A Blockchain Simulator for Evaluating Consensus Algorithms in Diverse Networking Environments

Peter Foytik
Old Dominion University
Norfolk, VA USA
pfoytik@odu.edu

Deepak Tosh
University of Texas at El Paso
El Paso, Texas USA
dktosh@utep.edu

Sachin Shetty,
Eranga Herath,
Sarada Prasad Gochhayat
Old Dominion University
Norfolk, VA USA
(sshetty, cmedawer, sgochhay)@odu.edu

Laurent Njilla
Air Force Research Lab
Rome, NY USA
laurent.njilla@us.af.mil

***Abstract*—The massive scale, heterogeneity and distributed nature of Internet-of-Things (IoT) presents challenges in realizing a practical and effective security solution. Blockchain empowered platforms and technologies have been proposed to address this challenge. In order to realize a practical Blockchain deployment for IoT, there is a need for a testing and evaluation platform to evaluate performance and security of Blockchain applications and systems. In this paper, we present a Blockchain simulator that evaluates the consensus algorithms in a realistic and configurable network environment. Though, there are several Blockchain evaluation platforms, they are either wedded to a specific consensus protocol and do not allow evaluation in a configurable and realistic network environment. In our proposed simulator, we provide the ability to evaluate the impact of the consensus and network layer that will inform practitioners on the appropriate choice of consensus algorithms and the impact of network layer events in congested or contested scenarios in IoT. To accomplish this a generalized representation for consensus methods is proposed. The Blockchain simulator uses discrete event simulation engine for fidelity and increased scalability. We evaluate the performance of the simulator by varying the number of peer nodes and number of messages required to find consensus.**

***Index Terms*—Blockchain, Internet-of-Things, Consensus, Distributed ledger, Simulator, NS3**

I. Introduction

Blockchain is a distributed digital ledger for keeping track of transactions, contracts and is characterized by its inherent transparency, resilience and immutable properties. These properties are desirable to realize a secure Internet-of-Things (IoT) architecture capable of ensuring auditability and accountability. Blockchain has the potential to track authenticity of data transactions among millions of devices. However, deployment of IoT devices in congested and contested environments can cause performance challenges. In addition to the properties that are already baked into the Blockchain technology, there are several claims about high transaction throughput and faster response time. Recent incarnations of the Blockchain technology has also manifested into a highly complex distributed system like IoT and it is unclear if the claimed advantages are due to the strengths of the fundamental components of Blockchain or the additional system implementation. There is a need to provide a quantitative analysis to validate the claimed performance advantages of a practical Blockchain platform prior to deploying the technology in a production environment.

Researchers and practitioners have proposed several Blockchain development platforms, simulation environments, and testbeds to evaluate Blockchains and the applications for various sectors [1], [2]. These environments analyze security and performance [3], scalability based on parameter tuning [4], transaction latency for PoW [5] and quality of the infrastructure [6]. These platforms and testbeds range from an open deployment on publicly accessible machines to permissioned or privately owned peer-to-peer networks and simulated architecture environments to support a wide range of Blockchain based applications. However, as researchers and practitioners have pointed out the main performance benefits of Blockchain are closely tied to the choice of consensus plane and the network plane [7]. The aforementioned environments offer a static network topology and do not provide the ability to configure the network plane, as needed, to evaluate the feasibility in resource constrained environments, such as, IoT. In addition, the consensus plane is not configurable, which therefore limits the ability to deploy hybrid consensus protocols in an on-demand basis. In addition, Blockchain technology's resilience property can only be put to test in the presence of a configurable network plane. This capability would prove the effectiveness for applications that would like to measure the resilience of the Blockchain technology against various network attacks and fragility of network.

