Simulation of the Internet Computer Protocol: the Next Generation Multi-Blockchain Architecture

Luca Serena[†], AoXuan Li[*], Mirko Zichichi[‡†], Gabriele D'Angelo[†], Stefano Ferretti[§], Su-Kit Tang[*]
[*]Faculty of Applied Sciences, Macao Polytechnic University, Macao SAR, China
[†]Department of Computer Science and Engineering, University of Bologna, Italy
[‡]Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
[§]Department of Pure and Applied Sciences, University of Urbino "Carlo Bo", Italy
{luca.serena2,gabriele.dangelo}@unibo.it, {aoxuan.li,sktang}@mpu.edu.mo
mirko.zichichi@upm.es, stefano.ferretti@uniurb.it

***Abstract*—The Internet Computer Protocol is a new generation blockchain that aims to provide better security and scalability than the traditional blockchain solutions. In this paper, this innovative distributed computing architecture is introduced, modeled and then simulated by means of an agent-based simulation. The result is a digital twin of the current Internet Computer, to be exploited to drive future design and development optimizations, investigate its performance, and evaluate the resilience of this distributed system to some security attacks. Preliminary performance measurements on the digital twin and simulation scalability results are collected and discussed. The study also confirms that agent-based simulation is a prominent simulation strategy to develop digital twins of complex distributed systems.**

***Index Terms*—Simulation, Performance Evaluation, Internet Computer Protocol, Blockchain, Agent-based Simulation.**

I. Introduction

Blockchains and more in general Distributed Ledger technologies (DLTs) are successful examples of distributed systems with a relevant impact on both economy and computer science. As always happens with any technology, DLTs have some relevant drawbacks but also some interesting advantages that can be used for building a new generation of distributed systems. To name a few: decentralization, data immutability, transparency, no third parties involved in transactions are few examples of properties that can be useful in the design and implementation of some specific systems aimed to provide new services to the final users or to improve the security aspects of supply-chains (both for physical and virtual goods, such as the software).

After a first generation of blockchains, corresponding to the initial success of Bitcoin, many improved blockchain platforms have been designed both to support new cryptocurrencies and for more generic purposes (e.g., Ethereum), succeeding in significantly improving some known problems. Limited scalability, the amount of time required to validate transactions recorded in the blockchain, decentralized management of blockchain updates and its controlling organization are all known problems of the first proposed blockchains.

The research on these topics is really active and even more advanced blockchain solutions are currently investigated, proposed and partially deployed. For example, the Internet Computer Protocol (ICP) architecture[1] is based on a network of networks aimed at combining the resources of several computers and distributing computation. This happens by means of a protocol that supports the reading, replication, modification, and procurement of decentralized applications.

One of the goals of the ICP is to coordinate a distributed system that is composed of many independently-operated data centers. The obtained system is then able to provide a general-purpose abstract platform that is largely transparent for end-users and supports distributed computations. In other words, the ICP makes it possible to easily build decentralized applications that run smoothly on this new multi blockchain-based distributed infrastructure. Multi blockchain means that the whole IPC network is partitioned in shards and each shard can be considered as a fully functioning blockchain, processing disjoint sets of transactions. Under the hood, this abstraction is made possible thanks to reliable message delivery, transparent accountability, and resilience that must be provided by the ICP and its embedded mechanisms.

The employment of new technological features goes along with the need to test and verify the proposed innovations before the final deployment. Modeling and simulations are thus powerful tools for performing tests on security, scalability, and sustainability of the designed technologies, providing sometimes useful insights about the feasibility of the innovations and the potential problems of the system. For these reasons, we aim to provide a digital-twin of the current ICP implementation that is based on efficient simulation techniques and that enables the developers to support the design and deployment of ICP by means of the so-called "what-if" analysis. In practice, the digital twin of the ICP is a simulator that is able to mimic the relevant behaviors of the ICP architecture and that can be used to investigate some specific problems (e.g., scalability and resilience to attacks). A preliminary design of the digital twin of the ICP has been proposed by the authors in [1]. From our point of view, the ICP is a good example of next-generation blockchain to be simulated since it embodies the advanced features that sharded/multi-blockchains promise to deliver [2].

[1]Authors are not sponsored or affiliated in any way with the DFINITY Foundation which is the not-for-profit organization that develops the Internet Computer Protocol.

