Appendable-block Blockchain Evaluation over Geographically-Distributed IoT Networks

Eduardo H. P. de Arruda*, Roben C. Lunardi*†, Henry C. Nunes*, Avelino F. Zorzo* and Regio A. Michelin‡

*Pontifical Catholic University of Rio Grande do Sul (PUCRS) - Porto Alegre, Brazil
†Federal Institute of Rio Grande do Sul (IFRS) - Porto Alegre, Brazil
‡Cyber Security CRC (CSCRC) - Sydney, Australia
E-mail: eduardo.arruda@pucrs.br, roben.lunardi@restinga.ifrs.edu.br, henry.nunes@edu.pucrs.br, avelino.zorzo@pucrs.br, regio.michelin@cybersecuritycrc.org.au

***Abstract*—In the last few years, different researchers presented proposals for using blockchain in the Internet of Things (IoT) environments. These proposals consider that IoT environments can be benefited from different blockchain characteristics, such as: resilience, distributed processing, integrity and non-repudiation of produced information. However, researchers faced some challenges to use blockchain in IoT, e.g., latency, hardware and energy constraints, and performance requirements. One of the prominent solutions is the appendable-block blockchain, which uses a hierarchical peer-to-peer (p2p) gateway-based architecture. Additionally, current proposals present simplified evaluation scenarios, usually performed in controlled environments, which do not include important network features, for example, latency. Consequently, a model to evaluate a geographically distributed environment, for example, in a situation in which health data have to be collected from different countries in a pandemic situation, can help to understand the behavior and possible flaws of blockchains. In order to evaluate appendable-block blockchains in a realistic scenario, this paper presents an analysis of different consensus algorithms in geographically distributed hosts, in which latency can impact the performance of main operations in a blockchain, such as block and transaction insertion.**

***Index Terms*—Blockchain, appendable-block, performance evaluation, consensus algorithms, geo-distributed IoT.**

I. Introduction

After promising results with cryptocurrencies, blockchain technology has also been applied in other fields [1] [2], such as government applications, supply chains, security services, and Internet of Things (IoT). All of them rely on some of blockchain characteristics, in particular, immutability (which guarantees that a transaction cannot be violated after being attached to the blockchain) and the distributed topology based on peer-to-peer (P2P) networking (avoiding single points of failure and the possibility of operating with a distributed trust system) [3]. Regarding applications of blockchain for IoT, several examples can be found in the literature [4] [5] [2] [6] [7].

Blockchain has a growing potential for applications in other areas, especially in those where it is necessary to operate on decentralized architectures dispensing trusted third parties. Therefore, exploring the potential of blockchain technology for building safer systems in different areas still runs up against some technical limitations in the existing frameworks. To illustrate, according to Zheng *et al.* [2], the limitation of 1MB of Bitcoin blocks as well as the low throughput of about 7 transactions per second limits its adoption in several different contexts.

To solve some of these limitations, SpeedyChain [5] [8] [9] introduced a blockchain design that decouples the data stored in the transactions from the block header, thus allowing faster insertion of data to the blocks. Furthermore, this kind of blockchain was called appendable-block blockchain [9]. Initially, it was presented as a solution for IoT environments, such as Smart Offices and Smart Homes [5]. Later on, it was adapted to allow vehicles to share their data while maintaining privacy, integrity, resilience, and non-repudiation in decentralized Intelligent Transportation Systems (ITS) for Smart Cities [8].

However, SpeedyChain was evaluated only in local networks or through emulated scenarios, not considering network latency and communication in geographically distributed scenarios. Consequently, the evaluation of different consensus algorithms can be impacted when considering geographically distributed IoT networks [10], in special the consensus algorithms that use a high number of messages. Consequently, a misleading latency evaluation can lead to a not properly chosen consensus algorithm.

In order to understand the behavior of appendable-block blockchain in a geographically distributed scenario, this work presents the evaluation of different consensus algorithms on SpeedyChain. This type of scenario is very important, for example, to control a global disease in a pandemic situation; every country could send data to the closest host possible; and, these data would be collected from different patients, for example, through IoT devices in a wireless network.

