applied sciences

MDPI

Article

A Real-Time Chain and Variable Bulk Arrival and Variable Bulk Service (VBAVBS) Model with λ_F

Nohpill Park [1], Abhilash Kancharla [1] and Hye-Young Kim [2,*]

[1] Computer Science, Oklahoma State University, Stillwater, OK 74078, USA; npark@cs.okstate.edu (N.P.); abhilak@okstate.edu (A.K.)

[2] School of Games, Hongik University, Jochiwon-eup, Chungcheongnam-do 339-701, Korea

* Correspondence: hykim@hongik.ac.kr; Tel.: +82-44-860-2683

Received: 3 April 2020; Accepted: 22 May 2020; Published: 25 May 2020

Abstract: This paper proposes a real-time chain and a novel embedded Markovian queueing model with variable bulk arrival (VBA) and variable bulk service (VBS) in order to establish and assure a theoretical foundation to design a blockchain-based real-time system with particular interest in Ethereum. Based on the proposed model, various performances are simulated in a numerical manner in order to validate the efficacy of the model by checking good agreements with the results against intuitive and typical expectations as a baseline. A demo of the proposed real-time chain is developed in this work by modifying the open source of Ethereum Geth 1.9.11. The work in this paper will provide both a theoretical foundation to design and optimize the performances of the proposed real-time chain, and ultimately address and resolve the performance bottleneck due to the conventional block-synchrony by employing an asynchrony by the real-time deadline to some extent.

Keywords: blockchain; Ethereum; real-time; queueing; mining; transaction pool

1. Introduction

Blockchain technology [1–20] is a new internet protocol to address the longtime-awaited solution for internet trust. Blockchain is a distributed ledger [4,5,7] of all the transactions that have been executed. Blockchains are distributed databases that a group of individuals controls and that store and share information. There are many different types of blockchains: Public, permissioned, private. Irrespective of the type of the blockchain, cryptography is used to allow each participant on any given network to manage the ledger in a secure way without the need for a central authority to enforce the results. The blocks are appended successfully to the blockchain, and more importantly, a block once appended cannot be removed from the blockchain. This makes the blockchain able to create trust in digital data. Blockchains basically eliminate the need for a third-party intermediary. When data is permanent and reliable in a digital format, one can transact business online in ways that, in the past, were only possible offline or would take a lot of time to process online.

The motivation and objective of this paper is to develop a real-time chain model in a theoretical manner, and to propose a novel embedded Markovian queueing model [19] of the $M^{1,n}/M^{1,n}/1$ type in order to establish a theoretical foundation to design a blockchain-based real-time system with particular interest in Ethereum, since there has been no adequate model to fulfill the above-mentioned objectives. The model assumes variable bulk arrivals of transactions in Poisson distribution, i.e., $M^{1,n}$, where n is the number of slots across all the mined transactions, and variable bulk service of transactions in exponential time, i.e., $M^{1,n}$, for posting in the current block, namely, variable bulk arrival and variable bulk service (VBAVBS) model with λ_F, where λ_F is a random variable employed specifically to address the stringent real-time deadline requirement. The primary performance measurements of interest are the average number of slots no matter how many transactions are mined under the assumption

of maximum number of slots per block as specified by n; the average waiting time per slot; and the throughput in terms of the average number of slots to be processed per time. The variable bulk arrival rate is assumed to be linearly proportional to the size of the transactions in a multiple of λ per slot, with success to arrive within the deadline, or some of the transactions with λ_F to fail to meet the real-time deadline. The variable bulk service is assumed to take place when the number of slots in the mined transactions reaches at every possible number i, where $0 \leq i \leq n$, i.e., a bulk processing of multiple transactions in multiple slots between 0 and n, inclusive, for posting in a block. Note that the real-time deadline requirement is considered under the underlying assumption that the size of each block is adaptive, in other words, varying block by block, and the proposed VBAVBS model with λ_F normalizes the probability for each size of the block throughout.

The paper is organized as follows. In the following section, preliminaries and design-variables for the proposed real-time chain and its performance modeling and analysis are reviewed, and some related works on blockchain dependability [9,11] modeling and analysis are introduced; then, the proposed real-time chain and its analytical model are derived and captured in the context of a queueing system; a section follows in order to demonstrate numerical simulations versus various blockchain-related parameters [10], and results are shown; finally, conclusions are drawn and a discussion completes the last section.

2. Preliminaries and Literature Review

A few key concepts and components of blockchain are listed as follows [4,5,7]. Public Key: Every node/user in the blockchain is assigned a public/private key. A transaction to be made on the blockchain has to be signed by a private key, while the public key is always visible to everyone on the blockchain; transaction: A transaction contains the details of the both the public keys (in case of smart contract [16]—contract address), a hash of the previous block, the current block number and the amount of crypto currency being transferred to, along with the gas needed to execute the transaction; block is a list of transactions recorded into the distributed ledger over time. The transactions are grouped together and put in the block which are then mined to be added to the blockchain. The immutability of the blockchain assures that the transaction once recorded in a block cannot be deleted or undone; chain: All the blocks have a unique block number. The block number is incremented by 1 for every block that is added to the chain. The hash in blockchain is created from the data that was in the previous block. The previous block hash will also be added to the current block details. Thereby, any change in the previous block completely alters the entre hash which would invalidate the blockchain; crypto currency: Ether is the crypto currency in Ethereum blockchain. Any transaction to be posted on the blockchain utilizes the respective crypto currency of the blockchain; gas: The amount of ether needed to post a transaction in Ethereum blockchain. The current blockchain is coded in such a way that the transaction with highest gas fees are given higher priority to be included in the block. This proposed real-time details out different algorithms that can be used to order transactions differently in the block; miner: Nodes that participate in creating a new block are called miners. Miners are responsible for adding blocks to the blockchain. The blocks are created based on the number of transactions present in the txnpool and other characteristics like gas fee, block size, arrival time; transaction pool: Transaction pool is an array datatype of transactions containing the pointers to the transactions that are not yet posted to the block. Pending transactions are the transactions that are not yet posted to the blockchain and these pending transactions reside in the transaction pool [8]. The different ordering of the transaction pool creates different mining [14,15] algorithms [13]; mining: The central process of blockchain-based computing to establish a trust among the nodes in the network connected in a P2P manner. Miners compete for the next new hash code which requires a computationally intensive process and is highly costly. In this context, there have been efforts made extensively to mitigate or eliminate the mining process.