978-1-6654-3326-6/21/$31.00 © 2021 IEEE

This paper is structured as follows. Section II provides the necessary background about the technologies that are at the basis of ICP (e.g., blockchain and the consensus protocols) and a discussion of the related work. Section III introduces the ICP distributed architecture; while in Section IV, the proposed ICP simulator is described both in terms of overall architecture and implementation. In Section V, the preliminary performance results about the developed ICP digital twin are reported and discussed together with a scalability evaluation of the proposed simulator. Finally, Section VI provides the concluding remarks.

II. Background and Related Work

In this section, we briefly describe the background technologies and methodologies that are necessary for understanding the ICP architecture, and then we discuss the related work.

A. Blockchains

A blockchain is a ledger, distributed among a set of network nodes, which generally stores information encoded as a set of ordered transactions. These transactions are grouped into blocks, then each block is linked to the previous block using its hash digest value. This chain of blocks composed using hash links to previous blocks makes tampering with the ledger very unlikely. The first technology implementing a blockchain was Bitcoin [3], in which transactions are mostly used to transfer value from one account to another. Ethereum [4], is considered as a second generation blockchain because it also supports *smart contracts* in its transactions, enabling thus the creation of decentralized applications.

A blockchain grows by appending new blocks. It is essential that all network nodes are working on the same blockchain. The majority of network nodes have to agree on the validation of transactions that are included inside the new block and the block itself. Moreover, they may also decide who is responsible for creating the new block. Therefore, a blockchain defines a set of protocols called *consensus algorithms*, which ensure all (or the majority of) network nodes agree on the present state of the blockchain and append the same block.

B. Consensus Protocols

A consensus protocol is concerned with the reaching of consensus among the networking peers for system availability and reliability. To reach consensus, mechanisms for incentivising the partaking network nodes have shown good enough results for such systems to remain reliable over time [5]. Namely, the blockchain network nodes use a protocol to reach a consensus whether a block is well-formatted and all transactions within are validated. If more than a single block is proposed to be appended at the same time, the network nodes have also to decide which block is appended to the chain. The wildly adopted protocols are proof-based algorithm and vote-based algorithm [6]. Under the proof-based algorithm, a network node may append a block only if it can solve a cryptographic puzzle and prove it to other nodes, e.g., Proof-of-Work. Under the vote-based algorithm, a block is appended to the chain only if a large enough part of the nodes proposes the same block, e.g., Proof-of-Stake.

C. Related Work

The ICP is a new technology that has been recently proposed and therefore specific simulators are not already available. Despite this, some simulators have already been proposed for the modeling and performance evaluation of blockchains and distributed ledger technologies. The different simulators mainly differ in the simulation methodology that is used and in the level of detail at which the system is modeled. Both these aspects have a very relevant influence on the simulation outcomes in terms of accuracy, and consequently, on the performance of the simulator.

For example, agent-based simulation is used in [7]. In this paper, some of the authors of this paper have modeled a generic blockchain (loosely based on the main Bitcoin characteristics) to study the likelihood and the effects of some typical network attacks to blockchains.

BlockSim [8] is a Python simulator that follows a discrete-event methodology. It aims to specifically consider the modeling and simulation of block creation through the proof-of-work consensus algorithm.

VIBES [9] is another simulator that aims to model and evaluate large-scale peer-to-peer networks and that it is able to simulate blockchain systems beyond Bitcoin and to support large-scale simulations with thousands of nodes.

A different approach is used in [10]. In this case, the modeling of the blockchain system is obtained by creating a ”high level” model that is based on queuing theory.

Finally, in [11], Monte Carlo simulations are used for building stochastic blockchain models.

Depending on the expected outcomes, the different simulation methodologies can be more appropriate. For the modeling of the ICP, we think that an agent-based approach is best choice since we need to consider some specific medium-level details of the communication protocol used in the distributed system and, a the same time, to provide an easy-to-use abstraction (i.e., the agents) to be understood by developers and researchers that lack of specific simulation expertise.

III. The Internet Computer Protocol (ICP)

The Internet Computer Protocol is a novel design of scalable blockchain that supports smart contracts (supposedly) at web speed [12]. It has been proposed as a response to the well known limits of current mainstream blockchains, with respect to limited scalability and high transaction fees [12]. Ethereum is currently the most popular smart contracts based blockchain, however, the latency for operating with this technology can greatly vary depending on the transaction fees and/or on the levels of supply and demand in the network. Generally, we can expect a latency between 30 and 60 seconds and that might not be optimal (or acceptable) for all kinds of applications [13].

