CSC 400: Project Proposal

By Jessica Boe

# Motivation

Mining for cryptocurrencies such as bitcoin in a proof-of-work environment is deliberately consumptive of energy. According to the Bitcoin Energy Consumption Index (Digiconomist), Bitcoin and Bitcoin Cash account for use of 0.31% of the world’s energy consumption. Ethereum is another type of crypto that currently relies on proof of work, allowing developers to build and deploy smart contracts with decentralized applications using its token, Ether. Ether can be traded like bitcoin or used to pay for services and transaction fees on the Ethereum network.

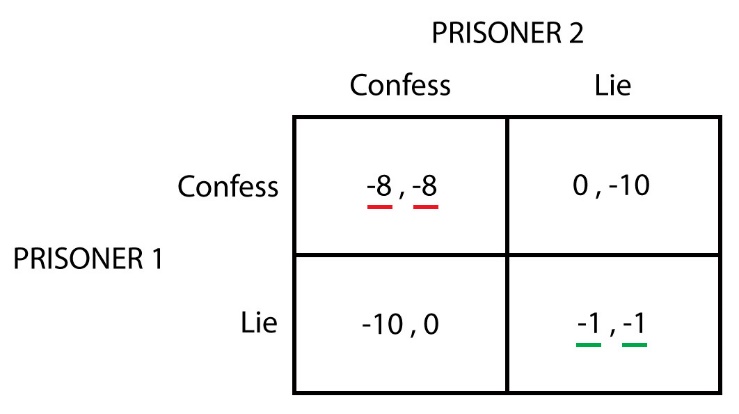
Due to all of the unnecessary costs of electricity and equipment that operating a cryptocurrency using proof of work entail, Ethereum is switching to a hybrid of a proof of work environment with what is known as a proof of stake environment. Using its new Casper FFG mechanism, or Casper the Friendly Finality Gadget, Ethereum will rely on virtual miners, or validators, who stake Ether to the system. Validators are rewarded when they follow the rules and lose some or all of their deposit when they break them. Proof of stake is used as a checkpoint every 50 blocks to offer evidence that the blocks hold valid transactions.

Because proof of stake for Ethereum is so new and unproven, it is very important for this transition to be successful for the extremely large Ethereum community, who combinedly hold over $50 billion of Ethereum (price data as of May 29, 2018). If the transition goes poorly, and long-term users and holders lose Ether because of people trying to cheat the system, people may lose faith in Ethereum. eth-sim is a simulator that shows how a crypto economy may look with different user types, population numbers, time periods, incentives, and disincentives for a proof of stake-based system, and whether the current incentive structure is playing out fairly or is being exploited.

# Overview

eth-sim will allow its user to set parameters and run a program via a console-based system that enables them to test whether a sample population of a crypto economy is adequately incentivized into validating transactions on the blockchain. The requirement for the amount of Ether to be staked can be tweaked, as a smaller population will have a large enough stake of Ether and thus be able to validate transactions. If the “rich” keep getting “richer”, what effects will that have on the overall economy if an actor behaves (or starts to behave) selfishly? If only a relatively small amount of Ether needs to be put at stake and the pool for potential validators grows larger, will the “bad” actors outweigh those acting fairly? The simulations will act according to a decision matrix in order to choose the “best” possible outcome for themselves. eth-sim will ultimately allow you to compute results, such as annual return on an actor’s stake in the system with a series of validators with different preprogrammed strategies. This simulator will take on a similar vein to Nicky Case’s “The Evolution of Trust”, which shows how actors behave with different incentive structures and allows the user to pick out the number of actors in a scenario. The link to this game can be found here: <http://ncase.me/trust/>

Different actors will have different preprogrammed behaviors. They will behave in various ways, according to how incentivized they are to act selfishly or to contribute to the system, which is one such parameter. The way an actor behaves will be based on a variant of the prisoner’s dilemma, as seen below.



An actor’s user type will either increase or decrease their way of behaving a certain way when they are randomly called into the validator pool. For example, an actor that will always be honest (in the case of the matrix above, “confess”) will have their chances of cooperating multiplied by a ratio that lends to them always behaving like that. There will be quite a few different actor types, and it is up to the user to decide how many of each variant will participate. There will be other ways that a user can attain Ether than by joining the simulation pool, for example, in randomly being chosen to play a round of this prisoner’s dilemma-themed game with other economy participants. Eventually, different actor types will fall out of consideration for the validation pool as they lose enough Ether to partake in staking, and more of the successful, richer user type will join the simulation, bringing its behaviors to a new generation of validators. After a certain user-defined period of simulated time, the simulator will output a series of files that test your settings. These include both high level and detailed statistics regarding the percentage of actors who acted fairly and unfairly, the actors who made the most, less, and average and their respective user classes, and how economically equal the system is with the given inputs.

# Architecture

The system will rely on user input through the command line to create a botnet of users with different Ethereum addresses. They will interact with a series of smart contracts that commit state-modifying transactions to the system. From here, a validation pool is chosen regularly (Ethereum’s average block time is currently about 14 seconds; with the new proof-of-stake system validating every 50 blocks, the average wait time can be fixed at 11.66 minutes per new validation pool chosen, or it could also be tweaked per user specifications). In the meantime, prisoner’s dilemma-styled transactions as described in “Overview” are taking place between a user-specified number of pairs of bots with unique Ethereum addresses, or this function can be disabled. To be included in the validation pool, the bot must hold the amount of Ether needed to be privy to the pool, and according to incentivization/disincentivization structures and a bot’s user type, the bot will or will not honestly partake in the validation pool. If disincentives outweigh the incentives, a user may behave badly; if this becomes a recurring theme, it could lead to massive inequality and catastrophe for the system. The system must interact with various smart contracts and an Ethereum test network, which will work in conjunction with the data being generated.

# Platform

The system will be implemented in Python, which interacts with eth-sim’s smart contracts through the Ethereum Virtual Machine using [Py-Env](https://github.com/ethereum/py-evm) and [Ethereum Tester](https://github.com/ethereum/eth-tester). The files that will be spit out will be in JSON format, which will be analyzed by a side process that will output cleaner files in the form of .txt and .csv documents. The user can then access these through a text editor or Excel-like program, respectively.

# User Interaction

Because the vast majority of eth-sim’s usefulness will be complex operations under the hood, the user interaction will be as simple as possible, seeking to inform the end user of the meanings of the various parameters and the implications they will have on the overall outcome. It will be console-based, prompting the user for inputs, in addition to explaining them. The user will also have the ability to run a scenario several times, allowing for more conclusive results that minimize outliers. This main menu will allow for:

1.Testing of one scenario

2. Testing of multiple scenarios

3. A detailed explanation of what is going on in the background, or “Help”

4. Exit

From their choice on the main menu, the console will close, and a series of files will be generated with explanations of the outcome, high-level statistics regarding the economy overall, and detailed event and transactional data for every bot generated.

eth-sim