Blockchain platforms, including the Hyperledger Suite [8], Corda [9], and Tendermint [10], etc., typically abstract the network plane for easy development of application specific use cases, which makes the life of application developer easy and simple. However, this abstraction can have major bottlenecks when such Blockchain platforms are incorporated in a resource constrained environments, such as, IoT. For instance, while evaluating Hyperledger Sawtooth platform in a simulated environment, with 6 validator nodes sending transactions at different rates, we observe that the average number of packets sent from the nodes grow exponentially, as seen in Fig. 1. For instance, if each node sends 10 transaction at every 2 seconds, the total number of packets sent under

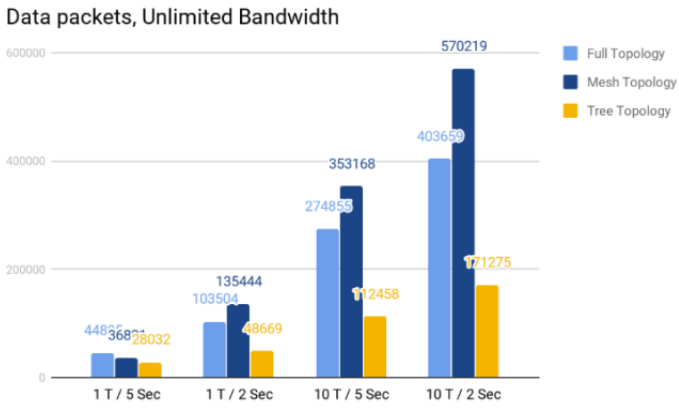

Fig. 1. Number of Packet Transmitted in Hyperledger Sawtooth

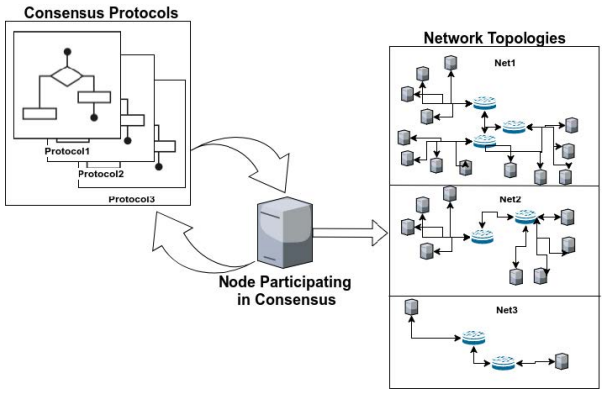

Fig. 2. Consensus and Network Layers in Blockchain Simulator

a completely connected network is more than 400K. The performance is highly dependent on the considered network topology and the underlying consensus scheme. Although, this evaluation is conducted for only the Hyperledger Sawtooth based on the Proof-of-Elapsed-Time (PoET) consensus model, the network plane metrics for other consensus models need to be investigated so we can better understand the feasibility of Blockchain integration in various resource constrained and diverse network connectivity environments.

The impact of the vagaries in the network layer on the Blockchain performance, especially, in IoT environments, has motivated us to develop a Blockchain simulator allows to enable a configurable network layer to evaluate the performance of selected consensus protocols. In this paper, we propose the development of a Blockchain simulator that utilizes a generalized representation of consensus protocols that would provide insights into the performance of the consensus protocols under various networking conditions. The simulation would provide users the ability to swap in and out various consensus protocols, configure the network settings, and assess the performance of the consensus protocols. The paper will present an approach to generalize consensus protocols, then discuss simulating a Blockchain based system using the generalized consensus protocol, the implementation of the simulator will be described, and finally the results will be presented. The paper shows that the generalized and simulated system was able to calibrate to the observed consensus protocol, and a sensitivity analysis is shown to provide some insight into the input effects to the simulator.

The paper is organized as follows. Section II presents the overview of the methodology followed to design the proposed simulator. In Section III, the implementation details and data collection techniques are described. Results related to Raft consensus model implemented in the simulator is presented in Section IV. Finally, Section V concludes the paper along with future plans on the proposed simulator.

II. Methodology

A. Overview

A set of abstraction layers, such as (1) Network; (2) Consensus; (3) Storage; and (4) View and Side, have been proposed [11] in past to provide a hierarchical design structure for the decentralized Blockchain ecosystem. Testing performance of specific consensus based algorithms is studied in the literature [12] [3], but a general model for network layer evaluation of Blockchains is yet to be proposed. It is important to understand the complexity and performance metrics at each layer prior to deployment of Blockchain-based solutions. Therefore, we propose the development of a simulator that implements these layers and provides statistical insights on performance and security metrics via realization of the Network and Consensus layer.

