Modelling Blockchain Systems: A Network and Attack Strategy Simulator

Jemin Andrew Choi, Claudia Chen, and Mengliang (Clark) Cai

Abstract - Modern blockchain systems have evolved to provide crucial applications in economics and finance. Despite a growing demand for secure and trustless cryptocurrencies, there have been many successful attacks against large blockchain systems. This paper presents an agent-based blockchain simulator that simulates various attacking strategies against common cryptocurrencies such as Bitcoin. We modelled the characteristics of a peer-to-peer network and block difficulty parameters to observe the effects of a double spend attack and a time warp attack. Our results highlight the vulnerabilities in existing blockchain designs and affirm the importance of choosing an appropriate block confirmation and difficulty adjustment period.

I. Introduction

Blockchain is a time-stamped series of immutable data that is managed by a cluster of computers. Blockchain technologies have become the basis for many distributed digital currencies based on cryptographic elements to secure their operations [1, 2]. The core idea behind such digital currencies is to replace the centralized control of money transmission ordinarily taken up by large organizations, such as banks, by a large decentralized peer-to-peer network. However, potential problems exist in current blockchain technology, such as limited scalability, network delays, transaction costs, and security flaws. Particularly, security flaws in blockchain technologies have caused large financial losses and resulted in lower public confidence in cryptocurrencies. In order to prevent attackers from disrupting transactions, it is imperative to carefully design the system so there is protection against malicious nodes in the blockchain. It is equally imperative to have a testing infrastructure to simulate different attacking strategies to investigate how certain protocols will behave under attacks.

In this project, we design and implement a blockchain simulator system that simulates the behaviour of a network of nodes that generate and propagate blocks. The simulator models key features and various parameters of blockchain systems. This includes network conditions, such as network delay, bandwidth limit, and packet hops in a simulated peer-to-peer network. The system models the Nakamoto Consensus with a Proof of Work (PoW) protocol. Other features of the system include difficulty adjustment and a visualizer of the block history per simulation run. The resulting log and visualization provide insight on how the parameters set at the beginning of each run of the simulator may affect the results.

Nodes with special or different behaviours can be easily added to the simulator, and thus the system can be used to simulate various attacking strategies. We have chosen to focus on cases where a malicious party controls the majority or a significant portion of the total hashing power in the network. For this project, we have implemented two types of attacks - the double spend attack (51% attack) and the time warp attack. We investigate how changing different parameters of our simulation may require the attacking party to have more or less power by designating a controllable percentage of the total nodes to be attackers and defining them to behave in certain ways to emulate the attack.

The determination of block timestamps on a peer-to-peer network is also a tricky problem that may have vulnerabilities. The problem stems from a need to keep the nodes of the network in a relatively synchronized time frame in a decentralized way. With Bitcoin, a block timestamp is determined as being valid if it is greater than the median timestamp of previous 11 blocks, and less than the network-adjusted time plus 2 hours. Thus the system is possibly vulnerable to time hi-jacking attacks such as the ones conducted on Verge in 2018, during which the block timestamps were manipulated by attackers which caused a significant decrease in PoW problem difficulty [11]. Consequently, we will investigate how the factors that determine the timestamp and difficulty adjustment play a role in the vulnerability of the system to such an attack. These factors include the flexibility of the window of viable block timestamps (i.e. Bitcoin allows up to 2 hours difference) as well as how the timestamp of a block is determined.

In this paper, we outline the design of the blockchain simulator that we have implemented, the resulting behaviour of the system when it is run with different parameters and attacker strategies, and a comparison of the simulator with existing blockchain simulators.

II. Design

The blockchain simulator consists of 3 main classes: simulator, block and node. The system is written in Python and can be run on a single machine. When the simulator is created, it sets up the simulation according to predefined parameters that can be customized by the user, such as the number of nodes, connectivity of the network, and block generation rate. The simulator is in charge of creating a set of node objects, building the network, starting the nodes and controlling the difficulty of the PoW problem simulation. When the simulator starts, each node will attempt to mine blocks and when a block is mined, it will be propagated to their neighbours. The simulator then stops after a maximum total number of blocks have been mined.

