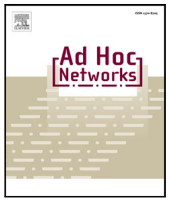
Ad Hoc Networks 158 (2024) 103461

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc



vCubeChain: A scalable permissioned blockchain

Allan Edgard Silva Freitas [a], Luiz Antonio Rodrigues [b,*], Elias Procópio Duarte Jr. [c]

[a] Federal Institute of Bahia (IFBA), Rua Emídio dos Santos, s-n, Salvador, 40301-015, Bahia, Brazil
[b] Western Parana State University (UNIOESTE), Rua Universitária, 2069, Cascavel, 85819-110, PR, Brazil
[c] Federal University of Parana (UFPR), Rua Evaristo F. Ferreira da Costa, 383-291, Curitiba, 82590-300, PR, Brazil

ARTICLE INFO

Keywords:
Distributed computing
Consensus
Blocksim
vCube

ABSTRACT

This work presents vCubeChain, a scalable permissioned blockchain based on the vCube virtual topology. vCube is a virtual hierarchical topology that presents several logarithmic properties. vCubeChain employs a leader election algorithm that relies on the failure detection information that vCube provides. The leader employs a vCube-based autonomic reliable broadcast algorithm to disseminate blocks, each consisting of multiple transactions. In case multiple leaders end up concurrently elected due to false suspicions, vCubeChain is proven to recover to a consistent state upon the discovery of contradictory blocks. vCubeChain is described, specified, and correctness and liveness draft proofs are presented. The blockchain was implemented on the Blocksim simulator, and a set of experiments are presented, including comparisons with Bitcoin, Ethereum and Hyperledge Fabric. Results demonstrate the scalability of the solution.

1. Introduction

A blockchain is a distributed ledger that stores transaction records on a set of processes connected through a network. The major advantage of blockchains compared to other alternative technologies is that there is no need for a central authority to ensure security properties [1]. A large, growing number of applications have been proposed for blockchains, in diverse fields, such as cryptocurrencies, digital government transactions, document copyright protection, and real estate transactions, among several others [2].

Blockchains combine distributed and secure computing techniques to maintain a data structure – the block chain – that guarantees the persistence of transactions stored by the processes that make up the system. It is possible to classify blockchains into two basic types: permissioned (private) and non-permissioned (public). A permissioned blockchain requires all processes to know each other in advance so that they can be properly authenticated to execute any system operation. Examples of permissioned blockchains include Hyperledger Fabric [3], Corda [4], among others [5]. Permissionless blockchains allow the participation of unknown processes, which do not need to trust each other. Examples include Bitcoin [6], Ethereum [7], Algorand [8], among others [9]. These blockchains use consensus mechanisms based on "Proof-of-Work" (PoW) or "Proof-of-Stake" (PoS) to validate new blocks. PoW results in a high expenditure of energy, and the winning process receives a reward, in the case of Bitcoin, the cryptocurrency itself. Ethereum 2.0 uses a PoS strategy, which requires users to "pledge" currency, in this case, Ether (ETH), to become validators of the [10] network.

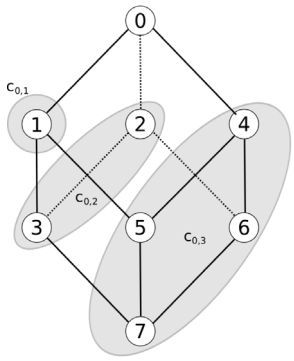
As processes trust each other in permissioned blockchains, instead of employing PoW or PoS, they can use classic consensus algorithms, such as Raft [11] or PBFT [12]. Those algorithms present strong consistency guarantees, but are expensive and do not scale well [13]. Furthermore, permissioned blockchains are generally based on a leader, who proposes the block to be stored and also manages the member processes.

Although permissionless blockchains can consist of a very large number of nodes, they have a very limited transaction flow compared to smaller-scale systems based on a predefined group of trusted processes running conventional consensus algorithms. However, in permissioned networks, the cost of broadcast mechanisms can be quadratic with respect to the number of participants, and are thus not scalable [14,15].

In this work, we present vCubeChain, a scalable permissioned blockchain. vCubeChain is based on the vCube hierarchical virtual topology [16]. The topology is maintained through a failure detector that forms a hypercube when all processes are correct and the number of processes is a power of two. As processes crash, vCube reorganizes itself, maintaining several logarithmic properties. vCubeChain is a permissioned blockchain, i.e., all processes are properly authenticated. A leader is elected using an autonomous reliable broadcast strategy to disseminate blocks across the network. Each block consists of multiple transactions. False suspicions may cause the election of multiple competing leaders simultaneously, allowing temporary forks to occur [6].

* Corresponding author.
E-mail address: luiz.rodrigues@unioeste.br (L.A. Rodrigues).

https://doi.org/10.1016/j.adhoc.2024.103461
Received 30 November 2023; Received in revised form 2 February 2024; Accepted 26 February 2024
Available online 11 March 2024



s	$c_{0,s}$	$c_{1,s}$	$c_{2,s}$	$c_{3,s}$	$c_{4,s}$	$c_{5,s}$	$c_{6,s}$	$c_{7,s}$
1	1	0	3	2	5	4	7	6
2	2 3	3 2	0 1	1 0	6 7	7 6	4 5	5 4
3	4 5 6 7	5 4 7 6	6 7 4 5	7 6 5 4	0 1 2 3	1 0 3 2	2 3 0 1	3 2 1 0

Fig. 1. Clusters of process 0 and the complete $c(i, s)$ of a 3-dimensional vCube.

In this situation, vCubeChain remains intact, always returning to a state consistent with the reconciliation of differences.

The blockchain was implemented on the Blocksim simulator, and a set of experiments are presented, including comparisons with Bitcoin, Ethereum and Hyperledge Fabric. Results are presented both for transaction processing time and the number of messages required, demonstrating the scalability of the proposed solution.

The rest of this work is organized as follows. Section 2 defines the system model, also including a brief description of the vCube virtual topology. In Section 3 vCubeChain is described and specified and proofs of correctness and termination are presented. The implementation of vCubeChain using simulation and evaluation results are described in Section 4. Related work is presented in Section 5. Finally, Section 6 concludes the paper.

2. System model

We assume a distributed system that consists of a set P of $n > 1$ processes $\{p_0, .., p_{n-1}\}$ that communicate by exchanging messages. Processes are also called *nodes*. Process p_i can also be referred to as process i. Each process can communicate directly with any other process, i.e., the system is fully connected and can be represented by a complete graph. A process can crash, and crashes are permanent. Each process can be in one of two states: a *correct* process is one that never fails; otherwise, the process has *crashed*. The operations to send and receive messages are atomic, but the broadcast primitives are not. The communication channels are perfect. Thus, messages exchanged between processes are never lost, corrupted or duplicated.