The ICP proposed *Chain Key Cryptography* [14], [15] algorithm, which helps the nodes in the blockchain to efficiently verify smart contracts and generate a new block

without synchronizing the full blockchain. It also introduced a decentralized autonomous government system, called *Network Nervous System (NNS)* [16], aimed to manage the blockchain and its evolution.

A. The Protocol

The Internet Computer is a network of interacting replicated state machines. Each state machine executes programs in discrete rounds: in each round, it takes an input, applies a state transition function to the input and the current state, it gets an output, and then obtains a new state. The state transition function of the Internet Computer is a universal function, capable of executing arbitrary Turing-complete programs, i.e., the Canisters, which are the equivalent of smart contracts. A fundamental characteristic of this distributed architecture is that the state machine is replicated in a subnet of nodes called replicas. Each replica executes a copy of the same state machine. Since a subnetwork must continue to function properly even if some replicas are faulty, and it is essential that inputs be processed in the same order, replicas in a subnet must execute a consensus protocol. This consensus protocol is based on a blockchain and on a Byzantine fault tolerance algorithm. More specifically, the Chain Key Cryptography is a group of cryptography protocols and primitives including threshold signatures, public key encryption, and non-interactive zero-knowledge proofs. We omit the detailed construction of Chain Key Cryptography, and one may refer to [15] for further information. Briefly, the Internet Computer protocol verifies all transactions with a single public key, and traversing through the full blockchain is a redundant activity.

B. Governance

The Internet Computer has a mix of permissioned and permissionless governance, called the DAO-controlled network model. While each subnet runs a permissioned consensus protocol, a permissionless Decentralized Autonomous Organization (DAO) determines rules and permissions of each subnet. The DAO of the ICP is specifically called the Network Nervous System (NNS) and it is governed through a Proof-of-Stake (PoS) consensus mechanism, where the voting power is determined by how much native token a member has staked (native tokens and their usage will be discussed in the following of this section).

NNS is a critical part of the ICP architecture, which oversees the network and governs all users and applications on the network. The system itself is open and decentralized, which means any user may participate in the governance process. Some notable features of NNS are:

- Upgrading the ICP.
- Registering new users.
- Deciding which entities provide replicas.
- Configuring the topology of the network.
- Providing a public-key infrastructure.

The NNS is realized by a set of canisters on a special system subnet. The registry canister stores the configuration of the IC, such as which replicas belong to which subnet and the public keys associated with subnets and individual replicas. The governance canister manages the decision making and voting. The ledger canister keeps track of the users' native token accounts and its transactions. The native token is an utility token called ICP and has three functions: (i) representing the governance voting power; (ii) minting new value as voting reward; (iii) incentivizing subnets to create and operate with canisters. Users can convert ICP to cycles (i.e., burned), and these cycles are used to pay for Canisters' operations.

C. Architecture Overview

The ICP has been designed as a four-layer architecture. The top level of ICP is the **canister**, which is functionally equivalent to the smart contract on other blockchains. The hosting and execution of canisters happens on **subnets**, and a subset is a composite of **nodes** placed in different (and independent) **data centers**. Each node is a physical computer providing storage and computation capability to the ICP. Those nodes are further organized and operated on different data centers. As for now, in the current ICP deployment, there are independent data centers across Asia, Europe, and North America, and each data center hosts up to 28 nodes [17]. Figure 1 reports a high-level representation of this design structure. At the time of writing, there are 65 data centers, 35 subnets, and 650 node machines. Each subnet has 13 nodes, and the NNS subnet has 40 nodes.

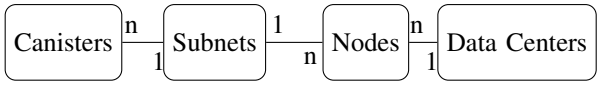

Fig. 1. The ICP high-level design architecture.

To achieve the needed fault tolerance, the canisters are replicated across different nodes within a subnet. More specifically, each replica runs the same canister and jointly notarizes the final result. Even in case some nodes experience a failure or are corrupted by malicious users, as long as there are large enough active nodes, then the canister remains available. In ICP, fault tolerance is guaranteed for the *Byzantine Faults* [18].

D. Consensus

Each subnet implements a consensus mechanism to order the inputs so that all replicas in a subnet can process such inputs exactly in the same order. This can be guaranteed when in a subnet of n replicas, at most $f < \frac{n}{3}$ of the replicas, are faulty or Byzantine. We emphasize the most critical concepts in this section and refer to [19] for more details.

