time.

- **Protocol Round Structure**: Mir uses epochs that correspond to views in pBFT. Each epoch has an epoch primary and an epoch leader, which allows several nodes to offer batches without conflict. Epoch transitions occur in response to suspected node failures or after a predetermined amount of batches are committed.

- **Epoch Leader Selection**: The primary node selects epoch leaders and broadcasts them to all nodes. Strategies for picking leaders can differ depending on characteristics such as execution history, fault patterns, and even blockchain stake. To prevent request duplication and censorship assaults, Mir divides the hash space into buckets and assigns each one to a specific leader. This avoids duplication assaults and guarantees that requests are not ignored by leaders.

- **Mitigating Request Duplication and Censorship Attacks**: Mir separates the request hash space into buckets and gives each bucket to a specific leader. This avoids duplication assaults and guarantees that requests are not ignored by leaders.

- **Request Partitioning**: Mir divides the request hash space into buckets, each with its own leader. Bucket redistribution occurs at regular intervals to prevent request censorship and ensure load balancing.

- **Parallelism**: The Mir implementation is highly parallelized, with each worker thread handling a single batch. It also uses several gRPC (Remote Procedure Cal) connections between nodes, which is critical for enhancing throughput, particularly in wide-area networks with a small number of nodes.

In essence, Mir-BFT preserves the safety elements of PBFT while making adjustments to improve its efficiency, particularly when dealing with several leaders proposing transactions at the same time.

BDLS

The BDLS consensus mechanism [4] is a Byzantine Fault Tolerance (BFT) protocol designed for blockchain systems, particularly focusing on maintaining safety and liveness in Type II partial synchronous networks.

Summary of BDLS

- **Participants & Rounds:** BDLS operates with $n = 3t + 1$ participants for each blockchain height, where each participant maintains a local variable containing candidate blocks known so far. The protocol iteratively runs rounds until consensus is reached on a block for the given height.

- **Decision Making:** During the rounds, participants send round-change messages to a leader, who then proposes a block to be locked or selected based on the majority's input. If a block gets enough commit messages, the leader decides on the value and broadcasts a decision. Participants move to the next height once a decision is received.

- **Synchronization & Timeouts:** The protocol ensures that all participants eventually synchronize on the same block height and provides mechanisms for timeout and round synchronization.

- **Efficiency in Partial Synchronous Networks:** Designed specifically for Type II partial synchronous networks, it assumes messages may get lost or re-ordered before a Global Stabilization Time (GST) but will be delivered reliably afterwards.

- **Linear Communication/Authenticator Complexity:** By using robust threshold signature schemes, BDLS aims to achieve linear complexity in terms of communication and authentication, making it more efficient than some existing protocols.

- **Locking Mechanism:** The protocol employs a unique locking and unlocking mechanism for candidate blocks, which helps prevent forks and ensures that once a block is committed, it is final.

- **Flexible Broadcasting:** BDLS incorporates both strong reliable broadcasting for decision dissemination and options for either mesh or star network broadcasting, depending on the application's needs.

- **Height and Round Synchronization:** It includes specific steps for height and round synchronization to ensure that all honest participants work on the same block height and round, reducing the likelihood of staying in outdated rounds or heights.

BDLS aims to improve the efficiency and reliability of blockchain consensus in partially synchronous environments by reducing communication complexity and ensuring robust decision-making through its unique locking mechanism and synchronization steps.

AOM

An innovative development in the field of Byzantine Fault Tolerant (BFT) protocols for data center networks is the Authenticated Ordered Multicast (AOM) [5]. The data center network infrastructure enables AOM to propose a transformational abstraction. It creates a system for transferable authentication in particular and provides assurances against non-equivocation.

BFT protocols had challenges prior to AOM ensuring a secure data center connection, including expensive performance, complicated latency, resource-intensive authentication, and scalability problems. The arrival of AOM caused a paradigm change. It transformed the scalability, dependability, and efficiency of the BFT protocol by providing transferable authentication and non-equivocation. Fault-tolerant networking in data centers has advanced significantly as a result of this crucial development, which removed earlier constraints

Features of AOM

Transferable Authentication: AOM promotes safe and dependable data transfer between network nodes by guaranteeing the preservation of data authenticity during network connection.

