A STUDY ON QUANTITATIVE DESIGN FOR

DYNAMIC BLOCKCHAIN-BASED COMPUTING

By

JONGHO SEOL

Bachelor of Science in Control and Instrumentation
Engineering
Kangwon National University
Samcheok, Kangwon
2000

Master of Science in Computer Science
Oklahoma State University
Stillwater, Oklahoma
2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2021

A STUDY ON QUANTITATIVE DESIGN FOR

DYNAMIC BLOCKCHAIN-BASED COMPUTING

Dissertation Approved:

Dr. Nohpill Park

Dissertation Adviser

Dr. Christopher Crick

Dr. Esra Akbas

Dr. Jonghoon Kim

Name: JONGHO SEOL

Date of Degree: JULY, 2021

Title of Study: A STUDY ON QUANTITATIVE DESIGN FOR DYNAMIC BLOCKCHAIN-BASED COMPUTING

Major Field: COMPUTER SCIENCE

Abstract: This research proposes novel embedded Markovian queueing model-based quantitative models in order to establish a theoretical foundation to design a dynamic blockchain-based computing system with a specific interest in Ethereum. The proposed models commonly assume variable bulk arrivals of transactions in Poisson distribution, i.e., $M^{1,n}$, where n the number of slots across all the mined transactions to be posted in a block or the current block. Firstly, a baseline model is proposed to have a static bulk service of transactions in exponential time, i.e., M^n, for posting the transactions in the current block, referred to as Variable Bulk Arrival and Static Bulk Service (VBASBS) queueing model of the $M^{1,n}/M^n/1$ type, in which note that n is fixed in order to demonstrate a static chain in terms of the size of the block. Secondly, an adaptive chain model, as a solution of dynamic blockchain in a reactive manner, is proposed based on a Variable Bulk Arrival and Variable Bulk Service (VBAVBS) queueing model of the $M^{1,n}/M^{1,i,n}/1$ type to provide a quantitative approach to design an adaptive chain that dynamically adapts the size of the block to varying performance trends, in which a state transitions from i back to 0, where $0 < i \leq n$, are tracked in order to demonstrate the dynamically adaptive size of the block. Lastly, an asynchronous chain model, as a solution of dynamic blockchain in a proactive manner, is proposed based on a Variable Bulk Arrival and Asynchronous Bulk Service (VBAABS) queueing model is developed and presented to study and demonstrate the fully asynchronous and staged asynchronous chains. The analytical models are simulated extensively to compare the basic performances of the proposed models such as the average transaction waiting time, the average number of slots per block, and throughput. Further, extensive experiments are conducted in order to validate the analytical results by redesigning the source code of Ethereum to implement and demonstrate each of the proposed chains such as the baseline, the adaptive, the fully-asynchronous and the staged-asynchronous chains. The analytical results and the experimental results will be compared and discussed extensively.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Blockchain technology [1, 2, 4, 5, 7, 10] is undergoing tremendous growth chocking itself up to its capacity and performance limits [58] and thereby resulting in cost spikes [19]. It is exigently sought to address and respond to these issues, being mostly concerned about the scalability [11, 12, 13, 14, 15, 16, 17, 24] dependability of the blockchain system, and privacy issues [53] in systems such as smart grid [35] and IoT [18, 25, 26, 27]. In this context, blockchain technology is emerging and gaining more and more attention from the technology community as an alternative to the current centralized-based internet protocol, namely, the decentralization with blockchain-based peer-to-peer internet protocol.

Blockchain technology also has been investigated by industry and research sectors for the benefit of a transparent decentralized network control [4, 7, 16, 28, 29]. The scalability and dependability issues of the blockchain system [23, 24] have been identified and addressed as the technology matures and saturates in its current form.

A quite extensive line of decentralized applications in IoT [18, 25, 26, 27] and MEC [33, 48] networks have been deployed and testbed to exercise various attempts to resolve those issues in [19, 34] with respect to security [52] and privacy [53] in IoT based smart grid. Blockchain is a decentralized network system [4, 7, 16, 28, 29] that employs web3 [43] as its underlying technology, based on the peer-to-peer communication protocol where all nodes store mutual data, as a basis for a reliable and trustworthy communication that does not require mutual verification. Due to this mutually trustworthy network requirement, there is a cost of extra block delay [38] for a smart contract [7, 49] to be used to transmit or store data as a transaction off [36] of the smart contract primarily due to a mining process to be performed to approve and post transactions in a block [7, 16, 50, 51] on the chain. In general, the network system used as a web2-based server-client network is relatively faster than the decentralized distributed network, that is, blockchain. Today, such decentralized systems and applications in many industries can be found as IoT and MEC [18, 25, 26, 27, 33], and more notably, blockchain technology for digital currency is used in central banks and eyed by many governments as a future digital currency option. However, the issue of transaction delay due to slower block processing (posting) speed must be addressed and resolved [19, 39] prior to extensive acceptance in the market. Note that it has been reported that the blockchain consensus algorithms [7, 16, 37] play the central role in the trustworthy system, it costs extra block time delay which leads to lower throughput of the system and increases waiting time. In this context, block size adjustment is being considered as an alternative solution to address the block delay with respect to the gas limit [53, 54] per block and to improve the performance [30, 31, 32, 44]. The gas limit per block proportionally influences block delay, scalability [12, 13, 14, 15, 17], and dependability [19].

Also, it has been reported that the consensus algorithms [7, 16] of the blockchain system is driving the mining process to keep the block time constantly to be delayed and eventually to lead to slowing down on the throughput and thereby limiting the capacity and even the functionality and operability of the network, which mandates an excessively powerful server and a high-efficiency yet extremely

costly network. After all, blockchain technology in its current form is facing a serious hurdle before finding itself as a true replacement of the current state of the art web2-based internet infrastructure. In this context, it is exigently sought to address and respond to these issues mainly about the scalability [15, 29] and performance [32] from the specific standpoint of the speed of block posting. As the blockchain technology-based network is seeking a way to break through those underlying technical issues, it has been considered that blockchain is proactively adapting to various performance criteria [30, 31, 32, 44] in order to speed up the execution of transactions with respect to block size and requirements. The basic network system of blockchain technology is a decentralized system that uses a web3 network, which is a peer-to-peer communication based on a distributed information network where all nodes store mutual data, as a reliable trust system that does not require mutual verification. Due to this mutually reliable network environment, there is an issue of block delay time until a contract is used to transmit or store data, i.e., as a transaction occurring, a mining process that is posting in a block as a confirmed transaction. The point is that the system used as a server client-based web2 network is relatively fast. Currently, decentralized applications in many industries are IoT and MEC, and more notably, blockchain technology for digital currency is used in central banks. However, the issue of transaction delay time due to block processing speed must be resolved. While it has been reported that the blockchain consensus algorithm makes it a trusted system, it causes block time delay, lowers the throughput of the system, and increases waiting time. In this context, block size and its conditions will improve blockchain performance by changing it and will solve the block size change and conditions for it in a way that can solve the scalability and dependability problems. The central idea driving the technology shift is the trust of the internet service which the legacy centralized-based internet service providers evidently have never convinced the users' community to sustain its credibility due to a few major reported breaches of trust no matter intentional or not, which triggered the technology drive towards blockchain which is today's and tomorrow's promise and the best-known solution of trust in internet services. However, as the blockchain technology matures, it

appears the technology is revealing either expected or unexpected performance flaws besides the main concern of trusts, such as scalability and speed issues, as the main hurdle from success. The key to the success of this technology in the market is how to bring the performance to the next level in order to cope with the market needs and requirements from the performance perspective.

A prerequisite to the performance needs and requirements is to establish an analytical model in order to establish an analytical model to evaluate and assure the performance of concern with high efficacy in a quantitative manner during the early design cycle. There have been a few analytically approached works [9] reported on dependability [20, 23, 24]. However, they are able to readily and adequately address the queueing nature of the transactions flow and block posting in particular, which will eventually prevent a synergistic and comprehensive analytical design of a dependable and high-performance blockchain system.

There are four different types of blockchain architectures to be proposed and studied in this dissertation for the above mentioned block size adjustment solution, namely, baseline chain [40] as the conventional solution without allowing the block size adjustment, the adaptive chain as a naïve solution to block size adjustment in a reactive manner, the fully asynchronous [56, 57] chain as another naïve solution yet in a proactive manner, and lastly, the staged asynchronous [56, 57] chain as a solution that is a hybrid form of the reactive and the proactive solutions.

The baseline chain [40] considered in this dissertation is the conventional chain (e.g., Ethereum [7, 16, 50]) with a fixed size for the blocks, and note that the size of a block is defined (represented) by the block gas limit [53, 54] which is fixed throughput unless otherwise controlled, and also note that in fact the physical size of each block in the baseline chain is practically fixed at 20~30KB [39] per block multiplied by the size of a node to represent a transaction hash which is in fact nothing to do with the size of the block to be adjusted as it concerned in this dissertation.

The adaptive chain is a chain in which the block size (i.e., the block gas limit as defined in this dissertation) is adaptive to the total of the gas fees used by the transactions [7, 16, 49, 54] in the previous block with a certain variation (or the average of a certain number of previous blocks) in a reactive manner (i.e., the size of the block stays static throughout the time period of block delay).

The asynchronous chain is an adaptive chain yet in a proactive manner, in other words, the size of the block is adapted on the fly. Two different types of asynchronous chains are proposed such that the fully asynchronous chain adapts the size of the block to the size of an arriving transaction on the fly immediately when a transaction arrives, which seemingly is an overly-costly, stringent and slow solution, yet a fully asynchronous and adaptive solution to the temporal requirement for a transaction posting in the block; and the staged asynchronous chain moderates the stringent proactive asynchrony of the posting of a transaction such that the adaptive posting is staged (or grouped) by the rages of the sizes of the transactions, e.g., instead of immediately posting each and every individual transaction in the block, the block posting is delayed for a stage delay to accommodate potentially more transactions within the range of the stage in an effort to relax the stringent asynchronous posting requirement of the fully asynchronous chain.

A variable bulk arrival and static bulk service (VBASBS) [40] of the $M^{1,n}/M^n/1$ type can be considered to analyze the performance of the baseline chain and can serve as the basis of analysis of such a variety of variations of chains as adaptive, fully asynchronous, and staged asynchronous chains as proposed in this paper. VBASBS is an embedded markovian model and defines the state of the chain by a single variable i, i.e., the number of slots pending in the current block for being posted, where note that a slot is approximated to 100,000 wei [39, 50] in this paper and note that a typical gas limit per block is set to 10 gwei [39, 50], and state transitions triggered from state i to j, where $0 \leq i, j \leq n$ and $i < j$, and n is the total possible number of slots to be posted in the current block, and a

state transition from n back to 0 takes place when the current block is topped off and posted in the chain.

As it turns to the adaptive chain, the VBASBS model can be extended to allow a state transition from each state, i, back to state, 0, in addition to the state transition from the state, n, back to state, 0, in other words, a block could be of any size and posted (i.e., flushed from any state $0 < i \leq n$) with the size that is adaptive to a certain condition such as (the average of) the total of the sizes of the transactions in the prior block(s), namely, a variable bulk arrival and variable bulk service (VBAVBS) of the $M^{1,n}/M^{1,n}/1$ type under the assumption that the block size adaptation takes place in a reactive manner after each round of block posting process and the size of the block (the block gas limit) stays static throughout the block posting process.

Lastly, in order to allow a block size to be adaptive in a proactive manner, namely, the asynchronous chain, the block size adaptation should take place on the fly such that the waiting time for a transaction to be posted be strictly asynchronous to its arrival time (namely, the fully asynchronous chain), or, at the most, be shorter than the possible maximum allowed block posting delay (e.g., block gas limit) (namely, a non-fully asynchronous chain such as the staged asynchronous chain as proposed in this paper). A variable bulk arrival and asynchronous bulk service of the $M^{1,n}/M^{ik,n}/1$ type can be considered, where ik indicates the staged group of states such that in the fully asynchronous chain each state i has only an arriving state transition from state 0 and only an outgoing state transition back to state 0 in order to model the strict asynchrony; and in the staged asynchronous chain the strict asynchrony is moderated to some extent by letting states in the range from $ik + 1$ to $(i + 1)k$ trigger state transition from l to m, where $ik + 1 \leq l, m \leq (i + 1)k$ and $l < m$ and $(i + 1)k \leq n$.

The primary interest of the baseline chain in Chapter III is to develop an embedded Markovian queueing model of the $M^{1,n}/M^{n}/1$ type in order to establish a theoretical foundation to design a

blockchain-based system with a focus on the stochastic behavior of the mined transactions waiting to be posted for the block delay as the bulk synchronization point. The model assumes variable bulk arrivals of transactions in Poisson distribution, i.e., $M^{1,n}$, where n the number of slots across all the mined transactions and static bulk services of transactions in exponential time, i.e., M^{n}, for posting in the current block, namely, a variable bulk arrival and variable bulk posting (VBASBP). The primary performance measurements to be taken are the average number of slots no matter how many transactions are mined under the assumption of the 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 vary linearly proportional to the size of the transactions in a multiple of λ (note that there is only a single stage of a queue of waiting transactions (in terms of slots) assumed for simplicity instead of assuming two independent arrival rates of the transactions, one for the transaction pool and another for the waiting queue for the block posting, thereby assuming only a single bulk arrival rate per slot λ, which might be to some extent different in practice), and the static bulk service is assumed to take place when the number of slots in the mined transactions reaches at n, i.e., a bulk processing of multiple transactions in multiple slots for posting in a block.

In Chapter IV, with the baseline model of VBASBS (i.e., a $M^{1,n}/M^{n}/1$ type), the primary interest is to develop an embedded Markovian queueing model of the $M^{1,n}/M^{1,i,n}/1$ type in order to establish a quantitative foundation to design a blockchain-based system with a focus on the stochastic behavior of the mined transactions waiting to be posted for the block time as potentially purging at every state, which is possibly being from any state, $P_i(0 < i \leq n)$ back into the state, P_0. As in the baseline model of VBASBS, the proposed model assumes variable bulk arrivals of transactions in Poisson distribution, $M^{1,n}$, where n the number of slots across all the mined transactions, but, variable bulk service of transactions in exponential time, $M^{1,i,n}$, for posting into the current block at any state in a slot, VBAVBS. The primary performance measurements are to be taken in comparison with the

baseline model, such as the average number of slots no matter how many transactions are mined under the assumption of the 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 vary linearly proportional to the size of the transactions in a multiple of λ note that there is only a single stage of a queue of waiting transactions (in terms of slots) assumed for simplicity instead of assuming two independent arrival rates of the transactions, one for the transaction pool and another for the waiting queue for the block posting, thereby assuming only a single bulk arrival rate per slot λ, which might be to some extent different in practice, and the variable bulk service is assumed to take place when the number of slots in the mined transactions reaches at any state, $1, 2, \dots, n-1, n,$ i.e., a bulk processing of single or multiple transaction(s) in single or multiple slot(s) for posting in a block.

In Chapter V, with the baseline models of VBASBS (i.e., a $M^{1,n}/M^{n}/1$ type) and the adaptive model VBAVBS (i.e., a $M^{1,n}/M^{1,i,n}/1$ type), the primary interest is to develop an embedded Markovian queueing model of the asynchronous chain, namely a Variable Bulk Arrival and Asynchronous Bulk Service (VBAABS) model is developed and presented to study and demonstrate the theoretical performance of an asynchronous chain in order to establish a quantitative foundation to design a blockchain-based system with a focus on the stochastic behavior of the mined transactions waiting to be posted for the block time as potentially purging at every single state or every staged state(s). Two types of asynchronous models are proposed as fully and staged asynchronous.

This dissertation is organized as follows. The preliminaries and review are introduced in the following Chapter II. The baseline chain model, VBASBS, is proposed and the numerical analysis is reviewed in Chapter III. The adaptive chain model, VBAVBS, is proposed and the numerical analysis is compared with between VBASBS and VBAVBS in Chapter IV. The asynchronous chain models, VBAABS, two different types of architecture, the fully asynchronous chain and the staged

asynchronous chain are proposed and the numerical analysis is reviewed in Chapter V. The implementation and experimental results of the proposed four types of models are presented and compared to the baseline chain model, VBASBS, to validate the efficacy and benefits of the rest of the models, in addition, the experimental environment and procedure are shown in Chapter VI. Lastly, the conclusion and discussions are drawn in Chapter VII.

CHAPTER II

PRELIMINARIES AND REVIEW

There have been a few works to address and investigate on various yet critical performance and dependability issues and problems as identified in various blockchain-based crypto computing systems. Various hypothetical and theoretical designs [6, 8, 9] of a few crypto computing solutions have been developed in order to establish an engine for preliminary yet extensive parametric simulation, and some results have been demonstrated and validated through isolated testing on Ethereum and Hyperledger open source-based prototypes [20, 23, 24]. 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 hypothetical and theoretical models emphasized and centered around the dependability of each of those crypto solutions to accommodate such capabilities as the on/off-balanced crypto computing [23], the real-time computing [24], the slim-computing [21, 46] and the hybrid computing [45]. Dependability for each of those crypto solutions has been identified and defined along with various performance variables and ultimately has provided a theoretical yet practical understanding of each crypto solution.

A prototype, to demonstrate some of those crypto solutions and to validate their hypothetical and theoretical results, has been built by identifying and isolating the insertion points for necessary technology modification within Ethereum and Hyperledger open source to start out with and to ultimately realize a new core blockchain for optimal crypto computing.

The on/off-balanced chain [23] has been proposed in order to investigate on how to assure the dependability of a crypto system built across on and off the blockchain facilitated and coordinated by using the proposed adaptive checkpoint and rollback algorithm. The theoretical foundation of the proposed checkpoint and rollback algorithm is to characterize the variables affecting the dependability such as security [49], authenticity and reliability with respect to the rates of hit by any events of those issues, the rates to detect and diagnose, and then the rate to vote for a consensus whether to mark a checkpoint and trigger a rollback or not. Based on the variables characterization in a stochastic manner, the steady state probabilities and state transition probabilities have been derived in order to assure the ultimate effective dependability of each individual dependability variable (i.e., security, authenticity, and reliability) [47], then, finally to assure the dependability in a compound manner with each variable assigned a weight depending on the nature of the systems specifications. Based on the theoretical study [3, 21] a prototype of the crypto system has been built to demonstrate the underlying architecture and operations and to justify the need for such system to take synergistic advantages from both on- and off-chains, with an experimental result of a benefit in gas fee [4, 7, 16] with respect to performance and dependability, which is the most exigently addressed issue today in blockchain systems especially in Ethereum network of blockchain. An astonishing result of gas fee saving has been demonstrated. It is expected that the crypto system will benefit more if more computationally intensive transactions are executed off-chain and vice versa. An isolated testing has been conducted for a demonstration of the purpose on the proposed checkpoint and rollback algorithm on Ethereum open source. The real-time chain [13, 19, 24] has been proposed in order to

investigate on an approach how to design and realize crypto computing (Ethereum blockchain-based [16]) under the stringent real-time requirement. In order to evaluate the efficacy of the approach, a new analytical metric has been defined and developed to estimate the dependability, referred to as block-dependability. The proposed block-dependability precisely models the probability for the mined transactions to be posted within the current, in other words, within the target block delay, further, namely, within the deadline required if their expected execution times are within the temporal range of the deadline. Various methods how to prioritize [17] and select transactions in the pending transaction pool in order to facilitate those transactions to be executed within their deadline requirements, such as the normal, random, sorted, and stratified [7], have been proposed. A set of performance variables, or parameters, such as the number of pending transactions in the pool, the average speed of the transactions, gas fees, deadlines as well as the number of miners, are identified and taken into the block-dependability analysis in order to reveal the influence of each variable on the block-dependability, versus each of those proposed prioritization and selection methods. Extensive parametric simulations are being conducted through an isolated testing on the Ethereum open source; the slim chain has been investigated in [20] in order to address and resolve the scalability issue [55] of blockchain-based crypto computing in which it is required that every node participating in the computation carry a full load of the entire chain of blocks all the way from the genesis block. As evidenced from the preliminary results of the on/off-balanced chain [23], the performance and dependability [20] could be significantly improved by balancing the amount of computations across on- and off-chains, which depends on the type of computations such that on-chain execution is more suitable for more dependability-stringent or computation-intensive transactions, and off-chain for less dependability-stringent or less data-intensive computation. A smart balancing of computation across on/off-chain is expected to relieve the spatial and temporal overhead of managing the otherwise explosively growing size of the blockchain, as referred to as a slim chain in this paper

(cf. the light chain is a technology to limit the temporal extent of the chain of blocks for synchronization). A novel theoretical model has been developed to evaluate the efficacy of the slim chain with respect to the dependability from a single transaction's standpoint in a stochastic manner. Further, the dependability will be traced with respect to whether the transaction of concern is to write off the chain or read from off the chain as the transactions writing intensively off the chain are more likely to be dependable than the ones reading intensively from off chain. Also, note that IPFS (Inter-Planetary File System) [11, 16, 22] is the off-chain storage system to be considered in the slim chain. A prototype with an isolated testing on the Ethereum open source is being built for validation purpose along with extensive parametric simulations; and the hybrid chain is being studied in order to investigate on a new blockchain network that is to be built particularly across private (i.e., permissioned) and main nets (i.e., permissionless). Note that the hybrid chain is distinguished from the earlier mentioned on/off-balanced chain such that the hybrid chain is concerned across two different types of nets (e.g., Hyperledger-based private net vs. Ethereum-based main net) while the on/off-balanced chain is concerned across on- and off-chain (e.g., Ethereum-based main net vs. cloud or IPFS). It is essential to build a dependable interface between the private network and the main network if business-to-consumer (or vice versa) transactions are the primary transaction of interest. In the course of interfacing across the private and the main nets, dependability is to be considered as one of the most critical design factors in order to ensure that the private transactions stay within the private territory and publicized transactions stay public in the main net, and further in order to facilitate a seamless yet dependable migration of transactions across the border. In this context, the efficacy of the privacy of the private network side and the publicity of the main net side is addressed and modeled by tracing a transaction's stochastic process at a steady state. A prototype for an isolated testing across the Ethereum and Hypercubes open source for validation purposes is being built for validation purposes along with extensive parametric simulations.

However, the dependability models or performance models in [20, 23, 24] cannot readily and adequately address the queueing nature of the transactions flow and block posting other than that they will provide a sound theoretical foundation for dependability analysis in various blockchain contexts in line with the proposed VBASBS model. Ultimately, a synergistic model will be highly desired and pursued to model and assure both the dependability and the performance as two primary and concurrent variables of the blockchain system, thereby synergizing those dependability models and the VBASBS model into a comprehensive and integrated analytical design tool.

CHAPTER III

BASELINE CHAIN MODEL

The baseline model is proposed as named, Variable Bulk Arrival and Static Bulk Service (VBASBS). In the proposed VBASBS model, an embedded Markovian single-server exponential queueing system (i.e., $M^{1,n}/M^n/1$) is considered without loss of generality, and the server (e.g., the server is the equivalence of the group of miners to select the transactions to be posted) serves the entire batch of customers (e.g., the customers are the equivalence of the transactions to be posted in the block) in the queue (e.g., a queue is the equivalence of a block to be mined and posted) all at once at the same time. Whenever the server completes a service (e.g., a service is an equivalence a process of posting a block), it then purges the queue (e.g., the equivalence of the posting a block) and then serves the influx of new customers incoming. 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 customers 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 (e.g., equivalently, posting and purging the entire queue).

Without loss of generality, it is assumed that customers arrive at an exponential rate of λ.

The underlying queueing process is assumed to take place with fixed-sized slots and the status of the queue is determined by the number of slots.

Given the assumptions as made above, the proposed VBASBP model employs an embedded Markovian queueing model 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 than the number of transactions whose size varies in the number of slots.

The state transition diagram of the baseline chain model is shown with all states, λ, and μ in Figure 1.

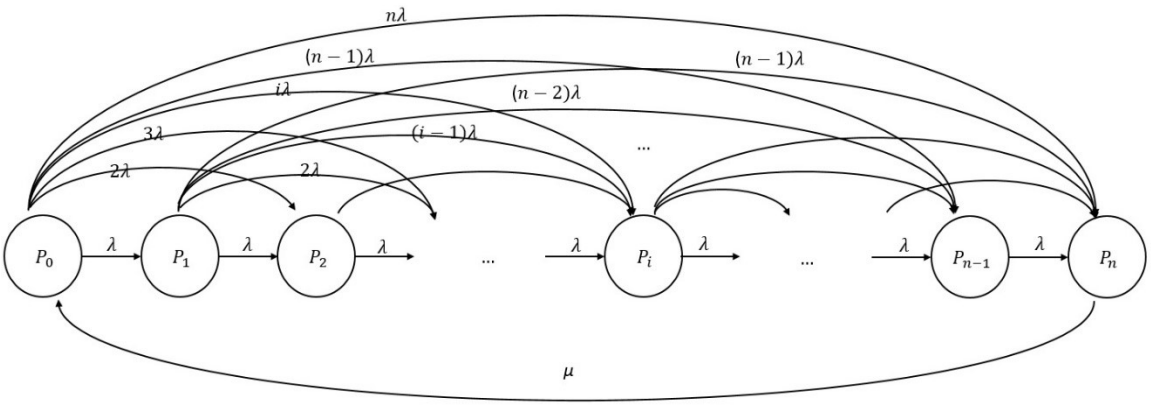

Figure 1. State transition diagram of the baseline chain model

P_0 : the state in which there is no transaction (i.e., no slot) arrived in the queue as of yet for the posting in the block, currently.

