A STUDY ON HIGH-PERFORMANCE AND DEPENDABLE BLOCKCHAIN-BASED COMPUTING

By

ABHILASH KANCHARLA

Bachelor of Technology in Electronics and Communications
Jawaharlal Nehru Technological University
Hyderabad, India
2009

Master of Science in Computer Sciences
Oklahoma State University
Stillwater, OK
2017

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
December, 2020

A STUDY ON HIGH PERFORMANCE AND

DEPENDABLE BLOCKCHAIN-BASED COMPUTING

Dissertation Approved:

Nohpill Park

Dissertation Adviser

Christopher Crick

Esra Akbas

Kim Jonghoon

Name: ABHILASH KANCHARLA

Date of Degree: DECEMBER 2020

Title of Study: A STUDY ON HIGH PERFORMANCE AND DEPENDABLE BLOCKCHAIN-BASED COMPUTING

Major Field: COMPUTER SCIENCE

Abstract: Blockchain technology is undergoing a tremendous growth choking itself up to its capacity and performance limits. It is exigently sought to address and respond to these issues, being mostly concerned about the scalability, performance and dependability of the blockchain system. This research will address and investigate on various, yet critical performance and dependability issues and problems as identified in blockchain-based crypto computing with specific respect to the on/off-balanced chain, the real-time chain; the slim chain; and a hybrid chain as to be proposed in this research. A hypothetical and theoretical design of each proposed crypto computing solution is developed in order to establish an engine for preliminary yet extensive parametric simulation, and the results are demonstrated and validated through an isolated testing built on Ethereum and Hyperledger open source-based prototype. With dependability referred to as the likelihood to be performed as desired, each hypothetical and theoretical model is built centered around the dependability of each proposed crypto solution to accommodate capabilities of the on/off-balanced crypto computing, the real-time computing, the slim-computing and the hybrid computing. A dependability model for each proposed crypto solution has been identified and defined along with various performance variables, and has ultimately provided a theoretical yet practical understanding of each crypto solution. A prototype, to demonstrate each proposed crypto solution and to validate its 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 focused on performance and dependability.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

Blockchain [1, 2] as a technology is undergoing tremendous changes in the recent years given the boom in the crypto currency market. While these changes are attributed mostly towards the crypto currency there are quite a few limitations that hinder blockchains rise in popularity in other domains such as B2B, health, insurance etc. This research addresses those limitations of the blockchain by implementing new algorithms, thereby improving the performance of the blockchain in areas primarily focused on speed, size, security and connectivity.

A blockchain is a data structure that makes it possible to create a digital ledger of data and share it among a network of independent parties. Depending on the sharing properties of the blockchain there are different types of structures defined when it comes to how much of the blockchain us meant to be visible: public, permissioned and private blockchains. Public blockchains are ones those are open to anyone to access and participate in the blockchain [6]. Bitcoin and Ethereum are two of the most well-known public blockchains. A permissioned blockchain will have control over which individuals can join the network.

Blockchains have laid a foundation where the need for trust has been taken out of the equation. Where before asking for “trust” was a big deal, with blockchains it’s small. The content of permissioned blockchain can be accessed only with appropriate access to the blockchain and one such example of a permissioned blockchain is Ripple. Blockchains incorporate hashing, which transforms any size of data into fixed length values. This process is done continuously from the genesis block (starting block in the blockchain). Any change to the transactions in the blockchain will change the hash addresses of all the subsequent hash addresses. This is the primary reason for blockchain being not so easy modifiable. Although there are multiple blockchain frameworks out in the market at the time of this writing, the entire research is implemented on Ethereum blockchain. In the blockchain world, consensus is the process of developing an agreement among a group of commonly mistrusting shareholders. These are the full nodes on the network. The full nodes are validating transactions that are entered into the network to be recorded as part of the ledger. Each blockchain has its own algorithms for creating agreement within its network on the entries being added [35].

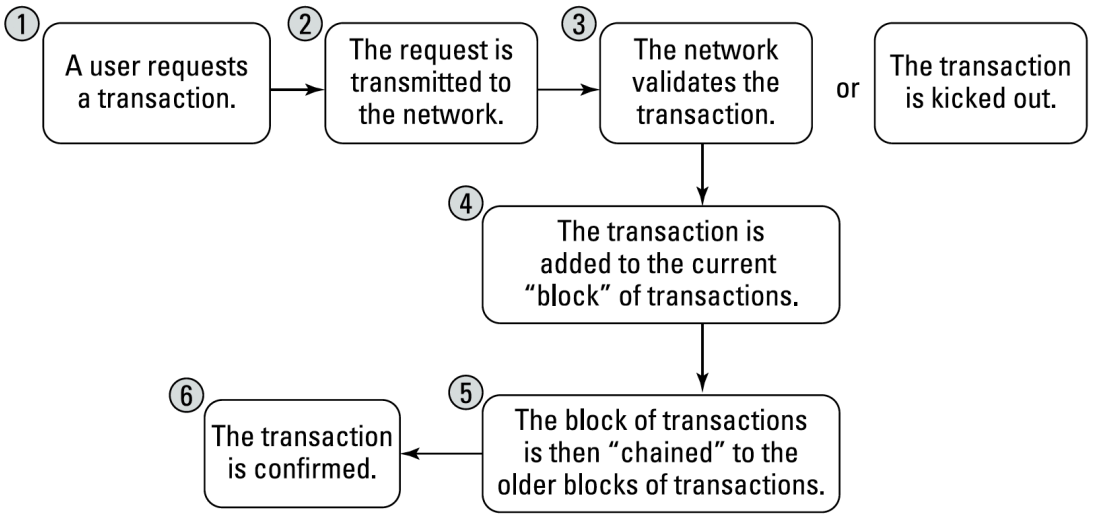


Figure 1. How blockchains work [35]

The primary advantages of blockchain are:

- No third-party involvement in making financial transactions: Unlike transferring fiat currency through banks and other financial institutions, a blockchain doesn't need a third-party to create trust in transferring crypto currency.
- Secure and publicly maintained ledger: The blockchain is structured in a Merkle tree, thereby making it near impossible to modify any content. Any modification that is done will affect all the subsequent hash addresses. Moreover, a public blockchain gives the transparency to the user.
- Impossible to modify any content that has already posted to the blockchain in the form of transactions.

This research focuses on disadvantages of blockchain, either by completely trying to overcome the disadvantage or by creating an alternate solution. The main disadvantages of blockchain are as:

- Latency: The average time to post a transaction in blockchain is much higher than a normal transaction involving fiat currency through any financial institution. It is proposed to address this problem by implementing a real-time blockchain wherein the transactions are picked up based on different sampling techniques to enhance the speed with which they are processed into the blockchain.
- Closed Network: A blockchain in general is not designed [4] to access any information from the internet thereby making it a closed network. An interface was developed to answer the likes of higher gas fees and also making the blockchain open to posting transactions from outside the network. Crypto solution has been addressed for both Intra-connectivity and inter-connectivity issues.
- 51% Attack: Given any security flaws in the blockchain or any breaches, the proposed checkpoint and rollback algorithm is designed to revert the blockchain to its previous data.

There are different types of blockchain based on the needs and requirements for the blockchain. The types of blockchain in brief are as follows:

- Public networks are large and decentralized, anyone can participate within them at any level — this includes things like running a full node, mining cryptocurrency, trading tokens, or publishing entries. They tend to be more secure and immutable then private or permissioned networks. They're often slower and more expensive to use. They're are secured with a cryptocurrency and have limited storage capacity with no capability to store any private information.
- Permissioned networks are viewable to the public, but participation is controlled. Many of them utilize a cryptocurrency, but they can have a lower cost for applications that are built on top of them. This feature makes it easier to scale project and increase transaction volume. Permissioned networks can be very fast with low latency and have higher storage capacity over public networks.
- Private networks are shared between trusted parties and may not be viewable to the public. They're very fast and may have no latency. They also have a low cost to run and can be built in an industrious weekend. Most private networks do not utilize a cryptocurrency and do not have the same immutability and security of decentralized networks. Storage capacity may be unlimited.

Blockchain technology is undergoing a breath-taking growth, choking itself up to its capacity and performance limits, mandating the departure of potentially significant volume of code and data off the chain, in order to relieve the blockchain spatially as well as temporally, thereby ultimately addressing and resolving the exigently sought scalability. However, as demonstrated in, the attempts to relieve the supposed-to-be on-chain volume of data and code to off-chain space are risky potentially affecting on the dependability of the computation in a negative manner unless otherwise countered, even if

expectedly offering performance benefits either spatially or temporally, which triggers the proposed research on the slim chain.

The dependability [23, 24, 29, 31] of the slim chain will be modeled with respect to such various variables as the on- and off-chain dependability from a single transaction's standpoint, the crypto rate, and, namely, the IPFS rate, and along further with the read rate from the off-chain space or the write rate to the off-chain space. Note that the off-chain space will serve as a storage for the code and data migrating from on-chain space, and IPFS will be the option under consideration for the purpose. Notice that, without loss of intuition, write operations from on-chain to off-chain will cause less potential dependability loss because they at the moment of writing do not influence the dependability back on-chain, whereas read operations will expose the on-chain dependability to potential risks of dependability loss by migrating potentially undependability-contaminated transactions back on the chain, and thus write and read rates are taken into account as two primary variables of the dependability model of the proposed slim chain. In this context, it is proposed in this research that rather than storing the details of the entire transaction onto the blockchain, the transaction is hashed into a 32-bit hex address. Only the hash address is stored in blockchain and the transaction itself resides on distributed ledger, namely, Inter Planetary File System (IPFS).

The research is organized as follows: the proposed approach to modeling and analysis of the dependability of the proposed on-off blockchain, slim chain, realtime chain and hybrid chain is described in the following section: each section is followed by demonstration of extensive parametric simulations versus various crypto solution-related parameters and results are shown along with a prototype demonstration in each section; then conclusions are drawn in the last section followed by Appendix which highlights some complex mathematical equations.

CHAPTER II

PRELIMINARIES AND REVIEWS

Blockchain technology [1] is undergoing a tremendous growth choking itself up to its capacity and performance limits and thereby resulting in cost spike. It is exigently sought to address and respond to these issues, being mostly concerned about the cost spike issue. In this context, firstly, it is proposed in this work particularly how to mitigate the cost of blockchain transactions specifically in terms of the gas fee on Ethereum [2] by developing a scheme which involves on- and off-chain transactions mixed (referred to as crypto mix) at a certain ratio (referred to as Crypto ratio), and managed and operated through an interface developed in house. Then, secondly, it is proposed how to maintain the authenticity and security of the off-chain transactions by using a crypto (i.e., on-chain) checkpoint and rollback protocol as proposed in this work. The interface developed in house provides a seamless context switching in between on- and off-chain transactions and it is demonstrated that significant cost saving would be realized depending on the size and type of the computations by the transactions such that the larger the size is and the more complex the computation is, the more substantially the cost saving is realized.

It is observed that there exists an optimal mix of on- and off-chain functions to minimize the expected total gas fee. In order to justify the effectiveness of the proposed crypto mix/ratio scheme, it is mandated to maintain the authenticity and security of the transactions executed on the off-chain functions. A novel crypto checkpoint and rollback algorithm is proposed in an adaptive manner with two different checkpoint intervals, one at a fixed length and another at a variable length triggered and adjusted by a suspicious transaction detected and reported by the backend blockchain analytics engine. A quantitative figure of merit [10], [11] is modeled and referred to in order to determine whether the blockchain analytics result hits beyond a threshold point or not.

Lastly, an on-chain voting is conducted every time a context switching takes place across the border between on- and off-chains in order to manage and exercise the centralized off-chain transactions in a decentralized manner on-chain. An experimental prototype is developed and a variety of decentralized applications [14] are developed to demonstrate various and extensive crypto mixes and ratios. Various checkpoint intervals and rollback frequencies are tested to demonstrate their efficiency and effectiveness with respect to the figure of merit of interest.

The performance of the transactions on blockchains is considered to be one of the most critical issues for the blockchain technology to address and resolve for any further breakthrough and momentum gain [16,19,20,22]. Scalability [17,23] is the immediate concern to do with the performance of the transactions and is primarily governed by the speed of the transactions without loss of practicality. The speed and along with appropriate prioritization and scheduling of the transactions in general, if possible, are ultimately doomed to influence the deadline requirements in real-time systems [18].

In this context, an analytical study is conducted in this work on the block-dependability. The block-dependability in this work is defined by the probability all the transactions within the

block-deadline (i.e., the deadline for a transaction to be posted in the target block) are to be posted within the target block. The analytical study will employ a probabilistic approach to identify a set of performance-variables that are doomed to influence the block-dependability such as the number of transactions, the average size of transactions, the speed (bytes/sec) of transactions with respect to deadline, gas fees and the number of miners, to mention a few. This set of variables but not limited to them will be taken into account with an implicit influence by miners as well, for the modeling and analysis of the block-dependability in the context of real-time blockchain-based crypto computing.

The block-dependability will be evaluated against various transaction prioritization and selection methods [23] such as normal (as is being exercised by the current Ethereum blockchain), random (as the baseline), sorted (as a heuristic based on certain characteristics such as gas fee by miners), and stratified (as another heuristic for a statistical optimization in the course of transaction selection process). Extensive analytical experiments will be conducted to reveal the direction of optimization how to realize the exigently-desired scalability and how to ultimately realize the real-time execution of transactions under stringent deadline requirements as more and more apps are in demand of real-time capability as the technology matures.

This research presents a work on how to assure the dependability of a crypto system built across on and off the blockchain by using the proposed adaptive checkpoint and rollback algorithm, and a prototype is developed for demonstration purpose. The theoretical background of the proposed checkpoint and rollback algorithm is studied to characterize the variables affecting the dependability such as security, 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 trigger a rollback or not. Based on the variables characterization in a stochastic manner, then steady state probabilities and state transition probabilities are derived in order to assure the ultimate effective dependability of each individual dependability variable (i.e.,

security, authenticity and reliability), 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, a protype of a crypto system is 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-chain blockchains, with an experimental result of a benefit in the gas fee. This is the most exigently addressed issue today in blockchain systems especially in Ethereum network of blockchains. An astonishing gas fee saving results are demonstrated. It is observed that the crypto system benefits more if more computationally intensive transactions are executed off-chain while vice versa.

LATENCY

This research also proposes an analytical approach how to design and realize a crypto computing (Ethereum blockchain-based) under stringent real-time requirement. In order to evaluate the efficacy of the approach, a new analytical metric is defined and developed to estimate the dependability, referred to as the block-dependability. The proposed block-dependability precisely models the probability for the pending transactions to be posted within the current, in other words, within the target block delay, namely, within the deadline required if their expected execution times are within the temporal range of the deadline. Various methods how to prioritize 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, are proposed and simulated. 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 to reveal the

influence of each variable on the block-dependability, versus each of those proposed prioritization and selection methods.

SIZE OF BLOCKCHAIN

This research proposes a crypto solution to address the size of the blockchain with slim-chain facilitated by IPFS. The proposed slim-chain distinguishes itself from the light-chain such that light-chain attempts to mitigate the size and delay overhead and issues of the blockchain by shrinking the extent of the synchronization window, whereas the proposed slim-chain attempts to slim down the size of the blockchain by keeping the smart contracts and data in an off-chain decentralized storage, namely, IPFS. Instead of keeping the entire transactions and data on the chain, selections are made on the transactions and data in a certain manner and store them off the chain in a decentralized storage such as IPFS. In order to address and resolve any potential reliability and security loopholes (referred to as the dependability together) due to the off-chain storage involvement, a new analytical model is developed to assure the dependability of the crypto computing with respect to the off-chain storage, and the off-chain storage is to be managed in a decentralized manner in association with the blockchain as well. The proposed decentralized management of the off-chain storage is to create an off-chain crypto link (e.g., a hash address) to a partition of the off-chain storage and those are kept on blockchain in place of the full load. An effective and efficient algorithm is proposed to select the contents under the influence of various constraints and conditions such as frequency ratio of off-chain reads issued by on-chain transaction versus writes in communication with the target off-chain data; and the number of off-chain crypto links, to mention a couple. If read/write ratio goes high, the overall dependability will be more influenced by and sensitive to the dependability of the off-chain storage; and there will be a trade-off between the number of off-chain crypto links and the speed of the off-chain transactions and an optimal balance between them will be sought as the more number of off-chain crypto links the slower the execution of the transactions involving off-chain data access. An

extensive parametric simulation will be performed, and a set of parameters will be identified for the optimal design for dependability of the proposed slim-chain-based crypto computing.

A few key concepts and components of blockchain are listed as follows:

- Public Key: Every node/user in the blockchain is assigned a public/private key. A transaction to be made on the blockchain has to be signed by a private key, while the public key is always visible to everyone on the blockchain.
- Transaction: A transaction contains the details of the both the public keys (in case of smart contract [16] – contract address), hash of the previous block, current block number and the amount of crypto currency being transferred to along with the gas need to execute the transaction; block is a list of transactions recorded into the distributed ledger over time. The transactions are grouped together and put it in the block which are then mined to be added to the blockchain. The immutability of the blockchain assures that the transaction once recorded in a block cannot be deleted or undone.
- Chain: All the blocks have a unique block number. The block number is incremented by 1 for every block that is added to the chain. The hash in blockchain is created from the data that was in the previous block. The previous block hash will also be added to the current block details. Thereby, any change in the previous block completely alters the entre hash which would invalidate the blockchain.
- Crypto Currency: Ether is the crypto currency in Ethereum blockchain. Any transaction to be posted on the blockchain utilizes the respective crypto currency of the blockchain; Gas: The amount of ether needed to post a transaction in Ethereum blockchain. The current blockchain is coded in such a way that the transaction with highest gas fees is given higher priority to be included in the block. This proposed real-time, details out different algorithms that can be used to order transactions differently in the block; Miner: Nodes that participate in creating a new block are called miners. Miners are responsible

for adding blocks to the blockchain. The blocks are created based on the number of transactions present in the txnpool and other characteristics like gas fee, block size, arrival time.

- Transaction Pool: Transaction pool is an array datatype of transactions containing the pointers to the transactions that are not yet posted to the block. Pending transactions are the transactions that are not yet posted to the blockchain and these pending transactions reside in the transaction pool [29]. The different ordering of the transaction pool creates different mining [14, 15] algorithms.
- Mining: The central process of blockchain-based computing to establish a trust among the nodes in the network connected in a P2P manner. Miners compete for the next new hash code which requires a computationally intensive process and is highly costly. In this context, there have been efforts made extensively to mitigate or eliminate the mining process.

There have been reported on various yet critical performance and dependability problems in [17-20], where extensive research have been conducted on theoretical designs of a few blockchain-based solutions in order to establish a theoretical yet substantial foundation. As the ultimate quality of crypto computing will be determined by its likelihood to be performed as commanded or desired, referred to as the dependability, those theoretical models emphasized and centered around the dependability of each of those crypto solutions to accommodate such capabilities as the on/off-balanced crypto computing [12], the slim-computing [30], the real-time computing [32], and the hybrid computing [31]. A theoretical study on performance will be in an ultimate interest to identify a theoretical intersection versus the dependability, which is the ultimate objective of the proposed Variable Bulk Arrival and Variable Bulk Service (VBAVBS) Model with λ_F in the context of the proposed realtime chain.

Crypto solution that addresses the inter-connectivity is a proposed double-tuple queueing model for hybrid chain, based on the single tuple VBASBS queueing model which is to be extended to a double-tuple to address the two chains into account in a Markovian manner, through which the effectiveness, efficiency and versatility of the VBASBS model will be demonstrated in this research. A novel queueing model of the type $M^{1,n}/M^{n}/1$ has been proposed in order to establish a quantitative model to guide the design of a blockchain-based network in Ethereum. The model assumes bulk arrivals of transactions in Poisson distribution, i.e., $M^{1,n}$, and static bulk service of transactions in exponential time, i.e., M^{n}, for posting a transaction in the current block, namely, VBASBS, while for double-tuple queues, is extended from the single-tuple queueing model as in VBASBS. In the proposed double-tuple VBASBS, the same assumptions are made such that the variable bulk arrival rate be assumed to vary linearly proportional to the size of the transactions in a multiple of λ per slot, respectively for each queue, 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 across multiple slots for posting in a block.

A hybrid chain has been proposed to investigate on a new blockchain network that is to be built across private and main nets, namely. In the hybrid chain, the dependability was the primary interest and concern of the work to build a dependable interface in between private and main nets if it is to support business to consumer (or vice versa) transactions for instance. A dependable interfacing across private and main nets, is one of the most critical design factors such that it be ensured that private transactions stay under the private control and public transactions stay public in the main net, and further in order to manage a seamless yet dependable execution of transactions across the border. The efficacy of the privacy of the private net and the publicity of the main net are addressed and modeled in the dependability model by tracing transaction's stochastic processes.

CHAPTER III

ON/OFF-BALANCED CHAIN

Checkpoint and rollback, abbreviated as CPR, is an algorithm designed to mitigate the security risks in a public blockchain. The primary role of the checkpoint and rollback is to reverse all possible transactions thereby ensuring that the system is reset to a checkpoint which is created earlier. In order to ensure security to the blockchain [3], checkpoints are made at every stage whenever necessary and then these checkpoints can be used to revert back the transactions executed with the cause of the checkpoint with insecurity (or an undependability). Checkpoints are either created manually or have a timed interval to create checkpoints into the system automatically. Note that the checkpoint and rollback cannot be implemented for all the transactions on the blockchain as there are few restrictions that are pointed out later in this chapter.

CPR will also not be possible on transactions that involve transferring of ether. The primary reasons for not being able to perform CPR on a monetary transaction are:

1. There be only one possession of private key as the other private key will be in the possession of the user [15]. Hence without a private key a transaction cannot be made just by using the public key.
2. Even if the private key is in procession, the funds transferred to the public address might have been used/transferred by the time rollback is initiated. Hence, the reversal of the transaction will not be possible as there will not be enough funds in the transferred account to transfer back.
3. Likewise, transactions involving transferring funds to a smart contract function as private key of a smart contract is in possession of the user that has created the smart contract in the first place.

Checkpoint and rollback (CPR) are possible on some types of transactions, detailed as below:

1. Transactions in which a variable is being written into smart contract. During a rollback, the variable could be written back to the previous value based on the checkpoint selected. Since blockchain records all the data in the form of blocks, the user could choose a checkpoint based on the time of the block. The rollback is assumed to be done through the same public key that is used to create the contract. In general, checkpoint and rollback are performed: To increase further security in the blockchain. Given a case where a user accidentally changed a variable in the smart contract, checkpoint and rollback can be useful to revert the changes. Checkpoint and rollback does not delete any transactions that are posted in the blockchain, but will be able to post transactions back again such that the state of the smart contract is reverted to the previous checkpoint.

2. Given a flaw in a smart contract, an intruder changes the contents of the smart contract. CPR will be able to revert the transactions changes mitigating the security attack.
3. CPR can also be performed to reset the state of the smart contract. Rather than create a new smart contract, CPR could be used to reset the smart contract to its default settings.

Some possible scenarios that can be posted for check pointing in the blockchain are as follows: Transaction history, Ether transfer, Change in the smart contract variables. The proposed procedure of CPR is as follows:

1. All the transactions towards a smart contract are recorded off-chain along with checkpoints, through which the value of the variables before the checkpoint is rolled back to incase needed. These checkpoints are kept track of in another smart contract, which also include the voting procedure.
2. A rollback proposal can be made through the voting smart contract. This rollback proposal will be timed for a period, before which the proposal should reach a predetermined number of votes for the rollback to happen. If in any case the number of votes is not reached, the proposal is discarded. Voting and proposing for a rollback will be only available to specific set of users in the blockchain, i.e., not everyone connected to the blockchain can vote for a proposal or raise one. The users who will be allowed to vote or propose will be determined later.
3. While rolling back, analysis must be performed on which other transactions are affected by the rollback indirectly due to the change in the values of the variables.

In the proposed checkpoint and rollback algorithm is the checkpoint interval is determined in an adaptive manner. The pseudo code for the checkpoint and rollback algorithm is shown in Appendix A.

Notice that the current checkpoint interval is adjusted to the difference of the new checkpoint time and the last checkpoint time to proactively respond to the report from the backend analytics engine and to adapt to the higher potential security threat by shortening the current checkpoint interval. On the other hand, if the check pointing routinely continues without any threat detected, then the current checkpoint interval can be extended back to prevent a monotonic decrease of the current checkpoint interval as shown in the else statement in the algorithm. To manage the checkpoint and rollback in a decentralized manner, voting will be conducted every time a rollback is to be performed or an adjustment in the interval of checkpoint. Note that checkpoint and rollback, itself, is a centralized algorithm and the proposed voting will mediate the centralization into a decentralization.

