Security and Performance Analysis of Quorum-based Blockchain Consensus Protocols

Gabriel Antonio F. Rebello[1,2], Gustavo F. Camilo[1], Lucas C. B. Guimarães[1], Lucas Airam C. de Souza[1], Otto Carlos M. B. Duarte[1]

[1]Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ
[2]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

***Abstract*—Consensus protocols for blockchain-based systems are crucial and the most complex part of the blockchain. The proof-of-work consensus protocol of Bitcoin cryptocurrency is the most popular, but it presents a low transaction rate and high energy consumption. This paper analyzes the main quorum-based consensus protocols for blockchains that are alternatives with higher throughput and energy-efficient. The paper focuses on Byzantine fault-tolerant protocols as they are more robust to security attacks. We classify and analyze quorum-based consensus protocols according to their main characteristics and performance, highlighting the flaws and the specific attacks that affect each consensus protocol and presenting countermeasures.[1]**

***Index Terms*—blockchain, consensus protocols, security**

I. Introduction

Blockchain is a disruptive technology that provides trust among a group of participants without mutual trust, in a decentralized manner, and without intermediaries. A collective agreement of the participants obtained in a distributed manner replaces the centralized decision. Therefore, a blockchain system must be able to incorporate new blocks into the chain through consensus[2] among the participants. The blockchain technology has been shown to provide security in several multi-agent contexts, such as data sharing in the Internet of Things [1], [2], e-health record sharing [3] and multi-tenant networks [4]. Satoshi Nakamoto [5] revolutionized the asset transfer area with the cryptocurrency Bitcoin and the innovative consensus proposal, called Proof of Work (PoW). The consensus through proof-of-work, however, is probabilistic because it can generate forks in the blockchain, requires a lot of energy consumption, presents a low transaction rate, and shows a tendency to centralize decisions in participants with more computational power.

The Bitcoin presents a throughput of seven transactions per second and one-hour latency when the user waits for six-block generation cycles, which is the recommended guarantee for fork decisions. These characteristics limit the use of Bitcoin in applications that are unable to wait an hour to be completed, or by credit card companies that process more than 56,000 transactions per second [6]. Furthermore, proof of work in Bitcoin annually consumes more than the energy generated by 160 countries individually and the energy expenditure to process a single Bitcoin transaction is enough to supply an average United States of America household over 52 days [7]. In response to the performance limitations of proof-of-work consensus, several new consensus protocols emerge as possible substitutes for the Bitcoin protocol. This paper analyzes and compares the main quorum-based[3] deterministic consensus protocols proposed as an alternative to proof-of-work. We classify the deterministic consensus protocols as practical Byzantine fault-tolerant and its variants, federated and delegated Byzantine fault-tolerant, and hybrid. We present and analyze the characteristics, the transaction rate, and the security of each class of deterministic consensus protocol. Unlike other protocol performance analysis papers, this paper emphasizes the security of the analyzed protocols, specifying the main threats, the attacks discovered on each platform, and the possible countermeasures. The analysis covers the main existing cryptocurrencies, such as: XRP, NEO and EOSIO. Besides, we analyze the execution model of Hyperledger Fabric, which is the largest private blockchain platform.

II. Related Work

The blockchain consensus area has been attracting the attention of several researchers due to the success of the Bitcoin cryptocurrency. There is, however, a need for a systematic analysis of the vulnerabilities and countermeasures to security attacks in quorum-based consensus protocols.

Vukolić compares blockchains based on deterministic consensus and proof of work [8]. Angelis *et al.* assess the performance of proof of authority consensus protocols and compare them with the PBFT [9] protocol. Nevertheless, the papers focus on discussing the scalability of consensus protocols instead of providing an in-depth discussion of the vulnerabilities that exist in each protocol. Xiao *et al.* [10], Joshi *et al.* [11], and Lashkari *et al.* [12] present different deterministic and probabilistic consensus protocols for blockchain. Despite describing several consensus mechanisms, the discussion of threats and security flaws in vote-based consensus algorithms is brief and does not provide countermeasures. Hasanova *et al.* and Oyinloye *et al.* analyze the security and performance of

[1]This paper was funded by CNPq, CAPES, FAPERJ and FAPESP (18/23292-0, 2015/24514-9,ma 2015/24485-9 2014/50937-1).

[2]Consensus is a type of agreement reached among all members of a group.

