Benchmarking Blockchains: The case of XRP Ledger and Beyond

Marios Touloupou, Klitos Christodoulou, Antonis Inglezakis, Elias Iosif, Marinos Themistocleous
Department of Digital Innovation, University of Nicosia
Nicosia, Cyprus
{touloupos.m, christodoulou.kl, inglezakis.a, iosif.e, themistocleous.m}@unic.ac.cy

Abstract

Blockchain and Distributed Ledger Technologies appear to be at a worldwide threshold of acceptance and adoption. Since their inception, several innovative projects have been proposing solutions to the blockchain trilemma, improving blockchain features and its technical limitations. However, the adoption of blockchain as a technology requires a comprehensive understanding and characterization of its technical aspects. The latter introduces an uncertainty for an organization to decide which blockchain protocol best meets its needs and demands. In general, there is a lack of proper testing and software engineering practices for assessing the usage of different blockchain protocols and understanding their performance. Toward that direction, this paper presents an architecture for a blockchain benchmarking framework that aims at the deployment and evaluation of different blockchain protocols. Moreover, we introduce a set of modules for testing and evaluating their behavior under different test-cases and scenarios. To illustrate the usefulness of the proposed architecture we demonstrate an instantiation with the deployment of a private XRPL Network. The experiments conducted in this work were focused on how XRPL behaves under heavy load.

1. Introduction

A distributed ledger is often described as a shared database which is accessed and maintained by a set of independent, possibly untrusted participants (i.e., nodes). Each participant can access and own an identical copy of the records (i.e., the ledger) exchanged within the network. All modifications or additions to the ledger are expressed immediately and agreed among the participants using a Consensus Algorithm (CA). Blockchain, which is considered as a type of a Distributed Ledger Technology (DLT), was first introduced within the concept of a cryptocurrency, while by then has received a lot of attention due to the unique characteristics it provides; i.e., security, anonymity, transparency, and decentralization [1]. The decentralized design of a blockchain lack of some central authority to synchronize the state of the processes; which is considered a major challenge. Thus, blockchains are providing mechanisms for (a) Coordinating the distributed nodes, and (b) Validating the state of transactions propagated in the network. These mechanisms are the CAs; which are responsible for achieving the aforementioned goals. Moreover, CAs provide reliability and liveness to the network, while they also defend it against any malicious (aka byzantine) attacks [2].

Since 2008; when Bitcoin was first introduced by Satoshi Nakamoto [3], blockchain has continued to grow and evolve. Moreover, for several years the world compared blockchain technology with Bitcoin; but soon it was realized that blockchain was introducing a radical change in the internet stack itself. The blockchain ecosystem soon realized the need for this technology to serve as a framework where applications can run on top of it while also be able to self-execute. Before this realization, blockchain was mainly used to serve Bitcoin's needs, executing financial transactions, and as a sybil attack prevention mechanism. In 2014, Ethereum [4] was proposed as the next generation of a blockchain protocol; enabling the development of the so-called distributed applications (dApps). Ethereum also gave birth to the concept of Decentralized Autonomous Organizations (DAO), a decentralized enterprise completely operating with the use of a set of smart contracts. In a later stage, several researchers and organizations focused on the scalability of blockchain protocols. Thus, several solutions were proposed for improving the transaction rate but also lower the latency within a network.

The choice of a blockchain protocol is challenging, especially for corporates that seek to use the technology for their products and/or services. Before the adoption of blockchain in a company's infrastructure, questions such as (a) Is blockchain applicable to the company's/organization challenges? (b) Which blockchain protocol is right for our needs? (c) To what extend does the selected blockchain can handle the security and privacy concerns of a client?, are discussed within their technical and management teams.

URI: https://hdl.handle.net/10125/80070
978-0-9981331-5-7

In an attempt to taggle the aforementioned challenges, this paper proposes an architecture for a blockchain benchmarking framework that aims to serve as a staged environment for supporting blockchain researchers and developers to test and validate the performance of a blockchain protocol under various settings and synthetic scenarios. In addition, the modules provided by the benchmarking framework aim to identify any performance bottlenecks during the network being under heavy load.

