CS554 Project #2

Blockchain Simulation

Instructions:

- *Assigned date: Saturday June 14th, 2025*
- *Due date: 11:59PM on Friday June 27th, 2025*
- *Maximum Points: 150; +50 for Extra Credit Points*
- *This assignment must be done individually*
- *Please post questions to professor by email*
- *Only a softcopy submission is required through Canvas*
- *Late submission will be penalized at 20% per day*

1 Your Assignment

In this assignment, you will build a discrete-event simulation of a simplified blockchain network using the Python SimPy framework (pip3 install simpy). You will model nodes, transaction pools, miners, difficulty adjustment, and coin issuance. By completing this exercise, you will practice:

- Designing event-driven simulations with SimPy processes and events
- Managing global state (network metrics, transaction pool, coin balances)
- Implementing Poisson/exponential arrival processes (mining & wallet txs)
- Writing modular, CLI-driven Python programs
- Applying difficulty-retarget logic and halving schedules
- Verifying correctness via summary statistics (block time, TPS, coin supply)

2 Assignment Requirements

Write a Python script named sim-blockchain.py that implements the following features. Command-line arguments must match the descriptions below.

2.1 Network & Blocks

1. **Nodes** (--nodes N): Create N peer nodes, each maintaining a set of stored block IDs. Randomly connect each node to --neighbors M distinct peers.
2. **Block structure**: Each block has a header (1,024 bytes) + (# transactions × 256 bytes). Track block ID, timestamp, time-since-last-block, transaction count, and size.
3. **Block propagation**: When a node stores a new block, it broadcasts it to neighbors (increment a global io_requests counter per send; add block size to network_data). Ignore duplicates.

2.2 Mining & Difficulty

1. **Miners** (--miners K, --hashrate H): Spawn K miner processes, each with hashrate H. Expected time per block ~ Exp(total_hashrate / difficulty).

2. **Difficulty** (--blocktime T, --difficulty D optional): If D not set, initialize to T × (K × H). After every 2,016 blocks, retarget:

   new_diff = old_diff × (target_blocktime / actual_avg_blocktime)

3. **Halving & coin issuance** (--reward R, default 50): Issue R coins per block. Every --halving H blocks (default 210000), halve the reward; after 35 halvings, reward → 0.

2.3 Transactions & Wallets

1. **Wallets** (--wallets W, --transactions X, --interval I): Generate W wallet processes; each sends X transactions into a global unconfirmed-pool at interval I seconds.
2. **Block filling**: On block creation, include up to --blocksize B transactions from the pool (FIFO), plus the mining reward transaction.
3. **Termination**:
   - If --blocks L is specified, run until **L** blocks have been mined or until all txs are processed (whichever comes first).
   - If --blocks omitted, run until all wallet transactions have been confirmed.

2.4 Reporting & CLI Options

- --print P (default 144): Print a summary every P blocks; with --debug, print every block. Summaries must include:

  [time] Sum B:blocks/totalBlocks complete% abt:avg_block_time(s) tps:confirmed_tx_per_sec infl:inflation% ETA:seconds Diff:xx Hash:xx Tx:total_tx C:coins Pool:pending_tx NMB:network_MB IO:io_requests

- Final summary:
  [******] End B:blocks abt:avg_block_time(s) tps:confirmed_tx_per_sec Tx:total_tx C:coins NMB:network_MB IO:io_requests

2.5 Workloads to evaluate

Run BTC, BCH, LTC, DOGE, and MEMO for 10 years with no user transactions, and report the number of coins created and simulation time taken. See the details for each blockchain in the table below. Some of these simulations might take a lot of time, potentially tens of minutes for each simulation, depending on how you implemented it. Since your simulation is not multi-threaded, you can run multiple simulations at the same time for different configurations as long as you have a dedicated CPU core per simulation process. Then run BTC, BCH, LTC, DOGE, and MEMO with a small, medium, and large workload of transactions.

- SMALL: 10 wallets with 10 transactions each generated with interval of 10.0
- MEDIUM: 1000 wallets with 1000 transactions each generated with interval of 1.0
- LARGE: 1000 wallets with 1000 transactions each generated with interval of 0.01

Chain	Block Reward	Halving Schedule	Block Time	Block Size Limit	Max TX per Block
Bitcoin (BTC)	50 BTC	210K blocks	600 sec	1 MB base	4K TX
Bitcoin Cash (BCH)	12.5 BCH	210K blocks	600 sec	32 MB	128K TX
Litecoin (LTC)	50 LTC	840K blocks	150 sec	1 MB	4K TX
Dogecoin (DOGE)	10 000 DOGE (static)	None	60 sec	1 MB	4K TX
MEMO	51.8457072 MEMO	9644K blocks	3.27 sec	8 MB	32K TX

2.5 Extra Credit

- Simulate network latency by adding delays on receive broadcasts; ensure that block propagation happens within the blocktime
- Simulate network bandwidth by taking into account block size; simulating larger block sizes should take longer to propagate
- Track per-wallet balances and include fee logic
- Allow simulator to load traces from real world, such as transactions, miners joining and leaving to mimic global hashrate, etc

3 What you will submit

You should hand in:

1. **Source code:** All of the source code, including proper documentation and formatting.
2. **Readme:** A detailed manual describing the structure of your files and directory organization. The manual should be able to instruct users how to run the program step by step. The manual should contain example commands. This should be included as readme.txt in the source code folder.
3. **Report:** A written document (typed, named report.pdf) describing the overall assignment completion, along with screen shots of sample output. Make sure to include your results for various runs in a table.

You will put everything outlined above in a folder called “proj2-lastname” and compress it with ZIP or TAR utilities.

4 Where you will submit

You will have to submit your final compressed archive through Canvas.
Grades for late submissions will be lowered 20% per day late.