Indian Institute of Technology Bombay

CS765 Spring 2023 Semester, Project Part-1

Report On

Simulation of a P2P Cryptocurrency Network

Submitted by: Sayantan Biswas(23m0806)
Shamik Kumar De(22m0822)

Abstract

Cryptocurrency networks, built upon blockchain technology, have gained significant traction in recent years due to their decentralized and secure nature. Understanding the behavior and dynamics of these networks is crucial for researchers, developers, and stakeholders alike. Discrete-event simulation offers a powerful tool for studying complex systems, enabling the emulation of network behavior under various conditions.

This report presents the design and implementation details of a discrete-event simulator for a peer-to-peer (P2P) cryptocurrency network. The simulator aims to accurately model the behavior of a decentralized cryptocurrency network, encompassing essential features such as peer-to-peer communication, decentralized structure, transaction generation, network topology, latency simulation, proof of work (PoW) consensus mechanism, blockchain formation, and visualization.

Contents

List of Figures

1 Introduction

In this report, we present the design and implementation details of a discrete-event simulator tailored specifically for peer-to-peer (P2P) cryptocurrency networks. Leveraging the principles of discrete-event simulation, our framework offers a controlled environment to model the intricate workings of decentralized cryptocurrency networks. The primary objective of this project is to develop a comprehensive simulation framework that accurately models the behavior of P2P cryptocurrency networks, encompassing key features such as peer-to-peer communication, decentralized structure, transaction generation, network topology, latency simulation, proof of work (PoW) consensus mechanism, blockchain formation, and visualization.

By building this discrete-event simulator [1], we aim to provide researchers, developers, and stakeholders with a powerful tool to explore and analyze the dynamics of cryptocurrency networks in a controlled and reproducible environment. Through systematic design and rigorous implementation, our simulator facilitates in-depth investigations into network scalability, security, performance, and resilience. It allows for experimentation with varying parameters and conditions to gain insights into the behavior of decentralized networks over time.

Each section of the report focuses on a specific aspect of the simulator. The next section delves into the implementation details. Section 3 discusses transaction generation, network topology, and latency simulation, Section 4 explores PoW simulation, Section 5 covers fork resolution and blockchain tree maintenance, and Section 6 discusses visualization and experimental analysis. In Section 7 we conclude our work. Through rigorous experimentation and analysis, the report provides valuable insights into the behavior and performance of decentralized cryptocurrency networks, contributing to the understanding of this rapidly evolving field.

2 Implementation Details

The simulation is written in Python and utilizes various libraries and tools for its functionality. The main simulation logic [2] is contained within the Simulation class. This class manages the initialization of peers, the creation of a random network topology, the scheduling and handling of events, running the simulation, and visualization of the blockchain tree.

- **Network Topology:** The network topology is randomly generated using the NetworkX library [3], with each peer represented as a node in the graph. Peers establish connections with a random subset of other peers.
- **Peers:** Peers are instances of the Peer class. Each peer has attributes such as ID, speed, CPU power, and a blockchain instance. Peers generate transactions, and blocks, and handle incoming events like receiving transactions and blocks.
- **Events:** Events are scheduled and managed using a priority queue. Events include transaction generation, transaction and block propagation, block mining, and block reception.
- **Blockchain:** The blockchain is represented using the Block class. Each block contains transactions, a reference to the previous block, and other relevant metadata.
- **Visualization:** The NetworkX library is used to visualize the network topology and the blockchain tree for each peer. Visualization aids in understanding the structure and dynamics of the simulated network.
- **File Writing:** The simulation writes each peer's blockchain tree details into separate files for further analysis and inspection.
- **Command Line Interface (CLI):** The simulation accepts command-line arguments for specifying parameters such as the number of peers, percentages of slow and low CPU peers, mean transaction time, mean block generation time, and simulation duration.

  The simulation leverages Python libraries such as , **NumPy** for random number generation, **NetworkX** and **Matplotlib** for graph manipulation network and blockchain visualization. The simulation aims to model the behavior of a decentralized network, including transaction propagation, block generation, and consensus mechanisms, providing insights into the dynamics of such systems.