The processes form a virtual hierarchical topology called vCube [16, 17]. The virtual topology is a complete hypercube if all processes are correct and the number of processes is a power of two, but keeps the hypercube properties for any number of correct processes. vCube implements a pull-based failure detector [18], in which processes execute tests and exchange test result information so that all correct processes can determine the state of each other process as either *correct* or *suspect* of have crashed. Up to $n-1$ processes can crash. After processes crash, vCube reorganizes itself autonomously, maintaining several logarithmic properties such as the maximum distance between processes and the maximum number of tests each process has to reply to. The system is assumed to be partially synchronous with a Global Stabilization Time (GST). Thus, informally, the system is initially asynchronous, but after the GST it becomes synchronous, i.e. there are known upper bounds for both the time to execute a task and for transmitting messages between processes [19]. As a result, the failure detector can make mistakes, i.e., before the GST, a correct but slow process can be incorrectly suspected of having crashed.

As mentioned above, a process running vCube executes tests on other processes to determine if they are correct or suspected of having crashed. The tested process is considered to be correct if the tester receives a response within the expected time interval. Otherwise, the process is suspected. Processes run tests on progressively larger clusters. Each cluster $s = 1, .., \log_2 n$ has 2^{s-1} elements, where n is the total number of processes in the system. The processes in each cluster s and the order in which they are tested by a process i are given by function $c_{i,s}$, defined below. The symbol $\oplus$ represents the exclusive binary operation OR (XOR):

$$c(i, s) = \{i \oplus 2^{s-1}, ci \oplus 2^{s-1}, 1, \ldots, c(i \oplus 2^{s-1}, s-1)\} \quad (1)$$

Tests are executed in rounds. In each testing round, the set of testers of a process i consists of each first correct process j in clusters $c(i, s), s = 1, \ldots, \log_2 n$. If the tested process is correct, the tester obtains new information about the state of the other processes. A testing round is completed after all the correct processes have run all their assigned tests.

Fig. 1 illustrates the hierarchical organization of a 3-dimensional vCube with $n = 2^3$ processes. The table shows the elements of each cluster $c(i, s)$ for the system. In this system, each process tests three clusters. As an example, the first cluster tested by p_0 is $c(0, 1) = (1)$; the other two clusters are $c(0, 2) = (2, 3)$ and $c(0, 3) = (4, 5, 6, 7)$. In this case, processes p_1, p_2 and p_4 are tested. In each testing round, each process is tested by at least one correct process. This ensures that, in at most $\log_2 n$ rounds, all processes have updated their local state information about all other processes.

In [20] a reliable broadcast algorithm is presented for vCube. That algorithm assumes a synchronous system, and the processes form a minimum spanning tree that reconfigures itself autonomically as processes fail and recover. The tree guarantees that the broadcast completes in logarithmic time. A correct process forwards messages to the first correct process in each of its clusters. A version assuming an asynchronous system was proposed by [21], which deals with false suspicions by continuously sending messages to suspected processes.

3. vCubeChain: a scalable permissioned blockchain

This section presents vCubeChain — a scalable permissioned leader-based blockchain. vCubeChain is a distributed ledger that provides safe storage of valid transactions across its processes organized on a vCube. vCubeChain assumes authenticated processes, i.e. all processes know and trust each other. Initially, clients send new transactions to any vCubeChain process. Only the leader can validate a transaction. If the process that receives a new transaction is not the leader, it forwards the transaction to the leader. The leader validates the transaction and forms a new block after there is a sufficiently large number of valid transactions. Each new block is disseminated throughout the blockchain via vCube's autonomic reliable broadcast algorithm, which is described later in this section.

Processes use the underlying vCube failure detection service to elect the blockchain leader. The leader is simply defined as the correct process with the highest identifier. As processes are authenticated, there are no impersonation attacks in which an adversary pretends to be the

Algorithm 1 vCubeChain executed by process i

```
 1: procedure Init( )
 2:     correct_i ← P
 3:     pending ← ∅
 4:     CheckLeader( )
 5:     if IAmLeader( ) then
 6:         append genesis_block to chain
 7:         RBcast(Msg-type=BLOCK,
             genesis_block)

 8: procedure NewTransaction(msg m)
 9:     Let T be the new transaction with m
10:     if IAmLeader( ) then
11:         ProcessTransaction(T)
12:     else
13:         pending ← pending ∪ {T}
14:         SendToLeader(T)

15: procedure ProcessTransactions(T)
16:     if Validate(T) then
17:         AddToBlock(T)
18:     else
19:         NotifySender(T)

20: procedure AddToBlock(T)
21:     if !ThereIsCandidateBlock( ) then
22:         create candidate_block
23:     append T to candidate_block
24:     if BlockCompleted( ) then
25:         append candidate_block to chain
26:         RBcast(Msg-type=BLOCK,
             new_block = candidate_block)

27: upon receive T from p
28:     if IAmLeader( ) then
29:         ProcessTransaction(T)
30:     else
31:         SendToLeader(T)

32: upon receive ⟨BLOCK, new_block⟩ from p
33:     append new_block to chain
34:     for all T ∈ new_block do
35:         pending ← pending \ {T}

36: upon receive ⟨LEADER, new_leader⟩ from p
37:     if IAmLeader( ) and new_leader > i then
38:         pending ← pending
                ∪ candidate_block
39:         candidate_block ← ∅
40:         leader ← new_leader
41:         for all T ∈ pending do
42:             SendToLeader(T)

43: procedure checkLeader( )
44:     new_leader ← max(correct_i)
45:     if new_leader ≠ leader then
46:         leader ← new_leader
47:         if IAmLeader( ) then
48:             RBcast(Msg-type=LEADER, new_leader)
49:             for all T ∈ pending do
50:                 ProcessTransactions(T)
51:                 pending ← pending \ {T}
52:         else
53:             for all T ∈ pending do
54:                 SendToLeader(T)

55: upon notifying crash(process j)
56:     correct_i ← correct_i \ {j}
57:     CheckLeader( )

58: upon notifying up(process j)
59:     correct_i ← correct_i ∪ {j}
60:     CheckLeader( )
```

leader. However, due to false suspicions and the latency to propagate node state information across the system, it is possible that more than one node considers itself the leader. We show that even if multiple processes consider themselves to be leaders, vCubeChain eventually converges to a single leader and maintains global consistency.

vCubeChain is specified in pseudo-code in Algorithm 1. Users can register transactions at any process using the NewTransaction procedure. If the selected process is not the leader, it sends the transaction to the leader. The leader then validates the transaction in the ProcessTransactions procedure and includes it in a candidate block using the AddToBlock procedure. The leader maintains this block with a set of valid transactions until it fills up. The block size, i.e., the maximum number of transactions in a block, is a configurable parameter. Once the new block (line 24) is complete, the leader disseminates it across the vCubeChain via RBCast (line 26), an autonomic reliable broadcast algorithm [21]. This algorithm relies on the vCube hierarchical topology to ensure that the dissemination of blocks throughout the system will require a logarithmic number of communication steps to complete.

vCubeChain adopts the typical blockchain data structure shown in Fig. 2. The first block is called *genesis* and is proposed by the first leader. Each block consists of a set of transactions as well as the hash of the previous block and the hash of the current block.

