ORIE 4580 Final Project

Ibrahim, Cameron
cai29@cornell.edu

Wang, Irina
iyw3@cornell.edu

November 2018

1 Executive Summary

Here we were tasked with the simulation and analysis of the transaction economy of a system built upon the theoretical virtual currency BigRedCurrency (BRC). In particular, we were to focus on the physics of the system as a function of the rate of transactions created, the maximum number of transactions allowed in a single block of the currency ledger and the rate at which blocks are processed in to the ledger. Features of the simulation of particular interest were the throughput of the system, the average delay experienced by transactions and the average rate of fees collected by block miners.

To this end, we modeled the system as a queue of transactions, with priority within the queue given by the size of the randomly generated accompanying fee. Transactions leave the queue in blocks of size K, which are processed at a rate μ. At this point, we calculate the appropriate metrics for block based on the transactions it contains. With this data, across multiple replications and settings of the parameters, we plot and gain an understanding of the systems behavior.

To this end, we found that the most effective means for minimizing the total average delay in the is to maximize μ and K available. Note: there are significantly diminishing returns on increasing K once it gets past the point where $\mu K \geq \lambda$. As such, it is not recommended that K be increased ad infinum, especially since there are security vulnerabilities associated using large block sizes. Furthermore, should one of μ or K be small, the other must be sufficiently large to satisfy the above inequality, in order to ensure that the queue length does not increase indefinitely.

For individuals making use of BRC, we found that the fees required to ensure a low delay on their transactions increases exponentially the smaller the delay gets. In particular, to achieve an average delay of less than 6 minutes, a fee greater than 0.2 BRC will be required. This decreases to roughly 0.1 BRC for an average of 12 minutes, and stays roughly in that range. The fee required to achieve a smaller delay are irrelevant of the actual rate of fees paid; rather, it relies only on the combination of fee rate as well as the total amount of the transaction. As such, smaller transactions will receive a relative penalty if they need to lower their delay amount.

Finally, when we assign a uniform chance of processing a block among the population of miners, we find that the expected profits of any given miner decrease exponentially. As

such, the fees produced by mining are really only sufficient for supporting a few miners, as the cost of mining would quickly out weight the expected profits, exceeding the so called "fair price" a miner should be willing to pay to participate in mining. Specifically, the cost $c \leq \frac{AverageRateofFees}{N^2}$ where N is the number of miners.

2 BigRedCoins as a Parameterized System

Virtual currencies have a huge potential for application in our current digital age.

However, the logistics of their actual use within a community requires thorough study before widespread, mainstream usage could be feasible. Here we perform one such study;

BigRedCoins (or BRC) is a proposed blockchain based cryptocurrency out of Llenroc University. BRC is presented as a means for creating an internal, virtual marketplace for students, somewhat akin to prepaid pusedocurrencies offered by many universities. Students within the marketplace may act as miners of the BRC ledger, and would be compensated with small fees accompanying the transactions in the mined block. Because of the limited number of transactions that may be included in a block, this creates a relative priority for transactions with higher accompanying fees.

As BRC would be traded only among the population of students within a University, it provides a more approachable context than attempting to study the physics of a virtual currency based economy in a large population. In particular, We narrow our examination to a few key parameters of the system:

These are:

K The maximum number of transactions that can be processed in a single block

λ The average rate that transactions are made across all students

μ The average rate at which blocks are processed by miners

These parameters are key to the feasible implementation of a virtual currency based economy, and so we seek here to identify minimal requirements for system function, as well as optimal configurations to prescribe system details to the potential creators of BRC. In order to discuss optimal configurations, however, we must first consider what metrics we want to consider.

Of particular concern is the stability of the system; essentially, that we can feasibly process all of the transactions being created. However, that is strictly the bare minimum that would qualify an effective system. We would additionally want to ensure that the throughput of the system remains high, such that problems won't arise during busy periods. Correspondingly, we want to ensure the all transactions are eventually reached in a reasonable period of time; thus, studying the delay experienced by transactions in the system is a must. What's more, in order to ensure the system remains functional, reasonable incentive must be provided to the miners to continue processing blocks. Therefore, we would want to sure that there is a significant rate of fees being collected, which relates closely to the throughput and delay behavior of the system.

As such, when we discuss optimal configurations, we wish to find systems such that we process transactions efficiently, ensuring that the queue is regularly emptied, or at the least, refreshed, all while providing reasonable incentive for block miners.