Our choice of focusing on consensus and network layer is driven by the need and potential of Blockchain deployment in IoT environments where nodes are heterogeneous and have stringent constraints on networking, computation, and communication capabilities. Given the bottom-up design [11], the consensus and network layer is the core of Blockchain platforms and its cost/complexity needs must be analyzed prior to integration of the additional layers. This approach will provide inner insights for the community of Blockchain adopters about the performance and security metrics of the underlying consensus protocols under various network conditions. In Fig. 2, the architecture overview is presented which incorporates different network topology for establishment of a P2P network and implements various distributed consensus protocols for evaluation of performance metrics. We present an overview of the proposed architecture wherein consensus protocols and the network architectures are interchangeable.

B. Establishing Discrete Event Simulator

We propose a discrete event simulator to implement the consensus and network layers wherein the simulation time step is considered to be system events rather than regular time steps to facilitate quicker computation. The design goals considered while developing the simulator are:

1) consensus protocols need to be generalized,

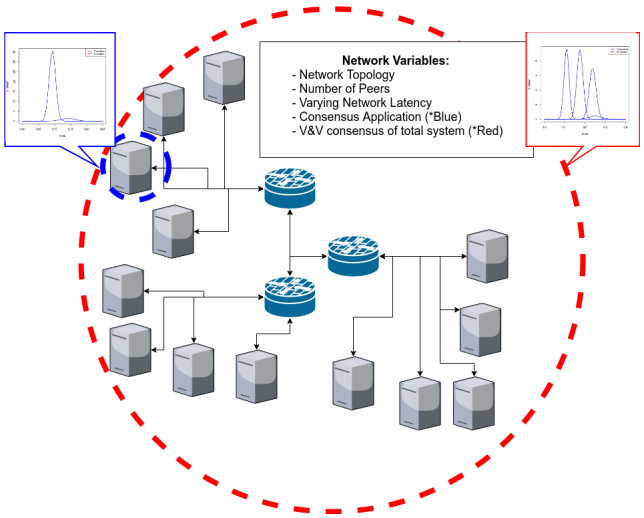

Fig. 3. Demonstration of the data collection from the ideal system (Blue indicating data gathered from individual nodes and red indicating data gathered from total system)

2) underlying network needs to be re-configurable,
3) consensus protocols need to be modeled to work with the underlying network topology,
4) simulator must be scalable to facilitate incorporation of heterogeneous scenarios and diverse network conditions.

The interchangeability of consensus protocols will ensure that the key aspect to the Blockchain systems performance is considered and evaluated instead of efficiency of the whole platform itself. The simulator will contribute to the benchmarking of consensus protocols for different network topologies. Using the simulator, developers will be able to make informed decisions on which consensus protocol would be preferred in certain types of network environments where the Blockchain is considered for by understanding the performance metrics of each consensus, which is discussed in the following subsection.

1) Performance Metrics: The key metrics considered in the simulator are discussed briefly in the following.

- *Throughput*: Number of successful transactions/second.
- *Latency*: Response time/transaction.
- *Fault Tolerance*: Operational resilience of the system to be able to meet aforementioned throughput and latency requirements in presence of node/network failures or cyber attacks.
- *Heterogeneity*: Meet above three requirements under heterogeneous network conditions and device characteristics.

The performance and resilience evaluations of consensus protocols on the considered Blockchain-integrated network are mainly conducted at the node level. Modeling the system at the node level enables better representation of the specific protocols' reliance on number of nodes and how they operate with varying network topologies. Therefore, we first observe and record the key aspects from an ideal system that has minimum influence from external factors, which is discussed more in the following subsection II-B2. Then, we use the observed data to model various components of the simulator for performance evaluation. Figure 3 shows the collection and use of the data from both the ideal system and the simulated system. Finally, we fine tune the developed model and validate to provide a meaningful measure on its effectiveness to represent the ideal system.