There have been reports on various, yet critical, performance and dependability problems in [17–20], where extensive research has been conducted on theoretical designs of a few blockchain-based solutions in order to establish a theoretical yet substantial foundation. As the ultimate quality of crypto computing

will be determined by its likelihood to be performed as commanded or desired— referred to as the dependability—those theoretical models emphasized and are centered around the dependability of each of those crypto solutions to accommodate such capabilities as the on/off-balanced crypto computing [12,21], the real-time computing [20], the slim-computing [17] and the hybrid computing [18]. A theoretical study on performance is of ultimate interest to identify a theoretical intersection versus the dependability, which is the ultimate objective of the proposed variable bulk arrival and variable bulk service (VBAVBS) model with λ_F in the context of the proposed real-time chain.

3. Proposed Variable Bulk Arrival and Variable Bulk Service (VBAVBS) Model with λ_F for Real-Time Chain

In the proposed real-time chain model, an embedded Markovian single-server exponential queueing system (i.e., $M^{1,n}/M^{1,n}/1$) is considered without loss of generality, and the server (the server is the equivalent of the group of miners to select the transactions to be posted) serves the entire batch of customers (the customers are the equivalent of the transactions to be posted in the block) in the queue (a queue is the equivalent of a block to be mined and posted) all at once, at the same time. Whenever the server completes a service (a service is the equivalent of a process of posting a block), it then purges the queue (i.e., the equivalence of posting a block) and then serves the influx of new incoming customers. Note that it is assumed that the service takes place within a certain amount of time, yet no transaction is assumed to arrive in the meantime. However, note that it is not unlikely to have new customer arrive if a significant amount of service time is assumed, from a practical point of consideration. It is assumed that the service time is exponential at $\frac{1}{\mu}$ when the server is serving the entire queue of any size between 0 and n, inclusively and equivalently, posting and purging the entire queue. Without loss of generality, it is assumed that customers arrive at an exponential rate of λ successfully within the deadline yet, at the rate of λ_F, the customers are assumed to fail to arrive within a specific deadline to meet the real-time requirement. The underlying queueing process is assumed to take place with variable-sized slots and the status of the queue is determined by the number of slots of any size in the current block.

Based on the assumptions above, the proposed VBAVBS model with λ_F also employs an embedded Markovian queueing model like the VBASBS (Variable Bulk Arrival And Static Bulk Service) in [19], and it defines the states as expressed in terms of the number of slots assigned to a block, and it traces the normalized number of slots allocated for the transactions in steady state rather than the number of transactions whose size varies in the number of slots.

- P_0: The state in which no transaction (i.e., no slot) has arrived in the queue as of yet for the posting in the block, currently [19].
- P_n: The state in which there are n number of slots (i.e., the capacity of the queue, equivalently, the maximum number of slots set and voted by the miners or voters) arrived in the queue for the posting in the block, currently [19].
- P_i: The state in which there are i number of slots (where $0 < i < n$) arrived in the queue for the posting in the block, currently [19].

The random variables employed to express the state transition rates are specified as follows.

- λ: The rate for a slot of a transaction to arrive successfully within the real-time deadline requirement, and the rate for a transaction to arrive is determined by the number of slots allocated for the transaction in a prorated manner such that a transaction with a size of j number of slots arrives at the rate of $j\lambda$, without loss of generality and practicality as well.
- λ_F: The rate for a slot of a transaction to arrive unsuccessfully past the real-time deadline requirement, and at the rate, the state will self-loop without making any state transition. This rate is the unique one to distinguish VBAVBS from VBASBS in which there was no real-time deadline requirement taken into consideration.

- μ: The rate for the slots of the transactions in the entire queue to be posted and purged. Notice that this is a single and unique state transition 1, 2 and 3.

The balance equations for VBAVBS with λ_F are shown in the following Equations.

$$(\lambda + 2\lambda + 3\lambda \ + \cdots + \ n\lambda + \lambda_F)P_0 = \frac{\mu}{n}P_1 + \frac{\mu}{n-1}P_2 + \frac{\mu}{n-2}P_3 + \ \cdots \frac{\mu}{n-(n-2)}P_{n-1} + \mu P_n \quad (1)$$

$$\left(\frac{\lambda(n)(n-1)}{2} + \lambda_F\right)P_0 = \mu\left(\frac{1}{n}P_1 + \frac{1}{n-1}P_2 + \frac{1}{n-2}P_3 + \ \cdots \frac{1}{n-(n-2)}P_{n-1} + P_n\right) \quad (2)$$

and

$$P_0 + P_1 + P_2 + \ldots + P_n = 1 \quad (3)$$

Appendix A shows the detailed solving of balance equations. Equations (4) and (5 describe the generalized value of P_i.

$$P_i = \ q_i^{-1}P_0\left[\sum_{j=1}^{i} j\left[\sum_{k=1}^{i-1}\left[\prod_{l=1}^{k-1} q_l^{-1}\right]\right]k\right] \quad (4)$$

where,

$$q_i^{-1} = \left(\frac{\lambda(n-i)(n-i+1)}{2} + \lambda_F + \frac{\mu}{n-i+1}\right)^{-1} \quad (5)$$

In the following, a few baseline performance measurements of primary interests in VBAVBS with λ_F are shown.

- L_Q: The average number of customers (i.e., equivalently the average number of transactions) in the queue (i.e., the block currently being mined) [19] with the new P_i

$$L_Q = \sum_{i=0}^{n} iP_i \quad (6)$$

- W_Q: The average amount of time a customer (i.e., equivalently, a transaction) in the queue (i.e., the block currently being mined) [19].

$$W_Q = \frac{L_Q}{\lambda} \quad (7)$$

- W: The average amount of time a customer (i.e., equivalently, a transaction) in the system (i.e., the transaction pool in the blockchain) [19].

$$W = W_Q + \frac{1}{\mu} \quad (8)$$

- L: The average number of customers (i.e., equivalently, the average number of transactions) in the system (i.e., the transaction pool in the blockchain) [19].

$$L = \lambda W \quad (9)$$

4. Numerical Analysis

The efficacy of the proposed VBAVBS model with λ_F is tested and verified through numerical analysis for the L_Q, W_Q, W and L versus n (i.e., size of a block), λ (i.e., successful transaction arrival rate or speed), λ_F (i.e., unsuccessful transaction arrival rate or speed) and $\frac{1}{\mu}$ (i.e., block posting time). Note that fitting the model against real data is not within the scope of the work, instead, various and extensive numerical simulations are conducted as a baseline validation.