[3]In this paper, we consider quorum-based protocols and deterministic protocols as synonyms.

several blockchain protocols [13], [14]. The works, however, focus on the vulnerabilities and countermeasures of proof-based protocols. Wang *et al.* present two vulnerabilities in the NEO [15] cryptocurrency consensus protocol. Christodoulou *et al.* analyze the security of the Ripple cryptocurrency in the presence of multiple adversaries [16]. Nevertheless, the papers focus on cryptocurrencies and specific protocols, without extensive comparison with others.

This paper clearly and concisely presents the required main characteristics of quorum-based consensus protocols. Besides, the paper describes the main quorum-based protocols for blockchains, focusing on specific security vulnerabilities of each protocol and comparing them. This paper is complementary to our analysis of proof-based protocols [17].

III. Quorum-based Blockchain Consensus

The consensus is the process by which, from a group of independent participants, all the correct participants reach the same decision to accept or refuse to add a new block in the blockchain. One of the consensus participants submits a proposal for a new block. In quorum-based consensus, participants exchange messages with two primitives:

1) $Propose(P, b)$: proposes a new block b to the set of consensus participants P. Only a special participant, the consensus leader, can send this primitive;
2) $Decide(b)$: informs the network that the participant validated and decided on the block b.

Consensus occurs every time the leader proposes a new block b, and the majority of participants validate and decide on the proposed block. Obtaining consensus is not trivial, as failures might occur both in the delivery of messages, which can be delayed or lost, and in the decision of the participants, that can fail due to power outages or malicious behavior. In an untrusted environment, consensus through message exchange occurs if and only if the protocol guarantees the following conditions [18].

- **Termination**: Every correct participant[4] eventually decides on a block b to be added in the blockchain.
- **Agreement**: The block b is identical in all correct participants.
- **Validity**: The decided block b by the correct participants is the block proposed by the leader at the beginning of the consensus.
- **Integrity**: A correct participant proposes the block b.

The consensus protocols seek to guarantee the four conditions that together provide the safety and liveness properties. The guarantee of the termination requirement provides liveness as the rounds of consensus continue to happen, and the system always incorporates new blocks. The guarantee of the termination, however, does not guarantee that the blocks are correct. The agreement requirement provides decision uniformity in all participants, and the validity and integrity requirements represent the correctness of the decision, ensuring that an honest participant proposed the block. Together, agreement, validity, and integrity provide the consistency property to the protocol. Consistency does not guarantee that the system always incorporates new blocks, but it does guarantee that the ones that are incorporated are always correct. Therefore, a consensus protocol must provide both properties to ensure the correction of the system even when failures occur.

Consensus protocols can tolerate two types of failures: crash faults or Byzantine faults. A crash fault participant does not respond and does not perform new operations during the consensus execution. The Byzantine failure is much more complicated since the failing participant can be a malicious agent that exhibits arbitrary behavior, deviating from the specified protocol, and taking any action. The malicious agent may behave well, responding correctly, may respond incorrectly, or may not respond at all. Also, a Byzantine failing participant may answer that it approves a block b to one participant and that it approves a block b' to another participant. Thus, in the Byzantine failure model, there is no precise information about the behavior of the participants or whether the system information is correct. The maximum number of malicious participants[5] that a quorum-based system can tolerate is one-third of the total network participants, including honest and malicious participants. This paper focuses on Byzantine fault-tolerant consensus, as they are robust to malicious behavior.

Communication systems are essential to obtain quorum-based consensus, as consensus participants must exchange messages to reach a result. Fisher, Lynch, and Paterson (FLP) published in 1985 one of the most important works concerning the consensus problem [19], known as the FLP impossibility. They prove that it is not possible to reach consensus on a completely asynchronous distributed system in which at least one participant can fail (any type of failure). Therefore, most of the consensus proposals found in the literature consider eventually synchronous communication systems that work asynchronously, respecting no time limit, most of the time. During periods of stability, however, the time for message delivery is limited. This model is realistic because it encompasses the behavior of best-effort networks, such as the Internet, which operates through times of stability and times of disturbance, in addition to having a deterministic solution to the consensus [19].

Unlike Bitcoin proof-of-work consensus, quorum-based consensus requires awareness of the identity and number of consensus participants to compute the number of votes in favor of a consensus proposal. Therefore, all participants need permission to participate in the consensus. Quorum consensus is permissioned consensus. Besides, authorized participants must be authenticated to avoid a Sybil attack. In Bitcoin, a Sybil attack is not serious because what is crucial is the processing power. In the quorum consensus, however, what matters is the votes and, therefore, Sybil attacks are critical. Thus,

[4] A correct participant is a participant that is not in a failed or Byzantine failure state. In this paper, we use "honest" as a synonym for "correct" and "malicious" as a synonym for "at fault."