P_n : the state in which there is n number of slots (i.e., which is 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.

P_i : the state in which there is i number of slots (where $0 < i < n$) arrived in the queue for the posting in the block, currently.

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, 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.

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 precisely from P_n back to P_0.

The following Figure 2 shows the state transitions around P_i for the balance equations.

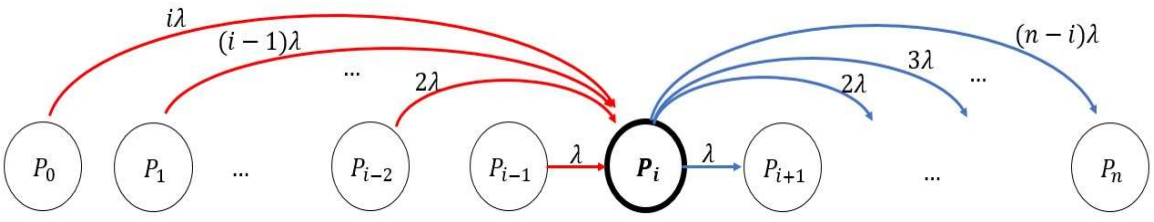

Figure 2. State transition around P_i for balance equations of the baseline chain model

1. Baseline Chain Model Equations

The balance equations for VBASBS are as follows.

$$(\lambda + 2\lambda + 3\lambda + \cdots + n\lambda)P_0 = \mu P_n$$

$$\left(\lambda \frac{n(n+1)}{2}\right) P_0 = \mu P_n$$

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

The following Figure 3 shows all incoming transitions to P_0 are equal to all outgoing transitions from P_0.

$$\mu P_n = (\lambda + 2\lambda + \cdots + (n-2)\lambda + (n-1)\lambda + n\lambda)P_0$$

$$(\lambda + 2\lambda + 3\lambda + \cdots + (n-i)\lambda)P_i = \lambda P_{i-1} + 2\lambda P_{i-2} + 3\lambda P_{i-3} + \cdots + i\lambda P_0 \tag{2}$$

$$P_1 = q_1 P_0 \tag{3}$$

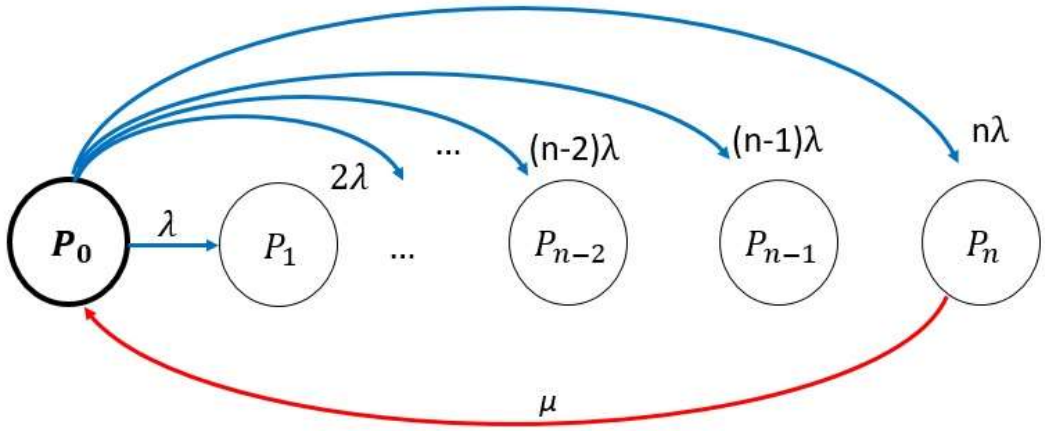

Figure 3. State transition around P_0 for balance equations of the baseline chain model

From Equations (1), (2), and (3), P_i, $0 < i < n$, can be expressed in terms of P_0 as follows.

$$P_2 = q_2(q_1 + 2)P_0 \tag{4}$$

$$P_3 = q_3(q_2(q_1 + 2) + 2q_1 + 3)P_0 \tag{5}$$

$$P_4 = q_4(q_3(q_2(q_1 + 2) + 2q_1 + 3) + 2q_2(q_1 + 2) + 3q_1 + 4)P_0 \tag{6}$$

$$P_4 = P_0q_4(1(q_3q_2q_1 + \ q_3q_22 + q_33) + 2(q_2q_1 + q_22) + 3(q_1) + 4) \tag{7}$$

$$P_0 + P_1 + P_2 + \cdots + P_n = 1 \tag{8}$$

P_i can be generalized and expressed as follows.

$$P_i = \ q_iP_0\left[\sum_{j=1}^{i} j\left[\sum_{k=1}^{i-1}\left[\prod_{l=1}^{k-1} q_l\right]k\right] + i\right] \tag{9}$$

Where,

$$0 < i < n$$

$$q_i = \frac{2}{(n-i)(n-i+1)}$$

$$q_l = \frac{2}{(n-l)(n-l+1)}$$

$$\prod_{l=1}^{k-1} q_l = \prod_{l=1}^{k-1} \frac{2}{(n-l)(n-l+1)} = \frac{2^{k-1}}{\frac{(n!)^2}{(n-0)(n-k+1)}} = \frac{2^{k-1}(n)(n-k+1)}{(n!)^2} \quad (10)$$

Then, it can be expressed in Equation (11) as follows.

$$\sum_{k=1}^{i-1} \left(\prod_{l=1}^{k-1} q_l \right) k = \sum_{k=1}^{i-1} \left(\frac{2^{k-1}(n)(n-k+1)}{(n!)^2} \right) k \quad (11)$$

$$= \frac{n}{(n!)^2} \left((n+1) \left(\sum_{m=0}^{i-2} 2^m m \right) - \left(\sum_{m=0}^{i-2} 2^m m^2 \right) + \left(n \frac{2\left(2^{i-2}-1\right)}{(2-1)} \right) \right) \quad (12)$$

In Equation (12), there are two sigma forms to solve as follows.

The first sigma form $\sum_{m=0}^{i-2} 2^m m$ can be expressed as follows.

$$\sum_{m=0}^{i-2} 2^m m = 2^0 0 + 2^1 1 + 2^2 2 + \cdots + 2^{i-3}(i-3) + 2^{i-2}(i-2) = 2^{i-1}(i-3) + 2 \quad (13)$$

The following Equation (14) rewrites equation (12) by taking Equation (13) in place.

$$\sum_{k=1}^{i-1}\left(\prod_{l=1}^{k-1} q_l\right)k$$

$$= \frac{n}{(n!)^2}\left((n+1)\left(2^{i-1}(i-3)+2\right) - \left(\sum_{m=0}^{i-2} 2^m m^2\right) + \left(n\frac{2\left(2^{i-2}-1\right)}{(2-1)}\right)\right) \quad (14)$$

In Equation (14), another sigma form $\sum_{m=0}^{i-2} 2^m m^2$ can be expressed as in Equation (15).

$$\sum_{m=0}^{i-2} 2^m m^2 = 2^{(i-1)}(i^2 - 6i + 11) - 6 \quad (15)$$

Then, the following is obtained.

$$\sum_{k=1}^{i-1}\left(\prod_{l=1}^{k-1} q_l\right)k$$

$$= \frac{n}{(n!)^2}\sum_{k=1}^{i-1}\left(2^{k-1}(n-k+1)\right)k$$

$$= \frac{n}{(n!)^2}\left((n+1)\left(2^{i-1}(i-3)+2\right) + \left(n\frac{2\left(2^{i-2}-1\right)}{(2-1)}\right) + \left((-1)\left(2^{(i-1)}(i^2-6i+11)-6\right)\right)\right) \quad (16)$$

Then, lastly, the following term can be solved as follows as shown in Equation (17).

$$\sum_{j=1}^{i} j\left(\sum_{k=1}^{i-1}\left(\prod_{l=1}^{k-1} q_l\right)k\right)$$

$$= \sum_{j=1}^{i} j\left(\frac{n}{(n!)^2}\left((n+1)\left(2^{i-1}(i-3)+2\right)+\left(n\frac{2\left(2^{i-2}-1\right)}{(2-1)}\right)+\left((-1)\left(2^{(i-1)}(i^2-6i+11)-6\right)\right)\right)\right)$$

$$= \sum_{j=1}^{i} j\left(\frac{n}{(n!)^2}\left(\left(2^{i-1}in-2^{i-1}3n+2n+2^{i-1}i-2^{i-1}3+2\right)+\left(2^{i-1}n-2n\right)+\left(-2^{(i-1)}i^2+2^{(i-1)}6i-2^{(i-1)}11+6\right)\right)\right)$$

$$= \sum_{j=1}^{i} j\left(\frac{n}{(n!)^2}\left(\left(2^{i-1}in-2^{i-1}3n+2n+2^{i-1}i-2^{i-1}3+2\right)+\left(2^{i-1}n-2n\right)+\left(-2^{(i-1)}i^2+2^{(i-1)}6i-2^{(i-1)}11+6\right)\right)\right) \quad (17)$$

Taking the Stirling's Approximation as shown in (18), Equation (17) can be rewritten as shown in Equation (19).

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \quad (18)$$

$$\sum_{j=1}^{i} j\left(\frac{n}{\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)^2}\left(\left(2^{i-1}in-2^{i-1}3n+2n+2^{i-1}i-2^{i-1}3+2\right)+\left(2^{i-1}n-2n\right)+\left(-2^{(i-1)}i^2+2^{(i-1)}6i-2^{(i-1)}11+6\right)\right)\right) \quad (19)$$

Where,

$$\sum_{j=1}^{i} j = \left(\frac{i(i+1)}{2}\right)$$

Equation (19) can be rewritten the equation as follows.

$$\sum_{j=1}^{i} j \left(\sum_{k=1}^{i-1} \left(\prod_{l=1}^{k-1} q_l \right) k \right) + i$$

$$= \left(\frac{i(i+1)}{2}\right) \left(\frac{n}{\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)^2} \right) \left(2^{i-1}(in - 2n + 7i - 14 - i^2) + 8\right)$$

Now, P_i can be expressed as follows.

$$P_i = q_i P_0 \left(\left(\frac{i(i+1)}{2}\right) \left(\frac{n}{\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)^2} \right) \left(2^{i-1}(in - 2n + 7i - 14 - i^2) + 8\right) + i \right) \qquad (20)$$

where,

$0 < i < n$

$$q_i = \frac{2}{(n-i)(n-i+1)}$$

$$P_i = \frac{2}{(n-i)(n-i+1)} P_0 \left(\left(\frac{i(i+1)}{2}\right) \frac{n}{\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)^2} \left(2^{i-1}(in - 2n + 7i - 14 - i^2) + 8\right) + i \right)$$

From Equation (8), $P_0 + \sum_{i=1}^{n-1} P_i + P_n = 1$ and P_0 can be solved as shown in Equations (21), (22), and (23).

$$P_0 \left(1 + \left(q_1 \left(\sum_{j=1}^{1} j + 1 \right) \right) + \left(q_2 \left(\sum_{j=1}^{2} j \left(\sum_{k=1}^{i-1} \left(\prod_{l=1}^{k-1} q_l \right) k \right) + 2 \right) \right) + \cdots \right.$$

$$\left. + \left(q_{n-1} \left(\sum_{j=1}^{n-1} j \left(\sum_{k=1}^{n-2} \left(\prod_{l=1}^{k-1} q_l \right) k \right) + 2 \right) \right) + \frac{\lambda}{\mu} \left(\frac{n(n+1)}{2} \right) \right) = 1 \qquad (21)$$

$$P_0 \left(1 + \sum_{i=1}^{n-1} \left(q_i \left(\sum_{j=1}^{i} j \left(\sum_{k=1}^{i-1} \left(\prod_{l=1}^{k-1} q_l \right) k \right) + i \right) \right) + \frac{\lambda}{\mu} \left(\frac{n(n+1)}{2} \right) \right) = 1 \qquad (22)$$

$$P_0 = \frac{1}{\left(1 + \sum_{i=1}^{n-1} \left(q_i \left(\sum_{j=1}^{i} j \left(\sum_{k=1}^{i-1} \left(\prod_{l=1}^{k-1} q_l \right) k \right) + i \right) \right) + \frac{\lambda}{\mu} \left(\frac{n(n+1)}{2} \right) \right)} \qquad (23)$$

From Equations (1), (8), and (9), all the remaining solutions for the balance equations for VBASBS (i.e., P_n from Equation (1) and P_i from Equation (8)) can be obtained).

The followings are a few baseline performance measurements of primary interests in VBASBS.

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).

$$L_Q = \sum_{i=0}^{n} iP_i \tag{24}$$

Where,

$$\sum_{i=0}^{n} iP_i = \sum_{i=0}^{n} i\left(q_i P_0 \left(\sum_{j=1}^{i} j\left(\sum_{k=1}^{i-1}\left(\prod_{l=1}^{k-1} q_l\right)k\right) + i\right)\right)$$

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

$$W_Q = \frac{L_Q}{\lambda} \tag{25}$$

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

$$W = W_Q + \frac{1}{\mu} \tag{26}$$

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).

$$L = \lambda W \tag{27}$$

2. Numerical Analysis

The primary objective of the simulation is to reveal the various preliminary performance of the blockchain system of interest such as L_Q, W_Q, W, and L versus n (i.e., size of a block), λ (i.e., transaction arrival rate or speed), and $\frac{1}{\mu}$ (i.e., block posting time). Note that the block posting time, $\frac{1}{\mu}$, is fixed at 15 seconds (i.e., $\mu = 0.0667$) in order to conduct the analysis under a practical parametric condition as typical block delay is known to be about 15 seconds in Ethereum. Note that this section is not to conduct a simulation to reveal against a particular blockchain system but to demonstrate a valid and baseline simulation model in the context of a queueing system.

The following graph plots L_Q (i.e., from Equation (24)) versus n for given pairs of λ and μ.

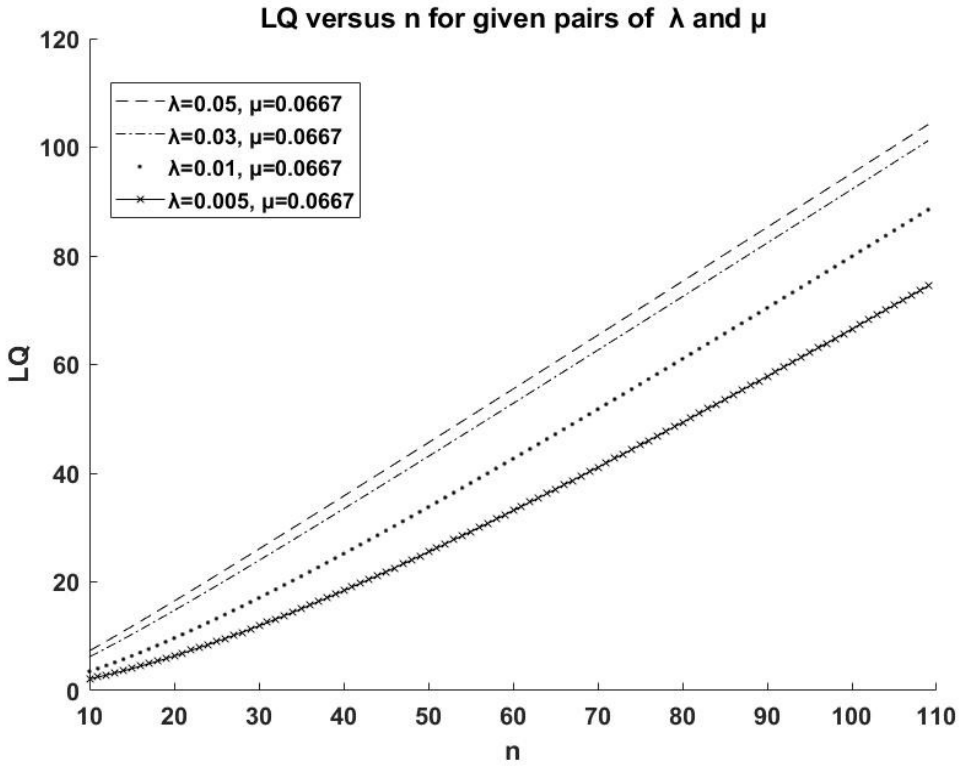

Figure 4. L_Q versus n for given pairs of λ and μ of the baseline chain model

Figure 4 demonstrates the validity of the proposed VBASBS model. Under the assumptions on the arrival rates and service times, it shows quite a monotonically increasing trend of the average number of slots in a block as a representation of the number of transactions in a normalized manner.

Notice that the average number of slots ultimately represents the population in the block on average no matter how many transactions they belong to. In fact, each state P_i, $0 \leq i \leq n$, represents a transaction with i number of slots and its steady state probability represents the normalized likelihood of the number of transactions of the size of i. Thus, it is claimed that $L_Q = \sum_{i=0}^{n} iP_i$, has a valid representation of the average number of transactions in terms of the average number of slots. In fact, tracking the number of slots facilitates the process of tracking the number of transactions which otherwise would be complicated to track due to the variability of the sizes of the transactions. Also, note that the arrival rates of lots (cf. transactions) at the transaction pool (i.e., L) and at the block (i.e., L_Q) are assumed to be identical in this simulation, for simplicity purpose, which might have been assumed differently if mandated to do so for practicality purpose. The following observations are drawn from the simulation results in Figure 4: as the size of the block increases, the average number of slots in the mined transactions to be posted in the block increases slower as the arrival rate (i.e., λ) decreases as expected and intuitively as well; the unpopulated portion on average (i.e., $n - L_Q$) is narrowing as the size of the block grows, which is to do with the level of the arrival rate such that the higher the arrival rate goes, the narrower the unpopulated portion turns; further, notice that L_Q grows monotonically without a sign of saturation and it is speculated that the monotonicity is expected as the block is modeled to be purged as soon as the number of slots in the mined transactions to be posted on the block hits n, which does not lose any generality from the standpoint of a queue of mined transactions to be

posted on a block as is the underlying assumption of the proposed VBASBS model; and lastly, notice that as the arrival rate grows higher, the growth rate of L_Q slows.

Note that an assumption is made for simplicity such that the block is to be purged back to 0 slot status exactly when it hits n without any consideration of non-full block posting, yet in practice, it is not impossible to have a non-full block to be posted when, for instance, a huge transaction is mined and might span across two or more number of blocks, which is left in this work as a future work to be addressed and resolved.

The following graph plots W_Q (i.e., from Equation (26)) versus n for given pairs of λ and μ.

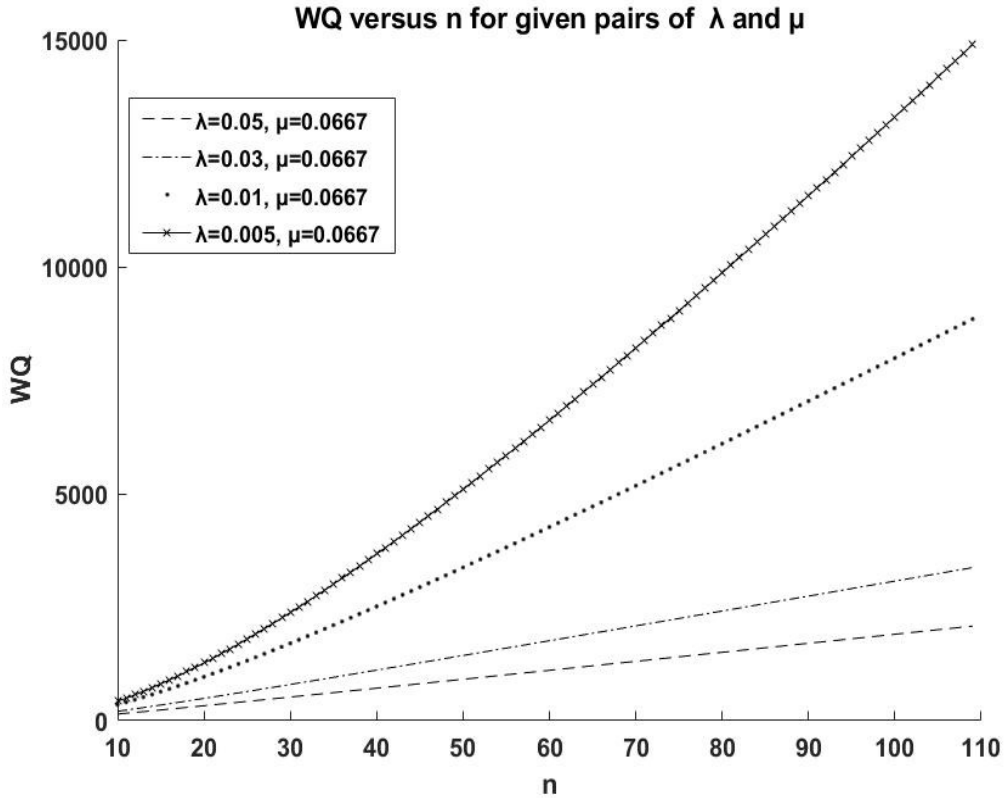

Figure 5. W_Q versus n for given pairs of λ and μ in the baseline chain model

As the proposed VBASBS model has been validated in the simulation as shown in Figure 4 without loss of intuition, Figure 5 also demonstrates the average waiting time of the mined transactions (or slots) for the posting on the block is proportional to $\frac{L_Q}{\lambda}$ in a monotonic manner. It is observed that for a given size of the block, the waiting time picks up as the arrival rate λ decreases; and the growth rate of the waiting time steepens as the arrival rate λ decreases as well.

Figure 6 plots W (i.e., from Equation (25)) versus n for given pairs of λ and μ.

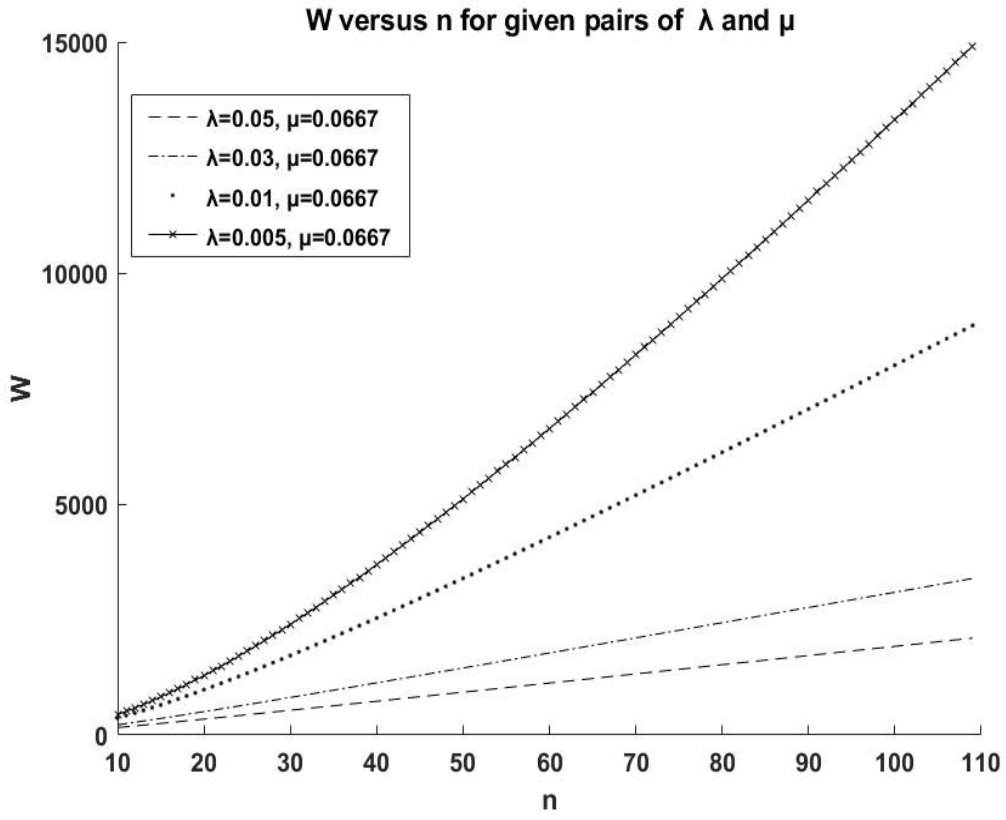

Figure 6. W versus n for given pairs of λ and μ in the baseline chain model

Figure 6 demonstrates the waiting time of the pending transactions (in terms of slots) in the transactions pool for the mining selection for the block, which is determined by $W_Q + \frac{1}{\mu}$ and it is observed to be just a matter as much as $\frac{1}{\mu}$ added to W_Q resulting in a slight increase in time.

The following graph plots L (i.e., from Equation (27)) versus n for given pairs of λ and μ.