2) Ideal System: An ideal system is a set up of the consensus protocols where the external engagements and influences are minimum. The purpose of the ideal system is to provide a platform that can be easily modeled which will provide results for the baseline scenarios. In this context, the ideal system has several nodes where every node may have either transient or permanent network connectivity with each other in a mesh topology. A good candidate for implementing this is to use a virtual network on a single machine using docker containers [13], [14]. The docker nodes in an ideal system act as validator nodes and run the consensus protocol. The number of networked nodes is kept variable to identify how performance metrics change as number of nodes scale. Logging is also enabled on the individual nodes to record performance aspects of the consensus protocol. In the next subsection, we discuss different kinds of gathered data and collection mechanism to better understand the state and performance of the system.

C. Data Collection

The ideal system provides the ground truth for the proposed simulator and will be run with each consensus protocol to be modeled. We collect various data points in order to represent the key performance metrics of the modeled system as discussed in subsection II-B1. The details of data points to be recorded are discussed below.

- *nMessage*: Number of exchanged messages to achieve consensus
- *nodeDelay*: Processing delay at node level required for achieving consensus.
- *nodeConsen*: Number of participating nodes required to achieve consensus
- *consenDelay*: Network delay for the consensus protocol to achieve consensus
- *consenPauseState*: Probability that the consensus state is paused for a special case
- *pauseStateTime*: Time that the consensus system remains in the paused state

D. Consensus Layer Modeling

To fully represent the consensus protocols effects on different numbers of participating nodes, limitations on network, computing resources, and network topology, several simulation runs with samples of the input data is needed. Hence, to complete these runs in a reasonable amount of time, discrete event simulation is used, where a networked message is considered as an event. The discrete event simulation engine schedules each message as the simulation runs based on how each modeled component affects the system. The component affects are modeled as statistical representations of observed data from the ideal system.

Each participating node runs a validation instance that models the actual workload of a peer in the consensus protocol. The validation process includes the necessary inputs required to represent the underlying consensus protocol steps. All of the input data for each state of the consensus protocol is stored in the form of distributed libraries or variables in the application. As the simulation runs, the participating peer either begins a new transaction or participate in the consensus of an existing transaction. The participating peer continues this operation until either a certain number of transactions have been processed or a period of time has elapsed.

Transaction processing starts when one of the current peers proposes a transaction to the system. It causes initialization of the transaction module, which then sends out messages to the participating peer nodes. The transacting node schedules the message to be delivered to every peer node that it can communicate with on the simulated network through utilizing the existing routing libraries. Based on the simulated arrival time of the message to each node, the simulation engine orders the events accordingly where the events are handled in the order they are scheduled in the event list.

As participating peer nodes receive messages from the consensus process, they internally record the number of messages processed during the current consensus iteration. If a threshold number messages have been processed by a peer node as determined by *nMessage*, then the node's state is assigned to "completed". Peer nodes with status "completed" send a "completed" message to the other participating nodes stating that it has completed its consensus round. If the peer node has not processed enough messages, it will schedule another process message with the simulation engine for a delay sampled value based on the *nodeDelay* distribution. The node continues to process messages until the appropriate number of messages have been processed and upon completion its state changes to "completed". The leader peer that started the transaction monitors the system until *nodeConsen* number of participating nodes reach a state of "completed". After enough participating nodes have reached a "completed" state, another random peer can start its transaction.

The general consensus system is modeled to change leaders periodically based on the consensus protocol it is modeling. If the system changes states, the transactions will be halted for period of time (*stateLength*). The next event would be scheduled with a new transaction and the start time is set after the delay *stateLength* completes.

III. Implementation

In this Section, we discuss the steps taken to implement the proposed Blockchain simulator in details. The simulation paradigm is considered to be discrete event simulation. The general consensus protocol described in the methodology is built on Network Simulator-3 (NS3).

A. Simulation Data Gathering

For this initial test, Raft [15] is chosen as the consensus protocol because of its simplicity and many available packages