Network

To simulate a blockchain network, we start by defining a node. In real blockchain systems, a node is a machine that is able to mine blocks and contribute to the Nakamoto consensus protocol. Nodes are also able to communicate with each other to verify and agree on a common block history.

Simulating nodes

To simulate a real node, the simulator initiates a thread with an agent that behaves according to a state machine. Each agent behaves independently of other agents and simulates the peer-to-peer networks commonly seen in blockchain networks. In our system, each node has its own thread and upon starting the thread, the node begins to mine for a block.

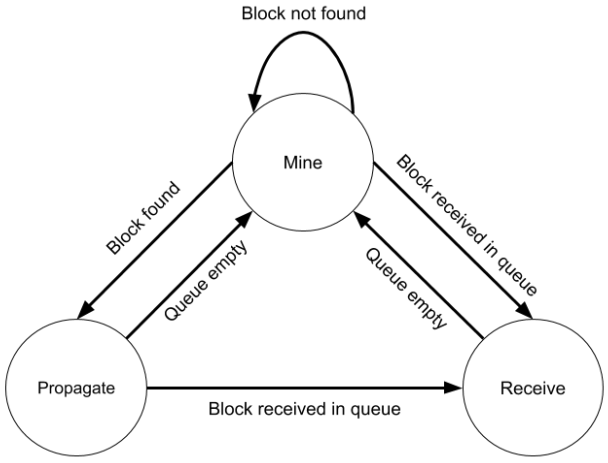

Figure 1: State diagram for a node agent.

Figure 1 shows the state machine diagram for a node. If the node is successful in mining a block, it moves to another state where it can begin to propagate the block to its neighbors. Else, if another node has found a block, it will propagate the block to its neighbors, and the neighbors will move to a state where it can update its longest chain by receiving new blocks.

Network Topology

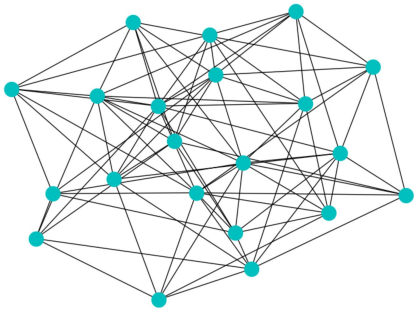

Figure 2: Visualization (generated by the simulator) of the random topology from the node network.

To create the node network, we first define how many peers each node will have. Given the number of neighbors per node, we shuffle the nodes and connect each node to a subset of others to create a random network topology for the simulation network. Figure 2 shows a visual example of the generated random topology from the simulator. The network generation can potentially be augmented with more organized and complex structures such as having chains of clusters of nodes, whereas each cluster simulates a region of nodes of close distance.

Unlike an actual blockchain network, the simulator builds the network for all nodes at once. Hence, to prevent any set of nodes from being isolated from the network, the simulator also adds an additional check to verify that the node graph is fully connected.

Network delay and bandwidth

The simulator also models the network delay and bandwidth between nodes. First, the simulator randomly assigns each node a specified bandwidth and a region. Nodes across farther regions experience higher network delay. To model bandwidth limitations in block transmission, we implemented a first-come-first-serve (FCFS) queueing system for each node. In order to propagate blocks through a gossip protocol, the sender must first add the block to its own queue, and then subsequently send the block to its neighbors' queues. If the block size exceeds the sender's bandwidth, the node experiences a delay in sending the full block. Contrary to network delay, which is equal for both senders and receivers, the bandwidth delay is capped based on the sender's and receiver's bandwidth.

One advantage of executing nodes on each thread is that the node's memory space is shared by all the nodes. This design makes it easy for senders to queue up blocks in the receiver's queue. To protect the critical sections from races, the simulator has global lock to ensure that a node's queue could not be modified by more than one node at once.