Non-equivocation Assurance: This primitive strengthens the general dependability and integrity of network communication by offering a defense against users retracting their earlier messages or activities**.**

Implementation: Hardware implementations that use public key cryptography and HMAC (Hash-based Message Authentication Code) authentication have shown that AOM is a feasible solution. These implementations have been shown on sophisticated programmable switches, demonstrating the AOM concept's viable and practical.

Impact on BFT Protocols: Matrix is a cutting-edge BFT protocol, and its design is based on AOM. Using the guarantees provided by AOM, Matrix maximizes authentication and cross-replica coordination in common circumstances. Evaluation results show that Matrix performs substantially better than current protocols in terms of throughput and latency, highlighting the major advantages of this novel network ordering abstraction for BFT systems in data centers.

The way BFT protocols function in data center networks has been revolutionized by AOM, or Authenticated Ordered Multicast. Prior to AOM, BFT had to overcome obstacles like excessive pricing, complicated authentication, delays caused by complications, and scalability difficulties. This environment had been modified by AOM, which provided secure data transfer and eliminated message denials. Utilizing cryptography for authentication, its useful hardware implementations demonstrated its feasibility. Based on AOM, Matrix is a BFT system that improves speed and efficiency over previous ones by streamlining authentication and coordination. With the introduction of AOM, data center fault-tolerant networking takes a major step forward, eliminating earlier constraints and greatly improving dependability [5].

BFT-SMaRt

BFT-SMaRt (Byzantine Fault Tolerant State Machine Replication) [6, 7, 8] is a robust protocol designed to ensure the integrity and fault tolerance of distributed systems, particularly in scenarios where nodes may behave maliciously or exhibit faults. One critical aspect of BFT-SMaRt lies in its handling of transactions within the consensus mechanism, the validation process, and the secure signing of blocks during the commit phase.

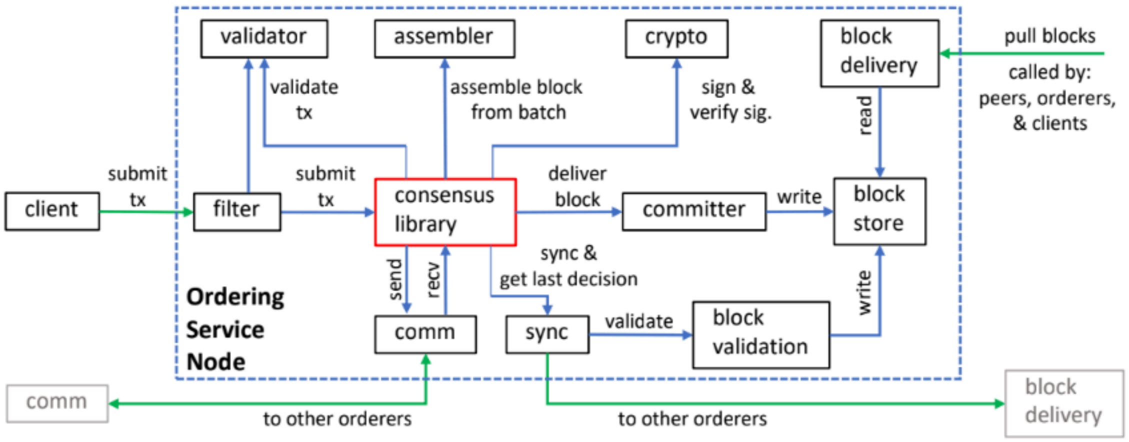

Fig. 1. The architecture of a Fabric BFT-based Ordering Service Node.

Fig. 1 shows the architecture of a Fabric BFT-based Ordering Service Node. The components are:

- Orderer node: Assembler assembles transactions into blocks
- Peer receives a Header - metadata stream from a randomly selected orderer
- Consensus consists of 3 phases: *Pre-prepare, Prepare* and *Commit*
- Client sends transactions to all the nodes in the network

In the context of block handling, BFT-SMaRt employs a systematic approach. Transactions, rather than being routed through an intermediary, are directly submitted to the BFT library. This strategy ensures that the BFT system actively monitors and prevents transaction censorship, enhancing the overall integrity and openness of the network.