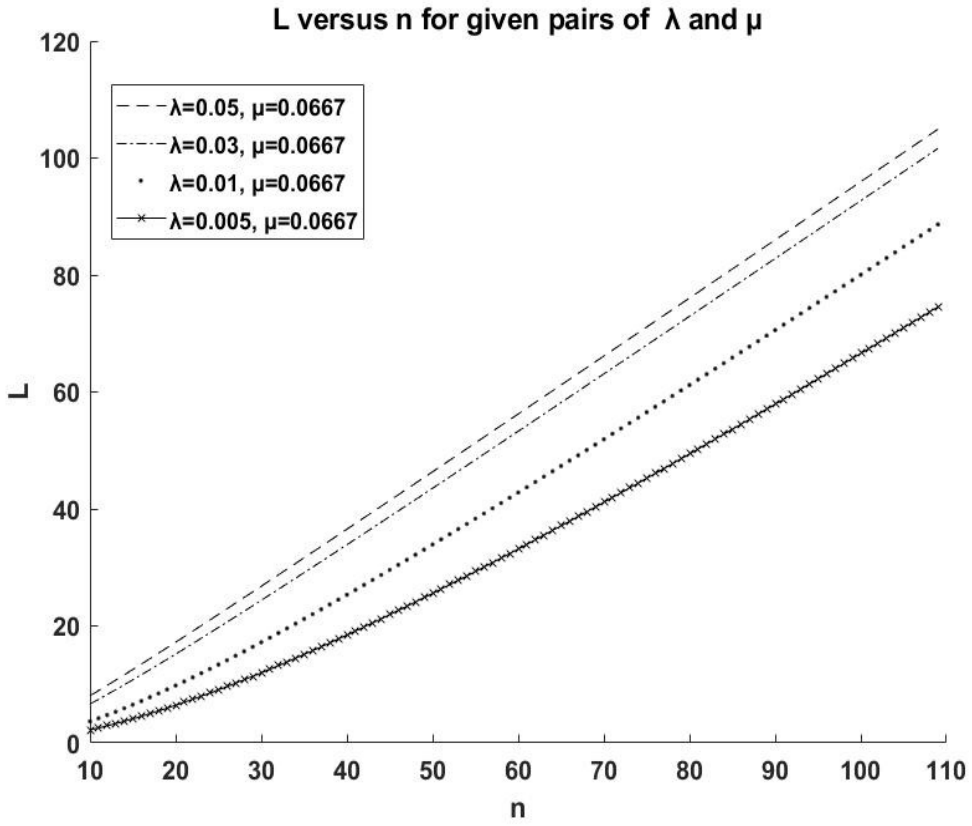

Figure 7. L versus n for given pairs of λ and μ in the baseline chain model

Figure 7 demonstrates the average number of transactions in the transaction pool waiting for mining selection and is determined by $L = \lambda W$. It is observed that at the given arrival rates of λ, L is set to be slightly higher than L_Q at a given n, which is speculated such that the identical arrival rates of λ as assumed accounts for it.

The throughput per block in the VBASBS model can be obtained in Equation (28).

$$\gamma = \mu P_n = \mu \frac{\lambda}{\mu} \frac{n(n+1)}{2} P_0 = \lambda \frac{n(n+1)}{2} P_0 \tag{28}$$

The following graph plots γ versus n for given pairs of λ and μ. In Figure 8, it is observed that as the size of the block (i.e., n) grows, γ increases; γ is independent of μ and solely affected by λ; and the higher λ picks up, the higher throughput achieved for a given size of the block, n.

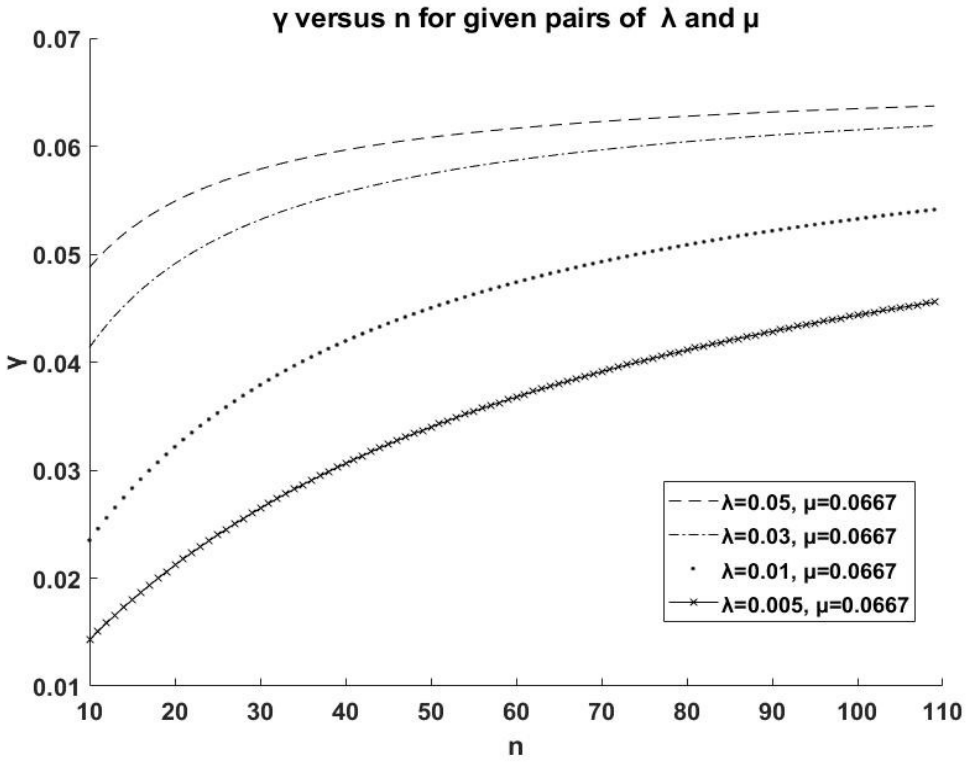

Figure 8. γ versus n for given pairs of λ and μ in the baseline chain model

CHAPTER IV

ADAPTIVE CHAIN MODEL

This chapter proposes an adaptive chain and presents a Variable Bulk Arrival and Variable Bulk Service (VBAVBS) queueing model of the $M^{1,n}/M^{1,i,n}/1$ type in order to provide a quantitative method to design an adaptive chain. The queueing model (i.e., VBASBS) of the type $M^{1,n}/M^{n}/1$ in Chapter III serves as the baseline for the proposed new model for the adaptive chain. The adaptive model assumes variable bulk arrivals of transactions in Poisson distribution, i.e., $M^{1,n}$, where n represents the number of slots across all the mined transactions, and variable bulk services of transactions, each of which applies to a block potentially of different capacity in terms of the number of slots in it, in exponential time, i.e., $M^{1,i,n}$, for posting in the current block, namely, VBAVBS. The difference between VBASBS and VBAVBS is that in VBASBS, the state P_n is the only state to transition back into P_0 (i.e., the capacity of the block is constant throughout) while in VBAVBS, every state P_i, $0 < i \leq n$, potentially transitions back into P_0 (i.e., the capacity of the block is variable adapting to a certain criteria as desired, e.g., asynchronous transactions control and any other stringent requirements imposed on the execution time of transactions), which is the major contribution of this research.

VBAVBS will reveal the performance advantages of the adaptive chain versus the baseline chain, i.e., VBASBS, with respect to the average time for a slot to stay (or wait) in the block and the average spatial requirement by the slots in the block. Numerical simulations are conducted on Matlab to compute the models and a comparative study will be demonstrated on VBAVBS versus VBASBS.

The state for the adaptive chain is basically identical to and only the state transition probabilities differ from the ones in VBASBS. The random variables employed to express the state transition rates are shown in Figure 9 and defined as follows.

P_0 : the state in which there is no transaction (i.e., no slot) arrived in the queue as of yet for the posting in the block, currently.

P_n : the state in which there is n number of slots (i.e., which is 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.

P_i : the state in which there is i number of slots (where $0 < i < n$) arrived in the queue for the posting in the block, currently.E

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, 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.

μ: the rate for the slots of the transactions in the entire queue to be posted and purged. Notice that this is an every state transition precisely from P_n back to P_0, P_{n-1} back to P_0,…, P_2 back to P_0, and P_1 back to P_0 with $\frac{\mu}{n-(n-1)}, \frac{\mu}{n-((n-1)-1)}, \ldots, \frac{\mu}{n-1}$, and $\frac{\mu}{n}$ respectively. The balance equations for the adaptive chain model are as follows.

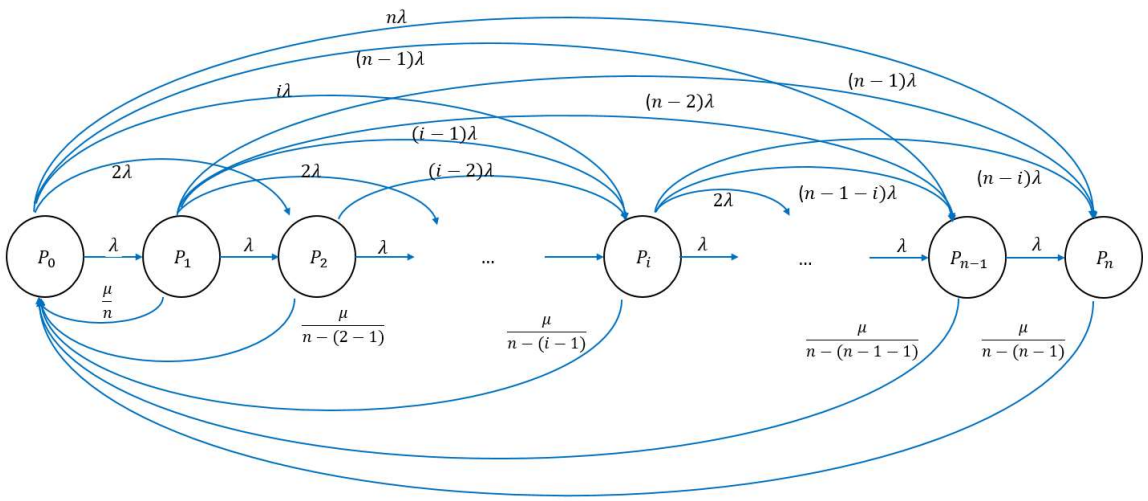

Figure 9. State transition diagram of the adaptive chain model

1. Adaptive Chain Model Equations

The balance equations for VBAVBS are as follows.

$$(\lambda + 2\lambda + 3\lambda + \cdots + n\lambda)P_0 = \lambda\frac{n(n+1)}{2}P_0$$

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

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

$$\left(\lambda + 2\lambda + 3\lambda + \cdots + (n-i)\lambda + \frac{\mu}{n-(i-1)}\right)P_i$$

$$= \lambda P_{i-1} + 2\lambda P_{i-2} + 3\lambda P_{i-3} + (i-1)\lambda P_1 + i\lambda P_0$$

Where,

$$0 < i \leq n,$$

Figure 10 shows the state transition diagram around P_i of adaptive chain model. All outgoing transitions equal all incoming transitions for P_i.

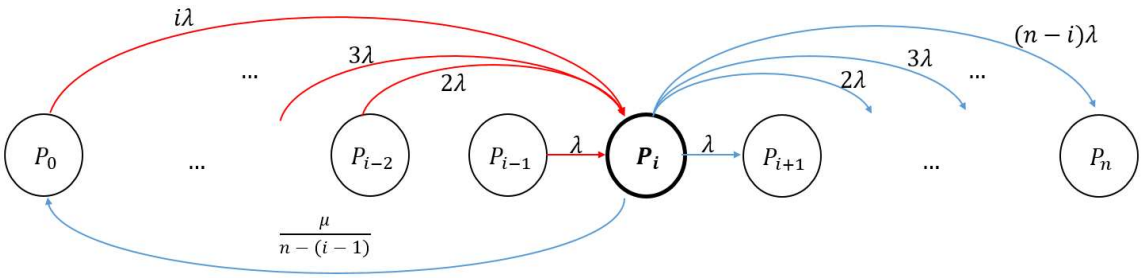

Figure 10. State transition diagram around P_i of the adaptive chain model

It can be expressed as $i = 1, 2, 3, \dots, n$, and find a generalized form as the following steps.

$i = 1,$

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

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

$$P_1 = {q_1}^{-1} P_0$$

$$q_1 = \frac{(n-1)(n)}{2} + \frac{\mu}{\lambda(n)}$$

$i = 2,$

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

Where,

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

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

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

$$q_2 = \frac{(n-2)(n-1)}{2} + \frac{\mu}{\lambda(n-2)}$$

$$P_2 = q_2^{-1}(q_1^{-1}P_0 + 2P_0) = q_2^{-1}(q_1^{-1} + 2)P_0$$

$$P_2 = q_2^{-1}(q_1^{-1} + 2)P_0$$

$i = 3,$

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

Where,

$$P_1 = q_1^{-1}P_0$$

$$P_2 = q_2^{-1}(q_1^{-1} + 2)P_0$$

$$P_3 = q_3^{-1}\left(q_2^{-1}(q_1^{-1} + 2) + 2q_1^{-1} + 3\right)P_0$$

$$q_3 = \frac{(n-3)(n-2)}{2} + \frac{\mu}{\lambda(n-2)}$$

$i = 4,$

$$\lambda\left(\frac{(n-4)(n-3)}{2} + \frac{\mu}{\lambda(n-3)}\right)P_4 = \lambda P_3 + 2\lambda P_2 + 3\lambda P_1 + 4\lambda P_0$$

Where,

$$P_1 = q_1^{-1}P_0$$

$$P_2 = q_2^{-1}(q_1^{-1} + 2)P_0$$

$$P_3 = q_3^{-1}\left(q_2^{-1}(q_1^{-1} + 2) + 2q_1^{-1} + 3\right)P_0$$

$$P_4 = q_4{}^{-1}\left(q_3{}^{-1}\left(q_2{}^{-1}(q_1{}^{-1} + 2) + 2q_1{}^{-1} + 3\right) + 2(q_2{}^{-1}(q_1{}^{-1} + 2) + 3q_1{}^{-1} + 4\right)P_0$$

$$P_4 = q_4{}^{-1}\left(q_3{}^{-1}q_2{}^{-1}q_1^{-1} + 2q_3{}^{-1}q_2{}^{-1} + 2q_3{}^{-1}q_1{}^{-1} + 3q_3{}^{-1} + 2q_2{}^{-1}\, q_1{}^{-1} + 4q_2{}^{-1} + 3q_1{}^{-1} + 4\right)P_0$$

$$q_4 = \frac{(n-4)(n-3)}{2} + \frac{\mu}{\lambda(n-3)}$$

The generalized form is as follows.

$$\lambda\left(\frac{(n-i)(n-i+1)}{2} + \frac{\mu}{\lambda(n-i+1)}\right)P_i$$

$$= \lambda P_{i-1} + 2\lambda P_{i-2} + \cdots + (i-1)\lambda P_{i-(i-1)} + i\lambda P_{i-i} \tag{29}$$

Where,

$$0 < i \leq n,$$

$$q_i = \frac{(n-i)(n-i+1)}{2} + \frac{\mu}{\lambda(n-i+1)}$$

From Equation (29), it can get P_i as following expressions.

$$\lambda q_i P_i = \lambda P_{i-1} + 2\lambda P_{i-2} + \cdots + (i-1)\lambda P_{i-(i-1)} + i\lambda P_{i-i}$$

$$P_i = q_i^{-1}\left(P_{i-1} + 2P_{i-2} + \cdots + (i-1)P_{i-(i-1)} + iP_{i-i}\right)$$

From Equation (29), all outgoing transitions from P_i equal all incoming transitions to P_i, which can be expressed as follows.

$$\left(\lambda + 2\lambda + 3\lambda + \cdots + (n-i)\lambda + \frac{\mu}{n-(i-1)}\right) P_i$$

$$= \lambda P_{i-1} + 2\lambda P_{i-2} + 3\lambda P_{i-3} + \cdots (i-1)\lambda P_1 + i\lambda P_0$$

$$\left(\lambda\left(\frac{(n-i)(n-i+1)}{2}\right) + \frac{\mu}{n-(i-1)}\right) P_i$$

$$= \lambda P_{i-1} + 2\lambda P_{i-2} + 3\lambda P_{i-3} + \cdots (i-1)\lambda P_1 + i\lambda P_0$$

From all the way steps above, it can be a generalized form of P_i as follows.

$$P_i = q_i^{-1} P_0 \left[\sum_{j=1}^{i} j \left[\sum_{k=1}^{i-1} k \left[\prod_{l=1}^{k-1} q_l^{-1}\right]\right] + i\right] \tag{30}$$

Where,

$$0 < i \leq n$$

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

$$= \frac{2\lambda(n-i+1)}{(n-i)\lambda(n-i+1)^2 + 2\mu} \tag{31}$$

It can be expressed by P_i by inserting q_i^{-1} Equation (31) into Equation (30) as following Equations (32).

$$P_i$$

$$= \left(\frac{2\lambda(n-i+1)}{(n-i)\lambda(n-i+1)^2+2\mu}\right) P_0 \left[\sum_{j=1}^{i} j \left[\sum_{k=1}^{i-1} k \left[\prod_{l=1}^{k-1} \left(\frac{2\lambda(n-i+1)}{(n-i)\lambda(n-i+1)^2+2\mu}\right)\right]\right] + i\right] \quad (32)$$

From Equation (8), it can be expressed in Equation (33) and (34).

$$P_0 \left(1 + \sum_{i=1}^{n} P_i\right) = 1 \quad (33)$$

$$P_0 = \frac{1}{\left(1 + \sum_{i=1}^{n} P_i\right)} \quad (34)$$

It shows transitions from P_n with μ to P_0 equals all outgoing transitions from P_0 with λ to P_n in Equation (35) and (36).

$$\frac{\mu}{n-(n-1)} P_n = \lambda \frac{n(n+1)}{2} P_0 \quad (35)$$

$$P_n = \frac{\lambda}{\mu} \frac{n(n+1)}{2} P_0 \quad (36)$$

Then, the following Equation (37) and Equation (38) can be used to compute P_0.

$$P_0 + P_1 + P_2 + \cdots P_{n-1} + P_n = 1 \quad (37)$$

It can be expressed as follows.

$$P_0 + \sum_{i=1}^{n-1} P_i + P_n = 1 \quad (38)$$

From the generalized expression P_i, the first inner production calculation is shown in the following Equations. The calculation inner product is expressed as a closed-form solution, but there are two imaginary numbers and one real number from the cubic equation denominated in the product equation. It shows three steps to solve the generalized form as following Equations (39), (40), and (41).

First, it is described to find a solution as a closed-form as follows.

$$\prod_{l=1}^{k-1} q_l^{-1}$$

$$= \prod_{l=1}^{k-1} \left(\frac{2\lambda(n-l+1)}{(n-l)\lambda(n-l+1)^2 + 2\mu} \right)$$

$$= \prod_{l=1}^{k-1} \left(\frac{2\lambda(n-l+1)}{-\lambda l^3 + \lambda l^2(3n+2) - \lambda l(n^2-4n-1) + \lambda n^3 + 2\lambda n^2 + \lambda n + 2\mu} \right) \quad (39)$$

$$= \frac{\prod_{l=1}^{k-1} 2\lambda(n-l+1)}{-\lambda \prod_{l=1}^{k-1}(l-r_1) \prod_{l=1}^{k-1}(l-r_2) \prod_{l=1}^{k-1}(l-r_3)}$$

$$= \frac{-2\dfrac{n!}{\big(n-(k-1)\big)!}}{\dfrac{(1-r_1)!}{(k-r_1)!}\dfrac{(1-r_2)!}{(k-r_2)!}\dfrac{(1-r_3)!}{(k-r_3)!}}$$

$$= \frac{1}{\dfrac{(1-r_1)!}{(-n+1-r_1)!}\dfrac{(1-r_2)!}{(-n+1-r_2)!}\dfrac{(1-r_3)!}{(-n+1-r_3)!}} \quad (40)$$

Where,

Taking a partial fraction of the cubic equation from the denominator and the nominator is to be 1 as follows.

$$-2\frac{n!}{\big(n-(k-1)\big)!}=1$$

$$\frac{n!}{\big(n-(k-1)\big)!}=-\frac{1}{2}$$

$$k=-n+1 \tag{41}$$

By Equation (41), the value k is applied to Equation (40).

Second, the calculation of the outer sum of the first product is shown as follows.

$$\sum_{k=1}^{i-1} k\left[\frac{1}{\left(\frac{(1-r_1)!}{(-n+1-r_1)!}\right)\left(\frac{(1-r_2)!}{(-n+1-r_2)}\right)\left(\frac{(1-r_3)!}{(-n+1-r_3)!}\right)}\right]$$

$$=\frac{(i-1)i}{2}\left(\frac{1}{\left(\frac{(1-r_1)!}{(-n+1-r_1)!}\right)\left(\frac{(1-r_2)!}{(-n+1-r_2)}\right)\left(\frac{(1-r_3)!}{(-n+1-r_3)!}\right)}\right)$$

Third, as a final step, the outside sum of the given equation can be solved in Equation (42).

$$\sum_{j=1}^{i} j\left[\frac{(i-1)i}{2}\left(\frac{1}{\left(\frac{(1-r_1)!}{(-n+1-r_1)!}\right)\left(\frac{(1-r_2)!}{(-n+1-r_2)}\right)\left(\frac{(1-r_3)!}{(-n+1-r_3)!}\right)}\right)\right]$$

$$= \frac{i(i+1)}{2}\left(\frac{(i-1)i}{2}\left(\frac{1}{\left(\frac{(1-r_1)!}{(-n+1-r_1)!}\right)\left(\frac{(1-r_2)!}{(-n+1-r_2)}\right)\left(\frac{(1-r_3)!}{(-n+1-r_3)!}\right)}\right)\right)$$

$$= \frac{i^4 - i^2}{4}\left(\frac{1}{\left(\frac{(1-r_1)!}{(-n+1-r_1)!}\right)\left(\frac{(1-r_2)!}{(-n+1-r_2)}\right)\left(\frac{(1-r_3)!}{(-n+1-r_3)!}\right)}\right) \quad (42)$$

$r_1, r_2,$ and r_3 can be formed from the following equations such that the denominator, Equation (42), $\frac{(1-r_1)!}{(-n+1-r_1)!}\frac{(1-r_2)!}{(-n+1-r_2)!}$, and $\frac{(1-r_3)!}{(-n+1-r_3)!}$ can be obtained from the cubic formula that has three roots. One real number is r_1 and two imaginary numbers are r_2 and r_3. Taking the Stirling's approximation, the denominator in Equation (42) can be obtained as follows.

$$(1-r_1)! = \sqrt{2\pi(1-r_1)}\left(\frac{1-r_1}{e}\right)^{(1-r_1)}$$

$$(1-r_2)! = \sqrt{2\pi(1-r_2)}\left(\frac{1-r_2}{e}\right)^{(1-r_2)}$$

$$(1-r_3)! = \sqrt{2\pi(1-r_3)}\left(\frac{1-r_3}{e}\right)^{(1-r_3)}$$

$$(-n+1-r_1)! = \sqrt{2\pi(-n+1-r_1)}\left(\frac{(-n+1-r_1)}{e}\right)^{(-n+1-r_1)}$$

$$(-n+1-r_2)! = \sqrt{2\pi(-n+1-r_2)}\left(\frac{(-n+1-r_2)}{e}\right)^{(-n+1-r_2)}$$

$$(-n+1-r_3)! = \sqrt{2\pi(-n+1-r_3)}\left(\frac{(-n+1-r_3)}{e}\right)^{(-n+1-r_3)}$$

From the above equations, r_1, r_2, and r_3 can be further derived by the following method.

$$r_1 = S + T - \frac{b}{3a}$$

$$r_2 = -\frac{S+T}{2} - \frac{b}{3a} + \frac{i\sqrt{3}}{2}(S - T)$$

$$r_3 = -\frac{S+T}{2} - \frac{b}{3a} - \frac{i\sqrt{3}}{2}(S - T)$$

Where,

$$a = -\lambda$$

$$b = \lambda(3n + 1)$$

$$c = -\lambda(n^2 - 4n - 1)$$

$$d = \lambda n^3 + 2\lambda n^2 + \lambda n + 2\mu$$

$$S = \sqrt[3]{R + \sqrt{Q^3 + R^2}}$$

$$T = \sqrt[3]{R - \sqrt{Q^3 + R^2}}$$

$$Q = \frac{3ac - b^2}{9a^2}$$

$$R = \frac{9abc - 27a^2d - 2b^3}{54a^3}$$

By given the coefficient of the cubic equation, it is generated the Q, R, S, T, r_1, r_2, and r_3 as following Equations (43), (44), (45), (46), (47), (48), and (49) respectively.