3 Transaction Generation and Network Topology

Transactions are generated by peers according to an exponential distribution for interarrival times. The network topology is constructed randomly, ensuring that each peer is connected to a suitable number of peers to maintain connectivity. A connected graph is ensured by regenerating the topology if necessary.

3.1 Implementation of Transaction Generation:

In our simulator, each peer generates transactions randomly over time. The inter-arrival time between transactions for any peer follows an exponential distribution, where the mean time T_{tx} can be adjusted as a parameter of the simulator. This ensures that transactions are generated in a probabilistic manner, mimicking the unpredictable nature of real-world transaction activity in cryptocurrency networks.

3.2 Theoretical Reasons for Choosing the Exponential Distribution

Several theoretical reasons support the suitability of exponential distribution for transaction generation in a cryptocurrency network:

- **Memoryless Property:**The exponential distribution possesses the memoryless property, which means that the time until the next event occurs is independent of the past. In a cryptocurrency network, previous transaction times do not influence each transaction's occurrence. This property aligns well with the decentralized and asynchronous nature of peer-to-peer networks.
- **Flexibility:** The exponential distribution allows for flexibility in modeling transaction generation. By adjusting the mean inter-arrival time parameter T_{tx}, we can control the average rate at which each peer generates transactions.
- **Statistical Elegance:** The exponential distribution is mathematically tractable and analytically elegant, making it convenient for simulation and analysis purposes. Its simple probability density function ($f(x) = \lambda e^{-\lambda x}$) facilitates calculations related to expected transaction rates, waiting times, and other relevant metrics.
- **Real-world Applicability:** Empirical studies have shown that transaction arrival patterns in decentralized networks often exhibit characteristics consistent with exponential distributions. While actual transaction behavior may deviate due to network dynamics and user interactions, the exponential distribution provides a reasonable approximation for simulation purposes.

Overall, the theoretical foundations and practical considerations make the exponential distribution a suitable choice for modeling transaction generation in our P2P cryptocurrency network simulator.

3.3 Implementation of Network Topology

In our simulator, we ensure that each peer is randomly connected to between 3 and 6 other peers, forming a dynamic network topology. Additionally, we perform checks to verify that the resulting network is a connected graph. If the generated network is not connected, we recreate the graph from scratch until a connected graph is achieved.

As part of our experimentation, we have created two distinct networks using our simulator: one consisting of 12 peers and another comprising 30 peers. These networks represent different scales of operation and allow us to evaluate the performance and behavior of the simulator under varying network sizes.

Figure 1: Network Topology with 12 peers

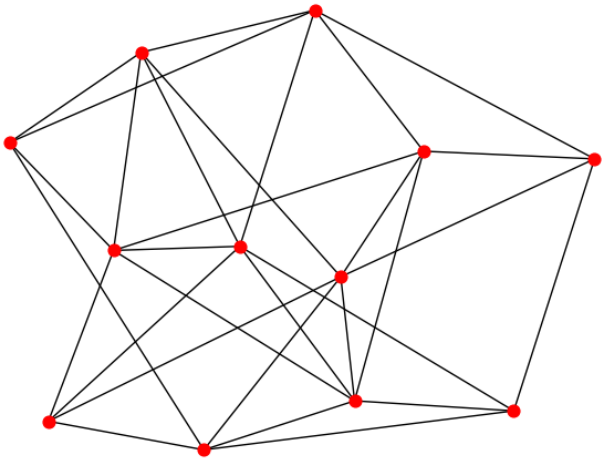
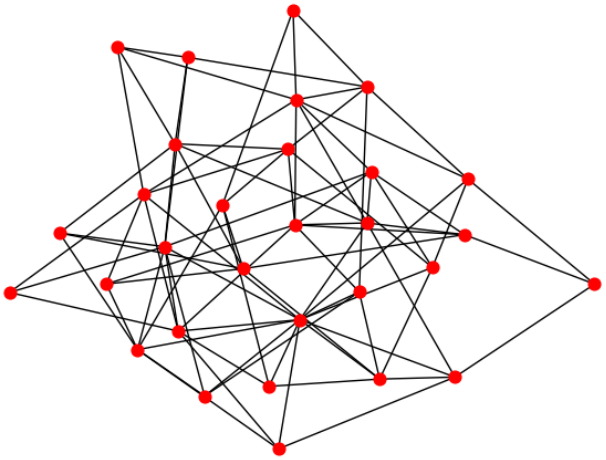


Figure 2: Network Topology with 30 peers



4 Simulating Latencies and Transaction Forwarding

Latencies between peers are simulated considering propagation delay, message length, link speed, and queuing delay. Transaction forwarding is implemented to prevent loops, ensuring that each transaction is propagated through the network efficiently.