The rest of the paper is organized as follows. Section 2 tries to build a common understanding around blockchain by providing a description of the layers that reflect a blockchain protocol, a discussion on their most important features and a description of the most commonly used CAs. In Section 3, a literature review around blockchain benchmarking frameworks is presented, while Section 4 demonstrates the proposed framework with a description of the proposed architecture and its integral parts. In Section 5, a use case scenario is described (i.e, the XRPL - Ripple's Case), providing a description on the features supported so far; along with an evaluation of the overall architecture. Finally, in Section 6, a discussion around the implications of the current research is provided while at the end, the paper concludes with our future plans and goals.

2. Background - Building a Common Understanding

2.1 Blockchain Preliminaries

Blockchain protocols are broadly classified into two main categories. Public networks [5], which are considered to be democratic since they promote equal participation to all, and on the other hand private networks which are usually isolated networks governed by an organization or by a single party. Accessing private networks usually demands having an invitation, while also accept some rules defined in the beginning of the network.

Transactions: In the beginning of cryptocurrencies, transactions were only used for transferring a digital asset to another person's account (i.e.,wallet). Nowaday, transactions may function in several different ways since further metadata can be encoded on them [6]. The participants of the blockchain may perform multiple transactions in time, either by transferring digital money or by adding a record, in the form of metadata, within the network. All unconfirmed transactions are entering a pool where the ones with the higher fee are executed first. After the successful execution and validation of the transaction the latter is attached on the next block. Finally, the block is appended on-chain and thus the blockchain is expanded by one block.

Blocks: Blocks organize transactions and other metadata that relate to the data structure (e.g., hashes, timestamps, nonce). Transactions in blocks are encoded into a Merkle Tree [7] while each block includes the hash of the previous block as a pointer and thus forming a chain. This method maintains the integrity of the system, while one can validate the whole chain all the way back to the genesis block [8]. Moreover, a block can only be appended and not altered. Thus, the latter provides enhanced security; since no entity can change the data once in a block.

Block time: Block time is considered the time required for a new verified block to be appended to the chain. Some blockchain protocols produce a new verified block every few seconds, while this process varies based on the system's complexity. Block time is a key characteristic of blockchain systems which is often the case that block time relates to higher transaction rates. For blockchains, it is often a challenge to find a balance between block times and security of the network; since time is required for data to be validated and broadcasted to all nodes in the network. For instance, Ethereum [9] can produce one validated block approximately every 15 seconds, while in Bitcoin it is on average of every 10 minutes [10].

2.2 Blockchain Layers

As depicted in Figure 1, a blockchain protocol can be reflected in mainly five layers. Bottom-up these are: the infrastructure layer, the network layer, the protocol layer, the services and the (optional) components layer , while at the top of the stack, the application layer.

Infrastructure Layer: A blockchain network is built on top of a P2P network supported by several machines in a decentralized manner. Some machines can operate on tasks at any given time while taking computing resources or storing from those in the network. Thus, at the lower level of the stack, the infrastructure layer can be considered as a group of machines operating together. (e.g., miners [11]).

Network Layer: Blockchain is considered as another layer on top of the internet, as without it, it would not operate. Currently, a blockchain protocol runs over TCP/IP. As such, the network layer consists of the connections between machines, and everything else that lays the groundwork for the network to operate.

Protocol Layer: The protocol layer defines the protocol rules and the incentivization structures if any. In addition, this is the layer that defines the consensus mechanism used by the protocol.

Services and Optional Components Layer: This layer, refers to the tools and interfaces for interacting with the protocol and for supporting the development of dApps. It is worth mentioning that some of these tools and technologies are offered off-chain, meaning that developers can build them offline, on their private computers.

