Blockchain: Research and Applications 3 (2022) 100026

Contents lists available at ScienceDirect

Blockchain: Research and Applications

journal homepage: www.journals.elsevier.com/blockchain-research-and-applications

Performance evaluation of permissioned blockchains for financial applications: The ConsenSys Quorum case study

Marco Mazzoni [a,*], Antonio Corradi [b], Vincenzo Di Nicola [a]

[a] Conio Inc., c/o The Orrick Building, 405 Howard Street, San Francisco, CA, 94105, USA
[b] Department of Computer Science and Engineering, University of Bologna, Viale Risorgimento 2, Bologna, Italy

ARTICLE INFO

Keywords:
Blockchain
Consensus
Benchmark
Quorum
Cryptocurrency

ABSTRACT

Given the availability of several blockchain technologies in permissioned contexts, blockchain application designers have to cope with the increasing complexity of choosing which technology and consensus algorithm best fit a specific use case. However, the lack of a standard framework allowing to assess the scalability of permissioned blockchain platforms and to compare performances and features of consensus algorithms makes the development of a sensible evaluation a costly and difficult time-consuming problem. Throughout this article, we propose a practical scalability and applicability evaluation of the Quorum blockchain and its consensus algorithms. Although we apply our evaluation workflow to a financial use case, we define a methodology that can be generalized to any permissioned blockchain technology. We leverage Hyperledger Caliper as a benchmarking tool, and Docker as a deployment tool, making our analysis easy to be repeated, cross-platform, and cost-effective.

1. Introduction

Over the past decade, with the rise and growth of Bitcoin and Ethereum, more and more companies, from FinTech to the retail industry, are more than willing to integrate blockchain-based solutions within their application portfolios. Accordingly, permissioned blockchains have been developed to satisfy the need to create closed consortia and private networks among a restricted number of companies and institutions. This allows to integrate the advantages of blockchain technologies, while keeping performances and privacy of data that permissionless blockchains may lack. In the heterogeneous and fast-growing environment of permissioned blockchains, it is becoming harder and harder to mark precisely and distinguish clearly the differences between features provided by any particular technology. Moreover, many voting-based consensus protocols are being proposed to replace permissionless consensus algorithms. Therefore, deciding which technology best fits a specific use case is a complex task: guarantees and performances of each blockchain platform need to be deeply explored, and a thorough analysis is challenging to perform.

Throughout this work, we illustrate a methodology to carry out an accurate analysis of a permissioned blockchain framework. The main contributions brought by our methodology are summarized as follows: fill the gap between theoretical analysis of consensus algorithms and practical scalability evaluation; systematization of tools and techniques to accomplish performance measurements; fully open-source approach (developed extensions for measurement tools are publicly available); use of standard containerization tools to deploy permissioned blockchain networks in order to dramatically drop setup times and increase interoperability. Therefore, by following our workflow, a blockchain application designer can achieve significant insight into the performances of a specific platform, by leveraging open-source software technologies and common hardware infrastructures, so to make costs of such analysis very low. As a case study, we selected the ConsenSys Quorum blockchain, since it introduces the possibility of executing private transactions between participants, and it offers several plug-and-play consensus protocols the blockchain can be run with, thus enabling a clear comparative analysis.

Section 2 gives an overview of Quorum architecture, focusing on the privacy feature. Then, in Section 3 we show how a pragmatic analysis of multiple consensus algorithms can be developed, based on their formal specifications, by outlining the main aspects to be examined and focused. In Section 4, we present our methodology, and then we describe metrics, workloads, and tools used for our experiments. In Section 5 outcomes are discussed and conclusions are delivered.

* Corresponding author.
E-mail addresses: mazzoni@conio.com (M. Mazzoni), antonio.corradi@unibo.it (A. Corradi), vincenzo@conio.com (V. Di Nicola).

https://doi.org/10.1016/j.bcra.2021.100026
Received 30 December 2020; Received in revised form 14 June 2021; Accepted 10 August 2021

2. Quorum features

Quorum [1] is an Ethereum-based [2] distributed ledger protocol, that enables the creation of a permissioned blockchain network with support to transaction privacy. The open-source project has been initially developed by J.P. Morgan Chase, with a focus on financial use cases; recently, it has been acquired by ConsenSys. Quorum has been introduced in the broad ecosystem of permissioned blockchains to meet the following needs:

- Execute private transactions and smart contract operations.
- Adopt multiple consensus mechanisms in a plug-and-play fashion.
- Provide flexible and expressive network permissions management.

Quorum is built after the official Go implementation of the Ethereum protocol (`geth`): the rationale behind this choice is that, by minimizing the changes required to Go-Ethereum, continuity with future versions of the public Ethereum code base is easily achievable. Each Quorum node consists of two main services:

1. **Quorum Client**: it is the extended `geth` client, which is responsible for executing the Ethereum peer to peer (p2p) protocol (by accepting connections only from participants to the permissioned network), and the consensus algorithm (see Section 3).
2. **Privacy Manager**: it is a software module that enables private transactions and private smart contract operations. It consists of two components: the Transaction Manager, and the Crypto Enclave (see Section 2.1 for more details).

Fig. 1 illustrates the architecture of a 3-nodes Quorum network, outlining the interactions between services and components of each node.

2.1. Transaction privacy

The main idea behind the introduction of the Quorum transaction privacy feature is to leverage cryptographic operations provided by the Privacy Manager. The payload of private transactions goes through an encryption process [3], so that only nodes specified in the transaction are allowed to access payload information. Accordingly, the block validation process is modified with respect to classical Ethereum, such that every node validates all public transactions and any private transactions they are allowed to, by executing the contract code associated with the transactions. What is more, a node that is not permitted to a specific private transaction will skip the execution of the contract code associated with that transaction.

Furthermore, it is worth noting that (as shown in Fig. 1) public transactions are handled by the Quorum Client, whereas private transactions need to be processed also by the Privacy Manager, which implies a higher computational overhead.

The default Privacy Manager adopted in the latest versions of Quorum is Tessera [4]: it is implemented in Java, and it is used to enable the encryption, decryption, and distribution of private transactions. The Tessera Transaction Manager is the central actor during private transactions processing: it interfaces with the other entities of the infrastructure, such as its corresponding Quorum Client and the other Tessera nodes, and manages the lifecycle of private data. More precisely, it accomplishes the following tasks:

- Self-manages and discovers all nodes in the network, by establishing p2p connections with the other nodes Transaction Managers, and by broadcasting peer/key information.
- Communicates with the Enclave for encrypting/decrypting private transaction payloads.
- Maintains a local off-chain database to store encrypted payload data.

The Crypto Enclave, instead, is designed to provide the encryption/decryption operations required by the Transaction Manager, as well as the key management. That enables all sensitive operations to be handled in full isolation, so preventing any leakage into external areas of program memory. Thus, the Transaction Manager, which handles peer management and database access, does not perform any encryption/decryption, greatly reducing the impact an attack can have.

Given the inherent duality of private and public transactions, a segmentation of the state database is implemented: all Quorum nodes maintain a local unique private state, made up of private transactions, and share a common public state, created through public transactions. In Ethereum, the blockchain state of each node is stored in special tree structures called tries (Modified Merkle Patricia Tries) [5]. Therefore, Quorum stores the state of public contracts in a public state trie that is globally synchronized, while the state of private contracts is stored in a private state trie, that is not synchronized globally. However, each node is able to verify it has the same set of transactions as the other participants, since the block validation process also includes a check of the `global Transaction hash`, namely, the hash of all transactions in a block, both public and private ones [6].