As already reported, the ICP consensus protocol is based on a blockchain. A tree of blocks is grown, starting from a special genesis block that is the root of the tree. Each block contains a payload, composed of (i) sequence of inputs and (ii) the hash of the block's parent in the tree. As the protocol progresses, there is always a path of finalized blocks in this tree, i.e., the chain of blocks.

The blockchain grows in rounds. In round h of the protocol, one or more blocks of height h are added to the tree. During

each round, each replica is assigned a unique rank pseudo-randomly. The replica of lowest rank is the leader of that round and will propose a block to add to the tree. If the leader is honest and all replicas are synchronized, there is one and only one block added to the blockchain. However, if the leader is malicious, or there are some network delays across the replicas, more than one leader might propose blocks at the same round. The tree will continue to grow each round, such that when the finalized path is extended in round h, the finalized path will be of length h. Thus, although the time for the response of the system to external inputs occasionally increases due to faulty replications, the throughput of the protocol should remain essentially constant. The ICP introduces **notarization** and **finalization** protocols to avoid such collisions and to guarantee the block grown.

Notarization ensures that there is a published and valid block in each round. In each round, some of the state of a subnet will be certified. This per-round certified state is validated using a threshold signature. For each proposed block, a replica will decide if the block is valid, and the replica will notarize the block and broadcast its notarization message if so. If more than two-thirds of replicas support the notarization, the block becomes notarized. A block is appended to the blockchain only if it is notarized.

In some cases, there could be more than one notarized block in a round, and the replicas have to decide which one they agree on. If a replica notarized only one block B in this round, it then finalizes block B by broadcasting a finalization message "I notarized and only notarized block B." A block is finalized if more than two-thirds of replicas support the finalization. The blockchain will only keep the finalized block and drop all other blocks in the same round.

E. Message Routing

Updating and maintaining the replicated state of a subnet is performed thanks to the message routing layer and the execution layer. Indeed, it is essential that all replicated state machines are updated in a completely deterministic fashion. The message routing layer handles two different types of messages: (i) ingress messages and (ii) cross-subnet messages. Ingress messages are messages from external users, while cross-subnet messages are from canisters running on other subnets.

The input messages are processed by the execution layer. This updates the state of the canisters on the replicated state machine and generates the corresponding output messages. These outputs are then processed again by the message routing layer. We can also distinguish between two types of outputs: (i) ingress message responses, that are responses to ingress messages which may be retrieved by external users; (ii) cross-subnet messages, generated to contact again another subnet, maybe in a chain of cross-subnet execution requests.

When a canister starts running, it has one input queue for ingress messages and various queues for each canister it is communicating with. During each round, the canister consumes some inputs from queues and updates the replica states. The output message will be transferred through a message routing protocol, which could be input for other canisters.

When external users want to access/make a request to a canister on the ICP, they first send a request to the boundary nodes. This boundary node represents the entry point for the ICP [12]. The boundary nodes then authenticate the request and relay the request to the corresponding subnet. Boundary nodes also provide caching, rate limitation, and denial of service protection. Notably, all communication through boundary nodes uses Transport Layer Security (TLS) for confidentiality, integrity, and authenticity.

Now that we have introduced the main characteristics of the ICP architecture, we can focus on the simulator that we have built as a digital twin of the ICP.

IV. SIMULATOR

The tool employed for building and running our experiments is LUNES (Large Unstructured NEtwork Simulator), an agent-based time-stepped simulator that allows for modeling the behaviour of a large number of simulated entities, which can communicate with each other via messages exchange. The software is well suited to simulate protocols that run in distributed environments, and it is designed to also support parallel and distributed execution, in order to properly manage simulations with a large number of simulated entities if necessary. LUNES has been designed to be easily expandable, allowing the users to customize the features of the agents and their behaviour in response to events.