4.1 Implementation of Latency Simulation

In our simulator, we accurately simulate latencies L_{ij} between pairs of peers i and j connected by a link. The latency is calculated using the formula $\rho_{ij} + \frac{|m|}{c_{ij}} + d_{ij}$, where: - ρ_{ij} represents the speed of light propagation delay, - $|m|$ denotes the length of the message in bits, - c_{ij} is the link speed between peers i and j, - d_{ij} is the queuing delay at peer i to forward the message to peer j.

Additionally, ρ_{ij} is chosen from a uniform distribution between 10ms and 500ms at the start of the simulation, and c_{ij} is set to 100 Mbps if both peers i and j are fast, and 5 Mbps if either of the nodes is slow. The queuing delay d_{ij} is randomly chosen for each message transmitted from i to j, following an exponential distribution with a mean of $96kbits/c_{ij}$.

4.2 Justification for Inverse Relationship between d_{ij} and c_{ij}

The link speed c_{ij} determines the maximum rate at which data can be transmitted between peers i and j. We set c_{ij} to a higher value (100 Mbps) for fast peers and a lower value (5 Mbps) for slow peers. This differentiation accounts for differences in network capabilities and ensures realistic latency simulation.

The queuing delay d_{ij} represents the time spent waiting in a transmission queue at peer i before forwarding the message to peer j. By sampling d_{ij} from an exponential distribution, we capture the stochastic nature of queuing delays, which can vary based on network congestion and other factors.

The mean queuing delay d_{ij} is inversely related to the link speed c_{ij} because as the link speed increases, the time spent waiting in the transmission queue decreases. This inverse relationship is intuitive because:

- Higher link speeds allow for faster data transmission, reducing the likelihood of congestion and queuing delays.
- Conversely, lower link speeds result in slower data transmission, increasing the likelihood of congestion and longer queuing delays.

By setting the mean queuing delay inversely proportional to the link speed, we ensure that the overall latency L_{ij} incorporates realistic queuing dynamics based on the network's transmission capacity. This helps to accurately simulate latency variations and network performance in our cryptocurrency network simulator.

4.3 Implementation of Loop-less Transaction Forwarding

In our simulator, each node adheres to the rule of forwarding transactions received from one peer to another connected peer. However, to prevent loops and ensure efficient transaction propagation, nodes follow specific criteria before forwarding a transaction:

- **Avoid Duplicate Forwarding:** A node checks whether it has already sent the same transaction to the destination peer. If the transaction has already been forwarded, the node refrains from re-sending it to avoid redundant transmission.
- **Prevent Circular Routing:** Additionally, a node verifies whether it received the transaction from the destination peer itself. If the transaction originated from the destination peer or was previously received, the node does not forward the transaction back to the same peer to prevent circular routing.

By implementing these rules, our simulator ensures loop-less transaction forwarding, promoting effective dissemination of transactions throughout the network without redundancy or circular paths.

5 Simulating Proof of Work

The PoW mechanism is simulated, wherein nodes validate received blocks and add them to their blockchain if valid. Block creation and mining follow a probabilistic model, with the mean interarrival time determined by the node's hashing power. Proper handling of block propagation and size constraints is ensured. The PoW process involves block creation, validation, mining, and propagation, as described below:

- **Genesis Block Initialization:** At the start of the simulation, all nodes possess the genesis block, ensuring a common starting point for the blockchain.
- **Unique Block Identification:** Each block is assigned a unique identifier (BlkID), ensuring no two blocks share the same identifier.
- **Block Structure:** A block contains:
  - Previous Block Identifier: Identifies the preceding block in the longest chain.
  - List of Transactions: Includes transactions not yet included in any blocks in the longest chain.
  - Coinbase Transaction: Rewards the mining node with 50 coins.