3 Simulation of the BRC Economy

It is useful for modeling the BRC currency to consider, if you would, a bus. For a bus, you have a number of people waiting in a queue at the bus stop for a bus to arrive. When the bus arrives, a number of people board at once. If the bus should reach capacity, the people remaining in the queue must then wait for the next one.

This, in a sense, is how we are treating the BRC economy. Generated transactions are placed in a queue, where they wait for a "bus" to arrive. When a "bus", or rather, the block containing the transactions, is processed, it immediately returns to pick up the next group of transactions. We don't particularly care where the "bus" arrives (i.e. which miner processes the block), only that it has arrived. What's more, if we generally assume that the students creating the transactions have sufficient funds to fulfill them (a generous assumption, but not a terribly impactful one), we no longer need to track who initiates/recieves each transaction.

We have then very neatly reduced our model to a queue, to which transactions arrive according to our defined rate λ. The transactions in the queue are processed in sequence, in groups of at most K. Each group is processed at the rate given by μ; it is common for cryptocurrencies to attempt to keep this rate constant, independent of the number of miners, and so we will do so here.

There are a number of variations on this form that we are able to consider, in terms of model architecture, rather than simply adjusting parameters. For example, it is likely that miners would want to attempt to get the most amount of fees possible for any particular block. As such, it is quite likely that they would prioritize transactions with high accompanying fees. We are able to to reflect this phenomenon in our model by sorting the queue by transaction value, such that fees with high transactions will be processed first.

To make clearer the impact that fees have, we break the possible fees into two classes: high rate and low rate. The high rate fee class will pay a 2% additional fee on top of the value of the transaction, while the low rate fee class will pay a 1% additional fee. While this seems restricting, keep in mind that the actual value of the fee is dependent on the actual value of the transaction, which we choose randomly from the range of $\{5,\ldots,25\}$, so there is actually some variation in the fee values.

4 Model Verification

Burke's Theorem states that for M/M/1 queues, in the steady state, arrivals and departures both follow a Poisson Process with the same rate parameter λ. This is ensured by setting the service rate to be greater than the arrival rate. In the BRC simulation, we treat the pushing of blocks as a modified version of the M/M/1 queue, where miners are treated as a single server with rate μ, and which can process K transactions simultaneously. We have the arrival rate of transactions set to be $\lambda = 120$, thus according to Burke's Theorem, the departure rate must also be equal to 120. The service rate, then, must be set to be greater than 120, and this is $\mu \times K$, the rate at which transactions are processed. We thus arrive at the requirement $\mu \times K > 120$ for steady-state simulations. With this requirement satisfied, the queue length will be steady and will continuously arrive back at 0. Any other values will cause the queue length to continuously increase. We use this fact to verify our model; we observe the queue lengths for different combinations of μ and K.

Top: $\mu = 30$. We test out 3 different values for K. When $K = 3$ (left), the service rate is $3 \times 30 = 90$, below the arrival rate of 120. When $K = 4$ (middle), the service rate is $4 \times 30 = 120$, exactly equal to the arrival rate of 120. When $K = 5$ (right), the service rate is $5 \times 30 = 150$, greater than the arrival rate of 120.

Bottom: $\mu = 10$. We repeat the process for a different value of μ, again observing the effect of different Ks.