Specifically for the Internet Computer, we used the simulator to reproduce the life-cycle of the ICP protocol, and to test its conduct under particular conditions. There are several key parameters of the model, whose values can have a significant influence on the behaviour of the system. Hereafter we report the most important:

- *DATA CENTERS*, which indicates the number of separate locations where the nodes of the system are located. In our default configuration, we set this value to 40, which is the current number of data centers of the ICP.
- *SUBNETS*, which indicates the number of subnets that constitute the ICP. In our default configuration, we set this value to 32, which is the current number of subnets that make up the ICP.
- *BOUNDARY NODES*, which indicates the number of boundary nodes of the system. According to real data, this value is currently set to 10.
- *NODES PER SUBNET*, which indicates the number of active nodes for each subnet. Again, in our default configuration we took inspiration from the real values of the ICP, where there are 13 nodes for each subnet. The exception is the subnet number 0, the NNS, which is made up of 40 nodes.
- *CLIENTS*, which indicates the number of clients that create and spread transactions in the system.
- *CROSS SUBNET INTERACTIONS*, which indicates the number of cross transactions required for each user

transaction. Of course, this value is not stable for all the transactions but for sake of simplicity we kept it constant, in order to evaluate how the number of cross transactions affect the performances of the protocol
- *BLOCK FREQUENCY*, which express, in hundredth of seconds, the average time required to validate a block (i.e., every *BLOCK FREQUENCY* steps, the block validation procedure starts)
- *DATA CENTERS DOWN*, which indicates the number of data centers currently not active.
- *ERROR RATE*, which indicates the probability for a block to be created incorrectly.

The total number of agents in the simulator is equal to $40 + (SUBNETS-1)*NODESPERSUBNET+CLIENTS$. Clients are special simulated entities that only have the task of creating requests and to forward them to one of the boundary nodes (the life cycle of a request is represented in Figure 2), while all the other agents are nodes of the ICP, and thus they are characterized by a specific subnet and data center of belonging. These agents also maintain locally a list of blocks and a list of non-finalized transactions and they are directly in contact with all the other nodes of the same subnet, with which they frequently exchange information about requests and blocks.

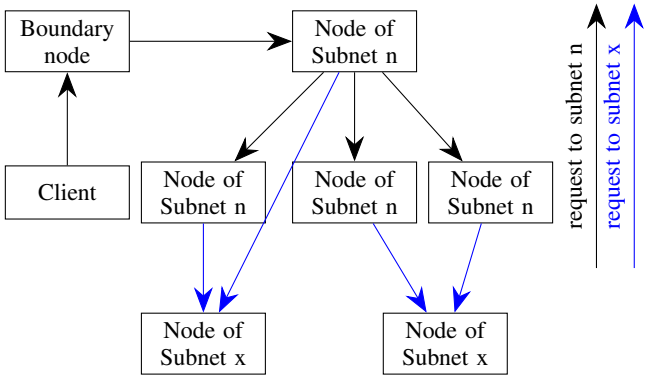

Fig. 2. Life cycle of a request before being inserted into one block

To evaluate the performance of the simulated system, we mainly made use of two metrics: the *delay*, which is the time between the creation of a transaction and its insertion in a finalized block and the *finality rate*, which is the percentage of transactions that actually ends up in a valid block.

A. Simulation Execution

Before starting the execution with LUNES, a preliminary phase is necessary in order to create two files, listing respectively the features of the nodes (i.e., which subnet they belong to and where are they located) and the latency between two locations. In the current version of the simulator, these parameters are generated randomly, but in a future extension, for sake of veracity, we could use measurements on the real ICP system to obtain more realistic configurations.

After the initialization phase, the execution of the simulation proceeds as follows:

- At each time-step, there is a certain percentage for the clients that generate a transaction.
- Transactions are directed to a boundary node, that in turn forwards each of them to an active node in the required subnet.
- Once the node of the required subnet receives the transactions, it forwards it to the other nodes of the subnet. Furthermore, if cross-requests are needed, the required cross-requests are sent to an active node of such subnets.
- At the beginning of each round, the block generation process starts. The nodes agree on who is the leader, which (if active) will create the block inserting all the transactions that do not have unsatisfied cross-requests. The block is then forwarded to the other nodes in the subnet. If the leader is not active (or it sends an incorrect block) then another node will be elected as a leader after a short time, following the same procedure.
- Once a correct block is received, a notarization message is created and forwarded among all the nodes of the subnet.
- As soon as $2/3*NODESPERSUBNET$ notarization messages are received, the nodes send a finalization request for the same block.
- When $2/3*NODESPERSUBNET$ finalization messages are received, the block is finalized and its transactions can finally be considered as validated.

The time granularity of the model (i.e., the size of each time-step) is one hundredth of a second, and we assume that the communication time for a message is between 0.01 sec and 0.2 sec. This assumption is based on a set of measurements that we obtained by using the tool provided in [20].