Consensus

For consensus, the system uses the Nakamoto longest chain consensus protocol with Proof-of-Work (PoW). Since actual PoW computations are very resource-extensive, we will simulate the PoW process, by letting each node have a probability proportional to the node's predetermined mining power. When the node attempts to mine, it has a chance to successfully mine a block by generating a random number. The block is accepted as a valid block mined if the random number is greater than the global difficulty of the blockchain network. The node's frequency of mining attempts is dictated by the node's mining power. Hence, nodes or pools with more mining power can do a greater number of calculations with more processing power and thus have a greater chance to solve the PoW problem.

Difficulty Adjustments

The simulator has a difficulty adjustment algorithm that changes the probability that a block is mined per mining attempt of each node. Based on the time interval between a set number of previous blocks and a target generation rate, the difficulty adjustment system will raise or lower the difficulty of PoW. Hence, as the difficulty falls, each miner will have higher probabilities of mining a block in the system. The difficulty is adjusted each time after a set number of blocks have been mined, which is called the difficulty adjustment interval. By adjusting the mining difficulty, we are able to stabilize the system's block generation rate at a relatively constant speed. Changes to the difficulty adjustment parameters can affect the amount of forking and various attacking strategies in the blockchain. As illustrated in Figure 3, the top system shows the blockchain without difficulty adjustment. The initial low difficulty results in many forks and orphans as the lower difficulty increases the chance that two nodes mine a block at about the same time. With difficulty adjustment turned on and set to a slower block generation rate, the bottom system of Figure 3 displays a decrease in forks as the algorithm increases the difficulty.

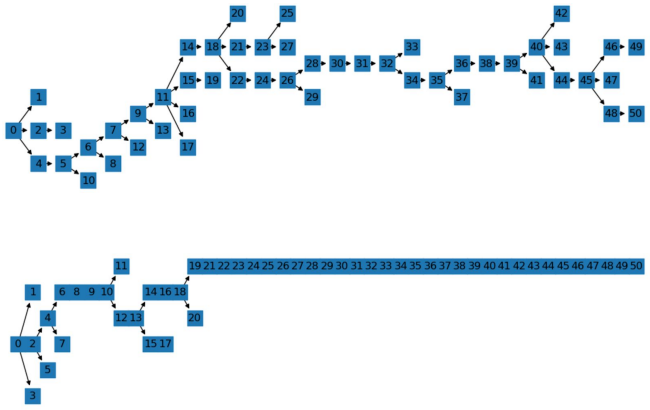

Figure 3: Visualization of the blockchain simulation without (top) and with (bottom) difficulty adjustment.

Attacks

To simulate various attacks, we introduced malicious nodes with different mining strategies. To simulate the double spend attack, we created a node that would wait for a specified number of blocks on the main chain before attempting to subvert the block. After waiting, the malicious node starts to mine blocks privately until it can overtake the longest chain developed by the honest nodes and broadcast the malicious blocks.

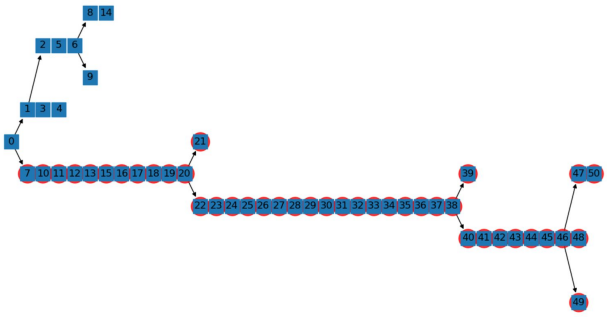

Figure 4: Visualization of a successful double spend attack. Red circles denote a block in the attacker's chain.

Figure 4 shows an example of a successful double spend attack. Although the honest nodes initially have a six block lead over the malicious node, the attack succeeds at subverting the main chain at block 15.