The environment was built on a cloud-based distributed commercial network infrastructure using Amazon Web Services (AWS) [11]. Also, the results obtained are compared to previously presented results for SpeedyChain to verify the impact that latency can have to appendable-block blockchains.

This paper was achieved in cooperation with HP Brasil using incentives of Brazilian Informatics Law (Law n 8.248 of 1991). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. Avelino F. Zorzo is supported by CNPq (315192/2018-6) and FAPERGS. This work was supported by the INCT Forensic Sciences through the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq − process # 465450/2014-8). Also, we thank the support from IFRS and Cyber Security Research Centre Limited whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme.

978-1-7281-7127-2/20/$31.00 ©2020 IEEE

II. Related Work

Different researchers proposed the adoption of blockchain in IoT [3] [5] [6]. However, as presented by Dedeoglu *et al.* [12], one of the main challenges for blockchain is scalability, where there is a trade-off between block size and network usage. Both of those impact overall performance. Additionally, they describe two ways to address the scalability problem: storage optimization and blockchain redesign.

The emergence of proposals in both of these fields (storage optimization and blockchain redesign) increased an already existing necessity to create standardized metrics that can be used to evaluate performance in different blockchains and applications. Furthermore, there are different aspects of blockchain that can influence performance and functionality. Some examples are block size, network configuration, application, access rights, cryptography algorithms, and consensus algorithms. This variability is a challenge since new metrics that can be used to compare different blockchain proposals have to be created. Hence, industry and academia have been working on this issue.

From the industry, the Hyperledger project [13] released a white paper that specifies metrics for analysis of blockchain performance [14]. These metrics can be used to measure any blockchain. The content covers the network topology for tests, workloads, and transactions.

Pongnumkul *et al.* [15] proposed a methodology to evaluate blockchains performance. In their work, the methodology is applied to two blockchains in a private environment: Ethereum and Hyperledger Fabric. According to the results, Hyperledger Fabric outperforms Ethereum.

Rouhani and Deters [16] presented a performance analysis towards a specific blockchain. In their work, a performance analysis between two different client node applications for the Ethereum blockchain is presented, *i.e.* Geth and Parity. Their conclusion mentions that there was a significant difference in the performance of each implementation, which is an aspect to be considered in the analysis. There are, also, some research that present performance analysis of the Hyperledger Fabric. There were different models proposed to predict the performance of Hyperledger in different topologies [17] [18].

The performance of inner working parts of blockchain were presented by Hao *et al.* [19]. In their work, the performance of different consensus algorithms was evaluated. Their work present an analysis of the Practical Byzantine Fault Tolerance (PBFT) and Proof-of-Work (PoW) consensus algorithms using the Hyperledger Fabric and Ethereum blockchains.

A testbed for testing a blockchain or a smart contract application was proposed by Shbair *et al.* [20]. Their results show that the testbed provides results close to the ones from a production environment. A framework for blockchain monitoring is proposed by Zheng *et al.* [2]. This framework includes metrics for monitoring overhead over logs collection.

Most of the related work present methodologies for performance analysis. Also, some works proposed tools to improve quality in performance analysis for blockchains. However, there is a lack of works tackling problems in the adoption of blockchains in geographically distributed IoT environments. These scenarios are subject to some network restrictions, such as communication latency times. Therefore, this work presents an evaluation testbed focused on IoT blockchain based on public cloud computing services.

III. Appendable-block blockchains

In this section, we present the fundamental concepts of appendable-block blockchain architecture and relevant details about the SpeedyChain framework [8] [9]. The discussion about appendable-block blockchains is presented considering the blockchain layer-based model proposed by Zorzo *et al.* [21]. This model is composed of four layers: Communication, Consensus, Data and Application Layer.