Figure 1 plots the average number of customers in system (L) versus the number if slots (n), for various λ and a μ (at 1/15).

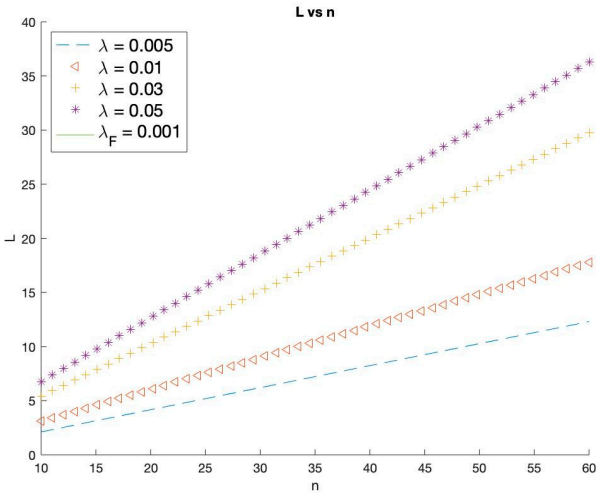

Figure 1. Average number of customers in system (L) vs number of slots (n).

L versus n, for various λ and a μ, are plotted in Figure 1. It is observed that L picks up as n increases, as expected, yet in a near linear manner. Also, it is observed that L picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

Likewise, Figure 2 shows the average number of customers in system (L) plotted against the rate of slots (μ) for various λ and an n (at 10).

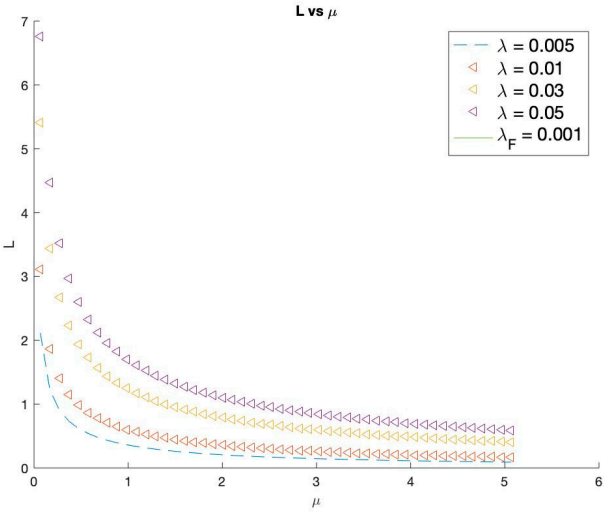

Figure 2. Average number of customers in the system (L) vs rate of slots (μ).

It is observed that L declines as μ increases, as expected. Also, it is observed that L picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

The following graph plots L_Q versus n, for various λ and a μ (at 1/15).

It is observed in Figure 3 that L_Q picks up as n increases, as expected, yet in a near linear manner. Also, it is observed that L_Q picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

Average number of customers in queue (L_Q) versus rate of slots (μ) is plotted in Figure 4, for various λ, and an n (at 10).

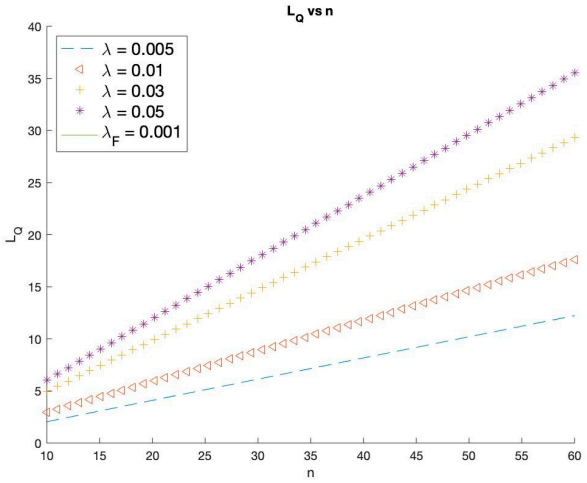

Figure 3. Average number of customers in the queue (L_Q) vs number of slots (n).

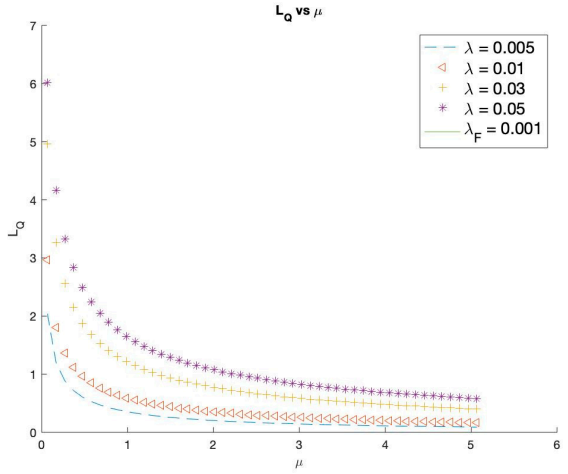

Figure 4. Average number of customers in the queue (L_Q) vs rate of slots (μ).

It is observed that L_Q declines as μ increases, as expected. Also, it is observed that L_Q picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

The following graph plots W versus n, for various λ and a μ (at 1/15).

In Figure 5, it is observed that W picks up as n increases, as expected, yet in a near linear manner. Also, it is observed that W picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

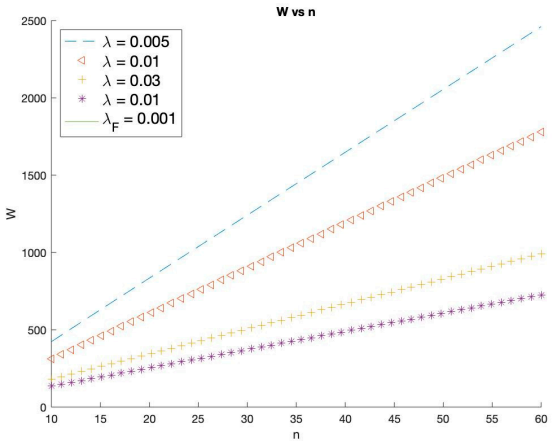

Figure 5. Average amount of time in system (W) vs number of slots (n).