3. Consensus protocols

Before diving into performance measurements of a permissioned blockchain technology, it is crucial for an application designer to have a

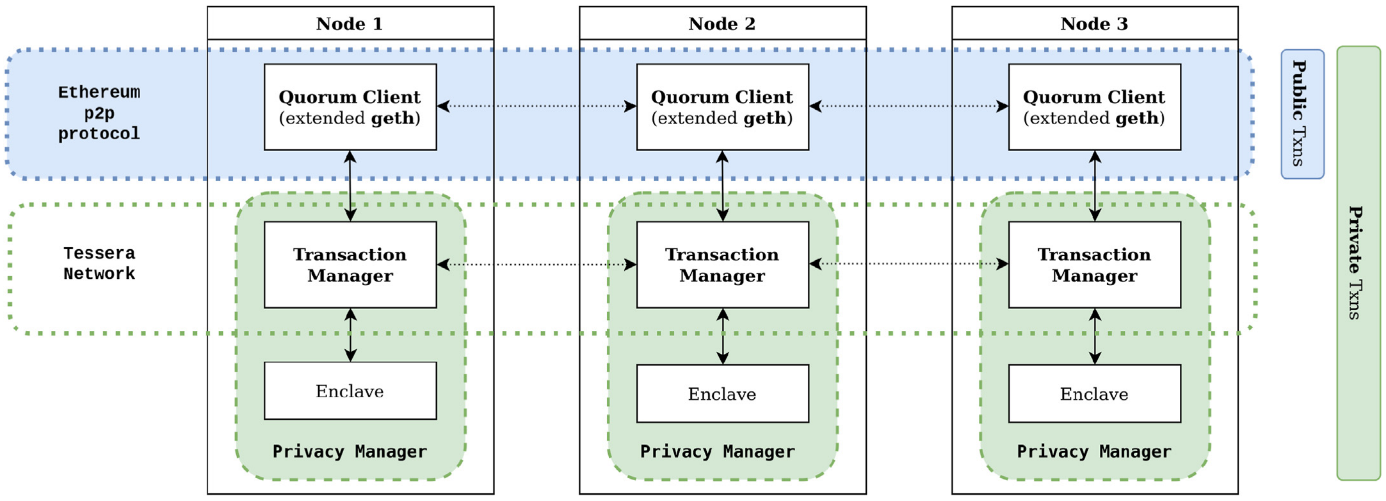


Fig. 1. Interaction schema of a Quorum network.

clear overview of the features and intrinsic differences of each consensus protocol provided by the platform. In this section, we develop a comparative analysis based on the formal specifications of consensus algorithms offered by Quorum. In order to ease the comparison process, we suggest exploring and focusing on the following fundamental features of each protocol:

- Fault-tolerance degree;
- Details of the leader election sub-protocol;
- Communication complexity during normal operation;
- Number of message rounds and block proposers per step;
- Quorum size;
- Consensus finality and the possibility of forks;
- Consistency guarantees with respect to the CAP Theorem [7].

More precisely, an analysis based on the CAP Theorem allows to better understand which properties among strong Consistency (C), Availability (A), and Partition-tolerance (P) a specific distributed system trades off.

A Quorum network can be deployed with three different consensus algorithms: Raft [8], Clique Proof-of-Authority (PoA) [9], and Istanbul Byzantine fault-tolerant (IBFT) [10]. Since permissioned blockchains are characterized by a strict node identity management and a relatively low number of participants, unlike permissionless environments, there is no need for cryptoeconomics consensus mechanisms (e.g., Proof-of-Work (PoW), Proof-of-Stake (PoS)) to keep the network safe. Thus, mining (PoW) or minting (PoS) activities related to incentives, are useless in permissioned blockchains: blocks are validated by an a priori selected group of authorities, hence any misbehavior can be detected, and suspected malicious nodes can be voted out.

3.1. Raft

Raft [8] is a distributed consensus algorithm based on Paxos [11]. It operates under the Crash failure model, where every node failure (e.g., message omissions, network partitions, or simply hardware failures) is assumed to be a node crash: hence, $n \geq 2f+1$ must hold, where n is the total number of nodes, and f is the maximum number of nodes that can exhibit crash failures. Raft works by first electing a leader in the group of nodes, then giving the leader complete responsibility for accepting transaction requests and managing the replication of the logs (i.e., blocks) on the other nodes. Every node can be in one of the three states: follower, candidate, or leader. It is important to point out that the leader election process is deterministic, thus the algorithm cannot make progress until a leader has been chosen by the majority of nodes; moreover, the leader, which acts as a block proposer, can remain the same for many rounds of the protocol execution, as long as a strict majority of followers receive periodic heartbeat messages from it (hence we refer to it as a strong leader).

Raft servers communicate via two types of remote procedure calls (RPCs): the `RequestVotes` RPC is used by candidates during elections to collect votes; the `AppendEntries` RPC is used by leaders to replicate log entries. A block is considered committed by the leader, whenever the `AppendEntries` RPC returns successfully from the majority of servers, thus as soon as the entry is replicated onto at least $f+1$ nodes (quorum size). Since the leader, during normal protocol operation, broadcasts n `AppendEntries` messages, and waits for at least f replies, Raft has an overall linear communication complexity, with two message exchanges per protocol step.

From a CAP Theorem [7] perspective, Raft can be considered as a C–P system: in fact, it guarantees partition tolerance (P), while providing strong consistency (C). When a network partition happens in a Raft cluster, only the partition with a majority of nodes can commit and execute client operations. The servers in the minority partitions are not able to commit client operations, should they be read or write operations: they simply do not reply to client requests, hence availability is sacrificed in favor of strong consistency. Therefore, Raft guarantees instant consensus finality [12]: as soon as the leader committed a block to the blockchain, all transactions included in the block are immediately confirmed, and the block cannot be eventually discarded. Hence, Raft prevents transient forks from occurring.

3.2. Clique PoA

Clique PoA [9] is a Proof-of-Authority consensus algorithm, purposely designed for enabling private networks within the Ethereum platform: it emulates the design of the Ethereum mainnet PoW consensus, so that every client can be extended with this consensus protocol with minimal effort. It is based on the idea that blocks may only be minted by a set of trusted signers called authorities: each authority is identified by a unique id, and a strict majority (i.e., $n/2+1$, out of n total trusted nodes) of authorities is assumed to behave honestly and follow the protocol.

In order for Clique to be a Byzantine fault-tolerant (BFT) protocol (e.g., avoid spamming attacks), the solution devised suggests that each authority is allowed to propose a block every $n/2+1$ blocks, where n is the number of nodes. Thus, at any time, there are at most $n-(n/2+1)$ nodes allowed to propose a block, and the subset of proposers rotate through the whole set of authorities. If a node exhibits arbitrary behaviors, it can be voted out from the authorities list: a vote can be expressed by each node at each epoch, and if a quorum of votes is reached, the faulty node is removed from the list (so that it can no longer affect the network).

Despite tolerating Byzantine failures, this algorithm is generally more efficient in terms of communication complexity with respect to traditional BFT protocols (such as practical Byzantine fault-tolerant (PBFT) [13] or IBFT [10]), because it only guarantees eventual consensus finality: it adopts an eager strategy, according to which the block proposers broadcast the block to the other authorities, and they update their copy of the blockchain accordingly (as in PoW algorithms). Since there can be multiple proposers at each protocol step, the possibility of blocks being concurrently appended in different orders by different nodes is contemplated: thus, transient forks can occur, but they are resolved afterwards by specific reconciliation protocols (e.g., GHOST [14]). Therefore, after analyzing Clique under the CAP properties, it can be inferred that it is an A-P protocol: it trades off strong consistency for availability (A), while being partition-tolerant (P). Since blockchain nodes may have different views of the chain due to transient forks, and since inconsistencies are later resolved, Clique satisfies the eventual consistency property.

3.3. IBFT

IBFT [10] is a Byzantine fault-tolerant protocol based on PBFT by Castro and Liskov [13]. Byzantine consensus is deterministically solved: first, a leader or proposer is elected (as in Raft), then each proposed block undergoes multiple communication phases between nodes before being committed to the blockchain. Four types of messages are exchanged among nodes:

(1) `PRE-PREPARE`, `PREPARE`, `COMMIT`: are used during normal operation of the algorithm.
(2) `ROUND-CHANGE` is used to choose a new leader when the current one is suspected to have failed or communication is not timely.

Currently, the practical implementation of IBFT adopted by Quorum [15] offers two leader election strategies:

- **Round-robin**: this is the default leader election strategy, in which a different proposer is selected for each block decision step.
- **Sticky proposer**: a new proposer is chosen only when a round-change is started, thus whenever a node is suspected of behaving maliciously.

In both cases, each node is aware of who will be the next block proposer, since it is decided by means of a deterministic calculation based on node IDs.

As PBFT, also IBFT makes sure there is only one proposer for each round; moreover, the proposer needs a quorum of responses by other servers in order to make any progress. This implies that in the case of a network partition which involves more than f nodes (over at least $3f+1$ nodes), the protocol does not commit any decision until the partition is resolved and communication becomes timely again (thus, consensus is not violated during partitions). For this reason, we can assert that, from a CAP point of view, IBFT is C–P: it guarantees strong consistency (C) and partition tolerance (P). Therefore, as in Raft, forks cannot occur and instant finality is ensured.