A. Communication Layer

Appendable-block blockchains adopt a layer-based IoT architecture [22]. This architecture compromises a (*i*) Perception Layer, in which devices interact with the physical world; (*ii*) a Transportation Layer, in which data are communicated to and among gateways; and (*iii*) an Application Layer, in which Service Providers interact with external users. In this architecture, devices produce data and send them to the gateways to append these data to the blockchain. Thus, devices can produce information and request gateways to append that into the blockchain. Each device can perform that in parallel. Service Providers can access the blockchain information only through the gateways.

B. Consensus Layer

It is possible to use different consensus algorithms in an appendable-block blockchain. The consensus algorithm should provide trust in an untrusted environment. Also, it should guarantee that the information inserted in the blockchain is valid. In this work, we assume that this validation is performed by the gateways through predefined rules and used during the verification by consensus algorithms.

Currently, there are different consensus algorithms used by blockchains. For this work, three different consensus algorithms were evaluated in SpeedyChain: (*i*) PBFT, (*ii*) dBFT, and (*iii*) PoW.

1) PBFT: Practical Byzantine Fault Tolerance (PBFT) consensus algorithm has the objective to reach consensus even when a subset of the nodes are faulty or malicious in a distributed network. When there are f faulty nodes in the network, PBFT requires $3f+1$ nodes to correctly reach consensus.

In the blockchain, when a new block is created, a leader node is selected to perform consensus. Then, the leader node starts the consensus mechanism by sending the block to the active validation nodes in the network for validation. If more than 2/3 of the active validation nodes vote to validate the new block, the block is appended to the blockchain [23]. PBFT has been used by many blockchain proposals for IoT in the last few years. However, PBFT mechanism suffers from poor scalability. In a large network, the number of messages and the waiting time for node responses can be high. Additionally, in a dynamic P2P scenario, where nodes frequently leave and rejoin the network, achieving consensus becomes difficult as active nodes can change their status during the consensus.

2) dBFT: Delegated Byzantine Fault Tolerance (dBFT), similar to PBFT, achieves consensus on new information based on votes. However, in the dBFT, validators (nodes that validate and vote) are elected by the requester for each consensus. If the requester does not trust a chosen validator, the first one can elect another node as a validator for the next consensus procedure. Then, the validators choose a node to be the leader that will create the block and start the consensus procedure. Consequently, just a small subset of the nodes are used to perform the consensus in dBFT [24]. When more than 2/3 of elected nodes validate the information, it is considered valid.

The adapted version of dBFT in SpeedyChain is similar to what was presented in PBFT. Some small differences are:

- A new leader is only elected if the leader presented a malicious behavior. Consequently, the leader will perform the consensus while it is trusted by other delegate nodes;
- Delegates are a set of gateways that will perform the consensus. Delegates are elected by the other gateways. As a simplification, a random selection was implemented. Nonetheless, a zero-knowledge voting using Smart Contracts could also be implemented;
- More than 2/3 of delegates should send positive votes;
- Every delegate gateway sends its vote to every gateway that is a delegate (and not to every gateway).

3) PoW: Proof-of-Work (PoW) consists of solving a resource-consuming puzzle to avoid overload of data in nodes that will share information [25]. Usually, the puzzle is a generation of a hash value for a specific data, and in blockchain scenario, for data contained in a block, varying a nonce value (a variable used for the purpose to generate different hash values) in order to obtain at least a predefined number of bits equal to zero at the beginning of the generated hash value.

After the block is created, it is broadcast to other peers, and it can be easily verified (it can compare the generated hash with the hash of the block). Also, the difficulty of the work can be adjusted over the time, *i.e.*, the number of zeros at the beginning of the target hash can be increased when the median time to insert a new block is lower than a predefined threshold. The PoW algorithm was adapted to SpeedyChain since it does not consider any incentive for the mining procedure. Also, it does not use any mechanism to adapt the algorithm difficulty, *i.e.*, the difficulty is fixed.