  -
- **Block Validation:** Upon receiving a block, a node validates all transactions to ensure that no peer's balance goes negative. Only valid transactions are considered for inclusion in the blockchain.
- **Mining Process:** When a node identifies a new longest chain, it initiates the mining process to create a new block. The node selects a subset of transactions from its pool, excluding those already included in the longest chain. The node calculates the mining time T_k based on its hashing power fraction h_k and the average interarrival time I between blocks. The mean of T_k is determined by I/h_k. The mining time T_k is drawn from an exponential distribution. If the node maintains the longest chain at $t_k + T_k$, it broadcasts the newly mined block to the network.
- **Block Propagation:** Upon broadcasting, the newly mined block propagates through the network, similar to individual transactions. The block size is limited to a maximum of 1 MB to ensure network efficiency and adherence to protocol specifications.

5.1 Justification for Mean Choice in Mining Time T_k

The choice of the mean for the mining time (T_k) is crucial for simulating realistic block creation rates and network dynamics within our simulator. By setting the mean

of T_k to I/h_k, where I represents the average block generation time, we ensure that nodes with higher hashing power mine blocks at a faster rate compared to nodes with lower hashing power. This approach mirrors real-world scenarios where nodes with higher computational resources (e.g., high CPU nodes) contribute more significantly to block creation and network security. In our simulator, a low value of I leads to higher forking while a high value of I leads to little to no forking.

Our implementation of PoW in the cryptocurrency network simulator accurately models the process of block creation, validation, mining, and propagation. By adhering to protocol specifications and simulating realistic network dynamics, we provide a comprehensive platform for studying blockchain behavior and evaluating the performance of decentralized systems.

6 Handling Forks and Maintaining Blockchain Trees

Fork resolution is implemented to identify the longest chain, ensuring network consensus. Each node maintains a tree of blockchains and records block arrival times for reference. This information is crucial for maintaining network integrity and ensuring the validity of transactions.

In our cryptocurrency network simulator, each node diligently maintains a tree structure containing all blockchains heard since the start of the simulation. Additionally, the node records the time of arrival for every block in its tree. At the end of the simulation, this information is appropriately written to a file for analysis and further processing.

6.1 Proper Resolution of Forks

Our simulator includes mechanisms to resolve forks by selecting the longest chain. Upon receiving conflicting blocks, nodes compare the lengths of competing chains and choose the longest one. This ensures network consensus and maintains the integrity of the blockchain by following the protocol's rules for chain selection.

7 Visualization and Experimental Analysis

Visualization tools are utilized to study the blockchain tree and analyze network behavior. **Matplotlib** creates the network topology of peers, representing the connections and interactions between nodes within the cryptocurrency network. On the other hand, **NetworkX** is utilized to create the blockchain tree of blocks, illustrating the hierarchical structure of the blockchain. This visualization allows for the analysis of block creation, propagation, and validation, as well as the identification of forks or conflicts within the blockchain.

7.1 Experimental Analysis of Simulation Parameters

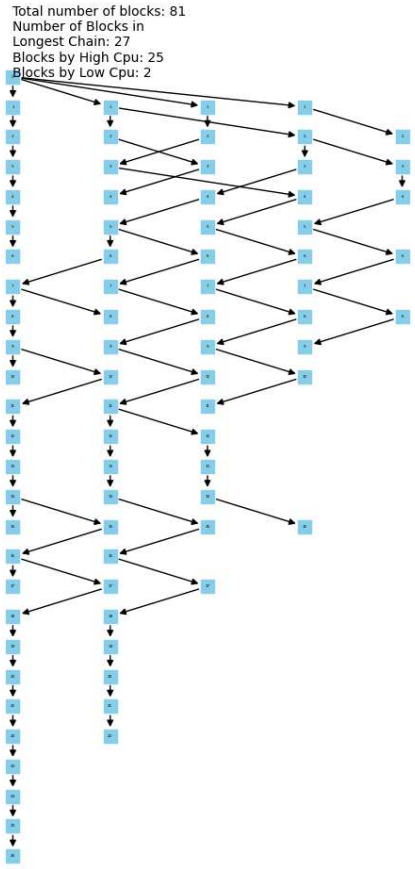
In our cryptocurrency network simulator, we conducted experiments by varying different parameters such as network size (n), percentage of slow nodes (z_0), percentage of nodes with low CPU power(z_1), transaction interarrival time (T_{tx}), mean block generation time(I), and simulation duration. We analyzed the ratio of blocks generated by each node in the Longest Chain of the tree to the total number of blocks it generates at the end of the simulation. Additionally, we examined the length of branches in the tree measured in the number of blocks.