3.4. Expectations

Characteristics of each consensus protocol are clearly organized in Table 1, so to allow a preliminary evaluation of their performances. Since Raft is a Crash fault-tolerant (CFT) protocol, we expect it to perform better than the other two (which instead are BFT protocols): the reason is to be searched in the failure model they cope with and, accordingly, in their communication complexity. It is worth reminding that, although Clique and IBFT are both resilient to Byzantine failures, the former is a probabilistic protocol and only provides eventual finality (i.e., weak consistency), whereas the latter guarantees instant finality (i.e., strong consistency). Thus, due to Clique having linear communication complexity, we expect its performance measurements to be closer to Raft. Hence, at first glance, we would expect IBFT to be outperformed by Clique, especially in larger networks, given its pessimistic approach to consistency—in Clique fork conflicts are resolved rather than prevented—and its quadratic complexity. However, IBFT deterministic leader election and instant finality could result in better transaction processing time than Clique, particularly in small networks. Furthermore, it would be interesting to compare the round-robin and sticky proposer election strategies within IBFT, to see whether there are relevant differences between the two.

4. Performance analysis

In order for a blockchain application designer to have a complete insight into the performances of a specific blockchain framework, it is necessary to connect and transform the comparative analysis developed on the formal specifications of consensus algorithms (Section 3) into tangible measurements and experimental results. As follows, we illustrate the methodology we developed for merging theoretical and practical results, and then we describe metrics (Section 4.2), tools (Section 4.3), and workloads (Section 4.4) useful to evaluate the performances and scalability of a permissioned blockchain.

Table 1
Overview of Quorum consensus protocols.

	Raft	Clique PoA	IBFT
Fault-tolerance	CFT $n \geq 2f+1$	BFT $n \geq 2f+1$	BFT $n \geq 3f+1$
Leader election	Deterministic strong leader	Probabilistic weak leader	Deterministic strong/Weak leader
Communication complexity	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$
#Message rounds per step	2	1	3
Quorum size	$f+1$	No quorum	$\left\lfloor \frac{n+f}{2} \right\rfloor + 1$
#Block proposers per step	1	$n-(\frac{n}{2}+1)$	1
Consensus finality	Instant	Eventual	Instant
Forks	No	Yes	No
CAP analysis	C–P	A–P	C–P

Note: During normal operation. PoA: Proof-of-Authority; IBFT: Istanbul Byzantine fault-tolerant.

4.1. Methodology

Our main objective is to show the methods we have defined to analyze a blockchain technology, so that any interested people can follow the details of it, for replication's sake. The algorithm in steps provides a complete and clear insight of a blockchain framework, starting from the theoretical foundations of the consensus layer, to the performances it is capable of under specific workloads and topology settings. Moreover, the fundamental goal of our workflow was its easy applicability to any blockchain platform. Our methodology involves two phases, the macro steps, and the micro ones. The macro-level describes how to drive the study of the blockchain framework and how to relate it with the scalability of its main features. The micro-level describes in a closer view how to operationally setup and perform benchmarking analysis on performances, by focusing on specific metrics.

The macro-level consists of the following steps:

(1) Study of the blockchain framework's main features. For Quorum, we focused on the privacy properties of transactions, which is the main breakthrough.
(2) Analysis of the consensus algorithm (or algorithms, if more) based on official documentation.
(3) Draw a summary of the analysis made in step (2), by focusing on fundamental criteria for evaluation of consensus protocols as listed in Section 3.
   - If multiple consensus algorithms are available within a specific framework, a comparative analysis as shown in Table 1 could help.
(4) Choice of benchmarking tools to perform measurements of the blockchain performances: refer to the available tools listed in Section 4.3.
   - In case the chosen tool does not support the blockchain framework of interest, specific extensions is implemented.
(5) Choice (or development, if not available) of deployment tools to run the blockchain network with a configurable number of nodes.
(6) Measurement of performances targeting transactions per second (TPS) and Latency metrics (see Section 4.2).
(7) Elaboration of results and comparison with theoretical analysis of steps (1) and (2).

The micro-level methodology is a finer grain detail of the above step (6) and it is in charge of defining how to perform effectively scalability tests on the blockchain network. It consists of several decisions that depend on the use case:

- Choice of the network topologies for the tests, typically starting with the smallest possible number of nodes, up to the network size required.
- Choice of the type of workloads: for instance, we choose the Smallbank (see Section 4.4) workloads because of the interest in financial applications.
- Choice of the different transaction send rates (see Section 4.5): the more send rates adopted, the more precise the final insight on performances will be.
- In the case of multiple consensus algorithms within the same framework, adjustment of the block production rate to make protocols comparable with each other with the same block proposing rate.
- Choice of well-dimensioned hardware to run the experiments on.

These steps represent our methodology for analysis and testing of a blockchain framework and they may apply both for permissioned and permissionless blockchains: in fact, all permissionless blockchains are typically open-source, and most allow setting up local networks using containerization approaches such as Docker.

4.2. Metrics

A standardization of performance evaluation metrics for blockchains has been proposed by the Hyperledger Performance and Scale Working Group (PSWG) [16]. Throughput and latency are identified as characterizing metrics to assess the scalability of a blockchain system. Specifically, transaction throughput is defined as the number of transactions per second processed by the blockchain network: in our case, we consider only successful transactions, hence we can also refer to the throughput as goodput. A transaction is successfully processed when it is included in a block and committed as part of the blockchain. Transaction latency is the time interval elapsed between the submission of a transaction, and the point in time in which the result is available on every correct node in the network (after consensus and propagation time).

Scalability can be measured by analyzing how throughput and latency vary when adding more blockchain nodes to the system: we expect a higher transaction latency when testing larger networks, since the consensus process has to reach a higher number of nodes. Accordingly, for the same reason, we expect throughput to go down when adding nodes to the network. Furthermore, especially when evaluating PBFT-like protocols such as IBFT, latency could be particularly high due to the quadratic communication complexity during normal operations, and to view changes of the system when electing a new blocks proposer.

4.3. Tools

Due to the big momentum of permissioned blockchains, several benchmarking frameworks specifically for this category of blockchain technologies have been developed recently: although the common effort to define a standard for performance analysis, blockchain communities often come up with different tools and framework architectures. The most notable benchmarking frameworks, at the best of our knowledge, are the following:

- Hyperledger Caliper [17].
- Blockbench by Dinh et al. [18].
- Chainhammer by Kruger [19].
- BCTMark [20].

The tool chosen for driving this testbed is Hyperledger Caliper because, beyond the clarity of its documentation, it is easily extensible and it relies on a broad community of developers. However, in order to test the Quorum blockchain, we developed an adapter for Hyperledger Caliper, since Quorum is not yet a supported SUT (System Under Test) in the current version of Caliper (v0.4.1). Our code is available on GitHub [21]: our project is based on a fork of Hyperledger Caliper v0.3.0 (March 2020). Furthermore, to ease the test process and to make our version of Caliper accessible to others, we wrapped it in a Docker image available on Docker Hub [22]. Using the image provided, it is possible to deploy Caliper Master and Workers (extended with support to Quorum) as Docker services.

Another tool we implemented [23] allows to easily spin up a Quorum network with a customizable number of nodes, using Docker Compose. Although it is only to be used for testing purposes, it supports Raft, IBFT, and Clique PoA consensus mechanisms, and it uses the Tessera transaction manager (which is more up-to-date and more production-ready than the Constellation [24] transaction manager). Our tool is based on another GitHub project [25] which only supports Constellation. Moreover, we allow for deterministic accounts and keys creation, which is very useful when it comes to setup Caliper tests which need to use the same private keys and the same Ethereum account addresses in all tests.

4.4. Workloads

For our benchmark, we used a simplified variation of an OLTP (Online Transaction Processing) workload called SmallBank [26]. Given that Quorum is purposely built for the financial sector, we used a benchmark which provides some functionalities reflecting a banking system. The implementation of the SmallBank logic is provided via smart contract: it can be found in the Hyperledger GitHub repository [27]. The smart contract is written in Solidity language and it contains three functions to simulate banking operations:

- `open` (write-only): stores strings representing fake bank accounts, associated with an integer representing the amount of money (write operation).
- `query` (read-only): given an account string, returns the amount associated with it (read operation).
- `transfer` (read-write): simulates a money transfer between two accounts (read and write operation).

Then we implemented Caliper workload files in order to allow worker processes to call the smart contract functions using the Quorum privacy feature.

4.5. Test structure and settings

The design of our test follows a tree structure. Each consensus protocol has been tested against both public transactions (base Go-Ethereum functionalities) and private transactions (Quorum enhanced privacy features). Then, for each protocol, we tested both public and private transactions against three different blockchain network configurations: 4-node, 8-node, and 16-node networks. In this way, we could analyze how throughput and latency vary based on the number of peers, so as to assess scalability. Furthermore, each network topology has been subject to five different transaction submission rates: 25, 50, 100, 200, and 400 TPS, for both read and write operations. Thus, considering 4 different consensus protocols (i.e.: Raft, IBFT sticky, IBFT round-robin, and Clique), the total number of experiment cases we obtained is then 120. Moreover, we executed each test case three times and we calculated both TPS and Latency as the average of the three measurements; we also calculated standard deviation on such results.