3.1. Block proposition

New blocks are proposed to vCubeChain as shown in Fig. 3. Initially, a client sends a new transaction to any vCubeChain process (Fig. 3(a)). If this process is not a leader, it forwards the transaction to the leader (Fig. 3(b)). The leader validates the transaction and forms a new block with a sufficiently large number of valid transactions (Fig. 3(c)). Finally, in step 4, the new block is disseminated throughout the blockchain via vCube's autonomic reliable broadcast algorithm (Fig. 3(d)).

After a specific process receives a new transaction, that process saves it in the `pending` buffer (line 13). This transaction will only be removed of that buffer when the process receives a block containing it (line 35) or a notification from the leader that it is not valid (according to the state of the ledger). In case the current leader is suspected and another leader is elected, the process resends all transactions from the `pending` buffer to the newly elected leader. If both the suspected and new leaders persist transactions into new blocks, then a *fork* occurs. vCubeChain detects and resolves *forks* after a finite time interval to ensure global consistency through the mechanism described in Section 3.3.

3.2. Leader election

A process learns who the leader is by executing the CheckLeader procedure. In the case of a leader failure, the process with the highest identifier among the correct processes is chosen. Processes invoke the CheckLeader primitive to learn whether there has been a leader change. When a correct leader is incorrectly suspected of having failed, another leader is elected, but as soon as it ceases to be suspected it can become

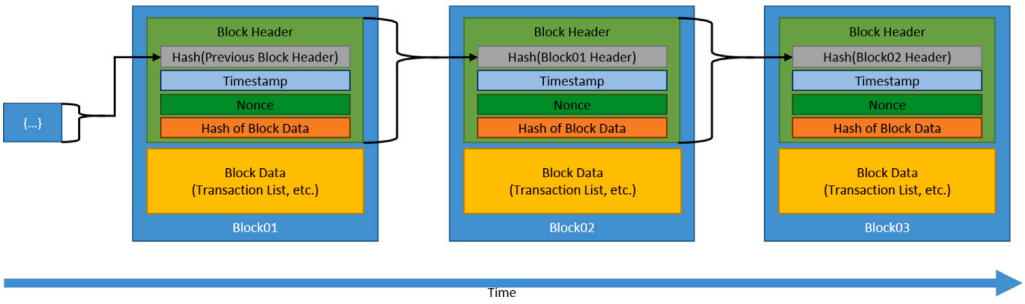
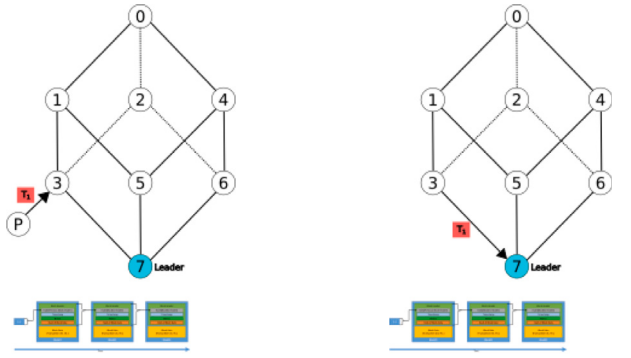


Fig. 2. Typical blockchain data structure (NIST).



(a) A new transaction T_1 is proposed to/by p_3 (that is not the leader)

(b) p_3 sends the transaction T_1 to the leader (p_7)

(c) p_7 receives many transactions and registers a new block

(d) p_7 broadcasts the new block to all other processes using vCube's spanning tree

Fig. 3. Transaction processing and new block proposition.

the leader again. vCubeChain assumes the GST timing model, according to which the leader will no longer be incorrectly suspected after a certain point in time.

In the example in Fig. 4, in Fig. 4(a), process 7 (p_7) is the leader but becomes unduly suspected. Then, process 6 is elected as the new leader. However, soon after, in Fig. 4(b), process 7 is again considered to be correct, and over time, the two processes, 6 and 7, are perceived as leaders by different system processes. Finally, in Fig. 4(c), the system stabilizes, and process 7 is again the sole leader.

The proposed strategy includes a consensus-based approach to eliminate inconsistencies resulting from the existence of multiple leaders, described in the next subsection.

3.3. Forks, consensus, and consistency

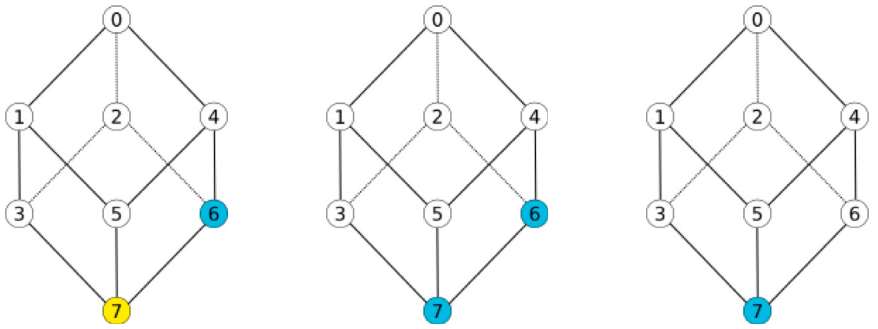
Dolev et al. (1987) proved that consensus can be reduced to reliable broadcast with message ordering. vCubeChain's consensus algorithm relies on that result. In normal operation, the leader receives transactions from clients and other system processes. After verifying their validity, the leader establishes a local order on the pending transactions by proposing a new block containing the transactions. The algorithm disseminates this block using vCube's reliable broadcast strategy, which guarantees delivery to all the correct processes after a finite time interval using an autonomic spanning tree.