7.1.1 Forking within the blockchain network

In our experimentation, we investigated the impact of varying parameters on the occurrence of forking within the blockchain network.

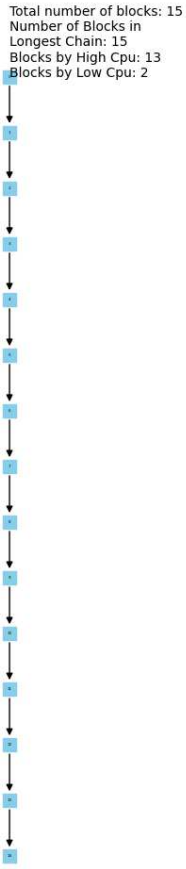
For the first set, with 30 peers, a slow percentage of 50%, low CPU of 50%, mean transaction generation time of 5 seconds, mean block generation time of 20 seconds, and a simulation duration of 3600 seconds, we observed a high level of forking.

Figure 3: Blockchain of a node with a high level of forking



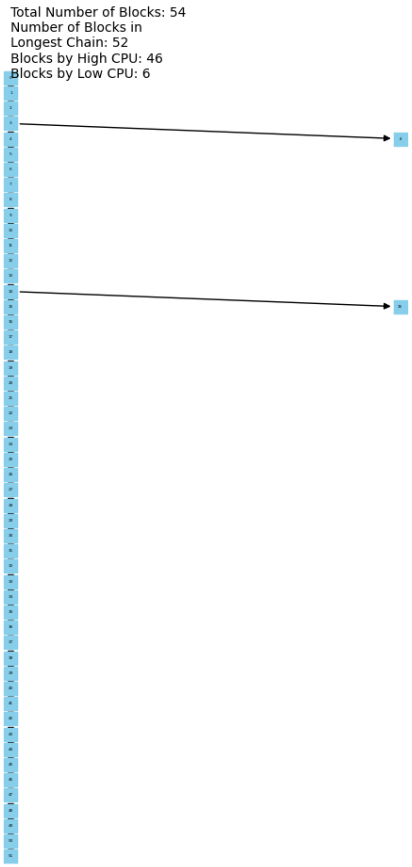
Conversely, with the second set of parameters, maintaining the same number of peers (30), slow percentage (50%), and low CPU (50%), but adjusting the mean block generation time to 140 seconds while keeping the mean transaction generation time at 5 seconds and the simulation duration at 5000 seconds, we noticed a significant reduction in forking instances.

Figure 4: Blockchain of a node with no forking



Thus we chose the value of I to be around 60 seconds for simulating real-world forking.

Figure 5: Blockchain of a node with moderate forking

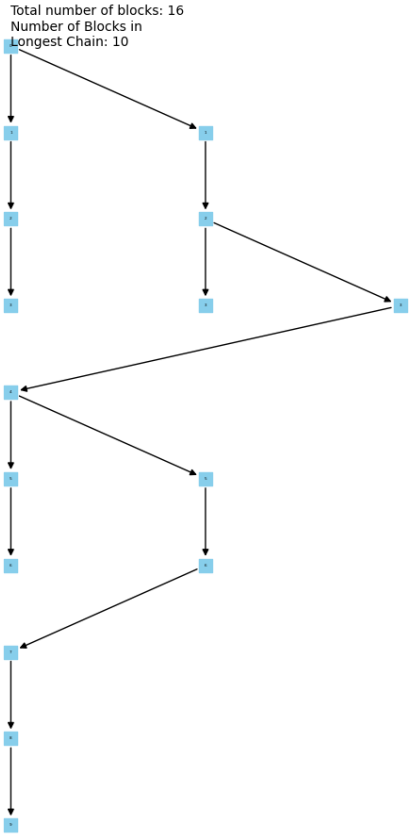


We also see that for all the cases, the number of blocks produced by nodes with high CPU power is much greater than those with low CPU power.