Another important variable to take into account when testing consensus protocols is block time, which is the frequency of block generation. Baliga et al. [2] evaluate the effects of block time variations on throughput and latency using Raft: their results show that, by increasing block time and keeping the input transaction rate constant, throughput remains more or less constant, whereas latency visibly increases. Since Raft default block time is 50 ms, while IBFT and Clique block time is 1 s, we increased Raft block time to 1 second as well, so to make protocols comparable.

When executing workloads with private transactions, we decided to make the smart contract private for all Quorum nodes, hence we set every node as a party to each private transaction. This would not be a valuable choice in a real-world application, because such a private transaction is equivalent to a common public transaction, where every node in the system is aware of its payload. However, we wanted to test how much overhead Tessera causes on the private transaction submission process, thus we emulated the worst case scenario in terms of computational complexity: every node Tessera service must encrypt (on write operations) and decrypt (on reads) the payload of each transaction for all participants in the network.

For conducting our experiments, we set up an Amazon EC2 instance with 16 virtual CPUs (8 cores with 2 threads per core), 64 GB of RAM and Ubuntu 18.04 installed on it. For the sake of simplicity, we deployed the blockchain nodes and the Caliper processes in the same machine, using Docker containers. We simulated a real network environment by injecting random delays between containers via `pumba` [29], a tool for emulating containers and network failures within Docker: more precisely, we caused a delay of 100 ms with a jitter of 50 ms, based on ping statistics for servers between Europe and the USA [30]. Our experiment has very affordable costs and it is easily reproducible; furthermore, time required to spin up the network and to manage it is very optimized thanks to Docker Compose.

In order to collect data through Caliper, we deployed 4 Caliper workers as Docker containers: each of them submitted transactions to one node of the blockchain. Thus, even in a network of 16 nodes, we set up the system to send transactions only to 4 Quorum nodes, then requests are propagated throughout the network. Our choice is justified by the fact that, generally, in a permissioned environment, an organization could decide to expose only a subset of the nodes to accept transaction (i.e., write) requests from the users, whereas the other nodes could either be "read-only" nodes or validator nodes participating in the consensus process. Moreover, when an input rate is chosen in Caliper, for instance, 100 TPS, it is divided among the workers so that they reach the send rate as a group. Therefore, there is no difference between having 4 or 8 workers in terms of transaction send rate. The only remarkable difference is in the fact that if 8 workers send requests to 8 different blockchain nodes, then it will take longer for the gossip algorithm to communicate transactions to the consensus leader, hence the latency will likely increase. We measured (see Table 2) whether the system performances were affected by a larger number of workers sending requests to distinct blockchain nodes (e.g., 4 workers send requests to 4 nodes, 6 workers to 6 nodes, and 8 workers to 8 nodes): as expected, for the same input rate (400 TPS) we obtained an increase in latency and an overall decrease of throughput.

However, in future experiments, the number of workers, as the maximum number of blockchain nodes, could be increased, without neglecting the hardware resources that should be changed accordingly.

5. Results

Throughout this section, we discuss experiments on both public and private transactions. Overall, we expect the resulting throughput of public transactions to be higher than the throughput of private ones, due to the overhead of Tessera cryptographic operations on private payloads. However, it is important to remark that executing transactions against a public contract is equivalent to testing the basic `geth` client, whereas Quorum-specific functionalities are tested through the execution of private transactions.

It is worth noting that, regardless of testing public or private transactions, 4-node networks are expected to perform generally better than 8-node networks, which in turn are expected to show better performances than 16-node ones. The reason for this is that blockchain systems do not physiologically achieve scale-out properties; nevertheless, we want to show that even when the SUT is subject to a growing input transaction rate, the resulting throughput trend should be sub-linear: this would imply that a network is capable to absorb an increasing workload without degrading performances.

5.1. Results of Raft

The write-only workload tested with Raft shows that, in public transactions execution, the three network topologies produce a similar throughput up to an input rate of 200 TPS. However, at 400 TPS input rate, the measured throughput of the 16-node network negatively diverges from the 4-node and 8-node networks, demonstrating that larger networks have generally lower performances. Latency, instead, shows an opposite trend with respect to throughput. With public transactions, 4-node and 8-node configurations result in the same latency, stable around 1 s (consider that block-time in Raft is set to 1 s). Overall, private transactions produce a higher latency than public ones, as expected.

Table 2
Effects of adding workers in an 8-node Quorum network running Raft with private transactions on the read-write workload.

Workers	Input Rate (TPS)	Throughput (TPS)	Avg Latency (s)
4	400	52.8	14.9
6	400	48.9	15.3
8	400	49.8	20.8

Similarities of the `open` workload throughput chart (Fig. 2a) and the `transfer` workload throughput chart (Fig. 3a), suggest that the complexity of write operations dominates over the read operations. Graphs of 4-node and 8-node networks tested against public transactions overlap, thus they resulted in approximately the same throughput. However, the reason for the throughput to slightly change and the latency to remain constant between the two scenarios is related to the way Caliper calculates throughput and latency.

(1) `throughput=succ/(final_max-create_min)`, where `final_max` is the latest time amongst all workers in which a transaction is considered as committed, and `create_min` is the earliest time of transaction creation.
(2) `latency=total_delay/succ`, where:
- single transaction delay (i.e., latency) is measured as `final-create`, where `final` is the time in which the transaction is set to success status, and `create` is the time of transaction creation;
- each worker calculates the sum of delays of transactions it has submitted: `delay_sum+=(final-create);`
- `total_delay` is the summation of all `delay_sum` returned by each worker.

Let's suppose to conduct two different experiments on two distinct machines at the same time, such that `final_max_1=final_max_2` and `create_min_1<create_min_2`. Assuming 1600 total transactions submitted, 4 workers, and 100 TPS input rate, it is possible to obtain the following measurements:

- `throughput_1=`1600`/(`1621925597`-`1621925576`)=`1600`/`21=76.19 TPS
- `throughput_2=`1600`/(`1621925597`-`1621925577`)=`1600`/`20=80 TPS

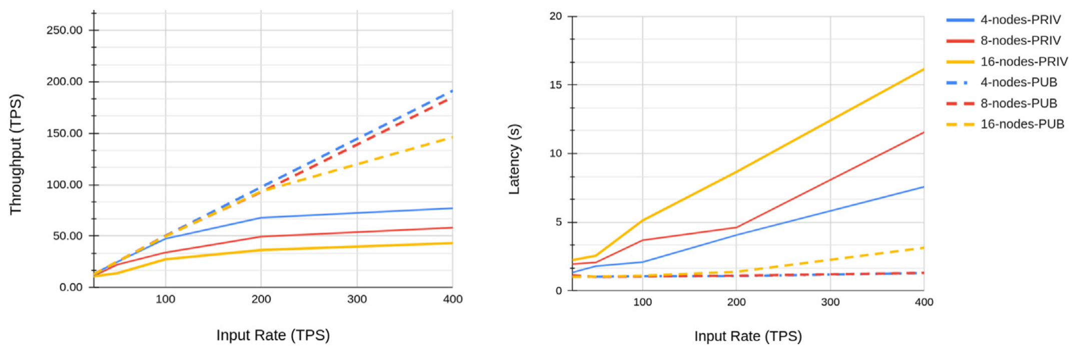
As regards the latency, we could realistically obtain the following results:

- `latency_1=`(1990+1975+2025+2010)`/`1600=5 s
- `latency_2=`(1800+2200+1770+2230)`/`1600=5 s

Thus, we obtained two different experiments in which latencies are the same, whereas `throughput_1` is less than `throughput_2`, because the time frame in which throughput is calculated changes by just 1 s. These results could be due to the 4 workers not being perfectly synchronized when starting to submit transactions. Hence, different throughput results and the same latency values for 4-node and 8-node networks are explained.

