CSCI 520: Blockchain Programming Project

Due: 5/2/18

Introduction

In this assignment you will put your distributed system development skills to use to build a protocol of your own design. You will implement a modification to the proof of stake mechanism that addresses two problems of proof of stake mechanisms - “rich get richer” and “lazy miner.”

Assignment

Your task is to implement a basic blockchain protocol and then extend it to support modified proof of work. As a starting point for the blockchain implementation you can use this video. (To simplify your task you do NOT need implement transaction signatures, as explained in this video.)

To understand proof of work and its limitations, please watch this video. Two of the problems with proof of work is that rich get richer (verifiers with more stake get to mint blocks with larger transactions) and that verifiers can be lazy (and fail to generate a block once they win consensus to create one). Below is an outline of a protocol that addresses these problems. Your task is to turn it into a protocol definition and to implement it.

Part 1 - Blockchain

Start by implementing the basic blockchain on four miners. You may adjust the complexity of the computational problem to create a suitable block generation rate. Your implementation will need to send blocks around to other miners, so that they may grow the blockchain (log) in their own storage.

You may assume that miners have a set of transactions to insert into each block. These transactions do not need to be signed, but miners should verify that spenders have the balance to spend.

Part 2 - Modified proof of stake

Let verifiers (miners) create blocks in each time instance (say every 20 seconds) according to a probability *p*, instead of by mining. Verifiers with a block interact through consensus to decide which one of them is allowed to add their block to the chain. The winner contacts other verifiers, who sign the block, if the containing transactions do not contain double spending transactions (spender has money to spend) and if the block creator does not exceed *p* in creating the block. Once a block has signatures from enough verifiers, such that their stake exceeds the total of the transactions, the block can be added to the blockchain. Finally, a reward for the verifier that created the block and those who signed it is recorded in the block.

This mechanism solves the “rich get richer” problem by assigning the reward to multiple miners that pool their stake. It also solves the “lazy miner” problem by making sure a block is generated before a verifier for the block is assigned.

It is your job to translate this protocol into a specification and to implement it.

Deployment

You must implement a system with four server nodes. Each node will be run on a different machine. The blockchain should be written to permanent storage, so that it will survive node crashes. You need to provide, at least, a minimal user interface (UI) to direct miner actions. A text-based UI is fine. The contents of the blockchain must also be viewable, for example by reading a file, so long as the file is human readable.

You will deploy your project on Amazon EC2, on four machines in different regions. You will use micro-instances. You should sign up for AWS Educate to get your $35 credit for use of Amazon's cloud infrastructure. This should be enough for you to deploy and demo the project. You should not do your development in AWS, but push your code from a repository (GitHub, BitBucket) to your AWS instances to test and demo. Please be careful to use a private repository, or otherwise protect private keys, so that your AWS accounts are not hacked.

Grading

Your grade will be based on a screencast that demonstrates the functionality of your code. I expect the demonstration to show the different types of events that miners may take. I will be looking for functionality and correctness of your mechanisms.

Specifically, you will be graded according to the following rubric:

6 points - Blockchain
2 - correctly mines a block
1 - sends a block to other miners
3 - miners verify incoming block and add to the blockchain

14 points - Proof of Stake
1 - a verifier generates a block with probability *p*
3 - verifiers reach consensus on accepted blocks
3 - block signed with sufficient proof of stake
1 - signed block contains no double spending events
2 - signed block does not exceed *p* for client
2 - verifiers get reward for creating a block
2 - verifiers get reward for signing a block

What to submit?

1. A link to a YouTube video showing your code running on AWS instances.
2. Your source code (unzipped) for D2L plagiarism check