Trust and validation mechanisms play a pivotal role in ensuring the correctness of transactions within the BFT-SMaRt protocol. Unlike conventional systems where followers might solely rely on the leader's decisions, BFT-SMaRt emphasizes an additional layer of security. Followers not only receive transaction proposals from the leader but also conduct transaction revalidation within the block proposal. This necessitated the introduction of specific APIs in the BFT-SMaRt library, enabling followers to verify transactions during the consensus process. This rigorous validation process significantly contributes to the protocol's resilience against potential faults or malicious behaviors by ensuring that transactions are correct and valid before finalization.

Block signing, a crucial step in ensuring the immutability and authenticity of the ledger, is handled meticulously within the BFT-SMaRt consensus protocol. Unlike traditional systems where block signing might be performed by a single node, BFT-SMaRt elevates security by distributing this responsibility across multiple nodes. Specifically, the block is signed during the commit phase, and not by a single node, but by a predefined subset of nodes denoted as Q nodes. This multi-node signing approach enhances the trustworthiness and tamper-resistance of the ledger, as a malicious actor would need to compromise multiple nodes simultaneously to tamper with the committed block.

The commitment to sign blocks during the commit phase and involve multiple nodes in the signing process reinforces the security and reliability of BFT-SMaRt. This distributed approach to block signing mitigates the risks associated with single-point vulnerabilities, offering a robust solution to ensure the authenticity and integrity of the ledger.

In summary, BFT-SMaRt's transaction handling, validation mechanisms, and block signing strategies collectively contribute to its robustness and resilience against Byzantine faults, offering a highly secure and reliable framework for distributed systems.

Comparison of Existing BFT Protocols

Protocol	Latency	Security	Throughput
pBFT	Higher latency	High	High
Zyzzyva	Low, optimized for high throughput	High	High, outperforming pBFT

Mir-BFT	Low latency, designed for efficient throughput	Robust to Byzantine faults	High, especially in WANs
BDLS	Not specified	Secure in Type II networks (partial synchronous network)	Not specified
AOM	Reduced in data centers	Enhanced with transferable authentication	Improved, demonstrated by Matrix protocol
BFT-SMaRt	Acceptable for most use cases	Compliant with Fabric standards, MSP integration	Low to Moderate

Fig. 2. Comparison of BFT Protocols

A comparative study of the candidates chosen were conducted considering the parameters *Latency, Security,* and *Throughput*, and is depicted in Fig 2. The study clearly shows that there is a need for a BFT consensus which is highly secure and scalable. To address this we present the BiniBFT Protocol in the next Section.