C. Data Layer

One important aspect to be considered in blockchains for IoT is how the data is structured in the blockchain. Additionally, there are differences when considering which cryptography algorithm is adopted for different purposes in the blockchain. Based on existent approaches, there are differences in the data layer adopted by different blockchain solutions [26] [5] [27] [28].

Initially, blockchain proposals structured their data through blocks that are composed by a block header and a summary (*e.g.*, using a Merkle Tree) of the transactions. In SpeedyChain the transactions are stored inside blocks, where the first transaction is linked to the block header (through a block header hash), while other transactions are linked to the previous transaction, through the hash of the previous block, creating a hash chain for transactions. Thus, this leads to an appendable block, *i.e.*, a block that can still receive new transactions after it was inserted into the blockchain. However, after a new block header is created, it is linked by a hash value, preserving its integrity. Also, new information inserted into the block ledger is both signed and hashed, guaranteeing both integrity and non-repudiation.

Cryptography algorithms, *e.g.*, asymmetric/symmetric ciphers and hash functions, are very important to appendable-block blockchains to ensure security (data integrity and data privacy). For example, Bitcoin uses the Elliptic Curve Digital Signature Algorithm (ECDSA) [29] in order to create the public/private key pair (used as a wallet address) and SHA-2 [30] hash function for the block hash. In the case of the Speedychain appendable-block blockchain [8], it relies on RSA [31] for the asymmetric cryptography algorithm and SHA-2 [30] as hash function.

D. Application Layer

The "Application" layer is responsible for managing the different applications that use a blockchain, for example, cryptocurrencies or data management.

A concept that has opened new perspectives for developing applications using blockchains is the "smart contract". A smart contract allows the execution of code inside a blockchain without centralized control. Once in the blockchain, the smart contract will be permanent, and it cannot be altered [32].

Since business logic can be applied to a smart contract, it has ample scope of applications, such as resource allocation, traceability, and auditability [32]. For example, a smart contract can be deployed and made accessible to a specific manufacturer of IoT devices. In the smart contract, for example, the device can check the last version of firmware available and receive a hash of the newest version. Although smart contracts are important, they are still under development in SpeedyChain [33]. In this paper, the execution of smart contracts will not be discussed.

IV. Performance Evaluation

Currently, SpeedyChain was evaluated both by Michelin *et al.* [8] and Lunardi *et al.* [9] using an emulated network specified on the CORE emulator [34]. The evaluation on both works did not consider variables present in real network environments, such as latency in the communication between nodes. Also, due to the use of an emulated environment, built on a single VM running over a single physical host, the performance evaluation may have been influenced by the fact that the gateways competed for the same resources (processor, memory and I/O) of the VM and, consequently, of the physical host. In order to help to understand these issues, we present in the next subsections the tested environment, metrics, test procedures, and obtained results in an actual cloud environment. We believe that a similar setting can be used to measure performance of IoT environments that use blockchains.