[5] This problem is known as the problem of Byzantine generals solved by Lamport [18].

messages exchanged by participants must be authenticated to identify and eliminate malicious participants. Authentication can take place through asymmetric encryption to generate digital signatures of messages, which requires a public key infrastructure system, or through symmetric encryption and use of message authentication codes (MAC), which requires prior sharing of secrets.

We divide quorum-based consensus protocols into three broad classes: the practical Byzantine fault-tolerant protocol and its derivatives, the delegated Byzantine fault-tolerant protocols, and the federated Byzantine fault-tolerant protocols. The proposals have different characteristics and performance, as they seek to focus on a certain property such as decentralization, with the rotation of the leader, or the transaction throughput, with the division of the consensus in federations. In turn, security attacks seek to delay or prevent consensus.

IV. Practical Byzantine Fault Tolerance

The Practical Byzantine Fault Tolerant (PBFT) protocol [20] is the first consensus protocol to solve the problem of the Byzantine generals in a practical manner. PBFT is named "practical" because it optimizes authentication and communication between participants while still working as expected in asynchronous environments such as the Internet. The consensus centralizes all update proposals on the consensus leader, which is called the *primary*. The rest of the consensus participants are replicas, called *backups*. The protocol implements a view-change protocol that guarantees liveliness even if the leader fails. Participants are sequentially numbered. Thus, when the replicas detect a failure in the leader and activate the view-change protocol, the next participant on the circular queue becomes the new leader, and a new round of consensus begins. Each participant knows all the n participants in the consensus and their respective digital signatures. Thus, participants know the minimum number of votes needed to approve an update and can easily verify the origin and authenticity of the messages.

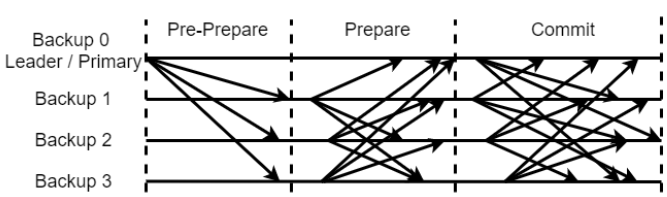

Fig. 1: Phase sequence of the PBFT protocol in a consensus round with four honest participants.

The main limitation of the PBFT protocol is scalability regarding the number of consensus participants, due to the high computational complexity in exchanging messages. The n participants need to exchange $O(n^2)$ messages to tolerate malicious behavior. Another limitation of the PBFT protocol is the impossibility of adding participants at runtime because if any participant leaves the network permanently, their behavior is considered a failure. This procedure hinders the use of PBFT in dynamic networks, where participants can enter and leave on demand. A critical attack on PBFT is to delay the completion time of the phases in the presence of malicious participants [21]. As timeouts perform the failure detection of the leader, a malicious leader can drastically reduce the consensus throughput by delaying message delivery as long as possible without being detected.

V. Federated Byzantine Agreement (FBA)

The main idea of the Byzantine fault-tolerant federated protocols is to partition the quorum and thereby significantly increase the throughput of transactions, reducing the number of messages exchanged and the cost per transaction. The Federated Byzantine Agreement (FBA) is a form of Byzantine agreement in which each Byzantine general is responsible for his/her quorum slice. While traditional Byzantine agreement protocols restrict admission to the consensus to prevent Sybil attacks, FBA grants each participant the freedom to select who to trust. Thus, even if an attacker creates multiple identities, he/she needs to convince a large number of legitimate participants to add malicious identities to their lists of trusted participants. The two main consensus protocols that implement the federated Byzantine agreement are the Stellar consensus protocol and the Ripple consensus protocol.

The Ripple Protocol Consensus Algorithm (RPCA) [22] was initially implemented in 2012 by Ripple Labs. The goal of its cryptocurrency, XRP, is to offer security against Byzantine failures while providing a higher throughput of transactions per second when compared to Bitcoin and Ethereum. The low latency required for the desired high transaction rate is achieved with the introduction of subnets of participants that are considered reliable within the main network, according to the FBA model.