$$Q = \frac{3(-\lambda)\left(-\lambda(n^2 - 4n - 1)\right) - \left(\lambda(3n + 1)\right)^2}{9(-\lambda)^2}$$

$$= \frac{3\lambda^2(n^2 - 4n - 1) - \left(\lambda^2(9n^2 + 6n + 1)\right)}{9\lambda^2}$$

$$= \frac{(3\lambda^2 n^2 - 3\lambda^2 4n - 3\lambda^2) - 9n^2\lambda^2 - 6n\lambda^2 - \lambda^2}{9\lambda^2}$$

$$= \frac{-6\lambda^2 n^2 - 18\lambda^2 n - 4\lambda^2}{9\lambda^2} \tag{43}$$

$$R = \frac{9(-\lambda)\left(\lambda(3n+1)\right)\left(-\lambda\left(n^2-4n-1\right)\right)-27(-\lambda)^2\left(\lambda n^3+2\lambda n^2+\lambda n+2\mu\right)-2\left(\lambda(3n+1)\right)^3}{54(-\lambda)^3}$$

$$= \frac{\left(9\lambda^3\left(3n^3-11n^2-7n-1\right)\right)-27\lambda^2\left(\lambda n^3+2\lambda n^2+\lambda n+2\ \ \right)-2\lambda^3\left(27n^3+27n^2+9n+1\right)}{-54\lambda^3}$$

$$= \frac{\left(27\ \ ^3n^3-99\lambda^3n^2-63\lambda^3n-9\lambda^3\right)+\left(-27\lambda^3n^3-54\lambda^3n^2-27\lambda^3n-54\lambda^2\mu\right)+\left(-54\lambda^3n^3-54\lambda^3n^2-18\lambda^3n-2\lambda^3\right)}{-54\lambda^3}$$

$$= \frac{\left(-207\lambda^3n^2-118\lambda^3n-11\lambda^3\right)+\left(-54\lambda^2\mu\right)+\left(-54\lambda^3n^3\right)}{-54\lambda^3} \tag{44}$$

$$S = \sqrt[3]{\frac{\left(-207\lambda^3n^2-118\lambda^3n-11\lambda^3\right)+\left(-54\lambda^2\mu\right)+\left(-54\lambda^3n^3\right)}{-54\lambda^3}}$$

$$+\sqrt[3]{\sqrt{\left(\frac{-6\lambda^2n^2-18\lambda^2n-4\lambda^2}{9\lambda^2}\right)^3+\left(\frac{\left(-207\lambda^3n^2-118\lambda^3n-11\lambda^3\right)+\left(-54\lambda^2\mu\right)+\left(-54\lambda^3n^3\right)}{-54\lambda^3}\right)^2}} \tag{45}$$

$$T = \sqrt[3]{\frac{\left(-207\lambda^3n^2-118\lambda^3n-11\lambda^3\right)+\left(-54\lambda^2\mu\right)+\left(-54\lambda^3n^3\right)}{-54\lambda^3}}$$

$$-\sqrt[3]{\sqrt{\left(\frac{-6\lambda^2n^2-18\lambda^2n-4\lambda^2}{9\lambda^2}\right)^3+\left(\frac{(-207\lambda^3n^2-118\lambda^3n-11\lambda^3)+(-54\lambda^2\mu)+(-54\lambda^3n^3)}{-54\lambda^3}\right)^2}} \tag{46}$$

$$r_1 = S + T - \frac{\lambda(3n+1)}{3(-\lambda)} \tag{47}$$

$$r_2 = -\frac{1}{2}(S+T) - \frac{\lambda(3n+1)}{3(-\lambda)} + \frac{i\sqrt{3}}{2}(S-T) \tag{48}$$

$$r_3 = -\frac{1}{2}(S+T) - \frac{\lambda(3n+1)}{3(-\lambda)} - \frac{i\sqrt{3}}{2}(S-T) \tag{49}$$

Now, P_i can be expressed as follows.

$$P_i = \frac{2\lambda(n-i+1)}{(n-i)\lambda(n-i+1)^2+2\mu}P_0\left[\frac{i^4-i^2}{4}\left(\frac{1}{ABC}\right)+i\right] \tag{50}$$

Where,

$$A = \frac{\sqrt{2\pi(1-r_1)}\left(\frac{1-r_1}{e}\right)^{(1-r_1)}}{\sqrt{2\pi(-n+1-r_1)}\left(\frac{(-n+1-r_1)}{e}\right)^{(-n+1-r_1)}}$$

$$B = \frac{\sqrt{2\pi(1-r_2)}\left(\frac{1-r_2}{e}\right)^{(1-r_2)}}{\sqrt{2\pi(-n+1-r_2)}\left(\frac{(-n+1-r_2)}{e}\right)^{(-n+1-r_2)}}$$

$$C = \frac{\sqrt{2\pi(1-r_3)}\left(\frac{1-r_3}{e}\right)^{(1-r_3)}}{\sqrt{2\pi(-n+1-r_3)}\left(\frac{(-n+1-r_3)}{e}\right)^{(-n+1-r_3)}}$$

Due to the two imaginary roots from the cubic formula equation, it has been shown to express all the steps to find a closed-form that has two imaginary roots in the closed-form in Equation (50). To avoid the imaginary value number in the result, it simulated the formula before going to get the two imaginary roots in Equation (48). From Equation (38), $P_0 + \sum_{i=1}^{n-1} P_i + P_n = 1$ and P_0 can be solved as shown in Equations (51), (52), and (53).

$$P_0\left(1 + (q_1\left[\sum_{j=1}^{1} j + 1\right]) + (q_2\left[\sum_{j=1}^{2} j\left[\sum_{k=1}^{i-1}\left[\prod_{l=1}^{k-1} q_l\right]k\right] + 2\right]) + \cdots\right.$$

$$\left. + (q_{n-1}\left[\sum_{j=1}^{n-1} j\left[\sum_{k=1}^{n-2}\left[\prod_{l=1}^{k-1} q_l\right]k\right] + 2\right]) + \frac{\lambda}{\mu}\left(\frac{n(n+1)}{2}\right)\right) = 1 \quad (51)$$

$$P_0\left(1 + \sum_{i=1}^{n-1}\left(q_i\left[\sum_{j=1}^{i} j\left[\sum_{k=1}^{i-1}\left[\prod_{l=1}^{k-1} q_l\right]k\right] + i\right]\right) + \frac{\lambda}{\mu}\left(\frac{n(n+1)}{2}\right)\right) = 1 \quad (52)$$

$$P_0 = \frac{1}{\left[1 + \sum_{i=1}^{n-1}\left(q_i\left[\sum_{j=1}^{i} j\left[\sum_{k=1}^{i-1}\left[\prod_{l=1}^{k-1} q_l\right]k\right] + i\right]\right) + \frac{\lambda}{\mu}\left(\frac{n(n+1)}{2}\right)\right]} \quad (53)$$

The performance measurements of primary interests in the adaptive model are expressed in L_Q, W_Q, W, L, and γ as following Equations respectively.

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).

$$L_Q = \sum_{i=0}^{n} iP_i$$

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

$$W_Q = \frac{L_Q}{\lambda}$$

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

$$W = W_Q + \frac{1}{\mu}$$

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).

$$L = \lambda W$$

γ: the throughput of the average number of customers (i.e., equivalently, the average number of transactions) arrives and services (posting a block) through the adaptive chain model in the system in Equation (54).

$$\gamma = \sum_{i=0}^{n} \frac{\mu}{n-(i-1)} P_i \tag{54}$$

Where,

P_i is Equation (50)

2. Numerical Analysis

The following graph plots L_Q (i.e., from Equation (38)) on the comparison of the baseline model versus the adaptive model in n for given pairs of λ and μ. Observe that as λ goes 0.005 to 0.05, at μ=0.0667, the ratio of L_Q between adaptive vs. baseline swings from 85%, 69%, 63%, and 67% to 18%, 21%, 31%, and 39%, respectively, as n increases in Figure 11.

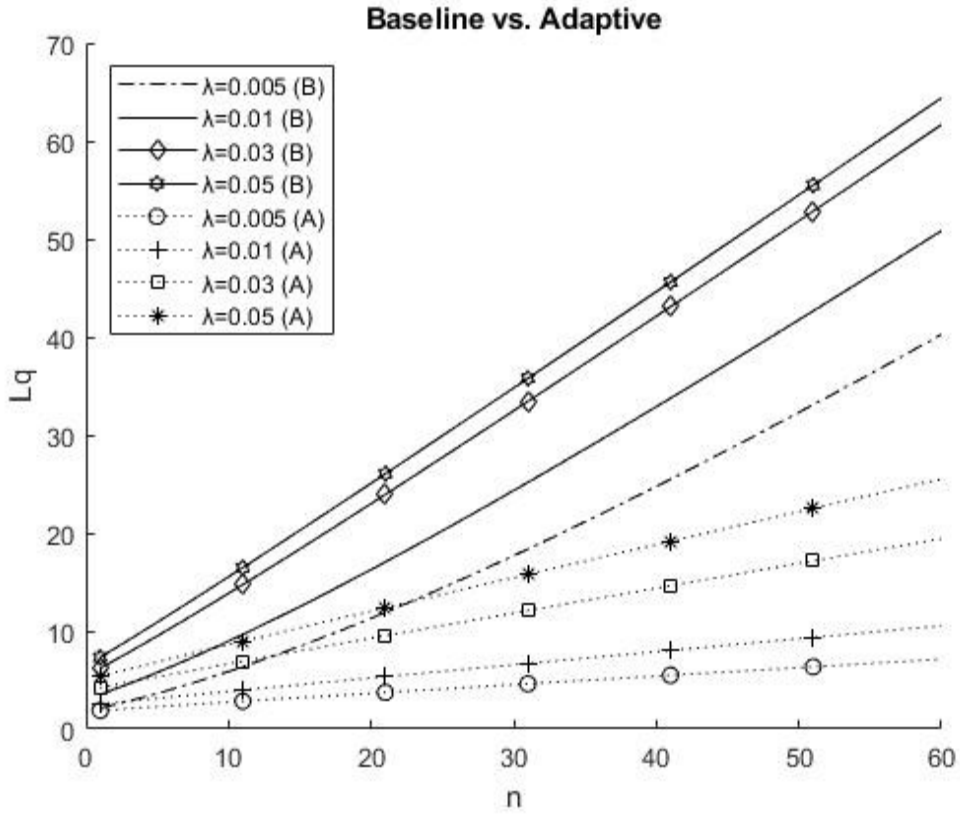

Figure 11. L_Q comparing baseline (B) and adaptive (A) chain model

The following graph plots W_Q (i.e., from Equation (25)) on the comparison of the baseline model versus the adaptive model in n for given pairs of λ and μ. Observe that as λ goes 0.005 to 0.05, at μ=0.0667, the ratio of W_Q between adaptive vs. baseline swings from 85%, 68%, 61%, and 63% to 17%, 20%, 31%, and 38%, respectively, as n increases in Figure 12.

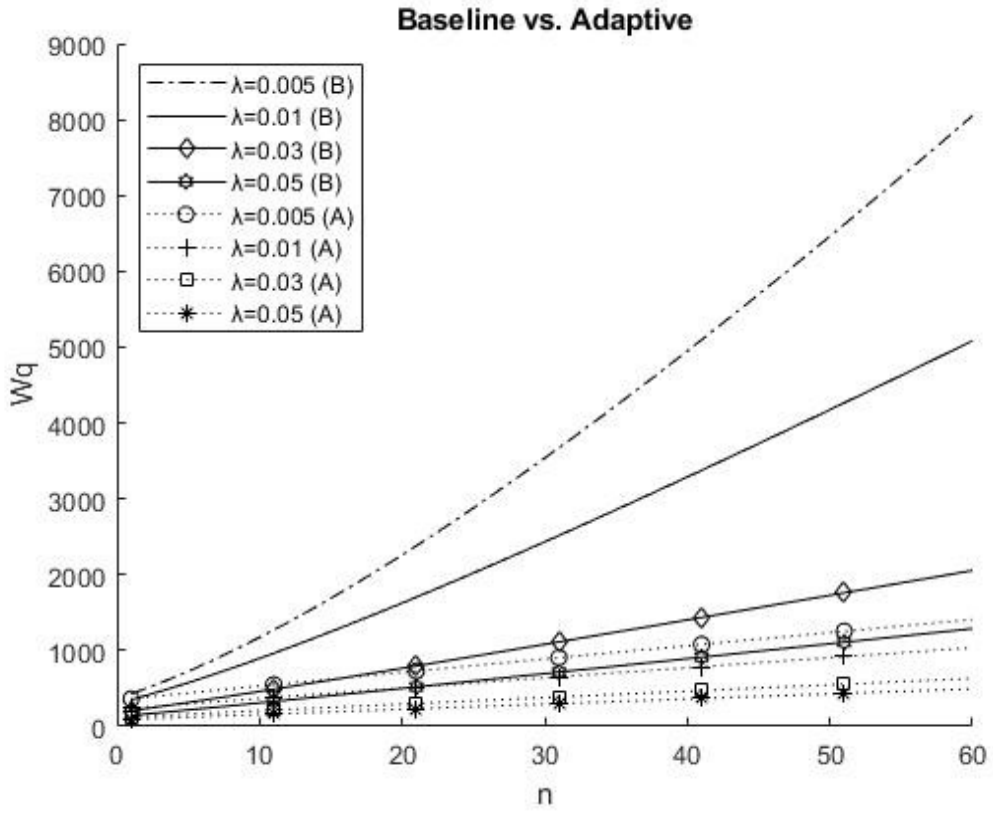

Figure 12. W_Q comparing baseline (B) and adaptive (A) chain model

The following graph plots W (i.e., from Equation (26)) on the comparison of the baseline model versus the adaptive model in n for given pairs of λ and μ. Observe that as λ goes 0.005 to 0.05, at μ=0.0667, the ratio of W between adaptive vs. baseline swings from 85%, 69%, 63%, and 67% to 18%, 21%, 31%, and 39%, respectively, as n increases in Figure 13.

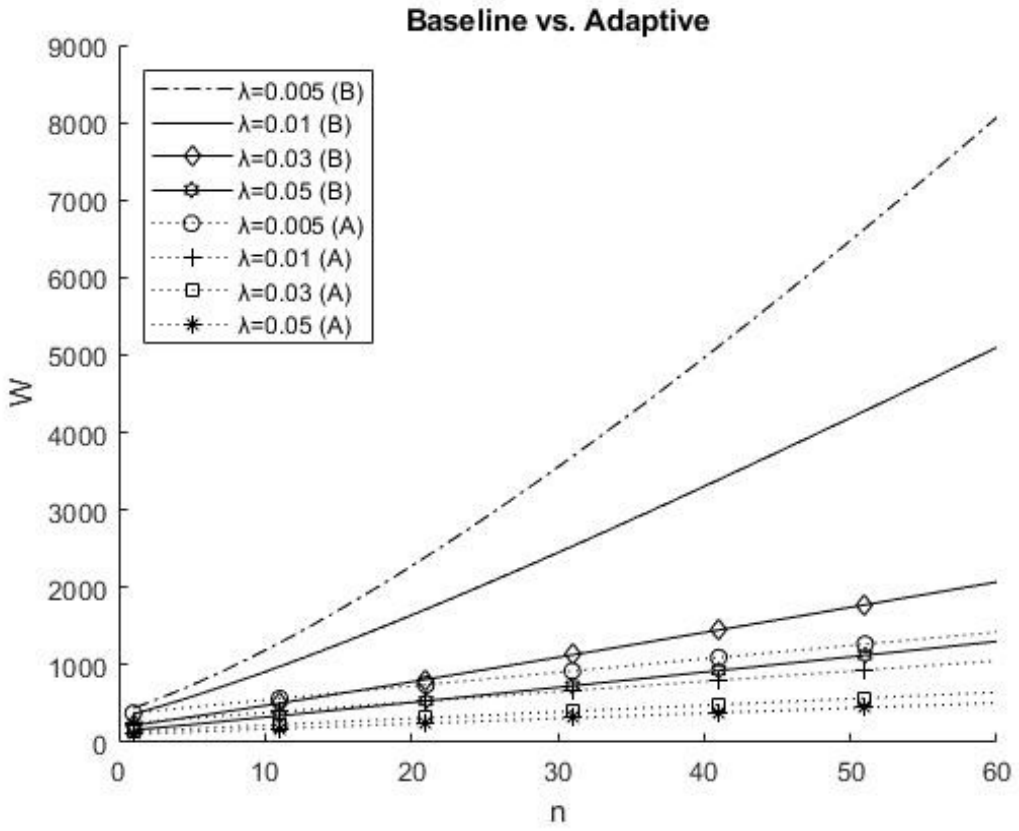

Figure 13. W comparing baseline (B) and adaptive (A) chain model

The following graph plots L (i.e., from Equation (27)) on the comparison of the baseline model versus the adaptive model in n for given pairs of λ and μ. Observe that as λ goes 0.005 to 0.05, at μ=0.0667, the ratio of L between adaptive vs. baseline swings from 85%, 69%, 63%, and 67% to 18%, 21%, 31%, and 39%, respectively, as n increases in Figure 14.

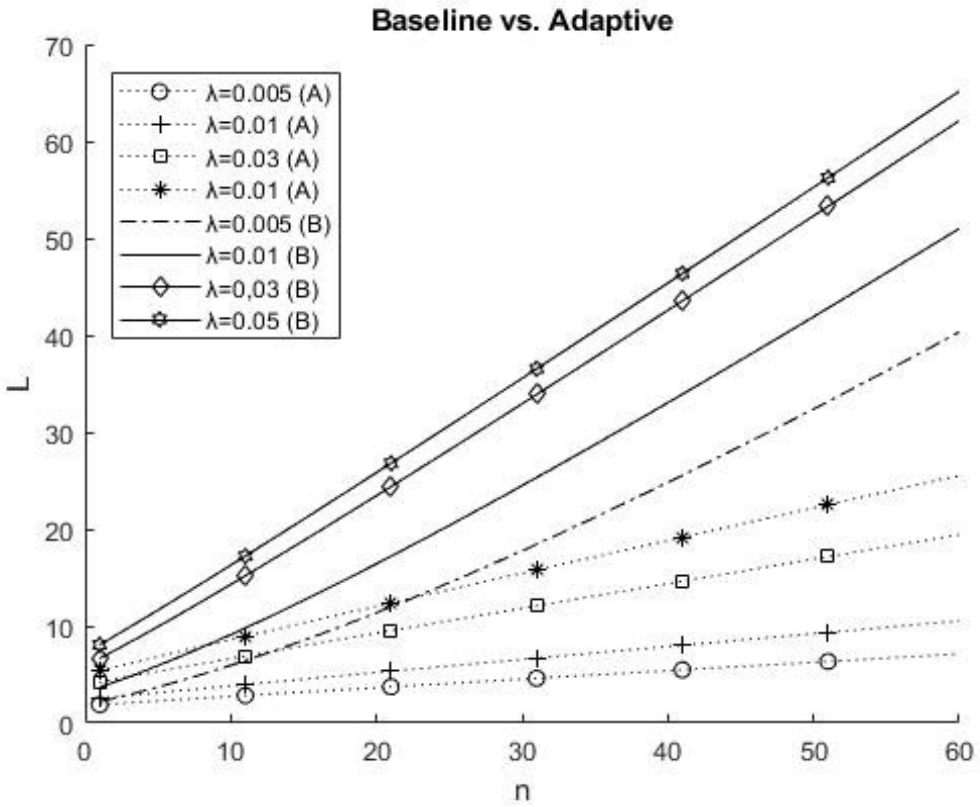

Figure 14. L comparing baseline (B) and adaptive (A) chain model

The throughput (γ) per block in the adaptive chain model can be obtained from Equation (54). The following graph plots γ versus n for given pairs of λ and μ. Observe that as λ goes 0.005 to 0.05, at μ=0.0667, the ratio of γ between adaptive vs. baseline swings from 85%, 68%, 61%, and 63% to 17%, 20%, 31%, and 38%, respectively, as n increases in Figure 15.

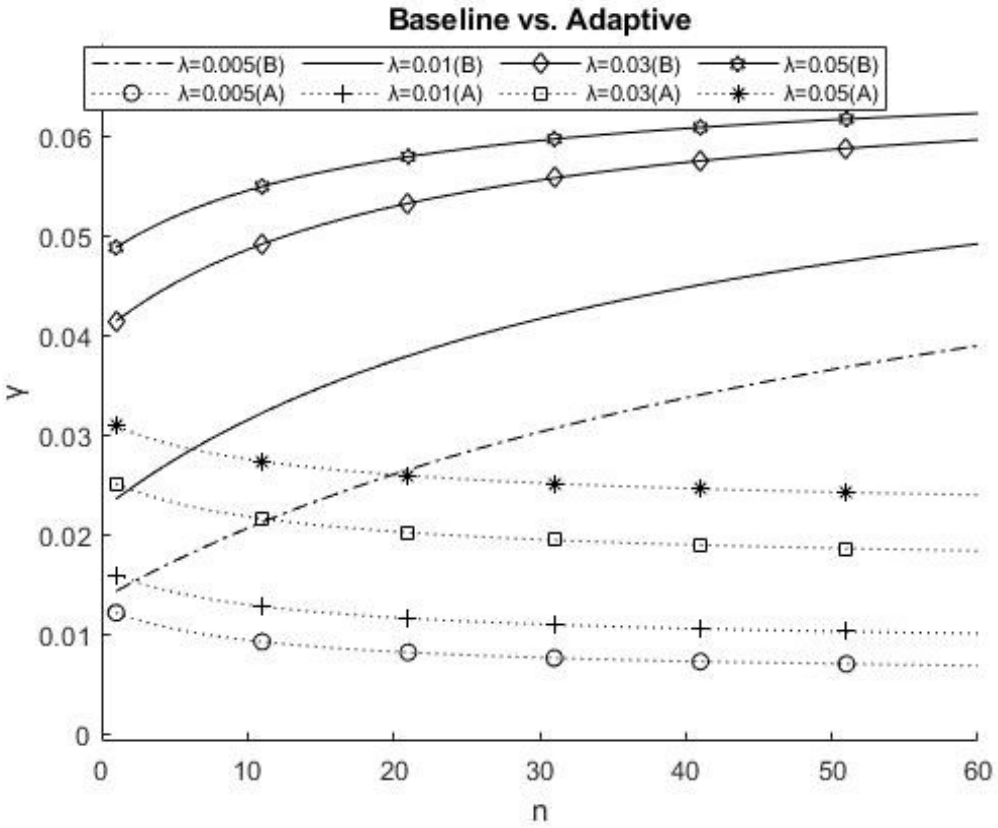

Figure 15. Throughput, γ, comparing baseline (B) and adaptive (A) chain model

CHAPTER V

ASYNCHRONOUS CHAIN MODEL

This chapter proposes an asynchronous chain and an analytical model, namely, a Variable Bulk Arrival and Asynchronous Bulk Service (VBAABS) model. It is developed and presented to study and demonstrate the theoretical performance of an asynchronous chain. The proposed asynchronous chain is to address and respond to the speed needs by such transactions as being exigently mandated to be executed in an asynchronous manner. Otherwise, all the transactions pending for a posting in the current block are synchronized by the block posting delay which is primarily determined by the gas limit on the block and the total gas used by transactions pending for the block. Two different types of asynchronous models are proposed and numerical analysis and experimental results are presented. In previous Chapters, III, and IV, the baseline chain model VBASBS and the adaptive chain model VBAVBS are presented. VBASBS and VBAVBS are the basis for the proposed asynchronous chain model VBAABS and the states can be defined in Section 1 as fully asynchronous chain model and in Section 2 as staged asynchronous chain model as follows.

1. Fully Asynchronous Chain Model Equations

VBASBS [40] and VBAVBS models are the basis for the proposed VBAABS model in this section and the states can be defined as follows. The state for the adaptive chain model, VBAVBS, is identical to and only the state transition probabilities differ from the ones in the baseline model, VBASBS.

μ: the rate for the slots of the transactions in the entire queue to be purged and posted into a block.

It is only noted in the adaptive model that it is an every state transition precisely from P_n back to P_0, P_{n-1} back to P_0, …, P_2 back to P_0, and P_1 back to P_0 with $\frac{\mu}{n-(n-1)}, \frac{\mu}{n-((n-1)-1)}, \ldots, \frac{\mu}{n-1}$, and $\frac{\mu}{n}$ respectively.

The state for the fully asynchronous chain model is that an arrived transaction is only posted to a current block. The definitions of P_0, P_n, P_i, λ, and μ are defined as follows.

P_0 : the state in which there is no transaction (i.e., no slot) arrived in the queue as of yet for the posting in the block, currently, and is onheld in the proposed VBAVBS model in this dissertation.

P_n : the state in which there is n number of slots (i.e., which is 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, and is onheld in the proposed VBAVBS model in this dissertation.

P_i : the state in which there is i number of slots (where, $0 < i < n$) arrived in the queue for the posting in the block, currently, and is onheld in the proposed VBAVBS model in this dissertation.

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, 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, which will be onheld in this dissertation as well.

μ: the rate for the slots of the transactions in the queue to be posted and purged. Notice that this is a single and unique state transition precisely from P_i, $1 \leq i \leq n$ back to P_0.

The following Figure 16 shows the state transition diagram around P_i of fully asynchronous chain model. All outgoing transitions equal all incoming transitions for every state.