Supposing that, in the face of false suspicions, two or more processes can become leaders, and all can propose blocks to the chain. As mentioned above, this causes the creation of forks. A fork consists of subchains that are not consistent with each other. However, as all leaders will reliably broadcast both subchains, all correct processes will have the same view of the blockchain, including the fork and derived subchains. In this scenario, the longest subchain is selected, so that transactions are proposed based on information from that longest subchain. Note that a leader validates transactions, i.e., if it receives a transaction derived from an unknown last block, the transaction fails to validate and will not be incorporated into any block proposition.

As forks can be created, vCubeChain requires a chain adjustment mechanism. As the leader employs reliable broadcast to send each block to all processes, which confirm the receipt of the block, the leader can determine which proposed blocks are stable. A block is stable if all processes perceived as correct have confirmed the receipt of that block. For the sake of consistency, we employ a stability window of B blocks. B is a configurable parameter that can be set case by case. In other words, if a process p is the leader and has at least B confirmed blocks, its chain can be considered stable. On the other hand, some blocks will fail the validation and are not persisted. Therefore, the transactions in those blocks (or subchains) that do not persist must be retransmitted. In our case, we employed $B = 2$. Other values can be employed, the window could be larger in case of a context of more instability. A larger value implies more transactions to be redone and more contention.

The consensus strategy adopted by vCubeChain is a compromise solution based on the premise that the perception of the correct processes will converge to a single leader after eventual disputes between multiple leaders. Thus, after the convergence, the leader will be able to resolve any fork after B blocks are confirmed, thus agreement is guaranteed across the system.

For example, in Fig. 5, there is a fork due to conflicting leaders 6 and 7, which is incorrectly suspected of having failed. Leader 7 proposes blocks 2 to 5, and after process 6 also becomes a leader it proposes block 2′. After some time, process 6 realizes that process 7 is still active and gives up the leadership. Thus, after the system remains stable for long enough process 7 becomes the sole leader. As mentioned above, the leader will wait for confirmations from the correct processes to assess block stability. Stable blocks are shown in green in Fig. 5. Process 7 uses block stability information in order to compute the difference between stable subchains. In this case, the stability window



(a) p_6 suspects p_7 and considers itself the new leader

(b) p_7 is considered correct. Two leaders are running concurrently

(c) p_6 relinquishes leadership and p_7 assumes as a single leader

Fig. 4. Example of improper leadership demotion followed by correction.

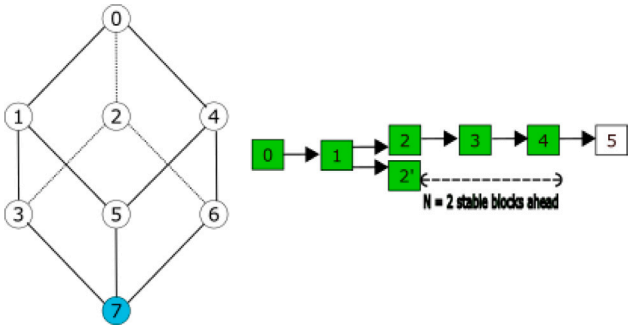


Fig. 5. A fork with two subchains with stable blocks.

is of length $B = 2$. Thus the leader determines the longest stable subchain, consisting of blocks 2 to 5. The transactions in block $2'$ are not validated and later will have to be reconciled.

Although the reliable broadcast mechanism requires a linear number of messages, in pre-GST degradation scenarios, this cost can become quadratic — that is the worst case, in which all processes are correct but unduly suspect all others. Nevertheless, from the GST the perception of the failure detector will converge, and all processes elect the same leader.

In scenarios without leader failures, the algorithm is straightforward: the leader reliably broadcasts blocks in order, which guarantees that the blockchain will work in the usual way. The failure of a leader may, given the instability of the system, raise disputes, and there may be multiple leaders during some time intervals. Even if there are forks, the processes will receive all proposed blocks from all competing leaders in the orders they established. However, as shown above, a fork is guaranteed to be reconciled after the GST, and described above.

3.4. Safety and liveness

We present draft proofs in this section that the safety of vCubeChain is guaranteed, even under timing conditions that may lead to false suspicions and multiple leaders or no leader at all. However, it is impossible to guarantee progress (i.e., termination) in all scenarios, a fact that is consistent with the FLP impossibility [22]. Progress is guaranteed only after the GST, when the leader is unique and can communicate with the other active processes in a finite time.

Lemma 3.1 (*Safety*). *All correct processes running vCubeChain eventually converge for the same blockchain, i.e., with the same blocks which include all correlated transactions.*

Discussion. vCubeChain employs a consensus strategy that is based on the autonomic reliable broadcast strategy presented. In a graceful execution scenario, only one correct leader proposes blocks that will be broadcast in a finite time to all correct processes. If there are instabilities, more than one leader may be elected, and all of them can propose new blocks based on the current chain. All those new blocks are broadcast and can generate forks that consist of multiple subchains. The reliable broadcast mechanism guarantees that all correct processes will eventually receive all blocks, including those of the subchains. After the system converges to a single leader again, this leader will assess block stability and eventually resolve any fork by determining that the longest subchain is the only valid chain. This causes all correct processes to update the corresponding blocks accordingly. □

Lemma 3.2 (*Liveness*). *All valid transactions are eventually committed across the blockchain.*

Discussion. In a graceful execution scenario, only one correct leader proposes blocks that will be broadcast in a finite time to all correct processes. In this case, liveness (i.e., termination) is guaranteed: a single leader can only propose a single sequence of blocks that result in a single chain, of which all correct processes are aware.

On the other hand, in scenarios with multiple leaders, forks can be created with multiple subchains. vCubeChain assumes the GST model, according to which any timing instabilities are solved within a finite time interval, and a new single leader is elected. The sole leader relies on block stability information to solve forks, determining a single valid chain. Transactions from deprecated subchains will be resubmitted, and deprecated blocks become invalid and are discarded. □

Theorem 3.3. *Algorithm 1 implements a permissioned blockchain that guarantees safety (i.e., correctness) and, in the presence of a sufficiently long period of timing stability, also guarantees liveness (i.e., the termination of the algorithm).*

Proof. The proof follows directly from Lemmas 3.1 and 3.2 (Termination). □

4. Simulation results

This section presents an evaluation of vCubeChain, including comparisons with Bitcoin, Ethereum, and Hyperledger Fabric. The blockchains were implemented using the Blocksim simulator [23]. Blocksim is a discrete event simulator developed in Python that allows the implementation of blockchains as "models". Both the implementations (i.e., the Blocksim models) of Bitcoin and Ethereum (permissionless blockchains) were already available. We implemented not only

Table 1
Main aspects of the evaluated solutions.