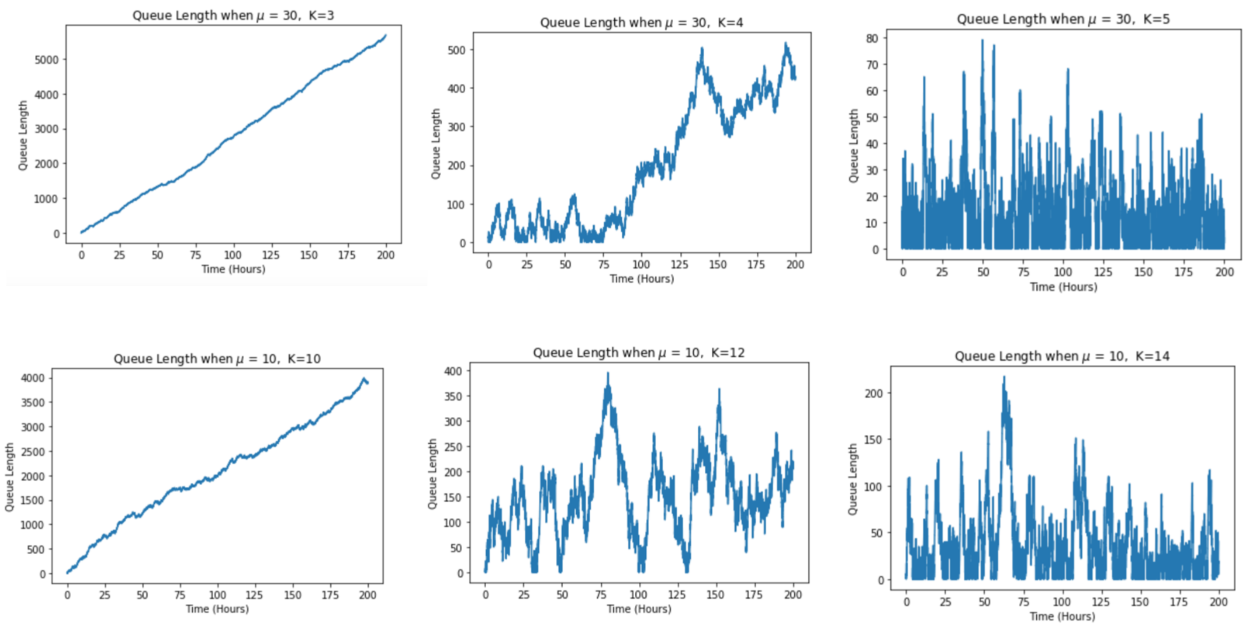


Figure 1: Queue lengths for transactions for various levels of K. Steady-state behaviour is reached only when a certain threshold for K is passed.

Clearly, we observe that when the service rate is set equal to the arrival rate, the queue length begins to stabilize. It is, however, not until the service rate is set above the arrival rate that the queue length truly stabilizes and exhibits steady-state behaviour, centering at 0. The result is the same for both values of μ, suggesting that this occurs not due to the actual values of μ and K, but to their values relative to each other - their product and its relations to λ. This matches our conjecture above, and thus verifies the correctness of the model.

In addition, we also check the throughput of the system, for $\mu = 30$ and $K = 5$. We plot the simulation results below:

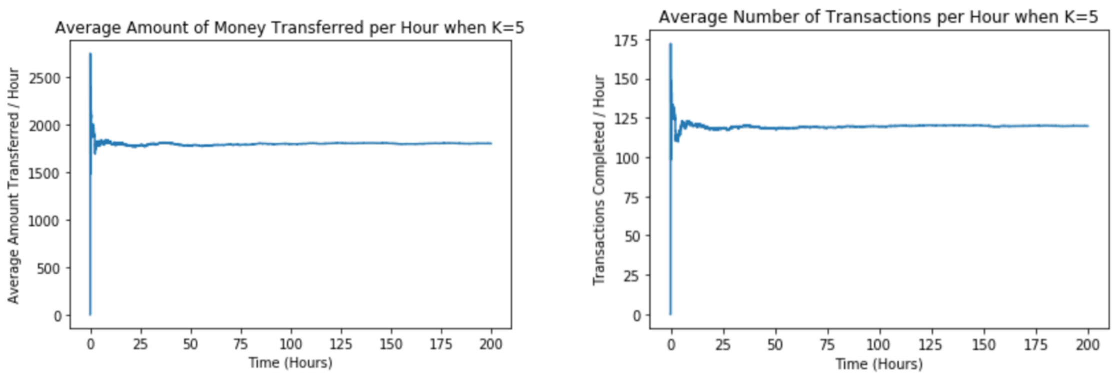


Figure 2: Throughput of the system, in terms of transactions and amount of BRCs. No warm-up period is set, as the time ran is long enough to cancel out the effects. Values at 200 hours: Amount = 1804, Number of Transactions = 119.7

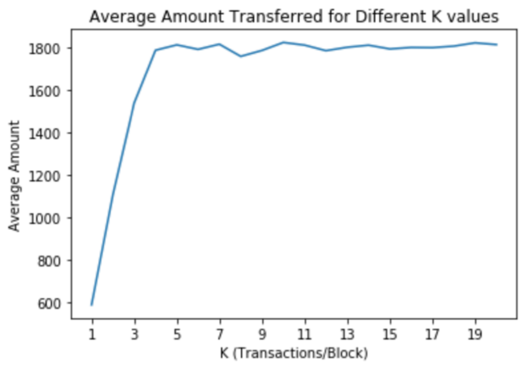
The arrival rate and departure rate are both set to be $120/hour$, and the simulation result matches almost perfectly, at 119.7 transactions per hour.