III. BiniBFT Framework

BiniBFT - A Consensus Mechanism with Random Polling

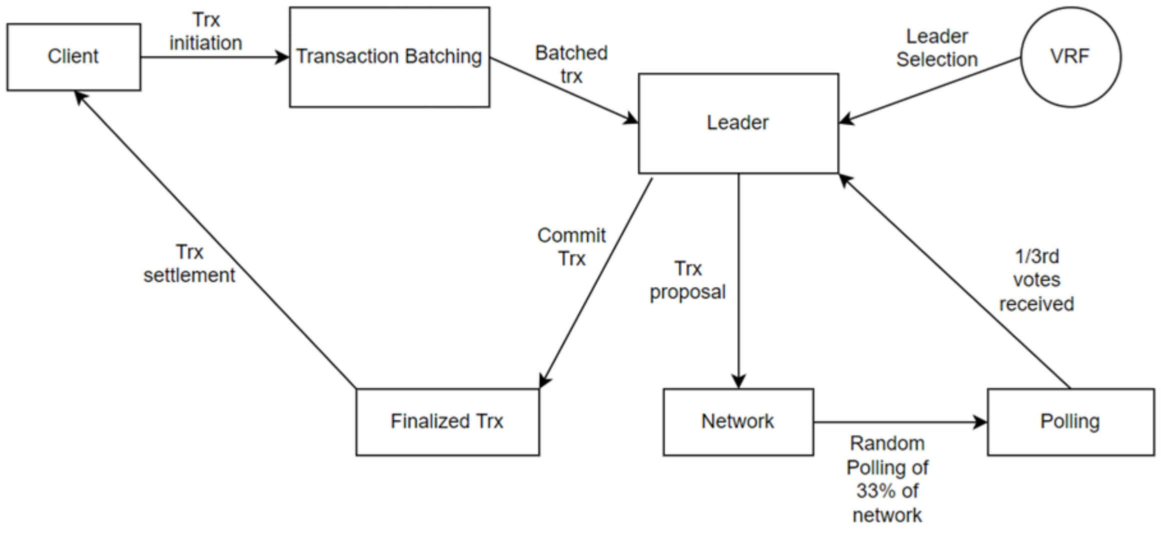

Fig. 3. BiniBFT Consensus

The BiniBFT consensus shown in Fig. 3 involves several steps, starting from transaction initiation by a client and transaction batching for efficiency. A leader is selected via a Verifiable Random Function (VRF) to propose these batched transactions. Simultaneously, a random polling of 33 % of the network is conducted for voting on the proposed transactions. Once sufficient votes are received, the transaction is committed and then finalized, officially recording it on the blockchain.

The transaction flow for the consensus mechanism is as follows:

1. **Transaction Initiation:** The process starts with the client initiating a transaction (Trx initiation). This is where a user or an application requests a transaction to be processed by the network.

2. **Transaction Batching:** After a transaction is initiated, it is grouped with other transactions into a batch (Batched trx) which improves the efficiency of processing multiple transactions together rather than individually.

3. **Leader Selection:** Once transactions are batched, a leader is selected to propose the batch of transactions to the rest of the network. The leader selection is aided by a Verifiable Random Function (VRF), which helps in choosing the leader in a way that is random but verifiable by other participants in the network.

4. **Transaction Proposal:** The leader then proposes the batched transactions (Trx proposal) to the network. This involves sending out the transaction data to other nodes in the network for validation.

5. **Polling:** Concurrently, there is a random polling of 33% of the network. This suggests that a subset of the network is selected randomly to vote on the proposed transactions. The chart indicates that the process waits until 1/3 of the votes are received.

6. **Transaction Commitment:** Once the leader has successfully proposed the transaction and the necessary votes are received, the transaction is committed (Commit Trx) by the network.

7. **Transaction Settlement:** After the transaction is committed, it is then settled (Finalized Trx), which means it is officially recorded on the blockchain. This completes the process, and the client is informed that the transaction has been finalized.