Characteristics	Bitcoin	Ethereum	Fabric	vCubeChain
Consensus	PoW	PoS	Kafka-based	this
Permissioned (P) Permissionless (L)	L	L	P	P
Nodes misbehavior	yes	yes	no	no
Finality	no	no	yes	no

vCubeChain but also Hyperledger Fabric, which was essential so that we could make a comparison with another permissioned blockchain. Blocksim's P2P network is configured with nodes (representing processes), each of which is initialized with a set of neighbors. It is also possible to configure the communication latency, both in terms of link transmission, as well as send and receive delay parameters. Next we present a brief description of the four Blocksim models employed in the evaluation.

Bitcoin. The Bitcoin [6] BlockSim model uses a block size limit of 1 megabyte and employs a probability distribution for the number of transactions per block that is based on real data from the Bitcoin network. The model provides two types of nodes: miners and non-miners. A non-mining node only validates blocks or validates and disseminates new transactions. Miners validate and group new transactions in a transaction queue to create candidate blocks that are later passed on to other nodes in the system.

Ethereum. The Ethereum [7] BlockSim model was implemented in a similar way as that of Bitcoin, also presenting mining and non-mining nodes. However, Ethereum implements the *gas limit*, which is the maximum amount of *gas* a transaction can spend (between 1 and 32,000, typically 21,000 and the *gas* limit for a block. For example, if the block has a *gas* limit of 10,000 and each transaction has a cost of 1000, each block can contain up to 10 transactions.

Fabric. We implemented the Hyperledger Fabric [3] Blocksim model using a simplified leader-based approach and a gossip protocol to broadcast transactions and blocks. For the sake of comparison with the other solutions available in the simulator, transaction validation by the endorsement policy is omitted, and we focus on the simulation of the propagation of valid transactions and blocks. Although Fabric supports multiple blockchains connected to the same ordering service (multiple channels), we simulate a one-channel scenario. The gossip protocol uses the push strategy, where each peer selects a random set of neighbors to send the messages to. A leader is elected to pull blocks and initiate the gossip distribution.

vCubeChain. The vCubeChain Blocksim model uses the same configuration as Bitcoin. However, transactions are sent only to the leader (the process with the highest ID), and the leader broadcasts blocks to all other processes using vCube's hierarchical reliable broadcast strategy.

The implementations for Bitcoin, Ethereum, and Fabric assume a fully connected network where all nodes communicate directly. The vCubeChain implementation uses the vCube virtual topology, which dynamically defines the neighbors of each node, according to the topology rules and failure detection information. For each message sent and received in the network, the processing time is computed according to the size of the corresponding messages. Blocksim simulates TCP, and the connections between processes are established as nodes first communicate.

Table 1 shows a summary of the main aspects of the four solutions (Bitcoin, Ethereum, Fabric, vCubeChain). In terms of the consensus solution employed; whether they are permissioned or permissionless; whether they allow node misbehavior, and finality (transactions cannot be rolled back).

Table 2
Locations where nodes are placed (simulation).

Sites	Non-miners	Miner/Leader
Cascavel/Brazil	$N/4$	0
Salvador/Brazil	$N/4$	0
Curitiba/Brazil	$(N/2)-1$	1

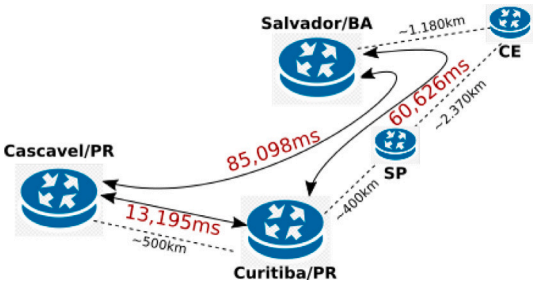


Fig. 6. Latency in milliseconds between nodes on each site (simulation).

4.1. Simulated scenarios

The algorithms were executed for systems with $N = 8, 16, 32, 64$, and 128 nodes (representing processes). The configuration parameters used were those defined in [23]. For Ethereum, the default value for the *gas limit* was 21,000 and a *block gas limit* of 2.1 million was defined. According to Table 2, the nodes are distributed across three sites. For $N = 8$, for example, there are two nodes in Cascavel/Brazil, two more in Salvador/Brazil, and four in Curitiba/Brazil. For the latter, one node is configured as a miner for both Bitcoin and Ethereum and as a leader for Fabric and vCubeChain. The number of random neighbors (i.e., the fanout) of each Fabric's gossip node was set to three, which is the default value adopted by Fabric [24].

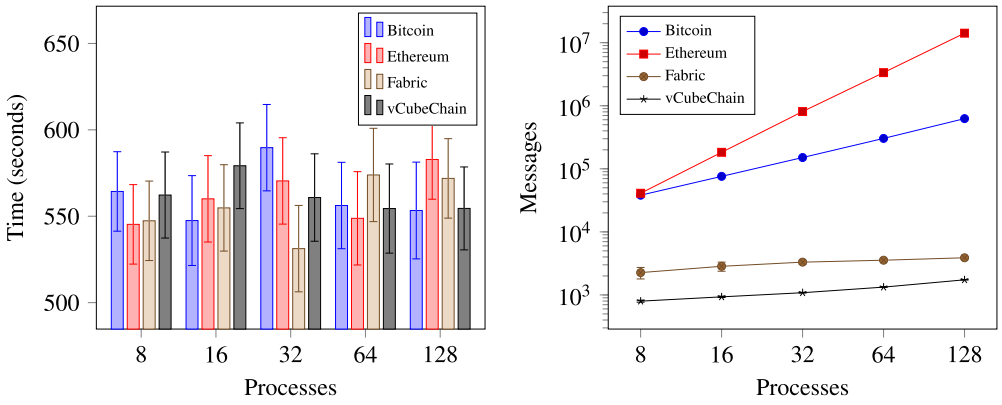
The values for the communication latency (in milliseconds) between nodes at each site were set using the parameters in Fig. 6. Those values were obtained from real RTT measurements on the RNP (National Research Network) network that connects the three sites. The normal distribution (mean and standard deviation) was used for nodes at different sites, and the inverse gamma distribution was used for nodes at the same location. The full parameters are available on Github.[1]

Results are presented for blockchain scalability, focusing on the execution times and number of messages required by the four different solutions. Simulations were performed on an Intel i7 with 512 SSD and 16 GB RAM memory. For each execution, a thousand transactions were generated, one transaction per round. A new round starts every 15 s. Transactions were randomly distributed among processes using Blocksim's transaction factory function.