The amount of money transferred also corroborates with the expected value. In the basic model, we have set the amount per transaction to be uniform between 5 and 25, thus, the mean amount per transaction is $\frac{5+25}{2} = 15$, and with a rate of 120 transactions per hour, the mean amount transferred per hour should be $15 \times 120 = 1800$. The simulation has the average at 1804, matching the expectations and again verifying the model.

5 Analysis of Replications

As detailed above, the steady-state behaviour, given that μ and K are set such that their product exceeds λ, is a transaction rate of λ transactions per hour, and transaction amount of BRCs/hour dependant on λ and the distribution for singular transaction amounts. We have verified this for a K value that places the product just above lambda. It is, however, still of interest to scrutinize the throughput of the system for various levels of $K's$, to have visual basis for our conjectures.

Below, for a μ value of 30, we have graphed the average transaction amount and average number of transactions for K ranging from 1 to 20.



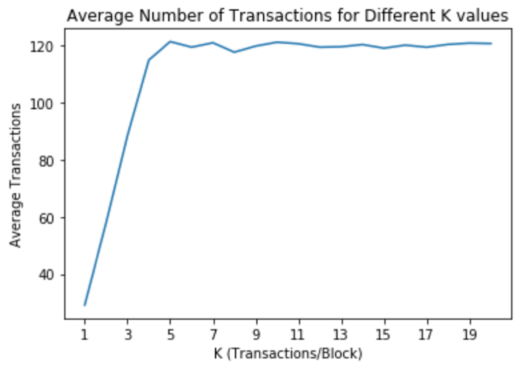


Figure 3: Throughput of the system for different K values

As could be observed, the throughput of the system does not reach its maximum - the steady state amount of 120 transactions and 1800 BRCs - until a certain K value is reached. From the simulation above, where $\mu = 30$, this threshold occurs at K > 4. This corroborates exactly with our prior analysis, where we concluded that $K \times \mu > 120$ for steady state to be achieved. Only at this level will there be enough transactions to be placed in each block, such that maximum throughput is achieved.

We then scrutinize the fees associated with each block. The collection rate of fees follow the same trend as the transfer rate of BRCs, as shown below. Again, the optimal value is achieved only when K goes above 4. To optimize the system, then, for a system with $\mu = 30$, up to this point, it seems advisable to take any value of K above 4.

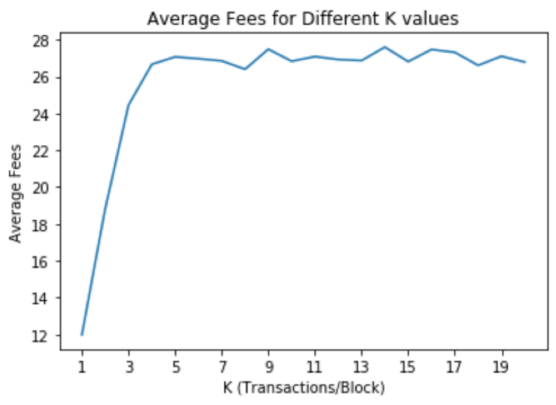


Figure 4: Collection rate of fees for different K values

What is more interesting to observe, perhaps, is the delay experienced by transactions, for various K levels. We collect this data by updating, after each block pushed, the average delay experienced by transactions, calculated as total delay time divided by the total number of transactions completed, and arrive at the plots shown below. A range of K values are chosen, and it could be seen that for values of K above the threshold, after a certain period of time, the average delay does approach a steady number. For K values below the threshold, the delay continuously increases. A good justification for this behavior is the increasing queue size. Due to the limited number of transactions allowed on each block, transactions must wait increasingly longer periods of time before they are processed. On the other hand, the steady-state simulations has a queue size that frequently returns to 0, thus maintaining the delay time at a controllable value.

