BigRedCoins Blockchain System Simulation

ORIE 4580/5580 Course Project

Qiushuhao Fu (qf42), Chenxin Guo (cg633), Chenyu Luo (cl2556), Kyna Zhu (jz893)

Monday, December 10, 2017

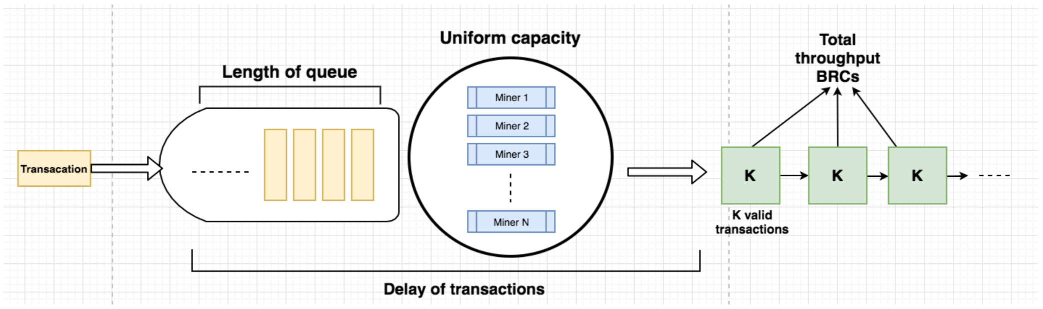
1. Executive Summary

A team at Llenroc University is developing a virtual currency system BigRedCoins (BRC), and has already conquer the problem in software engineering, networking and cryptography field. However, they want to know whether the system could stay stable after students start to use the system as an online ledger. By studying the delay and throughput of BRC transactions, the BRC team wants to figure out the parameters for the optimal configuration eventually.

We were tasked by the BRC team to construct a simulation model to help them test the stability of the system. The simulation model should capture the behaviors of transactions and miners by outputting the metrics including the transaction delays, waiting queue lengths, BRCs processed, and rate of transaction fee for a given duration. There are three main parameters that will affect the performance of the system: the arrival rate of the transactions **λ**, the mining rate **μ** and the block size **K.** For our simulation, we made the assumption that the transactions arrive at a constant rate of 120/hr and examined 64 possible pairs of rate **μ** and size **K**. Based on the model analysis, we restrained the optimal parameters to 4 possible solutions around **μ** = 30 blocks/hr and **K** = 10. Then, we constructed replications of simulation, came up with confidence intervals of these possible solutions, and made our recommendation for the system's optimal configuration based on our analysis.

As a result, under the assumption that transactions arrive at a stable rate of 120/hr, we recommend the BRC team to use **μ** = 30 blocks/hr and **K** = 9 to maximize the efficiency of its system.

2. Problem Description



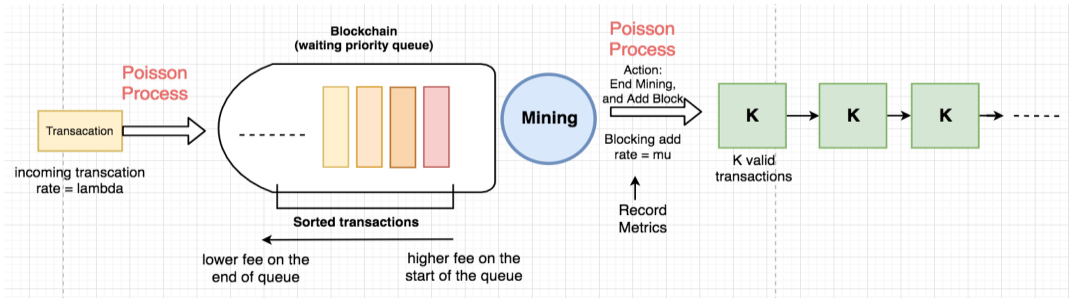
The system of BigRedCoins (BRCs) as an online ledger system has a similar format to other blockchain systems: each transaction is validated by miners who motivated by the transaction fee as a percentage of the total transaction amount. For every transaction, the delay is defined as the addition of waiting time in the common transaction queue and in the current block before being added/mined. The BRC team is worried about there may occur long delays on transaction validations.

In order to check the feasibility of the blockchain system, and understand the relationship of delays and other metrics, we wante to analyze the processing time, the rate of fee collected, and the throughput of system in terms of number of transactions and BRC (transaction amount). By analyzing the statistics, we are able to answer whether a certain system is stable and able to meet the demand. The current system has the following assumptions:

1) Transactions arrive according to a Poisson process with rate λ, the number of transactions per block is K, and the block mining rate follows the Poisson process with rate μ.
2) For the basic case, we hold λ = 120/hr and choose values of μ and K such that all transactions are processed. The only technical constraint imposed by the cryptographic protocols is that μ can be at most 30 blocks/hr.
3) The arriving transaction amounts (BRCs) are uniformly distributed as integers between 5 and 25, and are i.i.d.; each transaction chooses its fees to be either 1% or 2% of its amount with equal probability.
4) The higher the transaction fee, the more profitable for the miners to validate the transaction information sooner. For simplicity, each miner has identical computing power.

3. Modeling Approach and Assumptions

3.1 Approach


Transacation
incoming transcation rate = lambda
Poisson Process
Blockchain (waiting priority queue)
Sorted transactions
lower fee on the end of queue
higher fee on the start of the queue
Mining
Poisson Process
Action: End Mining, and Add Block
Blocking add rate = mu
Record Metrics
K
K
K
K valid transactions

In order to understand the dynamic inside the virtual system and suggest right system parameters, we build a DES simulator to capture the main details of the BRC system.

First, we construct the smallest unit “transaction” to record the information of its amount, transaction fee to miners as the percentage of the total amount, and the time transaction entering and leaving the system. Then, we create a modular called “blockchain” to handle incoming transactions. “Blockchain” is a priority queue where a transaction with a higher fee will be popped out to the mining process faster, namely will be verified and finalized by miners faster than those with a lower fee. Also, we have three important event handlers implemented as functions: add transactions, start mining, and end mining. Calling add transactions will randomize a transaction and append it to the blockchain. Calling start mining will move the transaction object to a temporary block, and wait until we have all K transaction ready to be verified. We analyze the influence of block size K by modifying K at here. Last, calling end mining will record the time each transaction delayed in the system, the amount of fee collected by miners, and mark the system as available for the next incoming mining event.