Application Layer: Last, the application layer is positioned at the top of the stack, and it is necessary for hosting the distributed applications. In this way, the dApp will be hosted on a decentralized network using Software as a Service (SaaS). This layer also makes all dApps easy to access and integrated with any other device.

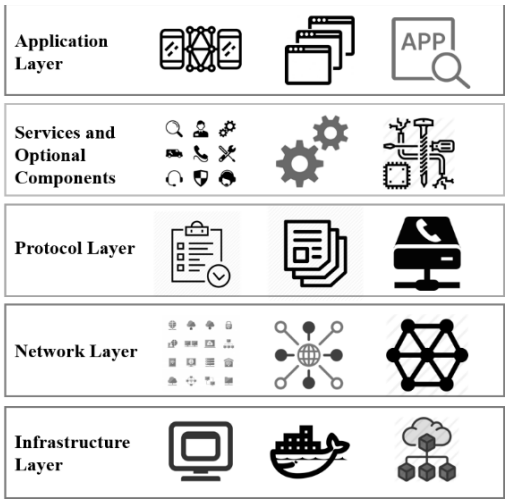

Figure 1. Blockchain stack/layers

2.3 Blockchain Consensus Algorithms

Consensus Algorithms are at the core of every blockchain protocol. Several CAs have been discussed in the literature [12], each one of them providing its own unique characteristics. Consensus Algorithms are responsible for the decision-making process of the active group of nodes that are participating in the network. Also, CAs keep the protocol active while nodes may not trust each other but they trust the algorithm that runs in the core of the blockchain protocol. A list of the most common CAs used in blockchain protocols follows.

Proof of Work (PoW): This CA is used to select a miner for the next block generation. Bitcoin uses the PoW CA. The fundamental concept behind this algorithm is to overcome a complex mathematical problem and give a solution fast. The complexity of the "puzzle" depends on the number of participants, the existing power and the network load. The hash of each block includes the hash of the previous block, which increases security and avoids any block breach from happening. Furthermore, this mathematical puzzle requires a lot of computational power and thus the node that solves the puzzle gets to mine the next block.

Proof of Stake (PoS): Proof of Stake [13] is the most common alternative to PoW. It is foreseen to be used in Ethereum which will shift from PoW to PoS. In this type of CA, validators invest in the system's tokens instead of investing in costly hardware to solve a complicated puzzle by locking up some of their coins. Validators validate blocks by placing a bet on them if they discover a block they believe can be added to the chain. Moreover, all validators get a reward proportionate to their bets based on the actual blocks added in the blockchain.

practical Byzantine Fault Tolerance (pBFT): In the late 1990s, a CA called Practical Byzantine Fault Tolerance was introduced. pBFT was created to perform well in asynchronous systems while it is designed to have a minimal overhead time. Its objective was to address a number of issues with existing Byzantine Fault Tolerance methods.

3. Related Work - Blockchain Benchmarking Frameworks

Currently, there are multiple studies regarding measuring the performance of blockchain protocols. Some of those studies are targeting public blockchains while some others the private ones. BlockBench [14] is a framework for analyzing private blockchain protocols. It is considered adaptable in terms of integration of any private blockchain while it can measure throughput, latency, scalability, and fault tolerance against different workloads. Additionally, the authors in [15] have considered the scalability of blockchain protocols an urgent concern. Thus, they have studied how different bottlenecks in the Bitcoin network can affect the overall throughput of the network. Based on the results of their work, they concluded to the fact that block size reparameterization should be considered as priority towards achieving next-generation, high-load blockchain protocols. In the work conducted in [16], the authors have studied the propagation time of blocks and transactions in the network concluding to the fact that the latter is the primary cause for blockchain forks. They have also demonstrated what can be achieved while pushing the network to its limit by introducing unilateral changes to the client's behavior. Based on the benchmarks demonstrated in [17], Parity has proved to be the fastest and lightest Ethereum client in terms of block processing time. Moreover, the authors in [18], have introduced a framework for analyzing existing PoW-based deployments and PoW blockchain variants in an attempt to compare the trade-offs between their performance and security provisions. Moreover, along the most popular blockchain benchmarking frameworks