STATE TRANSITION MODEL

There are two states that CPR would stay in: Secure and Insecure. Considering the two states, the model in Figure 1, presents a state transition diagram with the states, namely, P_{secure} and $P_{insecure}$, and the state transitions between those two states, namely, $q_{check-secure}$ and $q_{rollback}$ as defined below as well. Note that security is one of the variables that to define the dependability and in Figure 1, the model for the security is presented and likewise a model can be drawn for each variable, respectively.

P_{secure}: the probability for the system to be secure

$P_{insecure}$: $1 - P_{secure}$

$q_{check-secure}$: the rate for checkpoint to check secure

$q_{rollback}$: the rate for rollback to occur successfully

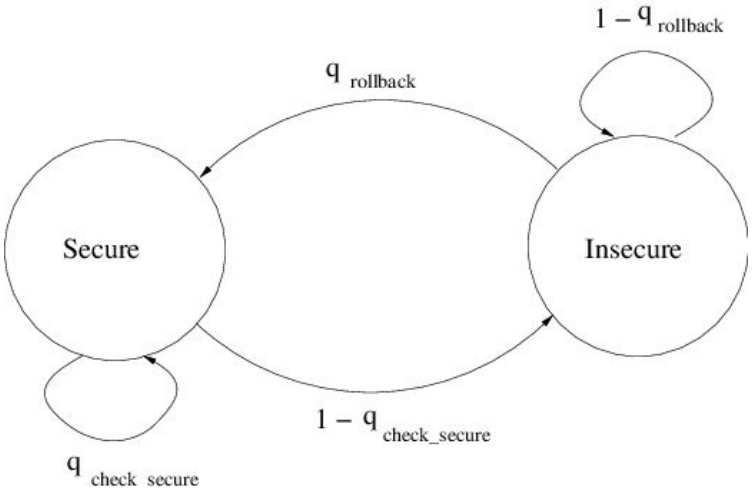


Figure 1. The proposed model for P_{secure}

In Figure 1,

$$q_{check-secure} = 1 - p_{hit}$$

where p_{hit} represents a rate to take a hit by an event such as hacking (in case of security concern), impersonation (in case of authentication concern), or failure (in case of reliability concern), to mention a few to which the proposed checkpoint and rollback algorithm can be applied.

$q_{rollback}$ can be expressed in a binomial manner as follows:

$$q_{rollback} = p_{hit} + 0(1 - p_{hit})$$

when hit by an event. Then, when the hit event is detected through a procedure such as a backend blockchain analytics engine for an attack, impersonation or failure, $q_{rollback}$ can be expanded as follows:

$$q_{rollback} = p_{hit}\left(p_{detect} + 0(1 - p_{detect})\right)$$

Then, in order to manage the potentially centralized process in a decentralized manner, a voting process can be employed to draw a consensus on whether to take a rollback to the last known safe

state of the system or not by the nodes involved on the blockchain network of interest. Thus, $q_{rollback}$ can be further expanded as follows.

$$q_{rollback} = p_{hit}\left(p_{detect}\left(p_{vote} + 0(1 - p_{vote})\right)\right)$$

Therefore, $q_{rollback}$ can be expressed as follows to take all those processes into account.

$$q_{rollback} = p_{hit} * p_{detect} * p_{vote}$$

The effective security ($Security_{eff}$) is defined as follows:

$$Security_{eff} = (1 - cryptorate) * P_{secure}$$

where $cryptorate$ refers to the rate of crypto (i.e., on-chain) portion in the system of concern, in other words, $(1 - cryptorate)$ refers to the rate of the off-chain portion. For simplicity, the security on-chain portion is perfect. The steady state equations of the model are derived as follows:

$$P_{secure} = P_{secure} * q_{check-secure} + P_{insecure} * q_{rollback}$$

$$P_{insecure} = P_{secure} * (1 - q_{check-secure}) + P_{insecure} * (1 - q_{rollback})$$

$$P_{secure} + P_{insecure} = 1$$

The equations are solved at equilibrium as follows:

$$P_{secure} = \frac{q_{rollback}}{1 - q_{check-secure} + q_{rollback}}$$

$$P_{insecure} = \frac{1 - q_{check-secure}}{1 - q_{check-secure} + q_{rolback}}$$

Therefore, the effective security can be obtained in a closed form as follows:

$$Security_{eff} = \frac{(1 - cryptorate) * q_{rolback}}{1 - q_{check-secure} + q_{rollback}}$$

Likewise,

$$Autheticity_{eff} = \frac{(1 - cryptorate) * q_{rollback}}{1 - q_{check-authentic} + q_{rollback}}$$

$$Reliability_{eff} = \frac{(1 - cryptorate) * q_{rollback}}{1 - q_{check-reliable} + q_{rollback}}$$

After all, a compound dependability can be expressed as follows:

$$Dependability_{compound} = w_{security} * Security_{eff} + w_{authenticity} * Authenticity_{eff} + w_{reliability} * Reliability_{eff}$$

where note that $w_{security} + w_{authenticity} + w_{reliability} = 1$,

and the distribution of weights is determined by the nature of the architecture and transactions involved. Based on the analytical models developed, numerical simulations have been conducted and the results are shown in Figures 2 and 3. In Figure 2, P_{secure} versus $q_{check-secure}$ are plotted at various $q_{rollback}$ values (e.g., 0.99, 0.9, 0.8, 0.7 and 0.5). It is observed as expected P_{secure} grows at a slightly higher rate than linear and at a steeper rate approaching $q_{check-secure} = 1$.

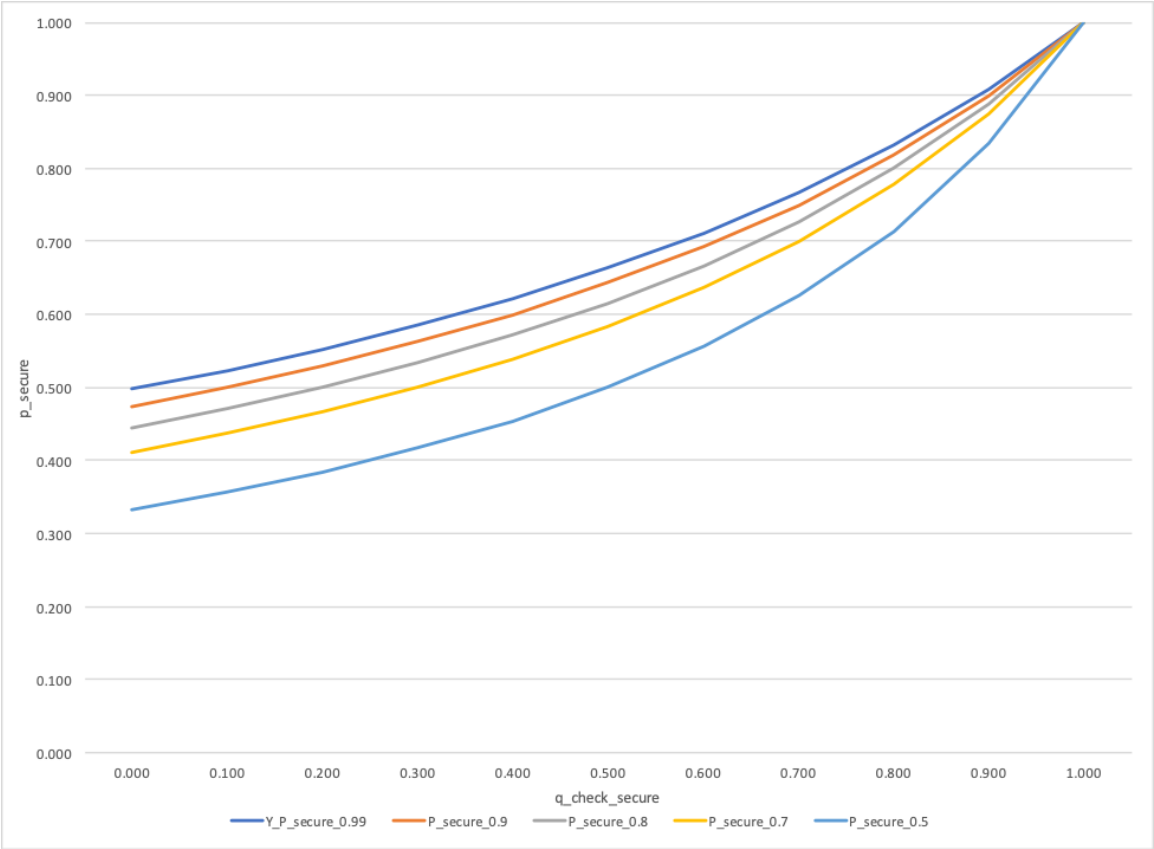


Figure 2. P_{secure} vs $q_{check-secure}$ at various values of $q_{rollback}$

In Figure 3, P_{secure} versus $q_{check-secure}$ are plotted at various $q_{rollback}$ values (e.g., 0.99, 0.9, 0.8, 0.7 and 0.5) and the $cryptorate$ (e.g., 0.1, 0.33, 0.5, 0.66, 0.9). Note that the crypto rate of 0.66 (i.e., 33% of the computation resides on-chain and the rest (i.e., 66% off-chain) has been realized in this work and its experimental results are shown in this work in terms of gas fee savings and execution time (i.e., block delay) later in this research using an interface that enables data to flow into the blockchain from outside the network.

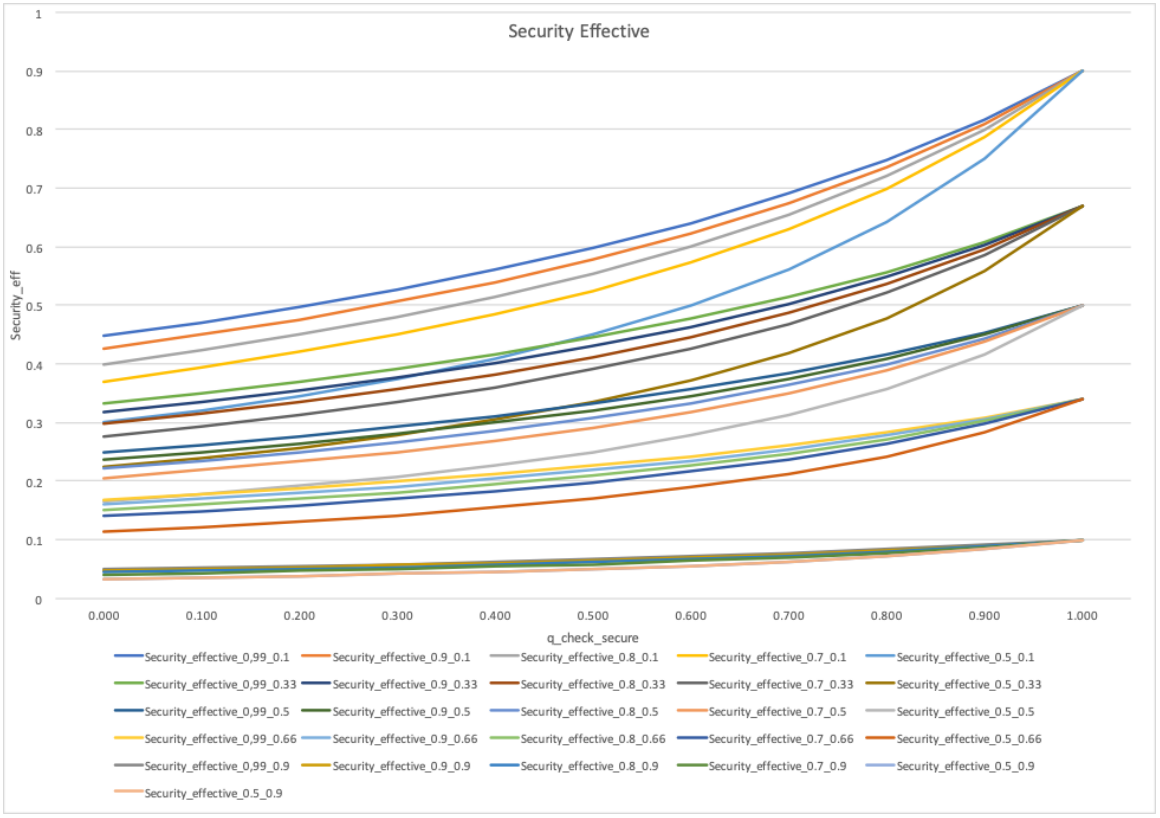


Figure 3. $Security_{effective}$ vs $q_{check-secure}$ at various values of $q_{rollback}$and $cryptorate$

THE PROPOSED INTERFACE BETWEEN ON- AND OFF-CHAIN

In this section, the interface developed in this work to coordinate the processes executing across back and forth on and off-chain is introduced and demonstrated, which triggers the need for the proposed checkpoint and rollback process. A blockchain network [1], [4], [12] in itself cannot access any information that resides on the web off-chain given the deterministic nature of blockchain. In order to access such information, an interface agent is proposed and developed as an API design which handles the movement of information from off-chain to on-chain and vice-versa. The interface agent will be fully synced with the current Ethereum blockchain. The interface agent will have a continuous watch on all transactions being posted in blockchain and

filters out the transactions that references the contract address. Such transactions are then processed off-chain and the results are posted back to the blockchain via another off-to-on transaction through interface. API will ultimately be the interface that helps make BaaS (Blockchain as a Service) computing model by connecting seamlessly to the cloud and blockchain. Interface agent will be integrated with a number of blockchain protocols and the oracle service will be also used for non-blockchain applications such as accessing simple information from the internet through blockchain. The functionality of the interface is to not only provide a login access to cloud server by posting the transaction in the public blockchain thereby maintaining transparency but also to be able to pull/process data from the blockchain to perform analytics [8] in the backend with Hadoop located off the blockchain [13]. The proposed interface will also hold various references (e.g., hash codes) which can be accessed from within the blockchain thereby providing simpler on-to off access.

The proposed interface agent is applicable to Ethereum blockchain and can be generalized for other types of blockchains as well. Each block has transactions that are cryptographically signed by the person starting the transaction and also contains a reference hash to the previous block. All the transactions posted in the public blockchain become immutable and are publicly visible to anyone accessing the same blockchain. Blockchain is designed to be non-deterministic, i.e., a node connected to the blockchain cannot access information outside the blockchain. Blockchain network in general is not designed to access data over the web in real time, the reason being that a result from web can change from time to time and is not deterministic. Hence, the work of accessing data outside the blockchain network is done by Oracles. Oracles are trusted data feeds that send information to the smart contracts thereby removing the need for smart contract to contact outside their network [4 – 6].

Note that Interface agent will have the transactions that are being processed to access functions both on- and off-chain posted onto the blockchain. The conventional systems would only have the

transactions that are being accessed from off to on-chain posted on the blockchain. The performance of Interface agent is measured as to how fast the requests to the transactions are processed and the result posted back to the blockchain. The targeted performance level is that all transactions in the new block are processed and posted in the immediate next block. Any transactions that are processed late or have any technical difficulties such as the server not responding to the off-chain request and such are sent to another queue where they are processed as soon as ready.

In order to further increase the performance of the Interface agent as a whole, the watch that is being put on the blockchain is timed at intervals so that the API determines an exact time at which the new block would be formed in the blockchain. Note that the interval time that the API will wait before watching the blockchain is determined after studying the pattern of new block times in Ethereum network for the previous blocks. Figure 4, presents the proposed interface model.

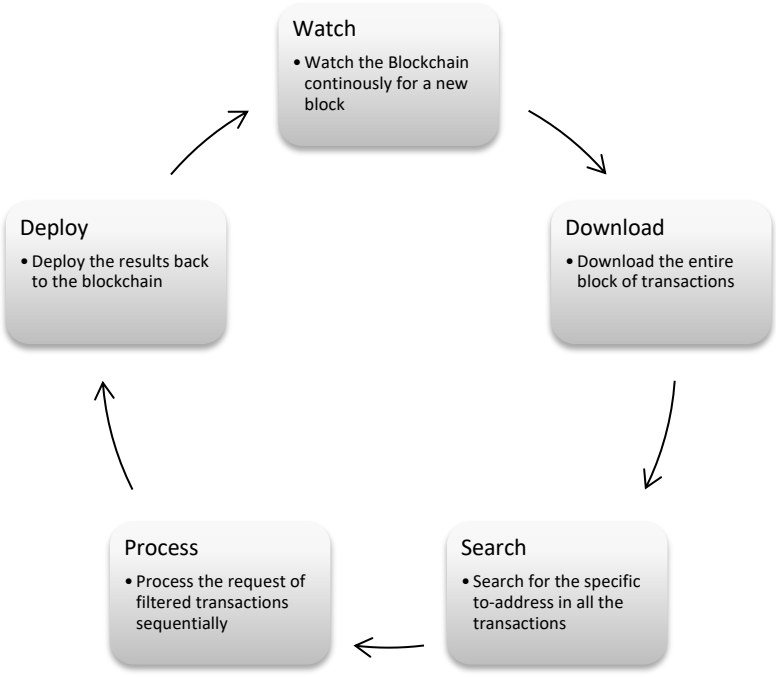


Figure 4. Interface model

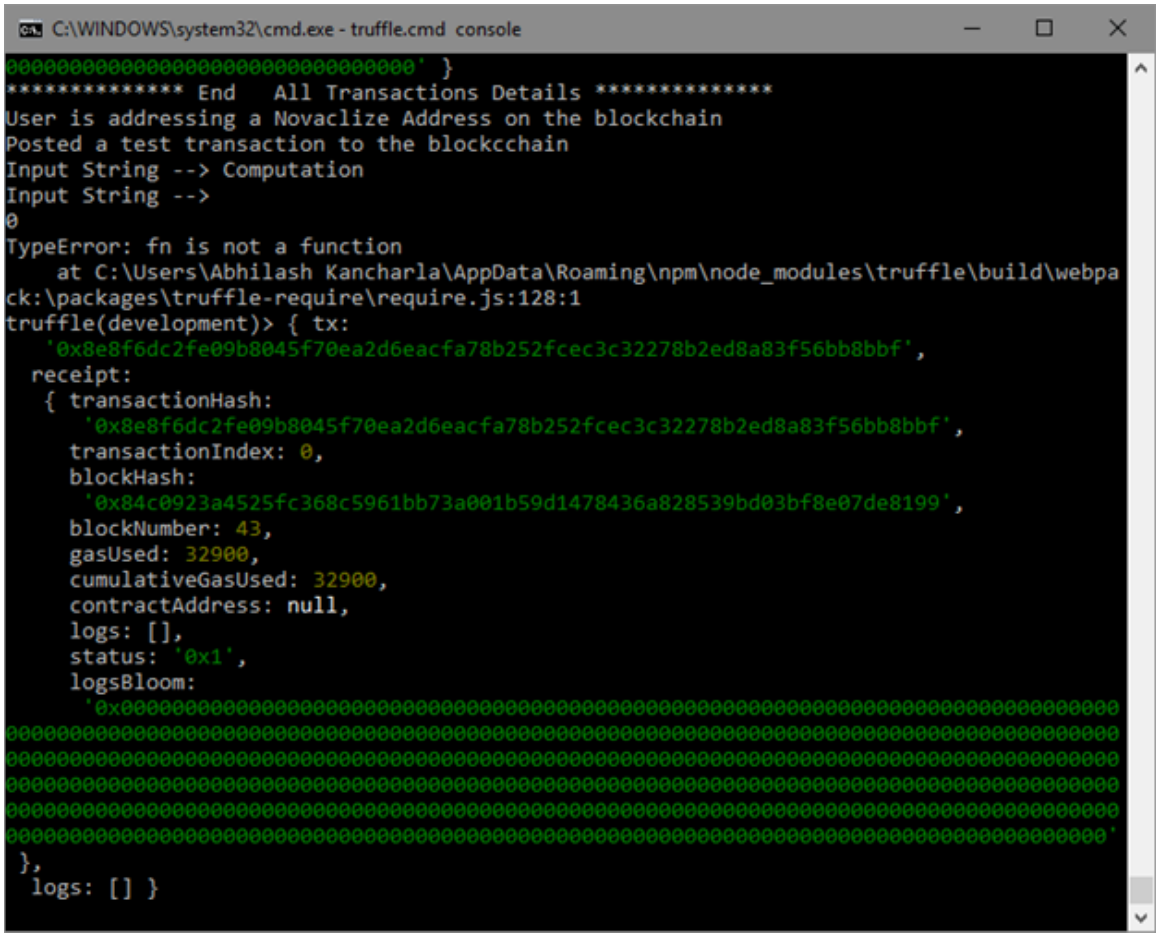


Figure 5. Transaction posted by the user of the blockchain to the specified contract

An example is shown in Figure 5 which depicts a transaction made by the user of the blockchain to a contract that the interface has a watch on. As soon as a new block is detected in the block-chain, all the transactions in the blockchain are scanned for any transactions referring to pre-determined contract. All the requests from the transactions referring to Interface agent are processed offline and the results are posted back to the block-chain. Figure 6 shows the transaction that was posted automatically by the interface once the request is processed; and Figure 7 shows the script posting the result of the processed transaction back to the blockchain.

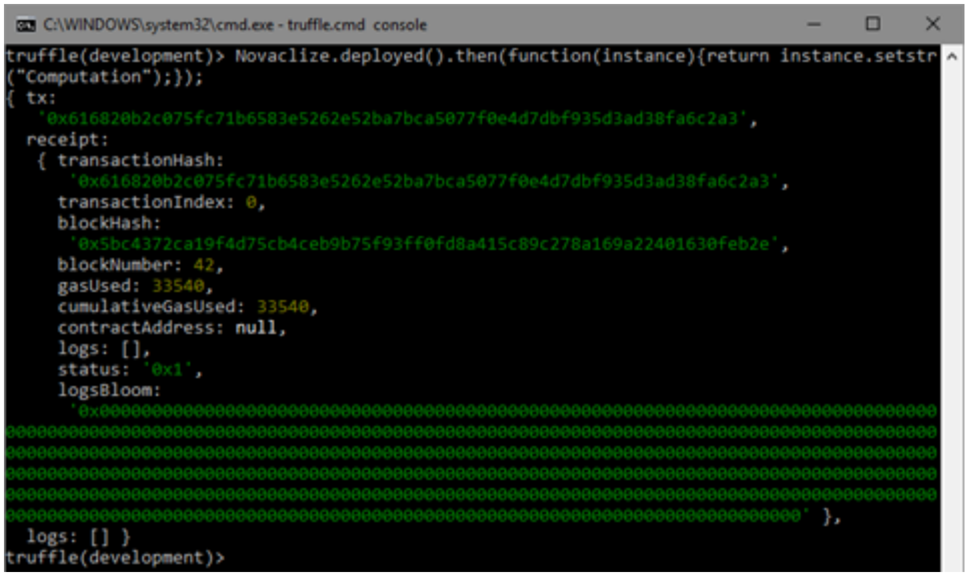


Fig 6. Script posting the result of the processed transaction back to the blockchain

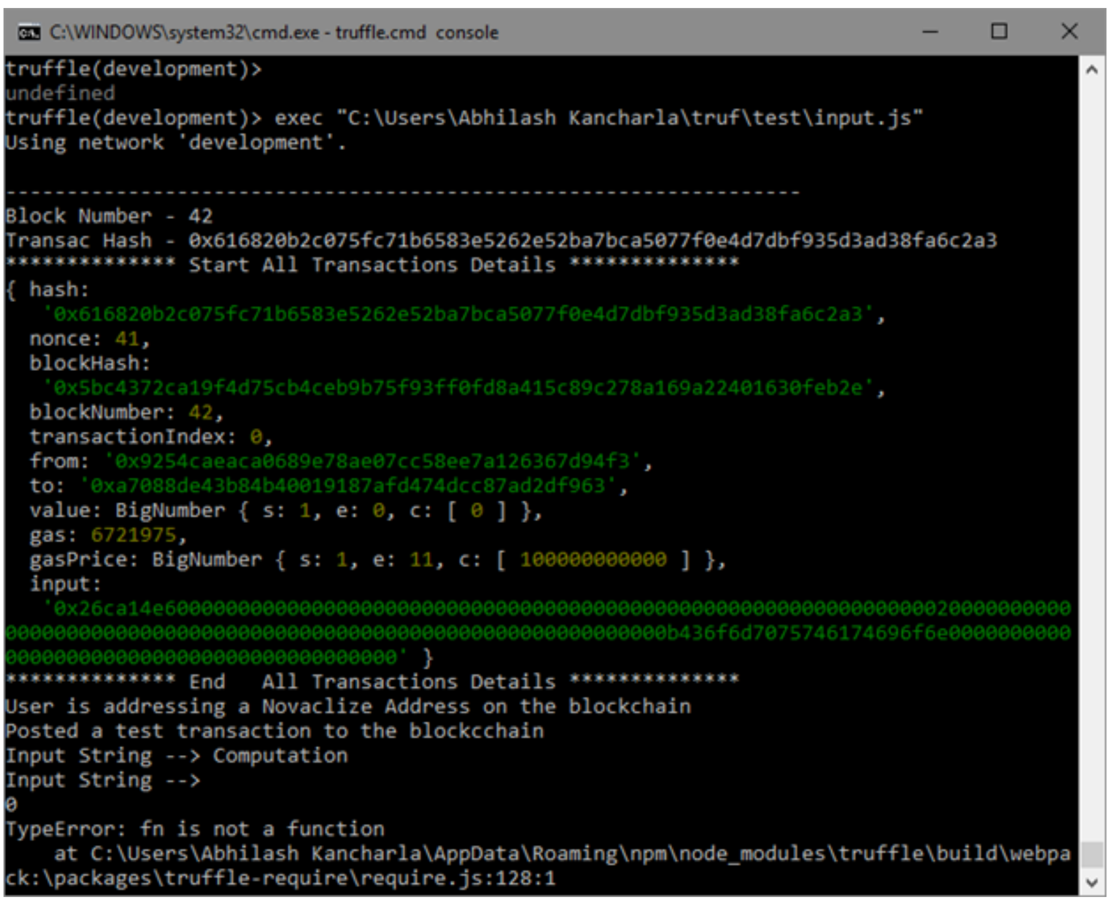


Fig 7. Script processing the transaction that is addressed to the specified contract

Interface will be able to handle different types of requests. Firstly, accessing a variable or a data from the http/https website and returning back to the user. Secondly, the user can also call a computational function. Calling a computational function can be costly if done inside the blockchain, Interface will help reduce the gas of the transaction by performing the computation offline and posting back the result to the blockchain. Figures 8 and 9 show the details of the transaction that were posted on to the blockchain by the interface captured by a GUI application called Ganache.