During the entire time horizon, we randomize the event of transaction arrival and adding block according to rate λ and μ. By changing the rate and block size K, we are able to observe how the key metrics change and find the optimal parameters.

a. Assumption

i. The mining process will not start until there are at least K transactions exist in the waiting list.

ii. Once the mining process starts, we will not add any new incoming transactions, even with a higher transaction fee, to the currently working block. These new transactions will be delayed in the waiting queue until the current block has been added.

iii. Since all the N miners have identical mining capacity, the mining rate is independent of N. We can simplify the mining process to one server with specific constraints on adding rate and rule.

iv. The initial configuration constraints are listed in section 2.

4. Model Specification

4.1 Objectives and Expectations

The main goal of the simulation is to study the steady-state behavior of the BRC system. Below are detailed descriptions of our objectives and corresponding expectations.

1) Given that the arrival rate λ is 120/hr, we hope to find the threshold values of K and μ which would guarantee that the system can eventually reach a steady-state.
   - Here, we consider the system has a steady-state if its queue length of transactions and delay in processing the transactions are able to converge in the long run. The system “explodes” if the queue length and delay of the system keep increasing and show no sign of converging.
   - Our model functions as M/M/1 system if we set K equals to 1. Given the arrival rate λ = 120/hr, the mining rate μ should be at least 120/hr for the system to achieve a steady state by workload conservation theory. Since μ has an upper limit of 30 blocks/hr, we expected the system to always explode under K=1 scenario. However, setting K = 1 is useful for us to verify our model. We will discuss details about model verification in section (4.2).
   - When K is greater than 1, the system will no longer be M/M/c. We expect the system to explode if the product of K and μ is smaller than λ. However, we will depend on the simulation process to figure out the exact threshold values of K and μ.

2) We hope to find an optimal pair of K and μ which would result in the shortest average queue length and delay in the long run. While a greater μ should always imply a more efficient mining process, an extremely large K may extend the average waiting time of each transaction.
3) We want to understand how the changes in K and μ would affect the steady-state values of (1) the number of transactions, (2) the transaction amount, and (3) the transaction fee that added to the blockchain per hour, in addition to the steady-state queue length and delay.
   - Assuming that all the five values are able to converge to eventually, we defined the period before when the values converge as the warm-up period and the average of the values after the warm-up period as the steady-state values.

4.2 Model Inputs

To achieve the goals described above, we selected 64 pairs **K** and **μ** (eight values each) to run the simulation and conducted the model analysis of the five metrics we described above. Since the value of **μ** has a strict upper bound of 30 blocks/hr, we picked the eight values of **μ** to be {1,3,5,10,15,20,25,30}. We selected the same set of values for **K**, such that we can observe the behaviors of the metrics when the values of **K** and **μ** change relative to each other.

We set the total running time of our simulation to be 300 hours, such that we could clearly distinguish the convergent cases from the non-convergent cases from the plots. We set the warm-up period to be 200 hours since all metrics appear to be quite stabilized after 200 hours (if they have ever shown signs of converging).

4.3 Model Verification

Recall that we mentioned in section (4.1), the system will be an M/M/1 if we set **K** = 1. To verify our model to be valid, we altered the arrival rate **λ** to be 15/hr, then we ran the simulation under eight scenarios by using **μ** = {1,3,5,10,15,20,25,30}. If our model functions properly, the queue length and delay should be able to reach a steady state when μ is greater than 15 and explode otherwise.

The simulation results are shown below. Figure 4.1 are plots of the delay in processing transactions and Figure 4.2 are plots of the queue length. The lines in the plots on the left represent changes in delay and queue length under all eight scenarios. The plots on the right only show the lines corresponding to **μ** = {15,20,25,30}, such that we can observe the behaviors of the metrics around the threshold value (**μ** = 15, i.e. **μ*K** = **λ**) more closely.

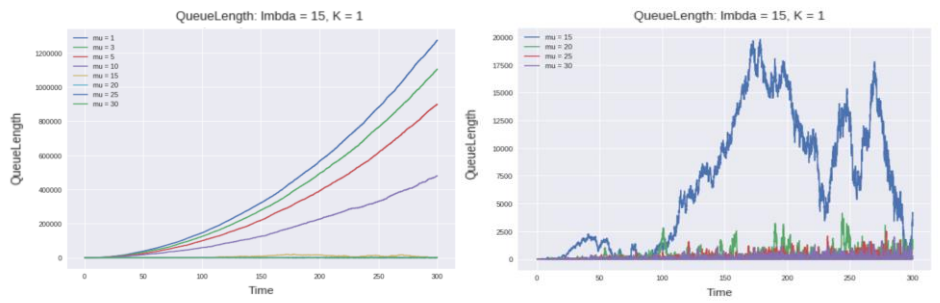


Figure 4.1: *queue length, given λ = 15/hr, K = 1*

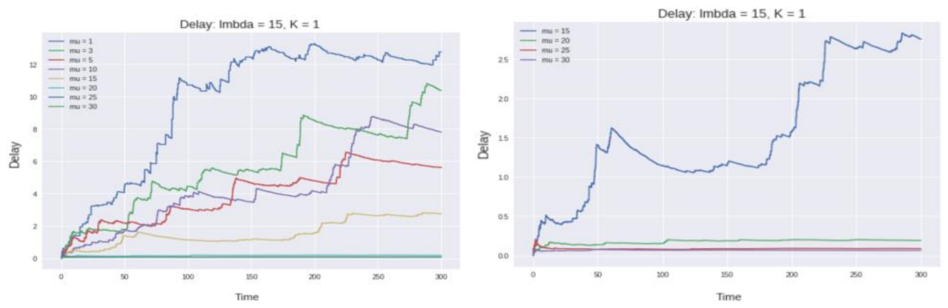


Figure 4.2: *delay, given λ = 15/hr, K = 1*

As shown in Figure 4.1, we could clearly see that the system explodes when μ = {1,3,5,10,15}. The pattern may not be obvious in the left plot of Figure 4.2; yet if we zoom in, we can find that the queue length is highly unstable when **μ** = 15, comparing to the situations when **μ** > 15. This is consistent with our expectation. Hence, we proceed to model analysis assuming our simulation model is valid.

5. Data Analysis

This section will be divided into five parts. The first part will be the model analysis of the results we obtained for the five metrics. In the second part, we will summarize our observations about and the threshold values of **μ** and **K** required for steady-state. In the third part, we will analyze the optimal pairs of K and μ based on our data analysis, and replicate the optimal scenarios such that we can better understand the distributions of each metric under steady-state. In the fourth and fifth part, we will dig deeper into the relationship between the transaction fees and delay, as well as the minimum number of active miners required for the system to be profitable.