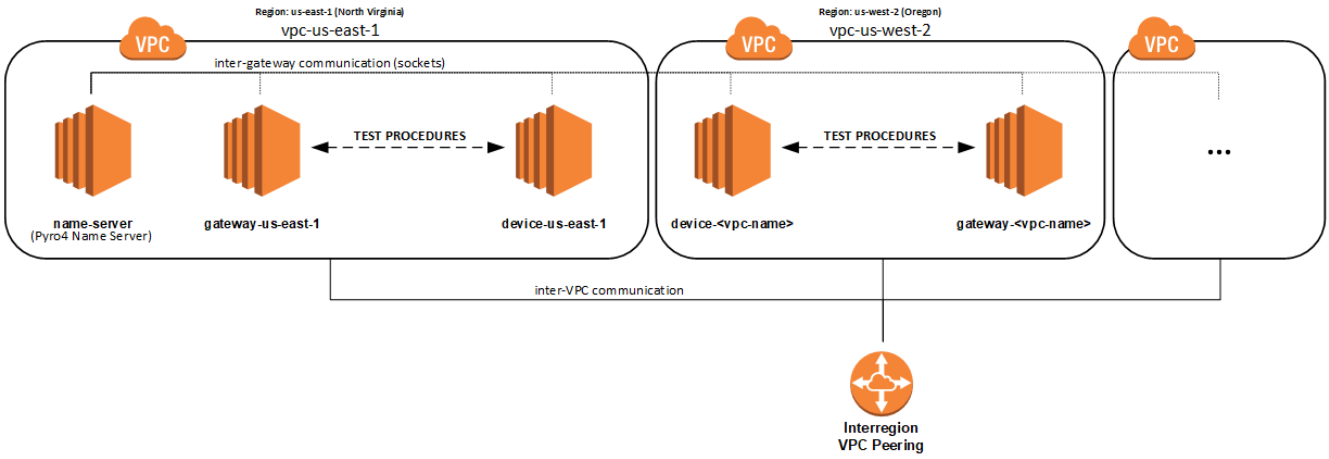

Fig. 1. Testbed environment

A. Tested Environment

In this work, we perform SpeedyChain performance tests in a geographically distributed IoT environment created on the infrastructure of a commercial cloud services provider. The evaluation environment, shown in Figure 1, is composed by Amazon Web Services (AWS) Virtual Private Clouds (VPCs) geographically distributed in different AWS regions.

Due to limitations imposed by Amazon, which determines that the use of certain regions should be preceded by an explicit authorization requested by the AWS account manager, the VPCs were created only in 4 regions: Asia Pacific (Tokyo), South America (São Paulo), US East (N. Virginia), and US West (Oregon).

SpeedyChain was encoded using the Python programming language version 2.7. It uses the Pyro4 [35], a native Python library that enables to build applications using Remote Procedure Calls (RPCs) to connect gateways and devices in a peer-to-peer network. Thus, it is necessary to have an instance of the Pyro4 Name Server accessible to all gateways and devices connected to the SpeedyChain network, as shown in Figure 1 to help to identify gateway address. In our test environment the name server (name-server - Pyro4 Name Server) runs in the AWS region us-east-1 (North Virginia).

In each AWS region, there is a VPC composed by an EC2 VM running instances of the SpeedyChain gateway and another EC2 VM to simulate the devices. In total, like the simulation performed in Michelin *et al.* [8], the test environment consists of 15 gateway instances. However, unlike that previous work, in this work, the network traffic between gateways, and between gateways and devices is subject to latency times and other characteristics of geographically distributed IoT networks. The evaluation of latency and other characteristics, including the analysis of AWS Service Level Agreements (SLAs), is beyond the scope of this work. Gateways located in the same VPC, that is, in the same network infrastructure, and gateways distributed in other VPCs, that is, geographically distributed between regions are subject to different latency times.

An automation script using the AWS CloudFormation service was created to generate the testing environment. Thus, all the VPC network components were specified in a JSON templateand CloudFormation Stacks were then created based on the template in each of the regions.

B. Metrics

In order to perform a broader evaluation of SpeedyChain, the time required to perform various operations involved in the insertion of new blocks and new transactions was measured:

- **T1**: Time to process and to add a new transaction into a local gateway;
- **T2**: Time to add a new transaction into a remote gateway;
- **T3**: Time to verify information from a device and to create a new device block (before consensus);
- **T4**: Time to add a new device block into the leader gateway (after consensus);
- **T5**: Time to add and to replicate a device block into all gateways (after consensus).

The measured times were collected using the Python Logger library, which stores the collected data in text format files. The impacts that the collection and time recording operations of the library introduce to the SpeedyChain operation are not evaluated, but we can expect them to be relatively low due to the simplicity of these operations.

C. Test Procedures

The test procedures are shown in Table I. Figure 2 shows a graphical representation of the test procedures. Due to time constraints, all test procedures performed simulated devices connected to the AWS region us-east-1 (North Virginia).

