IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 12, 2021, accepted July 26, 2021, date of publication July 28, 2021, date of current version August 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3101044

BlockPerf: A Hybrid Blockchain Emulator/Simulator Framework

JULIEN POLGE[1], SANKALP GHATPANDE[1], SYLVAIN KUBLER[2], (Member, IEEE), JÉRÉMY ROBERT[3], AND YVES LE TRAON[1], (Senior Member, IEEE)

[1]Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, L-1359 Esch-sur-Alzette, Luxembourg
[2]CNRS, CRAN, UMR 7039, Université de Lorraine, 54506 Vandoeuvre Cedex, France
[3]Cebi Luxembourg S.A., L-7327 Steinsel, Luxembourg

Corresponding author: Julien Polge (julien.polge@uni.lu)

This work was supported by the Luxembourg National Research Fund through the Stability 4.0 under Grant BRIDGES19/IS/13706587.

ABSTRACT Blockchain is increasingly used for registering, authenticating and validating digital assets (financial assets, real estate, *etc.*) and transactions, governing interactions, recording data and managing identification among multiple parties in a trusted, decentralized, and secure manner. Today, a large variety of blockchain technologies is expanding in order to fulfill technical and non-technical needs and requirements. Within this context, determining and most importantly evaluating the characteristics/performance of a given blockchain platform is crucial for system designers before deploying it. A number of blockchain simulators have been proposed in the literature over the past few years, as reviewed in this paper, but are often limited in several respects (lack of extensibility, do not allow for evaluating all aspects of a blockchain…). This paper extends and improves a state-of-the-art simulator (BlockSim) into a new simulator called "BlockPerf" to overcome those limitations. Both simulators are compared based on a real-life (benchmarking) Bitcoin scenario, whose results show that BlockPerf provides more realistic results than BlockSim, improving by around ≈50% (in average) the outcomes.

INDEX TERMS Blockchain, simulation, emulation, peer-to-peer, consensus, performance.

I. INTRODUCTION

Blockchain is increasingly applied to all sectors of our daily life, spanning from financial applications [1], [2] to industrial ones [3], [4]. Blockchain technology is a type of distributed ledger technology (DLT) that uses a ledger stored in a distributed manner and shared among its participants in the network [5].

Decentralization, consistency, anonymity and traceability are its intrinsic features making it an interesting technology for many applications in which such aspects must be tackled. However, it is never an easy task for researchers, developers, and practitioners to decide what blockchain technologies) they should select/implement, as application requirements may significantly vary from one application to another (e.g., in terms of what data should be stored, the number of transactions to be performed, etc.), without speaking about the multiple constraints of networking, computing power and communication that the application may face [6]. Several blockchain performance assessment frameworks have been proposed in the literature to overcome this difficulty, as presented in [7]–[9], but yet they often focus on functional aspects (e.g., type of consensus, support of smart contracts) and consider fixed performance values (collected from the literature) for dynamic criteria such as throughput, latency, block validation time, etc. The reason for this is that it is complex to formalize and model the performance of such dynamic criteria [6], [10], as they depend on a wide range of parameters and inter-dependent layers, as described in the six-layer model presented in FIGURE 1. Looking at significant blockchain technologies such as Bitcoin [11], Ethereum [12], or Hyperledger Fabric [13], layers differ from one framework to another, and even inside a given framework, different parameter configurations are made possible. Therefore, it is essential to evaluate the blockchain performance based on both real-life and/or simulated environments, without which the blockchain selection process cannot be optimal.

When looking at papers that experimentally evaluate blockchain performance, the majority requires the implementation of the whole system (i.e., using a large set of computers/machines) [14], which also applies to well-known practical/benchmarking tools such as BlockBench [15] or

The associate editor coordinating the review of this manuscript and approving it for publication was Jesus Felez.

Hyperledger Caliper. Such approaches/tools incur high costs for deployment, lack of scalability (e.g., to carry out large-scale experiments) and modularity. On the other hand, simulators can help to deploy and test blockchain technologies in large-scale infrastructure settings. To date, several blockchain simulators exist (e.g., BlockSim [16], PeerSim [17], Shadow [18], Vibes [19], etc.), but they are often limited in several respects. Recent literature reviews of existing blockchain simulation tools [20], [21] point out the fact that those tools often limit themselves to evaluate part of the blockchain system (i.e., they fail in covering all layers and associated performance metrics (a) to (o) emphasized in FIGURE 1). This is particularly stressed by Paulavicius *et al.* [20] in their systematic review and empirical survey of blockchain simulators, in which the authors conclude that *"there is no 'one-size-fits-all' Proof-of-Work blockchain simulator that is able to accurately simulate all the layers"*. To overcome such a limitation, a new hybrid blockchain emulator/simulator called *BlockPerf* is proposed, which extends the BlockSim simulator proposed by Faria and Correia [16]. One of the main improvement lies in the fact that BlockPerf relies on real network infrastructure (at the Network layer) while simulating the upper layers, leading to more realistic results than existing simulators.

State-of-the-art simulators and the extent to which they cover the six-layer model and associated performance metrics are reviewed and discussed in section II. The architectural design of BlockPerf, and how it extends BlockSim, is presented in section III. In section IV, a performance comparison analysis between BlockPerf and BlockSim is carried out based on a real-life (benchmarking) Bitcoin scenario; the conclusions and BlockPerf limitations being discussed in section V.

II. BACKGROUND AND RELATED WORK

As previously discussed, evaluating a blockchain technology or platform turns to be a complex process due to the various inter-dependent layers and associated parameters. To better explain this complexity, a slightly adapted version of the model introduced in [15] is considered in this paper, which corresponds to the six-layer model given in FIGURE 1. Although one may find other blockchain abstraction models in the literature, as the ones proposed by the ITU and ISO standardization bodies [22], [23], we adopted this six-layer model as it covers most aspects of a blockchain (DLT) platform, and it is straightforward to understand. Nonetheless, to allow readers to understand to what extent this model covers the ongoing ITU and ISO blockchain standard initiatives, and *vice-versa*, we provide a summary table in TABLE 1. Considering this model, sections II-A to II-F provide the necessary background regarding each of these layers, starting from top (Application) to bottom (Network), by detailing the key metrics – *(a) to (o)* – that a simulator should allow for measuring/tracking throughout a simulation run. Section II-G then discusses the extent to which existing blockchain simulators cover those layers and metrics.

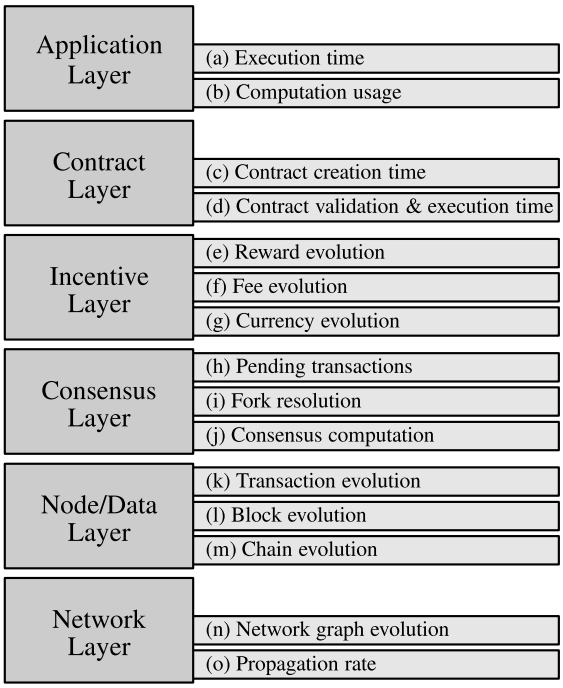


FIGURE 1. Blockchain abstraction layer model & associated metrics denoted by (a) to (o) in the rest of this paper.

TABLE 1. Extent to which the six-layer model introduced in FIGURE 1 covers the ITU and ISO blockchain standard initiatives, and *vice-versa*.

	Layer Focus					
Standards	Application	Contract	Incentive	Consensus	Node/Data	Network
ITU-T SG16 Q22 F.751.1	✔	✔		✔		✔
ITU-T SG16 Q22 F.751.2		✔		✔	✔	✔
ISO/DIS 22739	✔	✔	✔	✔	✔	✔
ISO/WD TS 23258		✔		✔	✔	✔
ISO/CD 23257				✔	✔	✔
ISO/CD TR 23576					✔	
ISO/AWI TS 23259		✔				

A. APPLICATION LAYER

This layer manages the user interface, APIs (Application Programming Interface), and computational resources (e.g., needed for blockchain element storage, wallet creation, etc.). In many blockchains, there is the possibility to configure a node to run as a full node (storing a local copy of all transactions and blocks) or as a lightweight node (in charge of creating transactions and sending them to full nodes for validation purposes). From a simulation (or emulation[1]) perspective, the following metrics should be measurable at this layer: (a)

1) *Execution time:* it refers to whether the simulator keeps track of the time needed to run the simulation;
2) *Computational resource usage:* it refers to whether the simulator keeps track of the resource usage evolution throughout the (simulation) run, which includes CPU of each node, swap space, storage capacity, etc.:

[1] Simulation/Simulator and Emulation/Emulator are interchangeably used in the rest of the paper.

B. CONTRACT LAYER

Blockchain systems consist of scripts, also referred to as ''smart contracts'', that run when predetermined conditions are met (e.g., used to automate the execution of an agreement so that all participants can be immediately certain of the outcome). At this layer, the following metrics should be measurable: (a)

3) *Contract creation time:* it refers to whether the simulator keeps track of the time needed to generate the different contracts (which can be of different sizes) over the simulation;
4) *Contract validation & execution time:* it refers to whether the simulator keeps track of the time needed to validate and execute the different contracts of the simulation scenario.