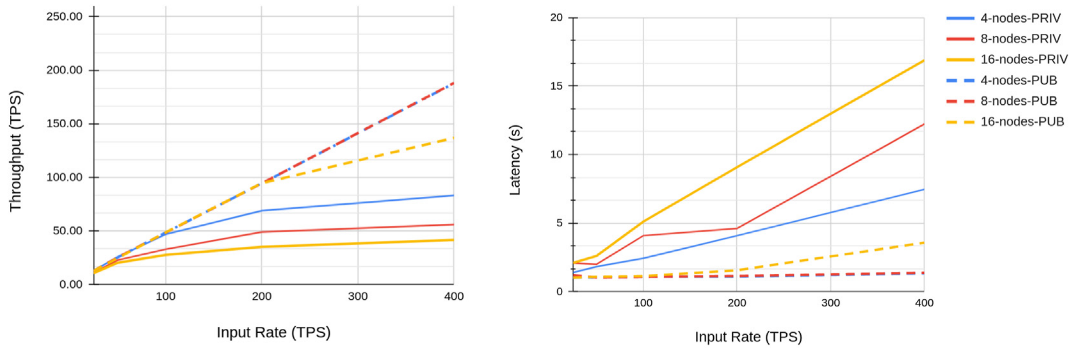
Furthermore, latencies registered in the read-write workloads are analogous to the write-only workloads, with small differences in absolute values.

When testing the `query` workload, we obtained a negligible constant latency—approximately 10 ms—in each network configuration; for this reason, we only reported the throughput of read operations (Fig. 4). On public contract execution, the read throughput graphs of the three networks overlap, whereas private transactions graphs show different slopes. The reason is that the read operation over public smart contracts is a simple look-up to the local `geth` datastore (LevelDB [31]) of the node receiving the request, while read operations executed on private contracts are forwarded to every Tessera Manager of nodes participating in the read private transaction. In the former scenario, since requests are always submitted to 4 blockchain nodes (see Section 4.5), the throughput is the same regardless of the network size, because read operations are only executed by means of a local look-up by the same subset of 4 nodes;



(a) Throughput (b) Latency

Fig. 2. Raft open (write-only) workload performance results.



(a) Throughput (b) Latency

Fig. 3. Raft transfer (read-write) workload performance results.

in the latter, every node is set as a private party to the read transaction, thus each node Tessera Manager is triggered by the read operation, causing the throughput to change based on the number of nodes in the network (due to Tessera cryptographic operations and communication overhead).

Moreover, unexpectedly, the 4-node network produces an overall higher throughput with private transactions rather than with public ones: this can be due to LevelDB being slightly slower than the Tessera H2 in-memory database [32] to retrieve transaction payload information in small networks.

As follows, we reported standard deviation results for both throughput (Table 3) and latency (Table 4) obtained with the `transfer` workload. Generally, we obtained a higher variance with private transaction tests; the highest variance in throughput is shown by the 16-node network with public transactions at 400 TPS input rate. Such variance is mainly due to the random delays produced by `pumba` [29]. Overall, both throughput and latency variances show acceptable values.

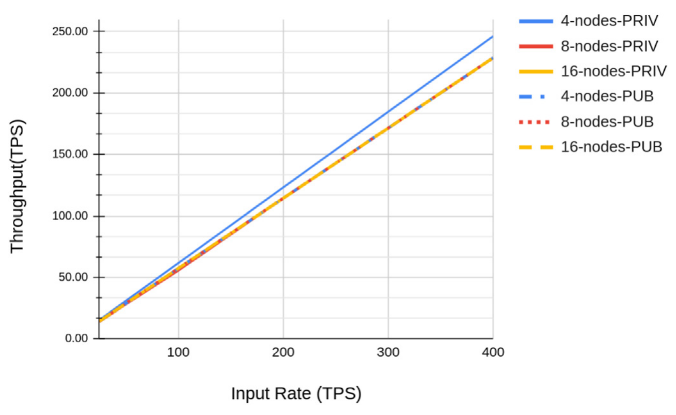


Fig. 4. Raft **query (read-only)** workload performance results—Throughput.

5.2. Results of Clique PoA

The probabilistic approach of Clique, designed to be integrated with the Ethereum Rinkeby test network, implies that forks can occur, but they are eventually resolved. Given the rotating leader schema and the voting mechanism to cope with byzantine failures, it resulted in slightly lower throughput values than Raft (a clear comparison is shown in Section 5.4). However, data produced the same patterns as for Raft: private transactions performed overall worse than public ones, confirming the overhead of Tessera operations on the system. The only exception to the pattern is represented by the 16-node network, whose throughput (Fig. 5a) visibly declines between an input rate of 200 TPS and 400 TPS: at 400 TPS input rate, the 16-node network with public transactions almost coincides with the throughput registered within the 4-node network with private ones; moreover, at the same data-point, the 8-node network and 16-node network resulted roughly in the same

Table 3
Raft standard deviation of Throughput measurements on transfer workload (TPS).

Input (TPS)	4-n-PUB	8-n-PUB	16-n-PUB	4-n-PRIV	8-n-PRIV	16-n-PRIV
25	0.00	0.00	0.00	0.17	0.81	0.98
50	0.00	0.00	1.15	0.36	0.50	0.81
100	0.92	1.50	1.39	0.38	3.74	3.24
200	0.25	0.16	0.57	4.05	0.58	2.47
400	1.18	1.01	9.18	2.58	0.87	0.58

Table 4
Raft standard deviation of Latency measurements on transfer workload (s).

Input (TPS)	4-n-PUB	8-n-PUB	16-n-PUB	4-n-PRIV	8-n-PRIV	16-n-PRIV
25	0.01	0.27	0.27	0.30	0.56	0.60
50	0.02	0.02	0.02	0.02	0.23	0.36
100	0.05	0.02	0.02	0.24	0.77	0.74
200	0.02	0.03	0.12	0.85	2.76	0.68
400	0.18	0.07	0.80	0.84	0.64	1.05

throughput. The decline of throughput of the 16-node network tested with public transactions could be due to race conditions occurring in Clique when block-time is very short (in this case, 1 s): this problem is better explained in Section 5.4.

The latency chart obtained with the `open` workload (Fig. 5b) shows that at 200 TPS input rate, the confirmation delay for a private transaction in a 4-node network is greater than the delay for a public transaction to be confirmed in a 16-node network; inversely, at 400 TPS the 16-node blockchain network with public transactions becomes slower than the 4-node network tested against private transactions.

These results prove that, whenever a large permissioned network (more than a dozen nodes) running a BFT consensus protocol is stressed with a high transaction submission rate, both latency and throughput are negatively affected (see Fig. 6). In addition to inevitably higher consensus times, in large networks the likelihood of submitting transactions directly to the leader drops when the number of nodes increases, resulting in some chunks of transactions being submitted to non-proposer nodes.

With regards to the read-only workload (Fig. 7), we obtained almost the same values as for Raft (Fig. 4): this confirms that read operations are independent of the specific consensus mechanism, since they do not update the nodes state. Moreover, it is worth noting that even in this case, the 4-node network tested with private read transactions performs better than all networks with public transactions, showing that the results are consistent with the ones obtained in Raft.

In Tables 5 and 6 we show respectively the throughput and latency standard deviations for each test case of the `transfer` workload (see Fig. 6): some variance results, especially those at 400 TPS input rate, are even more accentuated than in Raft. This made us suppose that Caliper workers precision is affected by the fact that each worker is a single-thread process: in fact, every worker is in charge of submitting transactions and listening for their confirmations, while gathering time data (this problem is also outlined in Section 5.5). Thus, especially in experiments with larger networks and with high transaction input rates, workers may be overloaded by tasks and some measurements could be subject to lower accuracy.

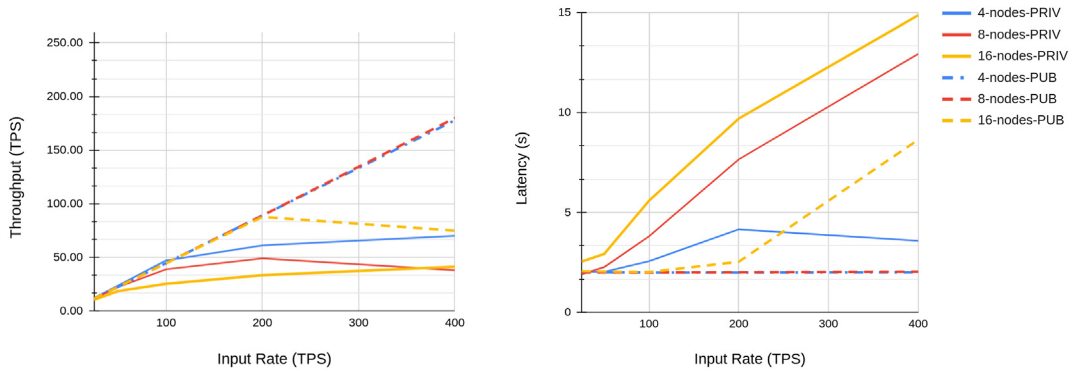
5.3. Results of IBFT

As stated in Section 3, IBFT achieves instant finality (hence, strong consistency): blocks are immediately confirmed to be included into the blockchain and no transient forks can happen, since each block undergoes a three-phase agreement procedure before being committed. This pessimistic approach to consensus could suggest that IBFT performs worse than Clique: in fact, it performs better under some network configurations (Section 5.4), probably due to its deterministic leader election mechanism. In this section, we only present results obtained with the read-write workload, since the transfer operation is the most commonly used in a banking scenario. We focus on the comparison between performances measured with the round-robin leader election policy and the sticky proposer, on both public and private transactions.