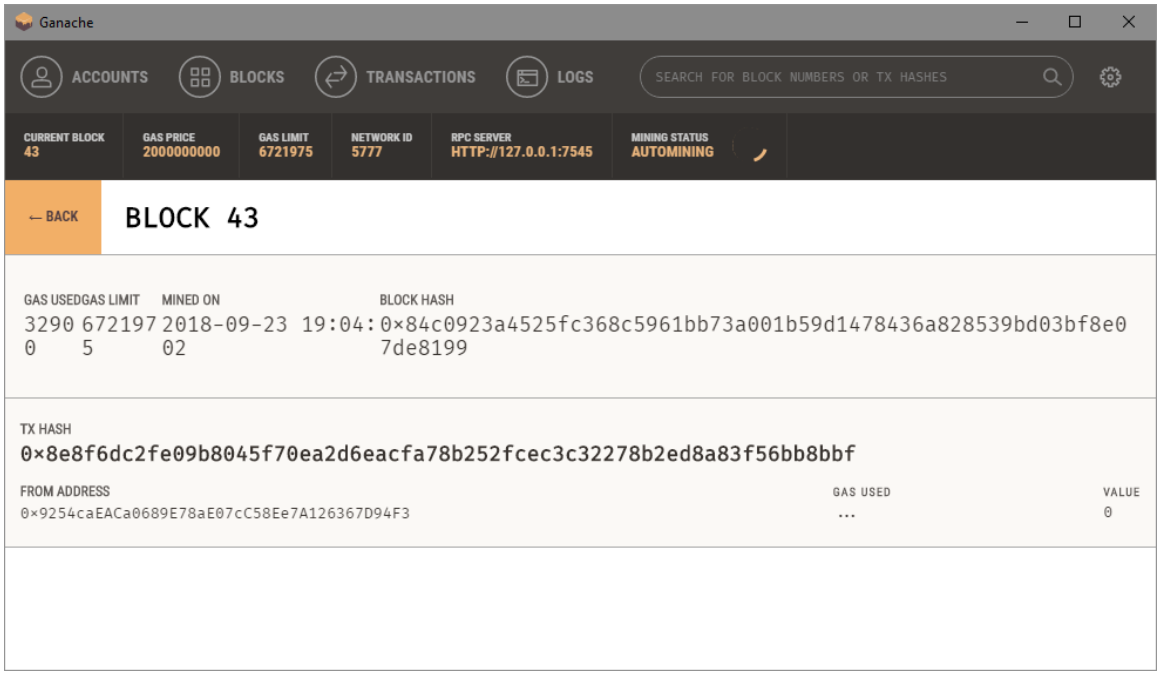


Fig 8. Posted transaction in block 43 in the blockchain

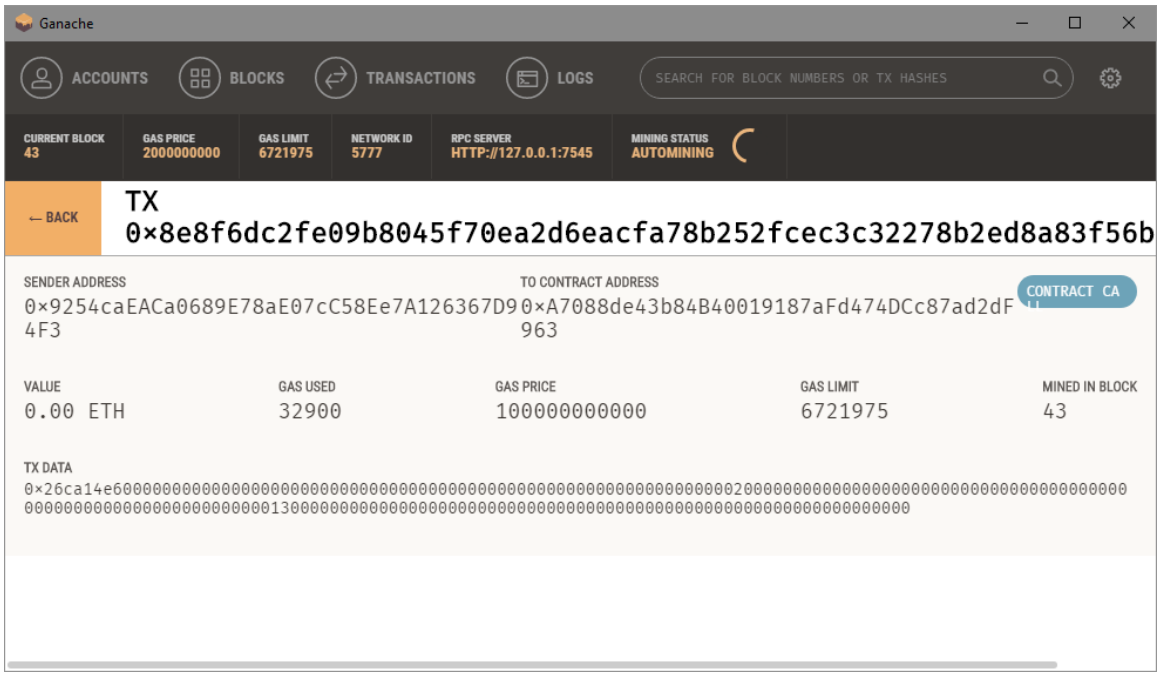


Fig 9. Details on the transaction hash posted on the blockchain

Interface agent will be able to revolve around three entities in general:

- Data Source: Data source is the source of information the user wants to access through Interface agent. There can be different aspects of data sources such as Information from cloud service regarding the correct login details; Information regarding a statistic that can be obtained from internet commonly known as web2; Information regarding the analytics related to the apps functions that are being performed by Hadoop [8], [9]; and not only information getting but also information posting, such as storing a piece of data on a cloud server or even a website, to mention a few. Note that Interface agent is supposed to provide all the above functionalities when it comes to dealing with data sources.
- Query: The query is the function that user uses to access different data sources that Interface agent provides. To distinguish from the data source that the user is wanting to access, a query function will contain a string argument apart from the usual arguments

needed for the data source. This way Interface agent will be able to identify the purpose of the query the user is accessing.

- Interface agent: The current network that the Interface agent is proposed to run in on Ethereum network. Interface agent after processing the current block is also supposed to interact with Hadoop [5] nodes that perform the analytics of all the B2B transactions that are processed through Interface agent in the Ethereum network. Interface agent will return back with the appropriate result by looking up the query from the entire stack of blockchain that Interface agent has downloaded. This would be faster, given that the query will not be searched from the blockchain again resulting in a delayed response, rather the query will be searched in the offline blockchain data that Interface agent maintains using Hadoop. The bigger the blockchain the slower would be the response for searching the entire blockchain. In this case, Hadoop is employed in the background to fasten up the search responses.

Given that Interface can access off-chain information through the blockchain, it is proposed that the entire smart contract can be split up into on-chain and off-chain. The amount of contract that rests on off-chain is left for the user to decide. Interface will have different options laid out and for each transaction posted into the block-chain the user will get it choose from the various crypto mix options wherein each option proposes different gas prices, execution times and security levels involved inside. A recommendation of the crypto mix is provided to the user based on the average gas prices of the transaction and other attributes. This crypto mix is further analyzed to see if that is the right choice or not for the customer using the Interface.

Fig 10. HTML Homepage

Fig 11. Crypto Rate / Mix

IMPLEMENTATION

Clicking on 1.3.4 on the homepage in Figure 10 displays four different types of smart contracts uploaded to blockchain as shown in Figures 11, 12, and 13, in which all the smart contracts have the same functionality with the difference as illustrated below: The steps to control the crypto rate and mix is shown in Figure 14.

1. Case 1: 100% Crypto on chain (Intensive computation) 100% crypto refers that all the functionality of the smart contract is located within the smart contract itself and any calls made to these functions are executed within the blockchain. The results and transaction fee for case 1 is displayed as shown in Figure 15.
2. Case 2: 66% Crypto Off chain (Intensive computation) 66% crypto refers to the fact that some of the functionality is present outside the blockchain. The calls to these functions are requested through placeholder functions which do not contain the functionality itself. The interface picks up the request accessed to these functions and processes and posts the result back to the blockchain. The reason this is 66% crypto is that the original smart contract had 3 functions and one of the functions is put off-chain, 66% of the earlier smart contract resides on the blockchain and the 33% approximately of the functionality reside off chain. The results and transaction fee for case 2 is displayed as shown in Figure 16.
3. Case 3: 100% crypto on chain (Simple computation) similar to #1, but the computation made inside the function is less intensive, i.e., takes fairly less gas compared to the same transaction posted in #1. The results and transaction fee for case 3 is displayed as shown in Figure 17.
4. Case 4: 66% crypto on chain (Simple computation) similar to #2, but the computation made inside the function is less intensive, i.e., takes fairly less gas compared to the same

transaction posted in #2. The results and transaction fee for case 4 and comparison among four cases is displayed as shown in Figure 18.

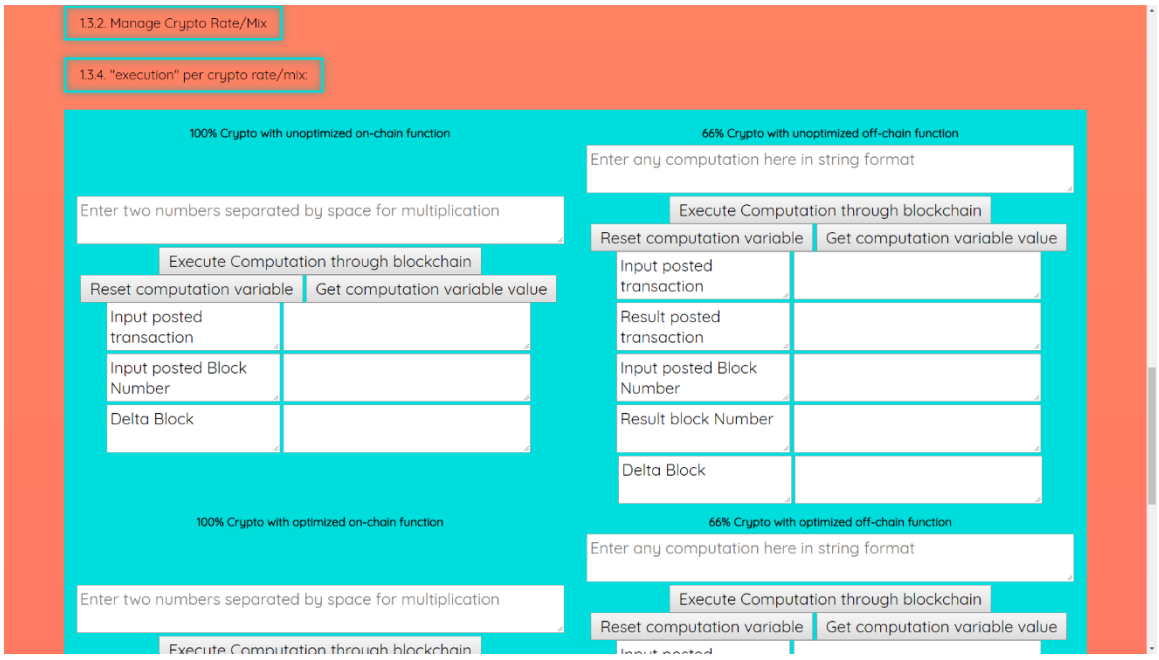


Fig 12. Four smart contracts – first part

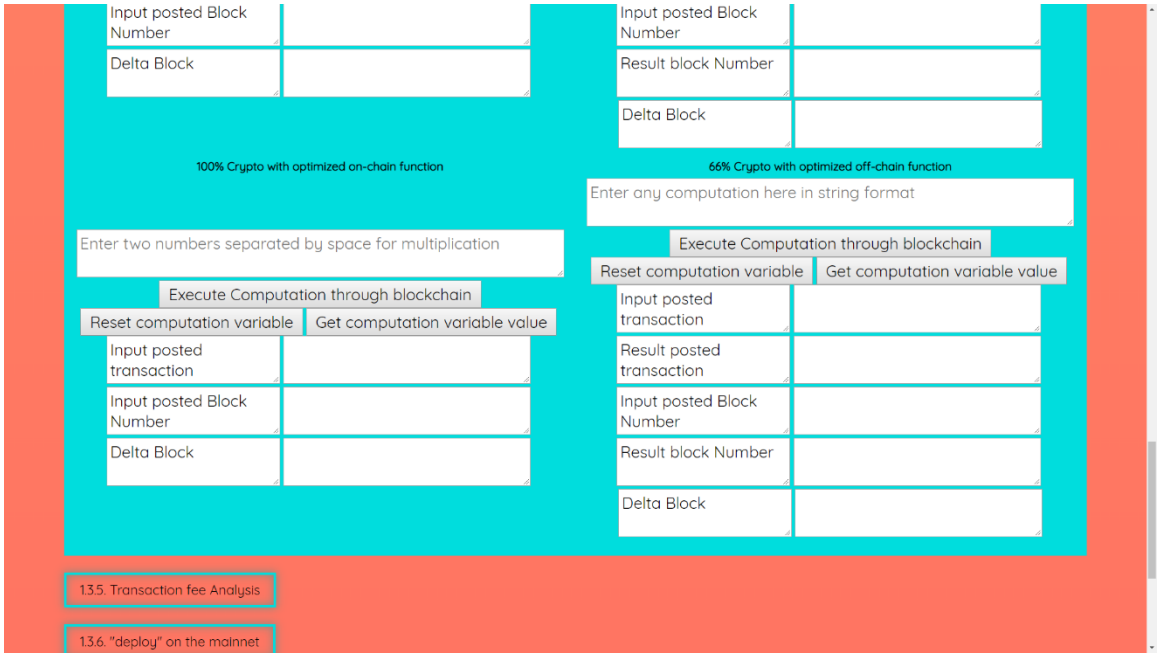


Fig 13. Four smart contracts – second part

Note that in Figure 18, all the four different smart contracts are made to perform multiplication with the same input. It is observed that each transaction posted uses different gas, where notice that the 100% crypto smart contract uses significantly more gas than 66% crypto smart contract. Therefore, it is concluded without loss of generality that the lower crypto rate (i.e., less portion of the code reside on-chain) the higher chance of saving gas fee (i.e., transaction fee) if the off-chain portion is computationally intensive yet less security sensitive.