Similarly, to simulate the time warp attack, we introduced a node into the system that would artificially spoof the timestamps of generated blocks. By manipulating timestamps of blocks, the malicious node can fool the adjustment algorithm into thinking that the past difficulty adjustment interval was too harsh and generated too few blocks. Consequently, the adjustment algorithm temporarily lowers the mining difficulty. An attacker can complete the

time warp attack by taking advantage of the lowered difficulty and start mining nodes at an unreasonable pace.

Visualizer

As the simulation runs, the simulator builds a graph of all the mined blocks using a Python package called NetworkX. Every block becomes a node of the graph and its relationship with its preceding block becomes an edge. Thus we are able to capture the relationships between the longest chain, any branches and any blocks that are orphaned. This graph can then be visualized by the system. Matplotlib, a Python 2D plotting library, is used to output the visualization. Simulated attacks can be visualized as malicious blocks created by attacker nodes can be easily distinguished in the visualization.

III. Evaluation

Attacking Strategies

Double Spending Attack

To evaluate our simulator, we started by experimenting with the classical example of a double spending attack. For this experiment, we fixed the number of nodes to 10 to simulate larger mining pools as honest nodes. We ran the simulation for 100 blocks, and if the attacker could not subvert the main chain in this period, the attack was considered a failure. All the experiments were run on an EC2 m5.large machine.

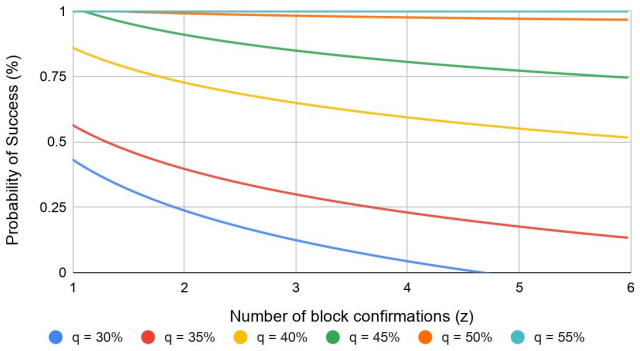

Figure 5: Graph showing attacker success rates based on number of blocks waited.

Figure 5 shows the results of our double spending experiment. The experiment also considered different attacking powers (q) and were plotted against the number of blocks waited (z) and the probability of subverting the main chain (q_z). The results correspond with existing literature and theoretical success rates for Bitcoin [1, 3, 6]. Although more block confirmations leads to lower chances of the attacker subverting the main chain, this also creates a tradeoff that transactions require longer wait times to go through. In addition, we predict that the simulation is optimistic compared to a real attack because of assumptions such as assuming large pools of honest nodes, which will result in fewer forks and in better defence against malicious nodes.

Time Warp Attack

In Bitcoin, difficulty adjustment is activated every 2016 blocks, and the adjustment is based on the time interval between the first and last blocks timestamp. There are mainly two standards for the block's timestamp in Bitcoin. The first rule is the timestamp in a block must not be more than 2 hours in the future. Another one is that the timestamp of a block cannot be greater than the median of the previous eleven blocks, which is called Median-Time-Past (MTP) [11]. In order to initiate a time warp attack, the attacker must create at least 6 out of 11 blocks with false timestamps to manipulate the median of the previous eleven blocks. If the above condition is achieved, the malicious attacker can modify the new block's timestamp to be equal or close to the median. Therefore, the system might deem the PoW difficulty is too high because the last adjustment interval has taken longer than expected. After repeating the above procedure multiple times, the attacker can fool the difficulty adjustment system and decrease the PoW difficulty significantly.

In our blockchain simulator, the time adjustment interval, total number of blocks and MTP interval parameter are relatively and proportionally small compared to real blockchain systems in order to reduce simulation time. In our experiment, the adjustment interval is reduced to 24 blocks and MTP interval to 5 blocks, with a maximum of 200 total blocks. In this case, we fixed the number of nodes to 10 to simulate larger mining pools as honest nodes, and one node as a malicious attacker.