is also IBM's Caliper [19]. Hyperledger Caliper is a blockchain benchmarking tool intended to run benchmarks on deployed smart contracts, allowing the analysis of throughput, latency and resource consumption of the smart contract. As of 2019, the authors in [20] categorize Hyperledger Caliper and Blockbench as the two most popular blockchain benchmarking frameworks; while their work demonstrate a comparison between these two. The work conducted in [21] introduce BCTMark which is a framework for benchmarking blockchain technologies on an emulated network. The researchers of this work have conducted their experiments on three blockchain protocols where they have measured different metrics such as CPU consumption and energy footprint for different numbers of clients. Not only that, but also in the work conducted in [22], the researchers characterized the performance feature of Quorum [23]. They have studied its throughput and latency characteristics with different workloads and CAs. In summary, using a suite of micro-benchmarks, they have explored how certain transaction and smart contact parameters may affect the latency of transactions.

4. Blockchain Benchmarking Framework - Proposition

Testing is a critical phase of the software engineering life-cycle; especially before moving an application to the production environment. Blockchain protocols are complex systems that comprise of many components ranging from the underlying communication network, cryptographic libraries, gossip protocols, consensus algorithms, virtual machines and game theoretical aspects. It is often the case that bootstrapping a private blockchain network on a private computer and use it for testing is a challenging task. It is even more challenging to compare various private blockchain implementations in terms of transactions throughput, latency, fault-tolerance, and scalability.

Having an isolated environment where you can introduce changes to the source code, test and debug the system without affecting the implementation of the production blockchain, is essential. Implementing a blockchain infrastructure considers several design choices such as network performance, network anomalies, node's misbehavior, etc. However, the latter introduces several challenges; while a blockchain network usually consists of several nodes running in different machines around the world (i.e., high level of distribution and decentralization).

To this end, we are proposing an initial setup of a blockchain benchmarking framework which is depicted in Figure 2 and is publicly available in our GitHub repository [24]. Currently, it is able to deploy a full mesh network with a given number of nodes/validators. Moreover, different scripts are developed for generating traffic in the network (i.e., in the form of payment transactions), a monitoring framework for capturing and visualizing data produced in the network, but also a connectivity manager, aiming for the adaptation of the network rules of the validators during runtime.

4.1 Architecture Overview

As depicted in Figure 2, the proposed benchmarking framework is consisted with four main building blocks. These are: (a) the Control & Configuration components, (b) the Validators' Network, (c) the Accounts Management and Traffic Generator, and lastly (d) the Monitoring Services.

4.2 Internal Components

Control and Configuration: In the control and configuration component, a set of scripts have been developed for the generation of the configuration files, bootstrapping the network, and adapting the connectivity between the nodes/validators. Specifically, the process of deploying a blockchain network with *n* number of nodes/validators is limited to a single line of scripting code *"./run_testnet X N*". Variable *X* defines the blockchain protocol to be deployed, where *N* defines the number of nodes/validators to be part of the network.

Validators' Network: One of our initial goals was to design a benchmarking framework which would provide dynamicity to the end user. To this end, our automation scripts were designed in such a way that changing the number in the deployment request results in a network topology with the number of nodes the user has defined. The upper limit of the number of nodes/validators that can be deployed, depends on the currently available resources.