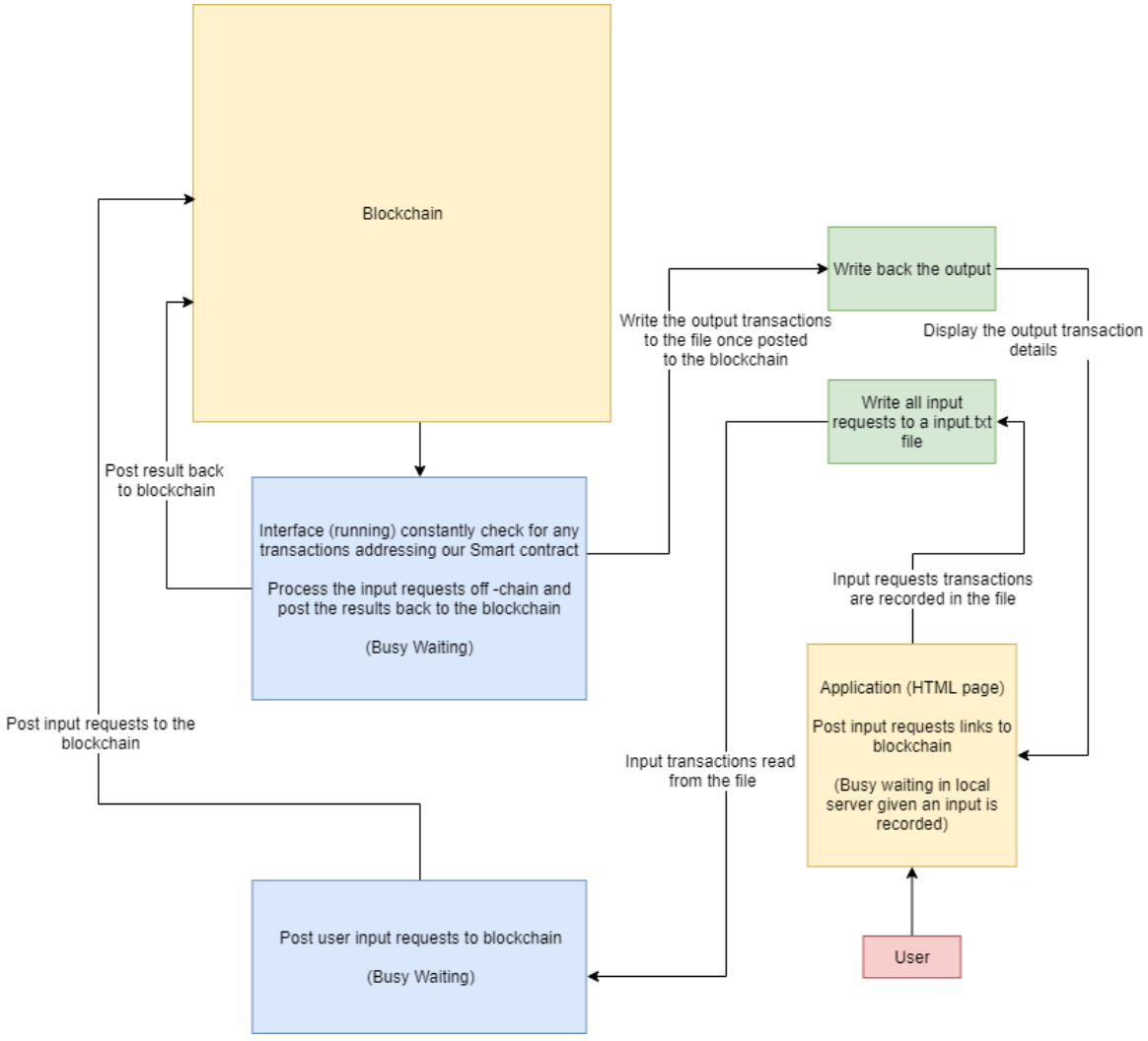


Fig 14. Crypto Rate / Mix flowchart

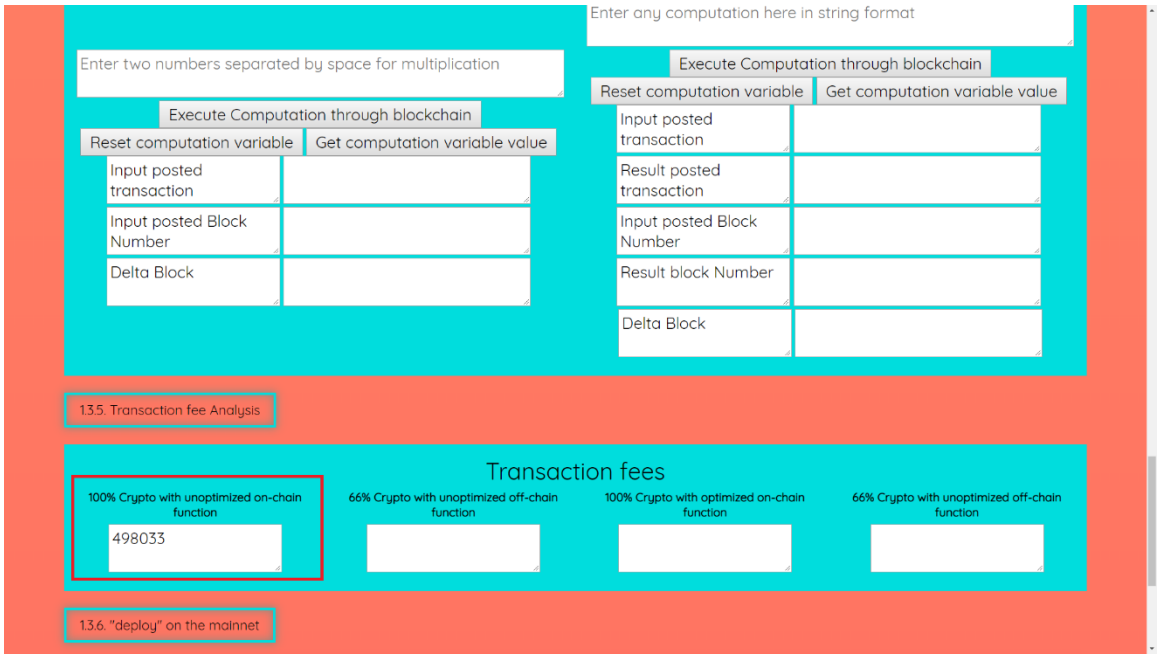


Fig 15. Case 1: Results and gas fee

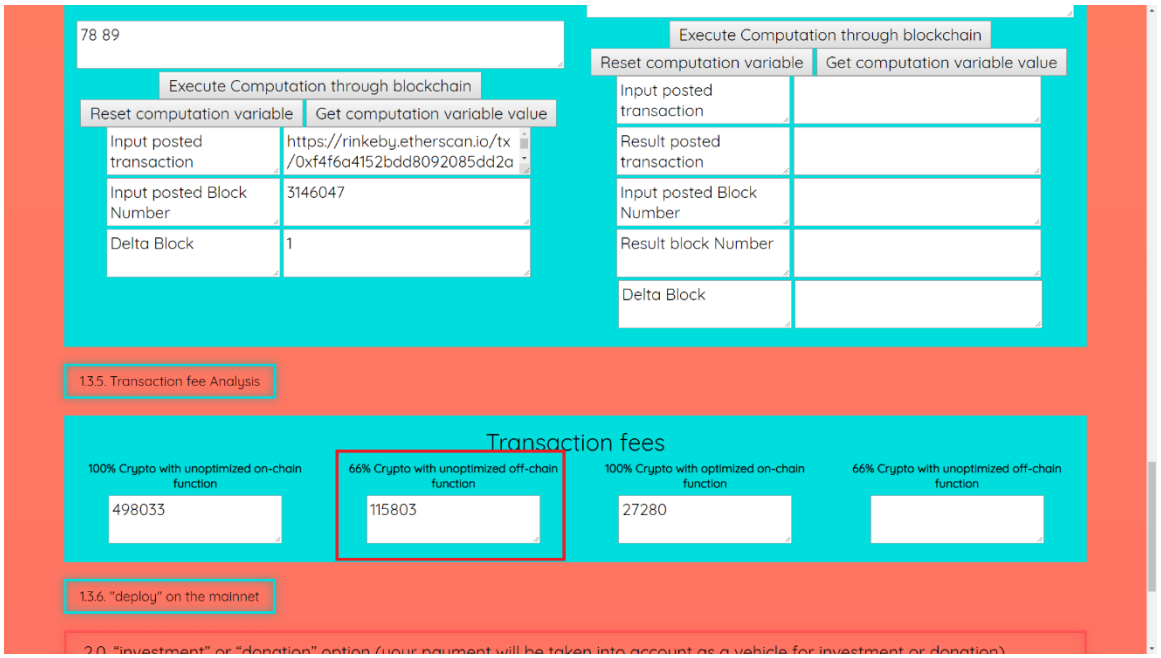


Fig 16. Case 2: Results and gas fee

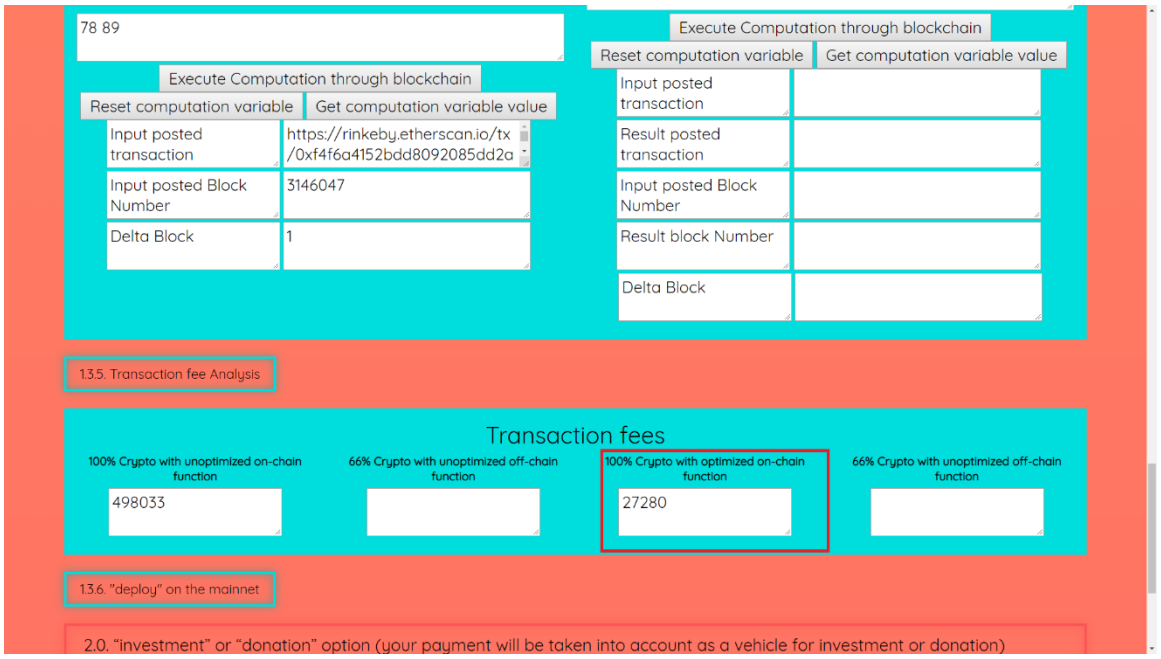


Fig 17. Case 3: Results and gas fee

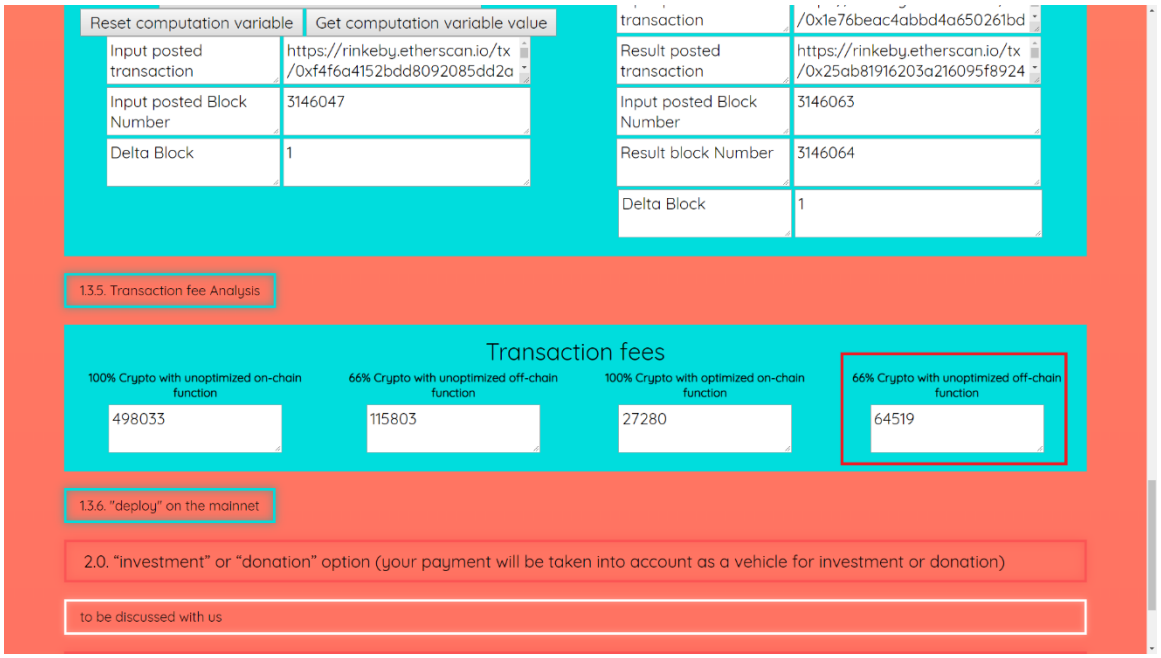


Fig 18. Case 4: Results and gas fee

FUTURE WORK

The interface with which transactions are posted to the blockchain from outside the blockchain has been implemented. The same interface would be used to build checkpoint and rollback algorithm. The interface developed will keep track of the variable changes (checkpoints) that are being made to one smart contract. These checkpoints will be later used to revert the state of the smart contract. The interface will also be linked to a voting smart contract present on the blockchain, through which the rollback process is voted. On successful voting, rollback is performed on the smart contract thereby reverting the variable states to the selected checkpoint. User voting need not be synced with the blockchain to make a vote, rather the user will be able to vote through the HTML page designed for the same purpose. The voting process will be timed, so that there would not be an indefinite wait time to decide on the rollback. The extended work will also be containing validation from the sources that were able to post the transactions into blockchain from outside the network.

CHAPTER IV

SLIM CHAIN

To utilize the full potential of a blockchain, one has to download an entire copy of the blockchain on one's local machine. The current size of both BitCoin and Ethereum have exceeded 300 GB and the size has been monotonically increasing over time. This current size of the blockchain would take up more than half the space of an average sized hard disk (512 GB). The proposed slim chain is a modification to the structure of the blockchain to address the issue on the size of the blockchain by creating an off-chain link to IPFS, to store a portion of information that is supposedly stored on blockchain. The model proposed for slim-chain consists of the states expressed from the standpoint of a transaction's life span in steady state. Note that the dependability in the following context is defined by the state of a transaction 100% reliable and secured as desired and defined, or undependable otherwise.

DEPENDABILITY MODEL

P_{on}: the state in which a transaction stays on-chain and dependable. Note that it is assumed that the status of a transaction on chain is dependable without loss of generality. However, it is assumed in this research that there exist potential risks of undependability on chain, as denoted by P'_{on} as follows, such as security breaches escaping the guard of the 51% rule or hardware/software failures, to mention a few, which is not impossible to hit in practice.

P'_{on}: the state in which a transaction stays on-chain but undependable. Note that this state is not a routine probabilistic complement of P_{on} and, instead, is a steady sate probability for a transaction to switch and stay its state into the undependable one in the course of its life span.

P_{off}: the state in which a transaction stays off-chain and dependable.

P'_{off}: the state in which a transaction stays off-chain and undependable. In practice, this state would be the one with the highest susceptibility to all sorts of potential off-chain readily high undependability risks such as all sorts of known web2-security breach risks without any sorts of security and dependability measure such as the 51% rule and hardware/software failures.

The random variables to express the state transition probabilities are as follows:

d_{on}, d_{off}: the dependability of a transaction on-chain and off-chain, respectively. Note that each of these variables are an instantaneous rate for a transaction to switch and stay its state into the dependable state either on- or off-chain, respectively (cf. P_{on} and P_{off}).

c, c': the crypto rate and un-crypto rate of a transaction, respectively. The crypto rate implies the rate at which a transaction stays on-chain, or un-crypto rate otherwise i.e., rate at which the transaction stays off-chain. As has been reported in [23], the crypto rate can be determined based on the size and type of computations by the transactions such that the larger the size is and the

more complex the computation is, the more substantially the cost saving is realized and ultimately significant cost (e.g., gas fee) saving would be achieved.

i, i': the IPFS rate and un-IPFS rate of a transaction, respectively. The IPFS rate denotes the rate a transaction is uploaded onto IPFS (inter-planetary file system) and at the same time (or as part of the transaction execution process) a hash code is created and linked back and posted into the blockchain in place of the original transaction that has been migrated off-chain and would have been in huge volume, otherwise.

r, r': the read rate on a transaction without modifying the transaction context and write rate (r') with modification on a transaction that is likely to cause an undependable consequence on it without loss of generality.

The state transition probabilities (i.e., $t(P_i, P_j)$: state transition probability from P_i to P_j) are defined as follows:

$t(P_{on}, P_{on}) = cd_{on}$, this state transition tracks and implies the sustainability of the dependability of a transaction on-chain such that if a transaction is currently dependable on-chain and it continues to stay on-chain (i.e., at the rate of c) and sustains its dependable state (i.e., at the rate of d_{on}), then it contributes to the steady state probability of P_{on} in a positive manner.

$t(P_{on}, P_{off}) = c'$, this state transition tracks and implies the rate to switch the execution location of a transaction from on- to off-chain with the dependability sustained such that if a transaction is currently dependable on-chain and it switches its execution location to off-chain (i.e., at the rate of c') and sustains its dependability (i.e., at the rate of 1.0, and note that the 1.0 implies that it is assumed that there is no dependability loss in the course of execution location switch from on- to off- chain), then it contributes to the steady state probability of P_{off} in a positive manner while to

P_{on} in a negative manner. However, note that, from the standpoint of the overall dependability, there is no loss of dependability of the transaction as far as this state transition is concerned.

$t(P_{on}, P'_{on}) = c' d_{on}$, this state transition tracks and implies the rate to switch the dependability status of a transaction from dependable to undependable such that if a transaction is currently dependable on-chain (i.e., at the rate of c) but it fails to sustain the dependability (i.e., at the rate of d'_{on}), then it contributes to the steady state probability of P'_{on} in a negative manner while to P_{on} in a negative manner.

$t(P'_{on}, P'_{on}) = c$, this state transition tracks and implies the undependability of a transaction on-chain such that if a transaction is currently undependable on-chain and it continues to stay on-chain (i.e., at the rate of c) and stays undependable (i.e., at the rate of 1.0, and note that the 1.0 implies that it is assumed that there is no change in the status of the undependability of the transaction as there is no reversibility in specific order to reverse the undependability issues as long as staying on-chain without loss of generality. By the way, also note that security breaches escaping the guard of the 51% rule or hardware/software failures are examples of malicious reversibility while the above mentioned reversibility is the one to remedy any undependability issues in a benignant manner.), then it contributes to the steady state probability of P'_{on} in a positive manner.

$t\left(P'_{on}, P'_{off}\right) = c'$, this state transition tracks and implies the rate to switch the execution location of a transaction from on- to off-chain with the undependability maintained such that if a transaction is currently undependable on-chain and it switches its execution location to off-chain (i.e., at the rate of c') and maintains its undependability (i.e., at the rate of 1.0, and note that the 1.0 implies that it is assumed that there is no remedy for dependability in the course of execution location switch from on- to off- chain), then it contributes to the steady state probability of P'_{off} in a positive manner while to P'_{on} in a negative manner. However, note that, from the standpoint

of the overall undependability, there is no change of undependability of the transaction as far as this state transition is concerned.

$t(P_{off}, P_{off}) = i'(r + r'd_{off})$, this state transition tracks and implies the sustainability of the dependability of a transaction off-chain such that if a transaction is currently dependable off-chain and it continues to stay off-chain (i.e., at the rate of i', that is, the rate not to be posted back on the chain, cf. i) and sustains its dependable state (i.e., at the rate of d_{off}) in case of write operations taken place (i.e., at the rate of r'), or only read operations have taken place (i.e., at the rate of r), then it contributes to the steady state probability of P_{off} in a positive manner. Note that the sum of all the outbound state transition probabilities from P_{off} is $i + i'\left(r + r'\left(d_{off} + d_{off}{}'\right)\right) = 1.0$, and this particular state transition probability is responsible for the portion of $i'(r + r'd_{off})$.

$t(P_{off}, P_{on}) = i$, this state transition tracks and implies the rate to post the hash address generated through IPFS of an off-chain transaction back on to the chain with the dependability sustained such that if a transaction is currently dependable off-chain and it is processed through IPFS and a hash address has been generated to be posted back on the chain (i.e., at the rate of i) and sustains its dependability (i.e., at the rate of 1.0, and note that the 1.0 implies that it is assumed that there is no dependability loss in the course of posting the hash address of the transaction from off- to on- chain), then it contributes to the steady state probability of P_{on} in a positive manner while to P_{off} in a negative manner. However, note that, from the standpoint of the overall dependability, there is no loss of dependability of the transaction as far as this state transition is concerned.

$t(P_{off}, P'_{off}) = i'r'd'_{off}$, this state transition tracks and implies the loss of the undependability of a transaction off-chain such that if a transaction is currently dependable off-chain and it

continues to stay off-chain (i.e., at the rate of i', that is, the rate not to be posted back on the chain, cf. i) but switches its dependable state to undependable state (i.e., at the rate of d'_{off}) only in case of write operations taken place (i.e., at the rate of r'), then it contributes to the steady state probability of P'_{off} in a positive manner while to P_{off} in a negative manner. Note that the sum of all the outbound state transition probabilities from $P_{off} is\ i + i'\left(r + r'\left(d_{off} + d'_{off}\right)\right) = 1.0$, and this particular state transition probability is responsible for the portion of $i'r'd'_{off}$.

$t\left(P'_{off}, P'_{on}\right) = i$, this state transition tracks and implies the undependability of a transaction off-chain such that if a transaction is currently undependable off-chain and it continues to stay off-chain (i.e., at the rate of i', that is, the rate not to be posted back on the chain, cf. i) and stays undependable (i.e., at the rate of d'_{off}) in case of write operations taken place (i.e., at the rate of r'), or only read operations have taken place (i.e., at the rate of r), then it contributes to the steady state probability of P'_{off} in a positive manner. Note that the sum of all the outbound state transition probabilities from $P'_{off} is\ i + i'\left(r + r'\left(d_{off} + d'_{off}\right)\right) = 1.0$, and this particular state transition probability is responsible for the portion of $i'\left(r'd'_{off} + r\right)$.

$t\left(P'_{off}, P_{off}\right) = i'r'd_{off}$, this state transition tracks and implies the rate to post the hash address generated through IPFS of an off-chain transaction back on to the chain with the undependability maintained such that if a transaction is currently undependable off-chain and it is processed through IPFS and a hash address has been generated to be posted back on the chain (i.e., at the rate of i) and stays undependable (i.e., at the rate of 1.0, and note that the 1.0 implies that it is assumed that there has been no remedy taken place to fix the undependability in the course of posting the hash address of the transaction from off- to on- chain), then it contributes to the steady state probability of P'_{on} in a positive manner while to P'_{off} in a negative manner. However, note that, from the standpoint of the overall undependability, there is no change of undependability of

the transaction as far as this state transition is concerned. Given that there are four states in the proposed state model , the five equations are highlighted in Equations 1 - and each of the equations shows the steady state probabilities of that particular state.

$$P_{on}(cd_{on} + cd'_{on} + c') = P_{off} i \tag{1}$$

$$P'_{on}(c + c') = P_{on} cd_{on} + P'_{off} i \tag{2}$$

$$P_{off}\left(i'\left(r + r' d_{off}\right) + i' r' d'_{off} + i\right) = P_{on} c' + P'_{off} i' r' d_{off} \tag{3}$$

$$P'_{off}\left(i' r' d'_{off} + i' r + i' r' d_{off} + i\right) = P'_{on} c' + P_{off} i' r' d'_{off} \tag{4}$$

$$P_{on} + P'_{on} + P_{off} + P'_{off} = 1 \tag{5}$$

ANALYSIS

The detailed solutions of the steady state equations are shown in Appendix B. The final solutions for the above steady state equations for the slim model are shown in Equations 6 - 11.

$$P_{on} = \frac{1}{Q_2} \tag{6}$$

$$P_{off} = \frac{cd_{on} + cd'_{on} + C'}{iQ_2} \tag{7}$$

$$P'_{on} = \frac{1}{Q_2}\left(cd_{on} + \frac{1}{Q_1}\left(cd_{on} c' + \frac{cd_{on} + cd'_{on} + c'}{i}\right) i\right) \tag{8}$$

$$P_{off'} = \frac{1}{Q_2}\frac{1}{Q_1}\left(cd_{on} c' + \frac{cd_{on} + cd'_{on} + C'}{i}\right) \tag{9}$$

$$Q_1 = i' r' d'_{off} + i' r + i' r' d_{off} + i - ic' \tag{10}$$

$$Q_2 = 1 + cd_{on} + \frac{icd_{on} c' + cd_{on} c'}{Q_1} + \left(\frac{cd_{on} + cd'_{on} + c'}{i}\right)\left(\frac{2i' r' d'_{off}}{Q_1} + 1\right) \tag{11}$$

Based on the solutions of the proposed slim chain model, the following simulation are conducted in order to demonstrate the expected dependability (i.e., P_{on}) versus various yet primary variables such as d_{on}, c and i; and another expected dependability (i.e., P_{off}) versus the variables such as

d_{off}, c and i. Note that, in fact, the overall dependability of interest is the sum of P_{on} and P_{off} in the proposed slim model.

The graphs on P_{on} versus d_{on} shown in Figure 1 are plotted in order to demonstrate the effect of d_{on} of a transaction on the steady state probability for the transaction to be in state P_{on} at high or medium (e.g., .9 or .5, respectively) of d_{off}, i and at low or medium (e.g., .1 or .5, respectively) of c, where the plots at the top (i.e., d_{off}, $i = .9$ and $c = .1$) and the 2nd from the top (i.e., d_{off}, $i = .9$ and $c = .5$) reveal that at high d_{off} and i, i.e., with the off-chain portion of the transaction life span is highly dependable along with highly dependable IPFS processes in place, the effect of d_{on} on P_{on} dilutes nearly-linearly logarithmically as d_{on} rises and c drops, and it is also observed that the gap between the top and the 2nd top P_{on} is widening as d_{on} rises, and also where the plots at the bottom (i.e., d_{off}, $i = .5$ and $c = .1$) and the 2nd from the bottom (i.e., d_{off}, $i = .5$ and $c = .5$) reveal that at medium d_{off} and i, i.e., with the off-chain portion of the transaction life span is moderately dependable along with also moderately dependable IPFS processes in place, the effect of d_{on} on P_{on} dilutes nearly-linearly logarithmically as d_{on} rises and c rises as well, and the gap between the bottom and the 2nd bottom P_{on} is widening as d_{on} rises at a medium;

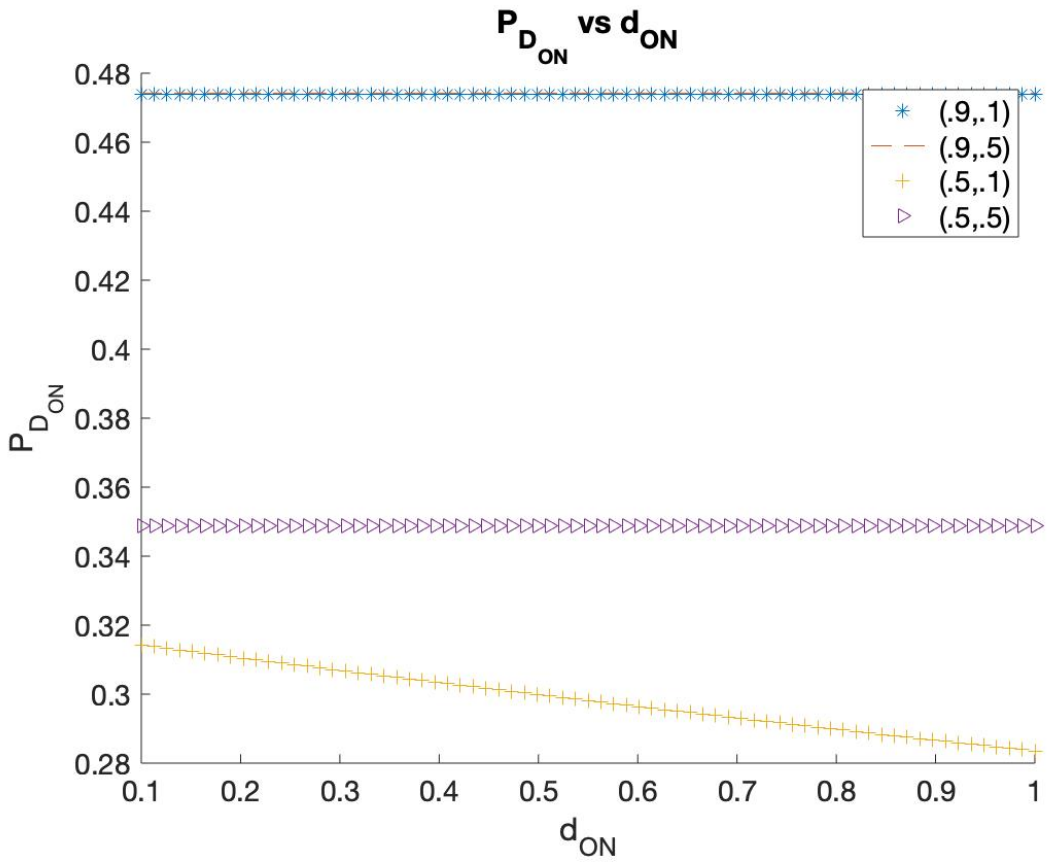


Figure 1: P_{on} versus d_{on}

P_{on} versus i at high or medium (e.g., .9 or .5, respectively) of d_{on}, d_{off} and at low or medium (e.g., .1 or .5, respectively) of c, where the plots at the top (i.e., d_{on}, $d_{off} = .9$ and $c = .1$) and the 2nd from the top (i.e., d_{on}, $d_{off} = .9$ and $c = .5$) reveal in Figure 2 that at high d_{on} and d_{off}, i.e., with both the on- and off-chain portions of the transaction life span are highly dependable along with yet low or moderate crypto rate c, the effect of i on P_{on} plots a positive concavity as d_{on} rises and as c drops, the concavity point is being delayed, and it is also observed that the gap of P_{on} between the top and the 2nd top is widening as i rises towards the concavity point then narrows back till the top curve and the 2nd top curve intersect, and also where the plots at the bottom (i.e., d_{on}, $d_{off} = .5$ and $c = .5$) and the 2nd from the bottom (i.e., d_{on}, $d_{off} = .5$ and $c = .1$) reveals that at medium d_{on} and d_{off}, i.e., with both on- and off-chain portion of the

transaction life span is moderately dependable, the effect of i on P_{on} also plots a positive concavity as d_{on} rises and as c drops, the concavity point is being delayed, and it is observed that the gap of P_{on} between the bottom and the 2nd bottom is widening yet much narrower than the case where at high d_{on} and d_{off}, as i rises towards the concavity point then narrows back till the bottom curve and the 2nd bottom curve intersect;

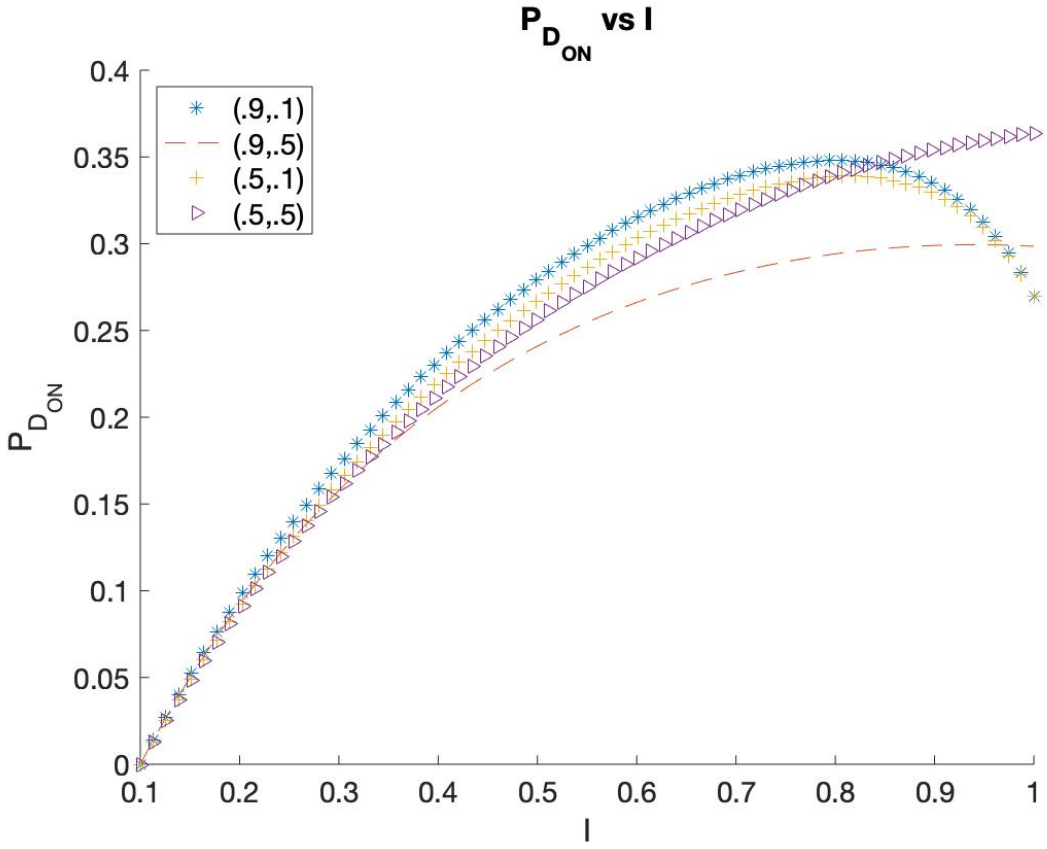


Figure 2: P_{on} versus i

Figure 3 P_{on} versus c at high or medium (e.g., .9 or .5, respectively) of d_{on}, d_{off} and i (notice that there are only two graphs plotted as c itself runs on the x-axis), where the plots at the top (i.e., d_{on}, d_{off},$i = .9$) and the bottom (i.e., d_{on}, d_{off},$i = .5$) reveal that at high d_{on}, d_{off} and i, i.e., with both the on- and off-chain portions of the transaction life span are highly dependable along with also highly dependable IPFS processes in place, the effect of c on P_{on} plots a negative

concavity as c rises and as d_{on}, d_{off} and i drop to .5, the concavity point is being delayed, and it is also observed that the gap of P_{on} between the top and the bottom is initially narrowing as c rises towards the concavity point then widens back through.