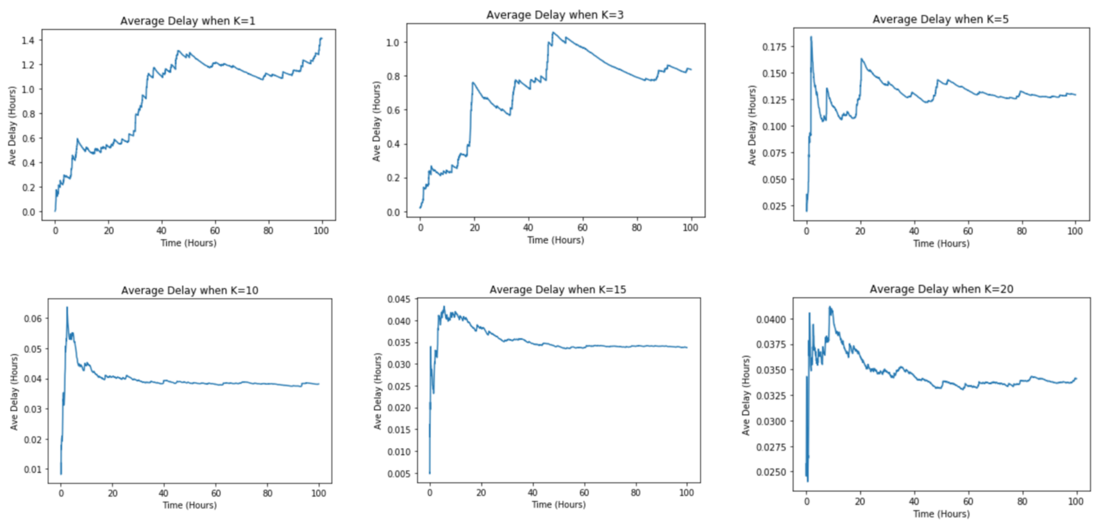


Figure 5: Delay experienced by transactions over time, for different K values

To find an optimal K value for this set-up, we plot the terminal delay value for each K, as shown below. It seems that, to achieve optimal delay - a value of 0.034 hours, from data - an even larger value of K than recommended above is required. From the graph, we want the K value to be at least 7, for the beginning of the smooth plateau.

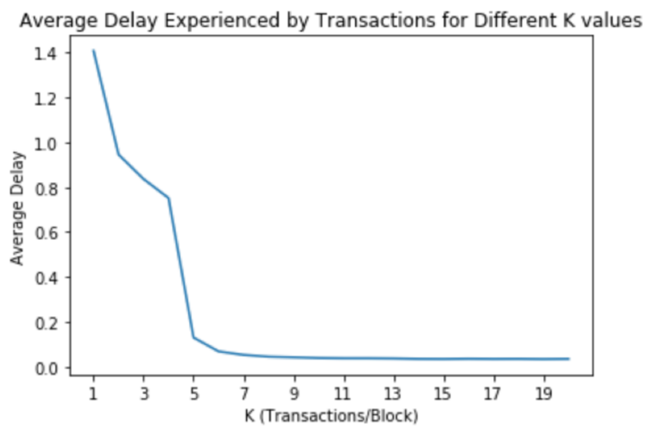
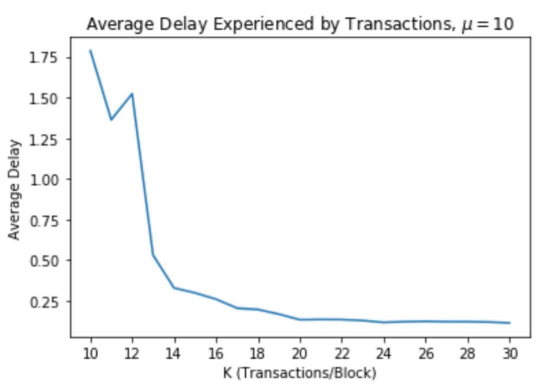


Figure 6: Delay experienced by transactions for different K values. Optimal delay = 0.034

To see if this requirement of a larger K value is not an isolated occurrence for this particular μ value of 30, we repeat the process for $\mu = 10$ and $\mu = 20$. The threshold K for these to achieve steady state are, respectively, 12 and 6. From the graphs below, we see that at these K values, the average delay are not in fact optimal. To achieve low delay, we wish to increase K to a value higher than the threshold value. This increase should also be relative to the threshold K value - the higher the threshold, the larger the difference is between optimal K and threshold K, as could be inferred from the three graphs in Figures 6 and 7.



