COL819: Programming Assignment 3: Implement Bitcoin

March 17, 2021

1 Logistics

- Due date: To be decided
- Maximum marks: 100
- Can be done in groups of two.
- Submissions on Moodle: Code and report has to submitted.
- Languages: Java/C++/Scala/Google Go/Python (We recommend: Java (version > 8))
- Report must be in LaTeX; use vector graphics for images. Do not save your images in the jpeg, png, or bmp formats. Only save your images in the pdf or svg formats. Draw your diagrams using inkscape and plot the graphs using the matplotlib library.

2 Problem Statement

In this assignment, you have to implement a complete Bitcoin system with multi-transaction support. Broadly, you need to handle the transactions, integrity checks, digital signatures, proof-of-work, and consensus requirements. After the implementation, you need to test it out by carrying out some transactions. Furthermore, you need to evaluate your Bitcoin implementation's security by adding dishonest nodes into the system. Your job is to keep adding dishonest nodes till a fork attack is successful. Finally, you are required to implement smart-contracts in your Bitcoin implementation.

3 Bitcoin [20 Marks]

In this part of the assignment, you need to implement a complete Bitcoin system. Few things to keep in mind:

- You need to implement nodes as separate processes or threads (same as GHS).
- The proof-of-work can be made simpler as per your system's capabilities.
- You need to handle the creation of the genesis block.
- You need to ensure the integrity of the blocks using Merkle trees.
- You have to assign a < public, private > key pair to every node.
- You need to ensure that Nakamoto consensus holds.
- Unless otherwise specified, assume your nodes are fault-free.

4 Transactions [25 Marks]

After successfully implementing Bitcoin, create a network of 10 nodes and perform some transactions using Bitcoins.

- Print the initial state of each node, i.e., after the genesis block creation.
- Print the transactions performed.
- Nodes are given a fixed number of Bitcoins as an incentive for adding a block.
- Print the final state of the node after all the transactions.
- Print a log of the transactions executed.

Also, please design a test case to show multi-transaction support.

Vary the value of the hyper-parameters of your blockchain, such as the hash size, properties of the Merkle tree like the arity, size of the nonce, and report the time and space implications because of these changes using a series of plots. Comment on the trends observed. This should be scalable for up to 100 nodes. You will be asked to run this during grading.

5 Security: Dishonest nodes [25 Marks]

A dishonest node attempts to fork a chain with malicious intent. However, the consensus protocol of Bitcoin protects against such attacks till the number of honest nodes in the system is greater than a particular threshold. Your task is to find that threshold, i.e., the number of honest nodes required to maintain Bitcoin's security, in a 10, 50, and 100-node system.

6 Report [30 Marks]

You need to write a report (limited to 10 A4-sized pages) with references. The report should contain all the details related to the implementation and the results generated and their explanations. Every single parameter has to be justified either from first principles or from prior work. You need to run each experiment until it reaches a steady state. This needs to be visually shown for a few cases. Finally, be creative. Add as many new experiments as you can with adequate justifications. The report should contain details of the different design choices that you made during the implementation, such as how to implement transactions at a high level, how to do you ensure proof-of-work, etc. It should also contain instructions on how to run the code.

6.1 Graphs

1. The effect of changing the arity of the Merkle tree. Plot a box-plot, with the arity on the x-axis, and the time to perform a transaction on the y-axis (perform 10 transactions).
2. Use a simple plot (line or scatter) to show the change in the size in bytes (x-axis) of the blockchain after k transactions for different arities of the Merkle tree (x-axis). Experimentally determine a value for k. Justify it.
3. Repeat the above two experiments to plot the change in a transaction's mean creation time (y-axis) upon changing the hash size (x-axis).
4. Repeat the above two experiments to observe the change in a transaction's mean creation time (y-axis) upon changing the nonce size (x-axis).
5. Pick a value for the Merkle tree arity, the hash size, and the nonce size based on previous experiments. Using them, plot the time taken by a transaction versus the number of nodes in the Bitcoin system. Use a box-plot with the transaction time on the y-axis and the number of nodes in the x-axis.

Quantify the time it takes to execute smart contracts.

7 General guidance

- Please stick to basic packages during implementation.
- If you are not sure if a particular package is allowed, ask on Piazza.
- Grading will be done based on the correctness of the code and the report's quality. There might be a demo, if required. So please ensure that the submitted code executed on your machine correctly.
- We will run MOSS on the submissions. Anyone found with a copied code, either from the internet or from another student, will be dealt with as per the class policy.