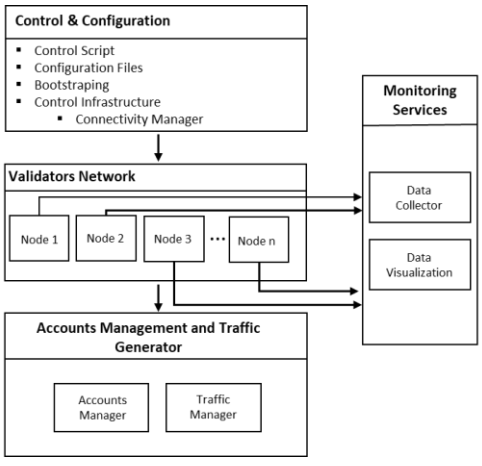

Figure 2. Blockchain benchmarking framework architecture

Accounts Management and Traffic Generator: Generating traffic in the network while the validators/nodes work on closing the next ledger provide useful insights for the blockchain under test. Closing a new ledger/block and attach it on the chain is performed by the execution of transactions. Transactions should also be signed (with the private key of the proposer), accepted, and validated following the underlying CA of the blockchain protocol. As it is depicted in the "Accounts Management and Traffic Generator" block of Figure 3, two components were developed responsible for generating accounts but also execute transactions in the network. The user may call the Traffic Generator Manager, giving as parameter the number of transactions to be executed but also the amount of coins to be exchanged.

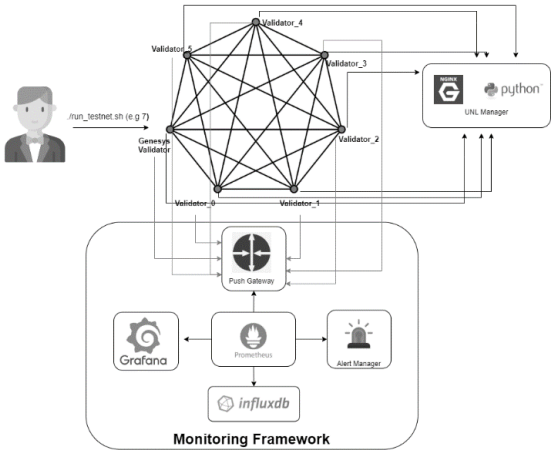

Figure 3. XRP Ledger network topology

Monitoring Services: Monitoring data is essential to understand the behavior of a system. Collecting as much data as possible is a key towards the identification of any system anomalies. Thus, we have identified the need of integrating a monitoring system to enhance the proposed benchmarking framework capabilities. In a nutshell, during the deployment of the benchmarking framework, an extra set of services are spawned forming the monitoring framework. The aim of the monitoring framework is to gather and visualize different data from the transactions performed in the network as well as data regarding the health of the nodes participating in the network. Moreover, the proposed monitoring framework is considered a black box to the blockchain protocol; while someone can build his/her custom metric exporter gathering data based on their needs.

5. Experimental Evaluation

In order to evaluate the performance of the proposed framework in terms of efficiency and ease of use, our experiments have been deployed in a Virtual Machine (VM) running on top of a bare metal server with the following specifications:

- Dell PowerEdge R640 Server
  - Intel® Xeon® Gold 6230 2.1G, 20C/40T, 10.4GT/s, 27.5M Cache, Turbo, HT (125W) DDR4-2933 X 2
  - 40 Cores, 80 Threads
  - 32GB RDIMM DDR4 2666MT/s Dual Rank X4

5.1 XRPL Client Setup

As previously mentioned, our aim was to develop a generic blockchain benchmarking framework; where researchers and/or developers would be able to deploy and test different blockchain protocols with different requirements and constraints. In our initial instantiation of the architecture we have successfully deployed a private XRP Ledger network. As it is depicted in Figure 3 we are able to deploy an XRPL Network consisted with *n* number of nodes/validators. The specifications of the VM used during this instantiation are the following:

- Ubuntu 18.04 LTS
  - Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz (6 Cores)
  - 12 GB RAM DDR4

5.2 Evaluation Metrics

During the evaluation process, several metrics were captured and stored in the monitoring system. Specifically, using the *Server Info* methods, provided by the *rippled* daemon, we retrieve the status of the server. Some of the metrics exposed in the monitoring system were: *ServerLatency, validationQuorum, loadFactor, Peers, Uptime, serverStateDurationUs, convergeTimeS* and *proposers*. Description of these metrics is provided by the XRPL website [25]. Moreover, a *Docker Stats* exporter has been deployed [26] in order to export the default metrics which the docker engine provides by default. Such metrics are: *CPU Usage, Memory Usage, NET I/O, and Block I/O*. These metrics are gathered and stored in an *InfluxDB* in the form of time series data.

