COL867: Programming Assignment 1

Simulation of P2P Cryptocurrency network

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. The cryptocurrency network must have the following properties.

1. There are “n” peers, where “n” is set at the time of initiation of the network. Some of these nodes (say z% set at the start of the simulation) are labeled “slow” and the others “fast”. We will use this classification below.
2. Each peer has a unique ID.
3. Each peer generates transactions randomly in time. The interarrival between transactions generated by any peer is chosen from an exponential distribution whose mean time can be set as a parameter of the simulator.
4. Each transaction has the format: “UniqueTxnID: ID_x pays ID_y C coins”. You must ensure that C is less than the coins currently owned by IDx (ID of the peer generating the transaction). IDy should be the ID of any other peer in the network.
5. Each peer is connected to a random number of other peers.
6. Simulate latencies L_ij between pairs of peers “i” and “j”. 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 prop_ij + (size_m)/c_ij + d_ij, where prop_ij is a positive minimum value corresponding to speed of light propagation delay, c_ij is the bottleneck link speed between “i” and “j” and d_ij is the queuing delay on the path randomly chosen from an exponential distribution with some mean. For a message consisting of only 1 transaction, assume that size_m=0, but for a message consisting of a block, assume that size_m=1 MB. prop_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. The mean of d_ij is set equal to 12kB/c_ij.
7. A node forwards any transaction heard from one peer to another connected peer, provided it has not already heard the same transaction from that peer, or provided it did not hear (receive) that transaction from that peer.
8. Peer “k” generates a random variable T_k on hearing a block at time t_k. Each block has a unique ID, say UniqueBlkID. All nodes have the genesis block. T_k is chosen from an exponential distribution with some mean (expected value) which can be set as a simulation parameter. Note that smaller mean value for T_k is equivalent to saying “k” has more CPU power in a proof-of-work system. You can choose the distribution for T_k yourself. If peer “k” has not heard another block before time t_k + T_k then it broadcasts a block which lists the UniqueBlkID of the previous block in the longest chain it has received so far (use time to break ties). The block is assumed to contribute 50 coins to “k” (i.e. ID_k) as mining fee and lists all unspent transactions heard by “k” until t_k + T_k.

The block propagates in the network just like individual transactions. Unspent transactions are all transactions heard but not included in the longest chain heard so far.

9. 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.

Use an appropriate visualisation tool to study the trees of different nodes. Experiment with choosing different values for different parameters (n, z, mean of transaction interarrival etc.) Summarize the structure of the tree for different types of nodes (fast, slow, low CPU, high CPU power etc.). How long are branches of the tree? Give insight to explain your observations.

In your submission on moodle, submit a single zip file 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. A report summarizing your findings along with pictures of typical trees and appropriate insight.