V. Experiments

In our experiments, we tried to evaluate the influence of certain parameters on the performance of the ICP, such as the error rate, the number of non-working data centers, the number of cross-requests required for each transaction, and the number of nodes per subnet.

If not specified otherwise, the following experiments were carried out with 2 cross-subnet interactions, a block frequency of 10 seconds and, as in the deployed ICP system, 32 subnets, 40 data centers and 13 subnets per node.

In Figure 3 is shown that the error rate is irrelevant as regards the finality rate that is achieved. In fact, such a parameter would only affect the delay, as reported in Figure 4. On the other hand, the number of non-functioning data centers strongly affect the finality rate, since if $1/3$ of the nodes of a subnet are located in inactive data centers, such a subnet will not be able to notarize and finalize blocks. However, in Figure 3, we can also notice that the plots of '0 nodes down' and '5 nodes down' overlap, because with so few data centers down the 100% of finality rate is always achieved. On the other hand, increasing the parameters leads to configuration where certain subnets do not have enough active nodes to carry on the blocks validation activities.

Figure 5 shows that, similarly to the average delay, also the variance of the delay is strongly influenced by the error

rate, while the impact of the number of inactive data centers is almost negligible.

The need of performing cross-requests has also a limited influence on the delay, due to the additional interactions with nodes belonging to other subnets. In Figure 7, we can notice, in fact, that in the scenario with 0 cross-requests required, the average delay is slightly smaller with respect to the other cases. Since the cross-requests are executed in parallel [21], the overhead for performing cross-requests does not rise with the increase of the parameter (i.e., the only significant difference is between 0 and 1 or more).

In Figure 6, it is shown the impact of the number of nodes per subnet. The most relevant differences occur when the number of non-working data centers is high, with a finality rate approaching 0% with 100 nodes per subnet and 20 inactive data centers. In fact, with a great number of nodes it is unlikely to have at least $2/3$ of the nodes in correctly-working data centers, while in the 13-node-per-subnet scenario the statistical variance makes it possible to have at least 9 working nodes for some subnet.

Finally, we report a preliminary evaluation of the simulator performance. Executing the simulator runs, we noticed that the only parameter that significantly affected the execution time is the number of simulated nodes. This increase in the amount of time required to complete the simulation runs is caused by the large increase in the number of messages exchanged between the nodes belonging to the same subnet, as reported in Table II. In Table I, it is shown the correlation between the number of nodes per subnet and the time required to perform 100000 time-steps of the simulation (i.e., 1000 seconds of simulated time). The reported experiments have been executed under a sequential version of the software, since with our default configuration (i.e., with 13 nodes per subnet and 40 subnets), the computational effort is relatively small and the simulation is mostly communication-bound. However, the simulator could be easily converted to a parallel/distributed approach if necessary for speeding up the execution.

TABLE I
CORRELATION BETWEEN THE NUMBER OF SIMULATED ENTITIES AND THE EXECUTION TIME OF THE SIMULATOR.

Nodes per subnet	**13**	**26**	**50**	**100**
Execution time	2.7 sec	12.3 sec	43 sec	470 sec

TABLE II
CORRELATION BETWEEN THE NUMBER OF SIMULATED ENTITIES AND THE NUMBER OF DELIVERED MESSAGES. TESTS WITH 2 CROSS-REQUESTS, 1% ERROR RATE AND 5 DATA CENTERS DOWN.

Nodes per subnet	**13**	**26**	**50**	**100**
Delivered Messages per second	14 763	58 563	216 723	773 546

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we showed that agent-based simulators are well suited to reproduce the behaviour of a blockchain, due to the opportunity to represent each node involved in the validation process and to model their behaviours. Specifically, we focused on the Internet Computer Protocol, a new-generation DLT where multiple sub-blockchains are run in parallel, allowing for a more efficient management of the users' transactions. LUNES, the software employed for modeling and simulating the ICP, has proved to be highly efficient, allowing us to simulate hours of blockchain activity in only a few seconds. The tool has also exhibited good scalability properties, even though with the increase of the simulated entities the execution time significantly increases due to the underlying characteristics of the communication strategy implemented in the ICP (i.e., gossip).

