Operating Systems and Distributed Systems

Fall 2023

Project 2 (both parts)

The second projects are open design projects on blockchain. You can have a max of 3 students per team, and you will design and implement a blockchain from scratch. This project has two parts with two separate deadlines, and we refer them as Part I and II, respectively, but you should consider it as a single project.

Part I

In the first part of the project, you will need to implement the Nakamoto (bitcoin) protocol as we introduced in the lecture. A block chain system contains at least two kinds of roles: **Clients** (wallets) that are used by the users of the chain can submit transactions to the chain, and the **Miners** (peers) who participates in the Nakamoto consensus protocol.

You only need to implement the core parts of it, specifically including at least the following:

- A client program that can submit transactions to the miners using RPC
- Miner local functionalities:
  - The offline data structure of a single block
    - (you do **not** have to implement the Merkel tree – you can use that as an improvement in Part II);
  - The offline chain with hash pointers to the previous block
  - A function that can verify the validity of a block, and the entire chain
- Supporting distributed transactions on miners
  - A transaction dissemination mechanism to broadcast transactions to other miners
    - Hint: as a starting point for this part, you can use a static list of all miner nodes, and just send to the list (you can choose other dissemination mechanisms as improvements in Part II)
  - A PoW function with adjustable difficulty
  - A mining algorithm to allow miners to find solution to the PoW puzzle
  - Broadcasting blocks to other miners when successfully solved the PoW puzzle.

- All other miners should check the validity of the block and only accept the valid ones (i.e. tolerate miners that are lying)
- Longest chain rule to choose the right fork if there is a one

To demo to the TAs, your implementation will need to:

1. Run at least 5 miner processes and let them execute the protocol above and generate a chain with at least 100 blocks without error
2. When you adjust the mining difficulty, the block generation speed changes with the same number of miners
3. Demonstrate the case when the blocks get corrupted, miners reject these invalid blocks.
4. Demonstrate the cases where if there is a lying miner (which did not correctly solve the PoW puzzle), the other miners should also reject the block.
5. Demonstrate the case when there is a fork, the longest chain rule is correctly applied.

To demonstrate the last four points, you need to write some sorts of "fake clients" / "fake miners" that simulat a client / miner that generate corrupted blocks.

You can use any open-source libraries in your project (but you cannot directly use an open source blockchain implementation, of course). You should continue to use the server environment you setup for Project 1 Part II. Again, please respect other teams by only using the range of ports assigned to your group.

In addition to demonstrating to the TAs, you should write a brief report on the baseline performance. You should define your own performance/correctness metrics, and you will need to document it clearly.

Things to submit

1) A link to your Git repository and giving TAs the access permissions.
2) A live demo to the TAs, TAs will arrange time with you for the demo
3) As Project I, a Makefile that can build your blockchain agents, and a `deploy` target that can copy your compiled binaries to the nodes in your node list.
4) A README file showing how to start your miner networks, the client, or any other tools / scripts to help the TAs to successfully complete the demo.
5) A written report (not too long, 3-4 pages should be enough), describing your design, assumptions, and key evaluation results. Also document the omissions / changes / additional assumptions about the system you made.

6) A written proposal for your improvement plan (for Part II), no longer than a

single page. You should describe what improvement you propose, what is the method and metric that you can use to show that your improvements work. Also, briefly describe how you plan to do it (no need for details). See the next part for suggestions on improvements, but you can choose your own improvements instead of following these suggestions.

Part II

For the second part of Project 2, you need to propose several improvements to your baseline. Minimal number of the improvements proposed equals to the number of team members.

The improvement can be anywhere from the consensus protocol, networking, local data structure, programming interface or incentive mechanism design, or even designing a better user interaction with the blockchain, or a better PoW algorithm (you need to argue why it is better). Or you can choose to add back features such as Merkel tree or node joining/ leaving protocols you omit in Part I.

For feature improvements, you should be able to demo the feature to the TAs. For performance improvements, you should have figures / reports showing that there are actual improvements.

At the end of the semester, before the end of week 18 (after the final exam). You will need to present a demo of your implementation and performance numbers at the end of the semester.

The project will be graded for the novelty of the improvement design ideas, complexity of the implementation, design document and report writing quality, as well as code quality and actual quality of the *improvements* in your evaluation (relative improvement over the baseline in Part I).

Things to submit

1) Continue to use the same Git repo as in Part I
2) A demo to the TAs, TAs will arrange time with you for the demo
3) TAs will NOT try to build and run this part, so it is up to you if you want to make a good Makefile or not (I suggest that you do for your own convenience)
4) Update the README of how to build and run your project – this is also for your convenience.
5) Revise / extend your written report and proposal from Part I (not too long, 8 pages including the part from Part I should be enough). In this part, you should add contents to describe how you achieve your improvement, and

any additional assumptions you add. Also, you need to show your key improvement results. You should also analyze how you achieved the improvements.

6) Note that the minimal number of the improvements proposed equals to the number of team members. i.e. with a three people team, you need at least show three improvements.