C. INCENTIVE LAYER

Participation incentive means the way how honest behavior is incentivized and dishonest discouraged. Incentives can be in the form of transaction fees and/or rewards [1]. This decision affects the implemented consensus algorithm (introduced in the next model layer) and, respectively, is affected by the selected algorithm. At this layer, the following metrics should be measurable: (a)

5) *Reward evolution:* it refers to whether the simulator keeps track of the amount of cryptocurrencies distributed from the consensus process (e.g., leader election or mining) over the simulation;
6) *Fee evolution:* it refers to whether the simulator keeps track of the reward fees that client nodes offer to miners to incentivize them to process their transaction(s).
7) *Currency evolution:* it refers to whether the simulator keeps track of the currency generation rate, which evolves differently according to the implemented blockchain (e.g., in Bitcoin or Ethereum, it evolves along with the hashing difficulty).

D. CONSENSUS LAYER

Consensus protocols are needed to validate the data to prevent and remove any duplicated entry and/or fraud [24], [25]. The type of blockchain to be implemented (public, private, consortium) influence profoundly the type of consensus protocol to be used, the most well-known being Proof-of-Work (PoW), Proof-of-Stake (PoA), or still Practical Byzantine fault tolerance (PBFT). At this layer, the following metrics should be measurable: (a)

8) *Pending transactions:* it refers to whether the simulator keeps track of the number of transactions, over time, that are waiting to be confirmed (such a waiting area is called ''Mempool'' in Bitcoin, or sometimes called transaction queues);
9) *Fork resolution:* it refers to whether the simulator keeps track of (i_1) the number of forks that appear within the chain; (i_2) the stale rate (i.e., block discarded) throughout the simulation;
10) *Consensus computation:* it refers to whether the simulator keeps track of the collective (or individual) computation effort required to validate transactions and blocks.

E. NODE/DATA LAYER

The node (or data) layer is responsible for structuring the data before appending it to blocks, whose structure usually includes information such as previous block hash, Merkle root, time, bits, etc. At this layer, the following metrics should be measurable: (a)

11) *Transaction evolution:* it refers to whether the simulator keeps track of the number of transactions generated per day (k_1) and whether the simulated transactions match the real-life data structure (k_2). Unlike k_1, k_2 is not a quantifiable metric but rather a boolean metric that states whether the simulator generates transaction following the real-world blockchain specification;
12) *Block evolution:* it refers to whether the simulator keeps track of (l_1) the number of blocks that are validated, mined and accepted as part of the longest chain; (l_2) the (average) time taken by each block to be validated; (l_3) the block sizes, which depend on the size of the transactions they include; and finally (l_4) the number of transactions included, on average, within a block.
13) *Chain evolution:* it refers to whether the simulator keeps track of the length of the chain over time (i.e., the number of blocks that form the longest chain), which is a good indicator of the load a new node would have to compute if it joins the network at a given point in time.

F. NETWORK LAYER

Blockchain is a pure P2P network, which is actually an overlay network [41] for distributed object storing, searching, and sharing (e.g., Ethereum relies on the Kademlia P2P protocol [41], [42]). At this layer, the following metrics should be measurable: (a)

14) *Network graph evolution* it refers to whether the simulation accurately follows the P2P protocol overlay network specifications (e.g., support for adding and discovering a node at any time) and keeps track of the network (node) evolution using network graph metrics such as Clustering Coefficient, Mean Geodesic Distance, and Diameter [43], [44];
15) *Throughput:* it refers to whether the simulator keeps track of the number of valid transactions per second (Tx/s) that have been incorporated as part of a valid block within the longest chain. The transactions that are considered here are the ones that reach the majority of nodes within the network depending on the underlying consensus (i.e., $\geq$ 50% for PoW, $\geq$ 66% for PBFT, etc.);

TABLE 2. Literature comparison—● symbol indicates that the simulator makes available the performance metric without any code modification and/or post-processing, while ◐ symbol indicates that such a modification and/or post-processing is needed to obtained the desired performance metric.

	Application		Contract		Incentive			Consensus			Node/Data			Network	
Framework	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)	(k)	(l)	(m)	(n)	(o)
DLSF [26]	◐	◐			●			●	◐	◐	●$_{k1,k2}$	●$_{l1,l2,l3,l4}$	◐		◐
Shadow [18]	●	●			◐				●		●$_{k1}$	◐	◐		●
Wang et al. [27]	◐	●			●			●			●$_{k1}$	●$_{l2,l4}$			
BlockSim-Net [28]	◐	●					◐		●	◐	◐$_{k1,k2}$	●$_{l1,l2,l3}$	◐		◐
eVibes Plasma [29]	◐		◐	●					●		◐	◐	◐		◐
Vibes [19]	●							◐			●$_{k1}$	●$_{l1,l2,l4}$	◐		●
eVibes [30]	◐			●					●		●$_{k1}$	●$_{l1,l2,l4}$			●
BlockSim [16], [31]	●	◐				◐			●		●$_{k1}$	●$_{l1,l2}$	◐		●
Goswami [32]	◐								◐		●$_{k1}$	●$_{l2,l4}$			
Bitcoin-Simulator [33]	◐						◐		◐		●$_{k1}$	●$_{l1,l2,l3,l4}$	◐		●
SIMBA [34]	◐								◐	◐	●$_{k1}$	●$_{l2,l3,l4}$	◐		◐
Chin et al. [35]	◐				◐		●		●		●$_{k1}$	●$_{l1,l2,l4}$	◐		◐
SimBlock [36], [37]	◐				◐				●	◐	●$_{k1}$	●$_{l1,l2,l4}$	◐		●
CIDDS [38]		◐								●				●	●
HIVE [39]				◐	◐	◐		◐	●	●	●$_{k1,k2}$	●$_{l1,l2,l3,l4}$			◐
Androulaki et al. [40]		◐							●	◐		●$_{l1,l4}$	●		
PeerSim [17]														●	●

G. RELATED WORK AND DISCUSSION

As of today, there are several blockchain simulators found in the literature, a summary of which is in TABLE 2. This table highlights what layers and associated metrics those simulators cover. Note that a simulator may, in some cases, cover a given layer but without necessarily providing as output a performance metric. This means that either a modification of the source code or post-processing treatments are required in order to compute the desired metric. For example, in BlockSim, the authors claim in their paper [31] that the simulator keeps track of the fee evolution, however, after analyzing it, we realized that it is not possible to retrieve it without modifying part of the code. Another example is HIVE [39] that emulates the behavior of smart contracts within the Ethereum environment, however, we found that it does not allow for analyzing the (h) metric (i.e., pending transactions). To make a distinction between simulators that make available a performance metric without requiring any code modification or post-processing, and the ones that require a modification/post-processing to obtain the metric, the following two symbols are respectively used in TABLE 2: ● (does not require any code modification and/or post-processing) and ◐ (does require a code modification and/or post-processing.

As a first simulator, let us mention Bitcoin-Simulator [33], which has been designed for educational purposes to help students to understand how the block generation rate (l_1) and block size (l_2) evolve over time. Although it is a well-designed pedagogical tool, it is quickly limited to carry out in-depth simulation analyzes. More research-oriented simulators have been proposed, such as Ethereum Hive [39] that have been proposed for emulating and evaluating Ethereum's smart contracts from a validation & execution time perspective (d_1-d_2). However, as revealed in [30], simulating a large number of nodes becomes difficult with limited computational resources. To overcome this limitation, a series of simulators including VIBES [19], eVIBES [30], eVIBES Plasma [29] ones, as well as CIDDS [38], were proposed, but unfortunately fall short of fulfilling the promise, as reported by Lathif *et al.* [38]. One of the main reasons for this is that those simulators do not adequately model the transactions at the Node/Data layer (i.e., k_2), considering them as empty in the majority of the simulators. Such a consideration poses several issues when evaluating the blockchain system as a whole, knowing that transaction and block sizes may have non-negligible impacts on the overall system performance [45]. To overcome this issue, a new range of simulators, including BlockSim [16], [31], SIMBA [34], DLSF and others (cf., TABLE 2), have considered the transaction/block structure, algorithms for wallet creation, message signing, and so forth, thus leading to more realistic performance evaluation results. Nonetheless, the network modeling is often too simplistic, not reflecting the real behavior of the P2P overlay network (i.e., possible communication delays, network congestion, packet losses, computation resource limitations, etc.), which, in our opinion, can have significant impact on the overall system performance evaluation process. A few simulators have been designed to consider the throughput and the network graph evolution, such as CIDDS, PeerSim. However, these two simulators neglect many of the upper layers and metrics, as highlighted in TABLE 2.

The drawbacks discussed in this section prove that designing a modular simulator that allows for testing/evaluating different blockchains, with different consensus protocols, different contract and transaction/block specifications, different incentive schemes, along with network infrastructures, turns to be a challenging task. FIGURE 2 provides an overview of the extent to which the reported blockchain simulators cover the different layers of the six-layer model previously introduced. It can be first observed that they primarily often

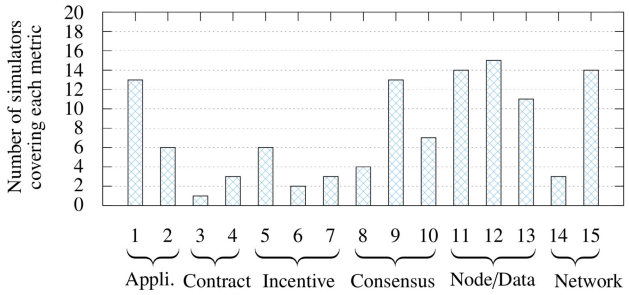


FIGURE 2. Overview of the extent to which state-of-the-art simulators – *the* 17 *reported in TABLE 2* – cover the six-layer model introduced in FIGURE 1.

neglect the Contract and Incentive layers. Second, key aspects of a blockchain systems at the Network and Data/Node layers, namely the consideration of the real P2P overlay network protocol behavior (n) and transaction/block structure (k_1), result in non-optimal performance evaluation results (non-optimal compared to the reality). A new hybrid emulator/simulator tool called "BlockPerf" is proposed in this paper to overcome this limitation. BlockPerf is hybrid in the sense that it emulates the network layer to correctly address the network layer while simulating the upper layers based on statistical data modeling approaches, as presented in the next section.