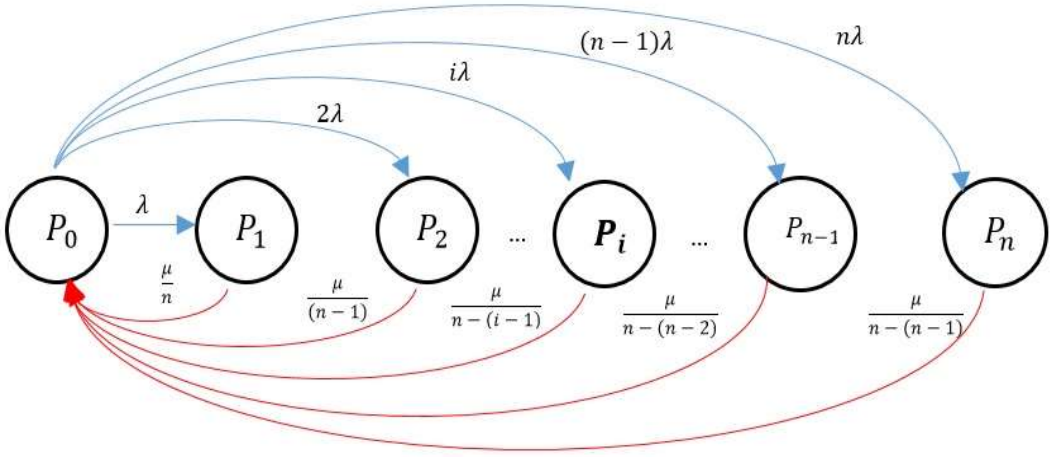

Figure 16. State transition diagram of the fully asynchronous chain model

The balance equations for the fully asynchronous chain model are as follows.

All outgoing transitions from P_0 are follows.

$$(\lambda + 2\lambda + 3\lambda + \cdots + n\lambda)P_0 = \lambda\frac{n(\mathrm{n}+1)}{2}P_0$$

All outgoing transitions from P_0 equal all incoming transitions to P_0 as follows.

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

$$= \mu\left(\frac{1}{n-0}P_1 + \frac{1}{n-1}P_2 + \frac{1}{n-2}P_3 + \cdots + \frac{1}{n-(n-2)}P_{n-1} + \frac{1}{n-(n-1)}P_n\right)$$

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

Where,

$\lambda\frac{n(\mathrm{n}+1)}{2}P_0$: all outgoing from P_0

$\frac{\mu}{n}P_1$: incoming from P_1 to P_0 with $\frac{\mu}{n}$

$\frac{\mu}{n-1}P_2$: incoming from P_2 to P_0 with $\frac{\mu}{n-1}$

$\frac{\mu}{n-2}P_3$: incoming from P_3 to P_0 with $\frac{\mu}{n-2}$

$\frac{\mu}{n-(i-1)}P_i$: incoming from P_i to P_0 with $\frac{\mu}{n-(i-1)}$

$\frac{\mu}{n-(n-1)}P_n$: incoming from P_n to P_0 with $\frac{\mu}{n-(n-1)}$

The balance equations for the asynchronous chain model are as follows.

$$(\lambda + 2\lambda + 3\lambda + \cdots + n\lambda)P_0 = \lambda\frac{n(n+1)}{2}P_0$$

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

$$= \mu\left(\frac{1}{n-0}P_1 + \frac{1}{n-1}P_2 + \cdots + \frac{1}{n-(n-2)}P_{n-1} + \frac{1}{n-(n-1)}P_n\right)$$

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

$$\left(\lambda + 2\lambda + 3\lambda + \cdots + (n-i)\lambda + \frac{\mu}{n-(i-1)}\right)P_i$$

$$= \lambda\left(\frac{(n-i)(n-i+1)}{2} + \frac{\mu}{\lambda(n-i+1)}\right)P_i$$

$$= \lambda P_{i-1} + 2\lambda P_{i-2} + 3\lambda P_{i-3} + (i-1)\lambda P_1$$

Where,

$0 < i \leq n$,

It can be expressed as $i = 1, 2, 3, \ldots, n$, and find a generalized form as the following steps.

$i = 1$,

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

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

$$P_1=\ {q_1}^{-1}P_0$$

Where,

$$q_1=\frac{(n-1)(n)}{2}+\frac{\mu}{\lambda(n)}$$

$i=2$,

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

Where,

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

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

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

$$q_2=\frac{(n-2)(n-1)}{2}+\frac{\mu}{\lambda(n-2)}$$

$$P_2=\ {q_2}^{-1}(\ {q_1}^{-1}P_0+\ 2P_0)={q_2}^{-1}(\ {q_1}^{-1}+\ 2)P_0$$

$$P_2=\ {q_2}^{-1}(\ {q_1}^{-1}+\ 2)P_0$$

$i = 3,$

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

Where,

$$P_1 = {q_1}^{-1}P_0$$

$$P_2 = {q_2}^{-1}({q_1}^{-1} + 2)P_0$$

$$P_3 = {q_3}^{-1}\left({q_2}^{-1}({q_1}^{-1} + 2) + 2{q_1}^{-1} + 3\right)P_0$$

$$q_3 = \frac{(n-3)(n-2)}{2} + \frac{\mu}{\lambda(n-2)}$$

$i = 4,$

$$\lambda\left(\frac{(n-4)(n-3)}{2} + \frac{\mu}{\lambda(n-3)}\right)P_4 = \lambda P_3 + 2\lambda P_2 + 3\lambda P_1 + 4\lambda P_0$$

Where,

$$P_1 = {q_1}^{-1}P_0$$

$$P_2 = {q_2}^{-1}({q_1}^{-1} + 2)P_0$$

$$P_3 = {q_3}^{-1}\left({q_2}^{-1}({q_1}^{-1} + 2) + 2{q_1}^{-1} + 3\right)P_0$$

$$P_4 = {q_4}^{-1}\left({q_3}^{-1}\left({q_2}^{-1}({q_1}^{-1} + 2) + 2{q_1}^{-1} + 3\right) + 2({q_2}^{-1}({q_1}^{-1} + 2) + 3{q_1}^{-1} + 4\right)P_0$$

$$P_4 = q_4{}^{-1}\left(q_3{}^{-1}q_2{}^{-1}q_1^{-1} + 2q_3{}^{-1}q_2{}^{-1} + 2q_3{}^{-1}q_1{}^{-1} + 3q_3{}^{-1} + 2q_2{}^{-1}\,q_1{}^{-1} + 4q_2{}^{-1} + 3q_1{}^{-1} + 4\right)P_0$$

$$q_4 = \frac{(n-4)(n-3)}{2} + \frac{\mu}{\lambda(n-3)}$$

The generalized form is as follows.

$$\lambda\left(\frac{(n-i)(n-i+1)}{2} + \frac{\mu}{\lambda(n-i+1)}\right)P_i$$

$$= \lambda P_{i-1} + 2\lambda P_{i-2} + 3\lambda P_{i-3} + \cdots + (i-1)\lambda P_1 + i\lambda P_0$$

Where,

$0 < i \leq n,$

$$q_i = \frac{(n-i)(n-i+1)}{2} + \frac{\mu}{\lambda(n-i+1)}$$

All outgoing transitions from P_i equal all incoming transitions to P_i as follows.

$$\frac{\mu}{n-(i-1)}P_i = i\lambda P_0 \tag{55}$$

Where,

$0 < i \leq n$

Solving Equation (59) in terms of P_0 yields as follows.

$i = 1,$

$$\frac{\mu}{n} P_1 = \lambda P_0$$

$$P_1 = \frac{n\lambda}{\mu} P_0$$

$$P_1 = q_1 \lambda P_0$$

$$q_1 = \frac{n}{\mu}$$

$i = 2,$

$$\left(\frac{\mu}{n-1}\right) P_2 = 2\lambda P_0$$

$$P_2 = \left(\frac{n-1}{\mu}\right) 2\lambda P_0$$

$$P_2 = q_2 2\lambda P_0$$

$$q_2 = \frac{n-1}{\mu}$$

$i = 3,$

$$\left(\frac{\mu}{n-2}\right) P_3 = 3\lambda P_0$$

$$P_3 = \left(\frac{n-2}{\mu}\right) 3\lambda P_0$$

$$q_3 = \frac{n-2}{\mu}$$

$i = 4$,

$$\left(\frac{\mu}{n-3}\right) P_4 = 4\lambda P_0$$

$$P_4 = \left(\frac{n-3}{\mu}\right) 4\lambda P_0$$

$$q_4 = \frac{n-3}{\mu}$$

$i = n$,

$$\left(\frac{\mu}{1}\right) P_n = (n)\lambda P_0$$

$$P_n = \left(\frac{1}{\mu}\right) (n)\lambda P_0$$

$$q_n = \frac{1}{\mu}$$

To determine P_0, we use the fact that the $\sum_{i=0}^{n} P_i$ should be 1.

$$P_0 + P_1 + P_2 + \cdots + P_n = 1 \tag{56}$$

All outgoing transitions from P_i equal all incoming transitions to P_i from Equation (55) as follows.

$$\frac{\mu}{n-(i-1)}P_i = i\lambda P_0$$

P_i can be generalized and expressed as follows.

$$P_i = \left(\frac{\mu}{n-(i-1)}\right)^{-1} i\lambda P_0$$

$$P_i = q_i i\lambda P_0 \tag{57}$$

Where,

$$0 < i \leq n,\ q_i = \left(\frac{\mu}{n-(i-1)}\right)^{-1}$$

Equation (56) can be replaced by Equation (58) as follows.

$$P_0 + \sum_{i=1}^{n} P_i = 1 \tag{58}$$

Equation (57) P_i can be replaced by Equation (59) as follows.

$$P_0 + \sum_{i=1}^{n} q_i i\lambda P_0 = 1 \tag{59}$$

$$P_0\left(1+\sum_{i=1}^{n} q_i i\lambda\right) = 1$$

$$P_0 = \frac{1}{\left(1+\sum_{i=1}^{n} q_i i\lambda\right)}$$

Where,

$$\sum_{i=1}^{n} q_i i\lambda = \lambda\left(1q_1 + 2q_2 + 3q_3 + \cdots + (n-2)q_{(n-2)} + (n-1)q_{(n-1)} + nq_n\right)$$

$$= \lambda\left(1q_1 + 2q_2 + 3q_3 + \cdots + (n-2)q_{(n-2)} + (n-1)q_{(n-1)} + nq_n\right)$$

$$= \lambda\left(1\left(\frac{\mu}{n-(1-1)}\right)^{-1} + 2\left(\frac{\mu}{n-(2-1)}\right)^{-1} + 3\left(\frac{\mu}{n-(3-1)}\right)^{-1} + \cdots + (n-2)\left(\frac{\mu}{n-(n-2-1)}\right)^{-1} + (n-1)\left(\frac{\mu}{n-(n-1-1)}\right)^{-1} + n\left(\frac{\mu}{n-(n-1)}\right)^{-1}\right)$$

$$= \lambda\left(1\left(\frac{\mu}{n}\right)^{-1} + 2\left(\frac{\mu}{n-1}\right)^{-1} + 3\left(\frac{\mu}{n-2}\right)^{-1} + \cdots + (n-2)\left(\frac{\mu}{3}\right)^{-1} + (n-1)\left(\frac{\mu}{2}\right)^{-1} + n\left(\frac{\mu}{1}\right)^{-1}\right)$$

$$= \lambda\left(1\left(\frac{n}{\mu}\right) + 2\left(\frac{n-1}{\mu}\right) + 3\left(\frac{n-2}{\mu}\right) + \cdots + (n-2)\left(\frac{3}{\mu}\right) + (n-1)\left(\frac{2}{\mu}\right) + n\left(\frac{1}{\mu}\right)\right)$$

$$= \frac{\lambda}{\mu}\left(1(n) + 2(n-1) + 3(n-2) + \cdots + (n-2)(3) + (n-1)(2) + n(1)\right)$$

$$= \frac{\lambda}{\mu}\left(\sum_{r=1}^{n} T_r\right)$$

$$= \frac{\lambda}{\mu}\left((n+1)r - r^2\right)$$

Where,

$$T_r = r\big(n-(r-1)\big)$$

$$= (n+1)r - r^2$$

To find the sum of the series above equation is defined in Equation (60) as follows.

$$\big(1(n) + 2(n-1) + 3(n-2) + \cdots + (n-2)(3) + (n-1)(2) + n(1)\big) = \sum_{r=1}^{n} T_r \qquad (60)$$

$$\sum_{r=1}^{n} T_r = \sum_{r=1}^{n} \big((n+1)r - r^2\big)$$

$$= (n+1)\sum_{r=1}^{n} r - \sum_{r=1}^{n} r^2$$

$$= \frac{(n+1)n(n+1)}{2} - \frac{n(n+1)(2n+1)}{6}$$

$$= \frac{n(n+1)}{2}\left((n+1) - \left(\frac{2n+1}{3}\right)\right)$$

$$= \frac{n(n+1)(n+2)}{6}$$

$$\frac{\lambda}{\mu}\left(\sum_{r=1}^{n} T_r\right) = \frac{\lambda}{\mu}\left(\frac{n(n+1)(n+2)}{6}\right)$$

$$P_0 = \frac{1}{\left(1 + \sum_{i=1}^{n} q_i i\lambda\right)} = \frac{1}{\left(1 + \frac{\lambda}{\mu}\left(\sum_{r=1}^{n} T_r\right)\right)} = \frac{1}{\left(1 + \frac{\lambda}{\mu}\left(\frac{n(n+1)(n+2)}{6}\right)\right)}$$

$$P_n = \left(\frac{1}{\mu}\right)(n)\lambda P_0$$

$$= \left(\frac{1}{\mu}\right)(n)\lambda \frac{1}{\left(1 + \frac{\lambda}{\mu}\left(\frac{n(n+1)(n+2)}{6}\right)\right)}$$

The followings are a few fully asynchronous model performance measurements of primary interests in the fully asynchronous model.

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).

$$L_Q = \sum_{i=0}^{n} iP_i$$

Where,

$$\sum_{i=0}^{n} iP_i = \sum_{i=0}^{n} i\left(\left(\frac{\mu}{n-(i-1)}\right)^{-1} i\lambda P_0\right)$$

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

$$W_Q = \frac{L_Q}{\lambda}$$

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

$$W = W_Q + \frac{1}{\mu}$$

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).

$$L = \lambda W$$

γ : the throughput per block in the fully asynchronous model can be obtained as follows.

$$\gamma = \sum_{i=0}^{n} \mu P_i = \sum_{i=0}^{n} \mu \frac{\lambda}{\mu} \frac{i(i+1)}{2} P_0 = \sum_{i=0}^{n} \lambda \frac{i(i+1)}{2} P_0$$

2. Staged Asynchronous Chain Model Equations

The state for the staged asynchronous chain model is that multiple arrived transactions are staged to be posted to a current block. The definitions of P_0, P_n, P_i, and λ are defined in the previous Section 1 Fully Asynchronous Chain Model Equations. In a staged asynchronous model, each stage is not related to the state in other stages. Each state in a stage has only λ limited in its stage.

μ: the rate for the slots of the transactions in the queue to be posted and purged. Notice that this is a multiple and unique state transition precisely from P_{is}, $1 \leq i \leq t$, back to P_0.

Figure 17 shows the state transition diagram of the staged asynchronous chain. The balance equations for the staged asynchronous chain are defined as follows.

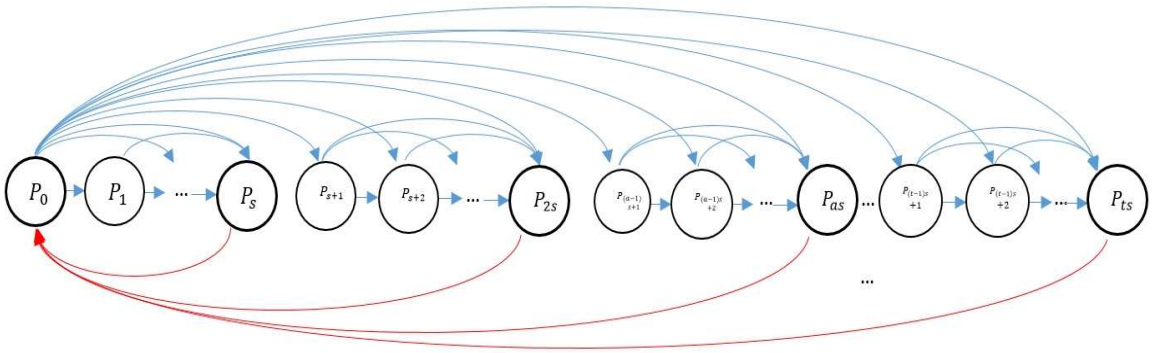

Figure 17. State transition diagram of the staged asynchronous chain model

Outgoing from P_0 is follows.

$$(\lambda + 2\lambda + 3\lambda + \cdots + n\lambda)P_0 = \lambda\frac{n(\mathrm{n}+1)}{2}P_0$$

Outgoing from P_0 equals incoming to P_0 as follows.

$$\lambda\frac{n(n+1)}{2}P_0 = \frac{\mu}{n-(s-1)}P_{1s} + \frac{\mu}{n-(2s-1)}P_{2s} + \frac{\mu}{n-(3s-1)}P_{3s} + \cdots + \frac{\mu}{n-\left((n-1)s-1\right)}P_{(n-1)s} + \frac{\mu}{n-(n-1)}P_n$$

$$= \mu\left(\frac{1}{n-(s-1)}P_{1s} + \frac{1}{n-(2s-1)}P_{2s} + \frac{1}{n-(3s-1)}P_{3s} + \cdots + \frac{1}{n-\left((n-1)s-1\right)}P_{(n-1)s} + \frac{1}{n-(n-1)}P_n\right)$$

Where,

$\lambda\frac{n(\mathrm{n}+1)}{2}P_0$: all outgoing from P_0

$\frac{\mu}{n-(s-1)}P_{1s}$: incoming from P_{1s} to P_0 with $\frac{\mu}{n-(s-1)}$

$\frac{\mu}{n-(2s-1)}P_{2s}$: incoming from P_{2s} to P_0 with $\frac{\mu}{n-(2s-1)}$

$\frac{\mu}{n-(3s-1)}P_{3s}$: incoming from P_{3s} to P_0 $with$ $\frac{\mu}{n-(3s-1)}$

$\frac{\mu}{n-(ts-1)}P_{ts}$: incoming from P_{ts} to P_0 with $\frac{\mu}{n-(ts-1)}$

$\frac{\mu}{n-(n-1)}P_n$ (μP_n): incoming from P_n to P_0 with $\frac{\mu}{n-(n-1)}$

$$\lambda \frac{n(n+1)}{2} P_0 = \mu \left(\frac{1}{n-(s-1)} P_{1s} + \frac{1}{n-(2s-1)} P_{2s} + \frac{1}{n-(3s-1)} P_{3s} + \cdots + \frac{1}{n-\left((n-1)s-1\right)} P_{(n-1)s} + \frac{1}{n-(n-1)} P_n \right)$$

$$P_0 = \left(\lambda \frac{n(n+1)}{2} \right)^{-1} \mu \left(\frac{1}{n-(s-1)} P_{1s} + \frac{1}{n-(2s-1)} P_{2s} + \frac{1}{n-(3s-1)} P_{3s} + \cdots + \frac{1}{n-\left((n-1)s-1\right)} P_{(n-1)s} + \frac{1}{n-(n-1)} P_n \right)$$

Stage 1 for P_{0s+1}:

$$\lambda(1 + 2 + 3 + \cdots + (s-1)) P_1 = \lambda P_0$$

$$\left(\frac{(s-1)s}{2} \right) P_1 = P_0$$

$$P_1 = \left(\frac{(s-1)s}{2} \right)^{-1} P_0$$

$$P_1 = q_1 P_0$$

$$q_1 = \left(\frac{(s-1)s}{2} \right)^{-1}$$

Stage 1 for P_{0s+2}:

$$\lambda(1 + 2 + 3 + \cdots + (s-2)) P_2 = \lambda P_1 + 2\lambda P_0$$

$$P_2 = P_1 + 2P_0$$

$$P_2 = \left(\frac{(s-2)(s-1)}{2} \right)^{-1} (P_1 + 2P_0)$$

$$P_2 = \left(\frac{(s-2)(s-1)}{2}\right)^{-1}\left(\left(\frac{(s-1)s}{2}\right)^{-1}P_0 + 2P_0\right)$$

$$P_2 = q_2(q_1P_0 + 2P_0)$$

Stage 1 for P_{0s+3}:

$$\lambda(1+2+3+\cdots+(s-3))P_3 = \lambda P_2 + 2\lambda P_1 + 3\lambda P_0$$

$$\left(\frac{(s-3)(s-2)}{2}\right)P_3 = P_2 + 2P_1 + 3P_0$$

$$P_3 = \left(\frac{(s-3)(s-2)}{2}\right)^{-1}(P_2 + 2P_1 + 3P_0)$$

$$P_3 = \left(\tfrac{(s-3)(s-2)}{2}\right)^{-1}\left(\left(\tfrac{(s-2)(s-1)}{2}\right)^{-1}\left(\left(\tfrac{(s-1)s}{2}\right)^{-1}P_0 + 2P_0\right) + 2\left(\tfrac{(s-1)s}{2}\right)^{-1}P_0 + 3P_0\right)$$

$$P_3 = q_3(q_2(q_1+2) + 2q_1 + 3)P_0$$

Stage 1 for P_{s-1}:

$$\lambda P_{s-1} = \lambda P_{s-2} + 2\lambda P_{s-3} + \cdots + (s-2)\lambda P_1 + (s-1)\lambda P_0$$

$$\lambda P_{s-1} = \lambda P_{s-2} + 2\lambda P_{s-3} + \cdots + (s-2)\lambda\left(\frac{(s-1)s}{2}\right)^{-1}P_0 + (s-1)\lambda P_0$$

$$P_i = P_{i-1} + 2P_{i-2} + \cdots + (i-1)\left(\frac{(s)(1+i)}{2}\right)^{-1}P_0 + iP_0$$

Stage 1 for P_s (posting stage 1):

$$\frac{\mu}{n-(s-1)}P_s = \lambda P_{s-1} + 2\lambda P_{s-2} + \cdots + (s-1)\lambda P_1 + s\lambda P_0$$

Stage 2 for P_{s+1}:

$$\lambda(1+2+3+\cdots+(s-1))P_{s+1} = (s+1)\lambda P_0$$

$$\left(\frac{(s-1)s}{2}\right)P_{s+1} = (s+1)P_0$$

$$P_{s+1} = \left(\frac{(s-1)s}{2}\right)^{-1}(s+1)P_0$$

Stage of 1 for P_{s+2}:

$$\lambda(1+2+3+\cdots+(s-2))P_{s+2} = (s+2)\lambda P_0 + \lambda P_{s+1}$$

$$\left(\frac{(s-2)(s-1}{2}\right)P_{s+2} = (s+2)P_0 + P_{s+1}$$

$$P_{s+2} = \left(\frac{(s-2)(s-1)}{2}\right)^{-1}\left((s+2)P_0 + P_{s+1}\right)$$

Generalized form for P_{as}:

$$\frac{\mu}{n-(as-1)}P_{as} = \lambda P_{(a-1)s-1} + 2\lambda P_{(a-1)s-2} + \cdots + (s-1)\lambda P_{(a-1)s+1} + as\lambda P_0$$

Where,

$1 \le a \le t,$

P_{as}: the a^{th} staged state and note that $t = \left\lceil \frac{n}{s} \right\rceil$ for P_{ts},

$$P_{as+i} = q_{as+i}P_0\left[\sum_{j=1}^{as+i} j\left[\sum_{k=1}^{as+i-}\left[\prod_{l=1}^{k-1} q_l\right]k\right] + as + i\right]$$

$$P_{as+i} = q_{as+i}P_0\left(\left(\frac{(as+i)((as+i)+1)}{2}\right)\left(\frac{n}{\left(\sqrt{2\pi}\left(\frac{n}{e}\right)^n\right)^2}\right)\left(2^{(as+i)-1}((as+i)n - 2n + 7(as+i) - 14 - (as+i)^2) + 8\right) + as + i\right)$$

Where,

- $0 \le a \le t, 0 \le i < s, t = \left\lceil \frac{n}{s} \right\rceil, s = \left\lceil \frac{n}{t} \right\rceil$
- $q_{as+i} = \frac{2}{(n-(as+i))(n-(as+i)+1)}$
- $P_{as} = \frac{\lambda(n-(as-1))}{\mu}\frac{n(n+1)}{2}P_0$

The performance measurements of primary interests in the asynchronous model are expressed in L_Q, W_Q, W, and L as following Equations respectively.

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).

$$L_Q = \sum_{i=0}^{n} iP_i$$

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

$$W_Q = \frac{L_Q}{\lambda}$$

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

$$W = W_Q + \frac{1}{\mu}$$

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).

$$L = \lambda W$$

γ : the throughput per block (γ) and the throughput can be expressed differently for the baseline, the fully asynchronous, the stage asynchronous chain, and the adaptive chain.