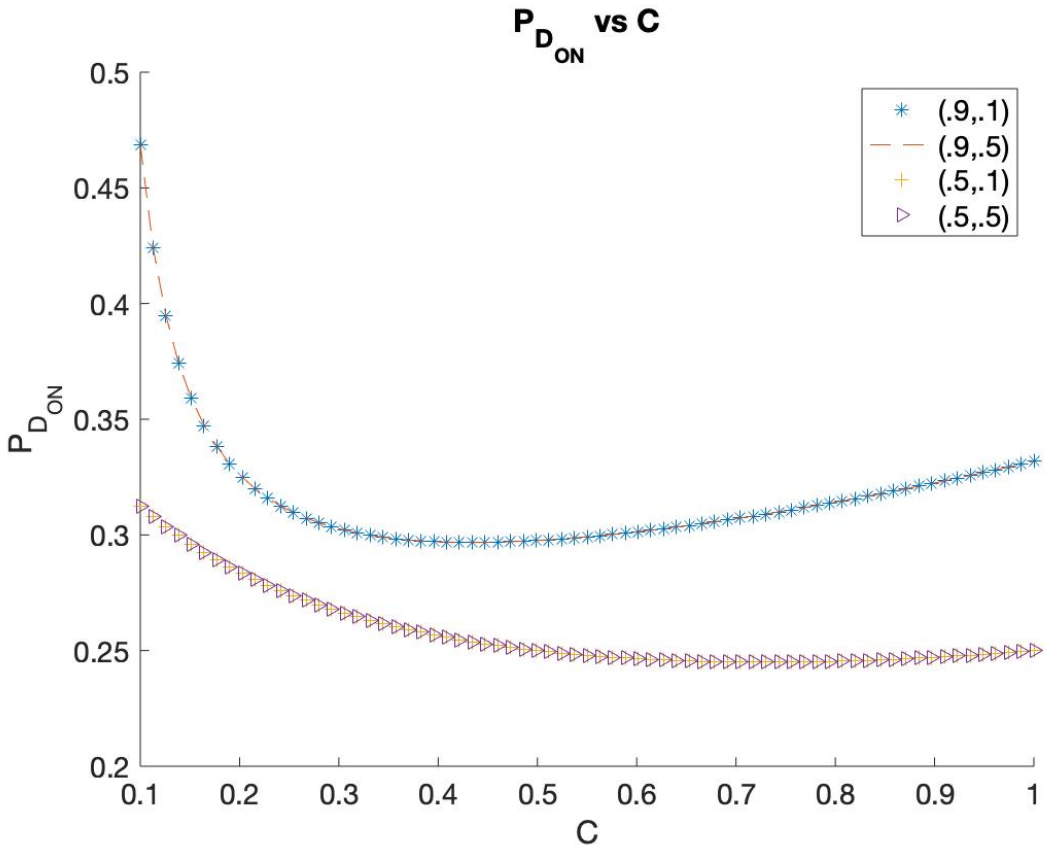


Figure 3: $P_{D_{on}}$ versus c

The graphs on P_{off} versus d_{off} demonstrated in Figure 4 are plotted in order to demonstrate the effect of d_{off} of a transaction on the steady state probability for the transaction to be in state P_{off} at high or medium (e.g., .9 or .5, respectively) of d_{off}, i and at low or medium (e.g., .1 or .5, respectively) of c, where the plots at the top (i.e., d_{on}, $i = .9$ and $c = .1$) and the 2nd from the top (i.e., d_{on}, $i = .9$ and $c = .5$) reveal that at high d_{on} and i, i.e., with the on-chain portion of the transaction life span is highly dependable along with highly dependable IPFS processes in place, the effect of d_{off} on P_{off} increases nearly-linearly and flat as d_{off} rises and c drops, and

it is also observed that the gap between the top and the 2nd top P_{off} is widening as d_{off} rises, and also where the plots at the bottom (i.e., d_{on}, $i = .5$ and $c = .1$) and the 2nd from the bottom (i.e., d_{on}, $i = .5$ and $c = .5$) reveal that at medium d_{on} and i, i.e., with the on-chain portion of the transaction life span is moderately dependable along with also moderately dependable IPFS processes in place, the effect of d_{off} on P_{off} slightly increases nearly-linearly and flat as d_{off} rises and c rises as well, and the gap between the bottom and the 2nd bottom P_{off} is maintained nearly consistent as d_{off} rises;

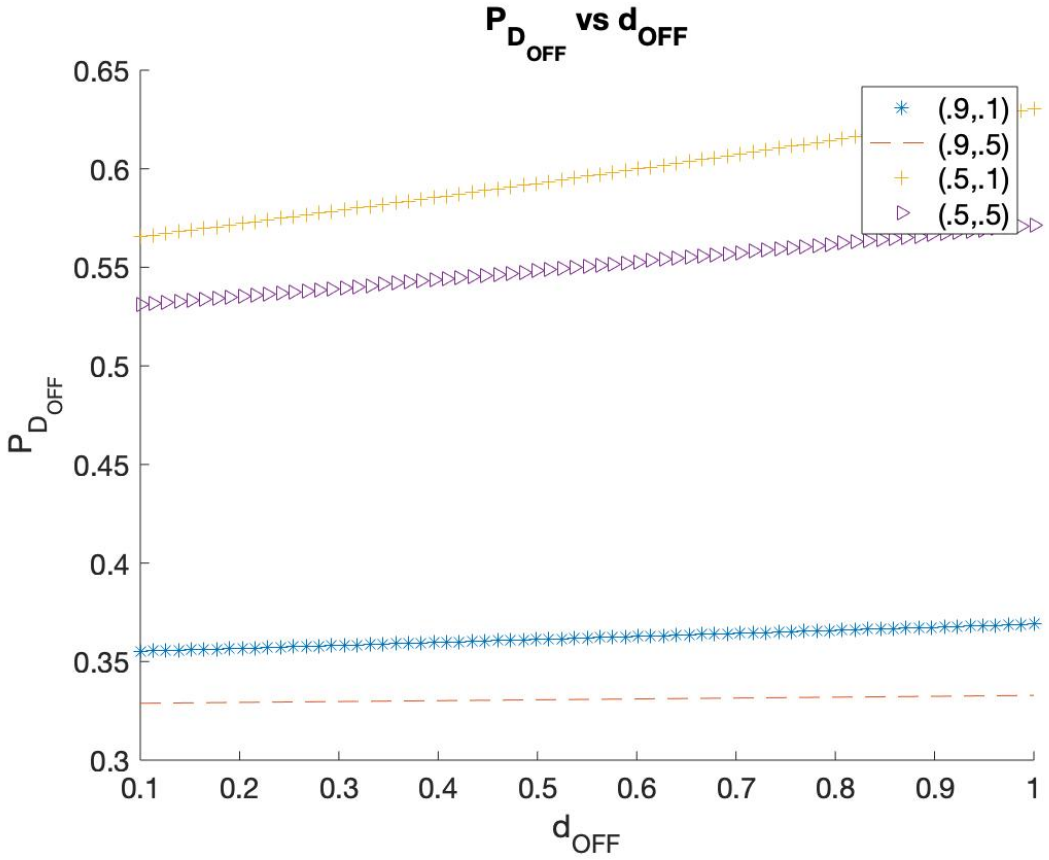


Figure 4: P_{off} versus d_{off}

P_{off} versus i (Figure 5) at high or medium (e.g., .9 or .5, respectively) of d_{on}, d_{off} and at low or medium (e.g., .1 or .5, respectively) of c, where the plots at the top (i.e., d_{on}, $d_{off} = .9$ and $c = .1$) and the 2nd from the top (i.e., d_{on}, $d_{off} = .9$ and $c = .5$), and at the bottom (i.e., d_{on},

$d_{off} = .5$ and $c = .5$) and the 2nd from the bottom (i.e., d_{on}, $d_{off} = .5$ and $c = .1$) reveal that at high d_{on} and d_{off}, i.e., with both the on- and off-chain portions of the transaction life span are highly dependable along with yet low or moderate crypto rate c, the effect of i on P_{off} plots decreasing trends in an intermingled manner regardless of various combinations of d_{on}, d_{off} and c as d_{off} rises, and it is observed that c does not influence them in a significant manner, curve intersect;

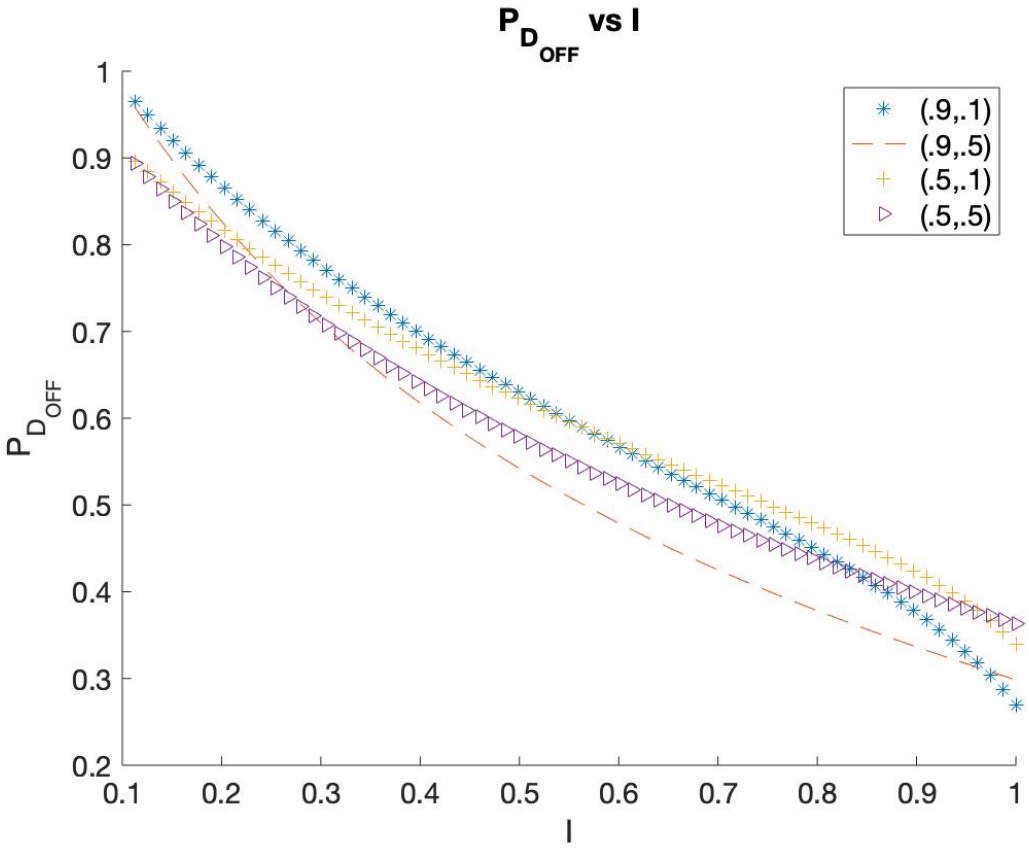


Figure 5: P_{off} versus i

and P_{off} versus c at high or medium (e.g., .9 or .5, respectively) of d_{on}, d_{off} and i (Figure 6) (notice that there are only two graphs plotted as c itself runs on the x-axis), where the plots at the top (i.e., d_{on}, d_{off}, $i = .9$) and the bottom (i.e., d_{on}, d_{off},$i = .5$) reveal that at high d_{on}, d_{off} and i, i.e., with both the on- and off-chain portions of the transaction life span are highly

dependable along with also highly dependable IPFS processes in place (i.e., d_{on}, d_{off}, $i = .9$) and with both the on- and off-chain portions of the transaction life span are moderately dependable along with also moderately dependable IPFS processes in place (i.e., d_{on}, d_{off},$i = .5$), respectively, the effect of c on P_{off} plots a slightly negative concavity as c rises and as d_{on}, d_{off}, and i picks up to .9, the concavity point is being delayed, and it is also observed that the gap of P_{off} between the top and the bottom is initially widening as c rises towards the concavity point then maintains consistently through.

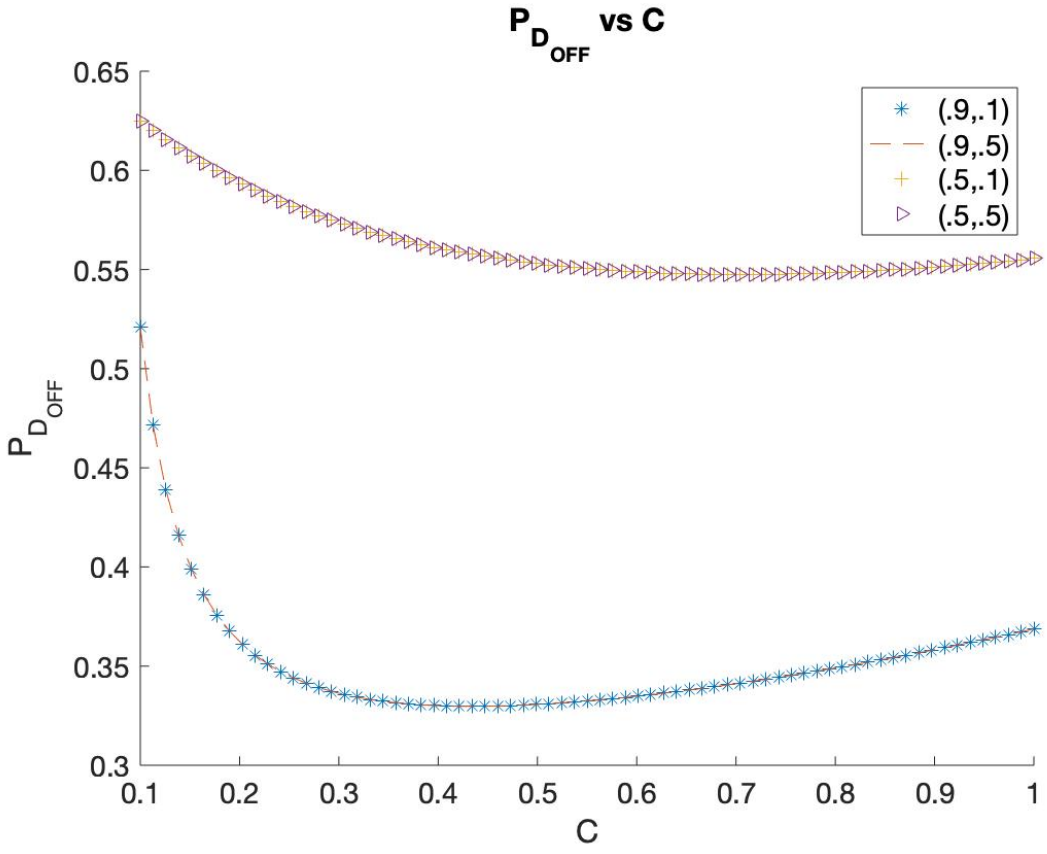


Figure 6: P_{off} versus c

Further, it is to be noted in this research that in order to build and demonstrate a slim chain, the open source of Ethereum is used to build upon and tested in an isolated manner. Most of the modifications are centered around the part where the transaction gets mined successfully and is

ready to be added to the blockchain. Modifications are made to decide whether a transaction is actually being added to the blockchain or the transaction is being added to the IPFS and only the return hash code from IPFS is added to the block. Upon doing so, the size of the blockchain is considerably reduced depending on the ratio of transactions sent to IPFS to those posted directly on the blockchain. The addition of a transaction in the slim chain can be described in a flow chart as shown below.

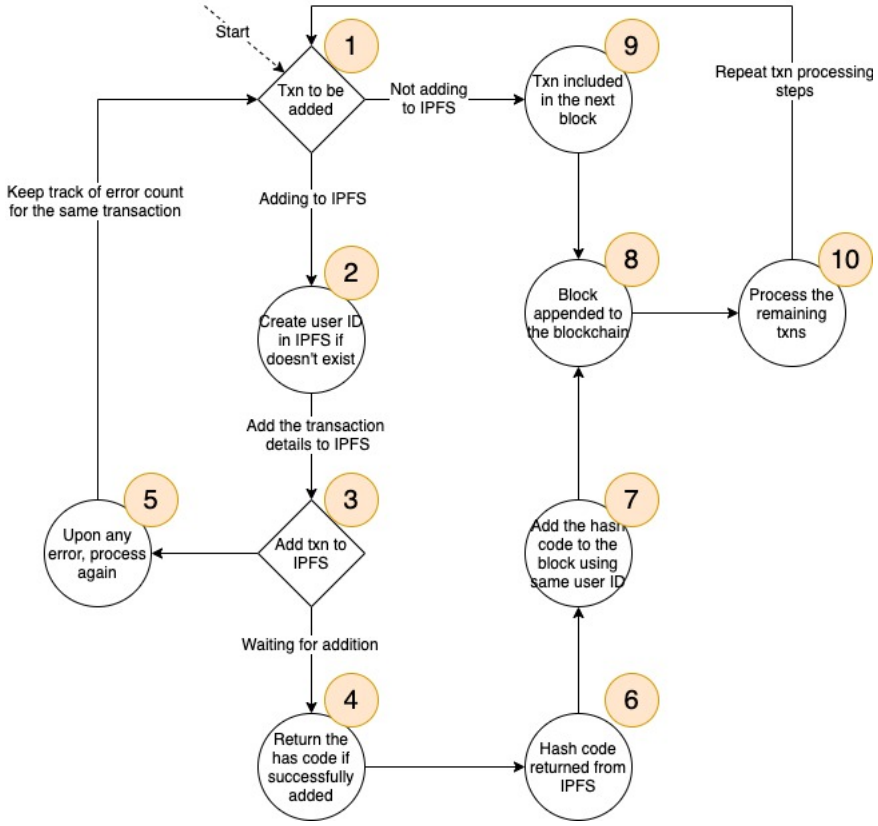


Figure 7. Flowchart for slim chain

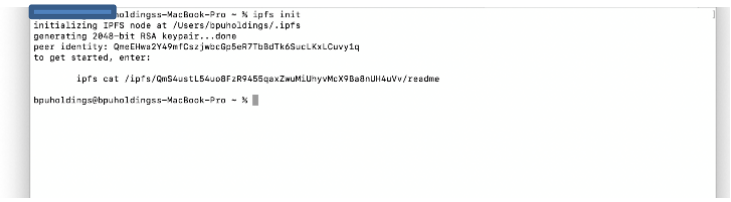
IMPLEMETATION

In the following, it is demonstrated that the IPFS public address can be seen in the terminal window.

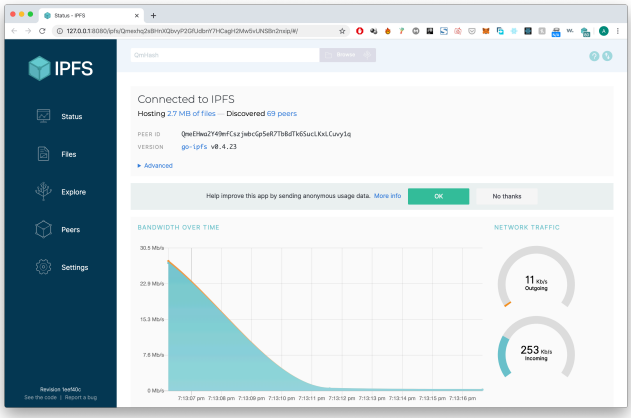
1. The transaction that is mined and to be added to the blockchain.

2. Addition to IPFS: A user ID will be created in IPFS using the same user ID that was used to post the transaction on to the blockchain. This user ID will be used later on to track the transactions uploaded to IPFS. Thereby, the user ID are in a way hash-mapped to one another.

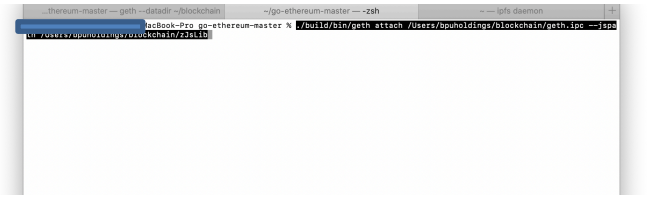
   In the following, it is demonstrated that the IPFS public address can be seen in the terminal window.



IPFS has started running using the command ‘IPFS daemon’. A visual representation of IPFS distributed storage system for this public address when viewed through the browser looks as follows:



3. Add transaction to IPFS: This step takes care of the addition of the transaction to the IPFS itself. A hash code is returned upon successful addition to the IPFS. This hash code will be used to address the transaction if necessary, in the future.
4. Hash code is returned back from the IPFS to the blockchain upon successful entry into IPFS. Using another console which is attached to the geth console, a contract is posted to test this particular scenario where the transaction also gets posted to IPFS. The screen of the console is shown as below:

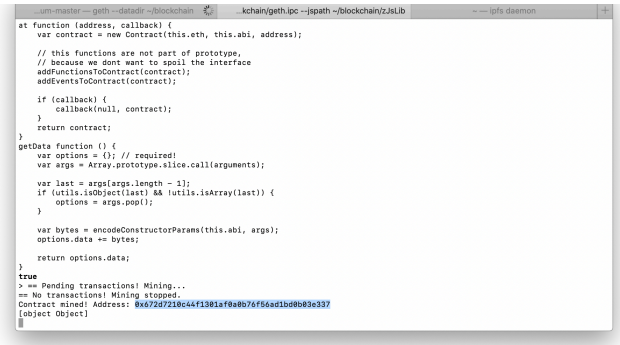


Few steps, which include creating a new private public key and setting the same public key as the etherbase account for this particular geth and also to start mining process which further increase the balance on the account. All these steps are taken to initialize the account with some ether in the private blockchain which can be seen as below:

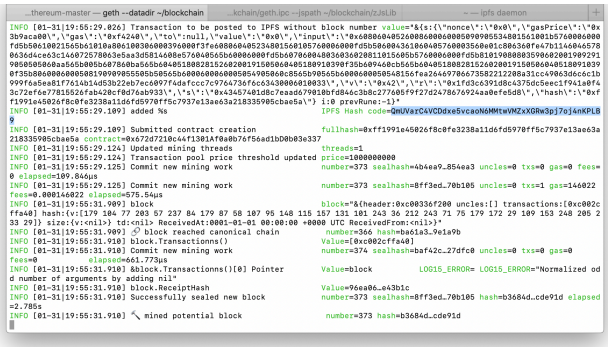
5. If any error persists while adding the transaction to IPFS, the necessary steps are repeated for addition. The error count will be kept track of to ensure that the delay of transaction being posted is minimized. Upon reaching a pre-defined error limit the transaction will be

added to the blockchain itself (this is only to reduce the time delay for a single transaction).