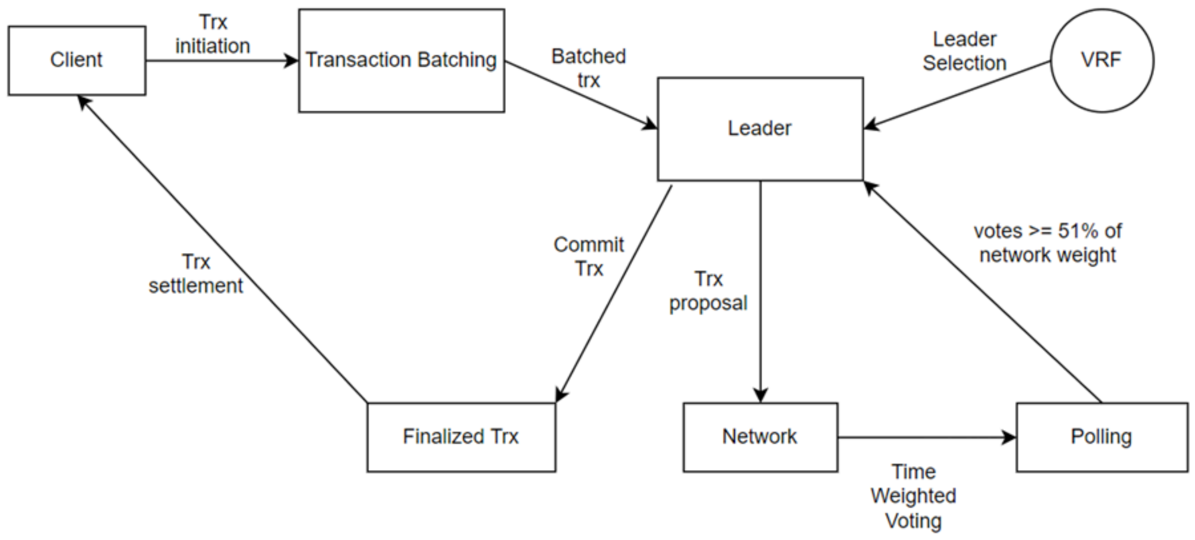

Fig. 4. Time Weighted BiniBFT Consensus

Fig. 4 shows the transaction flow for the Time Weighted BiniBFT consensus mechanism that uses Time Weighted Voting instead of Random Polling as shown in Fig. 3. The flow begins with transaction initiation and batching. The leader, selected through VRF, proposes the transactions to the network. The voting process here is characterized by Time Weighted Voting, where each node's weight increases over time, reflecting their tenure in the network. However, individual node weight is capped, and the collective weight of voting nodes must exceed 51 % of the total network weight for a decision. Successful proposals with enough weighted votes lead to transaction commitment and finalization.

The steps involved in the consensus process are as follows:

1. **Transaction Initiation:** A client starts the process by initiating a transaction.

2. **Transaction Batching:** Transactions are batched together for processing efficiency.

3. **Leader Selection:** A leader is selected through a Verifiable Random Function (VRF) to manage the batched transactions. This ensures randomness and fairness in leader selection.

4. **Transaction Proposal:** The selected leader proposes the batched transactions to the network.

5. **Time Weighted Voting (Polling):**

   a. The network nodes participate in polling to approve the transaction proposal. This polling uses the Time Weighted Voting mechanism.
   b. Each node in the network has a weight (w) that increases over time or epochs, representing their tenure in the network.

6. **Voting Constraints:**
   a. Individual node weight is capped to ensure no single node can dominate the decision due to its longevity (as shown in the second chart, where w(i) <= 0.33 * (network_w)).
   b. The sum of the weights (sum(w)) from the nodes participating in the polling must be greater than or equal to 51% of the total network weight to reach a decision (sum(w) >= (0.51 * network_w)).

7. **Transaction Commitment:** If the proposal receives enough weighted votes (>= 51% of network weight), the transaction is committed.

8. **Transaction Settlement:** The committed transaction is then finalized and recorded on the blockchain, and the client is notified of the transaction settlement.

The diagram (Fig. 5) illustrates a time-weighted voting mechanism across multiple epochs, which are discrete time periods in blockchain technology. For example, let us assume there are three actors (A, B, and C), each with a weight (w) that signifies their voting power in the network.