From our tests, it has emerged that both the error rate in block creation and the number of non-working data centers have a significant impact on the metrics, affecting respectively the delay and the finality rate. On the other hand, increasing the

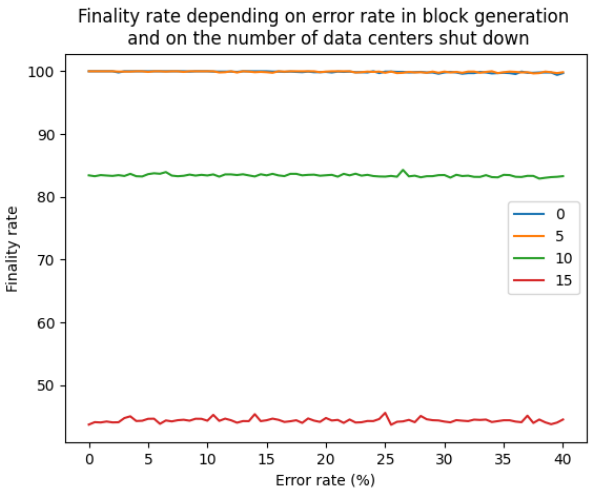

Fig. 3. Finality rate achieved depending on the error rate and number of data centers down.

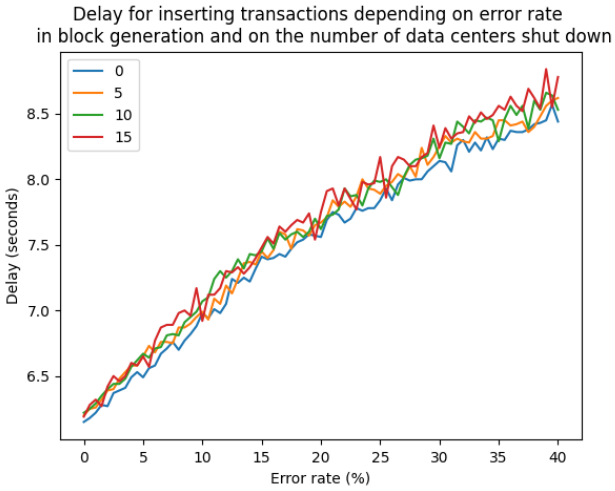

Fig. 4. Average delay depending on the error rate and number of data centers down.

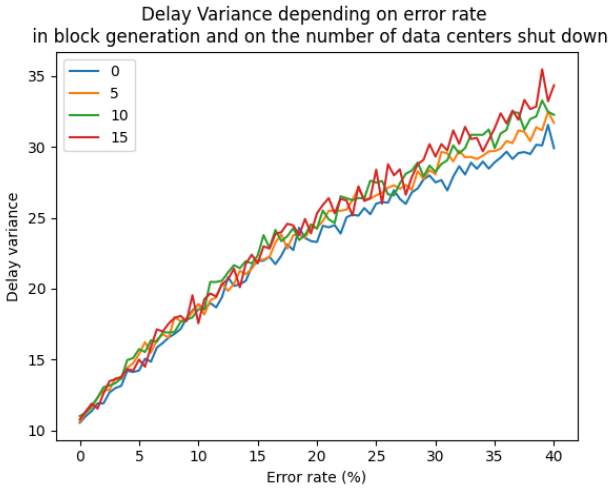

Fig. 5. Variance of the delay depending on the error rate and number of data centers down.

Fig. 7. Delay depending on the number of cross-requests. Error rate set as 0 for these tests.

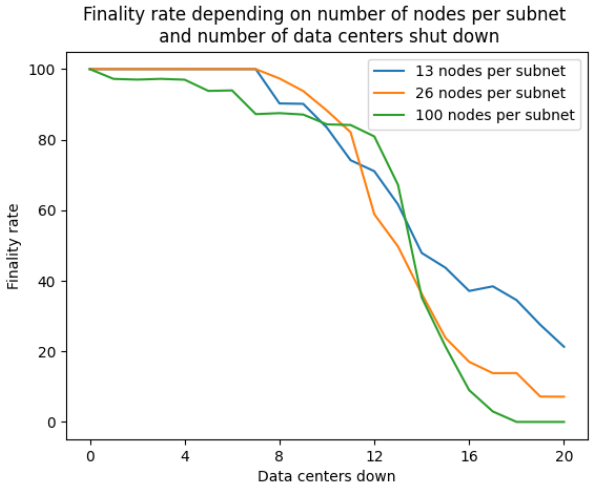

Fig. 6. The number of nodes per subnet has a significant influence on the final results. Error rate set as 0 for these tests.

nodes that participate in the system might probably improve the robustness of the blockchain, but this does not imply a greater resilience against data centers failure.

