COL867 Semester II 2017-18: HW1

Simulation of a P2P Cryptocurrency Network

January 2018

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. 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 percent set at the start of the simulation) are labeled “slow” and the others “fast”. 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 can be set as a parameter of the simulator.
Marks:
Correct implementation of transaction generation. [1]
(Mention in report) What are the theoretical reasons of choosing the exponential distribution? [1]

3. Each transaction has the format: “`TxnID`: ID_x pays ID_y C coins”. You must ensure that C is less than the coins currently owned by ID_x (ID of the peer generating the transaction). ID_y should be the ID of any other peer in the network.

4. Each peer is connected to a random number of other peers.
Marks: Correct random sampling along with justification of chosen distribution. [1]

5. 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 $\rho_{ij} + |m|/c_{ij} + d_{ij}$, 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 bottleneck link speed between i and j in bits per second, 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 $|m| = 0$, but for a message consisting of a block, assume that $|m| = 1$ MB (8×10^6 bits). 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. The mean of d_{ij} is set equal to 96kbit/c_{ij}.
Marks:
Correct implementation of ρ, c, d. [3]
(Mention in report) Why is the bottleneck link speed considered? [1]
(Mention in report) Why does d_{ij} mean depend on c_{ij}? [1]

6. 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.
Marks: Correct loop-less transaction forwarding. [1]

7. Peer k generates a random variable T_k on hearing a block at time t_k. Each block has a unique ID, say `BlkID`. 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. Justify the chosen mean value for T_k. Note that smaller mean value for T_k is equivalent to saying k has more CPU power in a proof-of-work system. If peer k has not heard another block before time $t_k + T_k$ then it broadcasts a block which lists the `BlkID` of the previous block in the longest chain it has received so far (use time to break ties in chains of equal length). 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.
Marks:
Correct implementation of T_k, with explanation for the choice of a particular mean. [2]
Proper resolution of forks, to prune the per node block tree to a chain. [5]

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. [2]

Use an appropriate visualization tool to study the trees of different nodes (suitable choices can be gnuplot, matlab, or any other visualization tool). 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 *detailed insights* to explain your observations.
Marks:
Proper study with a particular visualization tool using different parameters. [3]
Insight and critique of the observed values. [2]

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 detailing your findings along with pictures of typical trees and appropriate insight.
Marks: Proper documentation. [2]