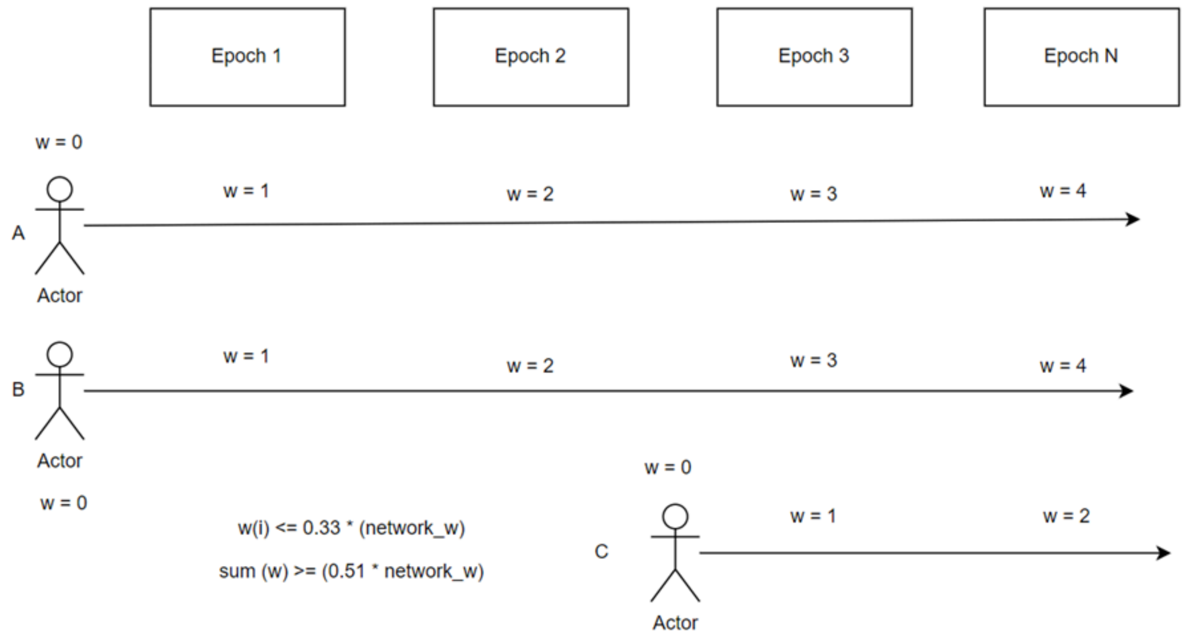

Fig. 5. Time-Weighted Voting

- **Actor A:** Starts with zero weight in Epoch 1 and gains one weight unit in each subsequent epoch, reaching a weight of 4 by Epoch N.

- **Actor B:** Begins with a weight of 1 in Epoch 1 and similarly gains weight over time.

- **Actor C:** Shows later entry into the system, starting at Epoch 3 with zero weight and increasing thereafter.

The weights are constrained by two rules:

1. An individual actor's weight (w(i)) can be at most 33 % of the total network weight (network_w).

2. The sum of weights (sum(w)) required to reach a consensus must be more than 51 % of the total network weight.

This model represents a consensus mechanism where more established participants (evidenced by their longer presence, hence higher weight) have more influence, yet no single actor can dominate the decision-making process thus improving security. It ensures a distributed and time-based contribution to the consensus. BiniBFT reduces latency by polling only a subset of the network of votes which increases throughput and scalability.

IV. Conclusion

This whitepaper provides a brief summary of the Hyperledger Fabric compatible BFT consensus mechanisms chosen as candidates to design a new BFT protocol. A comparative study of the existing candidate BFT consensus mechanisms were studied with the parameters latency, security, and throughput. The study clearly underlined the need for a BFT consensus with high security and scalability. This paper presents one such protocol, the BiniBFT consensus mechanism for Hyperledger Fabric. Two versions of BiniBFT namely, BiniBFT with Random Polling, and The Time-Weighted BiniBFT Consensus protocols are presented in detail. The low latency, high security, throughput, and scalability of the BiniBFT can make it to be used as the consensus for day-to-day applications using enterprise blockchain.

References

[1] Miguel Castro and Barbara Liskov Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139, 1999 https://pmg.csail.mit.edu/papers/osdi99.pdf

[2] Ramakrishna Kotla, Lorenzo Alvisi, Mike Dahlin, Allen Clement, and Edmund Wong Dept. of Computer Sciences University of Texas at Austin, Dec 2009 https://www.cs.cornell.edu/lorenzo/papers/kotla07Zyzzyva.pdf

[3] Chrysoula Stathakopoulou, Tudor David, Matej Pavlovic, Marko Vukolic, Mir-BFT: High-Throughput Robust BFT for Decentralized Networks arXiv:1906.05552v3 [cs.DC] 22 Jan 2021] https://arxiv.org/pdf/1906.05552.pdf

[4] Wang, Y. (2019). Byzantine Fault Tolerance in Partial Synchronous Networks. College of Computing and Informatics, UNC Charlotte. https://eprint.iacr.org/2019/1460.pdf

[5] Guangda Sun, Xin Zhe Khooi, Yunfan Li, Mingliang Jiang, and Jialin Li, The Case for Accelerating BFT Protocols Using In-Network Ordering, arXiv:2210.12955v1 [cs.DC] 24 Oct 2022 https://arxiv.org/pdf/2210.12955.pdf

[6] João Sousa, Alysson Bessani, (2014). State Machine Replication for the Masses with BFT – SMaRt
https://www.researchgate.net/publication/286593169_State_machine_replication_for_the_masses_with_BFT-SMART

[7] Barger, Artem, et al. "A byzantine fault-tolerant consensus library for hyperledger fabric." 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2021.https://arxiv.org/pdf/2107.06922.pdf

[8] Hyperledger Fabric v3.0.0-preview Release Notes - September 1, 2023 https://github.com/hyperledger/fabric/tree/v3.0.0-preview