JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

BLOCKCHAIN SIMULATOR

B V Sai Karthik[1] and Dr. G. Lavanya Devi[2]

[1]PG Scholar in Andhra University College of Engg. (A), Vishakapatnam

[2]Asst. Professor in Andhra University College of Engg. (A), Vishakapatnam

karthikyadav373@gmail.com

lavanyadevig@yahoo.co.in

Abstract : Blockchain technology has becoming increasingly popular in both academia and business. Block-chains are distributed ledgers where a number of network users who do not fully trust one another agree and reach a consensus on the current state of the ledger as a whole. It is incredibly challenging but also incredibly rewarding to understand the frontiers and promise of this technology. A quicker development rate, meanwhile, might be constrained by the absence of tools for assessing design and execution options. To address the issue, this project provides a discrete-event simulator that is flexible enough to evaluate different blockchain implementations. Therefore, these blockchains may be rapidly described and simulated by extending current The simulator has been used to model the Bitcoin and Ethereum networks, and the results have been contrasted with data gathered from the real networks. Running a simulation model for Bitcoin and Ethereum allows you to change the environment, get answers to numerous questions, and perform a thorough investigation of the entire system. The method can be applied to any blockchain system.simulation models.

Index Term: - Blockchain, BITCOIN, Ethereum, Discrete-event, Real Networks

I Introduction

A distributed ledger, or blockchain, is a type of data structure that only allows appends and stores an ordered list of transactions across numerous nodes connected by the Internet. Blockchains typically make the assumption that these nodes, who do not fully trust one another, might behave in a Byzantine manner. They must also concur on a transactional order that can tolerate Byzantine failures at the same time. The fact that previous transactions cannot be modified but new transactions can be added to the blockchain ensures the integrity of transactions. The original blockchain of the Bitcoin cryptocurrency system served as its centre, enabling nodes to store and duplicate digital money as system state. These virtual currency units are transferable between locations. In addition to bitcoin systems, the blockchain idea. The original blockchain of the Bitcoin cryptocurrency system served as its centre, enabling nodes to store and duplicate digital money as system state. These virtual currency units are transferable between locations. The idea of blockchain has grown beyond bitcoin systems, and Ethereum has emerged as a blockchain that can describe more complex states, enabling Turing complete code to be executed within a transaction, or what is generally known as smart contracts. Because Bitcoin and Ethereum operate in a public setting where nodes can join and leave the network without permission, they are known as permissionless blockchains. Records and smart contracts' independence and decentralisation have the ability to revolutionise significant industrial sectors. New blockchains have been created particularly to fulfil the needs of such closed settings as a result of rising industry interest. The idea of "permissioned block chains" has recently been put up to authorise a select few players. The two most well-known are Tendermint and Hyperledger Fabric. Deterministic Byzantine Fault-Tolerant (BFT) consensus methods are utilised on these permissioned blockchains. There aren't enough tools to evaluate blockchain systems despite the increasing interest in developing them for specific use cases. Consideration must be given to emulation, a modern evaluation technique that imitates a system's behaviour.

II Literature Survey

Bitcoin is the first permissionless blockchain, allowing any participant to join and leave the network without permission.

The Bitcoin network is peer-to-peer (P2P), where all the

participants in the network are equally privileged and equally powerful. The participants share the burden of providing network services, without the need for a centralised service, and they are both suppliers and consumers of resources.These participants, or nodes, communicate with each other primarily via the Internet, using the Bitcoin protocol. Bitcoin can be viewed as a *state transition system*, where there is a *state* which declares the ownership status of all existing bitcoins, and a *state transition function*, that takes a state and a transaction and outputs a new state.

Nodes in the Bitcoin P2P network have equal privileges, but they can play four different roles: wallet, miner, full blockchain database, and network routing. The network routing role is shared by all nodes, and it entails propagating and validating transactions and blocks as well as finding and keeping connections with peers. A node needs to fulfil these duties in order to take part in the network and the consensus protocol. Full nodes, or some nodes with the full blockchain database role, keep an accurate and complete copy of the ledger and are capable of independently verifying any transaction without the aid of a third party. Last but not least, mining nodes are responsible for gathering and compiling all network transactions into a block. after confirming

A state transition function (STF) takes a state and a transaction as inputs and produces a new state as an output. A banking system can be used as an example, where the account balance represents the state and a statement indicating the transfer of $10 from Alice to Bob represents a transaction. The STF deducts 10 dollars from Alice's account and adds 10 dollars to Bob's account. The STF, however, produces an error if Alice's account balance is less than $10.