4.2. Results

The average time taken to validate all transactions is shown in Fig. 7(a), computed as an average of 30 executions of each scenario and IC=95%. We assume the transaction validation times of the different models are roughly comparable. Actually, the transaction validation times were found to be similar for the four models. This uniformity across the four models led to some interesting findings. First and foremost, it highlights the importance of employing other metrics beyond time to determine the overall blockchain efficiency. With validation times being very similar, our attention shifted to resource utilization, scalability, and adaptability. In particular, resource utilization became a crucial metric, as algorithms that could achieve similar results with

[1] https://github.com/arluiz/FD-blocksim



(a) Transaction processing (b) Number of Messages

Fig. 7. Average time to validate all transactions and average number of messages in logarithmic scale for 30 executions with CI 95%.

fewer resources emerged as more sustainable and cost-effective. This aspect of the simulation allowed us to identify potential aspects of the algorithms that could possibly lead toward more environmentally friendly and economically viable solutions. Overall, we also highlight that the Ethereum simulation took a significantly longer time (several hours) to simulate than those of Bitcoin, Fabric, and vCubeChain (each just a few minutes).

The total number of messages employed by the Blocksim models is shown in logarithmic scale in Fig. 7(b). Both models that were already available in the simulator (Bitcoin and Ethereum) employ two message types, one for "header" and another for "body". For each message containing the transaction header, a request is sent for the transaction body, which in turn generates a new message containing the transaction content. The same is true for blocks. To ensure a fair comparison, we implemented Fabric and vCubeChain in exactly the same way.

In the case of vCubeChain, the number of messages employed keeps small due to the hierarchical broadcast strategy employed. On the hand, Fabric uses a three-neighbor gossip protocol. In the case of Bitcoin and Ethereum, the transmission is one-to-one. Moreover, processes in Bitcoin send initialization messages as they start communicating to spread their knowledge about new transactions and blocks.

In terms of the number of blocks created, for Ethereum, the number of transactions per block is determined by the *gas limit*. For Bitcoin and VCubeChain, A block size of 2 megabytes was employed. Therefore, all transactions are combined into a single block in all scenarios.

In a real scenario, considering the TCP connections in an error-free execution, vCubeChain opens fewer connections compared to Bitcoin, Ethereum and Fabric because it communicates only with the leader and $log_2 N$ neighbors in average and up to N neighbors in worst-case, in which a single process is correct. Bitcoin and Ethereum use a one-to-all strategy, i.e. there are $(N^2 - N)/2$ connections. In the Fabric implementation, each process communicates with 3 random neighbors, so the number of connections can vary from $3 * N$ to $(N^2 - N)/2$ depending on the selection of neighbors.

5. Related work

This section presents an overview of related work, including both permissionless and permissioned blockchains. Several permissionless blockchains use probabilistic approaches to consensus, such as Bitcoin's proof-of-work (PoW) [6]. That approach is actually an implicit leader election. In PoW, multiple processes compete to solve the cryptographic challenge and then submit a block proposal for the blockchain. Multiple processes may succeed, which results in a divergence expressed by a *fork*. To bring the blockchain back to a consistent state, Bitcoin adopts the rule of the survival of the longest chain, similar to vCubeChain's approach.

The so-called PoX (Proof-of-Anything) consensus mechanisms have been proposed with the purpose of reducing the computational cost of PoW [25]. Arguably, the most successful PoX strategy is Proof-of-Stake (PoS), used in platforms such as Ethereum [26] and Algorand [8]. PoS consensus mechanisms address inefficiencies inherent in conventional Proof-of-Work (PoW) protocols. Instead of relying on crypto mining, in PoS blockchains a leader is elected based on its stake, in terms of tokens or cryptocurrency. The leader verifies and records transactions. Nodes make stakes in a staking pool, the node with the highest stake is elected as validator/leader for the next proposal slot. In that proposal slot, that node (i.e., the leader) proposes a new block that a quorum must approve to be added to the blockchain. The quorum is also defined based on the stakes. As it makes new proposals, the leader earns rewards which are newly generated tokens. PoS is based on game theory: participants who have a higher amount of tokens for a longer time have higher chances to be elected as leaders and to the deciding quorum.

PoS does not prevent the occurrence of forks. The so-called multiple honest slots may arise by design [27]: they are proposal slots that emerge when two or more honest nodes are elected as leaders. That may occur for instance during network instabilities when the system falls into a non-synchronous timing situation that may result in asymmetries and cause more than one process to assume the leader role. Again, there are strategies to bring the blockchain back to consistency. Ethereum 1.0 [7] also employs the longest chain rule, like Bitcoin. However, Ethereum 2.0 [10] runs a fork-choice algorithm that measures the 'weight' of the multiple chains, using a variation of the GHOST fork rule [28].

Some hybrid approaches combine PoW and PoS or use slightly modified approaches [25]. For instance, the Proof-of-Activity (PoA) protocol employs PoW to create empty blocks and the PoS to verify blocks and add transactions [29]. Other hybrid approaches, such as the protocol proposed in [30], often elect a committee to verify blocks and confirm transactions. Those approaches try to minimize power consumption but employ game theory concepts of permissionless blockchains.

All those approaches based on PoW and PoS, and other PoX and hybrid approaches heavily rely on cryptocurrency or token rewards. Similar to those strategies of permissionless blockchains, vCubeChain can also cope with multiple leaders, and uses rules to reconcile the blockchain eventually.

In the case of permissioned blockchains, the platforms usually encapsulate in their core a traditional consensus approach, as in the case of Hyperledger Fabric [3] or of Corda [4] that allows an arbitrary consensus mechanism to be plugged.

Traditional approaches to consensus, such as Paxos [31] and Raft [11], explicitly use leader election. In Paxos, for example, the proposer

tries to become the leader in the first phase and get the proposal accepted. Alternatively, the proposer may fail to gain a majority and try later, or another leader may be elected. Conversely, Raft clearly distinguishes the leader election process, which is explicitly executed by the protocol.

Both the Paxos strategy (based on a majority of acceptors) and the Raft strategy (based on *heartbeat* message exchanges and timeouts) may experience performance degradations as the number of processes grows. Furthermore, despite of having the advantage of predictability, those deterministic strategies imply on a higher communication cost. On the other hand, probabilistic algorithms, despite of being more efficient in terms of communication, they can also present other challenges, such as the high computational cost of the proof-of-work approach.