5.1 Metics Model Analysis

a. Queue length of transactions

As we mentioned before, intuitively, we would expect to see the queue length converges to a stable value when **μ * K** > 120 and explodes otherwise. We used 64 pairs of **μ** and **K** to run the simulation, and the table below represents the data analysis result. Given **μ** and **K**, the values in the table are the average queue lengths measured after the system has been simulated for 250 hours. The boxes in pink are the scenarios when the product of **μ** and **K** is greater than 120. And the boxes in yellow represent when **μ** * **K** is smaller than 120.

Looking at the table, we may conclude that our intuition is correct. The bold red values highlight the situations when the system apparently explode. We can observe that these values fall exactly into the boxes in yellow.

K \ μ	1	3	5	10	15	20	25	30
1	**29658.97**	**29135.93**	**29031.98**	**27530.13**	**25903.33**	**25043.57**	**23719.72**	**22337.13**
3	**29458.71**	**27615.23**	**26475.11**	**22584.50**	**18475.29**	**15191.44**	**10833.05**	**7386.56**
5	**28634.05**	**26329.98**	**24034.53**	**17757.58**	**11521.71**	**4666.41**	61.31	10.72
10	**27319.32**	**22450.74**	**16846.27**	**5456.33**	16.52	8.38	6.11	5.29
15	**26255.32**	**19421.69**	**10416.87**	**31.41**	9.11	7.68	7.36	6.94
20	**24919.41**	**14209.24**	**5768.37**	13.52	10.90	9.83	9.41	9.30
25	**24235.47**	**11172.13**	102.91	16.15	12.22	11.91	11.83	11.70
30	**21991.04**	**7219.35**	48.48	15.53	14.61	14.26	14.24	14.16

Table 5.1: *Queue Length - μ = 15,20,25,30 and K=1,3,5,10,15,20,25,30 blocks/hr*

However, this table does not prove when the system actually reaches steady-state, and there are also ambiguous scenarios in the table. For instance, when **μ** = 5 blocks/hr and **K** = 25, the average queue length is around 103 transactions. Though this value is much smaller than the other values in bold, it is greater than 20 times the minimum value we have in the table. Therefore, we plotted the changes in queue length under all 64 situations and demonstrated two conditions in Figure 5.1 and 5.2 below.

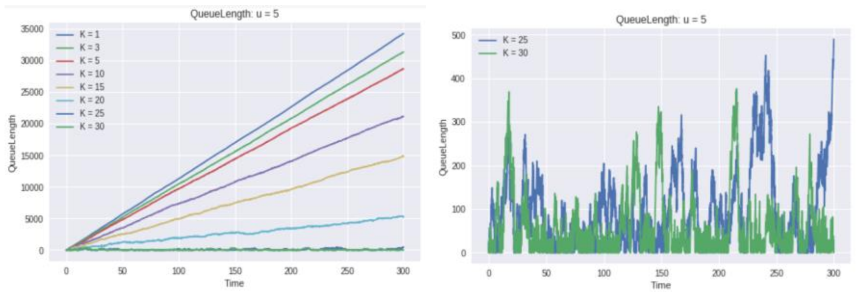


Figure 5.1: *Queue Length - μ = 5 blocks/hr*

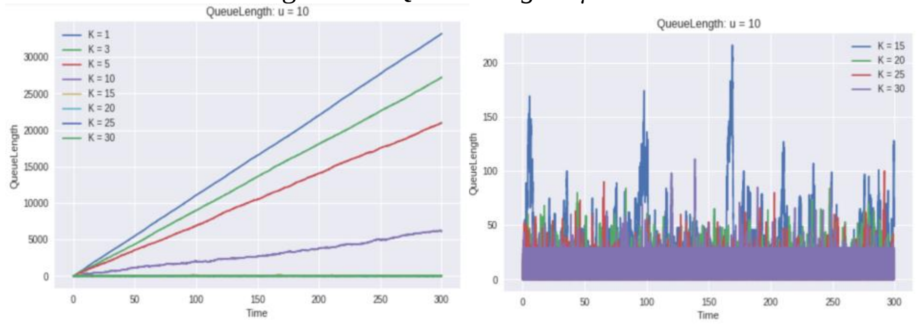


Figure 5.2: *Queue Length - μ = 10 blocks/hr*

As shown in the left plot of Figure 5.1, given that μ = 5 blocks/hr, the system explodes when **K** = {1,3,5,10,15, 20}. If we zoomed in the plot, we could see that the system is still quite unstable when **K** = 25, but it definitely enters its steady state when **K** = 30. And according to Figure 5.2, we could find that the system apparently explodes when **K** = {1,3,5,10}, given **μ** = 10 blocks/hr. It remains quite unstable when **K** = 15, and guarantees to enter its steady state when **K** equals to or is greater than 20.

b. Delay

In the basic model, given a fixed **λ** = 120/hr and a uniformly distributed transaction amount, whether delay could reach a steady state is determined by the relationship between transaction arrival rate λ, and the product of **μ** and **K** -- the utilization rate. Generally, the delay in the system will fail to converge to steady-state if the arrival rate is bigger than leaving rate (i.e. 120 > **μ * K**). As we can see in the following pictures, if we fix **μ** and increase **K** from a small value, then delay time will first explode; but when **μ * K**>120, the delay will reach a steady state.

Besides, from the table below, it shows that there is a trade-off between K and delay. As **K** increase from 1 to 30, the delay time first decreases and then increases. This is because the miners need to wait for **K** transactions before starting mining. A very large K provide more capacity but lead to a longer waiting time. In order to find the optimal pair of μ and K, we need to try several times on **K** (more detail given in optimal scenario analysis). In general, **μ * K**>120 and a reasonable K are two necessary conditions for an optimal pair of (**μ**, **K**).