To participate in the Ripple protocol as a validator, a participant must run a server capable of accepting and processing XRP cryptocurrency transactions. Each validator has a Unique Node List (UNL), which contains the set of participants considered to be reliable by the validator. The selection of reliable validators occurs in a manner that attempts to minimize the risk of Sybil attacks. In the process of reaching consensus, the validator considers only the votes of the servers in the UNL to determine which transactions should be appended to the blockchain. The protocol guarantees that all participants correctly achieve the same result in a consensus round and that it is impossible to validate a fraudulent transaction as long as less than 20% of participants are Byzantine.

The Ripple protocol is vulnerable to Sybil attacks, as an attacker can create multiple validators to try to gain control over the consensus. The UNL, however, is a powerful countermeasure against the Sybil attack because only validators considered reliable by other nodes in the network can have a direct influence on the consensus protocol. To avoid the impact of malicious validators, Ripple Labs maintains a standard UNL that contains trusted companies and groups interested in the growth of the currency. Besides, validators do not receive any form of incentive to perform consensus, which is a choice made to encourage only the presence of validators that are interested in the progress of the currency, in contrast with validators that only seek incentive.

Christodoulou *et al*. present an analysis of the Ripple protocol in adversarial environments [16]. The authors change the percentage of malicious participants in the network and the percentage of overlapping participants in UNLs. Their first result analyzes the impact of the presence of Byzantine participants in the time needed to reach consensus. In contrast, the second result analyzes the impact of decentralization in the protocol in that same measure. The authors confirm that the convergence time is not impacted if there are up to 20% Byzantine participants in the protocol. Besides, the impact for higher percentages can be mitigated as long as the UNLs of participants overlap. The authors also consider the Network Health Indicator (NHI), a measurement that indicates how many candidate blocks were validated during the consensus. The results show that, even with UNL overlap rates of up to 10%, the protocol achieves good NHI values if there are few Byzantine participants.

The Stellar consensus protocol (SCP), proposed in 2016 by David Mazières and used by the cryptocurrency Lumens (XLM), has as its main advantages the integration of low latency from the BFT consensus with the flexible trust and decentralized control of the FBA model. Each SCP consensus participant selects one or more sets of other participants, called quorum slices, that they trust to exchange messages and decide on a new block. At each round, the participant analyzes the different views of the different quorum slices to make its decision. A participant can belong to several quorum slices and establish different decision thresholds for each slice.

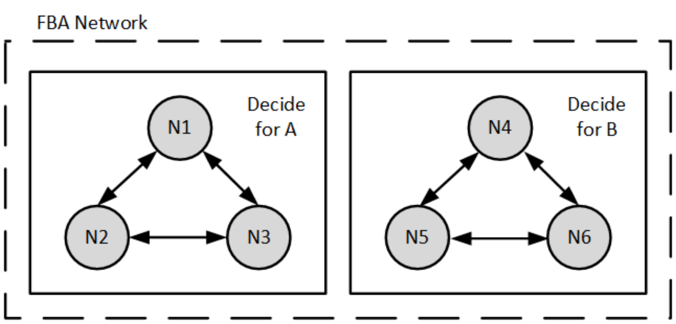

Fig. 2: Scenario of a network that uses the Stellar protocol and has no intersection between quorums. The arrows indicate the direction of trust of each participant. As the quorums do not intersect, there is no guarantee of agreement on the network.

The SCP's flexible trust model has a strong influence on the security of the protocol since the correct functioning of the protocol depends on how the quorum slices intersect. The SCP loses the guarantee of agreement when there is no intersection of complete quorums, as disconnected quorums can agree in contradictory blocks. Figure 2 shows an SCP network in which there is no guarantee of agreement. In the figure, the complete quorum formed by participants N1, N2, and N3 decides for a block A as the correct block. Meanwhile, the quorum formed by participants N4, N5, and N6 decides for a block B. Thus, the system does not reach a value agreed by the participants, compromising the property of agreement.

Kim *et al.* analyze the Stellar network to check the influence of each participant in the system and measure the degree of centralization [23]. The authors use a modified version of the PageRank algorithm called NodeRank, which considers: (i) the number of quorum slices that contain the participant; (ii) if any participant with strong influence has the participant in its quorum slice; (iii) the threshold value of the slice that contains the participant. The article finds that the most important participants in the Stellar network are 3 validators of the Stellar Foundation. Besides, the results show the cascading-failure risk, in which several participants fail due to the failure of some other participants. In the case of the Stellar network, the authors show that the consensus may not reach a decision if 2 of the 3 participants in the Stellar Foundation fail.