III. BLOCKPERF SIMULATOR DESIGN

This section describes how BlockPerf extends BlockSim, whose main extensions are summarized in FIGURE 3. These layer extensions and/or adaptations are respectively discussed through sections III-A to III-E. Note that, at this stage, BlockPerf does not address/model the "Smart Contract" layer and is part of future research work.

A. APPLICATION LAYER

At this layer, BlockPerf consists of a configuration module responsible for instantiating the run's parameters similar to the one used in BlockSim. BlockSim takes as input a list of parameters, including the size of blocks, statistical models for transaction validation, block validation, and node labels. Similarly, BlockPerf takes as input a `JSON` file that extends the parameters from BlockSim to include fork occurrences, the block size parameter, IP address of each node, including the 'main" one (corresponding to the `BitcoinNode` class, as summarized in TABLE 3) that tracks all the results using the `CliStats`, `CpuTimeSnapshoot` and `MemorySnapshot` classes. Optionally, the input file also considers the location to store all the logs and data for each node. Note that the `BitcoinNode` is responsible for tracking the execution time (a) of the overall run and collecting the metrics regarding the overall computational resource usage (b) of the nodes.

B. INCENTIVE LAYER

This layer is responsible for instantiating the models for rewarding participants, which vary from one blockchain to another. While BlockSim does not simulate the incentive layer,[2] BlockPerf models two types of rewards: one for the generation of the valid blocks (known as *block reward*) and the second for the inclusion of a particular transaction to the block (known as *transaction fee*). These reward operations are implemented via the `TxChain`, `BTCNode` and `TickEvent` classes (*cf.*, TABLE 3. As of today, the *block reward* is represented as a fixed amount of cryptocurrency that can be configured for the run, while the *transaction fee* is dependent on the size of the transaction, as formalized as in Eq 1, where Tx_i refers to the i^{th} transaction and f (size(Tx_i)) can be configured using different distributions laws (e.g., defined as a fixed fee, or following a Weibull, Log-normal, or Gamma distribution). All node wallets are continuously updated throughout the simulation run.

$$\text{Tx}_i^{\text{fee}} = f\ (\text{size}(\text{Tx}_i)) \tag{1}$$

The amount of new cryptocurrencies generated over time depends on the consensus layer and the overall computational resources of the nodes in the network. BlockPerf continuously monitors such resources and allocates the newly generated cryptocurrencies to the right (mining) nodes.

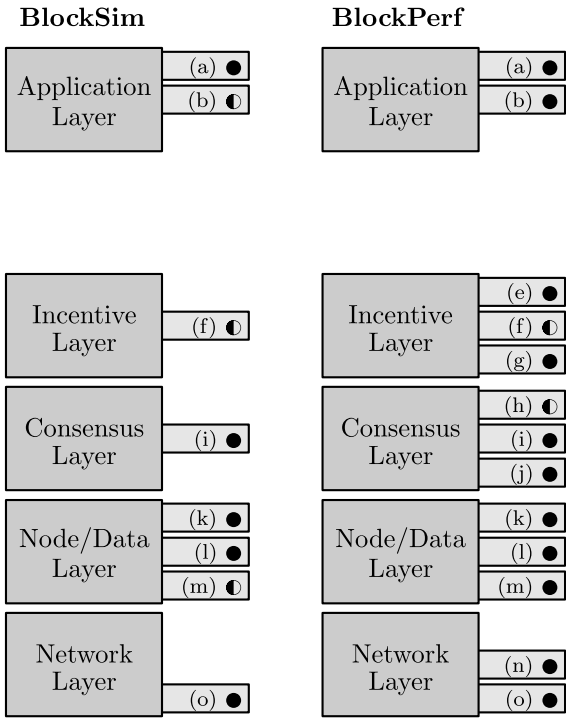


FIGURE 3. Illustration of how BlockPerf extends BlockSim - ● symbol indicates that the simulator makes available the performance metric without any code modification and/or post-processing, while ◐ symbol indicates that such a modification and/or post-processing is needed to obtained the desired performance metric.

C. CONSENSUS LAYER

In BlockPerf, the consensus protocol is implemented as a model that represents the consensus process and

[2]Even though the authors in [31] state that the incentive layer is partly covered, nothing from the source code provides clear evidence of this.

TABLE 3. Technical description of the major classes within BlockPerf, along with the layers they cover.

Class Name	Description	Application	Incentive	Consensus	Node/Data	Network
CliStats	Node state checking and logging (incl., parameters like operation time, computation)	✓		✓	✓	
CpuTimeSnapshot	Computation process and database tracking	✓			✓	✓
MemorySnapshot	Output file generation for the memory used by a given node	✓				
Proxy	Authentication management for network components and wallet information	✓				
TxChain	Chain operation handling (address generation, private-public keys, unspent Tx, etc.)		✓	✓	✓	
PublicBitcoinNode	Network stat monitoring (network graph, latency...)				✓	✓
BTCNode	Monitoring of transactions/block evolution, handshaking, etc.		✓		✓	✓
Node	Node operation handling (incl., creation of transaction, Queues)				✓	✓
TickEvent	Transactions/block timestamping		✓		✓	
TransactionQueue	Queue transaction creation (i.e., transactions to be generated by a given node)				✓	
BlockHeader	Header tracking for the chain that a node follows			✓		✓
Consensus	Consensus protocol model instantiation		✓	✓		
BitcoinNode	Connection handling and in charge of pushing model parameters to the network					✓
ZoneConfig	Node parameter consideration for the emulation layer (incl., IP addresses)					✓
Event	Event tick creation based on the models					
Runner	*main* class					

its operations. The consensus algorithm is coded within the `Consensus` class (see TABLE 3), which is in charge of selecting the miner/validator that builds the next block. The method of selecting the node varies from one blockchain to another, although in BlockPerf, as a first step, the Proof-Of-Work (PoW) algorithm has been implemented. As opposed to BlockSim that simulates the behavior of the consensus algorithm (i.e., the block validation and a random selection of miner), BlockPerf uses an extended approach where each mining node, similar to PoW algorithm, selects a random number, the input for all the non-confirmed transactions from its queue within the limit of block size (further discussed in Section III-D), as well as the reference of the previous known block, which allows for being closer to a real PoW process. This information is then combined and hashed recursively until a result is obtained. The mining node selects the transactions from the queue (sometimes called mempool), check whether they meet the balance requirements, verify whether the sender signatures match and that the sender's wallet has a sufficient amount of cryptocurrencies. In BlockPerf, the `BitcoinNode` class keeps track of the fork occurrences within the network.

D. NODE/DATA LAYER

Any node wishing to generate a transaction has to follow a set of operations, namely: (i) select a wallet address (from *wallet_list*) as the recipient; (ii) select a random value for the transaction; (iii) generate the transaction by signing it using its private wallet key (those operations being handled via the `TickEvent`, `Transaction Queue` and `TxChain` classes, as reported in TABLE 3). The set of transactions to be created per unit of time t, which is denoted by $\mathcal{X}(t) = \{\mathrm{Tx}_1, \ldots, \mathrm{Tx}_k\} \,|\, k \in \mathbb{N}$, can be configured using different distributions laws (e.g., a fixed amount of transactions per unit of time, or following a Weibull-like distribution). Every new transaction created by a node calls the propagation function from the network layer to transmit this new transaction to its neighbors, who then append it to their pool. Upon reception of a transaction (Tx_i), several operations are undertaken by the recipient node, as summarized in the flow chart FIGURE 4 (operations that have been added compared with BlockSim being highlighted in green). Unlike BlockSim, BlockPerf models transactions as they exist in the real system (i.e., each transaction can be traced back and follow a similar process within the system).

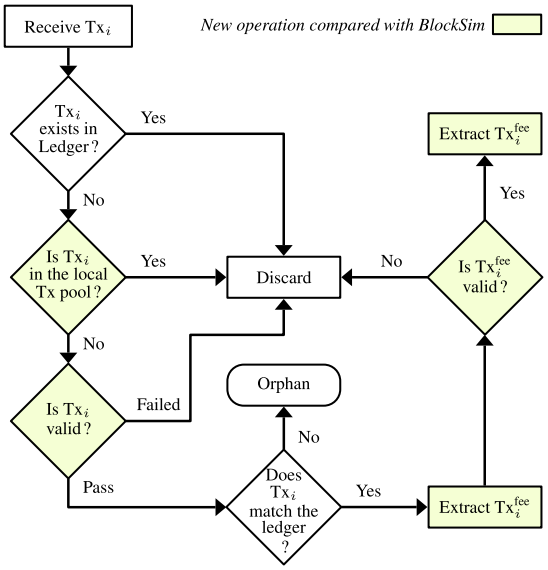


FIGURE 4. Flowchart of transaction operation sequence upon reception of Tx_i in BlockPerf.

As a second stage, the simulator has to handle the inclusion and/or removal of transactions – *from its transaction pool* – into blocks. Although the consensus layer governs operations towards deciding which block to be accepted by everyone within the network, the Data Layer is concerned with the reception of the block, being in charge of removing all the transactions which are enclosed in the last received

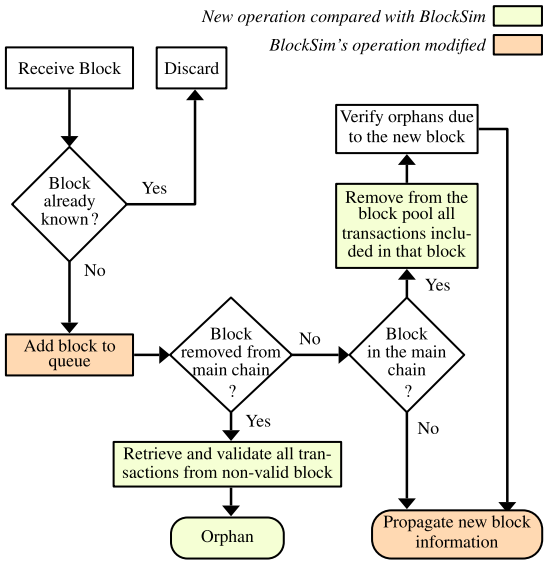


FIGURE 5. Flowchart of transaction inclusion/removal from a block in BlockPerf.

block from its transaction pool, and finally attaching the new block to its last known state of the chain (handled by the `Node` class). These operations follow the sequence of operations given in the flow chart of FIGURE 5 (new operations compared with BlockSim being highlighted in green, while BlockSim's operations that have been modified are highlighted in orange).