D. Results

After the execution of all planned test procedures and the collection and consolidation of the logs, the average times for each defined performance indicator were calculated. All metrics are represented by an average time (in milliseconds) over a minimum of one hundred repetitions for each scenario, with a confidence level of 95%, achieving the confidence intervals presented in Table II.

TABLE I
TEST PROCEDURES.

Devices	Transactions	Consensus
50	10	None
50	10	PoW (12 bits)
50	10	PoW (16 bits)
50	10	dBFT
50	10	PBFT

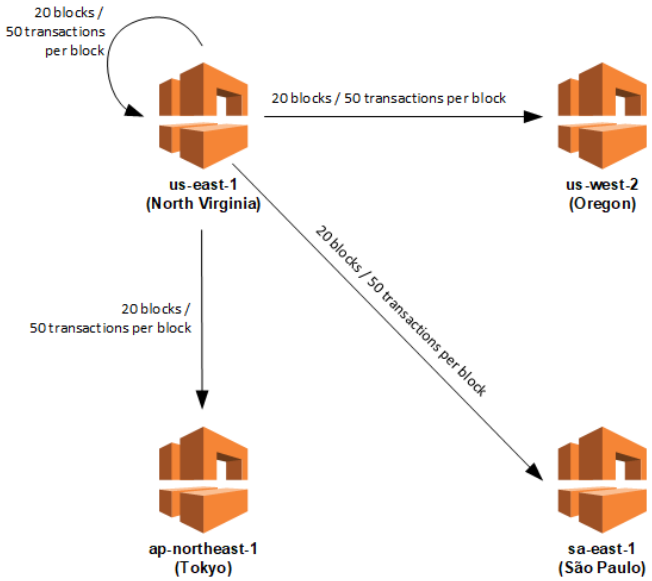

Fig. 2. Test Procedure

For the **T1** metric - Time to process and to add a new transaction into a local gateway, Table II shows that there is no significant variation in the times measured among the consensus algorithms since it is a local operation that occurred before the consensus execution.

The same is true for the **T2** metric - Time to add a new transaction into a remote gateway, since this metric is also related to a local operation, *i.e.* a remote gateway must store locally a transaction received from another gateway.

T3 metric - Time to verify information from a device and to create a new device block (before consensus), is also a local operation executed prior to the execution of the consensus algorithm, so it should not be affected by that (see Table II).

Differently, the **T4** metric - Time to add a new device block into the leader gateway (after consensus), represents the execution of the consensus algorithm, so it is largely influenced by communication latency, except in the test scenario in which there is no consensus algorithm execution (see Table II). By the obtained results, it is possible to observe that the execution time of the PoW algorithm has a direct influence on the difficulty derived from the number of nonce challenge bits. When increasing that number from 12 to 16 bits, the time spent increased less than expected, *i.e.* around 30%.

The **T5** metric - Time to add and to replicate a device block to all gateways (after consensus), is directly influenced by network latency times, since it is the measurement of the time needed to replicate a block of a device to the other SpeedyChain nodes (see Table II).

E. Discussion

Both Michelin *et al.* [8] and Lunardi *et al.* [9] evaluated SpeedyChain in an emulated network environment. In that context, the results were not influenced by high latency times and other variables present in distributed IoT environments. Consequently, in those test scenarios, the time required to validate and register a device block with 10 transactions was close to 20ms [8] without consensus and around 102ms using PBFT [9]. In the present work, the measured times ranged from 1,442.5ms, without the execution of a consensus algorithm, to 11,607.8ms with the PBFT consensus algorithm. Hence, we can show the impact that network latency in real environments has on the performance of the SpeedyChain blockchain. This same impact may affect different blockchains.

Due to messages exchanged during the consensus algorithms, latency has impact even in the PoW algorithm, which does not present a high number of exchanged messages. This could be observed when using a low difficulty (12 bits) in the PoW algorithm, in which more than 4,000ms were required to include a new block into the blockchain. The usage of very small difficulty in the PoW algorithm was used to show that latency is an important factor during the consensus procedure.