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

for the baseline chain model in Equation (28);

$$\gamma = \sum_{i=0}^{n} \mu P_i = \sum_{i=0}^{n} \frac{\mu}{n-(i-1)} \frac{n-(i-1)}{\mu} i\lambda P_0 = \sum_{i=0}^{n} i\lambda P_0 \tag{61}$$

for the fully asynchronous chain model in Equation (61);

$$\gamma = \sum_{a=0}^{t} \frac{\mu}{n-(i-1)} P_{as} \tag{62}$$

Where,

P_{as}: the a^{th} staged state and note that $t = \left\lceil \frac{n}{s} \right\rceil$ for P_{ts},

for the staged asynchronous chain model in Equation (62);

and

$$\gamma = \sum_{i=0}^{n} \frac{\mu}{n-(i-1)} P_i$$

for the adaptive chain model in Equation (54).

3. Numerical Analysis

The primary objective of the numerical simulation is to reveal the various preliminary performance of the blockchain system of interest such as W and L versus n (i.e., size of a block), λ (i.e., transaction arrival rate or speed), and $\frac{1}{\mu}$ (i.e., block posting time).

Figure 18 plots L based on Equation (27) with respect to n for given pairs of λ and μ. Observe that as λ=0.005 and at μ=0.0067, L grows in a monotonic manner with baseline, but in stage, it increases within each stage and gradually increases, repeating a slight descent at the end of the stage. In adaptive cases, it shows that it is growing very gently, and in the last fully asynchronous, it is a graph similar to $\frac{1}{n}$ was observed, with n increasing and L decreasing gradually but it looks almost identical to the x-axis.

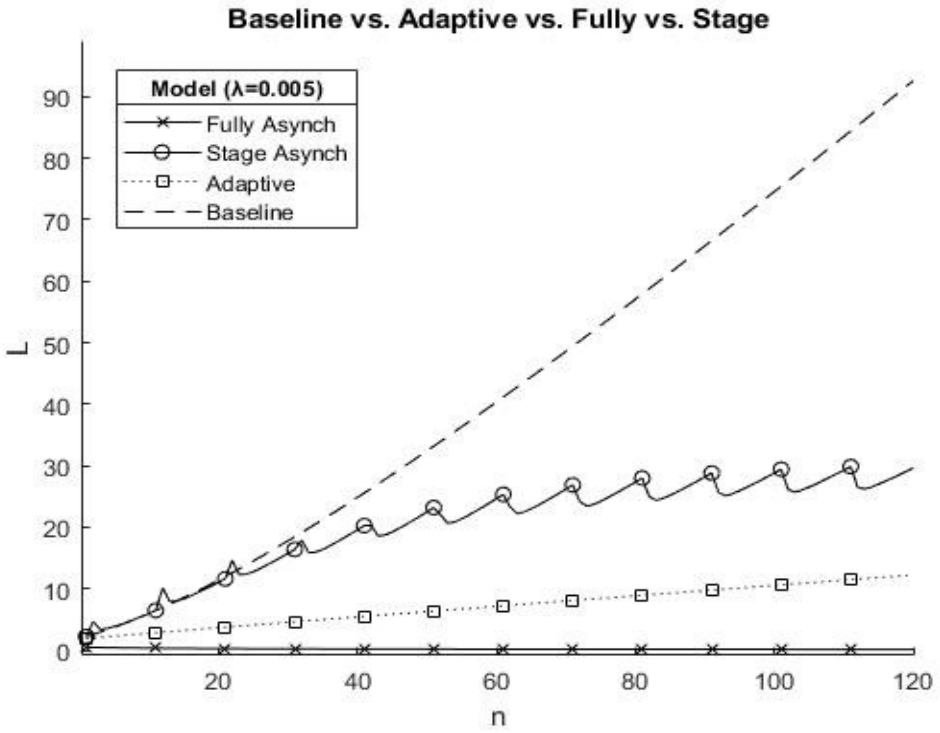

Figure 18. Baseline, adaptive, and asynchronous chain models for L, λ=0.005

Figure 19 plots L based on Equation (27) with respect to n for the given pairs of λ and μ. Observe that as λ 0.05 and at μ=0.0067, L grows in a monotonic manner with baseline, but in stage, it increases within each stage and gradually increases, repeating a slight descent at the end of the stage. In adaptive cases, it shows that it is growing very gently, and in the last full case, it is a graph similar to $\frac{1}{n}$ was observed, with n increasing and L decreasing gradually but it looks almost identical to the x-axis. Unlike the previous Figure 18, the angle of the graph of adaptive has risen slightly, and the stage shows an increase until the first n=20 and then increases little, and is lower than adaptive at around n=70.

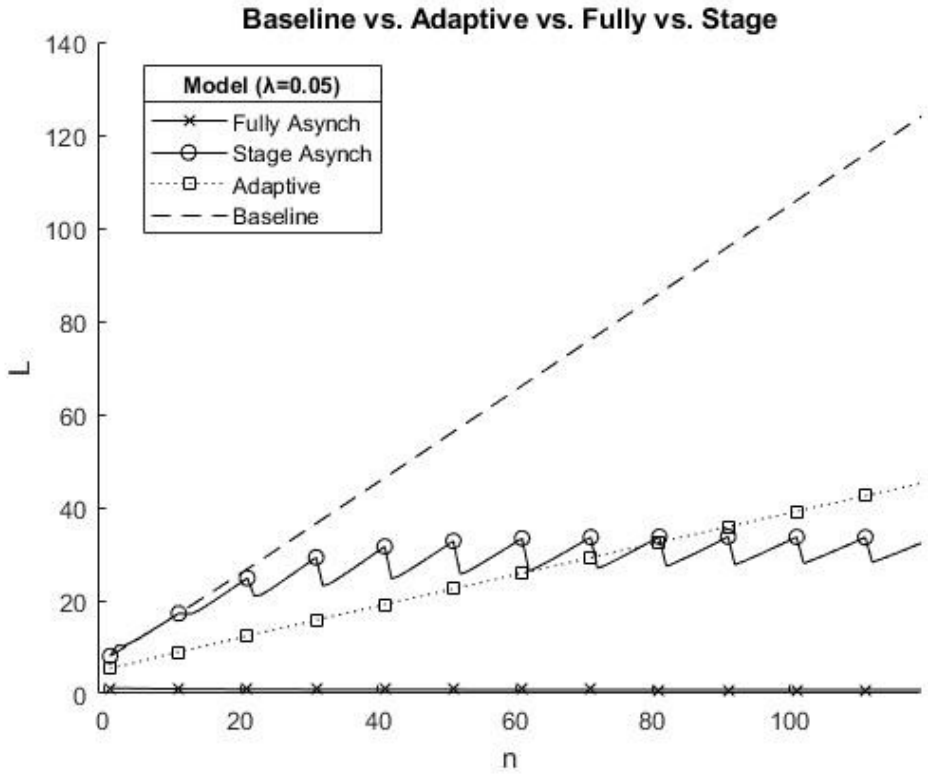

Figure 19. Baseline, adaptive, and asynchronous chain models for L, λ=0.05

Figure 20 plots W based on Equation (26) with respect to n for given pairs of λ and μ. Observe that as λ=0.005 and at μ=0.0667, W trends more or less the same patterns with Figure 18.

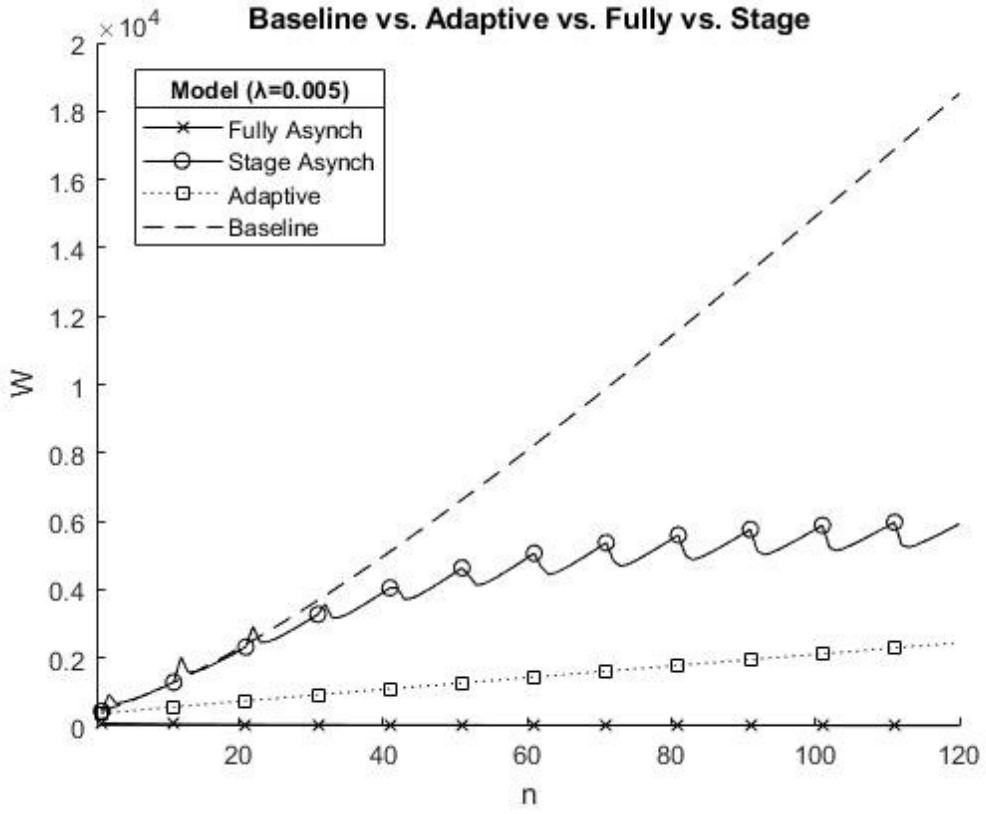

Figure 20. Baseline, adaptive, and asynchronous chain models for W, λ=0.005

Figure 21 plots W based on Equation (26) with respect to n for the given pairs of λ and μ. Observe that as λ=0.05 and at μ=0.0667, W trends more or less the same patterns with Figure 19. The difference from Figure 20 is observed to be a slight increase in the range of waiting time W at the y-axis.

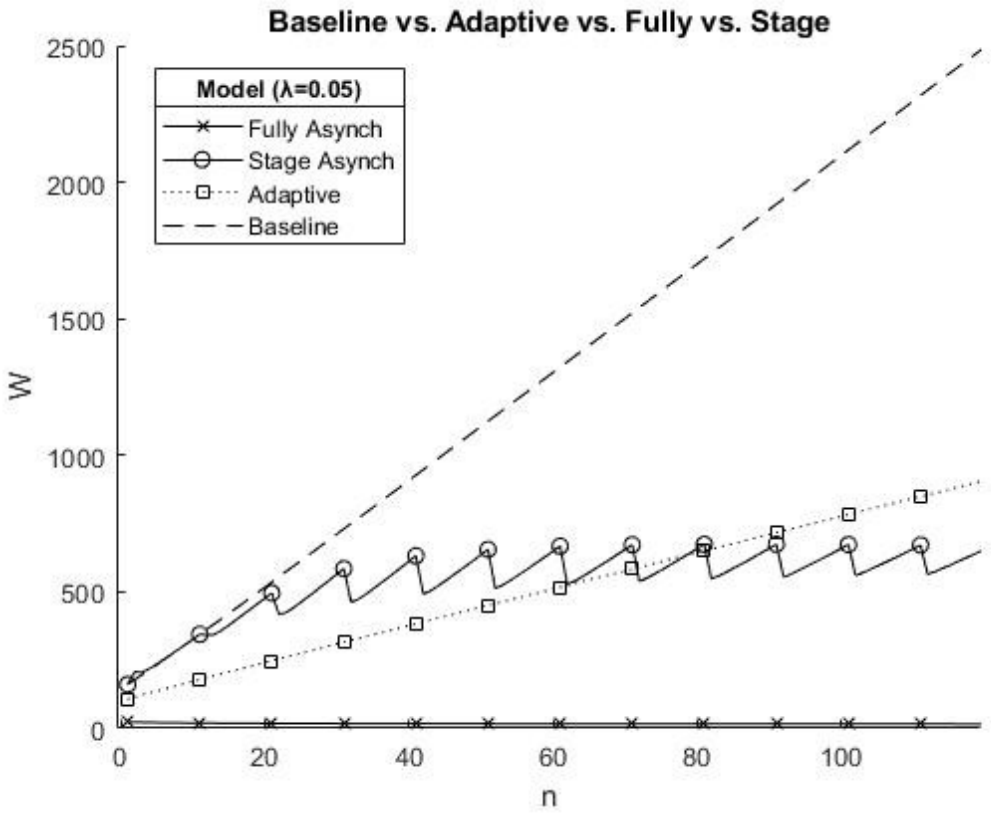

Figure 21. Baseline, adaptive, and asynchronous chain models for W, λ=0.05

Figure 22 plots the throughput per block (γ) in the four different models. The throughput per block can be obtained from the different equations such that for the baseline from Equation (28), the fully asynchronous from Equation (61), the staged asynchronous from Equation (62) and the adaptive from Equation (62). Observe that for λ=0.005 and μ=0.0667, the highest throughput is achieved by the fully asynchronous as expected, and likewise, the baseline throughput also increases along as the n increases, while the staged model increases and bounces back down at n=20 eventually. The adaptive model shows the lowest throughput consistently throughout as a justification for the proactively dynamic adaptive chain (i.e., the asynchronous chain) versus the reactively dynamic adaptive chain.

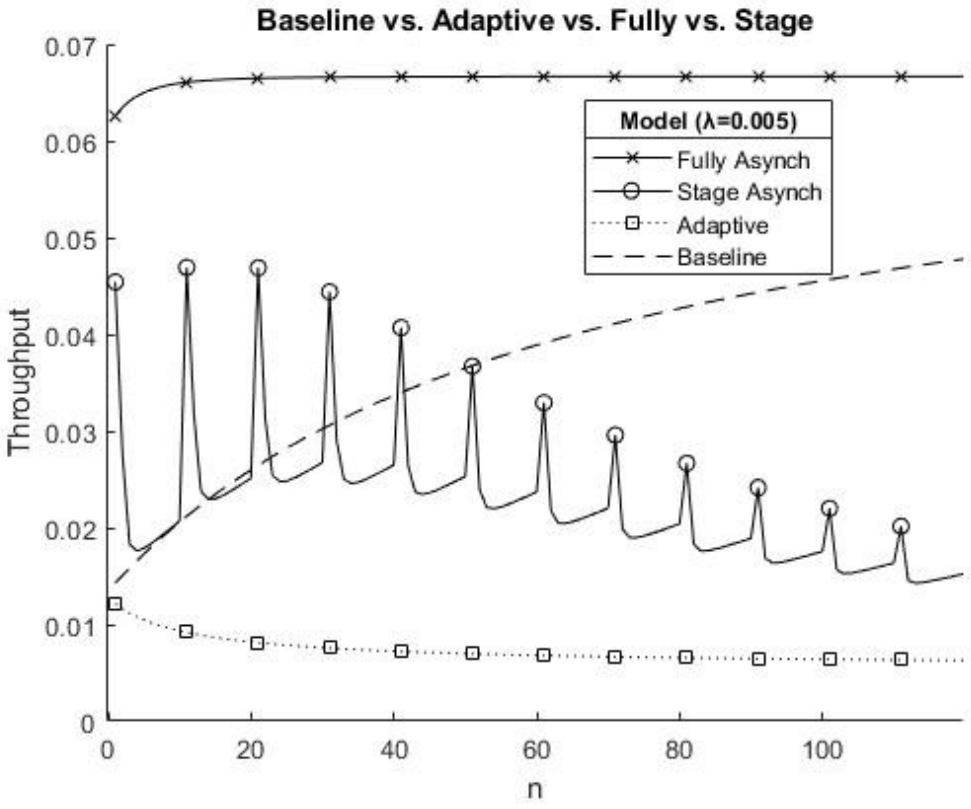

Figure 22. Baseline, adaptive, and asynchronous chain models for γ, λ=0.005

Figure 23 plots the throughput per block (γ) in the four different models likewise as shown in Figure 22 for the same μ=0.0667 yet for an elevated λ=0.05. It is observed that at an elevated transaction arrival rate, the fully asynchronous model definitely as expected, continues to exhibit the highest throughput, however the staged asynchronous model turns outperformed by the baseline model beyond *n*=20 which is cut quicker than in the case of Figure 22, and it is also noteworthy that the adaptive shows the lowest throughput until *n*=80, from there on the staged turns lower.

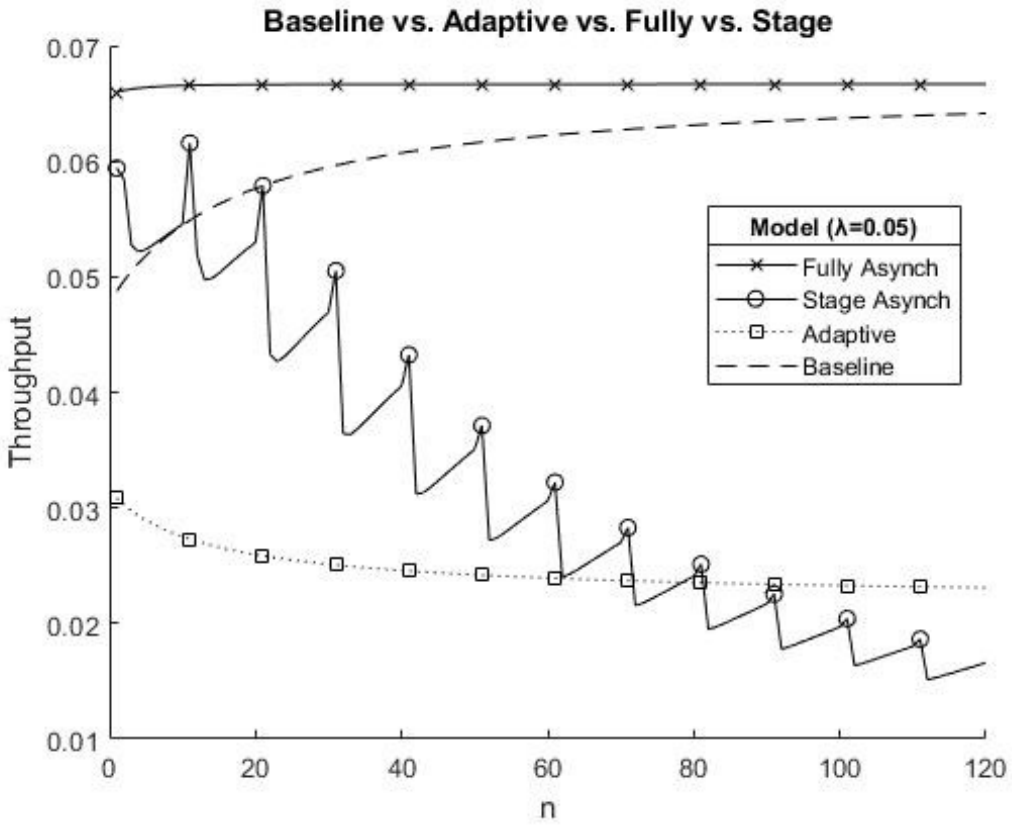

Figure 23. Baseline, adaptive, and asynchronous chain models for γ, λ=0.05

CHAPTER VI

IMPLEMENTATION AND EXPERIMENTAL RESULTS

This chapter presents algorithms of the adaptive chain model and two different asynchronous chain models such as the fully asynchronous and the staged asynchronous. The following algorithms and procedures are coded to allow comparative analysis of simulations and implementations for each model. The adaptive chain model algorithm is shown in section 1, the fully asynchronous chain model algorithm is shown in Section 2, and the staged asynchronous chain model is shown in Section 3. This chapter also describes the experimental environments such as hardware, software, network, and installation. Section 4 shows the implementation and experimental environment setup. Section 5 shows the implementation and experimental procedure. Lastly, it demonstrates the implementation and experimental results and compares four different chain models in Section 6.

1. Adaptive Chain Algorithm

    1) Set $block\_gas\_limit$, and reset $total\_gas = 0$
    2) Mine a transaction and read gas_fee_used and $total\_gas += gas\_fee\_used$
    3) If $total\_gas \leq block\_gas\_limit$ then go to 2)
    4) Post the current block and go to 1)

Adaptive Chain Procedure {

Set block_gas_limit

Initialize total_gas := 0

Loop for Block {

Wait for a transaction

If no transaction arrived in Block time {

Post the empty_current_block

}

A transaction arrived == true {

Mining the new_transaction

Read gas_fee_used

Check the transaction gas_fee_used

```
                    Count transactions = transactions + new_transaction

                    Updated current_block = transactions

                    Sum total_gas = total gas + gas_fee_used

                    If  total_gas > block_gas_limit {

                            Post the current_block

                    }

                    Else {

                            Loop for Block

                    }

            } //End A transaction arrived

        } //End Loop for Block

} //End Adaptive Chain Procedure
```

2. Fully Asynchronous Chain Algorithm

1) If $gas\ used \leq limit$, post the current block
2) Else, set $limit = gas\ used$ and go to 1)

Fully Asynchronous Chain Procedure {

Set block_gas_limit

Loop for Block {

Wait for a transaction

If no transaction arrived in Block time {

Post the empty_current_block

}

A transaction arrived == true {

Mining the new_transaction

Read gas_fee_used

If gas_fee_used ≤ block_gas_limit {

Post the current block

}

Else {

Block_gas_limit = gas_fee_used

}

} // End Loop for Block

} // End Fully Asynchronous Chain Procedure

3. Staged Asynchronous Chain Algorithm

1) Set staged gas range for each stage s_i
2) Mine a transaction, read gas_fee_used, determine the stage s_i, and total $t_{s_i} += g$
3) If $t_{s_i} \leq limit_{s_i}$, then go to 2
4) Post the current block with the transactions of t_{s_i}, and go to 2)

Staged Asynchronous Chain Procedure {

Set staged_block_gas_limit

Loop for Block {

Wait for a transaction

If no transaction arrived in Block time {

Post the empty_current_block

}

A transaction arrived == true {

Mining the new_transaction

Read gas_fee_used

If staged_gas_fee_used ≤ staged_block_gas_limit {

Post the current staged block

}

```
                    Else {

                        staged_block += new transaction

                    }

                } // End Loop for Block

} // End Staged Asynchronous Chain Procedure
```

4. Implementation and Experimental Environment Setup

This section introduces the hardware requirement, software requirement, operating systems, IDE (Integrated Development Environment), and network for the implementation and experimental environment setup.