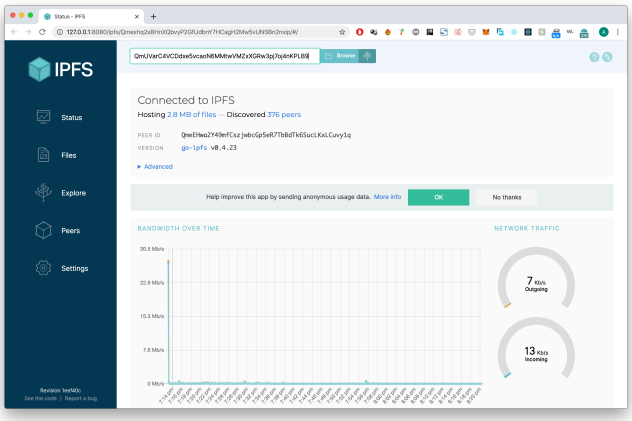
A simple contract was posted in Ethereum using the same console and the contract address is visible on the following screenshot:



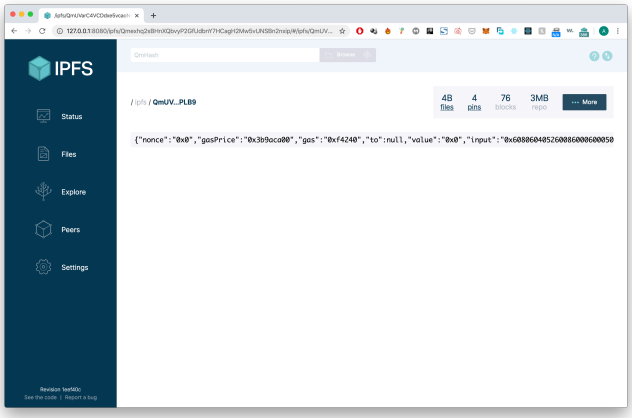
6. The hash code is received at the slim chain end from IPFS. The same contract block is posted to IPFS and the hash code is returned from IPFS is stored in the blockchain. The IPFS hash code returned from IPFS to geth can be viewed here:



7. Hash code is stored in the blockchain, rather than the entire transaction itself. The IPFS hash code accessed through the web browser is as below:



All the transaction details for the earlier contract are posted to IPFS block as an entire JSON file:



The transaction is added into the block which is further appended to the blockchain.

The above was an implementation of the proposed slim-chain model using Ethereum and IPFS as the two distributed ledgers to distribute the data on each other. The percentage of size reduction is dependent on how much of the initial blockchain data is put on IPFS. The more the data transferred to IPFS, the higher the percentage save in the size of the blockchain.

CHAPTER V

REALTIME CHAIN

All the transactions that are posted towards the blockchain enter into a storage area called transaction pool and the transactions that are not yet posted to the blockchain are pending transactions. Pending transactions are the transactions that are not posted to the blockchain yet submitted by the end-user, and they are present in the transaction pool [29]. In order to reduce the time taken for a transaction to be posted to the blockchain, the following potential issues have to be addressed and resolved:

- Mining process: such that as the difficulty is increased, the mining gets harder thereby taking much more time. The difficulty is a parametric variable that is controlled by the users of the blockchain.
- Variables set in the blockchain: such as the minimum number of transactions to be present in one block or minimum time to wait for the next block to be added to the blockchain.

A novel approach to the real-time chain proposed in this work is as follows. The pending transactions in the pool are sampled and sent to the miners. The algorithm of sampling is determined depending on the characteristics of the blockchain of concern. The three algorithms below are to be investigated for transaction sampling in this work, in order to prioritize the transactions in the pool for inclusion for posting in the block.

- Random sampling: A set of transactions are sampled randomly from the transaction pool and then processed. In other words, the prioritization is conducted randomly.
- Sorted sampling: the transactions are sampled based on certain characteristics of the blockchain and then sent to the miners. For example, the transactions can be sorted based on the transaction fee or gas fee, to mention a couple.
- Stratified Sampling: The transactions are stratified based on the characteristics of transactions and then a different number of transactions are sampled from each stratum. Statistically, the confidence level of the samples is expected to be improved than the random sampling, and further the stratified sampling is employed in this work with an expectation of improved block-dependability.

The variables to be taken into account to model the block-dependability in the real-time chain are as follows.

- Current # of nodes in the blockchain – n
- Current # active (i.e., participating) miners – m
- Gas fee of the transaction – g_t (integer, gas used determined after a transaction getting successfully mined)
- Actual fee of the transaction – a_t (integer, also determined after the transaction got mined to the blockchain)

- "From address" of the transaction – public key, which is a 32 bit hash address supposedly unique in the entire blockchain. This address is the public key of the user who is initiating the transaction in the blockchain.
- "To address" of the transaction – 32 bit hash address supposedly unique in the entire blockchain. This address is the address to which the transaction is being sent to. The address could be a public key of a user or the public key of a smart contract.

The variables to represent the time taken for transactions to be posted to blockchain are as follows:

- Normal – t
- Random – t_r
- Sorted – t_{sort}
- Stratified Sampling – t_s

Given the same set of inputs for all the above four different algorithms of sampling the pending transactions for mining, it is expected without loss of intuition (shown in Equation 12) that either the sorted or sampled algorithms be faster than the normal algorithm.

$$t_r, t_{sort}, t_s < t \tag{12}$$

A transaction is broadcasted by a wallet application, and then waits to be sampled up by a miner on the blockchain. Unless it is sampled up, the transaction stays present in a pool of unconfirmed transactions. This pool is a collection of unconfirmed transactions on the network that are waiting to be processed. These unconfirmed transactions are usually not collected in one giant pool, but more often in small subdivided local pools, namely, the Mempool. Every node in the blockchain has its own Mempool which will consist of all the pending transactions.

In general, unless otherwise coordinated, the way how it is decided as to which transaction to be the one that gets mined by a miner is such that miners usually sample the transaction with the highest fees. However, if necessary and justifiable from a particular performance perspective, such as the deadline as is proposed in this work, instead of sampling the transaction with the highest fees, miners can sample (or select) transactions from the pool by using the following algorithms, in particular order to facilitate and coordinate the proposed real-time chain, possibly in a direct or indirect manner.

- Normal – The current way in Ethereum as to how the transactions are pulled from the pool. The highest transaction fee will be sampled first with the highest priority.
- Random – A random transaction from the transaction pool will be sampled to be in the mining process thus making the possibility of any transaction being picked from the transaction pool equal to another.
- Sorted – All the transactions are sorted in the transaction pool in a way as demanded by a particular prioritization criterion. The criteria are determined by using one the variables determined earlier.
- Stratified – Using stratified sampling algorithm, the transactions will be sampled one per stratum. Transactions in the pool will be logically stratified using one of the characteristics of the transactions like gas, transaction fee, the amount of ether involved, to mention a few.

Let the total number of transactions in the pool be represented by n, and then each i^{th} transaction by t_i, $1 \leq i \leq n$. In the normal method of sampling transactions out of the pool, the highest fee transactions are given the priority. Equation 13 determines the probability of a transaction to be sampled for mining, which is dependent on the amount of fees that particular transaction holds.

$$p_{pick_i} \propto g_i \tag{13}$$

Where, g_i is the gas amount of the i^{th} transaction. The higher the gas fee included in the transaction the higher the probability of the transaction getting sampled. Suppose that the total sum of the fees in the transaction pool = S, the summation equation is described in Equation 14.

$$S = \sum_{i}^{n} g_i \tag{14}$$

Equation 15 determines the proportionality of the probability for the i^{th} transaction to be sampled.

$$p_{pick_i} \propto h\left(\frac{g_i}{S}\right) \tag{15}$$

Where, $h(\,)$ represents a function of g_i and $\frac{1}{S}$. In the proposed random algorithm of sampling transactions from the pool, every transaction has an equal chance of being sampled for mining and then posted into the blockchain. Equation 16 represents the probability for a transaction to be sampled in a random manner is proportional to a certain constant c.

$$p_{pick} \propto c \tag{16}$$

Note that the value of c is dependent on the number of transactions present in the transaction pool. In fact, this constant can be controlled as a variable as well depending on whether the sampling algorithm is adaptive or not. The block dependability d_{gi} is directly proportional to the p_{pick} as shown in Equation 17.

$$d_{gi} \propto p_{select} \tag{17}$$

In order to take the number of transactions into the block dependability, the following can be drawn. The higher the number of transactions in the pool, the lesser the block dependability of any transaction sampled turns. Equations 18, 19 and 20 describe such proportionalities associated with block dependability.

$$d_{gi} \propto \frac{1}{n} \tag{18}$$

The faster the transactions arrive at the mining pool, the lesser the block dependability turns as expressed in Equation 19.

$$d_{gi} \propto \frac{1}{s} \tag{19}$$

$$d_{gi} \propto m \tag{20}$$

Where, s is the average speed of the arriving transactions into the mining pool and m is the number of active miners connected to the blockchain at the particular instant of time. The higher the deadline, the higher the number of transactions to be sampled for mining process turns up, and the proportionality is expressed in Equation 21.

$$d_{gi} \propto t_d \tag{21}$$

Where, t_d is the temporal extent of the deadline for the transaction to be posted. Combining eq. 9 and 10 into a single proportionality equation, we have Equations 22 and 23.

$$d_{gi} = c_1 * \frac{1}{n} * c_2 * \frac{1}{s} * m * t_d \tag{22}$$

$$d_{gi} = c_r * \frac{m * t_d}{n * s} \tag{23}$$

where c_1 and c_2 are proportionality constants. In the proposed sorted algorithm of sampling transactions, the transactions are given priorities based on the characteristics of the transaction. If the sorting algorithm is used with respect to gas fee, the algorithm would reflect the same algorithm as normal one.

In the proposed stratified algorithm of sampling transactions from the pool, the transactions are sorted into different strata based on a pre-defined characteristic. Let S_i denote a stratum formed. Note that stratification process may be performed by considering whether the transaction is a

contract transaction or a transaction involving only transfer of ether if necessary, yet, which is not considered in this work.

Gas fees of the transactions can also be used as a characteristic to perform stratification of the transaction pool. The stratification is proposed through sampling strata based on the mean value of the gas fees present in the mining pool. The transactions are grouped together based on how far the current transaction is from the mean. With the mean of all the transactions in the mining pool being μ_g, the strata can be formed by using a constant $c * \sigma_g$. The strata formed from the transactions are grouped in a way such that one stratum has transactions all having gas fees falling in range of $\mu_g + c * \sigma_g$.

Assume that the total number of strata created are s and the total number of transactions n, then Equation 24 holds.

$$n = \sum_{i=1}^{n} S_i \tag{24}$$

Where, each stratum S_i consists of a group of transactions from the transaction pool homogenous in characteristic. The probability of sampling one stratum from all the strata at random is given in eq. 25.

$$p_{pick} \propto \frac{1}{s} \tag{25}$$

Where, s is the total number of strata in the transaction pool. Once stratification is performed, the transactions are sampled randomly from within each stratum in proportion to the size of each stratum. On another note, if the stratum sampled is based on a characteristic such as total gas fees of the stratum, then the probability of s_1 getting sampled will be the highest.

In this work, under a stringent deadline constraint as is to be addressed and enforced in the real-time blockchains, it is demonstrated how to identify the maximum number of transactions that can be posted into the target block as an attempt to realize a dependable real-time blockchain.

In the stratified sampling algorithm, the block dependability is defined for a stratum rather than for an individual transaction. The block dependability d_{si} of a particular stratum is given in Equation. 26.

$$d_{si} \propto g_{si} \tag{26}$$

Where, g_{si} is the total gas fees of i_{th} transaction.

$$d_{si} \propto m * t_d \tag{27}$$

In case of random transaction sampling, the transactions are sampled randomly from the pool, where the probability for a transaction to be sampled is equally distributed for any transaction as shown in eq. 28.

$$p_r = \frac{1}{n} \tag{28}$$

The probability for a transaction to be sampled for inclusion in the target block for posting is referred to as block-dependability in this work. As the probability of getting sampled is in equal amounts at the initial stage, the block-dependability for a deadline requirement is equal for all transactions. In other words, from the n transactions present in the pool and then, i transactions are sampled at random to be processed by miners (i.e., $p\left(\frac{i}{n}\right)$ the probability function $p(\,)$ of i and $\frac{1}{n}$). The number i is directly proportional to the deadline, i.e., a time constraint t, set beforehand. If t_{avg} is the average amount of time taken for a transaction to be processed by one miner and there are m such miners, in eq. 29 shows the proportionality.

$$p\left(\frac{i}{n}\right) \propto g\left(t, \frac{1}{t_{avg}}, m\right) \tag{29}$$

where $g(\,)$ represents a function of t, $\frac{1}{t_{avg}}$,m. In case of prioritized (or sorted) transaction sampling, let the speed at which the transactions get processed into the blockchain be s. In other words, s represents the number of bytes of transactions that can be processed in a unit time. Let the average time taken for the transactions to get into the blockchain be t_{avg} and t represents a deadline, then s can be expressed in eq. 30.

$$s = \frac{n\ (bytes)}{t\ (secs)} \tag{30}$$

Given a time constraint, i.e., deadline, the maximum number of bytes of transactions that can be processed for posting into the blockchain need to be identified in an attempt to realize a dependable real-time blockchain. Notice that once the number of bytes is determined, the transactions that are present in the transaction pool can be prioritized based on characteristics of transactions. Given a deadline time t, the total number of transactions that can be estimated to be completed within t, are the transactions that have the maximum size of s bytes/sec as $n = st$, where n is the number of bytes that can be processed within the given time t. From the transaction pool, the transactions are sampled in such a way that the total size all the sampled transactions doesn't exceed n bytes. Note that the transactions are then executed in a sequential manner by the miners.

Let i be the number of transactions sampled from the pool that satisfy the size constraint for deadline, and they will be the ones that are to be posted to the blockchain with higher likelihood compared to the rest of the pending transactions in the pool. Then, the miners are allotted the transactions sequentially. Given the current number of active and participating miners, one transaction will most likely be allotted to more than one miner. As soon as one of the miners

mines the transaction and post it to the blockchain, the same set of miners are given the next pending transaction from the prioritized items. If there are no more prioritized items, all the current pending transactions are reevaluated to determine the new i number of transactions. The size of each transaction is given by s_n, where $1 \leq n \leq i$. Equation 31 describes the summation property.

$$s_1 + s_2 + \cdots + s_i < s \tag{31}$$

The gas fees associated with t_i is g_i. As in the stratified approach, the gas fees for the transactions are monotonically increasing as shown in eq. 32.

$$g_1 \leq g_2 \leq \cdots \leq g_i \tag{32}$$

Depending on the number of transactions in the pool, miners will be given the highest critical transaction in the current prioritized transactions. Considering the gas fee for the transaction as a random variable x, the probability distribution of the transaction fees with an assumption of normal distribution is described in eq. 33.

$$P(x) = n\frac{1}{\sigma_g \cdot \sqrt{2\,\pi}}\, e^{\frac{-(x-\mu_g)^2}{2\sigma_g^2}} \tag{33}$$

σ_g represents the standard deviation of the transactions with respect to the gas fee, and μ_g represents the mean of all the current pending transactions in the transaction pool. When sampling the transactions based on a sorting by gas fees, particularly, the higher the gas fee is offered the higher is the probability for a transaction to be sampled by the miners. Also, if the transactions are sampled based on a sorted list by the miners that are currently active, then, without loss of generality and practicality, the maximum number of miners that can be working on one transaction is determined based on the sorting result.

Whether a transaction with a block-dependability $\left(d_{g_i}\right)$, i.e., a transaction with gas g_i, is prioritized or not, can be determined by how far the gas fee is from the highest gas fee offered in

the pending transaction pool. Hence, the block-dependability d_{gi} determined by proportionality are shown in eq. 34 and 35.

$$d_{g_i} \propto \frac{g - \sum_{n=1}^{i-1} g_n}{g} \tag{34}$$

$$d_{g_i} \propto 1 - \frac{\sum_{n=1}^{i-1} g_n}{g} \tag{35}$$

Where, g is the total gas of all the prioritized transactions and $\sum_{n=1}^{i-1} g_n$ represents the total gas of the transactions lower than the g_i. Equations 36 and 37 show that the block-dependability is also proportional to the time constraint set beforehand, i.e., the deadline, and also directly proportional to the number of active miners in the blockchain which is a variable unlike the typical deadline constraint which is constant for a set period of time.

$$d_{g_i} \propto t_d \tag{36}$$

$$d_{g_i} \propto m \tag{37}$$

Where, t_d is the deadline set and m is the number of active miners at an instant of time. Equation 38 shows that block dependability is in turn inversely proportional to the speed of the arrival of new transactions as follows.

$$d_{g_i} \propto \frac{1}{s} \tag{38}$$

where s is the average arrival speed of the transactions in the mining pool. Then, the block-dependability of a transaction with g is drawn in eq. 39 and 40.

$$d_{g_i} \propto t_d \cdot \left(1 - \frac{\sum_{n=1}^{i-1} g_n}{g}\right) \cdot \frac{m}{s} \tag{39}$$

$$d_{g_i} = c_2 \cdot t_d \cdot \left(1 - \frac{c_1 * \sum_{n=1}^{i-1} g_n}{g}\right) \cdot \frac{m}{s} \tag{40}$$

where c_1 and c_2 are the constants to fit the block-dependability.

PARAMETRIC SIMULATION AND RESULTS

Extensive parametric simulations are conducted, and the results are discussed in this section. The aforementioned random sampling algorithm of the transactions is simulated as the baseline; and the normal sampling algorithm is simulated as one of the sorted sampling algorithms in which the transactions are supposed to be sorted for prioritization by their gas fees offered in the transactions, and miners are supposed to sample the transactions from the ones with the highest gas fee offered. In the followings, the random sampling algorithm and the sorted sampling algorithms are simulated with respect to various performance parameters such as the number of transactions, the average speed of the transactions, the deadlines and the number of miners. The followings are the results from the random sampling algorithm of transactions from the mining pool and the observations made. Equation 40 is used to plot the below mentioned graphs.

Block dependability vs. Number of transactions: The higher the number of transactions in the mining pool, the lower the block dependability of the transactions being considered to be sampled. Figure 1 shows the graph of block dependability vs. the number of transactions. The graph shows 4 different cases, each of which with a different deadline selected. The higher the deadline the higher the block dependability of the transactions.

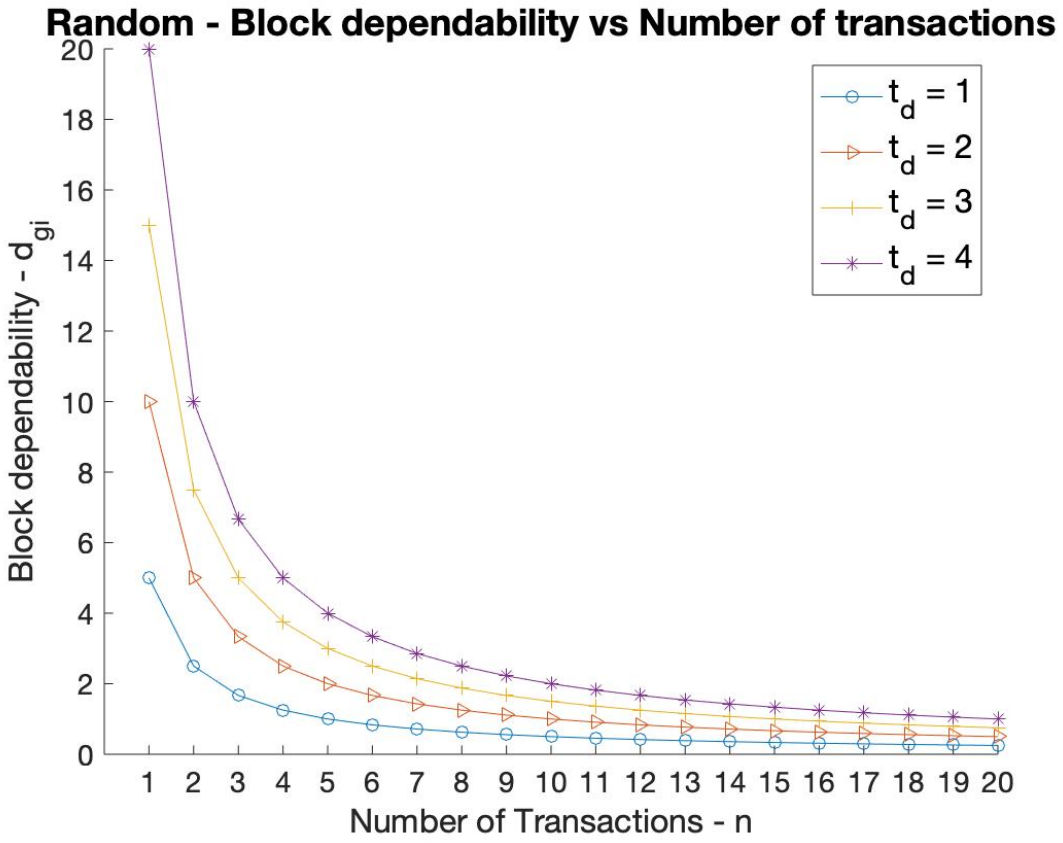


Figure 1. Block dependability vs. Number of transactions (Random)

Block dependability vs. Average speed of the transactions: The higher the average speed of the transactions coming into the transaction pool goes, the slower the processing and posting of those transactions goes, which would reduce the block dependability significantly, given a constant number of miners. Using the equations derived earlier for the random sampling from the transaction pool, it is shown in Figure 2 how the block dependability is related with the average speed of the transactions generated in the blockchain. Note that the average speed might vary considerably depending on the blockchain traffic.

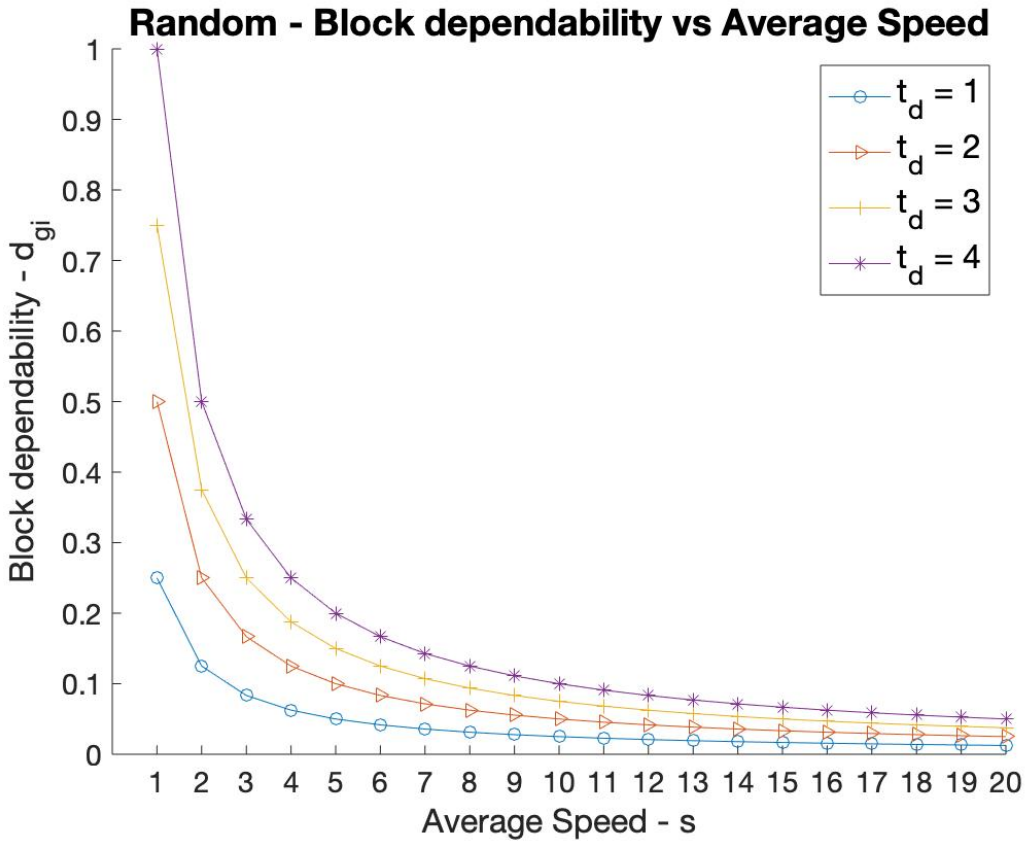


Figure 2. Block dependability vs. Average speed (Random)

Block dependability vs. Deadline: The relation between the block dependability and the deadline itself are shown in the case of the random sampling of transactions from the mining pool. If the deadline is extended, the higher would be the chance for a transaction to get into the targeted block. The increase in the block dependability is shown in Figure 3, assuming that the number of miners and, average speed of the incoming transactions is kept constant.