Unspent Transaction Outputs are a grouping of all digital currency in Bitcoin that have not yet been spent (UTXO). A bitcoin address, which is a general term for a cryptographic public key, is used to identify the owner of each UTXO.

There may be one or more inputs in a transaction. Each input makes use of a current UTXO and Normally the value of a UTXO is larger, or smaller, than the desired value of a transaction. Returning to the previous example, let us suppose that Alice wants to send 4.5 bitcoins to Bob. Alice needs to look for her UTXOs in the entire public ledger, which can be spent with the cryptographic keys controlled by her. Alice will not be able to send 4.5 bitcoins precisely. The smallest amount that she can send is 5 bitcoins (4 bitcoins UTXO plus 1 bitcoin UTXO). She then creates a transaction with two inputs and two outputs. The two inputs are the two UTXOs (4 bitcoins in a UTXO and 1 in another bitcoin UTXO). The first output is the exact amount that she wants to send (4.5 bitcoins) to Bob's associated address, while the second is the remaining 0.5 bitcoins, to be sent back to her. If Alice does not claim this change - by sending it to an address owned by her, the miner mining the block containing the transaction will be able to claim the change, as it will be considered a transaction fee. A transaction fee consists of the difference between inputs and outputs. Further explanations will be given about transaction fees.

In essence, a distributed ledger is an ordered list of transactions that is stored in a replicated, append-only data structure. For instance, a straightforward ledger can document financial exchanges between institutions or the exchange of goods between well-known parties. Every node in the network replicates the ledger in blockchains, and transactions are often put together into blocks. Blockchain can be pictured as a vertical stack of blocks, with the first block serving as the foundation, as shown in Figure 2.1. Each block has a unique hash that is produced on the block header by a cryptographic hash algorithm. These blocks are connected to the chain's first block via links (also known as parent block). Each block carries the hash of its own hash to establish that link.

III Implementation Study

3.1 Existing Methodology

General-purpose programming languages (like C++, Java, or Python) and specific-purpose simulation languages (like Arena and GPSS) are the two methods used to create simulation tools (Leemis and Park, 2006). The latter has various built-in capabilities (such as statistics, an event scheduler, and animation) that shorten the time needed to construct models, while the former is more flexible and well-known. According to Leemis and Park (2006), there is disagreement and dispute over which approach is best. The use of general-purpose languages to create simulation models is made possible by simulation frameworks, which are also important to note.

3.2 Proposed Methodology

BlockSim is a simulation framework that aids in the creation, application, and assessment of new and current blockchains. BlockSim allows extensive evaluation of certain assumptions on the simulation models without the burden of deployment and implementation of a real network, as well as quick and practical insights into how a specific blockchain system functions. A simulation model can be categorised based on a few characteristics. By incorporating probability distributions for specific events, the stochastic simulation model used by this simulator allows it to represent random phenomena. The implementation of random phenomena in terms of event probabilities, as described in Section 3.1, constitutes our proposed approach. Our models are regarded as dynamic since they may depict the system throughout a period of time that is selected by the user when they engage with BlockSim. Since an event-based system accumulates and modifies states

at discrete points in time, a discrete-event simulation (DES) model is appropriate for modelling a blockchain system. In this manner, just discrete points in time need be tracked to monitor the change of state variables, as opposed to continually across time (as in continuous simulation model). The simulator can therefore monitor thousands of nodes and events that only involve state changes.