Before running the tests, we had contrasting expectations about the possible outcomes. On one hand, the sticky proposer strategy sounds like a lightweight solution, since the leader is changed only in case it behaves maliciously or it crashes; however, in larger networks, if the proposer never fails, it could become a system bottleneck, since it can easily be saturated with requests. On the other hand, the round-robin strategy implies frequent view changes to be applied: since a block can be decided only if the majority ($> 2/3$) of nodes has the same view of the system, this could cause a significant delay on transaction processing within extended networks.

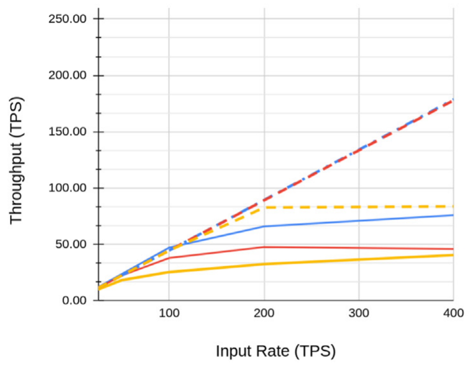
Results clearly represent our preliminary thoughts about the performances of the two strategies. Tests with public transactions (Fig. 8) show that 4-node and 8-node networks have the same transaction latency, steady around 2 s, whereas in 16-node networks the sticky proposer strategy slightly outperforms the round-robin. As regards throughput, 4-node networks overlap on the same values; on 8-node configuration round-robin performs better than the sticky proposer, as opposed to 16-node networks. Private transactions (Fig. 9) reported interesting results as well: on 16-node networks the round-robin algorithm results in a higher transaction throughput than the sticky proposer, while on 4-node networks the sticky proposer strategy beats the round-robin at 400 TPS input rate. Contrasting trends are shown by the latency chart (Fig. 9b): in the 8-node network, the sticky proposer presents a higher transaction delay up to 200 TPS input rate, whereas at 400 TPS the round-robin is slightly slower than the sticky proposer; the 4-node topology shows overall the lowest latency, even though it reaches the same latency as 8-node networks at 400 TPS input rate.

These results are alternatively in favor of the two leader election policies examined, reinforcing the thesis that both have advantages and disadvantages which are offset from one another; thus, it cannot be inferred which policy is the best just by looking at performance metrics. However, it is recommended to use the round-robin strategy, since it avoids single point of failure, resulting in a more secure choice for a BFT protocol.

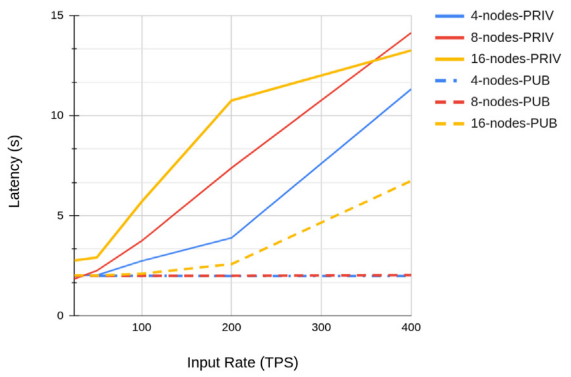


(a) Throughput (b) Latency

Fig. 5. Clique open (write-only) workload performance results.



(a) Throughput



(b) Latency

Fig. 6. Clique transfer (read-write) workload performance results.

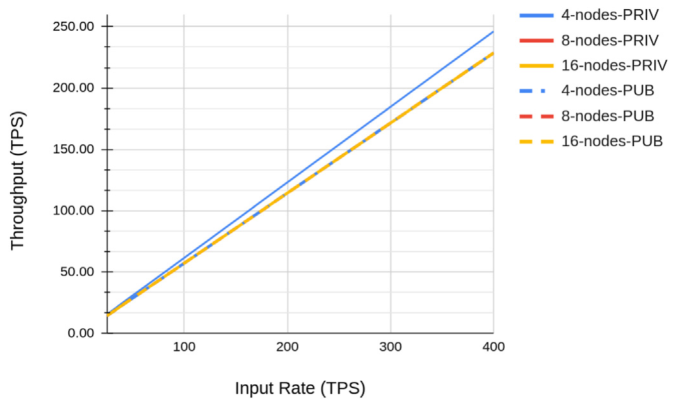


Fig. 7. Clique query (read-only) workload performance results—Throughput.

Table 5
Clique standard deviation of Throughput measurements on transfer workload (TPS).

Input (TPS)	4-n-PUB	8-n-PUB	16-n-PUB	4-n-PRIV	8-n-PRIV	16-n-PRIV
25	0.00	0.00	0.00	0.00	0.06	1.69
50	0.00	0.00	0.00	0.00	0.00	0.96
100	0.00	0.00	0.00	0.12	0.12	2.4
200	0.26	0.00	7.5	3.12	0.76	2.51
400	0.70	0.06	17.03	7.18	7.11	2.00

Table 6
Clique standard deviation of Latency measurements on transfer workload (seconds).

Input (TPS)	4-n-PUB	8-n-PUB	16-n-PUB	4-n-PRIV	8-n-PRIV	16-n-PRIV
25	0.04	0.00	0.02	0.01	0.33	0.78
50	0.01	0.00	0.01	0.02	0.36	0.40
100	0.00	0.00	0.07	1.24	1.36	0.58
200	0.01	0.00	0.29	0.56	1.69	1.17
400	0.01	0.01	1.1	7.13	2.37	0.82

5.4. Comparisons

In order to have a comprehensive view of permissioned blockchain performances with different consensus protocols, a comparative analysis should be developed on experimental results. Such comparison is the practical counterpart of the theoretical comparison presented in Table 2, thus it represents step (7) of the macro-methodology described in Section 4.1. We focus on condensing the results of the read-write workload with private transactions, since privacy is the novel feature introduced by this blockchain platform. Moreover, we only show IBFT with a round-robin leader election strategy, so to make it comparable with the rotating proposers subset of Clique PoA.

As we expected (see Section 3.4), overall Raft shows a higher throughput (Fig. 10) than the two byzantine protocols. One may note that Raft is lightly outperformed by both Clique and IBFT only within the 8-node configuration between 25 TPS and 200 TPS input rates. Notwithstanding, this is an exception with respect to the other results, that could be due to the injection of random delays between the blockchain nodes: if such delays are particularly long for the Raft leader, then the resulting throughput and latency can likely be penalized. However, this may be corrected by increasing the number of experiments for each test case (for instance, from 3 to 10), so to produce a much higher statistical population, and therefore more accurate data.

Moreover, with regards to throughput, Clique performs visibly better than IBFT in the 16-node network, whereas in the other two configurations, IBFT shows on average a slightly better throughput: this is a surprising result, because we supposed IBFT quadratic communication complexity would have been a point against it in every network topology. An explanation for this can be found in the way Clique implements the choice of a block proposer: since there could be multiple proposers for each block, transient forks can occur (see Section 3.2), and in the presence of forks, chain reorganization time must be taken into account. Moreover, Clique suggested block period is 30 s (since it has been developed to emulate the Ethereum mainnet), but we set it to 1 s: as it has been pointed out in this GitHub issue [33], 1 s is a quite small block period, and if block processing time exceeds the block period, race conditions between nodes can occur. However, IBFT higher complexity emerges in latency results, especially in the 16-node (Fig. 11b) network: here, in fact, quadratic communication complexity has a much higher impact on network performances than the presence of possible forks in Clique.

Another noticeable result is Raft latency in Fig. 11c, which overcomes the byzantine protocols latencies on an input rate of 400 TPS: this could be caused by the sticky leader (or strong leader) being congested on high input rates, therefore slowing down the entire transaction execution process.

Overall, the benchmark results gave us interesting insights into the real performances of consensus protocols applied to practical test scenarios. Experimental data helped us in better understanding which characteristics of each consensus algorithm have a major impact on throughput and latency in different blockchain network dimensions, and confirmed our expectations (see Section 3.4). Therefore, the evaluation of performance measurements is the completion of the theoretical analysis outlined throughout Section 3, and charts like Figs. 10 and 11 represent an effective practical insight of comparison reported in Table 1.

