CS765 Spring 2023 Semester, Project Part-1

Simulation of a P2P Cryptocurrency Network

Due Date: 23:59hrs, February 8th, 2023

Note: Marks will be awarded for the points mentioned within the text. The marks have been indicated in square brackets (e.g. [1]).

In this assignment you will build your own *discrete-event* simulator for a P2P cryptocurrency network. This assignment can be done in groups consisting of **at most 3** persons. A discrete-event simulator maintains an ”event-queue” from which the earliest event is executed. This event may create further future events which get added to the queue. For example, an event in which one node ”sends a block” to connected peers will create future events of ”receive block” at its peers. Reading material for discrete-event simulators is given at the end of this document.

You can use any programming language of your choice. If you include code from a publicly available source, then state the source in the comments of the code. **Not more than 15% of the code should be taken from such sources.**

The cryptocurrency network must have the following properties (note that all IDs mentioned below must be unique).

1. There are n peers, each with a unique ID, where n is set at the time of initiation of the network. Some of these nodes (say z_0 percent, where z_0 is a command line simulation parameter) are labeled “slow” and the others “fast”. In addition, some of these nodes (say z_1 percent, where z_1 is a command line simulation parameter) are labeled “low CPU” and the others “high CPU”. We will use this classification below.

2. Each peer generates transactions randomly in time. The interarrival between transactions generated by any peer is chosen from an exponential distribution whose mean time(T_{tx}) can be set as a parameter of the simulator.
Marks:
Correct implementation of transaction generation. [4]
(Mention in report) What are the theoretical reasons of choosing the exponential distribution? [2]

3. Each transaction has the format: “`TxnID`: ID_x pays ID_y C coins”. You must ensure that C is less than or equal to the coins currently owned by ID_x (ID of the peer generating the transaction) before including it in a block. ID_y should be the ID of any other peer in the network where transaction is destined. The size of each transaction is assumed to be **1 KB**.

4. (Network topology) Each peer is randomly connected to between 4 and 8 other peers. Check that the resulting network is a connected graph, and recreate the graph from scratch if it is not a connected graph. **Marks:** Correct implementation. [4]

5. Simulate latencies L_{ij} between pairs of peers i and j connected by a link. Latency is the time between which a message m was transmitted from sender i and received by another node j. Choose the latency to be of the form $\rho_{ij} + |m|/c_{ij} + d_{ij}$, where ρ_{ij} is a positive minimum value corresponding to speed of light propagation delay, $|m|$ denotes the length of the message in bits, c_{ij} is the link speed between i and j in bits per second, and d_{ij} is the queuing delay at node i to forward the message to node j. d_{ij} is randomly chosen from an exponential distribution with some mean $96kbits/c_{i,j}$. Note that $d_{i,j}$ must be randomly chosen for *each message* transmitted from i to j. ρ_{ij} can be chosen from a uniform distribution between 10ms and 500ms at the start of the simulation. c_{ij} is set to 100 Mbps if both i and j are fast, and 5 Mbps if either of the nodes is slow.
Marks: Correct implementation of ρ, c, d. [12]
(Mention in report) Why is the mean of d_{ij} inversely related to c_{ij}? Give justification for this choice. [2]

6. A node forwards any transaction heard from one peer to another connected peer, provided it has not already sent the same transaction to that peer, or provided it did not hear (receive) the transaction from that peer.
Marks: Correct loop-less transaction forwarding. [4]

7. Simulating PoW: All nodes have the genesis block at the start of the simulation. Each block must have a unique ID, say `BlkID` (You can choose any method to ensure this). Any peer, say peer k, maintains a tree of blocks as in bitcoin. When it receives a block from another peer, it validates all its transactions (no balance of any peer should go negative), and if the block is valid it adds the block to its tree.

When a received block creates a new longest chain at peer k (longer than the previous longest chain), say at time t_k, then we simulate PoW mining of a new block as follows. Peer k forms a block at t_k by selecting a subset of the transactions received so far and not included in any blocks in the longest chain. After forming a block, node k generates a random variable T_k, at time t_k, as follows.

Suppose the interarrival time between blocks on average is I (for example, I is 600 sec in Bitcoin), and node k has fraction h_k (where $0 < h_k < 1$) of the total hashing power. Then T_k is drawn from an exponential distribution with mean equal to I/h_k. If peer k is a high CPU node, then it is assumed to have 10 times higher hashing power than a low CPU node. (Note: ensure that the $\sum_k h_k = 1$ in the simulation).

If peer k has the same longest chain at time $t_k + T_k$ then it broadcasts a new block. (If it no longer has the same longest chain at $t_k + T_k$ then the event scheduled for mining a block at $t_k + T_k$ is terminated.) This block lists the `BlkID` of the previous block in the longest chain it has received so far and the list of transactions included in it. Once broadcast, the node starts creating the new block. The block is assumed to contribute 50 coins to k (i.e. ID_k) as a mining fee, and this is included as a coinbase transaction "TxnID:ID_k mines 50 coins". On the contrary, if node k receives the block from another node in the network, it validates all its transactions. Valid transactions are those where the sender has sufficient balance to perform the transaction. After validation, the node again starts block creation, as explained above.

The block propagates in the network just like individual transactions. A block can have size at max **1 MB** (8×10^6 bits) and actual size depends on the number of transactions encapsulated in the block. An empty block (with no transactions besides the coinbase) is assumed to be of size $1KB$. Ensure that you do not have too many transactions in the block such that the size of the block exceeds the maximum permitted size.
Marks: Correct implementation of block structure, block creation, block validation, mining, block propagation and T_k, with explanation for the choice of a particular mean. [20]
Proper resolution of forks, to find the longest chain. [8]

8. Each node maintains a tree of all blockchains heard since the start of the simulation. The node stores the time of arrival of every block in its tree. This information is written to a file at the end of the simulation.
Marks: Properly maintained tree file for *each* node. [10]

Use an appropriate visualization tool to study the blockchain tree (suitable choices can be gnuplot, matlab, or any other visualization tool). Experiment with choosing different values for different parameters (n, z, T_{tx}, $T_{d_{ij}}$, mean of T_k, etc.). Find the ratio of the number 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. How does this ratio vary depending on whether the node is fast, slow, low CPU, high CPU power etc.? How long are branches of the tree measured in number of blocks? Give *detailed insights* to explain your observations.
Marks:
Proper study with a particular visualization tool using different parameters. [12]
Insight and critique of the observed values. [8]

In your submission on Moodle, submit a single zip file (filename format: RollNo1_RollNo2_RollNo3.zip) containing:
1. Source code for simulator. You need not submit any code for the visualization tool.
2. README file with instructions for compiling and running.
3. Design document giving modular flow of code. Example `https://www.slideshare.net/VasishtaBhargava/discrete-event-system-simulation-control-flow-chart`
4. A report detailing your findings along with pictures of typical blockchain trees and appropriate insight.
Marks: Proper commenting of code [4], Design document [8], README file [2]

Useful links:
`https://people.orie.cornell.edu/mru8/orie3120/lec/lec10.pdf`
`http://cs.baylor.edu/~maurer/aida/desauto/chapter3.pdf`
`https://www.cs.cmu.edu/~music/cmsip/readings/intro-discrete-event-sim.html`