The following graph plots W versus μ, for various λ, and an n (at 10).

In Figure 6, it is observed that W declines as μ increases, as expected. Also, it is observed that W picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

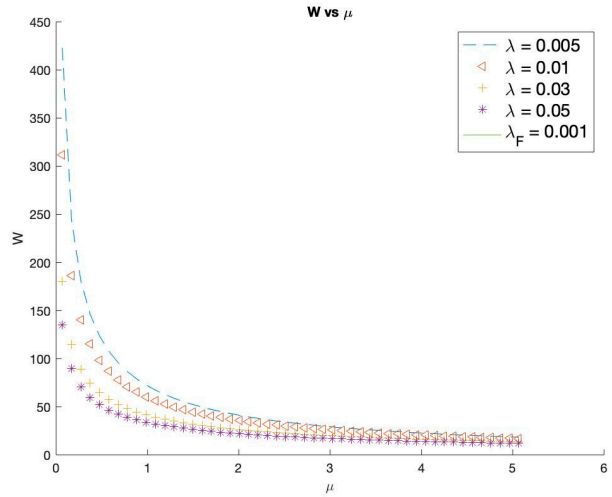

Figure 6. Average amount of time in system (W) vs rate of slots (μ).

Figure 7 plots W_Q versus n, for various λ and a μ (at 1/15).

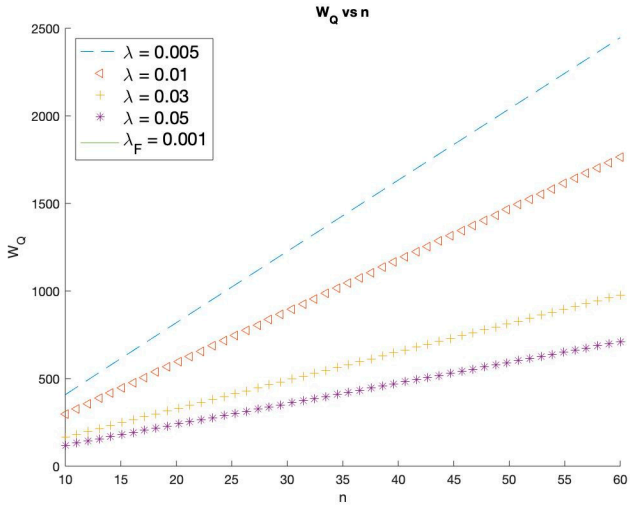

Figure 7. Average amount of time in queue (W_Q) vs number of slots (n).

It is observed that W_Q picks up as n increases, as expected, yet in a near linear manner. Also, it is observed that W_Q picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

The following graph plots W_Q versus μ, for various λ, and an n (at 10).

In Figure 8, it is observed that W_Q declines as μ increases, as expected. Also, it is observed that W_Q picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

Equation (10) shows the throughput per block in the presented model.

$$\gamma = \mu P_n = \mu \frac{\lambda}{\mu} \frac{n(n+1)}{2} P_0 = \lambda \frac{n(n+1)}{2} P_0 \tag{10}$$

The following graph plots γ versus n, for various λ and a μ (at 1/15). Note that γ is plotted versus full range of potential n values so that the γ at an n represents the normalized γ value in the full range up to n.

Figures 9 and 10 plot the throughput per block. It is observed that γ stays constant throughout, and P_n barely changes throughout.

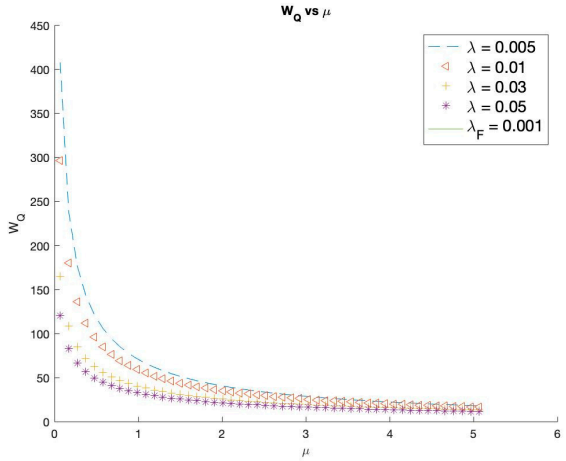

Figure 8. Average amount of time in queue (W_Q) vs rate of slots (μ).

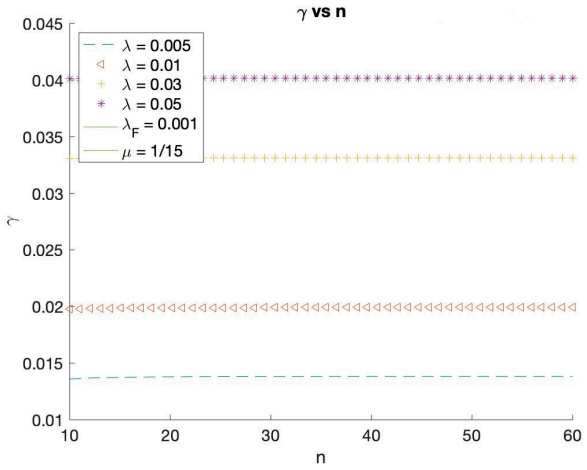

Figure 9. Throughput per block (γ) vs number of slots (n).

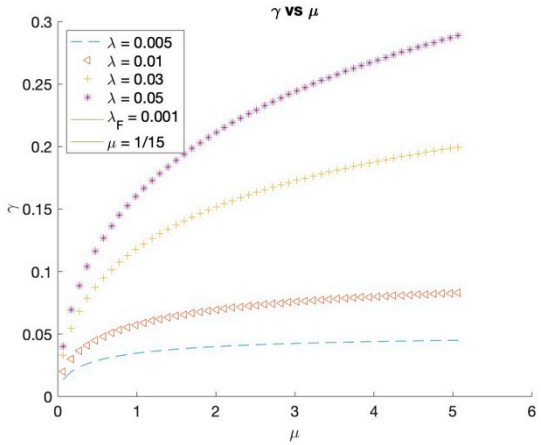

Figure 10. Throughput per block (γ) vs rate of slots (μ).

The following graph plots γ versus μ, for various λ, a μ (at 1/15) and a λ_F (at 0.001).

It is observed that γ picks up as μ increases, as expected. Also, it is observed that γ picks up as λ increases, as expected. Notice that the intervals in between the plots are quite proportionally spaced.