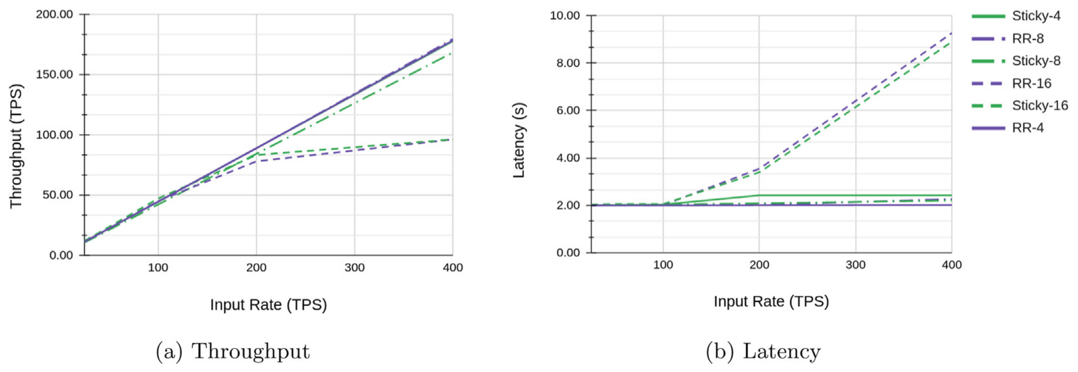


Fig. 8. IBFT transfer (read-write) workload with public transactions: comparison between sticky proposer and round-robin proposer policies.

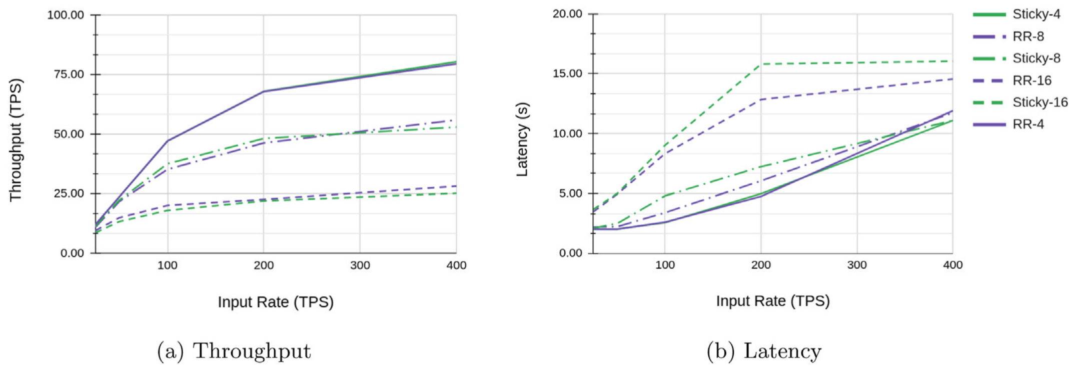


Fig. 9. IBFT transfer (read-write) workload with private transactions: comparison between sticky proposer and round-robin proposer policies.

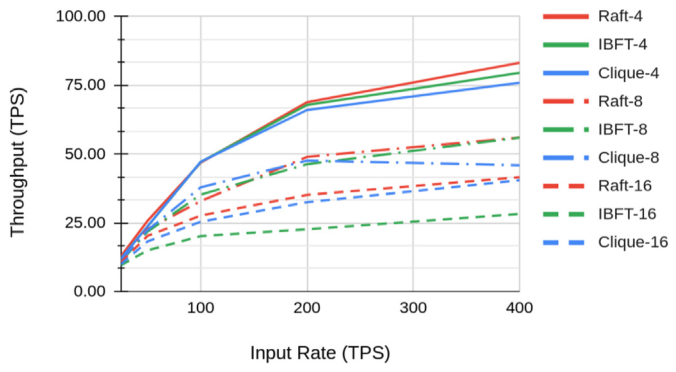


Fig. 10. Throughput transfer (read-write) workload with private transactions: comparison between Raft, IBFT round-robin, and Clique.

5.5. *Applicability and scalability*

With regard to scalability, we can state that all three consensus protocols tested against private transactions exhibit a remarkable decline in throughput when the number of peers in the network increases. On average, with private transactions, throughput decreases of 1/3 TPS whenever the number of nodes doubles. Throughput trends over all protocols and network configurations are visibly logarithmic: we did not obtain a significant increase in throughput when increasing the input rate from 100 TPS to 200 TPS, and from 200 TPS to 400 TPS. With regards to latency, it generally increases over all topologies. Moreover, we tried to push our tests over 400 TPS input rate, but it resulted in a large part of transactions failing. This is probably due to either:

- RPC server buffers of Quorum nodes being limited to 128 KB.
- Tessera transaction manager being a bottleneck for transactions execution.
- Caliper worker being a single-thread process in charge of both submitting transactions and waiting for their receipts.

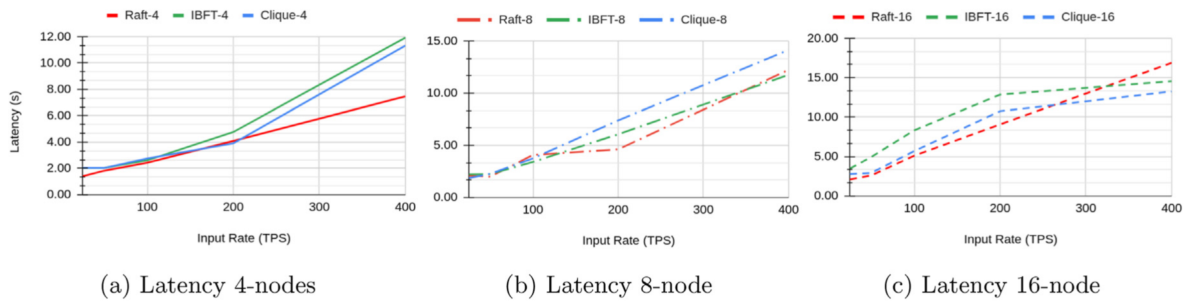


Fig. 11. Latency of transfer (read-write) workload with private transactions: comparison between Raft, IBFT round-robin, and Clique. The y-axis on (b) and (c) share their label with (a), which is Latency (s).

Baliga et al. [28] claim to solve the last issue in the bullet points above, by splitting the Caliper worker process in two separate processes, one for submitting transactions and one for listening: code of such a change has not been open-sourced, making this correction unavailable to the community. However, it is worth noting that all transactions submitted during our experiments were executed successfully (no rejected transactions): we did not make any architectural modifications to Quorum servers and Caliper workers architecture, in order to minimize changes required to the frameworks.

To sum up, our evaluation results show that IBFT represents the best choice for Quorum in financial applications, such as stable coins or CBDCs (Central Bank Digital Currencies) [34]; in fact, it guarantees strong consistency and fault tolerance to byzantine failures with performances comparable to both Raft and Clique. Furthermore, it reaches its optimal results with respect to the other protocols, if the network is deployed with a number of nodes in the order of a dozen. However, we are quite confident that by conveniently dimensioning Quorum blockchain nodes (i.e., by allocating enough hardware resources for production grade environments) and by improving Caliper as a benchmarking tool (as we expect), performances can be improved with respect to the experiments we have conducted. Nevertheless, obtaining the best throughput and latency results for a Quorum network is outside the scope of this research: we focused on showing a valid and repeatable methodology for analyzing a permissioned blockchain technology to achieve a tangible insight (with good approximation) on its scalability.

6. Related works

To the best of our knowledge, the first and only performance evaluation of Quorum prior to our work has been published by Baliga et al. [28] in 2018. They adopted Hyperledger Caliper as well, but neither the source code of Caliper extensions for Quorum nor deployment documentation are publicly available, making their experiments difficult to reproduce. Furthermore, they compared Raft and IBFT consensus mechanisms by performing benchmarks on a 3-node network for Raft, and a 4-node network for IBFT, which is the minimum peer number for both (respectively): in this way, they were not able to assess the real scalability of the networks because of the fixed number of peers.

In 2017, Dinh et al. [18] proposed Blockbench, the first framework for benchmarking permissioned blockchains: the paper focuses on comparing Ethereum, Parity, and Hyperledger Fabric. However, Hyperledger Caliper established itself as the reference benchmarking tool: it has been designed with flexibility and extensibility in mind, it provided a thorough documentation, and the project was maintained by a broad community of developers, since it was part of the Hyperledger umbrella project [35] hosted by the Linux Foundation.