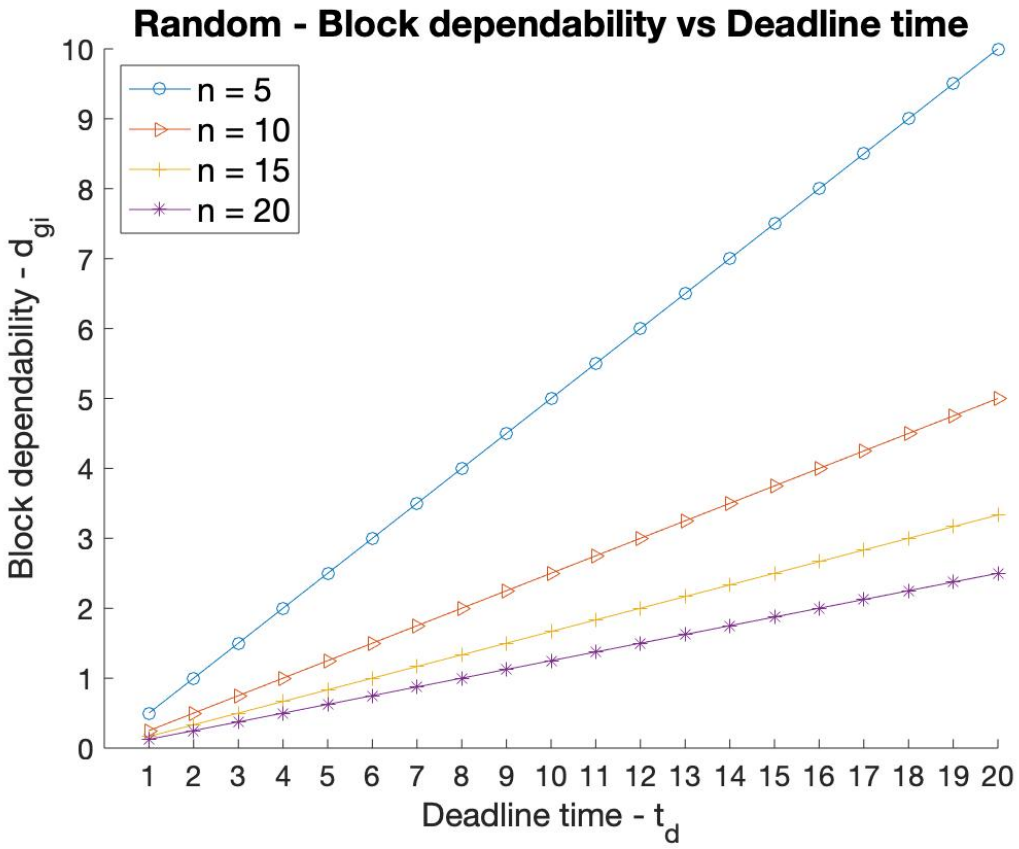


Figure 3. Block dependability vs. Deadline time (Random)

Block dependability vs. Number of miners: Lastly, in the case of random sampling of the transactions from the pool, it is demonstrated how the number of active miners present in the blockchain affect the block dependability of a particular transaction. The higher the number of miners goes the higher the block dependability turns as the processing of transactions would further speed up as shown in Figure 4.

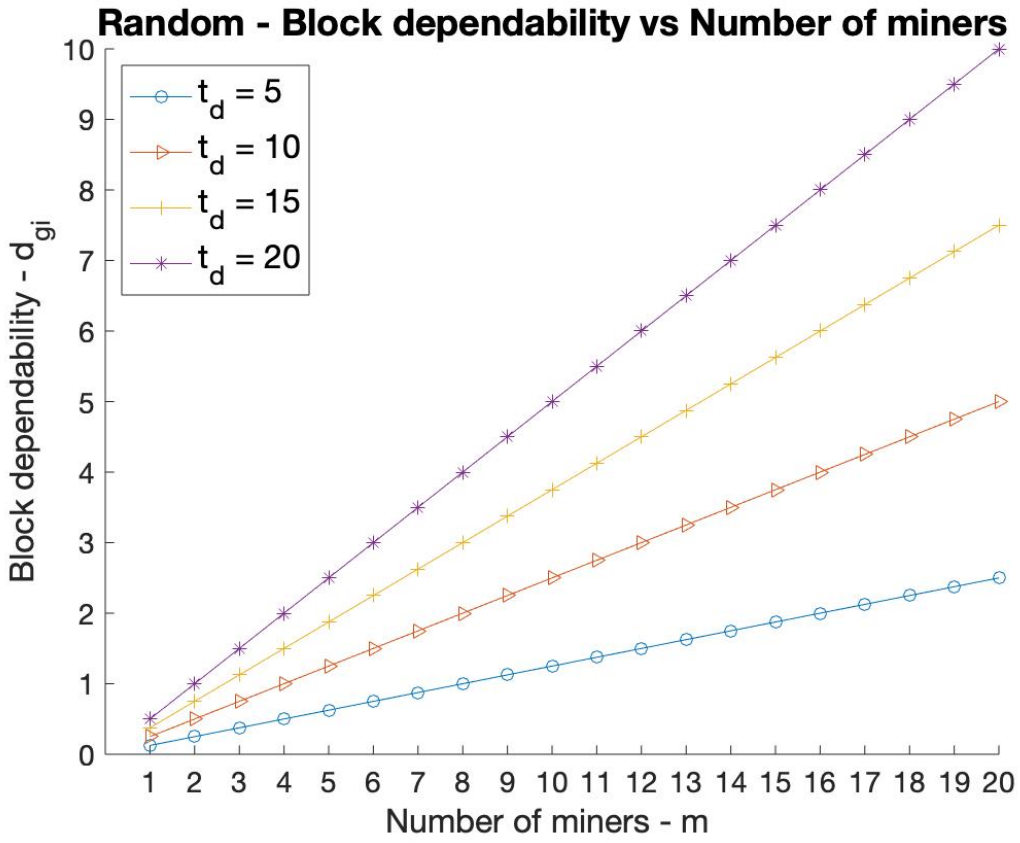


Figure 4. Block dependability vs Number of miners (Random)

The followings are the results from the sorted sampling algorithm (by gas fees in decreasing order as is exercised in the normal algorithm by the miners) of transactions from the mining pool and the observations made. Equation 40 is used to plot the following graphs.

Block dependability vs. Number of transactions: The relation of the block dependability vs. the number of transactions is very similar to the random case but with steeper slope for the transactions with higher gas fees as the transactions are sorted based on the gas fees. Figure 5 shows this scenario for different deadlines. As can be observed from the figure, the higher the number of transactions in the pool goes, the lower the block dependability goes. Also, as the number of transactions increases in the pool, the effect of deadline decreases.

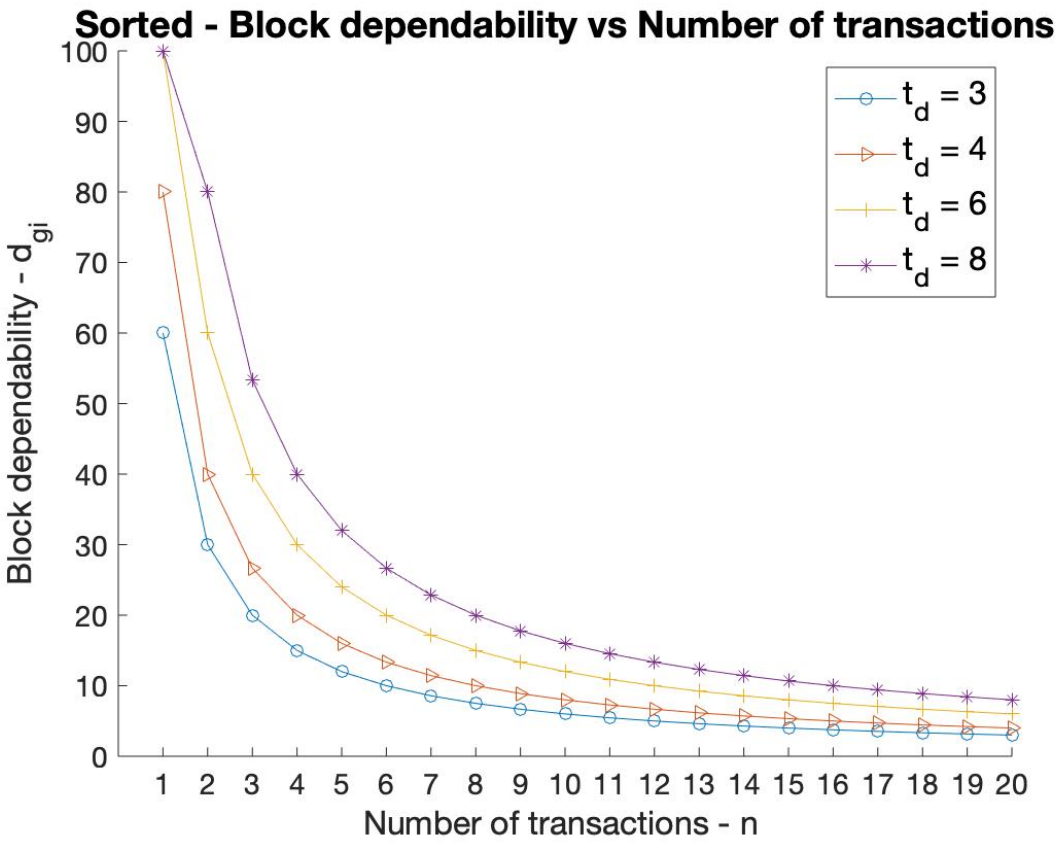


Figure 5. Block dependability vs. Number of transactions (Sorted)

Block dependability vs. gas fees of the transactions: As in this algorithm of sampling transactions from the pool, it involves sorting the transactions by the gas fees, and the transaction with the highest gas fee is supposed to be given the priority. Therefore, the block dependability of the transactions keeps increasing as the gas fees of that transaction increases. Note that there is only minimal increase in the block dependability beyond a certain gas fee. This value can be used to determine the minimum amount of gas fee that should be added to the transaction for it to have a high block dependability. The results of this scenario are shown in Figure 6.

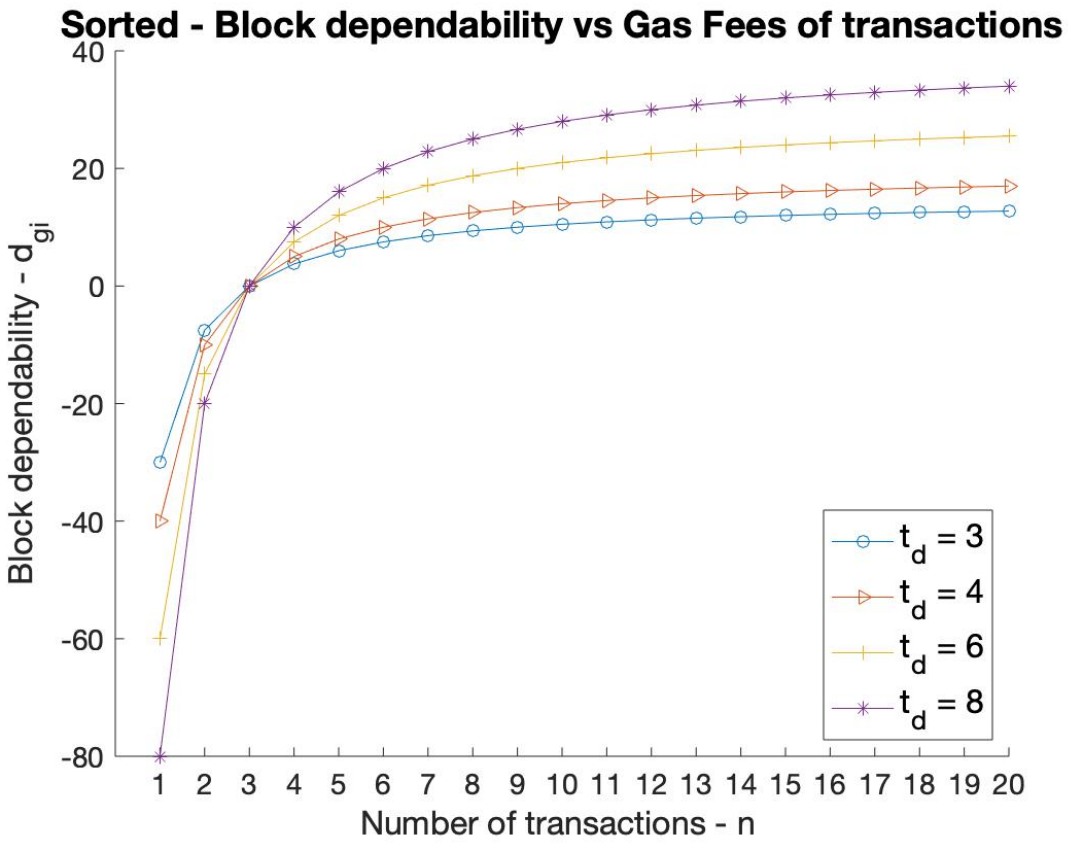


Figure 6. Block dependability vs. Gas fees (Sorted)

Block dependability vs. Speed of the transactions: As in the random case of comparing the block dependability with the speed of the transactions, the sorting algorithm also has a similar curvature but with steeper slope as well. Note that once the average speed of the transactions hits a certain threshold, the gas fees do not have that high of an effect on the block dependability. The faster the transactions are incoming into the transaction pool, the lower the block dependability of all the transactions becomes. Figure 7 shows the results from this scenario.

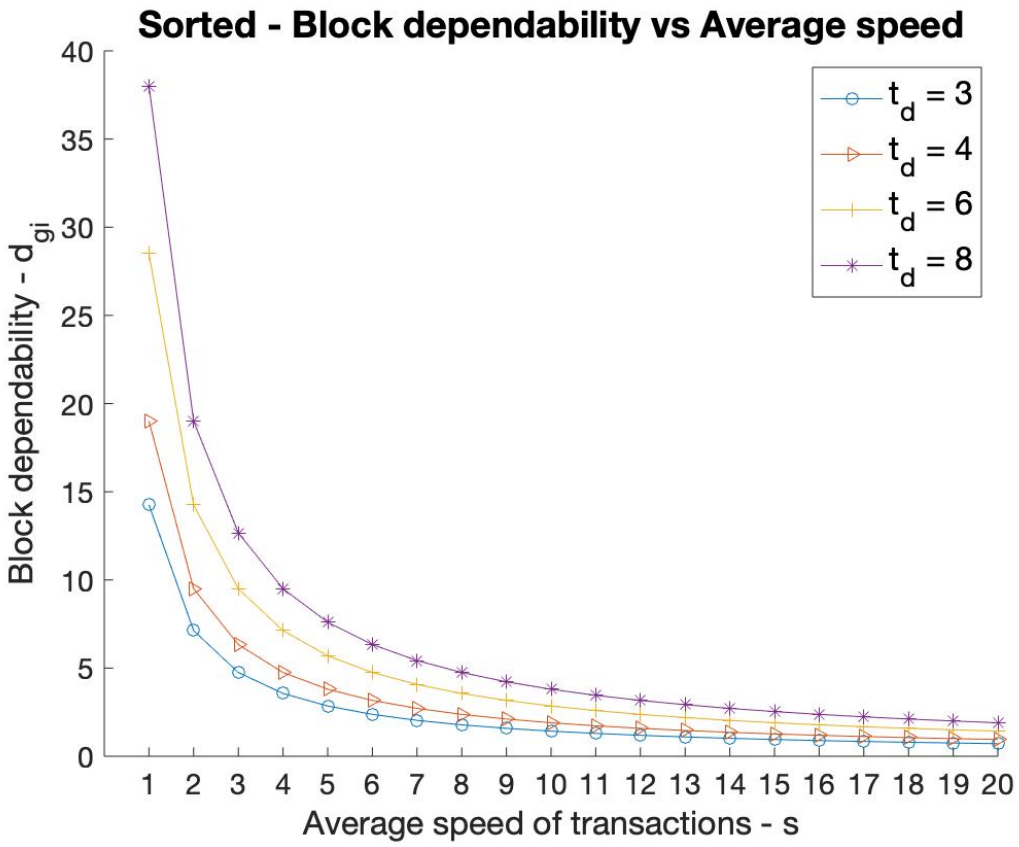


Figure 7. Block dependability vs. Average speed of the transactions (Sorted)

Block dependability vs. Number of miners: Lastly, in the sorted way of sampling transactions from the mining pool, the relation between the block dependability and the number of active miners present in the blockchain is shown. The higher the number of miners goes, the higher the processing of transactions would further speed up. Figure 8 shows the increase in block dependability with the increase in the number of active miners.

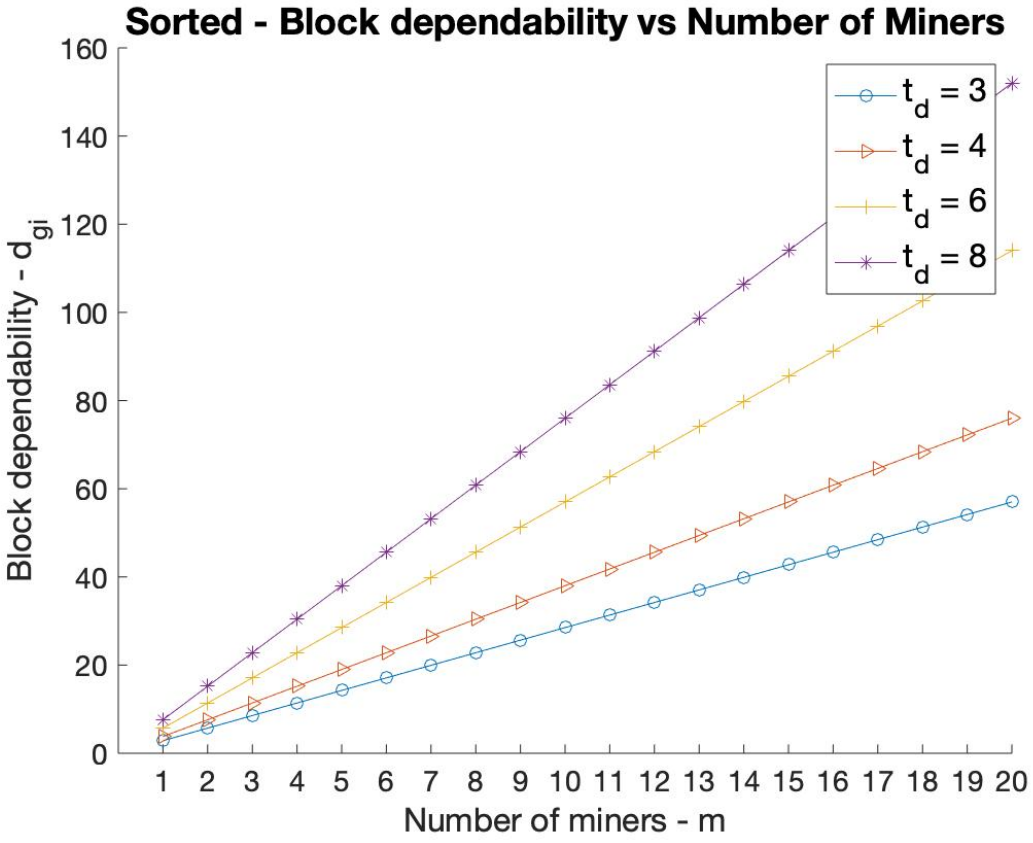


Figure 8. Block dependability vs Number of miners (Sorted)

PROPOSED VARIABLE BULK ARRIVAL AND VARIABLE BULK SERVICE WITH $\boldsymbol{\lambda_F}$ FOR REALTIME CHAIN

In the proposed Realtime Chain model, an embedded Markovian single-server exponential queueing system (i.e., $M^{1,n}/ M^{1,n} / 1$) is considered without loss of generality, and the server (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 the 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 of any size between 0 and n, inclusive (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 successfully within deadline yet at the rate of λ_F the customers are assumed to fail to arrive within a specific deadline to meet the realtime requirement. The underlying queueing process is assumed to take place with variable-sized slots and the status of the queue is determined by the number of slots of any size in the current block.

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

- 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 [19].
- P_n : the state in which there are 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 [19].
- P_i : the state in which there are i number of slots (where $0 < i < n$) arrived in the queue for the posting in the block, currently [19].

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

- λ: the rate for a slot of a transaction to arrive successfully within the realtime deadline requirement, and the rate for a transaction to arrive is determined by the number of slots allocated for the transaction in a prorated manner such that a transaction with a size of j number of slots arrives at the rate of $j\lambda$, without loss of generality and practicality as well.
- λ_F: the rate for a slot of a transaction to arrive unsuccessfully past the realtime deadline requirement, and at the rate, the state will self-loop without making any state transition. This rate is the unique one to distinguish VBAVBS from VBASBS in which there was no realtime deadline requirement taken into consideration.
- μ: the rate for the slots of the transactions in the entire queue to be posted and purged. Notice that this is a single and unique state transition 1, 2 and 3.

The balance equations for VBAVBS with λ_F are shown in Appendix C. The final values of the steady state values are given in Equation 41.

$$P_i = q_i^{-1} P_0 \left[\sum_{j=1}^{i} j \left[\sum_{k=1}^{i-1} \left[\prod_{l=1}^{k-1} q_l^{-1} \right] \right] k \right] \tag{41}$$

The followings are a few baseline performance measurements of primary interests in VBAVBS with λ_F.

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

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

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

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

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

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

- 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) [19].

$$L = \lambda W$$

ANALYSIS

The efficacy of the proposed VBAVBS model with λ_F is tested and verified through numerical analysis for the L_Q,W_Q,W and L versus n (i.e., size of a block), λ (i.e., successful transaction arrival rate or speed), λ_F (i.e., unsuccessful transaction arrival rate or speed) and $\frac{1}{\mu}$ (i.e., block posting time). Figure 1 plots Average number of customers in system (L) versus number if slots (n), for various λ and a μ (at 1/15).

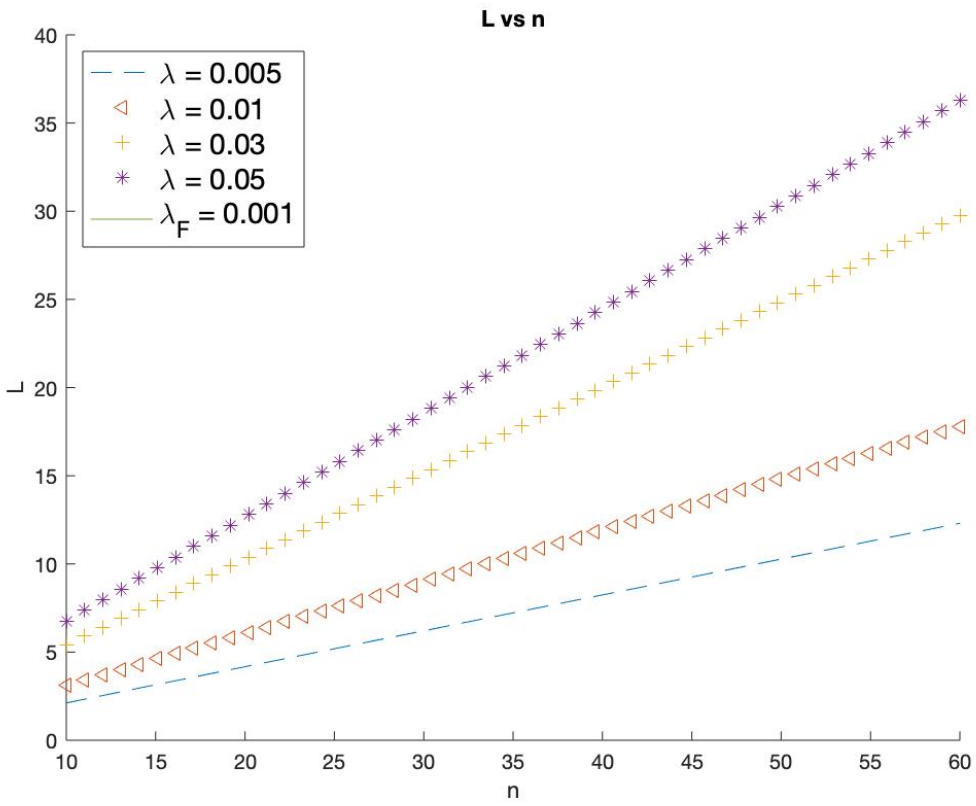


Figure 9. Average number of customers in system (L) vs Number of slots (n)

L versus n, for various λ and a μ, are plotted in Figure 9, and it is observed that L picks up as n increases as expected yet in a near linear manner. Also, it is observed that L picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced.