The following graph plots L versus λ_F, for various λ, a μ (at 1/15), and n (at 10).

It is was observed, in Figure 11, that L declines as λ_F increases such that as more numbers of transactions (or slots) fail to arrive within the required real-time deadline, less numbers of transactions will be accommodated. Also, it is observed that L picks up as λ increases, as expected. Notice that the intervals in between the plots are spaced narrower at higher λ, as expected.

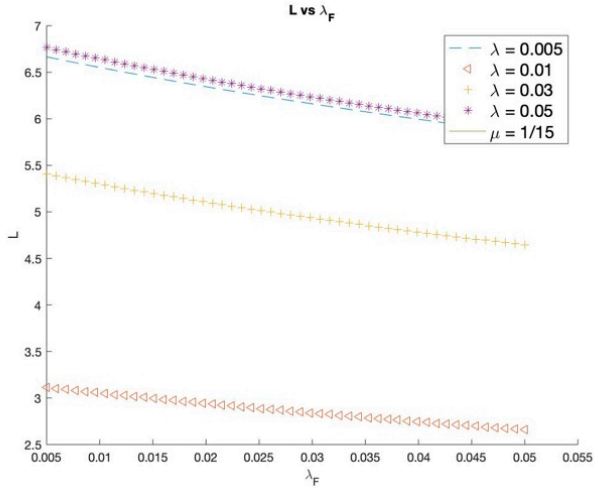

Figure 11. Average number of customers in system (L) vs rate of unsuccessful arrival (λ_F).

The following graph plots L_Q versus λ_F, for various λ, a μ (at 1/15), and n (at 10) in Figure 12.

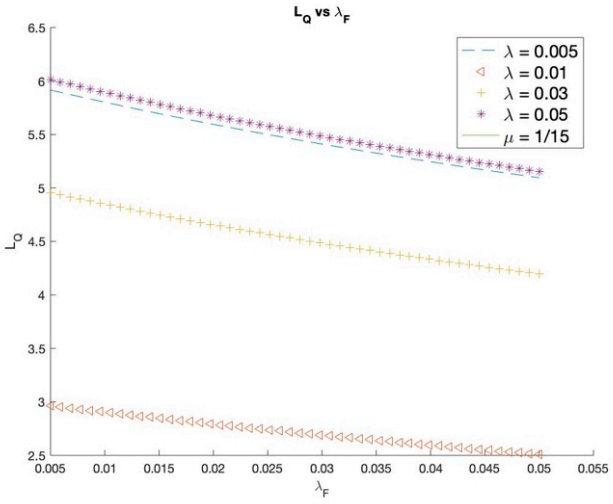

Figure 12. Average number of customers in queue (L_Q) vs rate of unsuccessful arrival (λ_F).

It is observed that L_Q declines as λ_F increases, and as a greater number of transactions (or slots) fail to arrive within the required real-time deadline, a smaller number of transactions will be accommodated. Also, it is observed that L_Q picks up as λ increases, as expected. Notice that the intervals in between the plots are spaced narrower at higher λ, as expected.

The following graph plots W versus λ_F, for various λ, a μ (at 1/15), and n (at 10).

Figures 13 and 14 plot the average amount of time in the system/queue with respect to rate of unsuccessful arrivals. It is observed that W declines as λ_F increases as a greater number of transactions (or slots) fail to arrive within the required real-time deadline, and the amount of time on average that each transaction (or slot) will be waiting in the block prior to posting is reduced. Also, it is observed

that W picks up as λ increases, as expected. Notice that the intervals in between the plots are spaced narrower at higher λ, as expected.

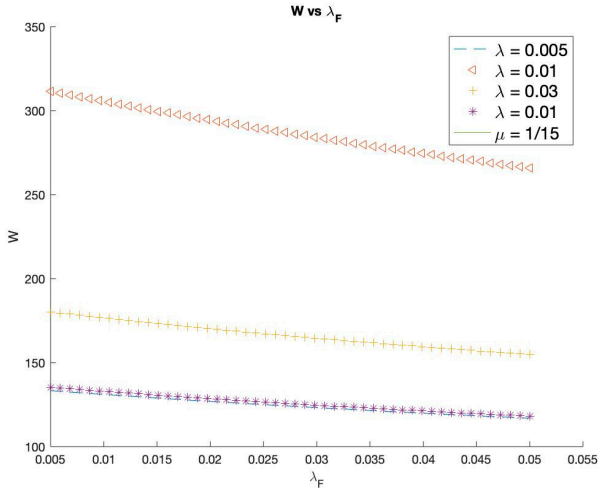

Figure 13. Average amount of time in system (W) vs rate of unsuccessful arrival (λ_F).

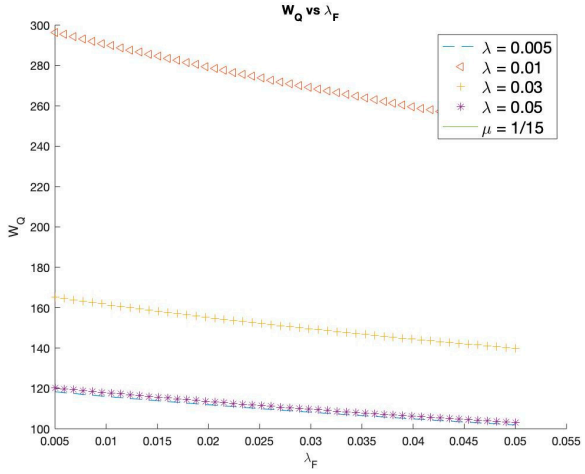

Figure 14. Average amount of time in queue (W_Q) vs rate of unsuccessful arrival (λ_F).

The following graph plots W_Q versus λ_F, for various λ, a μ (at 1/15), and n (at 10).

It is observed, in Figure 14, that W_Q declines as λ_F increases as a greater number of transactions (or slots) fail to arrive within the required real-time deadline, and the time on average is reduced in which each transaction (or slot) will be waiting in the block prior to posting. Also, it is observed that W picks up as λ increases, as expected. Notice that the intervals in between the plots are spaced narrower at higher λ, as expected.