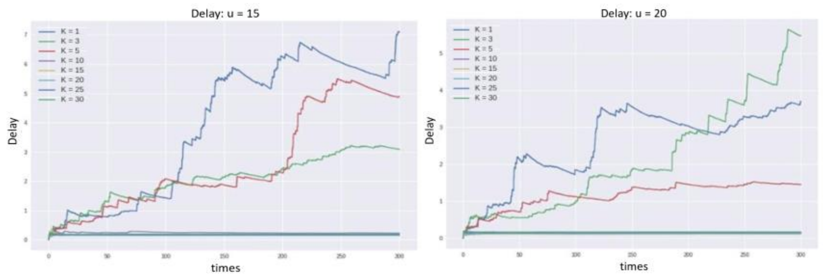


Figure 5.3: *Delay - (left) μ = 15 blocks/hr, (right) μ = 20 blocks/hr*

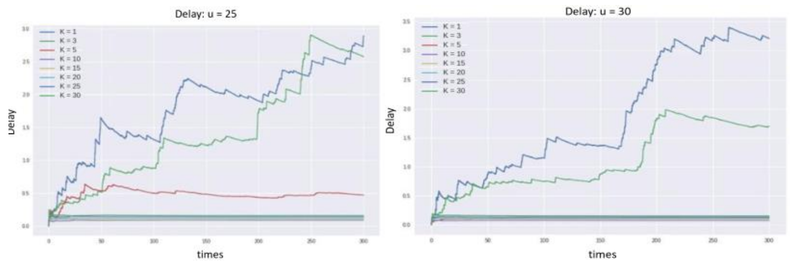


Figure 5.4: *Delay - (left) μ = 25 blocks/hr, (right) μ = 30 blocks/hr*

K \ μ	1	3	5	10	15	20	25	30
1	80.90	12.59	12.37	9.08	6.12	3.20	2.38	3.16
3	10.32	10.41	4.66	2.56	2.93	3.96	2.40	1.81
5	8.59	6.47	5.01	4.93	4.79	1.45	0.47	0.12
10	12.89	7.82	5.29	3.28	0.23	0.11	0.09	0.08
15	6.55	5.29	3.13	0.34	0.16	0.12	0.10	0.09
20	13.43	4.25	3.32	0.23	0.15	0.13	0.12	0.11
25	8.29	4.51	1.17	0.23	0.17	0.15	0.14	0.13
30	6.92	6.03	0.68	0.24	0.19	0.17	0.16	0.15

Table 5.2: *Delay - μ = 15,20,25,30 and K=1,3,5,10,15,20,25,30 blocks/hr*

c. Rate of Transactions

The rate of transactions refers to the number of transactions that successfully finished mining added to the blockchain per unit time. Figure 5.5 shows the average rates of transactions recorded from the simulation, given **μ** = 1 (left) and **μ** = 30 (right). We can observe that the rate of transactions added to the blockchain will always converge after the system operates long enough, regardless of whether the system can enter a steady-state in terms of other metrics. Generally speaking, this rate tends to rise as **μ** and **K** increase since the capacity of the system increases.

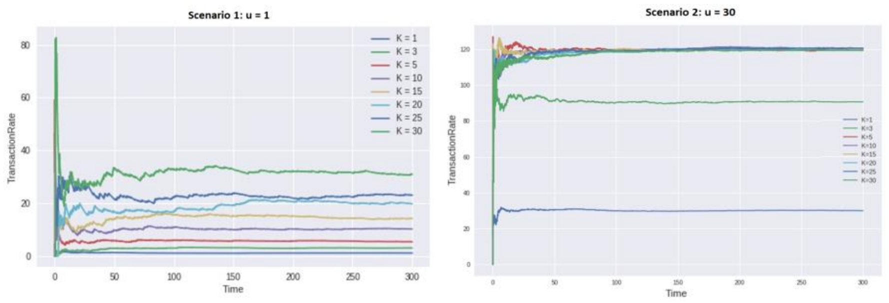


Figure 5.5: *Rate of Transaction - (left) μ = 1 blocks/hr, (right) μ = 30 blocks/hr*

Another interesting finding from the left plot of Figure 5.5 is that, when the values of **μ** and **K** are higher than a certain level, the average rate of transactions added to the blockchain will no longer increase. This finding can be confirmed in Table 5.3. The table shows that the rate of transactions will typically increase when the product of **μ** and **K** increases, then it will be stable once it reaches around 120/hr (highlighted in pink).

K \ μ	1	3	5	10	15	20	25	30
1	1.07	2.98	4.91	10.00	15.07	20.30	25.06	30.16
3	3.01	9.56	14.64	29.22	45.51	58.37	76.52	90.73
5	5.58	14.84	24.24	49.87	74.95	99.23	118.32	**119.46**
10	10.19	29.64	52.88	98.38	**119.82**	**121.16**	**119.86**	**120.72**
15	14.38	41.89	78.69	**119.66**	**119.16**	**120.37**	**120.08**	**120.12**
20	20.25	62.24	97.43	**119.01**	**119.33**	**120.21**	**120.92**	**120.00**
25	22.64	75.61	**119.35**	**119.64**	**120.84**	**119.75**	**119.63**	**120.43**
30	31.57	90.95	**120.06**	**119.64**	**119.77**	**119.95**	**120.02**	**119.38**

Table 5.3: *Rate of Transaction - μ = 15,20,25,30 and K=1,3,5,10,15,20,25,30 blocks/hr*

This finding meets our expectation since 120/hr is exactly our presumed arrival rate of the transactions. This can be verified by the little's law which states that in the long run, the output rate of the system should equal to the input rate. Hence, given the current setting of the system, it is possible to achieve a stable rate of adding transactions, if the team chooses the appropriate values of **μ** and **K**.

d. Rate of Transaction (BRCs) Amount

The rate of transaction amount (BRCs) is defined as the throughput of the system in terms of the total BRCs per time. The plots below clearly show that for a fixed **μ**, when the utilization (**μ*****K**) is smaller than the arrival rate, increasing **K** could increase the rate of BRCs. When the utilization is greater than the arrival rate, no matter how we change **K** value, the rates of BRCs tend to stabilize around 1800/hr. This result can also be verified by theory:

$$Rate\ of\ Fee\ = \frac{K \times E(Transaction\ Amount)}{E(Delay)}$$

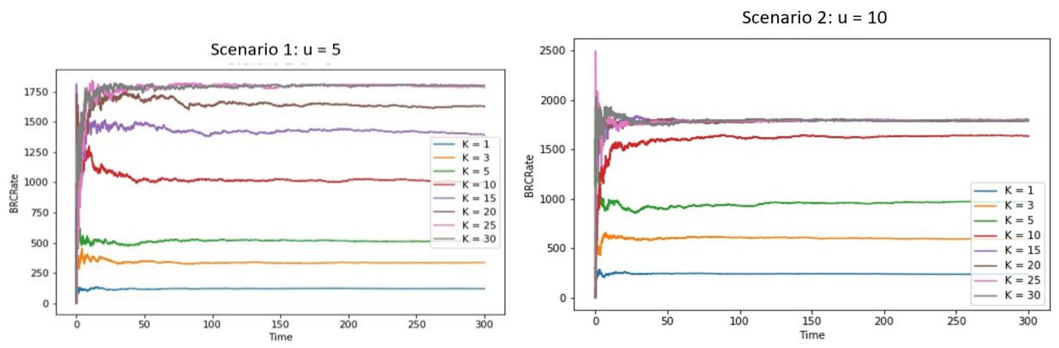


Figure 5.6: *Rate of BRC - (left) μ = 5 blocks/hr, (right) μ = 10 blocks/hr*

e. Rate of Transaction Fee

In the scenarios that delay could reach a steady state (**μ * K**>120), the rate of transaction fee will be stuck at a fixed value. This is reasonable, because if the delay reaches a stable level, every time a block is added to the blockchain, the amount of transactions are uniformly distributed, so in the long run (steady state), the amount is just around the fixed number. thereby the rate of fee which is the percentage of transaction amount will also around a fixed amount. The calculation procedure is given below:

$$Rate\ of\ Fee\ = \frac{K \times E(Transaction\ Fee)}{E(Delay)} = \frac{K \times E(\%\ \times\ Transaction\ Amount)}{E(Delay)}$$