Consequently, this work shows that current consensus algorithms used in SpeedyChain can be used to geographically distributed IoT environments when latency to create new blocks can be around few seconds. It is important to remember that block insertion is performed only in the first time that a device connects to a gateway or when a device needs to change its key pair. Additionally, improvements can be achieved adapting other consensus algorithms that consider environments with high latency.

TABLE II
RESULTS SUMMARY

Consensus	T1	T2	T3	T4	T5
None	4.1473 ± 0.0046	0.6873 ± 0.0033	0.0336 ± 0.0003	N/A	1442.5268 ± 1.0525
PoW (12 bits)	4.0365 ± 0.0049	0.5806 ± 0.0007	0.0337 ± 0.0003	1470.8114 ± 6.3921	4314.9415 ± 8.4733
PoW (16 bits)	4.2066 ± 0.0110	0.5818 ± 0.0007	0.0334 ± 0.0003	1911.1309 ± 96.2341	4736.4641 ± 96.5417
dBFT	4.0617 ± 0.0091	0.6142 ± 0.0010	0.0332 ± 0.0004	5162.4721 ± 0.8595	8230.6308 ± 10.0098
PBFT	4.1144 ± 0.0045	0.6514 ± 0.0008	0.0335 ± 0.0003	4250.0792 ± 182.3843	11607.8835 ± 234.3939

Times in milliseconds

V. Final Considerations & Future Work

Blockchain is a promising technology that is being adopted in different contexts. Consequently, different proposals emerged to be used in a variety of contexts. In special, blockchains for IoT proposals have to deal with processing, energy, communication and memory requirements that differ from other applications.

In order to evaluate appendable-block blockchains in a geographically distributed IoT, we used SpeedyChain with a cloud-based environment. In this scenario, we demonstrated that latency can be an important factor to be considered during the choice of consensus algorithms. Although results have shown that the SpeedyChain performance was affected by geographically distributed nodes, it appended new blocks in a few seconds.

In future work, we intend to perform new testing procedures in order to cover different scenarios for SpeedyChain usage. Also, we intend to perform tests with different consensus algorithms, such as the Federated Byzantine Agreement (FBA) and Cross Fault Tolerance (XFT) that can have a reduced exchanged messages compared to PBFT and dBFT. We also intend to propose possible improvements to the SpeedyChain blockchain in order to support higher latency for geographically distributed IoT. This can be achieved adapting consensus algorithms such as XFT or Ripple Protocol Consensus.

References

[1] F. Casino, T. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics and Informatics*, vol. 36, pp. 55–81, 2019.

[2] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, June 2017, pp. 557–564.

[3] M. Singh, A. Singh, and S. Kim, "Blockchain: A game changer for securing iot data," in *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Feb 2018, pp. 51–55.

[4] A. Dorri, S. S. Kanhere, and R. Jurdak, "Towards an Optimized BlockChain for IoT," in *2017 Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*. ACM, 2017, pp. 173–178.

[5] R. C. Lunardi, R. A. Michelin, C. V. Neu, and A. F. Zorzo, "Distributed access control on IoT ledger-based architecture," in *2018 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, April 2018, pp. 1–7.

[6] M. Seliem and K. Elgazzar, "Biomt: Blockchain for the internet of medical things," in *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, June 2019, pp. 1–4.

[7] V. Dedeoglu, R. Jurdak, A. Dorri, R. C. Lunardi, R. A. Michelin, A. F. Zorzo, and S. S. Kanhere, *Blockchain Technologies for IoT*. Singapore: Springer Singapore, 2020, pp. 55–89.

[8] R. A. Michelin, A. Dorri, M. Steger, R. C. Lunardi, S. S. Kanhere, R. Jurdak, and A. F. Zorzo, "Speedychain: A framework for decoupling data from blockchain for smart cities," in *2018 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, 2018, pp. 145–154.