VI. Delegated Byzantine Fault Tolerance

The delegated Byzantine fault-tolerant (dBFT) protocol follows the same phases of the BFT protocols of Section IV but centralizes the consensus in a shortened number of participants to provide higher throughput and scalability. The protocol uses the concept of reputation to choose the nodes participating in the consensus, and, comparable to the Spinning consensus, the leader changes circularly with each round. In the original dBFT implementation, however, there is no commit phase, which exists in the PBFT protocol, which makes the consensus vulnerable to Byzantine failures. The authors added the commit phase to dBFT to ensure that honest nodes agree on the state change.

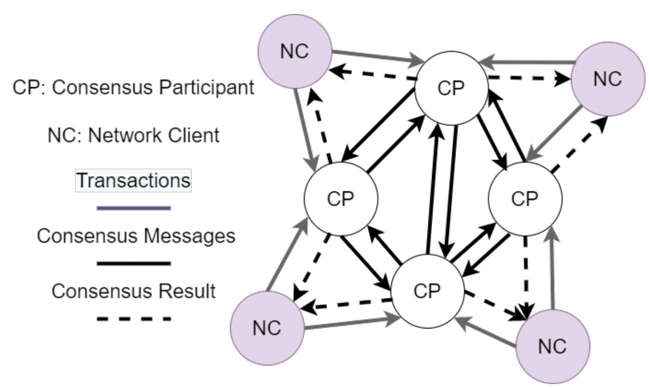

Fig. 3: dBFT consensus network topology.

According to Wang *et al.*, the dBFT protocol of the NEO [15] cryptocurrency exhibits a security vulnerability. A malicious node can create a deterministic fork, known as a spork, by exploiting the view-change protocol. A malicious leader can store messages and create two valid blocks approved by honest consensus participants with different views. Since the blocks are valid, honest participants can accept either one and create two different states on the network. One solution to these problems is to discard messages generated before the view change, making it impossible for honest participants to accept two valid states.

VII. Byzantine Fault Tolerant and Hybrid Delegated Proof of Stake

The Byzantine Fault Tolerance - Delegated Proof of Stake (BFT-DPoS) Hybrid Protocol associate the advantage of the high performance of proof of possession (PoS) protocols with the security and determinism of Byzantine fault-tolerant (BFT) protocols. An example of this class is the EOSIO [24] protocol proposed and developed by Daniel Larimer and used in the digital currency EOS. The protocol achieves more than 3000 transactions per second and the guarantees of determinism and security against Byzantine attacks offered by BFT. The protocol has two phases: the election of delegates and the production of blocks [25].

In the first phase, users who hold tokens (token holders) can use them to elect block producers through a voting system. Each EOSIO stakeholder can vote for up to 30 block producers per voting action. The 21 most voted producers act as delegates who produce blocks on behalf of the interested parties and participate in the BFT consensus[6]. Any member of the network can apply to become a block producer as long as they receive at least one vote from another token holder. However, in practice, the block producers set have small variations and consist of entities that invest directly in the growth of the cryptocurrency, such as the EOS New York and EOS Beijing consortia.

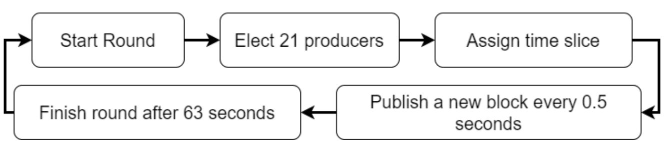

Fig. 4: EOSIO consensus protocol round execution diagram.

After elected, the delegates begin a round of block production through a Byzantine agreement. During the round, each delegate has six fixed 0.5 seconds time slots to produce blocks. The block production order is alphabetical and, if a producer fails to generate a block in the time slice, the next producer ignores the block and continues the process. Thus, forks can occur, and each round adds up to 126 blocks to the blockchain, lasting a total of 63 seconds. Figure 4 illustrates the main stages of a round. Finally, the protocol resolves the forks through Byzantine fault tolerance (BFT) before adding them to the blockchain. When 15 of the 21 producers, i.e., more than 2/3 of the delegates, confirm a block through signed messages, the protocol adds it to the blockchain.