5. A Demo: Real-time Chain

Implementation of real-time is shown in this section. Table A1 in Appendix B shows a detailed list of components used and the version numbers which were modified to create this demo. With deadline time taken into consideration, four transactions are posted at the same time but with random deadline times to simulate the priority to the deadline variable. In the standard Ethereum model, the transactions are sorted based on the price and nonce. The four transactions, if posted in the standard Ethereum model, would be included in blocks in a decreasing order based on price and nonce. In the modified real-time model, the transactions with deadline time closest to the current block are given higher priority. Figures mentioned in this section present an idea of the deadline times and block posting of the four respective transactions.

Figures 15–18 show transactions that are posted with random deadline times: 35, 36, 50 and 58. Figure 15 details a transaction that posted with deadline time variable set to 35. The transaction with a gas limit of 3,000,000 is posted in 25th block. Figure 16 shows another transaction with lower gas limit than the one shown in Figure 15. The gas limit of the transaction is 2,000,000, and the deadline time is set to 36. The contract hash and the block number details are shown below, with the transactions being posted in the 26th block.

```
INFO [05-06|23:27:58.741] transaction.go                           GasLimit=3000000
INFO [05-06|23:27:58.741] transaction.go                           IpfsFlag=true
INFO [05-06|23:27:58.741] transaction.go                           deadLineTime=35
INFO [05-06|23:27:58.745] Setting new local account                address=0x1091B529feE18aF76D3bb84431E48631A3990E6c
```

(**a**)

```
INFO [05-06|23:27:58.748] Submitted contract creation              fullhash=0xc7b3d670210d4bed64949f7d1e05a0006e11296448b4226
ae1b64303fe8fb37d contract=0xDf2e2935c5A3429B621ab940667BA50D6851d050
```

(**b**)

```
> web3.eth.getTransaction("0xc7b3d670210d4bed64949f7d1e05a0006e11296448b4226ae1b64303fe8fb37d")
{
  blockHash: "0x2a0e8b21a93adc5d5250b10573c02f0e6b5c4c743fed6f4f44604cca9b5aff60",
  blockNumber: 25,
  from: "0x1091b529fee18af76d3bb84431e48631a3990e6c",
  gas: 3000000,
```

(**c**)

Figure 15. (**a**) Transaction with deadline time of 35, (**b**) full hash of the transaction, (**c**) gas limit 3,000,000 of the same hash.

```
INFO [05-06|23:29:17.386] transaction.go                           GasLimit=2000000
INFO [05-06|23:29:17.386] transaction.go                           IpfsFlag=true
INFO [05-06|23:29:17.386] transaction.go                           deadLineTime=36
```

(**a**)

```
INFO [05-06|23:29:17.399] Submitted contract creation              fullhash=0xbae0e6ac1882bf7fb40b18e5ddd3cbca034759ee0b9ba24
225e8d17c407d308a contract=0xC98998A285FAc425c86a7209F35e2C57488a80bB
```

(**b**)

```
> web3.eth.getTransaction("0xbae0e6ac1882bf7fb40b18e5ddd3cbca034759ee0b9ba24225e8d17c407d308a")
{
  blockHash: "0xff486c15af71f395f408b21cbfde370be604d8c5bc8b7f4852bb035a6e1ad9f4",
  blockNumber: 26,
  from: "0xe68b8eecde3f994eb3cf1ddc661a4e3d3927a56f",
  gas: 2000000,
  gasPrice: 1000000000,
  hash: "0xbae0e6ac1882bf7fb40b18e5ddd3cbca034759ee0b9ba24225e8d17c407d308a",
```

(**c**)

Figure 16. (**a**) Transaction with deadline time of 36, (**b**) full hash of the transaction, (**c**) gas limit 2,000,000 of the same hash.

```
INFO [05-06|23:34:56.034] transaction.go                           GasLimit=4500000
INFO [05-06|23:34:56.035] transaction.go                           IpfsFlag=true
INFO [05-06|23:34:56.036] transaction.go                           deadLineTime=50
```

(**a**)

Figure 17. *Cont.*

```
INFO [05-06|23:34:56.037] Submitted contract creation              fullhash=0xe703512f34bf5c3b98b996a4d13552c7644fd985c7c2098
ebb6bab8f0b213fcf contract=0xD3EDD8481e3CE36F7F41915541BEf9Fe7c3750eB
```

(**b**)

```
> web3.eth.getTransaction("0xe703512f34bf5c3b98b996a4d13552c7644fd985c7c2098ebb6bab8f0b213fcf")
{
  blockHash: "0xdc9daf3a3f3bd1c028947c5ad2b2e948235273380041e38f0fe518a1fe40c266",
  blockNumber: 37,
  from: "0x9c8c462bee006d16c8b76920e09c6d465e7284f2",
  gas: 4500000,
  gasPrice: 1000000000,
  hash: "0xe703512f34bf5c3b98b996a4d13552c7644fd985c7c2098ebb6bab8f0b213fcf",
```

(**c**)

Figure 17. (**a**) Transaction with deadline time of 50, (**b**) full hash of the transaction, (**c**) gas limit 4,500,000 of the same hash.

```
INFO [05-06|23:36:51.812] transaction.go                           GasLimit=4000000
INFO [05-06|23:36:51.819] transaction.go                           IpfsFlag=true
INFO [05-06|23:36:51.819] transaction.go                           deadLineTime=58
```

(**a**)

```
INFO [05-06|23:36:51.832] Submitted contract creation              fullhash=0x512050338e04c45b24e45005b6fd0e5fb9fe71e8596d367
5a862242f376d7d70 contract=0x69B7f75E866687058fcac802975Cd1567CB4059B
```

(**b**)

```
> web3.eth.getTransaction("0x512050338e04c45b24e45005b6fd0e5fb9fe71e8596d3675a862242f376d7d70")
{
  blockHash: "0x7bb2a6407f26abce95ca1ec16b8ca689b25ba4b0596f3c3c02f3705e43722695",
  blockNumber: 39,
  from: "0xe68b8eecde3f994eb3cf1ddc661a4e3d3927a56f",
  gas: 4000000,
  gasPrice: 1000000000,
  hash: "0x512050338e04c45b24e45005b6fd0e5fb9fe71e8596d3675a862242f376d7d70",
```

(**c**)

Figure 18. (**a**) Transaction with deadline time of 58, (**b**) full hash of the transaction, (**c**) gas limit 4,000,000 of the same hash.