In future works, we might also examine other potentially problematic aspects of the system. For instance, other than considering errors in block creation, we could evaluate how the system would work in case of loss of messages. Furthermore, we could investigate the possible benefits of employing a dissemination protocol to forward messages among the nodes of the same subnet. Currently, due to the small number of validating nodes in the systems, nodes forward messages according to a pure broadcast strategy (i.e., by sending the message to all the other nodes). If the number of nodes per subnet will be increased in the future, the adoption of specific dissemination protocols could lead to a more efficient network traffic management. Another aspect that could possibly be tested is the economical sustainability of the system, considering factors such as the price of the tokens, the cost for inserting transactions, the reward for blocks creation, etc. Another aspect that will be addressed is the validation of the current simulator with respect to the deployed ICP architecture.

We argue that investigations over these kinds of DLTs could be important to evaluate some of the potential issues regarding the new generation of blockchains, since we are progressively moving toward the adoption of technologies that provide for better guarantees in terms of scalability, speed in transactions' validation and decentralized governance.

REFERENCES

[1] A. Li, L. Serena, M. Zichichi, G. D'Angelo, S.-K. Tang, and S. Ferretti, "Modelling of the internet computer protocol architecture," in *To appear in BLOCKCHAIN'22: 4th International Congress on Blockchain and Applications*, 2022.

[2] Y. Liu, K. Qian, K. Wang, and L. He, "Effective scaling of blockchain beyond consensus innovations and moore's law: Challenges and opportunities," *IEEE Systems Journal*, vol. 16, no. 1, pp. 1424–1435, 2021.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[4] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, 2014.

[5] G. Beuster, O. Leistert, and T. Röhle, "Protocol," *Internet Policy Review*, vol. 11, no. 1, 2022.

[6] G.-T. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," *Journal of Information processing systems*, vol. 14, no. 1, pp. 101–128, 2018.

[7] L. Serena, G. D'Angelo, and S. Ferretti, "Security analysis of distributed ledgers and blockchains through agent-based simulation," *Simulation Modelling Practice and Theory*, vol. 114, p. 102413, 2022.

[8] M. Alharby and A. van Moorsel, "Blocksim: A simulation framework for blockchain systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, p. 135–138, jan 2019.

[9] L. Stoykov, K. Zhang, and H.-A. Jacobsen, "Vibes: Fast blockchain simulations for large-scale peer-to-peer networks: Demo," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*, ser. Middleware '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 19–20.

[10] R. A. Memon, J. P. Li, and J. Ahmed, "Simulation model for blockchain systems using queuing theory," *Electronics*, vol. 8, no. 2, 2019.

[11] P.-Y. Piriou and J.-F. Dumas, "Simulation of stochastic blockchain models," in *2018 14th European Dependable Computing Conference (EDCC)*, 2018, pp. 150–157.
[12] D. Team *et al.*, "The internet computer for geeks," *Cryptology ePrint Archive*, 2022.
[13] L. Zhang, B. Lee, Y. Ye, and Y. Qiao, "Evaluation of ethereum end-to-end transaction latency," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2021.
[14] J. Groth, "Non-interactive distributed key generation and key resharing," Cryptology ePrint Archive, Report 2021/339, 2021, https://ia.cr/2021/339.
[15] Dfinity, "Chain key cryptography: The scientific breakthrough behind the internet computer," Jan 2022. [Online]. Available: https://medium.com/dfinity/chain-key-technology-one-public-key-for-the-internet-computer-6a3644901e28
[16] ——, "The network nervous system: Governing the internet computer," Sep 2021. [Online]. Available: https://medium.com/dfinity/the-network-nervous-system-governing-the-internet-computer-1d176605d66a
[17] ——, "Internet computer network status," June 2022. [Online]. Available: https://dashboard.internetcomputer.org/centers
[18] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making byzantine fault tolerant systems tolerate byzantine faults," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'09. USA: USENIX Association, 2009, p. 153–168.
[19] J. Camenisch, M. Drijvers, T. Hanke, Y.-A. Pignolet, V. Shoup, and D. Williams, "Internet computer consensus," Cryptology ePrint Archive, Paper 2021/632, 2021, https://eprint.iacr.org/2021/632. [Online]. Available: https://eprint.iacr.org/2021/632
[20] WonderNetwork, "Global ping statistics," 2022, https://wondernetwork.com/pings.
[21] Dfinity, "What is the internet computer?" June 2022. [Online]. Available: https://internetcomputer.org/docs/current/concepts/what-is-ic/