Another framework for blockchain testing was developed by De Angelis [36], which focused on the security properties of the system: in his work, he illustrated how to integrate a Byzantine client into the blockchain network in order to simulate several attacks and study their collateral effects. His work mainly focused on PoA algorithms, such as Aura [37] and Clique PoA [9].

7. Conclusion

In this paper, we outlined a general workflow for benchmarking permissioned blockchain platforms and related consensus algorithms. We presented as a practical case study the ConsenSys Quorum blockchain: it offers three plug-and-play consensus mechanisms, which have been analyzed, also taking into account the CAP trade-off problem [7]. The theoretical analysis (Section 3) performed on Raft, Clique PoA and IBFT, can be transposed to every consensus algorithm. Then, we outlined methodology and tools to drive and produce performance evaluations of blockchain technologies, by comparing different consensus mechanisms under different network configurations. While previous works limited the generalizability of the results, our approach was totally replicable, open-source and it provided new insights into the scalability quality of consensus protocols, by performing tests on a variable number of blockchain nodes.

Although this research clearly illustrates the effects of different parameters on performance and scalability of permissioned Quorum networks, it also raises many questions. For instance, further research is needed on security considerations: future works should address trade-offs between security and performance, or even effects of byzantine nodes on network safety. Moreover, an open-source modification to the Caliper framework could be proposed so to make the worker processes multithread (as in Ref. [28]), thus alleviating the transaction rejection issue on high input rates. This modification could also increase measurement accuracy.

Nevertheless, our tools are ready to be used to collect data on many more combinations of performance variables and network configurations of Quorum. Specifically, it could be meaningful to analyze memory and CPU consumption of blockchain nodes, by leveraging Prometheus [38] as a monitoring tool (already integrated with Caliper). Furthermore, the deployment tool we implemented [23], could be extended to allow new nodes to dynamically join the Quorum network without the need for static IPs: in that case, the overhead of the reconfiguration protocol of consensus mechanisms must be studied.

Overall, our tools can be taken as examples to implement new Caliper plugins to test other blockchain platforms in the financial context. Testing and comparing performances of different permissioned blockchains can be useful both for research scopes and for practical applications, since this technology is being considered for adoption in interesting financial scenarios, such as interbank transactions and Central Bank Digital Currencies [34]. In addition, alternative workloads can be proposed so to extend benchmarks to other use cases, such as supply chain.

To conclude, the workflow we proposed is flexible enough to be applied to any permissioned blockchain context: it is useful for assessing and analyzing performances of various consensus algorithms within the same blockchain framework, but it also enables cross-blockchain comparisons.

Author contributions

Marco Mazzoni: Conceptualization, Investigation, Methodology, Writing (original draft), Software; **Antonio Corradi:** Conceptualization, Validation, Writing (review & editing), Supervision; **Vincenzo Di Nicola:** Writing (review & editing), Resources, Supervision, Project administration.

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.

References

[1] ConsenSys, Quorum whitepaper. https://github.com/ConsenSys/quorum-docs/blob/master/Quorum\Whitepaper\v0.2.pdf.
[2] G. Wood, Ethereum: a secure decentralised generalised transaction ledger, in: Ethereum Project Yellow Paper 151, 2014, pp. 1–32.
[3] ConsenSys, Private transaction lifecycle. https://docs.goquorum.consensys.net/en/stable/Concepts/Privacy/PrivateTransactionLifecycle/.
[4] ConsenSys, Tessera. https://github.com/ConsenSys/tessera.
[5] Ethereum Foundation, Patricia tree. https://eth.wiki/fundamentals/patricia-tree.
[6] ConsenSys, GoQuorum FAQ. https://docs.goquorum.consensys.net/en/stable/Reference/FAQ/.
[7] E.A. Brewer, Towards robust distributed systems, in: Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing; 16–19 Jul 2000; Portland, OR, USA, ACM, New York, NY, USA, 2000, pp. 343477–343502.
[8] D. Ongaro, J. Ousterhout, In search of an understandable consensus algorithm, in: Proceedings of the 2014 USENIX Conference; 19–20 Jun 2014; Philadelphia, PA, USA, USENIX Association, 2014, pp. 305–320.

[9] P. Szilágyi, EIP-225: Clique proof-of-authority consensus protocol, Ethereum improvement proposals, no. 225 (March 2017), https://eips.ethereum.org/EIPS/eip-225.
[10] H. Moniz, The Istanbul BFT consensus algorithm, arXiv: 2002.03613 arXiv, 2020 arXiv preprint.
[11] L. Lamport, The part-time parliament, ACM Trans. Comput. Syst. 16 (2) (1998) 133–169.
[12] M. Vukolić, The quest for scalable blockchain fabric: proof-of-work vs. BFT replication, in: J. Camenisch, D. Kesdoğan (Eds.), In: Open Problems in Network Security, Springer, Cham, 2015, pp. 112–125.
[13] M. Castro, B. Liskov, Practical Byzantine fault tolerance and proactive recovery, ACM Trans. Comput. Syst. 20 (4) (2002) 398–461.
[14] Y. Sompolinsky, A. Zohar, Secure high-rate transaction processing in bitcoin, in: R. Böhme, T. Okamoto (Eds.), In: Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, 2015, pp. 507–527.
[15] Y.-T. Lin, Istanbul Byzantine fault tolerance. https://github.com/ethereum/EIPs/issues/650. (Accessed June 2017).
[16] Hyperledger, Hyperledger blockchain performance metrics. https://www.hyperledger.org/learn/publications/blockchain-performance-metrics, October 2018.
[17] Hyperledger, Hyperledger caliper. https://hyperledger.github.io/caliper/.
[18] T.T.A. Dinh, J. Wang, G. Chen, et al., BLOCKBENCH: A framework for analyzing private blockchains, arXiv:1703.04057, arXiv, 2017. arXiv preprint.
[19] A. Krueger, Benchmarking scripts "chainhammer". https://github.com/drandreaskrueger/chainhammer, 2018.
[20] D. Saingre, T. Ledoux, J.-M. Menaud, BCTMark: a framework for benchmarking blockchain technologies, in: AICCSA 2020 - 17th IEEE/ACS International Conference on Computer Systems and Applications, ; 2–5 Nov 2020; Antalya, Turkey, IEEE, Piscataway, NJ, USA, 2020, pp. 1–8.
[21] M. Mazzoni, Caliper. https://github.com/MarcoMazzoni/caliper, 2020.
[22] M. Mazzoni, Caliper image with quorum extension. https://hub.docker.com/r/marcomazzoni/caliper-quorum.
[23] M. Mazzoni, quorum-docker-nnodes. https://github.com/MarcoMazzoni/quorum-docker-Nnodes, 2020.
[24] ConsenSys, Constellation. https://github.com/ConsenSys/constellation.
[25] pccr10001, quorum-docker-nnodes. https://github.com/pccr10001/quorum-docker-Nnodes.
[26] M. Alomari, M. Cahill, A. Fekete, et al., The cost of serializability on platforms that use snapshot isolation, in: 2008 IEEE 24th International Conference on Data Engineering; 7–12 April 2008; Cancun, Mexico, IEEE, Piscataway, NJ, USA, 2008, pp. 576–585.
[27] Hyperledger, Hyperledger caliper benchmarks. https://github.com/hyperledger/caliper-benchmarks.
[28] A. Baliga, I. Subhod, P. Kamat, et al., Performance Evaluation of the Quorum Blockchain Platform, arXiv (2018) arXiv preprint.
[29] A. Ledenev, Pumba: chaos testing tool for Docker. https://github.com/alexei-led/pumba. Accessed on 5-20-2021.
[30] WonderNetwork, Global ping statistics. https://wondernetwork.com/pings.
[31] J. Dean, S. Ghemawat, LevelDB. https://github.com/google/leveldb.
[32] T. Mueller, H2 database engine. http://www.h2database.com/html/main.html.
[33] C. N. Samuel, Geth Clique block period 1, fork of chain inconsistency. https://github.com/ethereum/go-ethereum/issues/21191. Accessed on 5-16-2021.
[34] European Central Bank, Central bank digital currencies (CBDC). https://www.ecb.europa.eu/home/search/html/central_bank_digital_currencies_cbdc.en.html.
[35] Hyperledger, Hyperledger: blockchain technologies for business. https://wiki.hyperledger.org/.
[36] S. De Angelis, Assessing security and performances of consensus algorithms for permissioned blockchains, arXiv:1805.03490, arXiv, 2018. arXiv preprint.
[37] OpenEthereum, Aura - authority round. https://openethereum.github.io/wiki/Aura.
[38] Prometheus Authors 2014-2020, Prometheus - monitoring system and time series database. https://prometheus.io.