5.3 Results & Discussion

For the evaluation process we have performed three experimental scenarios altering the number of validators of the network. Starting from 10 validators, then 20 and 40 respectively, we have measured the mean converge time (i.e., time of the validators to close the last ledger) and how it is changed when more validators are participating in the consensus process but also while more proposers are in the network. Moreover, we have measured the server latency, which defines how each validator performs in terms of load, XRPL defines the server latency as "The amount of time spent waiting for I/O operations, in milliseconds. If this number is not very low, then the *rippled* server is probably having serious load issues."

Before executing the aforementioned experiments, the time to deploy an XRPL network in our benchmarking framework was measured using the built-in method of Linux based systems – time [28]. The latter is depicted in Table 1.

The deployment time of an XRPL network with 10 validators was measured at 1 minute and 4 seconds, while the deployment of a network topology consisted with 65 nodes was measured at 2 minutes and 50 seconds.

Table 1. XRPL network deployment time

Nodes/Validators	Time
10	1m4.727sec
20	1m10.671sec
40	2m11.798sec
60	2m43.028sec
65	2m50.124sec

5.3.1 Experiment 1

During our first experiment, an XRPL network topology was deployed with 10 validators acting as proposers, while 8 of them (i.e., 80% of the Network participants [29]) as validators. The network topology was a full mesh network - all nodes connected between each other. In this setup, using the Traffic Generator Manager, 1 million transactions were submitted to random accounts, exchanging a random amount of XRPs. The process was repeated 5 times while the number of successful transactions, execution time, converge time and server latency were captured and stored. Finally, the mean value of each metric was calculated while the results of this first experiment are depicted in Table 2.

5.3.2 Experiment 2

In the second experiment, the number of nodes/validators were increased to 20. In this case, 20 of the nodes were acting as proposers while 16 of them were acting as actual validators. The experiment was executed 5 times while the same metrics as the first experiment were also captured and visualized. The latter is depicted in Table 3.

5.3.3 Experiment 3

During the third experiment, an XRPL Network topology with 40 nodes/validators was deployed. In this case, the 40 of them were acting as proposers while 33 nodes were acting as validators. The same process as with the previous experiments was followed and repeated 5 times, while the same metrics were captured and visualized. The results of this experiment are depicted in Table 4.

Based on the results of the three experiments, it is realized that the mean time to execute 1 million transactions is increased based on the number of participants but also the number of the actual validators of the network. It is measured at about 2 hours during the first experiment, while it was increased by 2,5 hours when the network was consisted with 20 participants and 3 hours with 40 participants. Moreover, the mean converge time was also increased by 0.1 seconds in the network of 20 participants rather than 10 participants, and by 0.2 more during the experiment with 40 participants. Server Latency was also increased during the three experiments but not at a point where the network could become unresponsive. These outcomes are well justified since BFT algorithms (i.e., Ripple Consensus Protocol) demands more time to come into consensus when there are more participants in the validation quorum.

6. Conclusions

In this work, we proposed an initial setup of a blockchain benchmarking framework, aiming at the provision of the necessary tools to measure but also to visualize different metrics of the blockchain protocol under a test. Moreover, we have demonstrated the integration of the first use case using the *rippled* daemon. In this use case, we have executed three experiments while we have measured the server latency and the converge time of each network participant. Based on our results we conclude that those two metrics are correlated with the number of the network participants while both metrics increased when the number of participants increased. For future work we plan to involve the

integration of more blockchain protocols into our benchmarking framework, while also implement more complex test scenarios using the capabilities of the Connectivity Manager.