[9] R. C. Lunardi, R. A. Michelin, C. V. Neu, H. C. Nunes, A. F. Zorzo, and S. S. Kanhere, "Impact of consensus on appendable-block blockchain for iot," in *2019 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*. Association for Computing Machinery, 2019, p. 228–237.

[10] S. Imai, C. A. Varela, and S. Patterson, "A performance study of geo-distributed iot data aggregation for fog computing," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Dec 2018, pp. 278–283.

[11] Amazon Web Services, Inc., *AWS Documentation*, Feb. 2020. [Online]. Available: https://docs.aws.amazon.com/index.html

[12] V. Dedeoglu, A. Dorri, R. Jurdak, R. A. Michelin, R. C. Lunardi, S. S. Kanhere, and A. F. Zorzo, "A journey in applying blockchain for cyber-physical systems," in *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*, 2020, pp. 383–390.

[13] Linux Foundation, "Hyperledger," Feb. 2020. [Online]. Available: https://github.com/hyperledger

[14] Linux Foundation , "Hyperledger blockchain performance metrics," Feb. 2020. [Online]. Available: https://www.hyperledger.org/resources/publications/blockchain-performance-metrics

[15] S. Pongnumkul, C. Siripanpornchana, and S. Thajchayapong, "Performance analysis of private blockchain platforms in varying workloads," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, July 2017, pp. 1–6.

[16] S. Rouhani and R. Deters, "Performance analysis of ethereum transactions in private blockchain," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Nov 2017, pp. 70–74.

[17] A. Baliga, N. Solanki, S. Verekar, A. Pednekar, P. Kamat, and S. Chatterjee, "Performance characterization of hyperledger fabric," in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, June 2018, pp. 65–74.

[18] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of hyperledger fabric (permissioned blockchain network)," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, Nov 2018, pp. 1–8.

[19] Y. Hao, Y. Li, X. Dong, L. Fang, and P. Chen, "Performance analysis of consensus algorithm in private blockchain," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018, pp. 280–285.

[20] W. M. Shbair, M. Steichen, J. François, and R. State, "Blockzoom: Large-scale blockchain testbed," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2019, pp. 5–6.

[21] A. F. Zorzo, H. C. Nunes, R. C. Lunardi, R. A. Michelin, and S. S. Kanhere, "Dependable IoT using blockchain-based technology," in *2018 Eighth Latin-American Symposium on Dependable Computing (LADC)*, Oct. 2018, pp. 1–9.

[22] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, Nov 2014.

[23] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *1999 Third Symposium on Operating Systems Design and Implementation (OSDI)*, 1999, pp. 173–186.

[24] T. Crain, V. Gramoli, M. Larrea, and M. Raynal, "(Leader/Randomization/Signature)-free Byzantine Consensus for Consortium Blockchains," *CoRR*, vol. abs/1702.03068, 2017. [Online]. Available: http://arxiv.org/abs/1702.03068

[25] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Advances in Cryptology (CRYPTO)*, E. F. Brickell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 139–147.

[26] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," *Www.Bitcoin.Org*, p. 9, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[27] E. Foundation, "Ethereum White Paper," Feb. 2020. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[28] I. Foundation, "IOTA - Next Generation Blockchain," Feb. 2020. [Online]. Available: https://iota.org/

[29] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, Aug 2001.

[30] H. Handschuh, *SHA Family (Secure Hash Algorithm)*. Boston, MA: Springer US, 2005, pp. 565–567.

[31] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[32] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[33] H. C. Nunes, R. C. Lunardi, A. F. Zorzo, R. A. Michelin, and S. S. Kanhere, "Context-based smart contracts for appendable-block blockchains," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, May 2020, pp. 1–9.

[34] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim, "Core: A real-time network emulator," in *2008 IEEE Military Communications Conference (MILCOM)*. IEEE, 2008, pp. 1–7.

[35] Irmen de Jong, *Pyro - Python Remote Objects - 4.60 Documentation*, 2020. [Online]. Available: https://pythonhosted.org/Pyro4