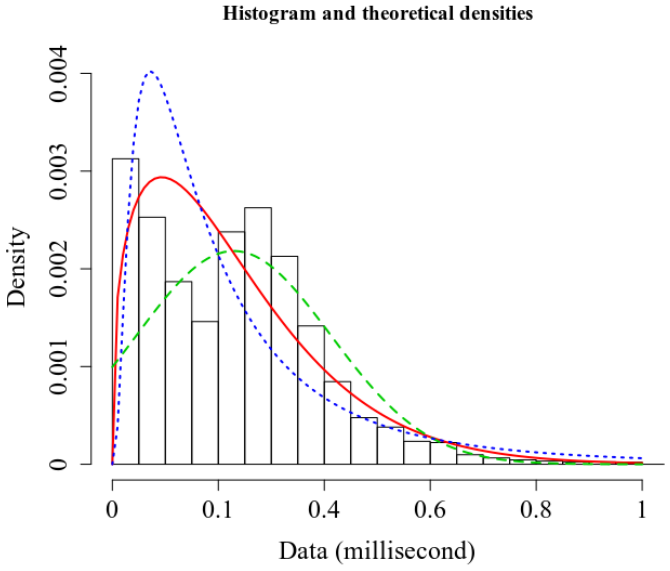

Fig. 4. Curve fitting node message process times in nanoseconds

that exist to execute just the consensus protocol without any additional features. We set up Raft to run on a single machine with three to five participating nodes. For each of these settings one of the nodes acts as a leader node throughout the simulation. The nodes run as docker containers on that machine and are connected through a virtual network to provide as ideal connectivity as possible so that the time to process each message does not get effected by network delays. As discussed in Section II-C, consensus data is collected for the Raft protocol to test the methodology. We record the time to find consensus for one thousand iterations and the time for each node to process a single message in each of the thousand iterations. We subsequently fit the gathered data to a distribution. Figure 4 shows the distribution of a node's processing data represented as a histogram and the result of the curve fitting (using R stats) for Weibull, Lognormal, and Normal distributions. It is observed that node's processing data for the Raft protocol follows the Weibull distribution as it best fits the ideal Raft consensus with a scale parameter of 1.35 and a shape parameter of 249857.5.

B. Network layer Implementation

The network simulator handles the flow messages among the participating peers. Researchers have implemented a proof-of-work consensus protocol with the Bitcoin-Simulator using NS3 [3]. It has the ability to generate large scale networks and communicate using various networking methods such as P2P and gossip-relay types of network. For this paper, we modified the main controller module, as well as the Bitcoin node and built a new application extending the Bitcoin node for the general consensus node.

The network layer is built to represent the ideal system that is used to gather data for the simulation. To best represent this, the simulator uses a mesh P2P network using the point-

to-point NS3 module. The links use data rate in the order of gigabits per second, and the link delay is set to zero with the intention of modifying this for calibration. Although the Bitcoin simulator is capable to generate more complex network topologies spanning multiple regions of the world, in this paper we consider that all nodes are connected through a mesh P2P network in the same region.

C. Consensus layer Implementation

As described in the methodology section, each participating node has certain variables and distributions to represent the behavior of the consensus protocol it is modeling. A leader node is specified at the beginning and will broadcast a message to the participating peers indicating the start of consensus. The message is sent using the NS3 socket class and the send ability. NS3 handles the message as a set of discrete events traversing the network. The peer nodes then proceed to process consensus messages as specified in the methodology section and consensus is found when *nodeCompleted* nodes have processed *nMessages*.

D. Calibration and Simulation Sensitivity

With the model developed and functioning based on the observed data from the participating nodes, the total system is calibrated to observe how close it performs with respect to the ideal system. In our proposed simulator, calibration is done by: 1) increasing network background traffic or adding delay to network to create more delay in the message processing, and 2) tuning the distributions using constants as a scaling factor for selected input datasets as needed. It is an iterative process and can use optimization methods to minimize the error between the model system and the observed data from the ideal system.

IV. Results and Discussion

In this section, we provide the results obtained both at network layer and consensus layer with respect to the time taken to find consensus in the system.

A. Network layer observation

The network delay is modeled using a lognormal distribution. The lognormal mean and variance are found to be good at 12.85 and 0.57 respectively. These values are fitted with an iterative process, running the simulation comparing results, then we use these results to adjust the parameters and feedback this change to the model. This process continues for several iterations until a close fit is found. The calibration is done for the ideal system, with three peers running the Raft implementation. The results of the calibrated model are shown in Fig. 5 with two scatter plots. The scatter plot on the left is calibration of the simulation to the ideal system. The x-axis represents time in seconds for a consensus transaction by the simulator. The y-axis represents time in seconds for a consensus transaction by the ideal system. The red straight line is the computed regression line of the two data sets. The left scatter plot shows that there is a tight grouping and positive