7.1.2 Chain Length and Blocks Produced

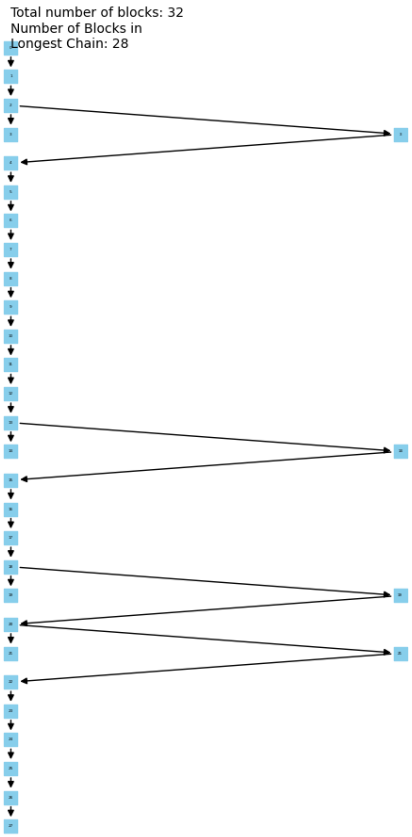
In the first parameter set, characterized by a high percentage of slow peers (90%) and low CPU peers (60%), a mean transaction generation time of 5 seconds, a mean block generation time of 60 seconds, and a simulation duration of 3300 seconds, the network experiences slower communication and processing capabilities. This delays block generation, propagation, and validation across the network. As a consequence, the competition for block mining may be less intense, leading to a shorter chain length and fewer blocks being added to the blockchain within the simulation duration. Additionally, the poorer network connectivity due to the prevalence of slow peers further exacerbates the challenges in block propagation, contributing to the observed outcome of a shorter longest chain and fewer total blocks.

Figure 6: Blockcahin of a node with shorter chain



Conversely, in the second parameter set with a lower percentage of slow peers (20%) and low CPU peers (40%), a mean transaction generation time of 5 seconds, a mean block generation time of 60 seconds, and a simulation duration of 3300 seconds the network exhibits improved communication and processing efficiency. Faster peers dominate the network, resulting in heightened competition for block mining. This increased competition, coupled with better network connectivity, facilitates faster propagation of blocks and validation, leading to a longer chain length and a greater number of blocks added to the blockchain within the simulation duration.

Figure 7: Blockcahin of a node with longer chain



8 Conclusion

In conclusion, this project provided a comprehensive discrete-event simulator designed to model blockchain networks. Each section of the report focused on a specific aspect of the simulator, providing in-depth insights into its implementation details and functionality.

In the first section, the project was introduced. The next section involved implementation details. Section 3 discussed transaction generation and defined the network topology. In Section 4 latency simulation and memoryless transaction forwarding were discussed Section 5 explored Proof of Work (PoW) simulation, outlining the logic behind block generation, mining, and validation. It delved into the intricacies of the PoW consensus mechanism and its role in securing the blockchain network. Section 6 covered fork resolution and blockchain tree maintenance, addressing the challenges associated with resolving forks in the blockchain and ensuring the integrity and consistency of the blockchain tree structure. Finally, Section 6 discussed visualization and experimental analysis, emphasizing the importance of visualizing the blockchain tree for better understanding and analysis of the simulated network dynamics. It provided insights into experimental results and performance analysis conducted using the simulator.

By examining these sections, we gained a comprehensive understanding of the simulator's capabilities and functionalities, contributing to our knowledge of decentralized systems and blockchain networks.

References

[1] "Chapter 3 - Event Driven Simulation",
Available online: http://cs.baylor.edu/~maurer/aida/desauto/chapter3.pdf

[2] H. Conrad Cunningham. "An Introduction to Discrete-Event Simulation",
Available online: https://www.cs.cmu.edu/~music/cmsip/readings/intro-discrete-event-sim.html

[3] Chatgpt,
Available online: https://chat.openai.com/