Figure 6 shows a visualization output of a simulation run during which a time warp attack has failed, and it is clearly seen that the PoW difficulty remains in a stable region after a few time difficulty adjustment. In the case of a successful time warp attack as seen in Figure 7, PoW difficulty significantly drops between tenth and twelfth difficulty adjustment, which indicates that a malicious attacker has successfully modified timestamp and manipulates timestamp in that period. However, honest nodes overpower the malicious node for thirteenth round difficulty adjustment and PoW difficulty gradually resets back up.

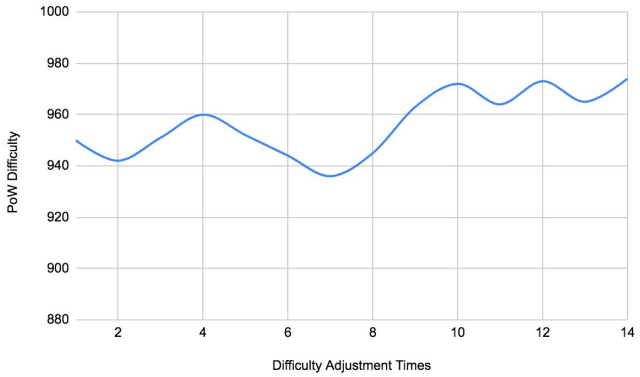

Figure 6: Graph of PoW difficulty over time when time warp attack fails.

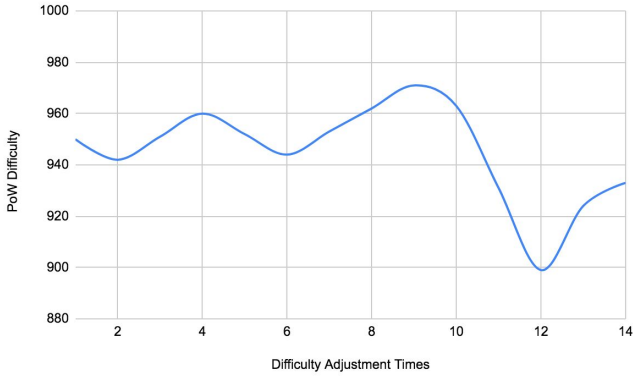

Figure 7: Graph of PoW difficulty over time when time warp attack succeeds.

We have also done an experiment to explore the relationship between the success rate for time warp attacks versus the adjustment interval in Figure 8. In the experiment, we fixed the number of nodes to 10, and one node as the malicious attacker with 51 percent hash power. The MTP interval was set at 5 blocks and generated 200 blocks. A successful time warp attack is defined as when the PoW difficulty in some difficulty adjustment interval is significantly decreasing due to incorrect timestamps.

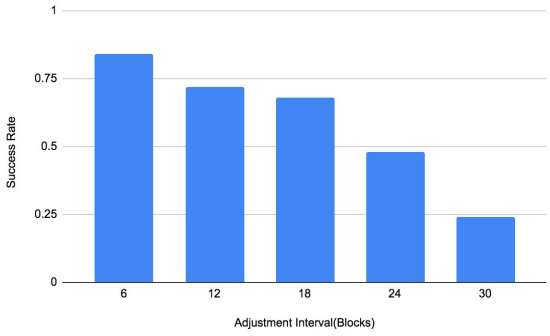

Figure 8: Time Warp Attack Success Rate vs. Adjustment Interval.

In the results shown on Figure 8, it is apparent that the time warp attack success rate decreases with increasing adjustment intervals. One possible reason for this trend is that the rule that a timestamp can be 2 hours in the future has more influence with smaller adjustment intervals, and the malicious attacker is more likely to manipulate an adjustment interval with fewer blocks. In other words, difficulty adjustment happens more frequently when adjustment intervals are smaller, and it results in the attacker having a higher probability to win the mining and propagation race. Hence, in the case of the 30 block adjustment interval, the success rate is greatly reduced. Although we did not run the simulation using very long block adjustment periods, such as 2016 blocks in Bitcoin, the results of the time warping attack indicate the vulnerabilities of blockchain systems with smaller difficulty adjustment periods. Hence, a blockchain that is more robust to time warp attacks should have higher difficulty adjustment intervals.