The linked blocks form a chain, but nodes may have a different state of the chain (or ledger) due to network communication delays (this is termed forks). As opposed to many existing simulators, including BlockSim, each node in BlockPerf has its local ledger implemented, which means that if a node discovers that the state of the chain is different from its own, it sends a request to neighboring nodes for getting the full state of the chain. This allows being closer to reality, which should result in more realistic simulation performance. This improvement, compared with BlockSim, is highlighted in the flow chart given in FIGURE 6 (*cf.* green boxes/steps).

E. NETWORK LAYER

The network layer of BlockPerf differs significantly from BlockSim, and many other simulators reported in TABLE 3, since, in BlockPerf, the blockchain network is emulated over distributed (physical) nodes. The benefit of doing so is that the simulation should lead to results closer to reality, as BlockPerf implements a P2P protocol similar to the ones used by real blockchain systems using advertisement messages (e.g., `GETADDR` and `ADDR`) to discover neighboring nodes. BlockPerf also integrates:

- *a wallet management module:* in charge of generating wallet addresses to uniquely identify all nodes. This module also integrates the protocol to broadcast wallet-related information in the network (via `WALLADD` messages) to make all nodes aware of existing addresses;

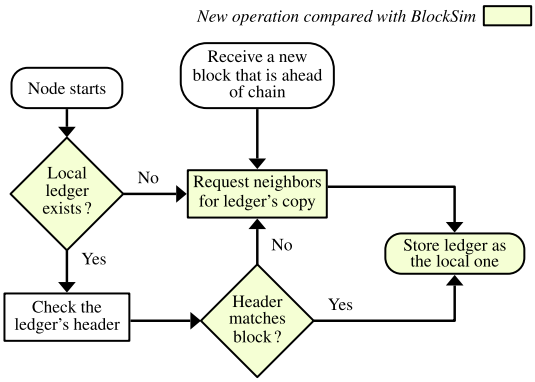


FIGURE 6. Flowchart of local ledger in BlockPerf.

- *a reputation score module:* in charge of assigning negative scores from the connections where malformed messages were received (e.g., if the transaction correctness fails, nodes keep receiving outdated block information). This module is fulfilled by the `PublicBitcoinNode` class, which is responsible for keeping a log of the neighboring nodes and associated scores. Such logs allow BlockPerf to keep track of the network graph evolution (n) over the simulation run.

F. BLOCKPERF IMPLEMENTATION

TABLE 3 provides an overview of the important programming classes underpinning BlockPerf, whose implementation is combination of C/C++ programming languages for the network layer, and Python for the upper layers. The source code of the BlockPerf simulator is available on GitHub.[3]

IV. VALIDATION AND EVALUATION

This section aims to compare the extent to which BlockPerf provides realistic results, but also to which it outperforms BlockSim. FIGURE 7 provides an overview of the experimental process that has been conducted in this respect, which consists of five main stages denoted by ① to ⑤ in FIGURE 7. First (①), a real-life blockchain network (`bitcoind`) consisting of 63 nodes spread over the world has been implemented, which corresponds to the benchmarking Testbed in this study whose metrics (a) to (o) have been collected/measured whenever possible (see ②). Based on those metrics, several parameters are specified as input parameters of BlockPerf and BlockSim (see ③ and ④), as will be presented in section IV-A. BlockPerf and BlockSim are then compared against the results obtained for the (benchmarking) Testbed in section IV-C (see ⑤). A discussion about our experiments is finally given in section IV-C.

A. BENCHMARKING TESTBED

The (benchmarking) Testbed is made of a modified version of the reference bitcoin implementation, namely

[3]BlockPerf repo: https://github.com/Deadlyelder/BlockPerf

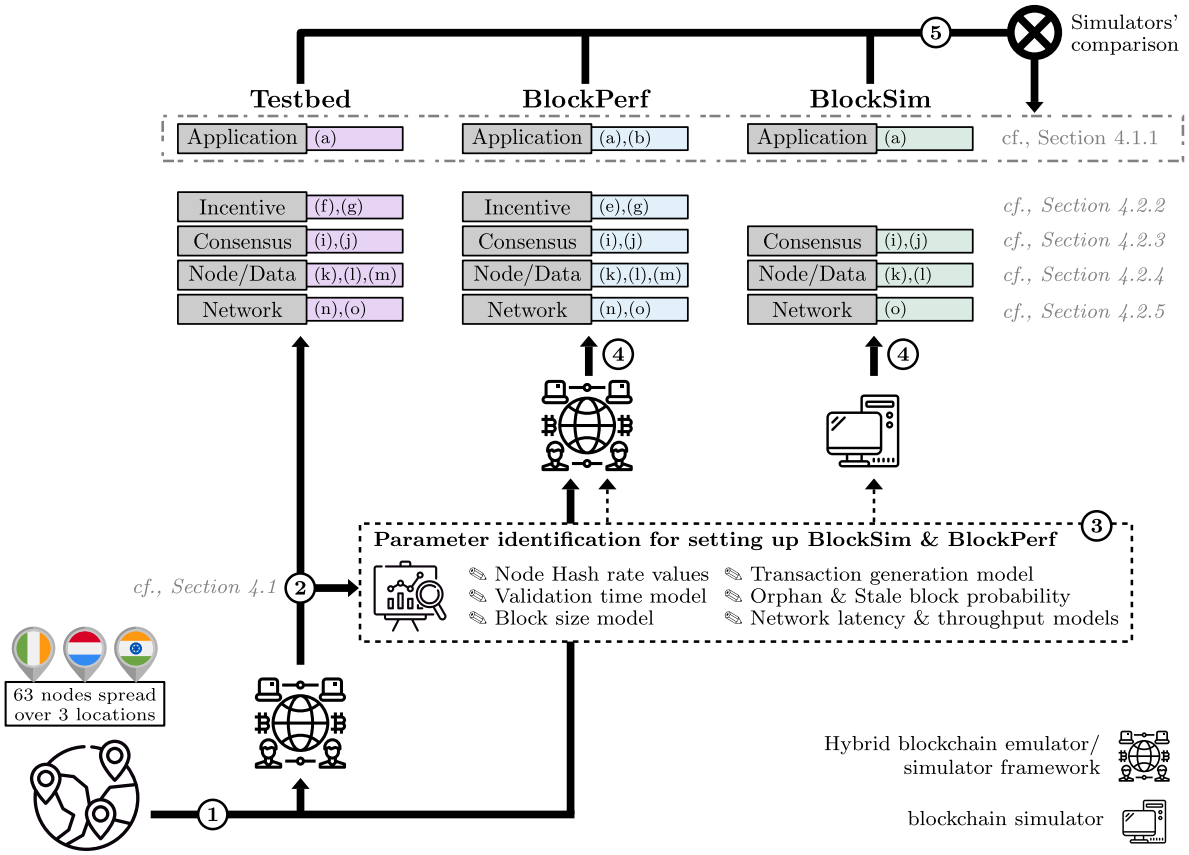


FIGURE 7. Experimental process set up and conducted in this paper to compare BlockSim and BlockPerf simulators.

TABLE 4. Overview of the results.

	Metrics	Evolution (cf.,)	Testbed	BlockPerf	BlockSim
Application	(a) Execution time (in hours)	-	264h	8h	0.5h
	(b) Computation usage (Average CPU)	-	80%	90%	-
	(Average memory)	-	40%	86%	-
	(Average storage)	-	2.6GB	2.9 GB	-
Incentive	(e) Average reward per day	Figure 8	-	0.043991	-
	(f) Fee evolution	-	*by default*	0.00001	-
	(g) Currency generat. rate per time unit	Figure 9	720 BTC	384 BTC	-
Consensus	(i_1) Total number of forks	-	2	5	2
	(i_2) Total number of blocks discarded	Figure 10	4	13	5
	(j) Min-Max consensus computation effort	-	10-45 MH/s	9-42 MH/s	1-2 MH/s
Node/Data	(k_1) Average Tx/day	Figure 11	7341	4504	14184
	(l_1) Total number of validated blocks	Figure 12(a)	1979	1118	4132
	(l_2) Average time to validate blocks	Figure 12(b)	10.7min	10.9min	10.4min
	(l_3) Average block size	Figure 12(c)	1.6 MB	1.1 MB	2 MB
	(l_4) Average number of Tx/block	Figure 12(d)	41	44	38
	(m) Chain's length (total number of blocks)	Figure 13	1980	1119	4133
Network	(n) Network graph evolution (Cluster Coefficient)	-	0.473217	0.469478	1
	(Mean Geodesic Distance)	-	2.08631	2.11437	1
	(Diameter)	-	4	4	1
	(o) Average throughput (Tx/s)	Figure 17	1.5	1.6	2

`bitcoind`.[4] The evaluation is performed during 11 days, whose blockchain network consists of 23 full nodes and 40 light nodes spread over three locations, namely Ireland, Luxembourg and India. During the course of the experiment, transactions are randomly created. Based on this experimental configuration setting, metrics (a) to (o) are measured, whenever possible, and then reused for two purposes:

1) to serve as benchmarking metrics to evaluate the extent to which BlockPerf and BlockSim deviate from the reality (see ⑤ in FIGURE 7);

[4] A slight difference can be noted, as our Testbed is a private network where all nodes are controlled and maintained throughout the experiment.

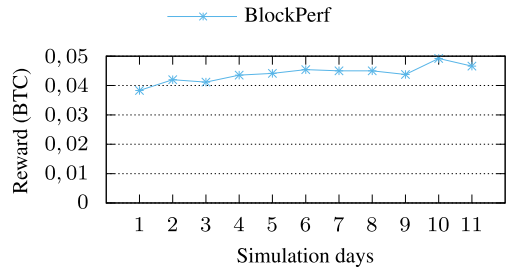


FIGURE 8. Reward evolution over days (e).