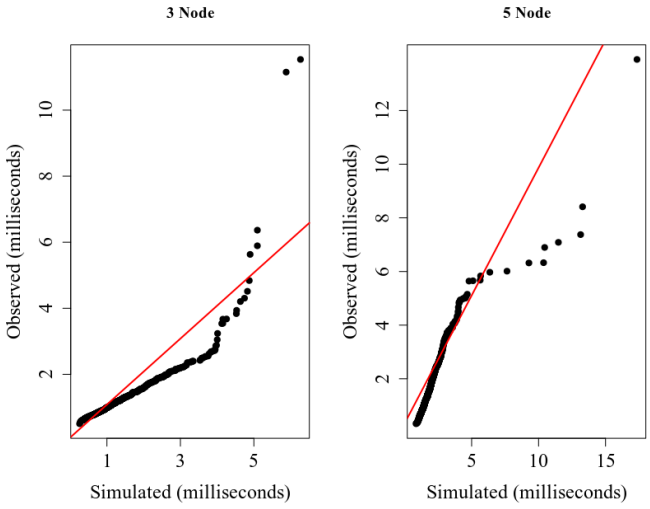

Fig. 5. Calibrating scatter plots of simulation consensus times versus observed consensus times in milliseconds

correlation with the simulation and the observed ideal system. The scatter plot also shows that for the few extreme values the two systems differ some. This is likely because the curve fitting is mostly matching the common results in the observed data. The distribution could be modified and calibration could be pursued further to better match the extremem values.

The scatter plot to the right in Fig 5 is the results of the calibrated model and simulation of the ideal system but run with 5-nodes instead of 3. The simulated results are compared with observed results for the same system. In this plot, the simulator is not calibrated for 5-peers, rather we wanted to observe if the behavior is good enough to represent the system's state using the calibration from the 3-peers scenario. This comparison in Fig 5 was done as a verification process to see if the behavior of the calibrated simulation is transferable to other scenarios.

Peers	Average (ms)	Standard Deviation	Minimum	Maximum
3	1318.48	27.87	-3.96%	4.81%
4	1505.0	26.44	-2.69%	5.72%
5	1651.89	30.73	-3.68%	3.30%
10	2168.09	40.53	-3.96%	4.06%
16	2575.94	45.39	-4.01%	4.24%

TABLE I
CONSENSUS TIMES (IN MS) FOR 1-MESSAGE SCENARIO

B. Consensus layer observation

In order to get insight into how much the simulation of the general consensus protocol changes and understand the effect of the number of peers and the number of messages on the time to find consensus consensus, we conduct sensitive analysis. In this analysis, we modify the number of peers and the number of messages required for the general consensus protocol, and observe their impact.

For this test, 30 simulation runs are conducted for each scenario to handle the stochastic nature of the sampled distributions used. The first scenario consisted of three required participating peers and the requirement for each one to process one message. The simulation processed 1000 consensus

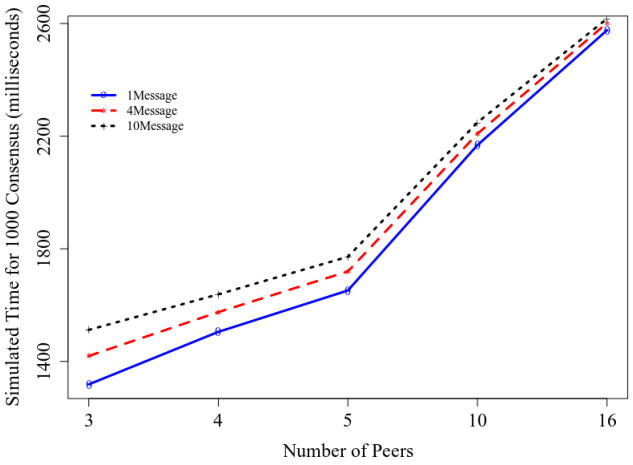

Fig. 6. Average time (in ms) to confirm 1000 transactions with different message requirements

transactions and then recorded the time it took to complete the transactions. Table. IV-A shows the results for the test while varying the number of peers. In this table, the average time to process 1000 consensus transactions is shown for each number of peers tested. We can observe that the time to reach consensus increases as the number of peers increases. The table also includes the standard deviation and the minimum and maximum values as a percentage of the average to provide information on the variance over 30 simulation runs. This gives an idea on the amount of variability the simulator is producing in its results for each number of peers used. Generally speaking the standard deviation is increases as the number of peers increases indicating more variance in the results. This is expected since the system becomes more complex with more participating peers.