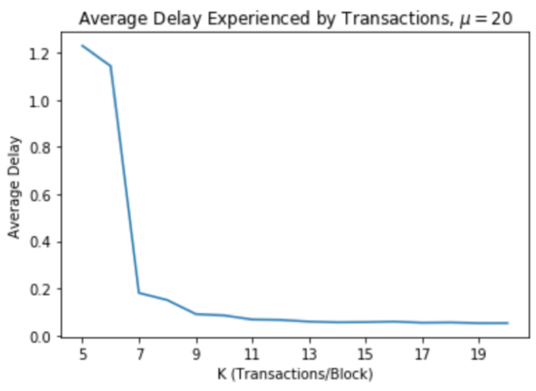


Figure 7: Delay experienced by transactions for different K values. Optimal delay = 0.053 for $\mu = 20$, Optimal delay = 0.115 for $\mu = 10$.

We also observe that, to achieve optimal delay, we wish to have the mining rate, μ, as large as possible. The three optimal delay values are, for $\mu = 10, 20, 30$ respectively, are $0.115, 0.053, 0.034$. Clearly, it is decreasing with the increase of μ. To observe the trend further, we ran 10 replications of the above for more values of μ, and arrived at the following distribution for optimal delay vs. μ.

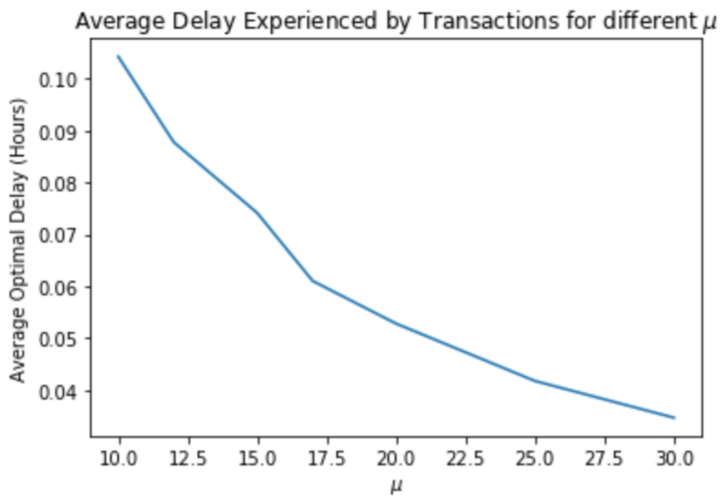


Figure 8: Delay experienced by transactions for various μ

This seems to follow an exponential decay, which may be a topic for further analysis. For our purposes, however, we can conclude that we wish to set μ to its maximum value of 30 blocks per hour, and K sufficiently larger than threshold, to achieve optimal delay.

Up to now, we have considered the delay of transactions indiscriminately of the transactions themselves. However, a significant question to be answered for members of the BRC system would be what fee is recommended in order to achieve a particular delay. We have implemented the model with the assumption that transactions with higher fees are processed first, thus there should be a distinction in delay time experienced by different levels of fees offered. An understanding of priority implied by the accompanying fee has important implications on effective use of the system. To study this, we ran 50 replications of our simulation with 12 transactions per block, 30 blocks per hour, and 120 arrivals per hour. An initial survey of the data gives the following box plots, for average transaction amounts and delay times. The notch gives the confidence interval for the median values. We see that the average delay indeed matches our optimal results from above.

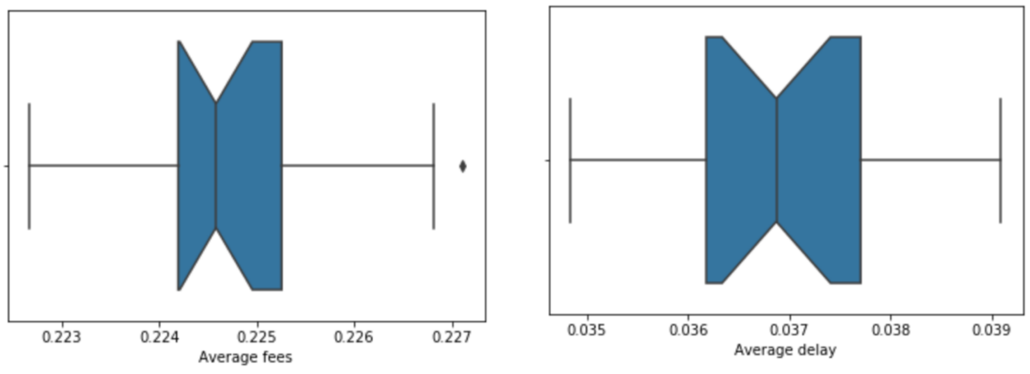


Figure 9: Box plots for distribution of transactions

We then took a subset of the resulting data and fit a Gaussian Process Model to learn the underlying function for the distribution of delay vs transaction fee.