IV. Related Work

There have been numerous simulators that aim to simulate various facets and protocols of blockchain, including different network queueing theories [4], various mining strategies [6], event-driven simulators [5, 8, 9], and agent-driven simulators [7]. Unlike the majority of event-driven simulators, our simulator includes some degree of parallelization with the use of Python's threading libraries. However, it does not run on multiple CPU cores like a pure agent-driven simulator [7]. This approach of threading the nodes sacrifices some performance, but it allows the simulator to be run on fewer resources. Despite the questions drawn on the capabilities of Python's threading library, we also found that the simulator could support basic functionalities for up to one thousand nodes.

Although there is an abundance of event-driven simulators, there are few existing agent-driven simulators for blockchain systems, perhaps due to complicated synchronization issues and reliance on more and better hardware resources. In Rosa's LUNES agent-simulator, the authors evaluate the effectiveness of a DoS (sybil) attack on the bitcoin network [7]. Although our simulator is not as elaborate as LUNES, we have applied different attacking strategies, such as a double spend attack and a time warp attack, to test the blockchain network from different areas. In addition, our simulator can also output a visualization for varying attacking scenarios to show when or how the main chain was subverted. Future work includes extending and elaborating on attacking strategies, observing the effects of different consensus protocols on the security of the network, such as GHOST [10], and distributing the simulator load to improve performance of the system.

V. Conclusion

In this paper, we presented a blockchain simulator that models realistic network conditions and consensus mechanisms. To understand past vulnerabilities in existing blockchain systems, we simulated two attacks: 1) the double spend attack and 2) the time warp attack. The results of our simulator shows the importance of a sufficient block confirmation and difficulty adjustment period. In the future, we hope that our blockchain simulator can be used to simulate other attacking strategies and to educate others about proper blockchain design through a visual analysis tool.

VI. References

[1] Satoshi Nakamoto. [n. d.]. Bitcoin: A peer-to-peer electronic cash system. http://bitcoin.org/bitcoin.pdf.

[2] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper 151 (2014), 1–32.

[3] Meni Rosenfeld. 2012. Analysis of hashrate-based double-spending https://bitcoil.co.il/Doublespend.pdf.

[4] R. Memon, J. Li, and J. Ahmed. 2019. Simulation Model for Blockchain Systems Using Queuing Theory. Electronics, vol. 8, no. 2, p. 234.

[5] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo. 2019. SimBlock: A Blockchain Network Simulator. IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops. https://arxiv.org/pdf/1901.09777.pdf

[6] A. Gervais, G. O. Karame, K. Wust, V. Glykantzis, H. Ritzdorf, and S. Capkun. 2016. On the security and performance of proof of work blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS ’16, pages 3–16, New York, NY, USA. ACM.

[7] E. Rosa, G. D’Angelo, and S. Ferretti. 2019. Agent-Based Simulation of Blockchains. Communications in Computer and Information Science Methods and Applications for Modeling and Simulation of Complex Systems, pp. 115–126. https://arxiv.org/pdf/1908.11811.pdf

[8] M. Alharby and A. V. Moorsel. 2019. BlockSim: A Simulation Framework for Blockchain Systems. ACM SIGMETRICS Performance Evaluation Review, vol. 46, no. 3, pp. 135–138.

[9] C. Faria and M. Correia. 2019. BlockSim: Blockchain Simulator. IEEE Blockchain 2019.

[10] Y. Sompolinsky, and A. Zohar. 2015. Secure high-rate transaction processing in bitcoin. In International Conference on Financial Cryptography and Data Security, Springer, pp. 507– 527.

[11] Zachary. 2018. What Is a Time Warp Attack? https://bitcoinnews.com/what-is-a-timewarp-attack/