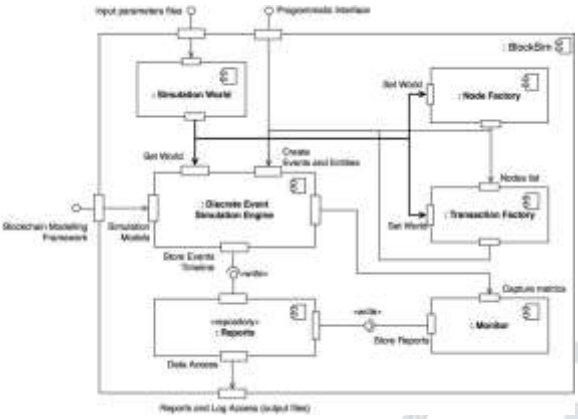

fig 2.1: Proposed Methdology Architure Diagram

3.3 Methodology and Alogrithms

3.3.1 Discrete Event Simulation Engine

Our Discrete Event Simulation Engine is implemented and run using the SimPy framework (DESE). SimPy is a Python-based framework for process-based discrete event simulation. SimPy processes can be used to simulate asynchronous networking or to construct multi-agent systems because they are built on Python generator functions. Functions can alternate execution by being specified to be exited and then later re-entered at the point of last exit by the programmer using generators. The Python yield keyword handles exiting and reentering.

3.3.2 Simulation World

The Simulation World component is responsible for managing the inputs of the simulator, which mainly are the probability distributions. Additionally, configurations to the simulator are also considered. These input parameters are needed in the simulation models, which are defined using the Blockchain Modeling Framework

3.3.3 Transaction and Node Factory

Batches of randomly generated transactions are produced by the transaction factory. The transaction factory will generate transactions in line with the transaction model depending on the simulated blockchain. Additionally, during simulation, the produced transactions will be disseminated.
is being executed by either a chosen node or a random node on a list. The user must also define the quantity of batches, the number of transactions per batch, and the time in seconds that must pass between each batch. The simulation's nodes are created and launched by the node factory. The node factory will generate nodes in accordance with the node.model depending on the blockchain being mimicked. The user can define the miner nodes' location, their number of miners and non-miners, and their hash rate range. after nodes

3.3.4 Monitor

A logging system in BlockSim records the time and origin node at which specific occurrences take place. The ability to observe each simulation step, the data that nodes are sending or receiving, and several other events is helpful for debugging. However, logs are not particularly helpful even when the simulation runs successfully because they do not contain statistically significant events, and tracing specific occurrences in log files can easily become cumbersome. The monitor's objective is to record any number of metrics during the simulation at any location in the models. Metrics should be simple for the user to update whenever necessary and should be automatically gathered and stored. We use this functionality to record the number of transactions added to the queue, broadcast, or received by each node.