Some Paxos approaches, such as Fast Paxos [32] and Multi-Paxos [33], assume a speculative strategy, which is efficient in graceful executions. In Fast Paxos, different values may be accepted in a fast round, i.e., it is prone to collisions, and a value may not be decided. In that case, the algorithm turns back to the classic version. In Multi-Paxos a distinguished replica is elected as the protocol leader to improve system performance by reducing the 2-phase commit protocol of classic Paxos to a 1-phase commit protocol for n consensus rounds. However, the protocol degrades to the classic version in a collision scenario in which two or more proposers compete for the leadership.

vCubeChain presents greater scalability even in the degraded scenario, as it employs the autonomic and hierarchical reliable broadcast strategy that does allow forks to occur but it will bring the system back to consistency once it becomes stable. Recall that vCubeChain relies on the underlying vCube both as a failure detector and for leader election. This strategy avoids contention in a synchronous network: the leader is the correct process with the highest identifier. If the network behaves asynchronously, multiple leaders can be elected: forks can appear, which are resolved a posteriori. This solution seeks to reduce contention in the permissioned blockchain — as occurs in consensus algorithms generally used in this environment, which also increases vCubeChain's scalability.

RingPaxos [34] uses a logical ring as an overlay network for saving messages in phase 1. PigPaxos [35] uses a piggy-backing approach to save messages and employs relays to reduce leader load. In comparison, our approach is also message savvy, but by using the vCube's autonomic spanning trees.

Other consensus approaches address Byzantine faults, such as PBFT [12], and Zyzzyva [36]. Currently, vCubeChain assumes crash faults.

6. Conclusion

This work presented vCubeChain, a hierarchical and autonomic permissioned blockchain. vCubeChain relies on vCube's failure detection and reliable broadcast mechanisms. A leader is elected based on failure detection information, the leader is responsible for making proposals for new blocks. During a period of instability, multiple leaders can be elected and forks can appear, but the system returns to a consistent state as soon as network conditions improve. vCubeChain was implemented through simulation and compared with Bitcoin and Ethereum. Results demonstrate the scalability of the solution.

Future work includes the investigation of vCubeChain in the context of Byzantine faults. Other future work should define strategies to deal with crash-recovery and dynamic processes, that join and leave the system over time. Currently, vCubeChain implements a complete stack blockchain, but we envision that specific parts of the proposed approach may be adapted as a pluggable module for other existing blockchain platforms.

CRediT authorship contribution statement

Allan Edgard Silva Freitas: Writing – original draft, Investigation, Formal analysis, Conceptualization. **Luiz Antonio Rodrigues:** Writing – original draft, Investigation, Data curation, Conceptualization. **Elias Procópio Duarte Jr.:** Writing – original draft, Validation, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was partially supported by the Brazilian Research Council (CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico) grant 308959/2020-5; FAPESP/MCTIC/CGI grant 2021/06923-0; and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] H.T.M. Gamage, H. Weerasinghe, N.G.J. Dias, A survey on blockchain technology concepts, applications, and issues, SN Comput. Sci. 1 (2020) 1–15, http://dx.doi.org/10.1007/s42979-020-00123-0.

[2] W. Chen, Z. Xu, S. Shi, Y. Zhao, J. Zhao, A survey of blockchain applications in different domains, in: Proc. 2018 Int'L Conf. on Blockchain Technology and Application, ICBTA '18, ACM, New York, NY, USA, 2018, pp. 17–21, http://dx.doi.org/10.1145/3301403.3301407.

[3] E. Androulaki, et al., Hyperledger fabric: A distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference, EuroSys '18, Association for Computing Machinery, New York, NY, USA, 2018, http://dx.doi.org/10.1145/3190508.3190538.

[4] R.G. Brown, J. Carlyle, I. Grigg, M. Hearn, Corda: an introduction, 2016, p. 15, URL https://docs.r3.com/en/pdf/corda-introductory-whitepaper.pdf.

[5] A.A. Monrat, O. Schelén, K. Andersson, Performance evaluation of permissioned blockchain platforms, in: 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, CSDE, 2020, pp. 1–8, http://dx.doi.org/10.1109/CSDE50874.2020.9411380.

[6] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Citeseer, 2008, URL http://bitcoin.org/bitcoin.pdf.

[7] D.D. Wood, Ethereum: a secure decentralised generalised transaction ledger, 2014, URL https://ethereum.github.io/yellowpaper/paper.pdf.

[8] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, N. Zeldovich, Algorand: Scaling Byzantine agreements for cryptocurrencies, in: Proceedings of the 26th Symposium on Operating Systems Principles, SOSP '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 51–68, http://dx.doi.org/10.1145/3132747.3132757.

[9] L. Peng, W. Feng, Z. Yan, Y. Li, X. Zhou, S. Shimizu, Privacy preservation in permissionless blockchain: A survey, Digit. Commun. Netw. 7 (3) (2021) 295–307, http://dx.doi.org/10.1016/j.dcan.2020.05.008.

[10] Ethereum.org, PROOF-OF-STAKE (POS), 2022, https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/.

[11] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference, USENIX ATC'14, USENIX Association, USA, 2014, pp. 305–320, URL https://dl.acm.org/doi/10.5555/2643634.2643666.

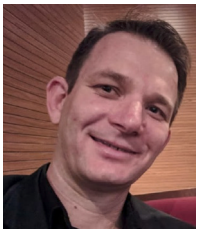
[12] M. Castro, B. Liskov, Practical Byzantine fault tolerance and proactive recovery, ACM Trans. Comput. Syst. 20 (4) (2002) 398–461, http://dx.doi.org/10.1145/571637.571640.

[13] D. Dolev, C. Lenzen, Early-deciding consensus is expensive, in: Proc. of the PODC '13, ACM, New York, NY, USA, 2013, pp. 270–279, http://dx.doi.org/10.1145/2484239.2484269.

[14] M. Vukolić, The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication, in: J. Camenisch, D. Kesdoğan (Eds.), Open Problems in Network Security, Springer International Publishing, Cham, 2016, pp. 112–125, http://dx.doi.org/10.1007/978-3-319-39028-4_9.