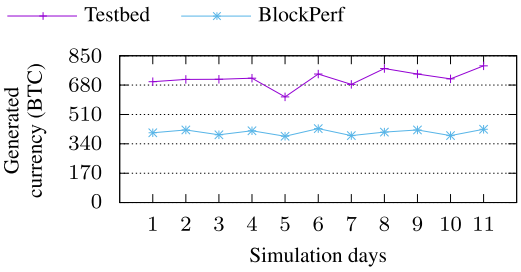


FIGURE 9. Currency generated over days (g).

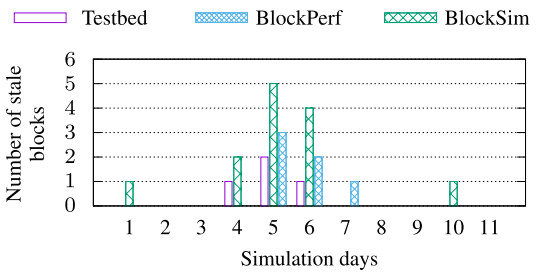


FIGURE 10. Stale blocks per days (i_2).

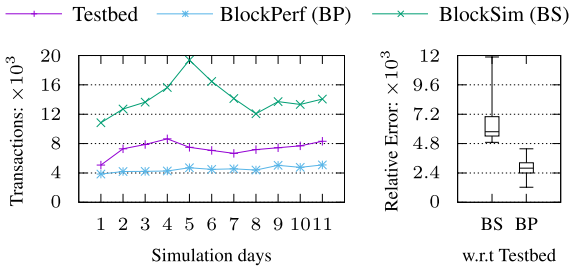


FIGURE 11. Transaction evolution (k_1) & relative absolute error with respect to (w.r.t) the Testbed.

2) to identify, from the Testbed run, several parameters that need to be configured as inputs of BlockPerf and BlockSim such as node hash rate values, block size distribution model, etc. (see ③-④). To this end, a similar process as the one defined in [16] has been applied to extrapolate the probability distribution model for each parameter.

TABLE 4 summarizes all the metric values obtained from the Testbed, and, for some of them refer to specific figures/graphs. For example, looking at metric k1, TABLE 4 reports the average number of transactions/day, while FIGURE 11 reports the exact number transactions over the 11 days of the experiment).

B. EXPERIMENT WITH BLOCKPERF AND BLOCKSIM & COMPARISON

In this section, the performance of BlockPerf and BlockSim is compared against the benchmarking Testbed results. BlockSim experiments have been conducted on a single machine with a 2.40 GHz Intel Xeon E5 CPU and 4GB of RAM. In contrast, BlockPerf has been deployed over distributed nodes – *located in Ireland, Luxembourg and India* – with the same computational resource (i.e., 2.40 GHz Intel Xeon E5 CPU and 4GB of RAM). Sections IV-B1 to IV-B5 discuss the comparison analyses regarding the different layers, as emphasized in FIGURE 7 (see ⑤). Note that for this comparative analysis, only the metrics that are covered and for which the results are available (i.e., metrics with ● in FIGURE 3 and listed in FIGURE 7) are considered.

1) APPLICATION LAYER

From an *Execution time (a)* perspective, BlockSim and BlockPerf allow for simulating the experiment at a faster pace compared to the testbed, the former being 528 times faster while the latter is 33 times faster. This substantial difference is explained by the fact that BlockSim does not rely on a real-life network layer. In terms of *Computational resource usage (b)*, the average CPU, memory and storage metrics have been reported in TABLE 4, although it cannot be concluded that one simulator is better than another for those metrics.

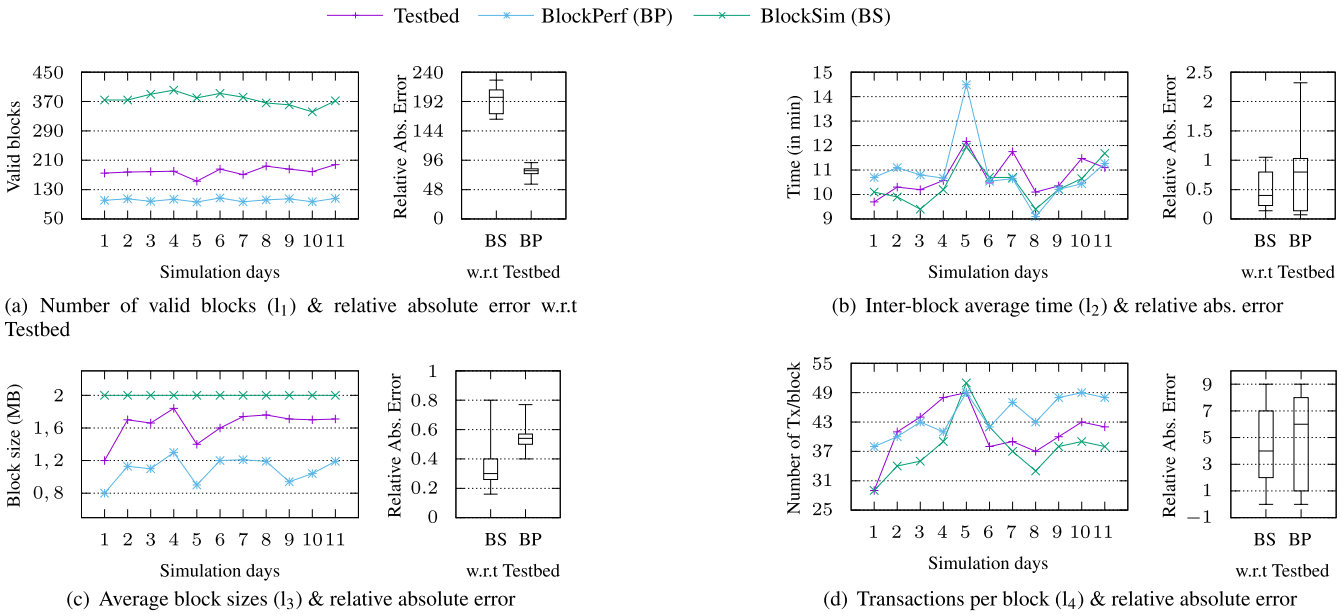
2) INCENTIVE LAYER

Unlike BlockSim and the Testbed, BlockPerf stores log about the *Reward evolution (e)*, as they are generated within the run. FIGURE 8 provides insight into the evolution of the reward over the 11 days of simulation. On average, the reward is 0.043991 BTC per day, equally distributed to the miners with a standard deviation of 0.0002 throughout the run. Although no comparison for this metric could be made with the Testbed and BlockSim, we nonetheless report those results as they could serve as benchmarks for other researchers.

The *Currency generation rate (g)* can be compared with the Testbed, but not with BlockSim (metric not available). FIGURE 9 shows the evolution of the currency generated over the course of the simulation, 384 BTC being generated on average by BlockPerf, against 720 BTC generated by the Testbed. Despite this difference, what is interesting to note is that that the evolution in BlockPerf and Testbed follows a similar trend.

3) CONSENSUS LAYER

During the simulation, a number of *Fork resolutions* (i_1) are handled by the Testbed (2 in total, one at day 4 and one at day 5), by BlockSim (2 in total, one at day 4 and one at day 5), and by BlockPerf (4 in total, three at day 5 and two at day 6). These forks led to the appearance of stale blocks (i_2), whose occurrences over the 11 days of the experiment have been reported in FIGURE 10. Two observations can be made: (i) stale blocks mostly appear in the same timeframe (between



(a) Number of valid blocks (l_1) & relative absolute error w.r.t Testbed

(b) Inter-block average time (l_2) & relative abs. error

(c) Average block sizes (l_3) & relative absolute error

(d) Transactions per block (l_4) & relative absolute error

FIGURE 12. **Block evolution (l_1-l_2).**

day 4 and 6); (ii) BlockPerf provides more realistic results than BlockSim, as the number of stale blocks in BlockPerf (4 in total) is closer to the Testbed (6) than BlockSim (13).

Regarding now the *Consensus computation effort* (j), BlockPerf also provides more realistic results, ranging between 9-42 MH/s, while the Testbed ranges between 10-45 MH/s and BlockSim between 1-2 MH/s. The reason for such a difference is that, in BlockSim, even when specifying as input parameters the min and max hash rate values (i.e., 10-45 MH/s identified through the benchmarking phase, *cf.* section IV-A), it seems that BlockSim does not consider that parameters in the simulation run, always applying 1-2 MH/s as hash rate range.

4) NODE/DATA LAYER

As the first metric of that layer, let us look at the *Transaction evolution* (k_1) in FIGURE 11, which shows the number of transactions generated daily by the Testbed and the two simulators. It can be observed that BlockPerf closely follows the Testbed's transaction evolution, thus providing more realistic results than BlockSim. Indeed, BlockPerf and BlockSim respectively generate 4504 and 14184 Tx/day on average, while the Testbed generates 7341 Tx/day. For a more accurate view on the extent that BlockPerf leads towards realistic results, we show the relative absolute errors of BlockPerf and BlockSim with respect to the results obtained from the testbed is represented in the form of a boxplot in FIGURE 11. For example, the max value of the BS (BlockSim)-related boxplot (i.e., $\approx$ 12000) indicates that, over the 11 days of experiments, the maximal difference in terms of number of transactions generated by the Testbed and BlockSim is 12000, which – *if we look at the transaction evolution in FIGURE 11* – corresponds to day 5. Looking at the error