Since percentage (%) and transaction amounts are independent:

$$Rate\ of\ Fee = \frac{K \times E(\%) \times E((Transaction.Amount)}{E(Delay)}$$

$$= \frac{K \times ((1\% + 2\%) \times 0.5) \times ((5 + 25) \times 0.5)}{E(Delay)} = \frac{K \times 0.15 \times 15}{E(Delay)}$$

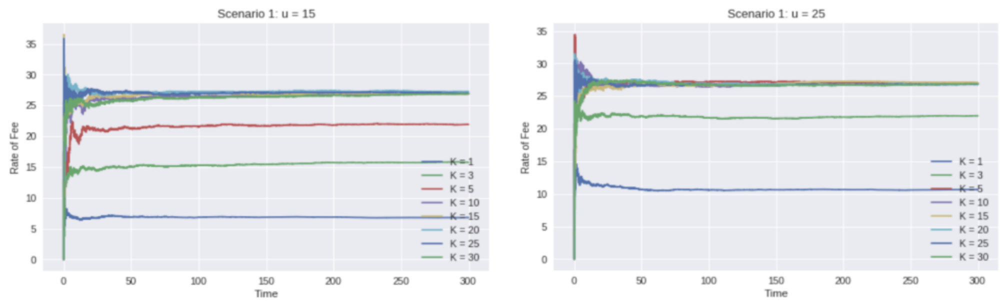


Figure 5.5: *Rate of Fee - μ = 15,25 and K=1,3,5,10,15,20,25,30 blocks/hr (continued on next page)*

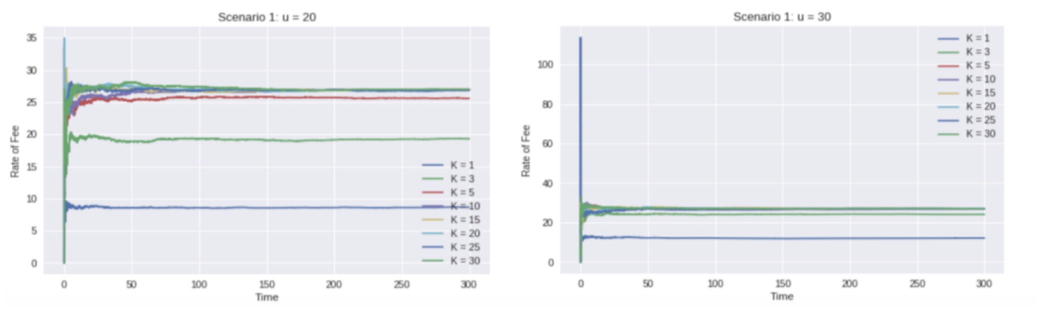


Figure 5.5: *Rate of Fee - μ = 20,30 and K=1,3,5,10,15,20,25,30 blocks/hr*

5.2 Threshold Scenario Analysis

We may not have a clearly defined formula describing the relationships between the threshold values of **λ*, μ*** and **K***, which determine whether the system can have a steady state or not. Yet, given that **λ** is constant 120/hr, we can have a glimpse of the relationship between the threshold **μ*** and **K*** with the data we collected for the model analysis.

We selected the threshold values from the 64 pairs of (**μ** , **K**). We decided the values by observing the figures of queue length and delay, searching for the pairs of (**μ , K**) that first make the system changes from a unstable (exploding) process to a stable one (with steady-state). The detailed plots can be found in the Appendix. Figure 5.6 shows the relationship between **μ*** and **K*,** based on our observations from the model analysis.

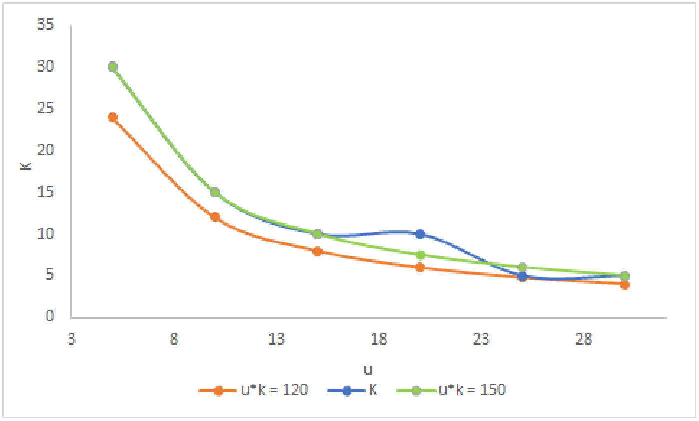


Figure 5.6: threshold *μ* vs. K**

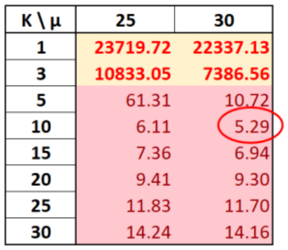
According to the plot, it is reasonable to deduce that the product of **μ** and **K** is still a good indicator of whether the system can have a steady state, but the product should be greater than a level that is higher than the arrival rate of 120/hr. In Figure 5.6, the blue line represents our observed **μ*** and **K***, whereas the orange line and the green line each represents where the product of **K** and **μ** is 120 and 150. We can tell from the plot that the blue line (plotted based on observation) is slightly higher than the orange line, especially when **μ** is small; instead, it appears to be closer to the green line, where **μ*K** = 150. We will still need more simulation data to verify our observation.

5.3 Optimal Scenario Analysis

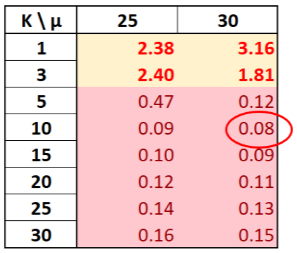
a) Result from Data Analysis

Based on the analysis of queue length and delay, we observe that the system functions most efficiently (shortest average queue length and delay) when **μ** = 30 blocks/hr and **K** = 10.