The main weakness of the EOSIO protocol is the centralization of only 21 delegates who are elected by voting. This model has clear vulnerabilities: (i) As each vote has a weight proportional to the participant's assets, collusion among a few participants with large possessions is sufficient to elect malicious delegates. Evidence of this type of attack already exists in practice, through patterns of voting gangs identified in the EOSIO blockchain [26]; (ii) The network is susceptible to double-spend attacks with only $\lfloor \frac{n}{3} - 1 \rfloor = 6$ malicious delegates. As each participant can vote for up to 30 delegates simultaneously, the election of 6 malicious delegates can be easily achieved. The EOSIO community mitigates this problem by frequently electing the same participants based on their reputation. This practice, however, centralizes the process of producing blocks in a fixed set of entities that can be targets of attacks. Lee *et al.* demonstrate that an attacker is able to modify the behavior of a trusted delegate through denial of service attacks and memory hijacking [27]. EOSIO's developers have recognized the vulnerabilities presented and plan to fix them in the future [27], [28]. (iii) After being elected, delegates have the same power regardless of the number of votes received. This feature minimizes the cost of collusion attacks, as attackers need to bet only a small set of assets sufficient to elect the least voted delegates.

Although consensus occurs only among elected delegates, EOSIO also suffers from problems similar to public platforms like Ethereum, as any participant can initiate a smart contract or issue a transaction anonymously. A recent study indicates that more than 30% of users on EOSIO correspond to botnets, and more than 300 attacks have already been detected in decentralized applications [30]. Unlike Ethereum, EOSIO allows modifying smart contracts already published in the blockchain, which opens space for code modification attacks in contracts that are not open source [29]. EOSIO smart contracts also have a functionality that allows the contract to automatically manage a user's tokens, which in practice causes millionaire financial losses and abuses [27], [30].

VIII. Blockchain Platforms Security and Performance Analysis

Hyperledger Fabric is an open-source platform for the development of permissioned blockchain [31]. The platform uses general-purpose programming languages for writing smart contracts, and its modular architecture allows the implementation of different consensus algorithms. Despite proposing to implement a BFT blockchain, Fabric does not yet present an implementation of a BFT protocol[7] and features three consensus mechanisms: Solo, Kafka/Zookeeper, and Raft. A group of nodes called orderers uses the consensus mechanism to define the order of a set of transactions in a block.

Hyperledger Fabric innovates compared to other blockchains in the transaction execution using the execute-order model (XO). The XO model, shown in Figure 5, executes transactions before ordering. In contrast to the order-execute model (OX), the XO model allows parallel transaction processing, since it orders after executing. The XO model enables a high transaction throughput on the Hyperledger Fabric, which reaches 3500 transactions per second [31]. Another advantage of the XO model is the efficient handling of non-deterministic transactions. While in the OX model these transactions generate forks for producing different outputs, Fabric maintains the agreement in the

[6]The total number of 21 producers is the result of a previous work by the author in which users could vote to define the number of producers.

[7]https://hyperledger-fabric.readthedocs.io/en/release-2.2/Fabric-FAQ.html?highlight=bft#bft

TABLE I: Analysis of blockchain consensus protocols.

Protocol	Consensus	Max. throughput	# validators	Vulnerabilities	Countermeasures
PBFT	Traditional BFT	≈ 300 tx/s	Dozens	Delays in round finalization.	Change the primary and monitor transaction rate.
NEO	Delegated BFT	≈ 10000 tx/s	7	View-change exploration [15] and deterministic fork (spork).	Refuse messages after view-change and implement commit phase.
EOSIO	BFT-DPoS	≈ 3000 tx/s	21	Exploiting smart contracts [29], [26], manipulating delegates [27] and subverting consensus through voting gangs [30].	Restrict the functionality of smart contracts and increase the number of delegates in the consensus.
Ripple	FBA	≈ 1500 tx/s	Dozens	Sybil attacks and exploiting quorum intersections.	Automatic use of standard UNL updated by Ripple Labs.
Stellar	FBA	≈ 1000 tx/s	Dozens	Exploiting quorum intersections [23].	Insertion of trusted validators to provide the basis of a quorum.

network by discarding non-deterministic transactions. The discarding is possible due to the pre-order execution that detects the inconsistency in the validators' messages and invalidates them before the orderer adds the transaction in a block. This advantage allows Fabric to uses general-purpose programming languages in smart contracts, as opposed to blockchains that use the OX model, that require specific languages to prohibit non-deterministic functions.