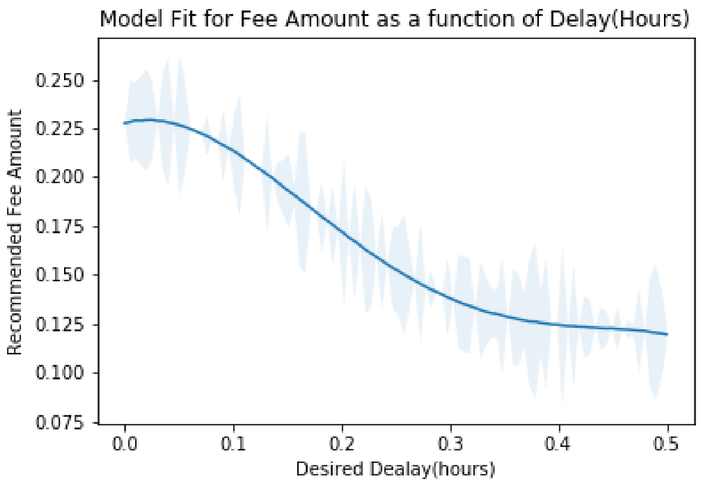


Figure 10: Prescribed Fee Amount as a function of Desired Delay(hours)

In this plot, we see the mean value of the learned function, with a shaded region showing a one standard deviation interval about the mean function. Notice that the amount of fees recommended decays exponentially as the desired delay increases. This makes intuitive sense; as we have seen previously, the queue tends to empty itself out over a few blocks. The competition to get into the block then tends to be fierce, necessitating greater and greater fee amounts to guarantee entry, where as when higher delays are acceptable, it is rather irrelevant. It is also worth noting that this function prescribes a total fee in order to guarantee smaller delays. As such, a transaction with a small amount desiring a low delay would have a much higher fee rate relative to it's total amount.

Desired Delay(Hours)	Recommended Fee
0.1	0.21331024
0.2	0.17179489
0.3	0.13788986
0.4	0.12442398
0.4	0.11941147

These fees serve as an incentive for the miners processing block to include a given transaction over another. Additionally, the total collection of fees serve as the incentive for people to mine blocks at all. However, one thing we have yet to mention is that there is often a cost associated with mining. As such, it is important to consider how the average profit received by miner's changes as a function of the total population of miners. We modify our model slightly, such that for each block, we pick a miner uniformly at random to assign fees to. We then take the average profits of the miners for each simulation and plot them.

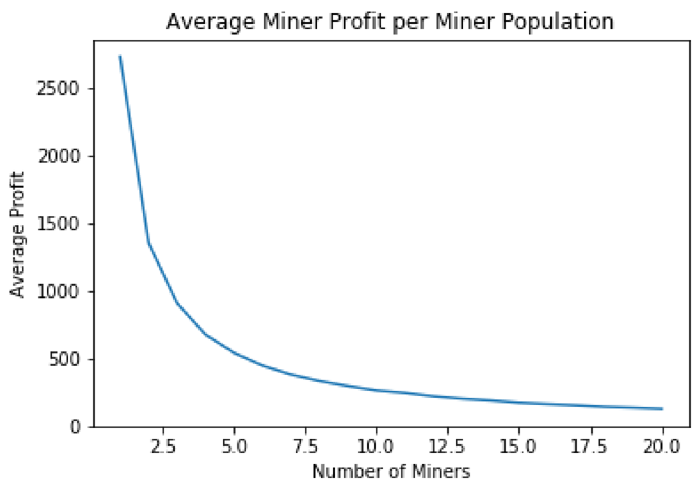


Figure 11: The average profit an individual miner delays exponentially as the number of miners increase

As we see here, there is somewhat of a tragedy of the commons scenario going on, where even a few additional miners causes the average profits to plunge greatly. Specifically, $c \leq \frac{AverageRateofFees}{N^2}$ where N is the number of miners. As such, even relatively small costs associated with mining can drive many miners out, as if the costs associated with mining should exceed the expected profits (exceeds the "fair price"), then there one should clearly not attempt mining, as the average miner would simply be losing money.