We can observe from Table 5.1 and 5.2 (in section 5.1) that, when the system reaches its steady-state, the average queue length and delay decreases as μ increases. However, there is a turning point $\widehat{K}$ for each μ. When K is smaller than $\widehat{K}$, the average queue length will decrease as K increases. Yet if K is greater than $\widehat{K}$, the averages queue length delay will start to increase as K becomes larger. It occurs because, when K becomes too large, transactions tend to spend more time waiting in the queue to for enough transactions to arrive so that the miners can start to mine the block. Therefore, the optimal μ should be the maximum μ that we could have, i.e. **μ** = 30 blocks/hr. The corresponding $\boldsymbol{\widehat{K}}$ should be the optimal K value for the system.



K \ μ	25	30
1	23719.72	22337.13
3	10833.05	7386.56
5	61.31	10.72
10	6.11	5.29
15	7.36	6.94
20	9.41	9.30
25	11.83	11.70
30	14.24	14.16



K \ μ	25	30
1	2.38	3.16
3	2.40	1.81
5	0.47	0.12
10	0.09	0.08
15	0.10	0.09
20	0.12	0.11
25	0.14	0.13
30	0.16	0.15

Table 5.4: *Queue Length (Left), Delay (Right)*

The scenario when the queue length and delay reach their minimum is circled out in Table 5.4. The optimal scenarios (μ = 30 and K = 10) measured by the two metrics are consistent. Under the optimal scenario, we obtained a minimum average queue length of 5.29 transactions and a minimum average delay of 0.08 hours.

b) Optimal Scenario Recommendation

Though our selections of μ and K are quite sparse from the model analysis, the observations still gave us a good sense of what the optimal parameters (μ and K) should be like. In order to find a more accurate optimal scenario for the system, we repeated our simulation process by fixing μ to 30 blocks/hr and tried a narrower range of K that centered around K = 10.

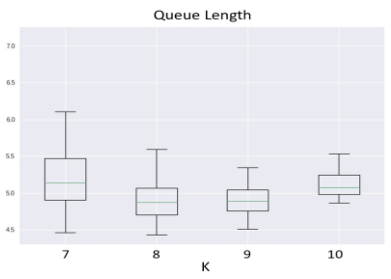
Queue Length	
K \ μ	30
5	9.582
6	6.660
7	5.151
8	4.809
9	4.915
10	5.196
11	5.433
12	5.563

Delay	
K \ μ	30
5	0.114
6	0.082
7	0.077
8	0.074
9	0.075
10	0.074
11	0.080
12	0.081

Table 5.5: *Queue Length (Left), Delay (Right)*

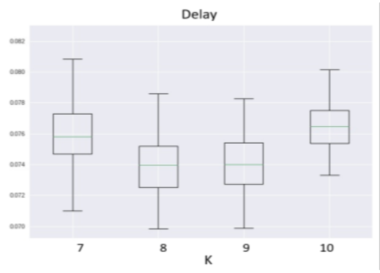
Table 5.5 demonstrates the average steady-state queue length and delay of the system, given different values of K. It appears the system will operate most efficiently when **K** = {7,8,9,10}. To select the optimal solution, we replicated the simulation 100 times under each of the four scenarios. The distributions of queue length and delay are shown in Figure 5.7/5.8 and Table 5.6/5.7. If we only focus on the median values (the green line), we can observe that the system tends to have the shortest queue length when **K** = 8. The scenarios when **K** = 8 and **K** = 9 have equally good performance in terms of the average delay. However, if we focus on the variance of the data, as shown by the height of the boxes in the figures and the 95% C.I. in the tables below, we can find that the efficiency of the system is more stable when **K** = 9, especially if we use the queue length to measure the efficiency.

Therefore, given that the transactions arrive at a stable rate of 120/hr, we recommend the BRC team to use **μ** = 30 blocks/hr and **K** = 9 to maximize the efficiency of its system.



Queue Length					
K	Min	Median	Mean	95% C.I.	Max
7	4.461	5.135	5.232	(4.284 , 6.181)	7.114
8	4.434	4.872	4.921	(4.342 , 5.499)	5.707
9	4.511	4.891	4.917	(4.474 , 5.360)	5.953
10	4.862	5.075	5.114	(4.785 , 5.444)	5.533

Figure 5.7, Table 5.6: *distribution of queue length, given K = {7,8,9,10}*



Delay					
K	Min	Median	Mean	95% C.I.	Max
7	0.070	0.076	0.076	(0.071 , 0.081)	0.082
8	0.070	0.074	0.074	(0.070 , 0.078)	0.080
9	0.070	0.074	0.074	(0.071 , 0.078)	0.078
10	0.073	0.076	0.076	(0.074 , 0.079)	0.080

Figure 5.8, Table 5.7: *distribution of queue length(left) and delay(right) given K = {7,8,9,10}*

The distributions of the other three metrics under the four scenarios are shown in Figure 5.9. Despite some little differences in the height of the boxes (spread/variance of the data), the four scenarios behave quite similarly in terms of the number of transactions, the transaction amount and the transaction fees that added to the blockchain per unit time. The average rates of transactions all center at around 120/hr. The average rates of fee have their median at around 27/hr. And the average rates of transaction amount have their median at around 1800/hr.

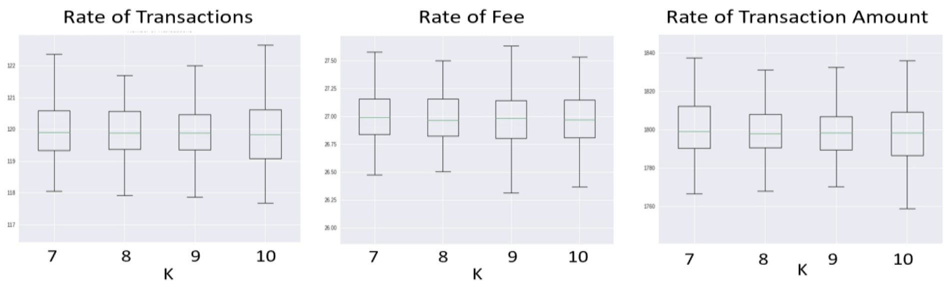


Figure 5.9: *distribution of Rate of Transactions(left), Rate of Fee (Middle), and Rate of Transaction Amount (right) given K = {7,8,9,10}*

5.4 Delay vs. Transaction Fee

To dive deeper on the economics of the system, we further examined the relationship between rate of transaction fee and delay. We tried to answer how does the delay changes based on the offered fees. To tackle the problem, we fix the transfer amount as constant and allow changing the transaction fee by changing its percentage. In particular, we choose a set of percentage rate ranging from 0.01 to 0.1 with 0.005 as increment, and classified the transactions by their transaction fee into 20 groups. Then, we run the simulation for fixed mining rate and block size (as long as the utilization is greater than the arrival rate, here we show the plot with **μ** = 5, **K** = 30), and track the average delay for each group with the different transaction fees. The results showed a negative relationship between the two metrics which is very intuitive, namely the higher transaction fee a miner provides, the shorter delay they would experience. For a fixed pair of **μ** and **K**, we can simply give the estimated transaction fee a miner should offer if he wants a particular delay time by looking at the plots. (In the appendix, we also provide the case where utilization is smaller than arrival rate. It can be verified that when the system explodes, the relation between transaction fee and average delay is no longer monotonic.)