A transaction with a gas limit higher than 2,000,000 would be given higher priority in the standard Ethereum model. Such a transaction is posted here to highlight the difference in the priorities given to the transactions even though the transactions have a lesser gas limit. Figure 17 shows a transaction that was posted with a gas limit of 4,500,000 and a deadline time of 50. Since the deadline time was higher than both the previous transactions, this transaction is posted in block 37.

Figure 18 shows another such scenario where the gas limit is higher and the deadline time is farther than the current block, which gives the transaction lesser priority. The transaction is posted with a gas limit of 4,000,000 and the deadline time set to 58. The block number that the transaction got posted is the 39th block of the chain.

6. Conclusions and Discussion

This paper has proposed an analytical approach how to design and realize a real-time chain (Ethereum blockchain-based) under a stringent real-time deadline requirement. A new and novel analytical model has been proposed to estimate the performance of the real-time chain in a quantitative manner, referred to as the variable bulk arrival and variable bulk service with λ_F where λ_F is a random variable employed specifically to address the stringent real-time deadline requirement. The variable bulk arrival rate is assumed to vary linearly, proportional to the size of the transactions in a multiple of λ per slot with success to arrive within deadline, and some of the transactions in λ_F to fail to meet the real-time deadline, and the variable bulk service is assumed to take place when the number of slots in

the mined transactions reaches at every possible number i, where $0 \leq i \leq n$, i.e., a bulk processing of multiple transactions in multiple slots between 0 and n, inclusive, for posting in a block. Note that the real-time deadline requirement is considered under the underlying assumption that the size of each block is adaptive, in other words, varying block by block, and the proposed VBAVBS model with λ_F normalizes the probability for each size of the block throughout. Numerical simulations are conducted on Matlab to verify and demonstrate the efficacy of the model and reveal a good agreement with what was expected intuitively as a baseline validation. The primary performance measurements to be taken are L, L_Q, W_Q, W and γ versus n (i.e., size of a block), λ (i.e., successful transaction arrival rate or speed), λ_F (i.e., unsuccessful transaction arrival rate or speed) and $\frac{1}{\mu}$ (i.e., block posting time). The simulation results demonstrate the efficacy and validity of the proposed real-time chain in a quantitative manner with respect to the proposed VBAVBS model with λ_F as far as it is concerned about L, L_Q, W_Q, W and γ versus various n, λ, λ_F, μ. It is noteworthy that λ_F, as a real-time-specific random variable, exhibits quite a significant impact on L, L_Q, W_Q, W. A demo has been developed to demonstrate a real-time chain with modification on the Ethereum open source Geth 1.9.11.

Author Contributions: N.P.: overall supervision of the research and the manuscript. A.K.: a supervisee of the research and main contributor to the simulations, the demo and the manuscript. H.-Y.K.: overall supervision of the research, the manuscript and the publication cost. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Korean Government (MSIT) (No. 2019R1A2C1008533).

Acknowledgments: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) (No. 2019R1A2C1008533). This work was also supported by the 2020 Hongik University Research Fund.

Conflicts of Interest: Authors declare no conflict of interest.

Appendix A

Solution for the balance equations shown in Section 3 are provided here. Equation (A1) represents the steady state probability for P_1.

$$\left(\lambda + 2\lambda + 3\lambda + \cdots + (n-1)\lambda + \lambda_F + \frac{\mu}{n-0}\right)P_1 = \lambda P_0 \tag{A1}$$

Equations (A2)–(A5) process the simplification.

$$\left(\frac{\lambda(n-1)(n-2)}{2} + \lambda_F + \frac{\mu}{n-0}\right)P_1 = \lambda P_0 \tag{A2}$$

$$\left(\frac{\lambda(n-1)(n-2)}{2} + \lambda_F + \frac{\mu}{n-0}\right)P_1 = \lambda P_0 \tag{A3}$$

$$P_1 = \lambda\left(\frac{\lambda(n-1)(n-2)}{2} + \lambda_F + \frac{\mu}{n-0}\right)^{-1} P_0 \tag{A4}$$

$$P_1 = \lambda q_1 P_0 \tag{A5}$$

Similarly, P_2 can be expressed as follows:

$$\left(\lambda + 2\lambda + 3\lambda + \cdots + (n-2)\lambda + \lambda_F + \frac{\mu}{n-1}\right)P_2 = \lambda P_1 + 2\lambda P_0 \tag{A6}$$

$$\left(\frac{\lambda(n-2)(n-3)}{2} + \lambda_F + \frac{\mu}{n-1}\right)P_2 = \lambda(P_1 + 2P_0) \tag{A7}$$

$$P_2 = \lambda\left(\frac{\lambda(n-2)(n-3)}{2} + \lambda_F + \frac{\mu}{n-1}\right)^{-1}(Q_1 P_0 + 2P_0) \tag{A8}$$

$$P_2 = \lambda q_2(Q_1 P_0 + 2P_0) \tag{A9}$$

$$P_2 = \lambda q_2(Q_1 + 2)P_0 \tag{A10}$$

$$P_2 = Q_2 P_0 \tag{A11}$$

P_3 can be expressed as follows, and simplified in Equations (A12) and (A13).

$$\left(\lambda + 2\lambda + 3\lambda + \cdots + (n-3)\lambda + \lambda_F + \frac{\mu}{n-2}\right)P_3 = \lambda P_2 + 2\lambda P_1 + 3\lambda P_0 \tag{A12}$$

$$\left(\frac{\lambda(n-3)(n-4)}{2} + \lambda_F + \frac{\mu}{n-2}\right)P_3 = \lambda(P_2 + 2P_1 + 3P_0) \tag{A13}$$

Similarly, P_i, $0 < i < n$ can be expressed as follows:

$$\left(\lambda + 2\lambda + 3\lambda + \cdots + (n-i)\lambda + \lambda_F + \frac{\mu}{n-i+1}\right)P_i = \lambda P_{i-1} + 2\lambda P_{i-2} + \ldots + i\lambda P_0 \tag{A14}$$

$$\left(\frac{\lambda(n-i)(n-i+1)}{2} + \lambda_F + \frac{\mu}{n-i+1}\right)P_i = \lambda(P_{i-1} + 2P_{i-2} + \ldots + iP_0) \tag{A15}$$