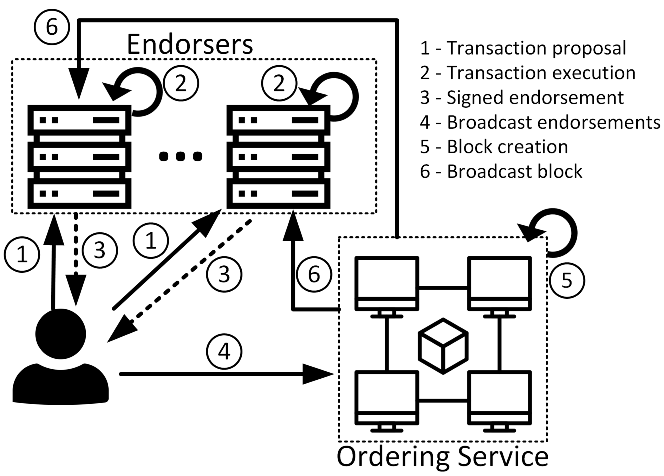

Fig. 5: The Hyperledger Fabric execute-order (XO) architecture. Endorsers execute transactions before creating a block, preventing non-deterministic behavior in the system.

The main disadvantage of the XO model is the inefficiency in handling transactions that change the same state [32]. As the validating nodes process transactions in parallel, conflicting transactions can generate different outputs depending on the order in which they were executed and cause an inconsistency in the network. To avoid this, Fabric discards these transactions at the ordering stage. This solution, however, decreases the network throughput and forces the customer to resubmit the discarded transaction. Gorenflo *et al.* proposes a solution to this problem with the adoption of the XOX model execute-order-execute, which adds an execution phase after ordering [33]. The additional phase re-executes the conflicting transactions that would be discarded and make unnecessary resubmit the transaction.

IX. Conclusion

Byzantine fault-tolerant protocols provide the best safety in high transaction rate blockchains [22], [20]. Table 1 presents a comparison between the protocols analyzed in this paper. The practical Byzantine fault-tolerant (PBFT) consensus protocol is robust in terms of safety. However, with a malicious primary, its performance can be extremely degraded, harming the system's liveliness. The protocol is unsuitable for dynamic networks, in which the consensus nodes set frequently change. Another difficulty to use the PBFT is the protocol low nodes number scalability. The PBFT variation, called Spinning, changes the primary each round, mitigating the problems of liveliness. It suffers, however, from the same PBFT problems due to scalability and dynamic networks.

The Ripple and Stellar protocols are based on the federated Byzantine agreement to increase throughput, decrease transaction costs, and deal with dynamic node sets. The main idea is to slice the quorum and thereby ease confidence. The protocol is resistant to Sybil attacks, due to several participants have to trust and add the attacker identities to the quorum slices. The security and performance of the Federated Byzantine Agreement depend directly on the number of participants, the number of quorum slices, and the size of the intersection between the slices. In both the Ripple and Stellar protocol, the system mitigates this problem by sharing a standard list of trusted participants for all network new users. The user can modify the list later.

The dBFT and EOSIO present a high performance compared to PBFT and are suitable for dynamic environments or with a large number of participants. It is vital for security, though, how to select the consensus participants, since not all nodes are responsible for generating new blocks. NEO Foundation and the company Block.One recognized the security flaws found in the implementation of dBFT and EOSIO, respectively, and will correct in the next versions.

References

[1] G. F. Camilo, G. A. F. Rebello, L. A. C. de Souza, and O. C. M. Duarte, "A secure personal-data trading system based on blockchain, trust, and reputation," in *2020 IEEE International Conference on Blockchain*. IEEE, 2020, pp. 379–384.

[2] D. M. F. Mattos, D. S. V. de Medeiros, and D. Mosse, "Enabling technologies for running iot applications on the cloud," in *Annals of Telecommunications*. Springer, 2022, pp. 1–2.

[3] D. C. Nguyen, M. Ding, P. N. Pathirana, and A. Seneviratne, "Blockchain and ai-based solutions to combat coronavirus (COVID-19)-like epidemics: A survey," *IEEE Access*, vol. 9, pp. 95 730–95 753, 2021.

[4] S. K. Taskou, M. Rasti, and P. H. Nardelli, "Blockchain function virtualization: A new approach for mobile networks beyond 5G," *IEEE Network*, 2022.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008, Last access: 20 July 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[6] BitcoinWiki, "Bitcoin Scalability," 2022, Last access: 20 July 2022. [Online]. Available: https://en.bitcoin.it/wiki/Scalability

[7] Digiconomist, "Bitcoin Energy Consumption Index," 2022, Last access: 20 July 2022. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption

[8] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *International Workshop on Open Problems in Network Security*. Springer, 2015, pp. 112–125.