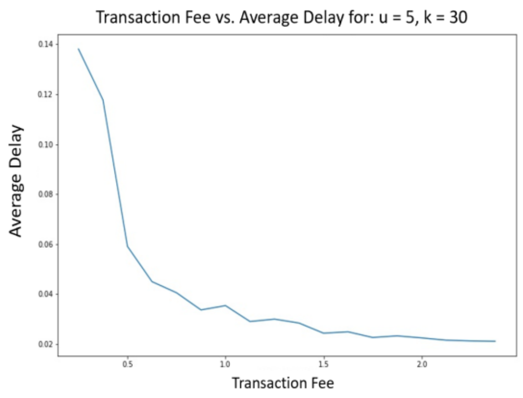


Figure 5.10: *Relation Between Transaction Fee and Average Delay, μ = 5, K = 30*

5.5 Break-Even Active Miners (Optional Analysis)

In order to analyze the economic equilibrium, we first modified the basic model to introduce mining cost. Supposing a miner's cost is **c** (BRC/hour), the system reaches equilibrium when the same cost per hour is equal to revenue per hour. Here, cost is **μ** ***c** and revenue is rate of fee. In this way, the mining only starts when the total rate of fee of **K** transactions is higher than the cost of the miners; otherwise, they will wait for the next transaction to come until the total revenue from the top **K** transactions is higher than the cost.

After considering the cost, our model results change, especially the rate of fee at the steady state. In the previous basic model, the rate of fee is around a fixed number in steady-state, no matter which pair of **μ** and **K**. However, after taking the cost **c** into consideration, different pairs of **μ** and **k** will lead to different rate of fees at steady state.

Taking **c** = 0.6 as an example, we fixed **μ** and changed **K** in a range from 5 to 15, the steady-state values of transaction fees show an upwards trend (Figure 5.11). There is also a special case when **K** = **μ** . In this scenario, the mining procedure will not start since the cost will always lower than the revenue.

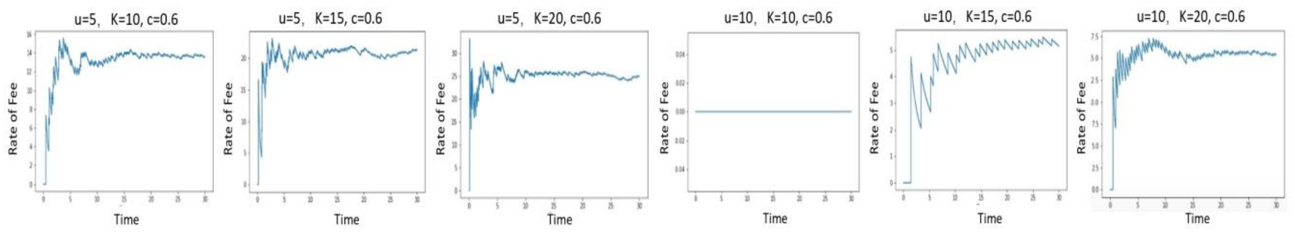


Figure 5.11: *Rate of Fee, c = 0.6, μ = {5,10}, K = {10,15,20}*

Similarly, we also tried to compare the effect of different **c** on rate of fees given the fixed pairs of **μ** and **K**. Figure 5.12 shows the situations when **c** = 0.7. Comparing Figure 5.12 to Figure 5.11, we can find that increasing **c** will cause the rate of fees to increase too. This is sensible, since the revenue will rise along with the cost if the system is in an economic equilibrium situation.

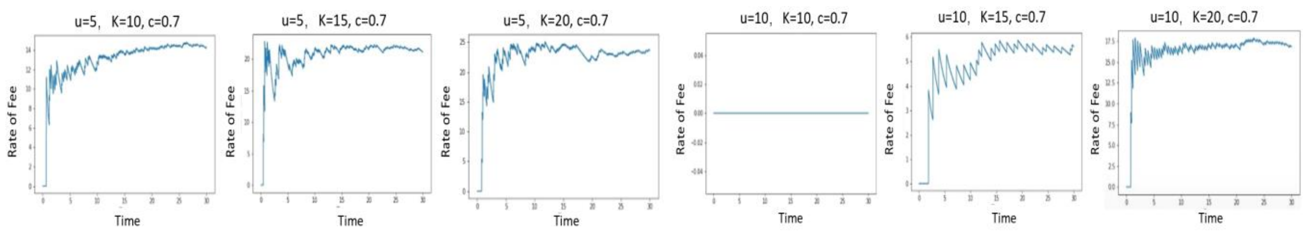


Figure 5.12: *Rate of Fee, c = 0.7, μ = {5,10}, K = {10,15,20}*

Finally, our terminal aim is to find the number of miners in the system so that the system is in economic equilibrium. Given the transaction fees in steady-state situation and the cost per hour, we could calculate that the maximum number of miners in system is $\frac{Transaction\ Fee}{c}$(make sure transaction fee > **c*N**).

6. Conclusion

In short, given that the transactions arrive at a constant rate of 120/hr, we recommend the BigRedCoins Blockchain System to set the mining rate **μ** to be 30 blocks/hr and the capacity of each block **K** to be 9 transactions. The resulted average queue length will be no more than 5 transactions, and the average delay in processing the transactions should be no longer than 5 minutes (around 0.074 hours), as shown by the tables in section 5.2 (b). In addition, the higher transaction fee a miner offers, the shorter average waiting time he will experience. At steady state, the waiting time for any miner who offers high transaction fee is bounded by the average mining service time.