3.3.5 Alogrithm

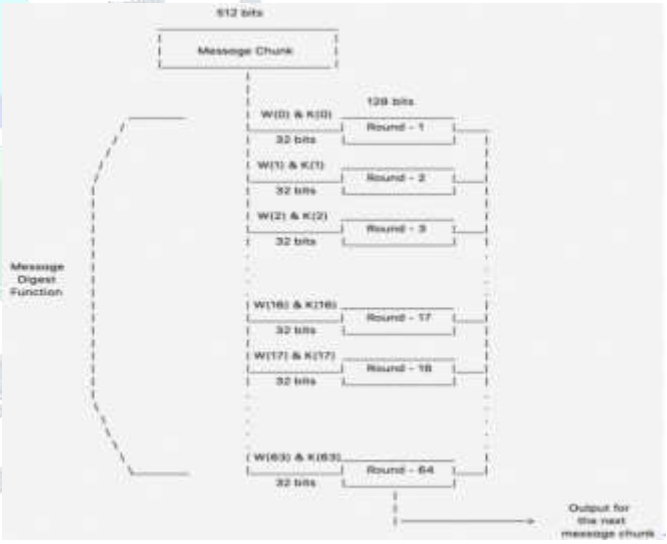

Fig 3.1: Block Diagram for SHA-256 Algorithm

IV Results and Evolution Metrics

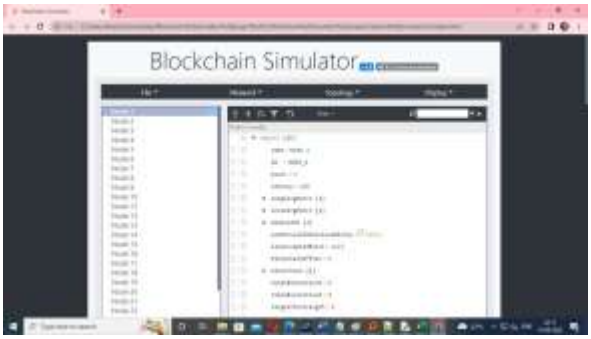

Fig 4.1**:** Block simulator home page

Fig 4.2: Miner network diagram which displays the chains between the nodes

Fig 4.3: The report when the block sim miner was update gives the complete report about the power consumption and account balance and no of peers connecte

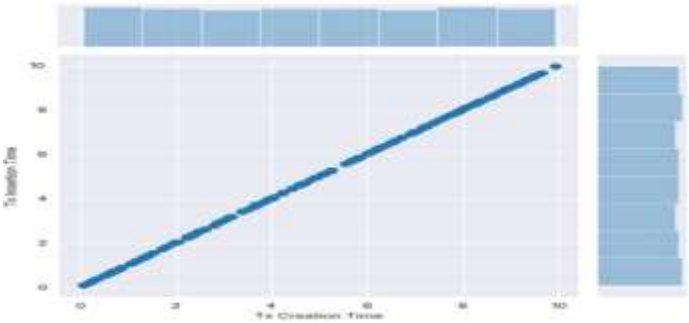

Fig 4.4: Node insertion time graph analysis

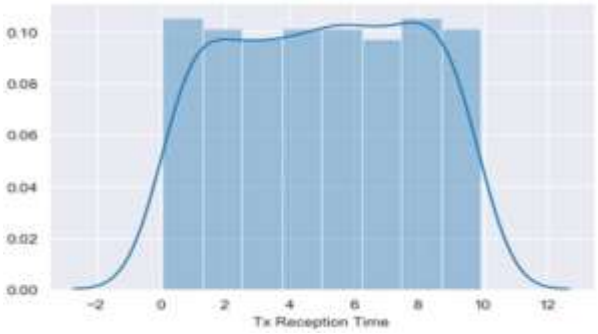

Fig 4.4: Node reception time graph

V Conclusion

Our efforts in developing a blockchain simulator have made it adaptable enough to easily reproduce different blockchain systems and add to or replace other models. use multiple models or consensus procedures. The fact that BlockSim can run thousands of nodes on a single host in a reasonable length of time makes it superior to competing technologies like emulation in addition to all of the other advantages mentioned above. We also showed how well the Ethereum system was modelled and how easy it was to change the model and simulation environment's parameters to look into novel use cases. Finally, we used BlockSim to explore a range of exciting genuine use cases, each of which made a substantial contribution to our comprehension of blockchain networks.

VI References

[1] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena,E. Shi, E. G. Sirer, *et al.*, "On scaling decentralised blockchains," in *International Confer- ence on Financial Cryptography and Data Security*, pp. 106–125, Springer, 2016.

[2] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzan- tine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 51–68, ACM, 2017.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[4] G. Wood, "Ethereum: A secure decentralisedgeneralised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.

[5] A. Nordrum, "Wall street occupies the blockchain-

financial firms plan to move trillions in assets to blockchains in 2018," *IEEE Spectrum*, vol. 54, no. 10, pp. 40–45, 2017.

[6] 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*, p. 30, ACM, 2018.

[7] J. Sousa, A.Bessani, and M. Vukoli´c, "A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform," *arXiv preprint arXiv:1709.06921*, 2017.

[8] E. Buchman, "Tendermint: Byzantine Fault Tolerance in the age of blockchains," Master's thesis, The University of Guelph, June 2016.

[9] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pp. 173– 186, USENIX Association, 1999.

[10] M. Correia, G. S. Veronese, N. F. Neves, and P. Ver´ıssimo, "Byzantine consensus in asyn- chronous message-passing systems: a survey," *International Journal of Critical Computer- Based Systems*, vol. 2, no. 2, pp. 141–161, 2011.