[9] S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "PBFT vs Proof-of-Authority: applying the CAP theorem to permissioned blockchain," in *Italian Conference on Cyber Security*, January 2018. [Online]. Available: https://eprints.soton.ac.uk/415083/

[10] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[11] A. P. Joshi, M. Han, and Y. Wang, "A survey on security and privacy issues of blockchain technology," *Mathematical Foundations of Computing*, vol. 1, no. 2, p. 121, 2018.

[12] B. Lashkari and P. Musilek, "A comprehensive review of blockchain consensus mechanisms," *IEEE Access*, vol. 9, pp. 43 620–43 652, 2021.

[13] H. Hasanova, U. Baek, M. Shin, K. Cho, and M.-S. Kim, "A survey on blockchain cybersecurity vulnerabilities and possible countermeasures," *International Journal of Network Management*, vol. 29, no. 2, p. e2060, 2019.

[14] D. P. Oyinloye, J. S. Teh, N. Jamil, and M. Alawida, "Blockchain consensus: An overview of alternative protocols," *Symmetry*, vol. 13, no. 8, p. 1363, 2021.

[15] Q. Wang *et al.*, "Security analysis on dBFT protocol of NEO," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 20–31.

[16] K. Christodoulou, E. Iosif, A. Inglezakis, and M. Themistocleous, "Consensus crash testing: Exploring ripple's decentralization degree in adversarial environments," *Future Internet*, vol. 12, no. 3, p. 53, 2020.

[17] G. A. F. Rebello, G. F. Camilo, L. C. Guimarães, L. A. C. de Souza, G. A. Thomaz, and O. C. Duarte, "A security and performance analysis of proof-based consensus protocols," *Annals of Telecommunications*, pp. 1–21, 2021.

[18] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, Jul. 1982. [Online]. Available: http://doi.acm.org/10.1145/357172.357176

[19] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *Journal of the ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985.

[20] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[21] Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Prime: Byzantine replication under attack," *IEEE transactions on Dependable and Secure Computing*, vol. 8, no. 4, pp. 564–577, 2010.

[22] D. Schwartz *et al.*, "The ripple protocol consensus algorithm," *Ripple Labs Inc White Paper*, vol. 5, no. 8, 2014.

[23] M. Kim, Y. Kwon, and Y. Kim, "Is Stellar as secure as you think?" in *IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2019, pp. 377–385.

[24] D. Larimer, "EOS.IO White Paper," 2017, Available at: https://developers.eos.io/welcome/latest/protocol/consensus_protocol. Last access: 20 July 2022.

[25] EOSIO, "EOSIO development documentation," 2018, Available at: https://developers.eos.io/welcome/latest/protocol/consensus_protocol. Last access: 20 July 2022.

[26] Y. Zhao, J. Liu, Q. Han, W. Zheng, and J. Wu, "Exploring EOSIO via graph characterization," in *International Conference on Blockchain and Trustworthy Systems*. Springer, 2020, pp. 475–488.

[27] S. Lee, D. Kim, D. Kim, S. Son, and Y. Kim, "Who spent my EOS? On the (in)security of resource management of EOS.IO," in *13th USENIX Workshop on Offensive Technologies*, 2019. [Online]. Available: https://www.usenix.org/conference/woot19/presentation/lee

[28] Block One, "EOSIO RAM resource exploit patch," 2018, Available at: https://block.one/news/eosio-ram-resource-exploit-patch/. Last access: 20 July 2022.

[29] N. He, R. Zhang, H. Wang, L. Wu, X. Luo, Y. Guo, T. Yu, and X. Jiang, "EOSAFE: Security analysis of EOSIO smart contracts," in *30th USENIX Security Symposium*, 2021, pp. 1271–1288.

[30] Y. Huang *et al.*, "Understanding (mis) behavior on the EOSIO blockchain," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 4, no. 2, pp. 1–28, 2020.

[31] E. Androulaki *et al.*, "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3190508.3190538

[32] J. A. Chacko, R. Mayer, and H.-A. Jacobsen, "Why do my blockchain transactions fail? A study of hyperledger fabric," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 221–234.

[33] C. Gorenflo, L. Golab, and S. Keshav, "XOX Fabric: A hybrid approach to blockchain transaction execution," in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, to be published.