7. Appendix

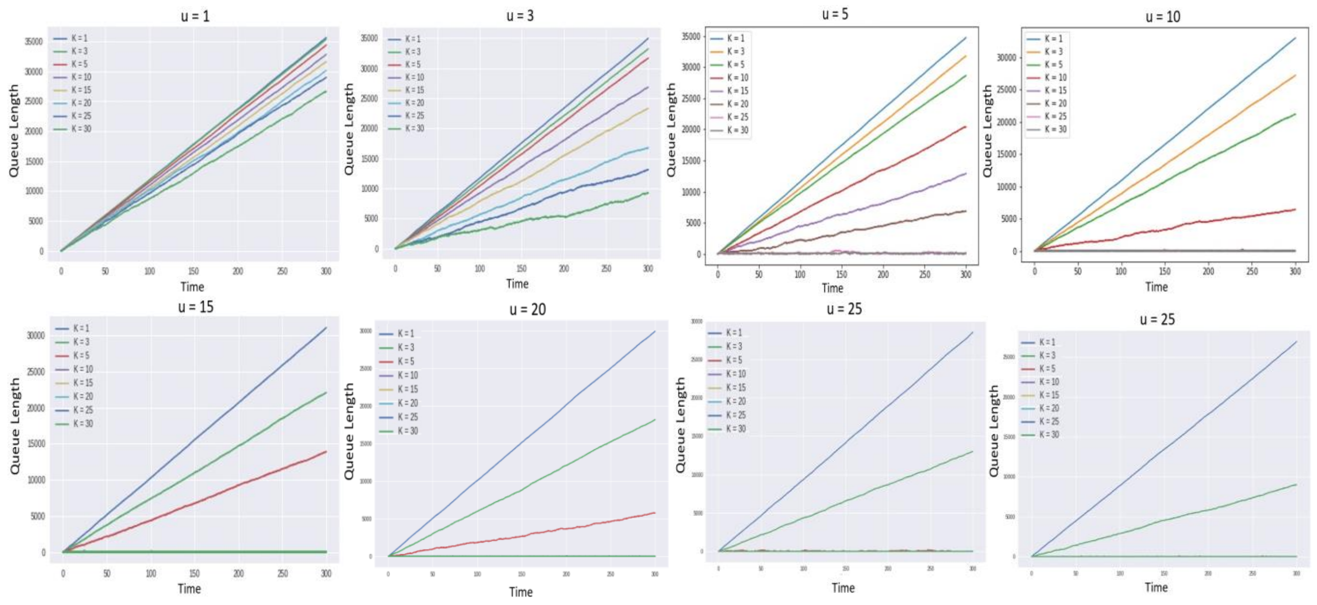
7.1 Basic Simulation Model Structure

```
Algorithm 1: OneTimeSimulate
INPUT: Arrival Rate, Miner Rate, Block Size, End Time
initiate Total Rate = Arrival Rate, Time = 0
Sample Next Event Time randomly from Exponential (scale = 1/Total Rate)
while Time < End Time do
    Set Time to Next Event Time
    Generate Event
    if Event is Transaction Arrival then
        Create a new Transaction object, with transaction amount randomly sampled from Uniform (5,25)
        and fee percentage randomly selected from (0.01,0.02)
        if No block is being mined and Queue length is greater than or equal to Block Size then
            Start Mining
    if Event is End Mining then
        Transactions leave the block
        Record the delay experienced by each transaction
        if Queue length is greater than or equal to Block Size then
            Start Mining
    Update Next Event Time
    Update Time = Time + Next Event Time
    Record the metrics
end while
```

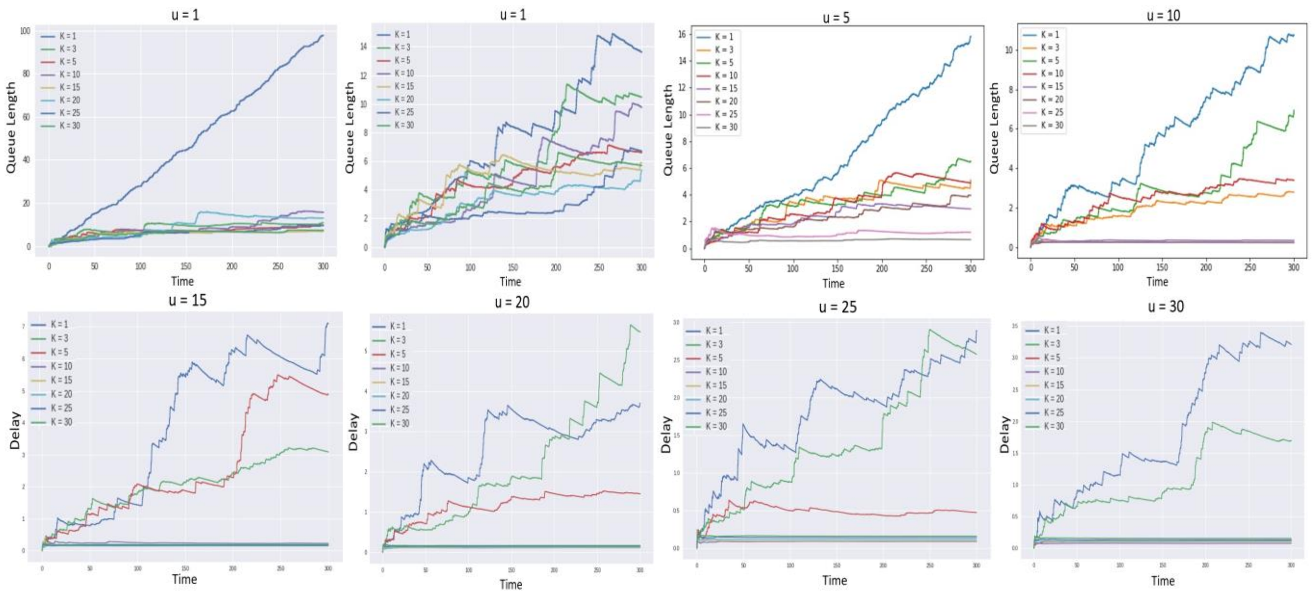
Please refer to the appended Jupyter Notebook file for detailed simulation code.

7.2 Threshold Analysis

(a) Queue Length Plots

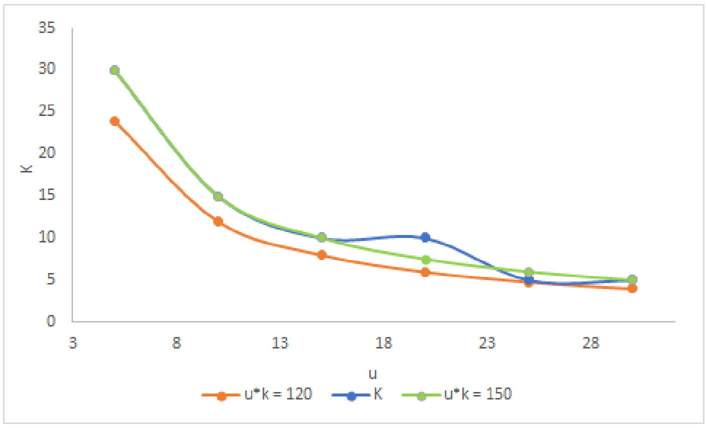


(b) Delay Plots



(c) Threshold Values

Mu	K
5	30
10	15
15	10
20	10
25	5
30	5


K
u
u*k = 120
K
u*k = 150