1) Hardware specification
    - Apple iMac
    - Intel Core i5 (2.3GHz)
    - 8GB Memory
    - 1TB Hard Disk

2) Operating systems
    - macOS Catalina version 10.15

3) Software
    - Ethereum Protocol with Golang (https://github.com/ethereum/go-ethereum)
    - Go Language (version 1.13.4, https://golang.org/) for Geth (Go Ethereum)
    - Solidity Language for the smart contract (transaction)

4) IDE (Integrated Development Environment)
    - Remix (https://remix.ethereum.org/ [42]) for Smart Contract using Web3 [43] provider

Web3 is a JavaScript library (API) to interact with Ethereum node remotely or locally and works to connect to an Ethereum node remotely or locally using JSON RPC with Remix (https://remix.ethereum.org)

- Visual Studio Code for Go Ethereum (https://code.visualstudio.com/docs/languages/go)
- GoLand for Go Ethereum (https://www.jetbrains.com/go/promo/?gclid=Cj0KCQjwwLKFBhDPARIsAPzPi-KGXVlE_gNvXybyjdoAbp0CtljkJOR4yL31HDnzkInKr9jEQl4h39EaAlAKEALw_wcB)

5) Network

- A local private Ethereum network using the Proof of Authority consensus engine (Clique)

5. Implementation and Experimental Procedure

This section shows the procedure of each step of commands to make a geth execution file, block information, transaction information, and a mining start/stop. it. The remix shows the transaction from the solidity program language

1) Remix Configuration [41, 42]

- Deploy & Run: Compiled smart contract (transaction) to be posting in a block
  - to send contract compiled transactions to the environment
- Environment: Web3 Provider
  - to connect to a remote node by providing the URL with geth
- Web3 Provider: Remix and Geth
  - to run Remix and a local test node
- Account: “0xa4040DA9353CD9ba03C7e9BCe0876d450fea08B5”
  - to use my accounts list for transaction
- Gas Limit: set 3000000 as default
  - to set the max Gas Limit of the smart contract in Remix
- Value: set 0 as default
  - to send wei, gwei, finney, ether if the smart contract has the payable function

2) Compile and make geth (go ethereum source code)

- Compiling all geth source code files and make the latest geth executable file version
- Command: “make geth” in the source directory in Figure 24

Figure 24. Build Go Ethereum source code

3) Run the geth

- Run the geth with rpcport, rpccorsdomain, unlock my account, and target gaslimit in bin folder
- Command: ./geth --datadir “data file name ” --port 30304 --networkid 1234 --rpc --rpcport “8545” --rpccorsdomain http://remix.ethereu.org --“allow-insecure-unlock” “my account” --password password.txt --targetgaslimit ‘10000000’ in Figure 25

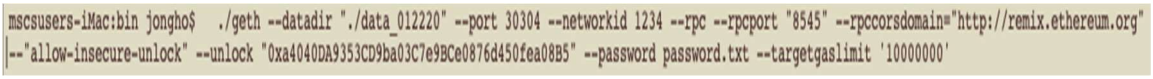

Figure 25. Run command with geth

4) Mining

- CPU Mining with Geth in Geth JavaScript Console
- Start CPU mining and stop CPU mining
- Command: miner.start(number of threads), miner.stop() in Figure 26

```
mscsusers-iMac:data_012220 jongho$ geth attach /Users/jongho/go-ethereum/build/bin/data_012220/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.9.9-unstable-c2d65d34-20191202/darwin-amd64/go1.13.4
coinbase: 0xa4040da9353cd9ba03c7e9bce0876d450fea08b5
at block: 515878 (Sun, 02 May 2021 10:51:59 CDT)
 datadir: /Users/jongho/go-ethereum/build/bin/data_012220
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> miner.start(4)
null
>
```

Figure 26. CPU mining in JavaScript console attach

5) Geth JavaScript Console [43]

- Web3.js – Ethereum JavaScript API provided to interact with a local or remote ethereum node by HTTP, IPC or WebSocket
- Command: geth console, geth attach in Figure 27

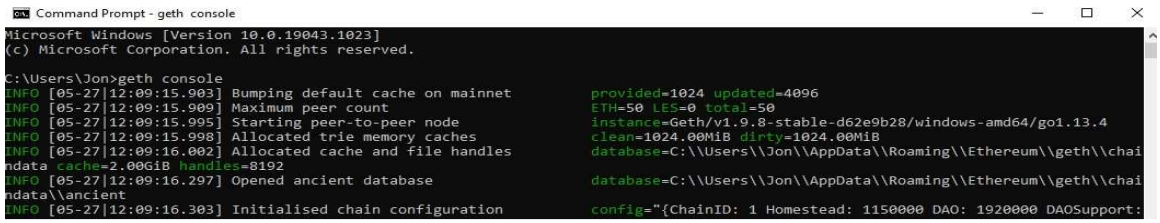

Figure 27. Run geth console in geth JavaScript console

6) Block

- Find the block information with block number
- Command: eth.getBlock('block number') in JavaScript console or attach in Figure 28

```
> eth.getBlock('515894')
{
  difficulty: 2914853,
  extraData: "0xd983010909846765746888676f312e31332e348664617277696e",
  gasLimit: 17000000,
  gasUsed: 0,
  hash: "0x866c14a3f66d33ccfda6408abcb2beb8dd0a99259f3a89d54fc098ea1dfbed9b",
  logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",
  miner: "0xa4040da9353cd9ba03c7e9bce0876d450fea08b5",
  mixHash: "0x3ca48f7a7c1fc50ad112ca9fc362e83a346e523fe822a2d18e711065bf9baf1c",
  nonce: "0x50763f760ac49664",
  number: 515894,
  parentHash: "0x7c26dc486629a8548477ee760ab31080badb4c5be312c1b0b4fe4c3366e93989",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 541,
  stateRoot: "0x77b212b68c2b2ba1ee4f5bb15cd5a6ae7091c0e5c7bbd1d595d17e5d4ff074d1",
  timestamp: 1622046383,
  totalDifficulty: 1184464707567,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

Figure 28. Find the empty block information with geth command in console or attach

7) Information of Block creation with transactions

- There is all information of the transaction, block, mining, time, hash code, elapsed, and so on in Figure 29

```
 gasCeil  :  500000
 contrib  :  0
 decay    :  16600
 params.MinGasLimit :  5000
limit:  17000000
2021/05/26 11:26:06 NewBlock:  2021-05-26 11:26:06.883763 -0500 CDT m=+441.310571978
# New block created:  true  transactions:  [0xc0004750e0]
2021/05/26 11:26:06 NewBlock elapsed :  48.772µs
INFO [05-26|11:26:06.883] Commit new mining work for              number=515893 sealhash=1c05a8…b81b31 uncles=0 txs=1 gas=135147 fees=0.000135147 elapsed=468.
571µs
2021/05/26 11:26:06 NewBlock:  2021-05-26 11:26:06.883891 -0500 CDT m=+441.310699966
# New block created:  true  transactions:  [0xc0004750e0]
2021/05/26 11:26:06 NewBlock elapsed :  31.153µs
INFO [05-26|11:26:23.540] Successfully sealed new block (confirmed new block) number=515893 sealhash=1c05a8…b81b31 hash=7c26dc…e93989 elapsed=16.657s
INFO [05-26|11:26:23.540] 🔗 block reached canonical chain          number=515886 hash=64f1d6…7e08dc
INFO [05-26|11:26:23.540] 🔨 mined potential block (a new mined block unconfirmed, but created a block)  number=515893 hash=7c26dc…e93989
INFO [05-26|11:26:23.540] ** CalcGasLimit () **
 parent.GasLimit():  17000000
 limit    :  16983597
 gasFloor :  10000000
 gasCeil  :  8000000
 contrib  :  197
 decay    :  16600
 params.MinGasLimit :  5000
 parent.GasLimit():  17000000
 limit    :  16983597
 gasFloor :  17000000
 gasCeil  :  500000
 contrib  :  197
 decay    :  16600
 params.MinGasLimit :  5000
limit:  17000000
2021/05/26 11:26:23 NewBlock:  2021-05-26 11:26:23.540505 -0500 CDT m=+457.968232825
# New block created:  true  transactions:  []
2021/05/26 11:26:23 NewBlock elapsed :  26.782µs
INFO [05-26|11:26:23.540] Commit new mining work for              number=515894 sealhash=911aa1…b2ae00 uncles=0 txs=0 gas=0      fees=0          elapsed=360.
199µs
2021/05/26 11:26:23 NewBlock:  2021-05-26 11:26:23.540602 -0500 CDT m=+457.968329579
# New block created:  true  transactions:  []
2021/05/26 11:26:23 NewBlock elapsed :  7.838µs
INFO [05-26|11:26:25.125] Successfully sealed new block (confirmed new block) number=515894 sealhash=911aa1…b2ae00 hash=866c14…fbed9b elapsed=1.584s
INFO [05-26|11:26:25.125] ** CalcGasLimit () **
 parent.GasLimit():  17000000
 limit    :  16983400
 gasFloor :  10000000
 gasCeil  :  8000000
 contrib  :  0
 decay    :  16600
 params.MinGasLimit :  5000
 parent.GasLimit():  17000000
 limit    :  16983400
 gasFloor :  17000000
 gasCeil  :  500000
 contrib  :  0
 decay    :  16600
 params.MinGasLimit :  5000
limit:  17000000
INFO [05-26|11:26:25.125] 🔗 block reached canonical chain          number=515887 hash=8b87ad…db8fcc
2021/05/26 11:26:25 NewBlock:  2021-05-26 11:26:25.125365 -0500 CDT m=+459.553179806
INFO [05-26|11:26:25.125] 🔨 mined potential block (a new mined block unconfirmed, but created a block)  number=515894 hash=866c14…fbed9b
# New block created:  true  transactions:  []
2021/05/26 11:26:25 NewBlock elapsed :  15.609µs
INFO [05-26|11:26:25.125] Commit new mining work for              number=515895 sealhash=d5f621…3702b9 uncles=0 txs=0 gas=0      fees=0          elapsed=309.
901µs
2021/05/26 11:26:25 NewBlock:  2021-05-26 11:26:25.125469 -0500 CDT m=+459.553284211
# New block created:  true  transactions:  []
2021/05/26 11:26:25 NewBlock elapsed :  8.002µs
```

Figure 29. Go Ethereum JavaScript Main Console

8) Detailed block information in JavaScript console

- eth.getBlock(‘block number’) shows the detailed information as follows
- Find a transaction in a block and detailed information in a block in Figure 30

```
> eth.getBlock('515893')
{
  difficulty: 2916276,
  extraData: "0xd983010909846765746888676f312e31332e348664617277696e",
  gasLimit: 17000000,
  gasUsed: 135147,
  hash: "0x7c26dc486629a8548477ee760ab31080badb4c5be312c1b0b4fe4c3366e93989",
  logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000",
  miner: "0xa4040da9353cd9ba03c7e9bce0876d450fea08b5",
  mixHash: "0xc8757a5ac8db37c2861b3c16a99193726ac05d89a1502e76cdb73db19bb922b6",
  nonce: "0x58448b9edc453bc6",
  number: 515893,
  parentHash: "0x2d6ebc6f9177279e838bad9b606a04dfac2857c279613eccf56532e4607527d4",
  receiptsRoot: "0x44bba3dffe229ee534d24805c6a0d18eda191961124e47901ae84bf112fc084e",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 968,
  stateRoot: "0xb9439ee02d17dd0cc9194e44afae95668e9ffac399bbc0488eb30e2c826aa720",
  timestamp: 1622046360,
  totalDifficulty: 1184461792714,
  transactions: ["0x3d1b849a4de91844e7a032e102a433db36564405c3f95ce93da561b82988bbf6"],
  transactionsRoot: "0x1499b279db805518366815ecf7396ee6f2291f2508d7f63b36eb4bcac1fd3d05",
  uncles: []
}
> eth.getBlock('515894')
{
  difficulty: 2914853,
  extraData: "0xd983010909846765746888676f312e31332e348664617277696e",
  gasLimit: 17000000,
  gasUsed: 0,
  hash: "0x866c14a3f66d33ccfda6408abcb2beb8dd0a99259f3a89d54fc098ea1dfbed9b",
  logsBloom: "0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000",
  miner: "0xa4040da9353cd9ba03c7e9bce0876d450fea08b5",
  mixHash: "0x3ca48f7a7c1fc50ad112ca9fc362e83a346e523fe822a2d18e711065bf9baf1c",
  nonce: "0x50763f760ac49664",
  number: 515894,
  parentHash: "0x7c26dc486629a8548477ee760ab31080badb4c5be312c1b0b4fe4c3366e93989",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 541,
  stateRoot: "0x77b212b68c2b2ba1ee4f5bb15cd5a6ae7091c0e5c7bbd1d595d17e5d4ff074d1",
  timestamp: 1622046383,
  totalDifficulty: 1184464707567,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
```

Figure 30. Transaction information in a block and empty block in JavaScript console

9) A solidity source code for a smart contract to request a transaction from Remix

- Remix provides the solidity for a smart contract to deploy and run transactions in three kinds of environment such as JavaScript VM, Injected Web3, and Web3 Provider in Figure 31

- A smart contract is coded by solidity source code to send a transaction to the Geth using Web3 Provider
- The Web3 Provider is used to connect Geth with Remix IDE

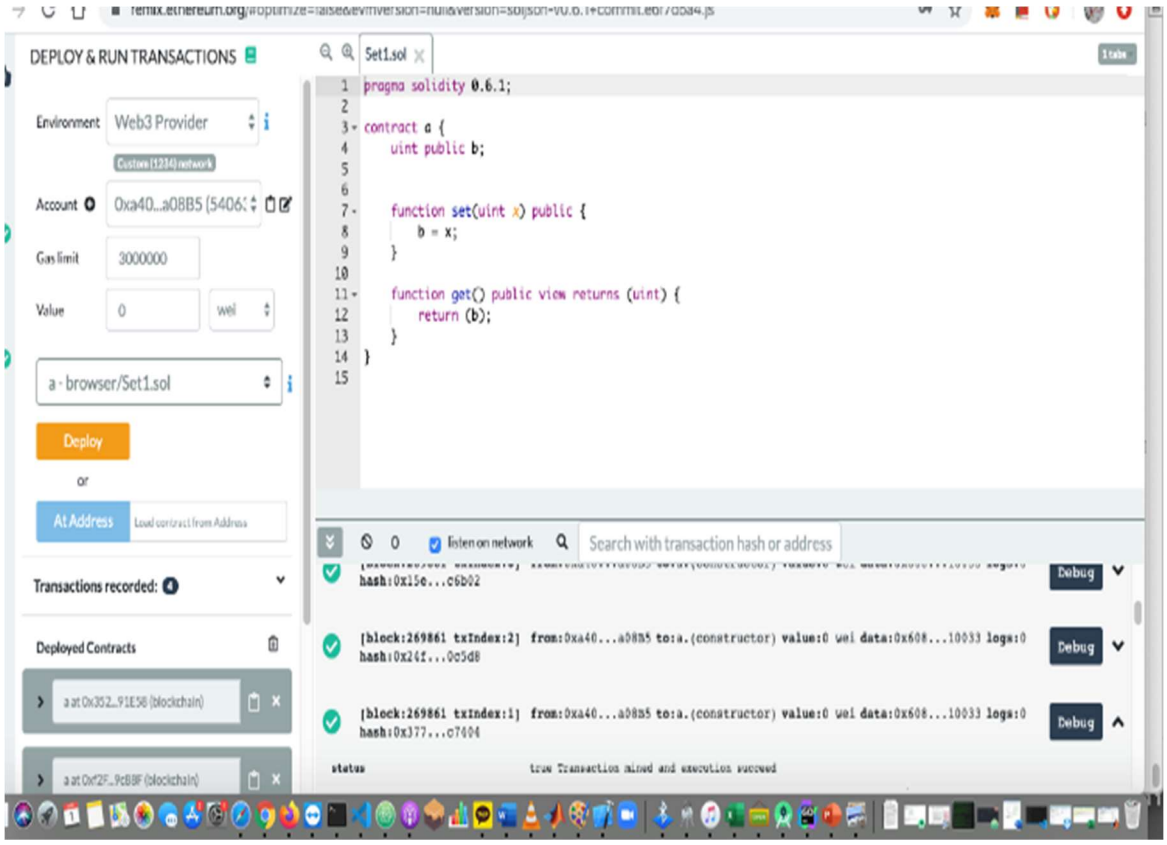

Figure 31. A simple smart contract of the solidity source code in Remix IDE [41]

Figure 32 plots the experimental and implemented average waiting time of transactions given an arrival rate (e.g., average 0.2 sec) and given a posting rate. The average waiting time is obtained by averaging the arrival time of each transaction minus its posting time, the average of the simulated waiting time versus the number of transactions arrived. The adaptive chain shows the lowest waiting time throughout, followed by the fully asynchronous and the staged asynchronous chains as the next slowest, and it is observed that the time spent waiting for posting with a large number of transactions in the naïve baseline chain is quite linearly proportional to the number of transactions as expected, and which is the primary motivation for the proposed dynamic blockchain either reactive or proactive.

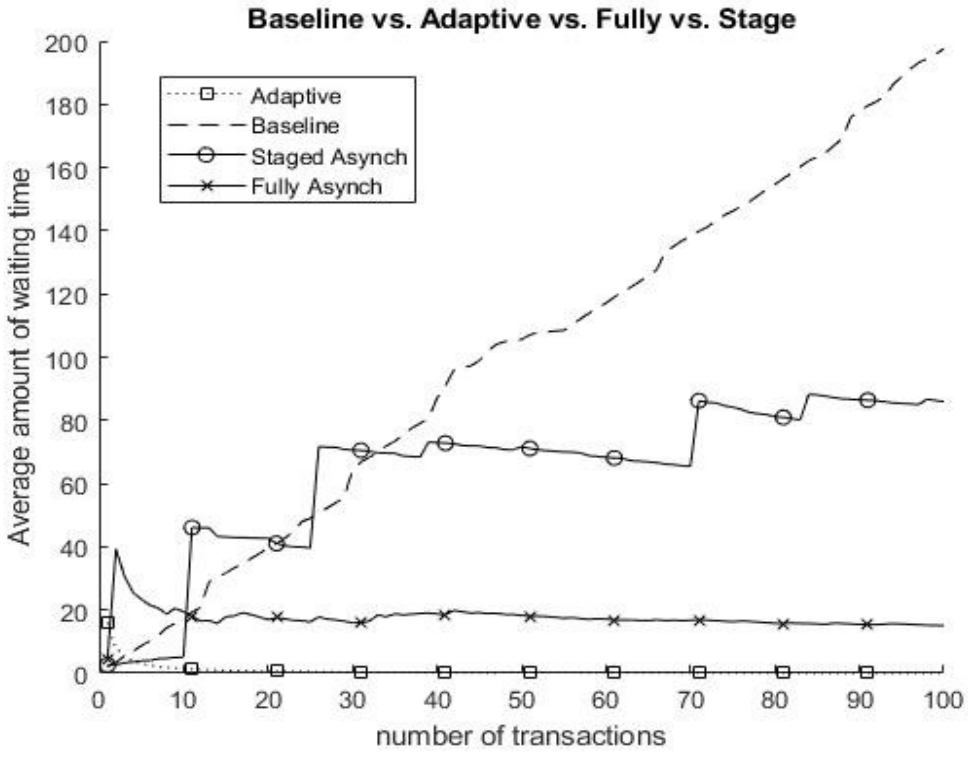

Figure 32. Baseline, adaptive, and asynchronous chain models for the waiting time

Figure 33 plots the throughput per block with respect to the number of transactions given for a given arrival rate (e.g., average 0.2 sec) and given a posting rate. The fully asynchronous chain initially and briefly maintains the highest throughput, then drops down slightly below the baseline chain beyond $n = 20$ and maintains it. Overall, the baseline and the adaptive and the staged asynchronous chains demonstrate a good agreement with the numerical simulation results. The throughput is defined by the number of transactions arrived over the block delay (i.e., block posting time minus block creation time), and calculated as follows.

$$Throughput = \frac{total\ number\ of\ transactions\ up\ to\ the\ point}{total\ time\ span\ up\ to\ the\ point}$$

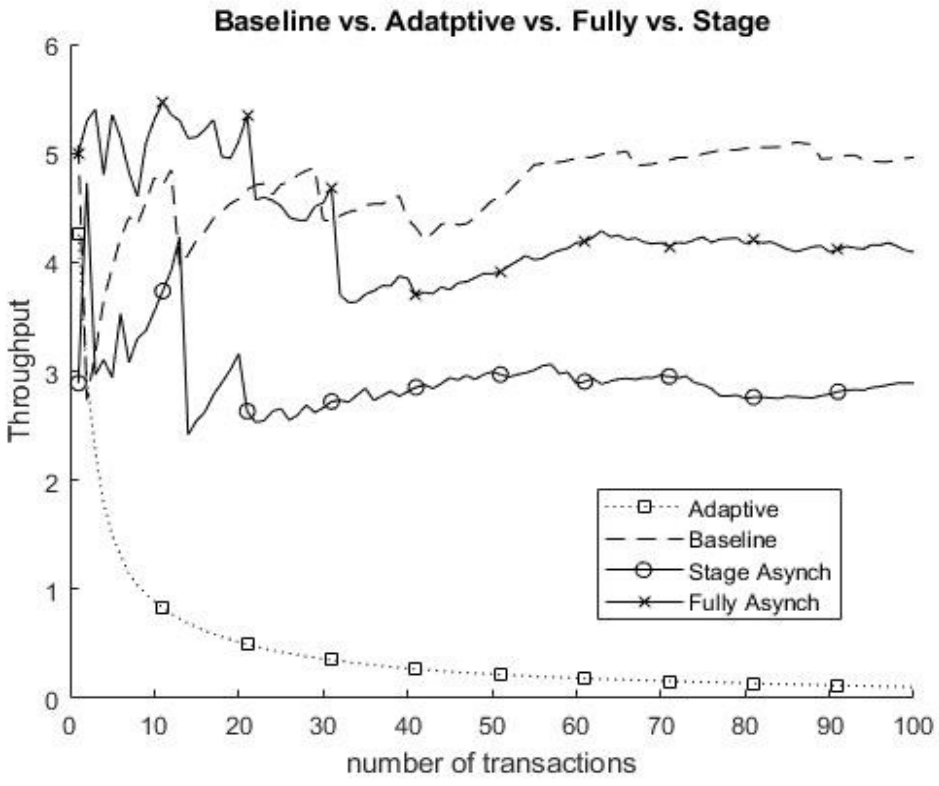

Figure 33. Baseline, adaptive, and asynchronous chain models for the throughput

Figure 34 plots the average # of slots per block with respect to the number of transactions given for a given arrival rate (e.g., average 0.2 sec) and given a posting rate. In the experimental results as shown in Fig.9, the average number of slots is obtained by averaging the number of 100,000 wei versus the number of transactions arrived, whereas, in the theoretical (numerical) results as shown in Fig. 1-6, the average number of slots (equivalently, the average number of 100,000 wei) versus the number of transactions arrived is simulated. The baseline, the adaptive, the fully asynchronous and the staged asynchronous chains show in good agreement with the numerical results.

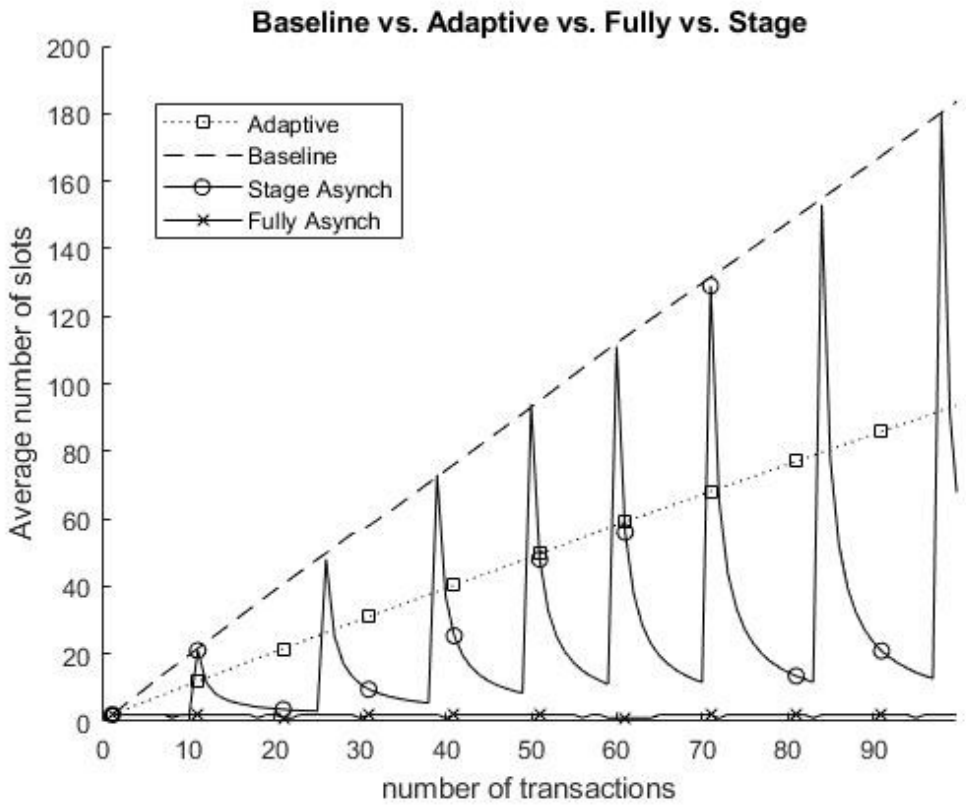

Figure 34. Baseline, adaptive, and asynchronous chain models for the number of slots

CHAPTER VII

CONCLUSION AND DISCUSSIONS

This dissertation has presented a variety of models of blockchain architecture to be proposed and studied for a variety of dynamic block size adjustment solutions. The baseline model has demonstrated the baseline chain as the conventional solution without allowing any block size adjustment. The adaptive model has demonstrated an adaptive chain as a naïve solution to block size adjustment in a reactive manner. The asynchronous model has demonstrated two different types of chains such as the fully asynchronous chain as another naïve solution yet in a proactive manner, and the staged asynchronous chain as a solution that is a hybrid form of the reactive and the proactive solutions.

The models have been expressed by developing four different types of embedded Markovian queueing models of the $M^{1,n}/M^{n}/1$ type (VBASBS) for the baseline chain, the $M^{1,n}/M^{1,i,n}/1$ type (VBAVBS) for the adaptive chain, and the $M^{1,n}/M^{ik,n}/1$ type (VBAABS) for the asynchronous chain precisely for the fully asynchronous as well as the staged asynchronous chain, in order to establish a theoretical foundation to design a blockchain-based system with a

focus on the stochastic behavior of the mined transactions waiting to be posted for the block delay along with the assumptions of the static bulk service, the variable bulk service, and the fully and staged asynchronous bulk services, respectively. As shown in the results of the numerical analysis and the experimental results, the theoretical models have been extensively simulated to compare the basic performance of the proposed models in the perspective of the average transaction waiting time, the average number of slots per block, and throughput.

The baseline model has presented the assumption of the static bulk service taken place when the number of slots in the mined transactions reaches n, i.e., a bulk processing of multiple transactions in multiple slots for posting in a block in numerical simulation, on the other hand, the adaptive model has presented every state $P_i, 0 < i \leq n$, potentially transitions back into P_0, which represents the normalized size of the block at various and random sizes of the block in numerical simulation. The adaptive model has shown the performance advantages to the baseline model through the average number of transactions, the average number of waiting time, but except throughput. The staged asynchronous model has presented staged state transitions precisely from, $P_{is}, 1 \leq i \leq t$, back into P_0, which represents the asynchronously scalable staged size of the block at various and random sizes of the block in numerical simulation. The staged asynchronous model has shown the performance advantages against both the baseline and the adaptive models from the number of transaction around $n > 60$ through the average number of transactions, the average number of waiting time, but except throughput. The fully asynchronous model has presented a single and unique state transition precisely from $P_i, 1 \leq i \leq n$, back into P_0, which represents an arrived transaction only to be posted to a current block. The fully asynchronous model has shown the performance excellence against the other models through the average number of transaction, the average number of waiting time and throughput in numerical simulation.

A reasonable agreement has been observed with the theoretical results overall as presented in the section of implementations and experimental results. The proposed staged asynchronous chain model has demonstrated as expected as the most alternative yet promising solution to the baseline chain model in order to improve the speed that is desired by such transactions as being exigently mandated to be executed in an asynchronous manner, ultimately in order to realize a dynamic blockchain-based computing.

REFERENCES

[1] Vladimir Soloviev, “Fintech Ecosystem in Russia”, 2018 Eleventh International Conference “Management of large-scale system development” (MLSD). DOI: 10.1109/MLSD.2018.8551808

[2] Nir Kshetri, and Jeffrey Voas, “Blockchain in Developing Countries”, IT Professional (Volume: 20, Issue: 2, Mar./Apr. 2018). DOI: 10.1109/MITP.2018.021921645

[3] Guido Perboli, Stefano Musso, and Mariangela Rosano, “Blockchain in Logistics and Supply Chain: A Lean Approach for Designing Real-World Use Cases”, IEEE Access, pp(99):1-1 · October 2018

[4] Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System”, 2008, [online] Available: https://bitcoin.org/bitcoin.pdf.

[5] Chris Dannen. 2017. Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners (1st. ed.). Apress

[6] Diego Vegros, and Jaime Saenz, 2010. Peer-To-Peer Networks and Internet Policies, Nova Science Publishers, Inc., NY, USA

[7] Vitalik Buterin, “A Next Generation Smart Contract & Decentralized Application Platform”, *White Paper*, 2014. Available online at: http://www.the-blockchain.com/docs/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[8] Israel Koren and C. Mani Krishna, “Common Network Topologies and Their Resilience,” in *Fault-Tolerant Systems*, San Francisco, CA: Morgan Kaufmann Publishers, 2007, pp. 112-135

[9] Pouria Pirzadeh, Michael Carey, and Till Westmann, "A performance study of big data analytics platforms," *2017 IEEE International Conference on Big Data (Big Data).* DOI: 10.1109/BigData.2017.8258260

[10] Morgen E. Peck. "Blockchains: How They Work and Why They'll Change the World," *IEEE Spectrum* (Volume: 54, Issue: 10, October 2017). DOI: 10.1109/MSPEC.2017.8048836

[11] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler, "The Hadoop Distributed File System," *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST).* DOI: 10.1109/MSST.2010.5496972

[12] Hemang Subramanian. "Decentralized Blockchain-Based Electronic Marketplaces," *Communications of the ACM*, Volume 61, Number 1, December 2017. https://doi.org/10.1145/3158333

[13] Dejan Vujii, Dijana Jagodi, and Sinia Rani, "Blockchain technology, bitcoin, and Ethereum: A brief overview," *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH).* DOI: 10.1109/INFOTEH.2018.834554

[14] Sara Rouhani, and Ralph Deters, "Performance Analysis of Ethereum Transactions in private blockchain," *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS).* DOI: 10.1109/ICSESS.2017.8342866

[15] Mrs. Anamika Chauhan, Om Prakash Malviya, Madhav Verma, and Tejinder Singh Mor, "Blockchain and Scalability," *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C).* DOI: 10.1109/QRS-C.2018.00034

[16] Gavin Wood, "Ethereum: A Secure Decentralised Generalised Transaction Ledger", 2014. Ethereum Project *Yellow Paper*, http://gavwood.com/paper.pdf

[17] Ingo Weber, Vincent Gramoli, Alex Ponomarev, Mark Staples, Ralph Holz, An Binh Tran, and Paul Rimba, "On Availability for Blockchain-Based Systems," *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS).* DOI: 10.1109/SRDS.2017.15

[18] Federico Lombardi, Leonardo Aniello, Stefano De Angelis, Andrea Margheri, and Vladimiro Sassone, "A Blockchain-based Infrastructure for Reliable and Cost-effective IoT-aided Smart Grids," *Living in the Internet of Things: Cybersecurity of the IoT – 2018.* DOI: 10.1049/cp.2018.0042

[19] Qi Zhang, Petr Novotný, Salman Baset, Donna N. Dillenberger, Artem Barger and Yacov Manevich, "LedgerGuard: Improving Blockchain Ledger Dependability," *ICBC* (2018).

[20] Jongho Seol, Abhilash Kancharla, Nicole Park, Nohpill Park, and Indy Nohjin Park, "The Dependability of Crypto Linked Off-chain File Systems in Backend Blockchain Analytics Engine," *International Journal of Networked and Distributed Computing* Vol 6, Dec 2018.

[21] Ying Liu, Kai Zheng, Paul Craig, Yuexuan. Li, Yangkai Luo and Xin Huang, "Evaluating the Reliability of Blockchain Based Internet of Things Applications," *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*, 2018, pp. 230-231, doi: 10.1109/HOTICN.2018.8606026.

[22] Juan Benet, "IPFS – Content addressed, versioned, P2P file system (DRAFT 3)," 2014. arXiv:1407.3561v1, [online] Available : https://ipfs.io/ipfs/QmV9tSDx9UiPeWExXEeH6aoDvmihvx6jD5eLb4jbTaKGps

[23] Abhilash Kancharla and N. Park, "A Realtime Crypto Computing and Block-Dependability," *IEEE SC2 2019*

[24] Abhilash Kancharla, Indy Park, Nicole Park, and N. Park, "Dependable Industrial Crypto Computing," *IEEE ISIE 2019*. DOI: 10.1109/ISIE.2019.8781245

[25] Keke Gai, Yulu Wu, Liehuang Zhu, Zijian Zhang, and Meikang Qiu, "Differential Privacy-Based Blockchain for Industrial Internet-of-Things", *IEEE Transactions on Industrial Informatics* (Volume: 16, Issue: 6, June 2020). DOI: 10.1109/TII.2019.2948094

[26] Keke Gai, Yulu Wu, Liehuang Zhu, Meikang Qiu, and Meng Shen, "Privacy-Preserving Energy Trading Using Consortium Blockchain in Smart Grid," *IEEE Transactions on Industrial Informatics* (Volume: 15, Issue: 6, June 2019). DOI: 10.1109/TII.2019.2893433

[27] Keke Gai, Yulu Wu, Liehuang Zhu, Lei Xu, and Yan Zhang, "Permissioned Blockchain and Edge Computing Empowered Privacy-Preserving Smart Grid Networks," *IEEE Internet of Things Journal* (Vomue:6, Issue:5, Oct 2019). DOI: 10.1109/JIOT.2019.2904303

[28] Fabian Knirsch, Andreas Unterweger and Dominik Engel, "Implementing a blockchain from scratch: why, how, and what we learned," *EURASIP Journal on Information Security*, volume 2019, Article number: 2 (2019)

[29] Di Yang, Chengnian Long, Han Xu, Shaoliang Peng, "A Review on Scalability of Blockchain," *ACM ICBCT'20: Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, March 2020. Pages 1–6. https://doi.org/10.1145/3390566.3391665

[30] Parth Thakkar, Senthil Nathan N, Balaji Viswanathan, "Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform," *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 08 Nov 2018. DOI: 10.1109/MASCOTS.2018.00034

[31] Suporn Pongnumkul, Chaiyaphum Siripanpornchana, and Suttipong Thajchayapong, "Performance Analysis of Private Blockchain Platforms in Varying Workloads," *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, Sep 2017. DOI: 10.1109/ICCCN.2017.8038517

[32] Peilin Zheng, Zibin Zheng, Xiapu Luo, Xiangping Chen, and Xuanzhe Liu, "A detailed and real-time performance monitoring framework for blockchain systems," *ACM ICSE-SEIP '18: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, May 2018. Pages 134–143. https://doi.org/10.1145/3183519.3183546

[33] Mengting Liu, F. Richard Yu, Yinglei Teng, Victor C. M. Leung, Mei Song, "Distributed Resource Allocation in Blockchain-Based Video Streaming Systems With Mobile Edge Computing," *IEEE Transactions on Wireless Communications* (Volume: 18, Issue: 1, Jan. 2019). DOI: 10.1109/TWC.2018.2885266

[34] Shishir Rai, Kendric Hood, Mikhail Nesterenko, and Gokarna Sharma, "Blockguard: Adaptive Blockchain Security," *Distributed, Parallel, and Cluster Computing*, Jul 2019. arXiv:1907.13232 [cs.DC]

[35] Songpu Ai, Diankai Hu, Tong Zhang, Yunpeng Jiang, Chunming Rong, Junwei Cao, "Blockchain based Power Transaction Asynchronous Settlement System," *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring).* DOI: 10.1109/VTC2020-Spring48590.2020.9129593

[36] J. Lind, O. Naor, I. Eyal, F. Kelbert, P. Pietzuch, E.G. Sirer, "Teechain: A Secure Payment Network with Asynchronous Blockchain Access," arXiv:1707.05454

[37] Rafael Pass, Lior Seeman, Abhi Shelat, "Analysis of the Blockchain Protocol in Asynchronous Networks," Published in *Advances in Cryptology* – EUROCRYPT 2017

[38] Wei Chih Hong, Ying Chin Chen, Ren Kai Yang, Bo Li, Jung San Lee, "Efficient peer-to-peer E-payment based on asynchronous dual blockchain," *Journal of Internet Technology*, vol 21. Issue no. 5. Published 2020

[39] Ethereum Average Block Size Chart, https://etherscan.io/chart/blocksize

[40] Jongho Seol, Abhilash Kancharla, Zuqiang Ke, Hyeyoung Kim, Nohpill Park, "A Variable Bulk Arrival and Static Bulk Service Queueing Model for Blockchain," *BSCI '20: Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, October 2020, Pages 63–72, https://doi.org/10.1145/3384943.3409423

[41] Remix Documentation – Ethereum IDE, Edit on GitHub, https://remix-ide.readthedocs.io/en/latest/index.html

[42] Remix – Ethereum IDE, Deploy & Run Transactions in the Blockchain, https://remix-project.org/

[43] web3.js Documentation – Ethereum JavaScript API, https://web3js.readthedocs.io/en/v1.2.9/#

[44] Quan-Lin Li, Jing-Yu Ma, and Yan-Xia Chaing, "Blockchain Queue Theory", arXiv:1808.01795v1 [cs.CE], 6 Aug 2018

[45] Abhilash Kancharla, Jongho Seol, Nohpill Park, Tao Feng, "A Hybrid Chain and A Double-Tuple Variable Bulk Arrival and Static Bulk Service Model," *2021 IEEE International Conference on Blockchain and Cryptocurrency* (ICBC 2021)

[46] Abhilash Kancharla, Jongho Seol, Nohpill Park, Hyeyoung Kim, "Slim Chain and Dependability," BSCI '20: *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, October 2020, Pages 180–185, https://doi.org/10.1145/3384943.3409435

[47] Tian Min, and Wei Cai, "A Security Case Study for Blockchain Games," *2019 IEEE Games, Entertainment, Media Conference (GEM).* DOI: 10.1109/GEM.2019.8811555

[48] Dongjing Xu, Liang Xiao, Limin Sun, and Min Lei, "Game theoretic study on blockchain based secure edge networks," *2017 IEEE/CIC International Conference on Communications in China (ICCC).* DOI: 10.1109/ICCChina.2017.8330529

[49] Ting Chen, Zihao Li, Yuxiao Zhu, Jiachi Chen, Xiapu Luo, John Chi-Shing Lui, Xiaoding Lin, Xiaosong Zhang, "Understanding Ethereum via Graph Analysis," *ACM Transactions on Internet Technology*, April 2020, Article No.:18, https://doi.org/10.1145/3381036

[50] Ethereum. Ethereum Homestead Documentation. Retrieved from http://www.ethdocs.org/en/latest/

[51] Ethereum. Community. Etherscan, The Ethereum Blockchain Explorer. Retrieved from https://etherscan.io/

[52] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen, “A survey on the security of blockchain systems,” *Future Generation Computer Systems.* 107. 10.1016/j.future.2017.08.020.

[53] Galal H.S., Youssef A.M., “Trustee: Full Privacy Preserving Vickrey Auction on Top of Ethereum,” *Financial Cryptography and Data Security.* Science, vol 11599. Springer, Cham. https://doi.org/10.1007/978-3-030-43725-1_14

[54] Chen T. et al. (2017) “An Adaptive Gas Cost Mechanism for Ethereum to Defend Against Under-Priced DoS Attacks,” In: Liu J., Samarati P. (eds) *Information Security Practice and Experience. ISPEC 2017*. vol 10701. Springer, Cham. https://doi.org/10.1007/978-3-319-72359-4_1

[55] Decker C., Wattenhofer R. (2015) “A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels,” In: Pelc A., Schwarzmann A. (eds) Stabilization, Safety, and Security of Distributed Systems. SSS 2015. Lecture Notes in Computer Science, vol 9212. Springer, Cham. https://doi.org/10.1007/978-3-319-21741-3_1

[56] Jakub Sliwinski and Roger Wattenhofer, “ABC: Proof-of-Stake without Consensus,” arXiv:1909.10926 [cs.CR]

[57] Zeta Avarikioti, Eleftherios Kokoris-Kogias, Roger Wattenhofer, and Diony-sis Zindros, “BRICK: Asynchronous Payment Channels”

[58] Sara Rouhani and Ralph Deters, "Performance analysis of ethereum transactions in private blockchain," *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2017, pp. 70-74, doi: 10.1109/ICSESS.2017.8342866.

APPENDICES

Matlab source code for the baseline chain model (VBASBS)