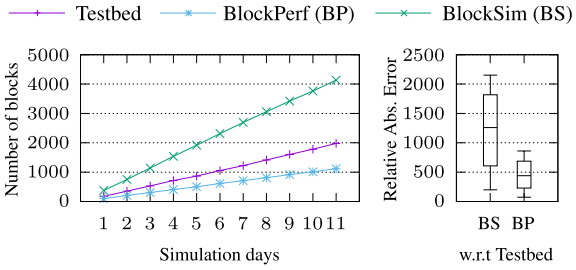
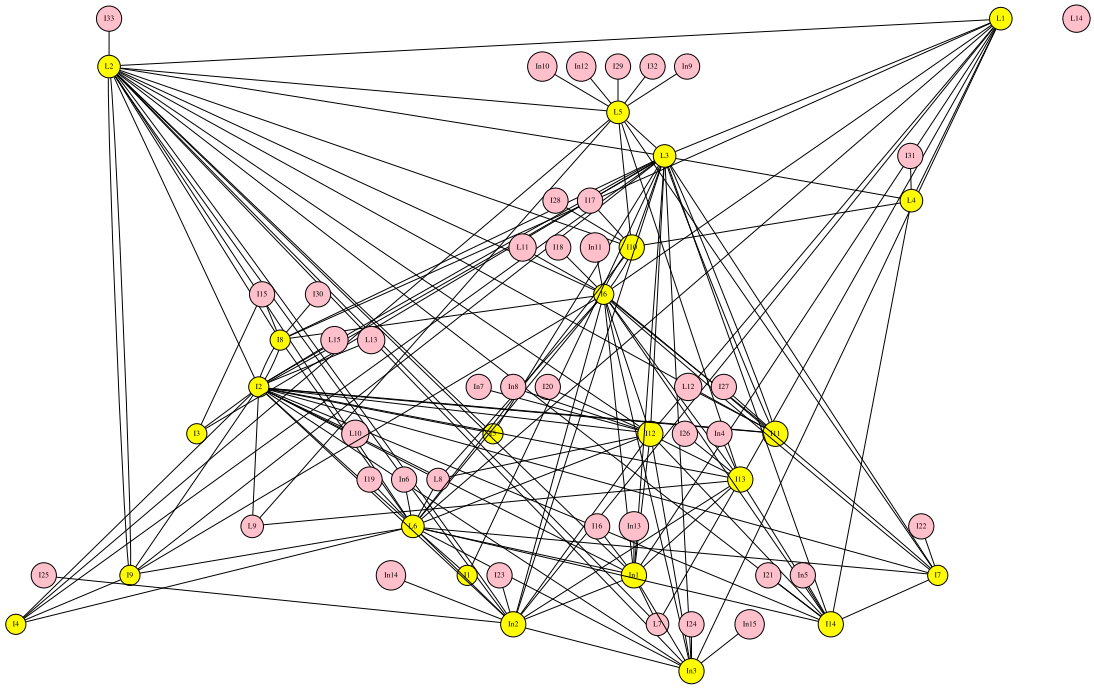


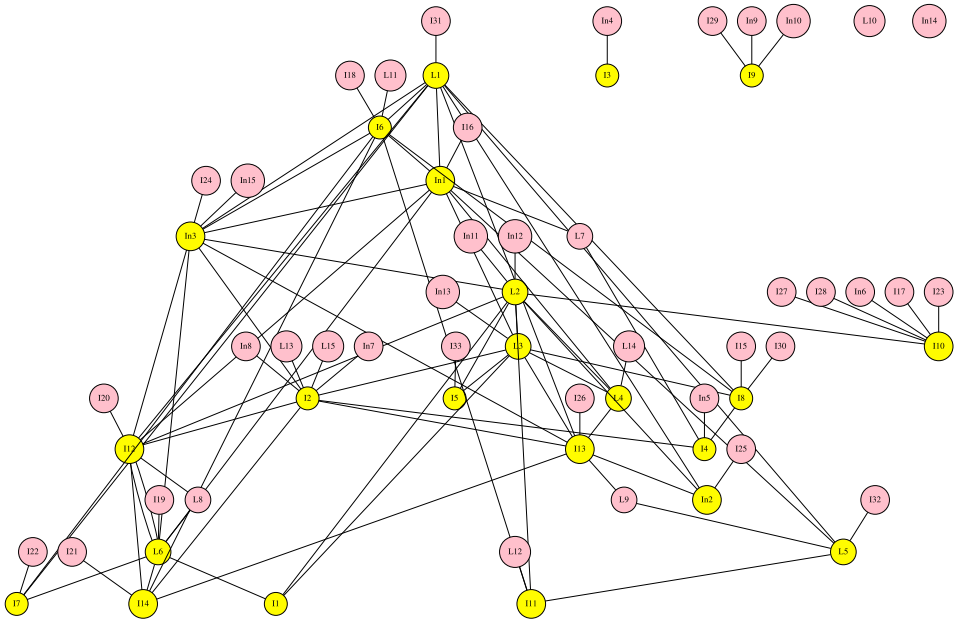
FIGURE 13. **Chain's length (m).**

median values, it can be noted that the difference/error is quite significant in BlockSim, being approximately twofold higher compared to BlockPerf. In addition of k_1, let us also note that BlockPerf follows the *real-life transaction specification* (k_2), while BlockSim adopts a more simplistic model, which may explain part of the difference in results of the transaction evolution above-discussed (k_1).

Let us now discuss about the *Block evolution* (l), which consists of four sub-metrics (l_1 to l_4). FIGURE 12(a) shows the evolution of the number of valid blocks (l_1) over the 11 days of experiment. It can be observed that BlockPerf produces fewer blocks than the testbed, which is likely due to the unpredictable nature of the network connections and the occurrence of forks. On the other hand, BlockSim produces a number of blocks ($\approx$400/day) that is twice higher than the Testbed ($\approx$200/day). Overall, as evidenced through the boxplot in FIGURE 12(a), BlockPerf provides more realistic results than BlockSim, whether in terms of min, quartiles and max values. Looking now at the second sub-metric (l_2: block validation times), both BlockPerf and BlockSim provide similar results to the testbed, as shown in FIGURE 12(b). Indeed, the three curves followed the same trend and the relative absolute error was $\leq$ 1 min (i.e., one-tenth of the total time



(a) P2P network topology underlying the Testbed at day 1 of the experiment

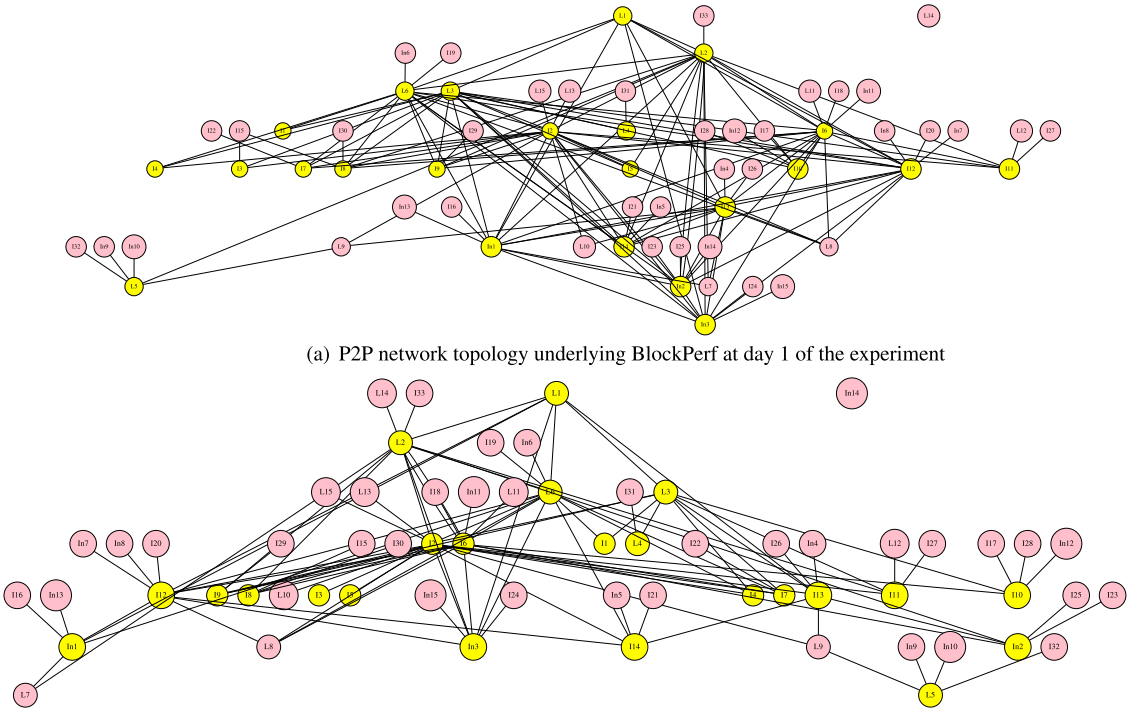


(b) P2P network topology underlying the Testbed at day 2 of the experiment

FIGURE 14. Testbed's network graph evolution over two days of experiment.

required by the Testbed). FIGURE 12(c) then provides insight into the evolution throughout the run of the block validation times (l_3), along with the difference – *relative absolute error to be precise* – between BlockPerf/BlockSim and the Testbed. It is interesting to note that, when looking at the boxplot, BlockSim provides closer results with the Testbed than BlockPerf; however, when looking at the day-by-day block validation times, BlockPerf evolves in a similar way to the Testbed, while BlockSim does not. Finally, regarding l_4 (i.e., the average number of transactions mined within blocks per day), the two simulators provide similar results to the Testbed, following a similar trend over the course of the run and having similar differences/errors compared to the Testbed (the min and max errors being the same for BlockPerf and BlockSim).