[15] R. Guerraoui, J. Hamza, D.-A. Seredinschi, M. Vukolic, Can 100 machines agree?, 2019, http://dx.doi.org/10.48550/ARXIV.1911.07966, https://arxiv.org/abs/1911.07966.
[16] E.P. Duarte, L.C.E. Bona, V.K. Ruoso, VCube: A provably scalable distributed diagnosis algorithm, in: 2014 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems, 2014, pp. 17–22, http://dx.doi.org/10.1109/ScalA.2014.14.
[17] E. Duarte, T. Nanya, A hierarchical adaptive distributed system-level diagnosis algorithm, IEEE Trans. Comput. 47 (1) (1998) 34–45, http://dx.doi.org/10.1109/12.656078.
[18] E.P. Duarte Jr., L.A. Rodrigues, E.T. Camargo, R.C. Turchetti, The missing piece: a distributed system-level diagnosis model for the implementation of unreliable failure detectors, Computing 105 (2023) http://dx.doi.org/10.1007/s00607-023-01211-8.
[19] D. Dolev, C. Dwork, L. Stockmeyer, On the minimal synchronism needed for distributed consensus, J. ACM 34 (1) (1987) 77–97, http://dx.doi.org/10.1145/7531.7533.
[20] L.A. Rodrigues, L. Arantes, E.P. Duarte Jr., An autonomic implementation of reliable broadcast based on dynamic spanning trees, in: 2014 Tenth European Dependable Computing Conference, 2014, pp. 1–12, http://dx.doi.org/10.1109/EDCC.2014.31.
[21] D. Jeanneau, et al., An autonomic hierarchical reliable broadcast protocol for asynchronous distributed systems with failure detection, JBCS 23 (15) (2017) http://dx.doi.org/10.1186/s13173-017-0064-9.
[22] M.J. Fischer, N.A. Lynch, M.S. Paterson, Impossibility of distributed consensus with one faulty process, J. ACM 32 (2) (1985) 374–382.
[23] C. Faria, M. Correia, BlockSim: Blockchain simulator, in: 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 439–446, http://dx.doi.org/10.1109/Blockchain.2019.00067.
[24] N. Berendea, H. Mercier, E. Onica, E. Riviere, Fair and efficient gossip in hyperledger fabric, in: 2020 IEEE 40th International Conference on Distributed Computing Systems, ICDCS, IEEE Computer Society, Los Alamitos, CA, USA, 2020, pp. 190–200, http://dx.doi.org/10.1109/ICDCS47774.2020.00027, URL https://doi.ieeecomputersociety.org/10.1109/ICDCS47774.2020.00027.
[25] C.T. Nguyen, D.T. Hoang, D.N. Nguyen, D. Niyato, H.T. Nguyen, E. Dutkiewicz, Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities, IEEE Access 7 (2019) 85727–85745, http://dx.doi.org/10.1109/ACCESS.2019.2925010.
[26] F. Saleh, Blockchain without waste: Proof-of-stake, Rev. Financ. Stud. 34 (3) (2020) 1156–1190, http://dx.doi.org/10.1093/rfs/hhaa075.
[27] A. Kiayias, S. Quader, A. Russell, Consistency of proof-of-stake blockchains with concurrent honest slot leaders, in: 2020 IEEE 40th International Conference on Distributed Computing Systems, ICDCS, 2020, pp. 776–786, http://dx.doi.org/10.1109/ICDCS47774.2020.00065.
[28] Y. Sompolinsky, A. Zohar, Secure high-rate transaction processing in bitcoin, in: R. Böhme, T. Okamoto (Eds.), Financial Cryptography and Data Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 507–527, http://dx.doi.org/10.1007/978-3-662-47854-7_32.
[29] I. Bentov, C. Lee, A. Mizrahi, M. Rosenfeld, Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y, ACM SIGMETRICS Perform. Eval. Rev. 42 (3) (2014) 34–37.
[30] R. Pass, E. Shi, Hybrid Consensus: Efficient Consensus in the Permissionless Model, in: A. Richa (Ed.), 31st International Symposium on Distributed Computing (DISC 2017), in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 91 (2017) 39:1–39:16, http://dx.doi.org/10.4230/LIPIcs.DISC.2017.39.
[31] L. Lamport, The part-time parliament, ACM Trans. Comput. Syst. 16 (2) (1998) 133–169, http://dx.doi.org/10.1145/279227.279229.
[32] L. Lamport, Fast paxos, Distrib. Comput. 19 (2) (2006) 79–103, http://dx.doi.org/10.1007/s00446-006-0005-x.
[33] R. Van Renesse, D. Altinbuken, Paxos made moderately complex, ACM Comput. Surv. 47 (3) (2015) http://dx.doi.org/10.1145/2673577.
[34] P.J. Marandi, M. Primi, N. Schiper, F. Pedone, Ring paxos: A high-throughput atomic broadcast protocol, in: 2010 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN, 2010, pp. 527–536, http://dx.doi.org/10.1109/DSN.2010.5544272.
[35] A. Charapko, A. Ailijiang, M. Demirbas, PigPaxos: Devouring the communication bottlenecks in distributed consensus, in: Proceedings of the 2021 International Conference on Management of Data, SIGMOD '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 235–247, http://dx.doi.org/10.1145/3448016.3452834.
[36] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, E. Wong, Zyzzyva: speculative byzantine fault tolerance, in: Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles, 2007, pp. 45–58.

Allan Edgard Silva Freitas is a Full Professor at Federal Institute of Bahia (IFBA), Brazil. He received his Ph.D. in Computer Science from the Federal University of Bahia (UFBA). He is an effective member of the Brazilian Computer Society (SBC) and the Association for Computing Machinery (ACM). He served as chair of several conferences and workshops, including Brazilian Symposium on Computer Networks and Distributed Systems (SBRC'2016). He is the vice-coordinator of the Special Committee on Fault-Tolerant Systems of the SBC (CE-TF). His main interests are distributed systems, computer networks, fault tolerance, and Dependability.



Luiz Antonio Rodrigues is a Full professor at Western Parana State University (Unioeste), Brazil, and a member of the GPISC (Computer Systems Research and Innovation Group). He received his Ph.D. in Computer Science from the Federal University of Parana (UFPR) including a year-long internship int the LIP6 at Sorbonne University. He is an effective member of the Brazilian Computer Society (SBC) and was the coordinator of the Special Committee on Fault-Tolerant Systems of the SBC (CE-TF 2019–2020). His main interests are in computer science, with a focus on computer networks, fault tolerance, distributed systems, and Dependability.

Elias P. Duarte Jr. is a Full Professor at Federal University of Parana, Brazil. His research interests include computer networks and distributed systems, their dependability and algorithms. With over 300 peer-reviewer papers, and 130 students supervised, Prof. Duarte is Associate Editor of the Computing (Springer) journal and IEEE Transactions on Dependable and Secure Computing, and has served as chair of more than 25 conferences and workshops, including TPC Chair of GLOBECOM'2024, SRDS'2018 and ICDCS'2021. He chaired the Brazilian National Laboratory on Computer Networks (2012–2016), and is a member of the Brazilian Computer Society and a Senior Member of the IEEE.