```
iterations =120;
LQ = zeros(1,iterations)
W = zeros(1,iterations)
WQ = zeros(1,iterations)
L = zeros(1,iterations)
Gamma = zeros(1, iterations)
TestP = zeros(1, iterations)
n=10;
Psum = 0;
lambda = 0.005 %  % 0.005 -./ 0.01 -/ 0.03 -d/ 0.05 -h/,
mu = 1/15;


for iterations = 1:iterations
    Q = zeros(1,n);
    a = zeros(1,n);
    P = zeros(1,n);
    temp = ones(1,n);

    for i = 1 : n-1
        temp(i) = 2^(i-1)*(i*n-2*n+7*i-14-i^2)+8;

        a(i) =
   (i*(i+1)/2)*(n/(sqrt(2*pi*n)*(n/exp(1))^n)^2)*temp(i)+i;
        Q(i) = a(i) * (2/((n-i)*(n-i+1)));
    end
        Psum = sum(Q)+1+(lambda/mu)*((n*(n+1))/2);
        Pzero = 1/Psum;
```

```
for i = 1:n-1
        P(i) = Q(i)*Pzero ;
       end


P(n) = (lambda/mu)*((n*(n+1))/2)*Pzero
        Psum = sum(P)+Pzero;


    for i = 1:n
        LQ(iterations) = LQ (iterations) + i * P(i) ;
    end

    for i = 1:n
        WQ(iterations) = LQ(iterations) / lambda;
    end

    for i = 1:n
        W(iterations) = WQ(iterations) + 1/mu;
    end

    for i = 1:n
        L(iterations) = lambda * W(iterations);
    end
    for i = 1:n
        Gamma(iterations) = P(i)*mu;
    end

    for i = 1:n
        TestP(iterations) = P(n);
    end

    n = n + 1;
end

hold on
x = 1:1:(iterations);

plot(x, Gamma,'k x-', 'MarkerIndices', 1:10:length(L))
%plot(x, L, 'k --', 'MarkerIndices', 1:10:length(L)) % 'k .-'
   or 'k .'
%plot(x, TestP, 'k -x', 'MarkerIndices', 1:10:length(L))
%plot(x, LQ, 'k --', 'MarkerIndices', 1:10:length(L)) % 'k .-'
   or 'k .'
%plot(x, W, 'k x-', 'MarkerIndices', 1:10:length(L))
%plot(x, WQ, 'k --', 'MarkerIndices', 1:10:length(L)) % 'k .-'
   or 'k .'
title('Baseline ')

hold off
```

Matlab source code for the adaptive chain model (VBAVBS)

```
iterations =120;
LQ = zeros(1,iterations)
W = zeros(1,iterations)
WQ = zeros(1,iterations)
L = zeros(1,iterations)
Gamma = zeros(1, iterations)
n=10;
mu_n=10;
Psum = 0;
lambda = 0.005% % 0.005 :o/ 0.01 :+/ 0.03 :s/ 0.05 :*/,
mu = 1/15;

for iterations = 1:iterations
    Psum = 0;
    SUM_a1 = 1;
    SUM_a2 = 0;
    SUM_a3 = 0;
    SUM_a4 = 0;
    q = zeros(1,n);
    for i = 1:n
       q(i) = (2*lambda*(n-i+1)) / ((n-i)*lambda*(n-
   i+1)^2+(2*mu));
    end

    for i= 1 : n
        SUM_a3 = 0;
        for j=1 : i
            SUM_a2 = 0;
            for k=1 : i-1
                SUM_a1 = 1;
                for l= 1 : k-1  % Product of l=1 to k-1
                    SUM_a1 = SUM_a1*q(l);
                end
                SUM_a2 = SUM_a2 + k * SUM_a1;
            end
            SUM_a3 = SUM_a3 + j * SUM_a2;
        end
        SUM_a4 = SUM_a4 + q(i)*(SUM_a3 + i);
    end

    P0 = 1/(1+(SUM_a4));


    Psum = 0;
    SUM_a1 = 1;
    SUM_a2 = 0;
    SUM_a3 = 0;
    SUM_a4 = 0;
```

```
    for i= 1 : n
        SUM_a3 = 0;
        for j=1 : i
            SUM_a2 = 0;
            for k=1 : i-1
                SUM_a1 = 1;
                for l= 1 : k-1  % Product of l=1 to k-1
                   SUM_a1 = SUM_a1*q(l);
                end
                SUM_a2 = SUM_a2 + k * SUM_a1;
            end
            SUM_a3 = SUM_a3 + j * SUM_a2;
        end
        SUM_a4 = SUM_a3 + i;
        P(i) = q(i)* P0 * SUM_a4;
    end

    Psum = sum(P) + P0;

    for i = 1:n

        LQ(iterations) = LQ(iterations) + i * P(i);

        Gamma(iterations) = Gamma (iterations)+ P(i)*mu/(n-(i-
  1));
     end

        WQ(iterations) = LQ(iterations) / lambda;

        W(iterations) = WQ(iterations) + 1/mu ;

        L(iterations) = lambda * W(iterations);


    n = n + 1;
end


hold on

x=1:1:(iterations);

plot(x, L, 'k :s', 'MarkerIndices', 1:10:length(L))
plot(x, Gamma, 'k :s ', 'MarkerIndices', 1:10:length(L))
plot(x, LQ, 'k :s', 'MarkerIndices', 1:10:length(L))
plot(x, W, 'k :s', 'MarkerIndices', 1:10:length(L))
plot(x, WQ, 'k :s', 'MarkerIndices', 1:10:length(L))
%hold off
```

Matlab source code for the asynchronous chain model (VBAABS)

I. Fully Asynchronous Chain model

```
iterations =120;
LQ = zeros(1,iterations)
W = zeros(1,iterations)
WQ = zeros(1,iterations)
L = zeros(1,iterations)
Gamma = zeros(1, iterations)
n=10;
Psum = 0;
Psumbefore = 0;
lambda = 0.05 %  % 0.005 -./ 0.01 -/ 0.03 -d/ 0.05 -h/,
mu = 1/15;


for iterations = 1:iterations
    Q = zeros(1,n);
    a = zeros(1,n);
    P = zeros(1,n);

    for i = 1 : n-1

        a(i) = i*lambda;
        Q(i) = a(i) * ((n-(i-1))/mu);
    end
        Psumbefore = 1+((lambda/mu)*((n*(n+1)*(n+2))/6));
        Pzero = 1/Psumbefore;

    for i = 1:n-1
        P(i) = Q(i)*Pzero ;
    end

        P(n) = (1/mu)*i*lambda*Pzero
        Psum = sum(P)+Pzero;


    for i = 1:n
       LQ(iterations) = LQ(iterations) + i * P(i) ;
    end

    for i = 1:n
       WQ(iterations) = LQ(iterations) / lambda;
    end

    for i = 1:n
       W(iterations) = WQ(iterations) + 1/mu;
    end
```

```
    for i = 1:n
       L(iterations) = lambda * W(iterations);
    end
    for i = 1:n
       Gamma(iterations) = Gamma(iterations) + P(i)*mu;
    end
    n = n + 1;
end

hold on
x = 1:1:(iterations);

%plot(x, Gamma, 'k -o', 'MarkerIndices', 1:10:length(L))
%plot(x, L, 'k -x', 'MarkerIndices', 1:10:length(L)) % 'k .-'
   or 'k .'
%plot(x, P, 'k -*')
%plot(x, LQ, 'k -s', 'MarkerIndices', 1:10:length(L))
plot(x, W, 'k -o', 'MarkerIndices', 1:10:length(L))
%plot(x, WQ, 'k -x', 'MarkerIndices', 1:10:length(L))
title('Fully asynchronous')

hold off
```

II. Staged Asynchronous Chain model

```
iterations =120;
LQ = zeros(1,iterations)
W = zeros(1,iterations)
WQ = zeros(1,iterations)
L = zeros(1,iterations)
Gamma = zeros(1, iterations)
n=10;
Psum = 0;
lambda = 0.005; %  % 0.005 -./ 0.01 -/ 0.03 -d/ 0.05 -h/,
mu = 1/15; % set 1/15
s = 10;

for iterations = 1:iterations
    Q = zeros(1,n);
    a = zeros(1,n);
    P = zeros(1,n);
    outer = zeros(1,n)


    for i = 1 : n-1
        if rem(i, s) == 0
            outer(i)= (lambda/mu)*(((s-i)*(s-i+1)))/2
        else
            temp(i) = 2^(i-1)*(i*n-2*n+7*i-14-i^2)+8;
            a(i) =
  (i*(i+1)/2)*(n/(sqrt(2*pi*n)*(n/exp(1))^n)^2)*temp(i)+i;
            outer(i) = a(i) * (2/((n-i)*(n-i+1)));
        end
    end


        Psum = sum(outer)+1+(lambda/mu)*((n*(n+1))/2);
        Pzero = 1/Psum;

    for i = 1:n-1
        P(i) = outer(i)*Pzero ;
    end
        P(n) = (lambda/mu)*((n*(n+1))/2)*Pzero
        Psum = sum(P)+Pzero;



        for i = 1 : n
            LQ(iterations) = i * P(i)
        end


    for i = 1:n
```

```
        WQ(iterations) = LQ(iterations) / lambda
    end


    for i = 1:n
       W(iterations) = WQ(iterations) + 1/mu
    end

    for i = 1:n
       L(iterations) = lambda * W(iterations)
    end
    for i = 1:n
        if rem(i, s) == 0
           Gamma(iterations) = Gamma (iterations) + P(i)*mu
        else
            Gamma(iterations) = P(i)*mu
        end

    end
    n = n + 1;
end

hold on
x = 1:1:(iterations);

plot(x, Gamma, 'k -.','MarkerIndices', 1:10:length(L))
%plot(x, L, 'k -o','MarkerIndices', 1:10:length(L)) % 'k .-'
   or 'k .'
%plot(x, P, 'k -*')
%plot(x, LQ, 'k -o','MarkerIndices', 1:10:length(L))
%plot(x, W, 'k -o', 'MarkerIndices', 1:10:length(L))
%plot(x, WQ, 'k -o','MarkerIndices', 1:10:length(L))
%title('Baseline vs. Adaptive')

hold off
```

.

VITA

Jongho Seol

Candidate for the Degree of

Doctor of Philosophy

Dissertation: A STUDY ON QUANTITATIVE DESIGN FOR DYNAMIC BLOCKCHAIN-BASED COMPUTING

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2021.

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in 2005.

Completed the requirements for the Bachelor of Science in Control and Instrumentation Engineering at Kangwon National University, Samcheok, Kangwon, Republic of Korea in 2000.

Experience:

Teaching Instructor for CS 1003, Computer Proficiency online and regular courses in the Department of Computer Science at Oklahoma State University, Stillwater, OK, from Spring 2019 to Summer 2021

Proposal Senior Engineer and System Lead Engineer at Emerson Automation Solutions, Seongnam, Korea, from Nov 2007 to June 2017

Research Software Engineer at LG Electronics Mobile Handset Research and Development Center, Seoul, Korea, from June 2005 to Oct 2007