Lastly, P_n can be expressed as follows:

$$(\lambda_F + \mu)P_n = \lambda P_{n-1} + 2\lambda P_{n-2} + \ldots + n\lambda P_0 \tag{A16}$$

$$(\lambda_F + \mu)P_n = \lambda(P_{n-1} + 2P_{n-2} + \ldots + nP_0) \tag{A17}$$

P_0 can be solved by substituting Equation (4) into Equation (3):

$$P_0 + P_1 + P_2 + \ldots + P_n = 1 \tag{A18}$$

$$P_0 + q_1^{-1}P_0\left[\sum_{j=1}^{1} j\left[\sum_{k=1}^{0}\left[\prod_{l=1}^{k-1} q_l^{-1}\right]\right]k\right] + q_2^{-1}P_0\left[\sum_{j=1}^{2} j\left[\sum_{k=1}^{1}\left[\prod_{l=1}^{k-1} q_l^{-1}\right]\right]k\right] + \ldots + P_n = 1 \tag{A19}$$

As, P_0 is a term that is common in all the terms in Equation (A19):

$$P_0\left[1 + q_1^{-1}\left[\sum_{j=1}^{1} j\left[\sum_{k=1}^{0}\left[\prod_{l=1}^{k-1} q_l^{-1}\right]\right]k\right] + q_2^{-1}\left[\sum_{j=1}^{2} j\left[\sum_{k=1}^{1}\left[\prod_{l=1}^{k-1} q_l^{-1}\right]\right]k\right] + \ldots\right] = 1 \tag{A20}$$

Once the value of P_0 is known, the other steady state probabilities of the Markovian model can be obtained by using Equation (4) consecutively. Given that all the steady state probabilities are known, the values of L, L_Q, W and W_Q are calculated.

Appendix B

The configuration settings for the components used in the real-time demo model described in Section 5 are shown in Table A1.

Table A1. Configuration settings of Ethereum and other components.

Component	Version
Geth/Ethereum	1.9.11
Architecture	amd64
Go version	go1.14
Operating system	darwin
Nodejs	v12.14.1
web3	1.2.7

References

1. Rouhani, S.; Deters, R. Performance Analysis of Ethereum Transactions in private blockchain. In Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 24–26 November 2017.
2. Chauhan, A.; Malviya, O.P.; Verma, M.; Mor, T.S. Blockchain and Scalability. In Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion, Lisbon, Portugal, 16–20 July 2018.
3. Kuzuno, H.; Karam, C. Blockchain explorer: An analytical process and investigation environment for bitcoin. In Proceedings of the 2017 APWG Symposium on Electronic Crime Research (eCrime), Scottsdale, AZ, USA, 25–27 April 2017.
4. Nakamoto, S.; Bitcoin, A. A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 19 March 2019).
5. Ethereum: A Super Decentralized Generalized Transaction Ledger. Available online: http://gavwood.com/paper.pdf (accessed on 19 March 2019).
6. Weber, I.; Gramoli, V.; Ponomarev, A.; Staples, M.; Holz, R.; Tran, A.B.; Rimba, P. On Availability for Blockchain-Based Systems. In Proceedings of the 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), Hong Kong, China, 26–29 September 2017.
7. Peck, M.E. Blockchains: How They Work and Why They'll Change the World. *IEEE Spectr.* **2017**, *54*, 26–35. [CrossRef]
8. Qin, R.; Yuan, Y.; Wang, F.Y. Research on the Selection Strategies of Blockchain Mining Pools. *IEEE Trans. Comput. Soc. Syst.* **2018**, *5*, 748–757. [CrossRef]
9. Cinque, M.; Esposito, C. How to Assess the Dependability of Applications on Top of the Blockchain: Novel Research Challenges. In Proceedings of the 2018 14th European Dependable Computing Conference (EDCC), Iasi, Romania, 10–14 September 2018.
10. Lombardi, F.; Aniello, L.; de Angelis, S.; Margheri, A.; Sassone, V. *A Blockchain-Based Infrastructure for Reliable and Cost-Effective Iot-Aided Smart Grids*; IET: London, UK, 2018.
11. Zhang, Q.; Novotny, P.; Baset, S.; Dillenberger, D.; Barger, A.; Manevich, Y. LedgerGuard: Improving Blockchain Ledger Dependability. Available online: https://www.researchgate.net/publication/324939637_LedgerGuard_Improving_Blockchain_Ledger_Dependability (accessed on 15 April 2019).
12. Seol, J.; Kancharla, A.; Park, N.; Park, N.; Park, I. The Dependability of Crypto Linked Off-chain File Systems in Backend Blockchain Analytics Engine. *Int. J. Netw. Distrib. Comput.* **2018**, *6*, 210–215. [CrossRef]
13. How to Check the Reliability of a Blockchain Project. Available online: https://coinjournal.net/sponsored-story-how-to-check-the-reliability-of-a-blockchain-project/ (accessed on 19 March 2019).
14. Worstall, T. Fascinating Number: Bitcoin Mining Uses $15 Million's Worth of Electricity Every Day." Forbes, 3 December 2013. Available online: http://www.forbes.com/sites/timworstall/2013/12/03/fascinating-number-bitcoin-mining-uses-15-millions-worth-ofelectricity-every-day/ (accessed on 19 March 2019).
15. Chadha, G.K.; Singh, A. Bitcoin Block-Chain Mining. In Proceedings of the 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 29–31 May 2019; pp. 152–157.
16. Lee, W.-M. *Beginning Ethereum Smart Contracts Programming*; With Examples in Python, Solidity and JavaScript; Apress: New York, NY, USA, 2019.
17. Kancharla, A.; Jongho, S.; Park, N.; Kim, H. Slim Chain and Dependability. In Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI 2020), Taipei, Taiwan, 1–5 June 2020.
18. Kancharla, A.; Park, N.; Ke, Z.; Kim, H. Hybrid Chain and Dependability. In Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI 2020), Taipei, Taiwan, 1–5 June 2020.
19. Jongho, S.; Kancharla, A.; Park, N.; Kim, H. A Variable Bulk Arrival and Static Bulk Service Queueing Model for Blockchain. In Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI 2020), Taipei, Taiwan, 1–5 June 2020.

20. Kancharla, A.; Park, N. A Realtime Crypto Computing and Block-Dependability. In Proceedings of the 9th IEEE International Symposium on Cloud and Service Computing, Kaohsiung, Taiwan, 18–21 November 2019.
21. Kancharla, A.; Park, N. Dependable Industrial Crypto Computing. In Proceedings of the 28th International Symposium on Industrial Electronics (IEEE-ISIE 2019), Vancouver, BC, Canada, 11–14 June 2019.