6 Conclusions

As above, we have concluded that in order for steady-state to be achieved, the $\mu, K, and \lambda$ rates must be set such that $\mu \times K > \lambda$. Any value that do not satisfy this requirement will result in an non-optimal throughput, increasingly large queues, and, subsequently, increasing delay times - which, essentially sums up a failed BRC model. For maximum efficiency to be achieved, in terms of throughput of transaction numbers and amounts, the K value must be greater than the threshold value - defined to be the value where $\mu \times K = \lambda$. For values of K and μ such that the total transaction processing rate is greater than λ, the rate of transaction arrivals, the throughput cannot increase further, as goverened by Burke's theorem.

However, in addition to satisfying the minimum requirement for K, in order to optimize the delay time of transactions, we wish to maximize the μ value, and also raise K to a value where the optimal delay time plateaus.

In terms of delay times for specific transactions, we found that the cost associated with decreasing delay increases exponentially as the delay approaches 0. This indicates a relative penalty on small transactions that require quick processing times, as the delay relies only on the total fee, not on the transaction rate itself.

Finally, the we find that the average profits of the miners suffers from the tragedy of the commons, as the shared profits decrease the average profit exponentially as the number of miners increase. This, in turn prohibits the system from supporting large numbers of miners, as the costs would quickly exceed the fair price that the system would dictate. Specifically, $c \leq \frac{AverageRateofFees}{N^2}$ where N is the number of miners.

7 Appendix A

7.1 Technical Model Specifications

Here we describe, in detail, using domain specific technical terminology, the specifications for simulation model.

7.1.1 BRC Model

Assumptions:

- At the start of the simulation, all students have sufficient funds to make desired transactions (i.e. no transaction will be rejected), and no students will open or close accounts will the simulation is running.
- The source and destination for transactions are picked uniformly at random from the population of students.
- As soon as a block is created, all other miners become aware of it. Furthermore, a given block is produced by a uniformly random miner.
- The rate μ of block creation is held constant.
- For transaction T_i the transaction amount $a(T_i)$ is given by $Unif\{5, \ldots, 25\}$
- For transaction T_i, the transaction fee $f(T_i)$ is given by $Unif\{0.01, 0.02\}$

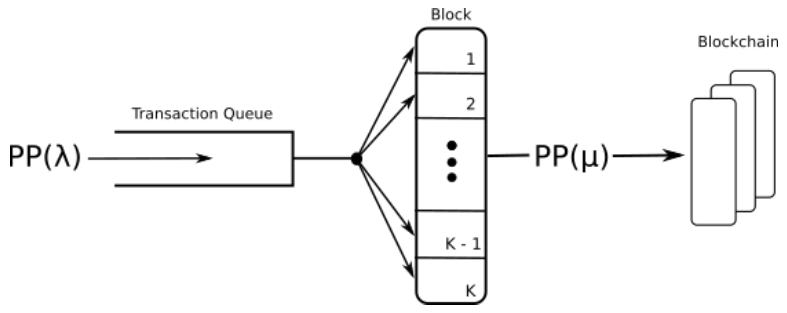


Figure 12: A diagram of the basic BRC model where transactions arrive at rate λ and blocks are completed with rate μ

Transactions arrive according to a Poisson Process with rate λ. When a block is created, up to K transactions may enter service. Blocks are created according to a Poisson Process with rate μ

7.1.2 Code

Our code consists mainly of two objects:

Transaction The transaction object contains valuable information, such as time that it entered the system, its amount, and its fee. Because the ordering within the queue is not first come first served, we must keep track of the arrival of the transaction in order to calculate the wait and delay times.

Queue Having the individual class lets us avoid a large amount of the coordination that has to be done with naked poisson processes. Here, the queue takes care of a great amount of the management for us. We keep track of the time each block is created, the number of transactions within, the total amount transferred per block, the total amount of fees per block, a list of waiting entities, etc. Using this, in the event of a block push, we merely call the queue's pushBlock method, which takes care of all the recording of information.

7.1.3 Gaussian Processes

We made use of Gaussian Processes for model fitting at points in this project. Gaussian Processes, given a kernel function and a dataset $D = \{(x_i, y_i\}_i$, learn a mean function and covariance function for the data. Where they arise, we use an RBF covariance kernel. More can be learned about Gaussian Processes here: `https://scikit-learn.org/stable/modules/classes.html#module-sklearn.gaussian_process`