The last metric of the Node/Data layer refers to the length of the chain (m), whose evolution over the 11 days of the experiment for the Testbed and the two simulators are plotted in FIGURE 13. It can be observed that BlockPerf provides more realistic results than BlockSim, which becomes



(a) P2P network topology underlying BlockPerf at day 1 of the experiment

(b) P2P network topology underlying BlockPerf at day 2 of the experiment

FIGURE 15. **BlockPerf's network graph evolution over two days of experiment.**

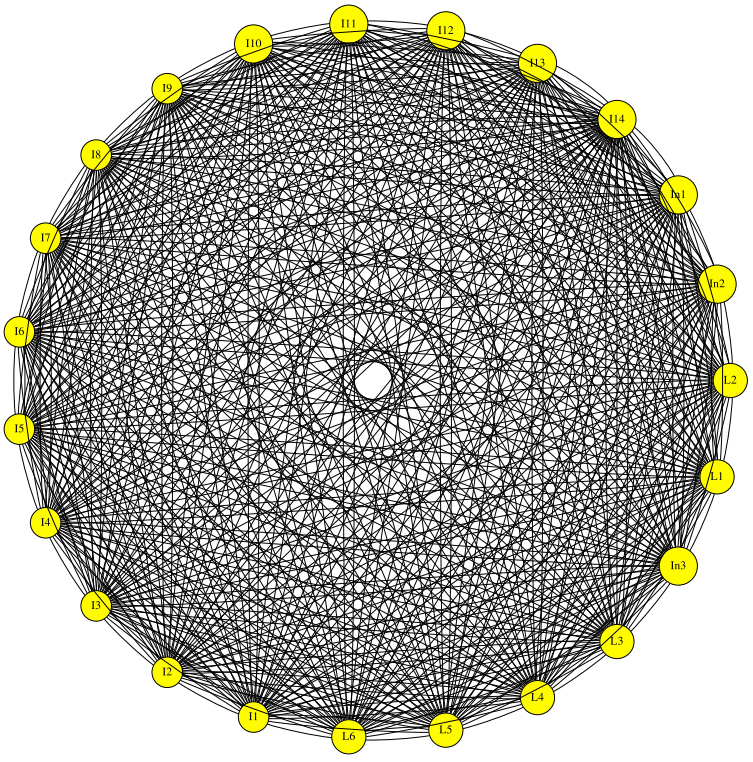


FIGURE 16. **BlockSim's network graph remaining unchanged throughout the simulation run.**

increasingly significant over time. Indeed, the longer the simulation run, the higher the difference between BlockSim and the Testbed. This is also confirmed by the boxplot given in FIGURE 13, the minimal error values of the two simulators being approximately the same, while the quartiles, median and max values become increasingly higher for BlockSim.

5) NETWORK LAYER

Two distinct metrics are analyzed at this layer, namely how the network graph evolves over time (n), and what throughput performance is possible with the implemented blockchain and scenario (o).

Regarding the first metric (n), FIGUREs 14, 15 and 16 provide the network graph evolution over two consecutive days (days 1 and 2) respectively regarding the Testbed, BlockPerf and BlockSim. Pink nodes correspond to light nodes and yellow ones to full ones, each node being denoted by the country's initial (L: Luxembourg, I: Ireland; In: India) and the node's numbering. A twofold observation can be drawn from those graphs. First, unlike BlockPerf and the Testbed, BlockSim assumes that all nodes are interconnected in a fully connected mesh throughout the experimental run, adding that it does not distinguish lightweight and full blockchain nodes. A second observation is that the P2P logical network infrastructure evolves quite substantially from one day to another, whether regarding the Testbed or BlockPerf, which are due, among other things, to eventual connection losses, ongoing traffic,.... The Diameter,[5] mean geodesic distance,[6] and clustering coefficient[7] reported in TABLE 4 show that the graphs evolve in a similar manner for BlockPerf and BlockSim, which is not the case for BlockSim.

Throughput is a key metric that provides a good performance indicator of a given blockchain system. It is nonetheless tricky to analytically formalize it, as it depends on multiple processes across the different layers. Indeed Application (for generation), Node/Data (for accessing rewards), Consensus (mining), Incentive (for rewards), and Network layers (for final throughput), all combined, influence the throughput of a given blockchain system. FIGURE 17 shows the evolution of the average throughput for the considered scenario, along with the boxplot showing the relative absolute errors of BlockSim and BlockPerf when compared to the Testbed. The results show that throughput ranges between 1 and 2.5 Tx/s for both the Testbed and the two simulators (BlockPerf having slightly closer results to the Testbed than BlockSim).

C. DISCUSSION

The previous section has shown that BlockPerf outperforms BlockSim for most of the metrics (a)-(o), or provide more realistic results to be more precise. We firmly believe that this is mainly due to the assumption made by BlockSim (i.e., non-consideration of the network-level change within the system). Furthermore, our approach presents several advantages. First, it covers all the layers (except the contract one) and

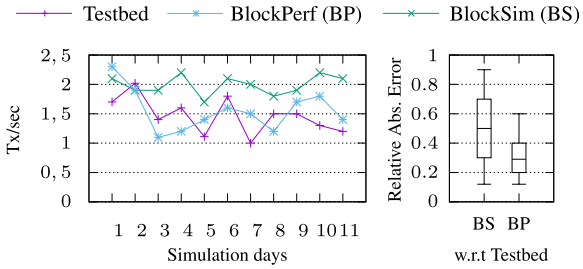


FIGURE 17. Throughput (o) & relative absolute error w.r.t Testbed.

allows to obtain a higher number of performance metrics. Also, by emulating the network layer, our approach replicates a more realistic behavior, thus leading to more realistic results than BlockSim. Finally, even if it is not yet the case, our approach is aimed at making the plugging of any blockchain technology easier, as our code is structured following the six-layer model described in FIGURE 3.

One may wonder whether the number of nodes used in our experiments is not too small. This parameter is not critical in this study as the objective is not about showing how efficient a given blockchain is (e.g., from a scalability standpoint), but rather on showing that the proposed simulator provides more realistic evaluation results than an existing one (BlockSim in this case). As a consequence, we limited the experiment to be large enough to capture a realistic scenario, and short enough to be manageable (in terms of costs).

One may also wonder whether the size of the datasets is large enough to draw relevant conclusions about the comparison analysis. In this regard, let us note that all the graphs given in FIGUREs 8 to 17 only report the average values of all the measurements obtained on a daily basis for the corresponding metrics. Just to give an indication about the number of measurements that have been collected every day, around 10000 measurements/day were collected regarding metrics (e) and (k_1), 200 measurements/day regarding (l_1)-(l_3), and about 50 regarding l_4. Given this amount of measurements, we confidently state that the conclusions drawn from our comparison analysis are relevant (statistically speaking).

V. CONCLUSION, LIMITATIONS & PERSPECTIVES

A. CONCLUSION

Today, a large variety of blockchain technologies is expanding in order to fulfill technical and non-technical needs and requirements. Within this context, determining and, most importantly, evaluating the characteristics/performance of a given blockchain platform is crucial for system designers before deploying it. In this respect, several blockchain simulators have been proposed in the literature over the past few years. Still, they are often limited in several respects (lack of extensibility, failure in covering all aspects/metrics underpinning a blockchain system, etc.). In this paper, the six-layer model introduced by [15] (Application, Contract, Incentive, Consensus, Node/Data, Network) is considered, against which 15 performance metrics have been mapped.

[5]Diameter: $max\left(dist\left(n_i, n_j\right)\right)\ \forall\, n_i, n_j \in \mathcal{N}$, with $\mathcal{N}$ the set of nodes (full and lightweight) from the network graph.

[6]Mean geodesic distance: $mean\left(dist\left(n_i, n_j\right)\right)\ \forall\, n_i, n_j \in \mathcal{N}$, with $\mathcal{N} = \{n_1, \ldots, n_N\}$ the set of nodes (full and lightweight) from the network graph.

[7]Clustering coefficient of entire graph: $C = \frac{1}{N}\sum_{i=1}^{N} C_i$, where C_i refers to the clustering coefficient of a node n_i that is calculated as follows: $C_i = \frac{2.L_i}{k_i(k_i-1)}$, k_i referring to the degree of node n_i and L_i to the number of edges between the k_i neighbors of n_i.

To overcome the limitations of state-of-the-art blockchain simulators, a new one called BlockPerf is proposed in this paper, an extension to the existing BlockSim simulator. BlockPerf tries to cover as much as possible the layers mentioned above along with its metrics. A comparative analysis with BlockSim is carried, which has demonstrated that BlockPerf provides more realistic results than BlockSim (e.g., at the Node/Data and Network layers) respectively improved by 39% and 55% in average. However, several limitations remain to be addressed in the future, as discussed in the next section. The comparison analysis conducted in this paper is built upon a benchmarking bitcoin scenario.

B. LIMITATIONS & PERSPECTIVES

Several limitations to BlockPerf are to be addressed for it to be a more exhaustive simulator. First, the deployment cost of deploying nodes within BlockPerf on different geographical locations is not a simple and straightforward task, as it requires careful planning and deployment of nodes, connecting them with the main interface within BlockPerf. However, such a layer is, in our opinion, crucial to obtain simulation results closer to reality.

Second, BlockPerf, in its current form, only supports Bitcoin-related experiments, and some efforts still remain to be done to allow for simulating different blockchain platforms. For example, a blockchain-based system such as the IOTA, which uses graph-based transaction chains, would require more extensive tuning of BlockPerf to replicate the effects of its real-world deployment.

Third, the *Contract layer* still remains to be developed within BlockPerf. The ability of deploying contracts has been a key element and argument within all the blockchain technologies that emerged over the last years, and being able to analyse them within the simulation environment would likely provide further insights on its usage. However, covering/integrating such a layer in a simulator is particularly challenging because each smart contract execution happens within a virtual computing environment (e.g., EVM in Ethereum), which is called by every node execute the instructions contained by that contract. This environment itself relies on other layers for execution, including the execution of certain transactions or changing the state of nodes, which makes it difficult to obtain realistic execution effects on the overall system, as discussed in [46]. Given this complexity, we believe that the adoption of a hybrid approach that uses the virtual computing environment as an emulation layer – *in a similar manner as done in BlockPerf with the network layer* – is an efficient way to cover/integrate the smart contract layer in a simulator.

REFERENCES

[1] A. A. Monrat, O. Schelén, and K. Andersson, "A survey of blockchain from the perspectives of applications, challenges, and opportunities," *IEEE Access*, vol. 7, pp. 117134–117151, 2019.

[2] J. A. Jaoude and R. G. Saade, "Blockchain applications–usage in different domains," *IEEE Access*, vol. 7, pp. 45360–45381, 2019.

[3] J. Al-Jaroodi and N. Mohamed, "Blockchain in industries: A survey," *IEEE Access*, vol. 7, pp. 36500–36515, 2019.