7. Acknowledgment

This work has been partially supported by the University Blockchain Research Initiative (UBRI) project, funded by the Ripple's Impact Fund, an advised fund of Silicon Valley Community Foundation (Grant id: 2018-188546).

8. References

[1] T. Ahram, A. Sargolzaei, S. Sargolzaei, J. Daniels, and B. Amaba, "Blockchain technology innovations," in *2017 IEEE Technology and Engineering Management Society Conference, TEMSCON 2017*, Jul. 2017, pp. 137–141, doi: 10.1109/TEMSCON.2017.7998367.

[2] "IEEE Xplore Full-Text PDF:" https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8632190 (accessed Mar. 05, 2021).

[3] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System." Accessed: Mar. 05, 2021. [Online]. Available: www.bitcoin.org.

[4] K. Sultan, U. Ruhi, and R. Lakhani, "Conceptualizing Blockchains: Characteristics & Applications," *Proceedings of the 11th IADIS International Conference Information Systems 2018, IS 2018*, pp. 49–57, Jun. 2018, Accessed: Jun. 22, 2020. [Online]. Available: http://arxiv.org/abs/1806.03693.

[5] K. Okupski, "Bitcoin Developer Reference Working Paper Last changes: 30th July 2016," 2016. Accessed: Jun. 22, 2020. [Online].

[6] M. Vukolić, "Rethinking permissioned blockchains," in *BCC 2017 - Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, co-located with ASIA CCS 2017*, Apr. 2017, pp. 3–7, doi: 10.1145/3055518.3055526.

[7] T. Neudecker and H. Hartenstein, "Network layer aspects of permissionless blockchains," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 1, pp. 838–857, Jan. 2019, doi: 10.1109/COMST.2018.2852480.

[8] Z. Li, A. V. Barenji, and G. Q. Huang, "Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform," *Robotics and Computer-Integrated Manufacturing*, vol. 54, pp. 133–144, Dec. 2018, doi: 10.1016/j.rcim.2018.05.011.

[9] P. Kumari and P. Kaur, "A survey of fault tolerance in cloud computing," *Journal of King Saud University - Computer and Information Sciences*, 2018, doi: https://doi.org/10.1016/j.jksuci.2018.09.021.

[10] O. Boireau, "Securing the blockchain against hackers," *Network Security*, vol. 2018, no. 1, pp. 8–11, 2018, doi: 10.1016/S1353-4858(18)30006-0.

[11] F. Calvão, "Crypto-miners: Digital labor and the power of blockchain technology," *Economic Anthropology*, vol. 6, no. 1, pp. 123–134, Jan. 2019, doi: 10.1002/sea2.12136.

[12] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A Survey of Distributed Consensus Protocols for Blockchain Networks," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, p. 1, Apr. 2020, doi: 10.1109/COMST.2020.2969706.

[13] F. Imbault, M. Swiatek, R. de Beaufort, and R. Plana, "The green blockchain: Managing decentralized energy production and consumption," Jul. 2017, doi: 10.1109/EEEIC.2017.7977613.

[14] T. Tuan *et al.*, "BLOCKBENCH: A Framework for Analyzing Private Blockchains," doi: 10.1145/3035918.3064033.

[15] K. Croman *et al.*, "On Scaling Decentralized Blockchains (A Position Paper) Initiative for CryptoCurrencies and Contracts (IC3) 1 Cornell." Accessed: Nov. 19, 2020. [Online].

[16] C. Decker, R. Wattenhofer, and E. Zurich, *Information Propagation in the Bitcoin Network*. .

[17] "Performance Analysis | Parity Technologies." https://www.parity.io/performance-analysis/ (accessed Nov. 19, 2020).

[18] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the Security and Performance of Proof of Work Blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 3–16, doi: 10.1145/2976749.2978341.

[19] "Performance testing smart contracts developed within VS Code using Hyperledger Caliper – IBM Developer." https://developer.ibm.com/technologies/blockchain/tutorials/blockchain-performance-testing-smart-contracts-vscode-caliper/ (accessed Mar. 24, 2021).