In order to further observe the impact of the number of messages on the time to find consensus, we ran the above described simulation while increasing the number of messages. Fig 6 shows the same data for the 1 message test as shown in Table IV-A , but compared to similar tests with 4 message consensus and 10 message consensus. With this figure we observe increments in time to process 1000 consensus transactions between the different scenarios. Also, the time difference between different number of messages decreases when the number of peers increase. This finding provides some insight into the factors that cause the most delay. In this case it is shown that the number of peers has a greater effect on the delay of the system than the number of messages.

V. Conclusions and Future Work

We present a Blockchain simulator for testing and evaluating consensus algorithms in a realistic and configurable network layer. The unique characteristics of the simulator are a general consensus algorithm operating in a realistic and configurable network environment. The discrete simulation engine allows to specify the consensus algorithm operations at near real-time fidelity without loss of scalability. In future, we plan to implement and calibrate additional consensus algorithms to accommodate bootstrapping scenario and evaluate performance in diverse network configurations. Future work with this model and simulation will include the effects of the consensus special state using the *consenPauseState* variable, which will be modeled to get a better comparative test between different consensus protocols.

References

[1] 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 *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: ACM, 2017, pp. 1085–1100. [Online]. Available: http://doi.acm.org/10.1145/3035918.3064033

[2] H. Technologies, "Caliper: A blockchain benchmark framework," 2017, last accessed 29 April 2019. [Online]. Available: https://github.com/hyperledger-archives/caliper

[3] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM, 2016, pp. 3–16.

[4] S. Goswami, "Scalability analysis of blockchains through blockchain," 2017, last accessed 29 April 2019. [Online]. Available: https://digitalscholarship.unlv.edu/cgi/viewcontent.cgi?article=3979&context=thesesdissertations

[5] 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 *Blockchain – ICBC 2018*, S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham: Springer International Publishing, 2018, pp. 18–31.

[6] R. Yasaweerasinghelage, M. Staples, and I. Weber, "Predicting latency of blockchain-based systems using architectural modelling and simulation," in *2017 IEEE International Conference on Software Architecture (ICSA)*, April 2017, pp. 253–256.

[7] Q. T. Zhong and Z. Cole., "Analyzing the effects of network latency on blockchain performance and security using the whiteblock testing platform." 2018, last accessed 29 April 2019. [Online]. Available: https://www.whiteblock.io/library/analyzing-effects-network.pdf

[8] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18. New York, NY, USA: ACM, 2018, pp. 30:1–30:15. [Online]. Available: http://doi.acm.org/10.1145/3190508.3190538

[9] I. G. M. H. Richard Gendal Brown, James Carlyle, "Corda: An introduction," 2016, last accessed 29 April 2019. [Online]. Available: https://docs.corda.net/releases/release-M7.0/_static/corda-introductory-whitepaper.pdf

[10] E. Buchman, "Tendermint: Byzantine fault tolerance in the age of blockchains," 2016, last accessed 29 April 2019. [Online]. Available: http://atrium.lib.uoguelph.ca/xmlui/handle/10214/9769

[11] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, D. Song, and R. Wattenhofer, "On scaling decentralized blockchains," in *Financial Cryptography and Data Security*, J. Clark, S. Meiklejohn, P. Y. Ryan, D. Wallach, M. Brenner, and K. Rohloff, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 106–125.

[12] N. Segent, "Performance evaluation of a consensus algorithm with petri nets," in *Proceedings of the Seventh International Workshop on Petri Nets and Performance Models*. IEEE, 1997, pp. 143–152.

[13] "Docker documentation," Aug 2018. [Online]. Available: https://docs.docker.com/

[14] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, p. 30.

[15] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm (extended version)," 2013.