[4] U. Bodkhe, S. Tanwar, K. Parekh, P. Khanpara, S. Tyagi, N. Kumar, and M. Alazab, "Blockchain for industry 4.0: A comprehensive review," *IEEE Access*, vol. 8, pp. 79764–79800, 2020.

[5] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[6] C. Fan, S. Ghaemi, H. Khazaei, and P. Musilek, "Performance evaluation of blockchain systems: A systematic survey," *IEEE Access*, vol. 8, pp. 126927–126950, 2020.

[7] S. Maranhão, J.-M. Seigneur, and R. Hu, "Towards a standard to assess blockchain & other DLT platforms," in *Proc. ITU*, 2019.

[8] F. Gräbe, N. Kannengießer, S. Lins, and A. Sunyaev, "Do not be fooled: Toward a holistic comparison of distributed ledger technology designs," in *Proc. 53rd Hawaii Int. Conf. Syst. Sci.*, 2020.

[9] J. Polge, J. Robert, and Y. L. Traon, "Permissioned blockchain frameworks in the industry: A comparison," *ICT Exp.*, vol. 7, no. 2, pp. 229–233, Jun. 2021.

[10] W.-T. Tsai, R. Wang, S. Liu, E. Deng, and D. Yang, "COMPASS: A data-driven blockchain evaluation framwework," in *Proc. IEEE Int. Conf. Service Oriented Syst. Eng. (SOSE)*, Aug. 2020, pp. 17–30.

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash systemm," *Decentralized Bus. Rev.*, p. 21260, 2008.

[12] *A Next-Generation Smart Contract and Decentralized Application Platform*. Accessed: Apr. 13, 2021. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[13] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, and S. Muralidharan, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, p. 30.

[14] S. Smetanin, A. Ometov, M. Komarov, P. Masek, and Y. Koucheryavy, "Blockchain evaluation approaches: State-of-the-art and future perspective," *Sensors*, vol. 20, no. 12, p. 3358, Jun. 2020.

[15] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K.-L. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, May 2017, pp. 1085–1100.

[16] C. Faria and M. Correia, "BlockSim: Blockchain simulator," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 439–446.

[17] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in *Proc. IEEE 9th Int. Conf. Peer Peer Comput.*, Sep. 2009, pp. 99–100.

[18] A. Miller and R. Jansen, "Shadow-bitcoin: Scalable simulation via direct execution of multi-threaded applications," in *Proc. 8th Workshop Cyber Secur. Experimentation Test (CSET)*, 2015, pp. 1–8.

[19] L. Stoykov, K. Zhang, and H.-A. Jacobsen, "VIBES: Fast blockchain simulations for large-scale peer-to-peer networks," in *Proc. 18th ACM/IFIP/USENIX Middleware Conf., Posters Demos*, Dec. 2017, pp. 19–20.

[20] R. Paulavicius, S. Grigaitis, and E. Filatovas, "A systematic review and empirical analysis of blockchain simulators," *IEEE Access*, vol. 9, pp. 38010–38028, 2021.

[21] H. Huang, W. Kong, S. Zhou, Z. Zheng, and S. Guo, "A survey of state-of-the-art on blockchains: Theories, modelings, and tools," *ACM Comput. Surveys*, vol. 54, no. 2, pp. 1–42, Apr. 2021.

[22] *Technical Paper HSTP.DLT-RF: Distributed Ledger Technology: Regulatory Framework*. Accessed: Apr. 12, 2021. [Online]. Available: https://www.itu.int/dms_pub/itu-t/opb/tut/T-TUT-DLT-2019-RF-PDF-E.pdf

[23] *Blockchain and Distributed Ledger Technologies—Reference Architecture*, Int. Org. Standardization, Geneva, Switzerland, Apr. 2020.

[24] L. M. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," in *Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, May 2018, pp. 1545–1550.

[25] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 2567–2572.

[26] *DSLF: Distributed Ledger Simulation Framework*. Accessed: Jan. 25, 2021. [Online]. Available: https://github.com/i13-msrg/dlsf

[27] B. Wang, S. Chen, L. Yao, B. Liu, X. Xu, and L. Zhu, "A simulation approach for studying behavior and quality of blockchain networks," in *Proc. Int. Conf. Blockchain*. Cham, Switzerland: Springer, 2018, pp. 18–31.

[28] N. Agrawal, R. Prashanthi, O. Biçer, and A. Küpçü, "BlockSim-Net: A network based blockchain simulator," 2020, *arXiv:2011.03241*. [Online]. Available: http://arxiv.org/abs/2011.03241
[29] *Evibes Plasma Simulator*. Accessed: Jan. 25, 2021. [Online]. Available: https://github.com/i13-msrg/evibes-plasma
[30] A. Deshpande, P. Nasirifard, and H.-A. Jacobsen, "EVIBES: Configurable and interactive ethereum blockchain simulation framework," in *Proc. 19th Int. Middleware Conf. (Posters)*, Dec. 2018, pp. 11–12.
[31] M. Alharby and A. van Moorsel, "BlockSim: A simulation framework for blockchain systems," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 135–138, Jan. 2019, doi: 10.1145/3308897.3308956.
[32] S. Goswami, "Scalability analysis of blockchains through blockchain simulation," Univ. Nevada, Las Vegas, NV, USA, 2017.
[33] 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 *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 3–16.
[34] S. M. Fattahi, A. Makanju, and A. M. Fard, "SIMBA: An efficient simulator for blockchain applications," in *Proc. 50th Annu. IEEE-IFIP Int. Conf. Dependable Syst. Netw.-Supplemental Volume (DSN-S)*, Jun. 2020, pp. 51–52.
[35] Z. H. Chin, T. T. V. Yap, and I. K. T. Tan, "Simulating difficulty adjustment in blockchain with SimBlock," in *Proc. 2nd ACM Int. Symp. Blockchain Secure Crit. Infrastruct.*, Oct. 2020, pp. 192–197.
[36] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "SimBlock: A blockchain network simulator," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 325–329.
[37] R. Banno and K. Shudo, "Simulating a blockchain network with SimBlock," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 3–4.
[38] M. R. A. Lathif, P. Nasirifard, and H.-A. Jacobsen, "CIDDS: A configurable and distributed DAG-based distributed ledger simulation framework," in *Proc. 19th Int. Middleware Conf. (Posters)*, Dec. 2018, pp. 7–8.
[39] *Hive Simulation Tool*. Accessed: Jan. 25, 2021. [Online]. Available: https://github.com/ethereum/hive
[40] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2013, pp. 34–51.
[41] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.
[42] L. Zheng, X. Helu, M. Li, and H. Lu, "Automatic discovery mechanism of blockchain nodes based on the Kademlia algorithm," in *Proc. Int. Conf. Artif. Intell. Secur.* Cham, Switzerland: Springer, 2019, pp. 605–616.
[43] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys.*, vol. 74, p. 47, Jan. 2002.
[44] M. A. Javarone and C. S. Wright, "From bitcoin to bitcoin cash: A network analysis," in *Proc. 1st Workshop Cryptocurrencies Blockchains Distrib. Syst.*, Jun. 2018, pp. 77–81.
[45] G. Leduc, S. Kubler, and J.-P. Georges, "Innovative blockchain-based farming marketplace and smart contract performance evaluation," *J. Cleaner Prod.*, vol. 306, Jul. 2021, Art. no. 127055.
[46] J. Xu, "A simulator for heavy-duty smart contracts in blockchain," Ph.D. dissertation, Creative Compon., Iowa State Univ., Ames, IA, USA, 2021, p. 827.

JULIEN POLGE received the M.Sc. degree in complex systems engineering from the University of Lorraine, France, in 2017. He is currently pursuing the Ph.D. degree with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. He is part of the SEcurity, Reasoning and VALidation Group (SerVal) headed by Pr. Yves Le Traon. His research work is conducted in collaboration with the industry and focuses on the industrial Internet of Things, machine learning, and distributed ledger technologies.

SANKALP GHATPANDE received the M.S. degree from the University of Luxembourg, in 2016. He is a Research and Development Engineer with a position of Research Associate at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg. Since 2016, he has worked in industry and academic research and development projects, national and international, as an engineer focused on blockchain, the IoT, and cryptography. He has also been active in multiple volunteering projects, involving implementations of specific software(s), code-analysis, and multiple open source projects. Recently, he has been involved in data analysis and machine learning project within the financial domain with industrial partner.

SYLVAIN KUBLER (Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science and engineering from the Université de Lorraine, France, in 2009 and 2012, respectively. He is currently an Associate Professor at the Research Center for Automatic Control (CRAN), Université de Lorraine, and previously an Research Associate at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, from 2015 to 2017, and a Postdoctoral Researcher at Aalto University, from 2013 to 2015. He has broad expertise in the Internet of Things, semantic web, blockchain, and multicriteria decision support systems. He was awarded the Best Thesis in Automatic Control from the IFAC French Workgroup GdR MACS/Club EEA.

JÉRÉMY ROBERT received the M.Sc. and Ph.D. degrees in computer science and engineering from the University of Lorraine, France, in 2009 and 2012, respectively. He is currently a Digital Technology Specialist at Cebi Luxembourg S.A., an automotive and household appliances industry. He has broad expertise in industrial and embedded networks since his Ph.D. research focused on the use of switched Ethernet embedded in the future space launchers. Since 2015, his work was more about heterogeneous data communication challenges in the industrial Internet of Things and the implementation of messaging services and high-level data formats. As the major research work was conducted in collaboration with the industry, he naturally joined Cebi Luxembourg S.A. for supporting the deployment of the Industry 4.0 Project at the group level.

YVES LE TRAON (Senior Member, IEEE) is currently a Full Professor of computer science at the University of Luxembourg, in the domain of software engineering, with a focus on software testing, software security, and data-intensive systems. He was the Head of the CSC Research Unit, Department of Computer Science, from 2013 to 2016. He is currently the Vice-Director of the Interdisciplinary Centre for Security, Reliability and Trust Center (SnT) and the Head of the SEcurity, Reasoning and VALidation Group (SerVal), which is composed of around 25 researchers.