[20] R. Wang, K. Ye, and C. Z. Xu, "Performance Benchmarking and Optimization for Blockchain Systems: A Survey," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Jun. 2019, vol. 11521 LNCS, pp. 171–185, doi: 10.1007/978-3-030-23404-1_12.

[21] D. Saingre, T. Ledoux, and J. M. Menaud, "BCTMark: A Framework for Benchmarking Blockchain Technologies," in *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, Nov. 2020, vol. 2020-Novem, doi: 10.1109/AICCSA50499.2020.9316536.

[22] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Performance Evaluation of the Quorum Blockchain Platform," *arXiv*, Jul. 2018, Accessed: Mar. 24, 2021. [Online]. Available: http://arxiv.org/abs/1809.03421.

[23] "ConsenSys Quorum | ConsenSys." https://consensys.net/quorum/ (accessed Mar. 24, 2021).

[24] "UNIC-IFF/ripple-docker-testnet: Ripple/XRP Private Testnet setup scripts for Docker Engine." https://github.com/UNIC-IFF/ripple-docker-testnet (accessed Nov. 19, 2020).
[25] "https://xrpl.org/server_info.html." https://xrpl.org/server_info.html (accessed Mar. 03, 2021).
[26] "Docker Stats exporter for Prometheus." https://github.com/wywywywy/docker_stats_exporter (accessed Mar. 03, 2021).
[27] "Prometheus - Monitoring system & time series database." https://prometheus.io/ (accessed Mar. 05, 2021).
[28] "How to get script execution time from within the shell script in Linux." https://www.golinuxcloud.com/get-script-execution-time-command-bash-script/ (accessed Mar. 05, 2021).
[29] K. Christodoulou, E. Iosif, A. Inglezakis, and M. Themistocleous, "Consensus Crash Testing: Exploring Ripple's Decentralization Degree in Adversarial Environments," *Future Internet*, vol. 12, no. 3, p. 53, Mar. 2020, doi: 10.3390/fi12030053.

Table 2. Evaluation results of 1st experiment

	Run	Nodes	Validator Quorum	Transactions	Successful Transactions	Time (Hours)	Converge Time (Sec)	Server Latency (Sec)
	1	10	8	1000000	975621	2.058881831	2.7	1
	2	10	8	1000000	976020	2.057368169	2.7	1
	3	10	8	1000000	975554	2.059850487	2.7	1
	4	10	8	1000000	975570	2.058161562	2.7	1
	5	10	8	1000000	975769	2.059236403	2.7	1
Mean Value					**975706.8**	**2.05869969**	**2.7**	**1**

Table 3. Evaluation results of 2nd experiment

	Run	Nodes	Validator Quorum	Transactions	Successful Transactions	Time (Hours)	Converge Time (Sec)	Server Latency (Sec)
	1	20	16	1000000	941914	4.290780171	2.8	1.3
	2	20	16	1000000	942104	4.285953195	2.8	1.3
	3	20	16	1000000	942833	4.289543764	2.8	1.3
	4	20	16	1000000	942147	4.283003006	2.8	1.3
	5	20	16	1000000	942522	4.287276685	2.8	1.3
Mean Value					**942304**	**4.287311364**	**2.8**	**1.3**

Table 4. Evaluation results of 3rd experiment

	Run	Nodes	Validator Quorum	Transactions	Successful Transactions	Time (Hours)	Converge Time (Sec)	Server Latency (Sec)
	1	40	32	1000000	931876	5.234565751	3	1.5
	2	40	32	1000000	931673	5.276554455	3	1.5
	3	40	32	1000000	931871	5.128935424	3	1.5
	4	40	32	1000000	932792	5.243675606	3	1.5
	5	40	32	1000000	933779	5.986525675	3	1.5
Mean Value					**932398.2**	**5.374051382**	**3**	**1.5**