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

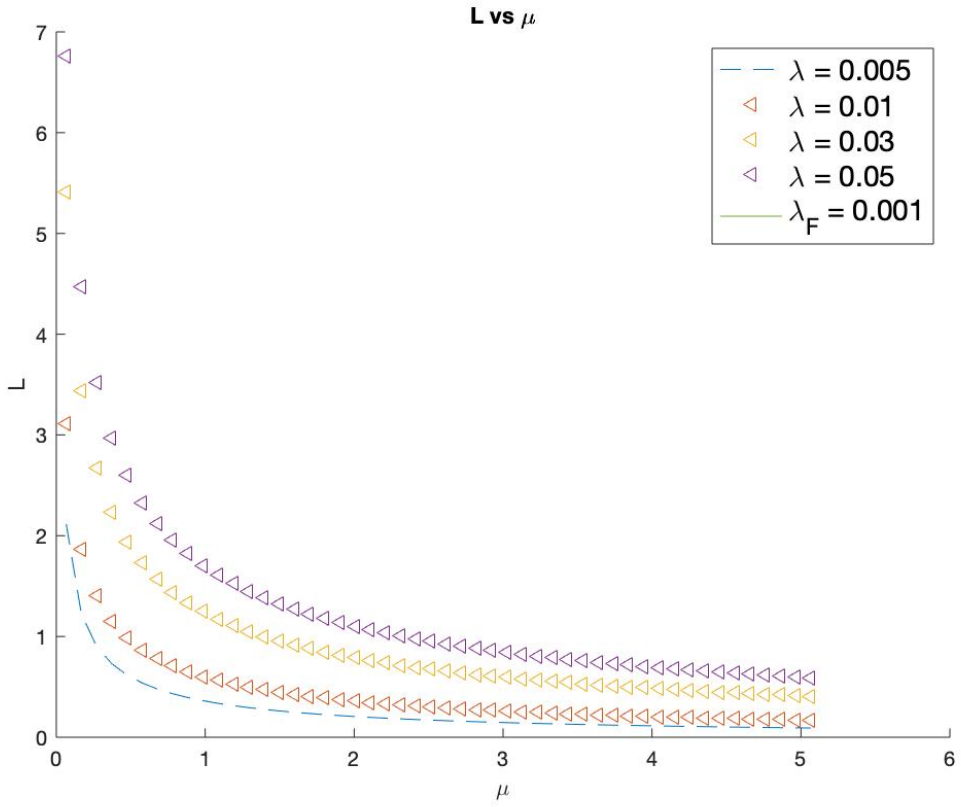


Figure 10. Average number of customers in the system (L) vs Rate of slots (μ)

It is observed that L declines as μ increases as expected. Also, it is observed that L picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced. The following graph plots L_Q versus n, for various λ and a μ (at 1/15).

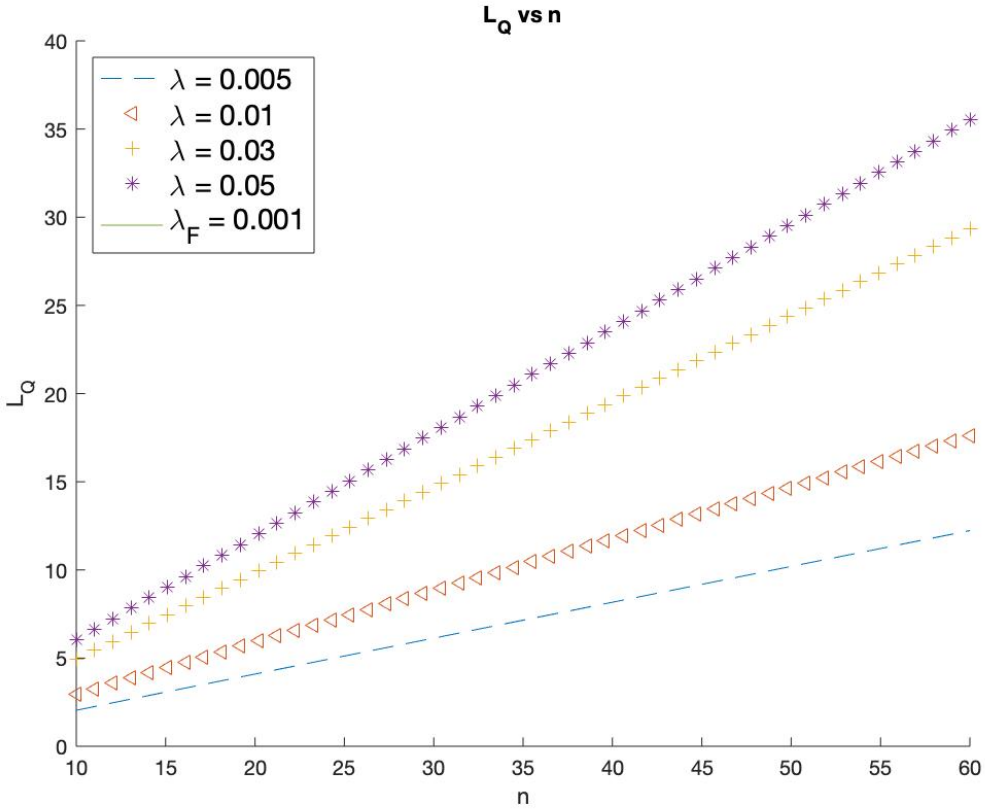


Figure 11. Average number of customers in the queue (L_Q) vs Number of Slots (n)

It is observed in Figure11 L_Q picks up as n increases as expected yet in a near linear manner. Also, it is observed that L_Q picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced. Average number of customers in queue (L_Q) versus rate of slots (μ) is plotted in Figure 12, for various λ, and an n (at 10).

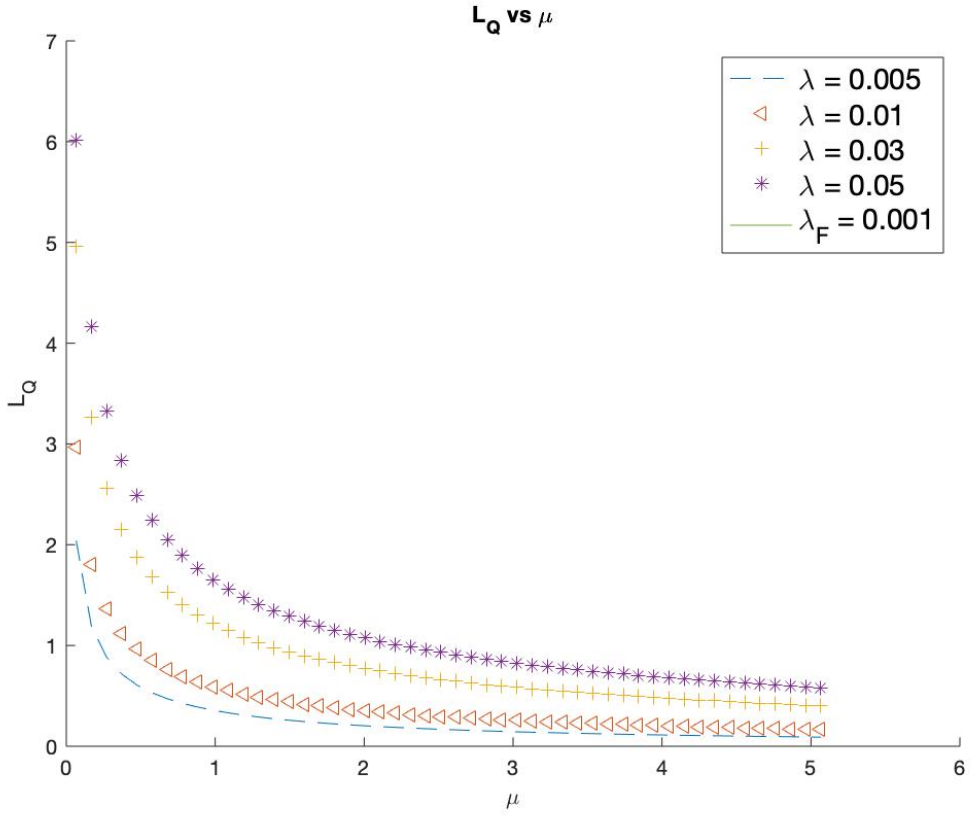


Figure 12. Average number of customers in the queue (L_Q) vs Rate of slots (μ)

It is observed that L_Q declines as μ increases as expected. Also, it is observed that L_Q picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced. The following graph plots W versus n, for various λ and a μ (at 1/15).

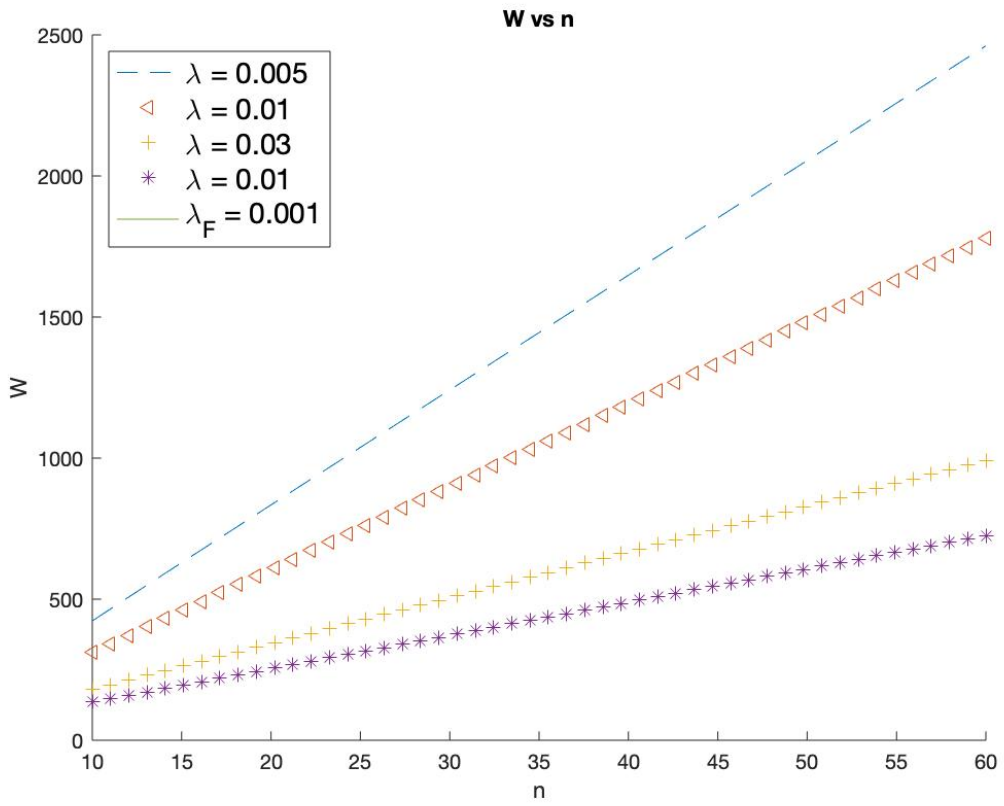


Figure 13. Average amount of time in system (W) vs Number of slots (n)

It is observed in Figure 13 W picks up as n increases as expected yet in a near linear manner. Also, it is observed that W picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced. The following graph plots W versus μ, for various λ, and n (at 10).

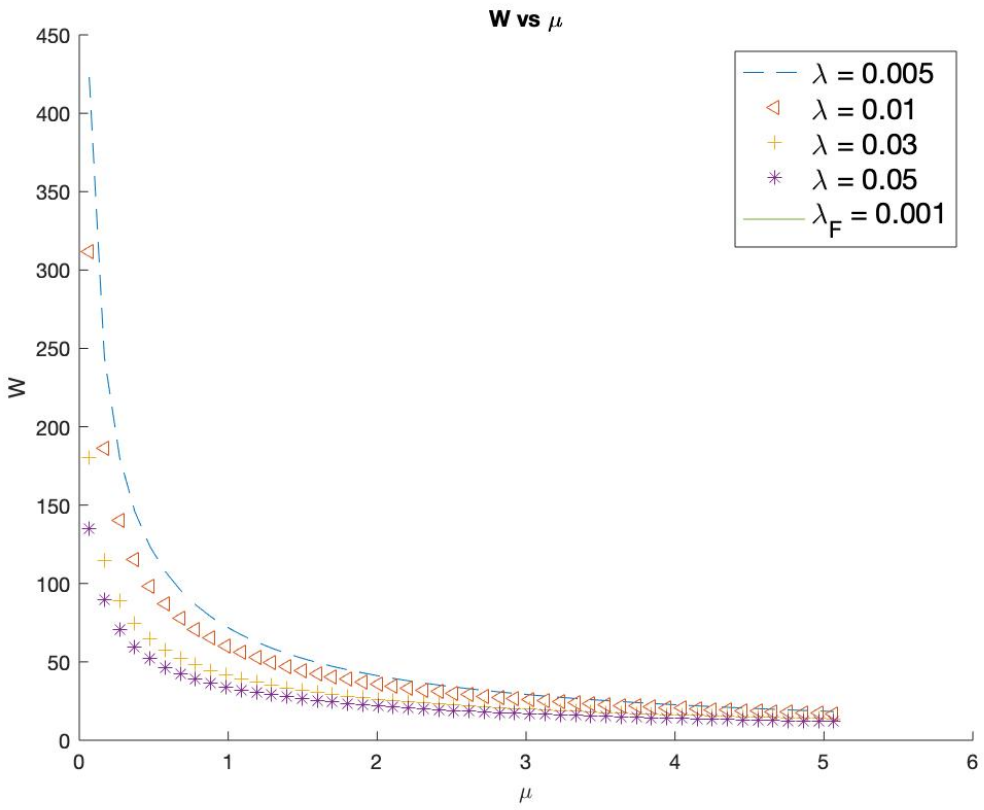


Figure 14. Average amount of time in system (W) vs Rate of slots (μ)

It is observed that in Figure 14 W declines as μ increases as expected. Also, it is observed that W picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced. Figure 7 plots W_Q versus n, for various λ and a μ (at 1/15).

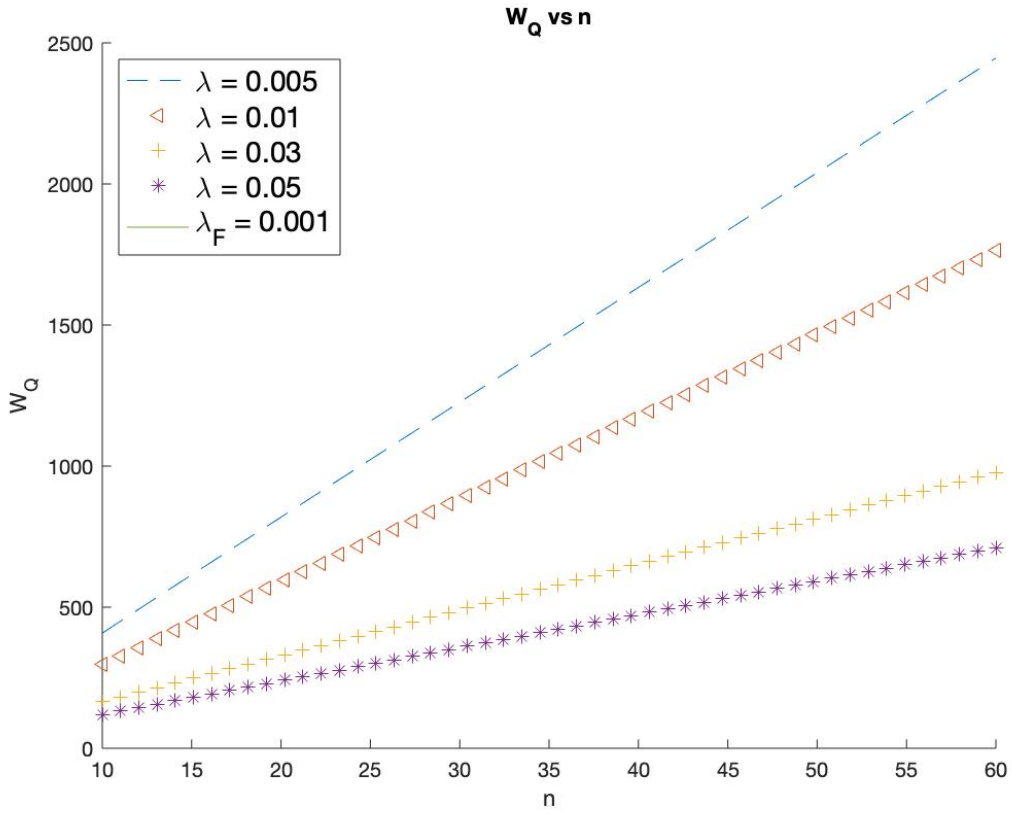


Figure 15. Average amount of time in queue (W_Q) vs Number of slots (n)

It is observed that W_Q picks up as n increases as expected yet in a near linear manner. Also, it is observed that W_Q picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced. The following graph plots W_Q versus μ, for various λ, and n (at 10).

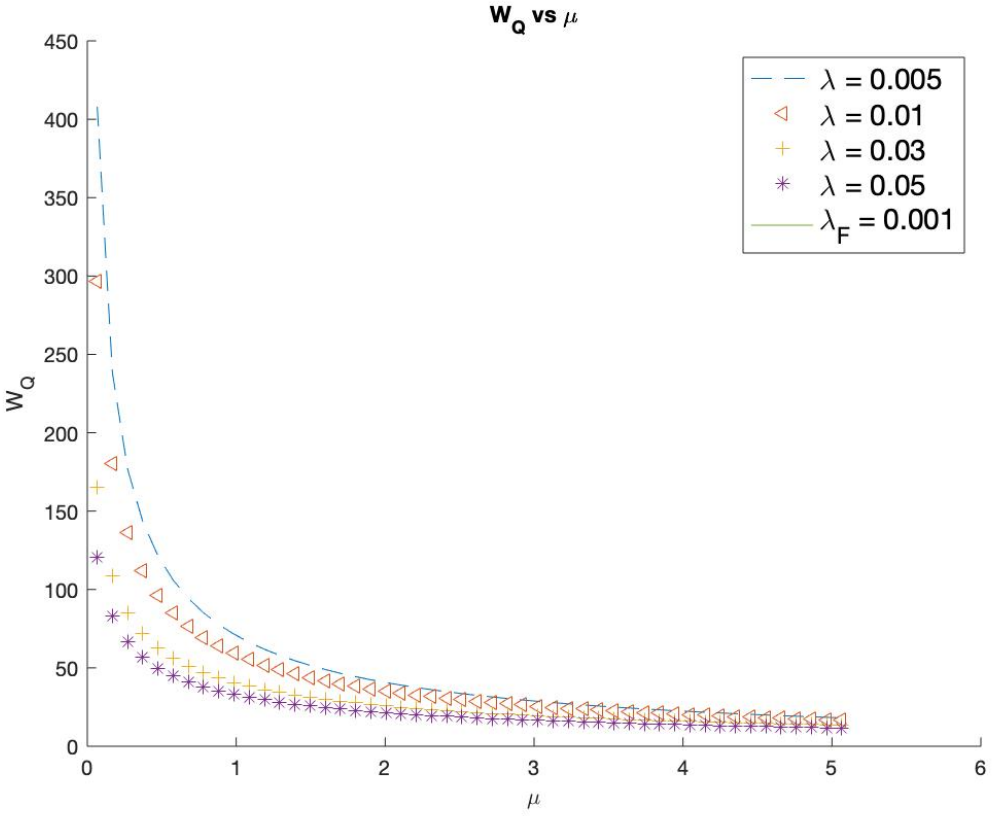


Figure 16. Average amount of time in queue (W_Q) vs Rate of Slots (μ)

It is observed in Figure 16 W_Q declines as μ increases as expected. Also, it is observed that W_Q picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced. Equation 42 shows the throughput per block in the presented model.

$$\gamma = \mu P_n = \mu \frac{\lambda}{\mu} \frac{n(n+1)}{2} P_0 = \lambda \frac{n(n+1)}{2} P_0 \tag{42}$$

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

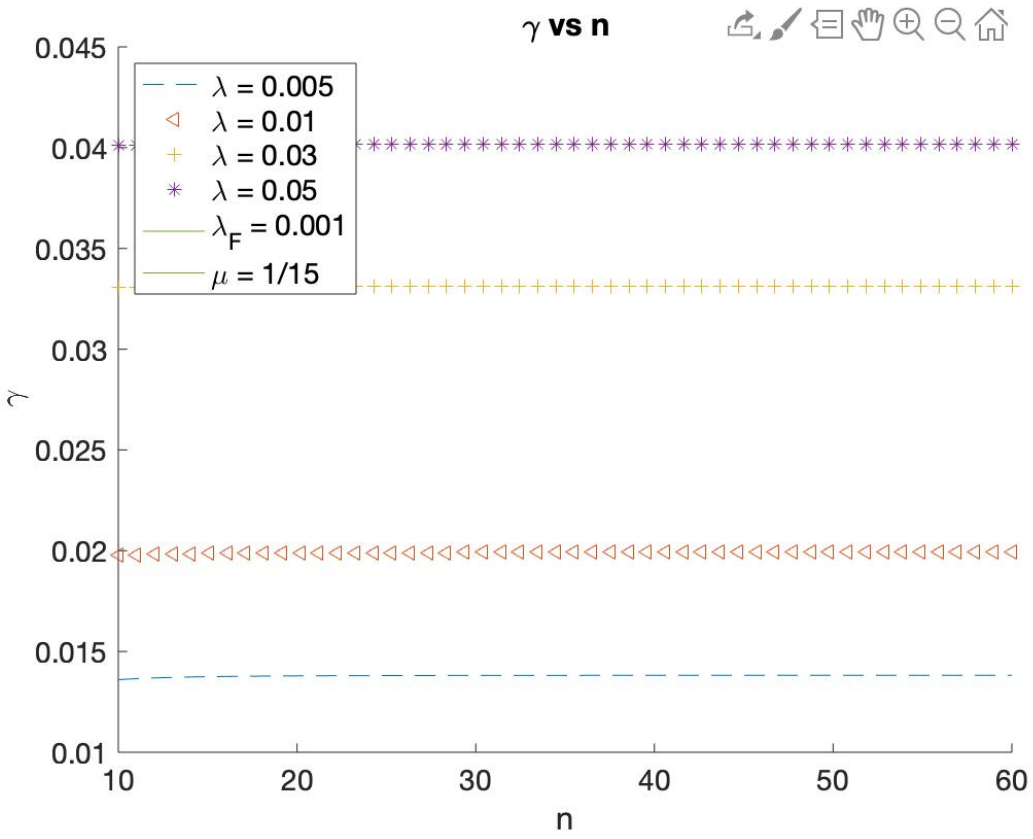


Figure 17. Throughput per block (γ) vs Number of slots (n)

Figures 17 and 18 plot the throughput per block. It is observed that γ stays constant throughout as P_n barely changes throughout. The following graph plots γ versus μ, for various λ, a μ (at 1/15) and a λ_F (at 0.001).

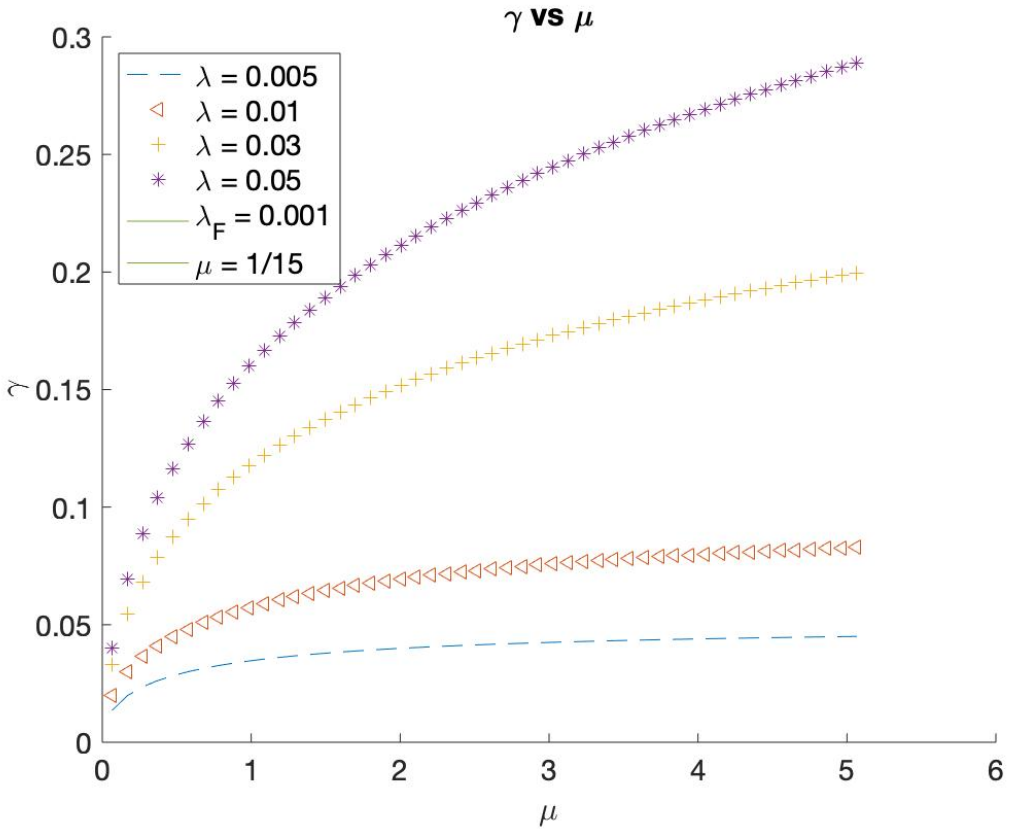


Figure 18. Throughput per block (γ) vs Rate of slots (μ)

It is observed that γ picks up as μ increases as expected. Also, it is observed that γ picks up as λ increases as expected. Notice that the intervals in between the plots are quite proportionally spaced. The following graph plots L versus λ_F, for various λ, a μ (at 1/15), and n (at 10).

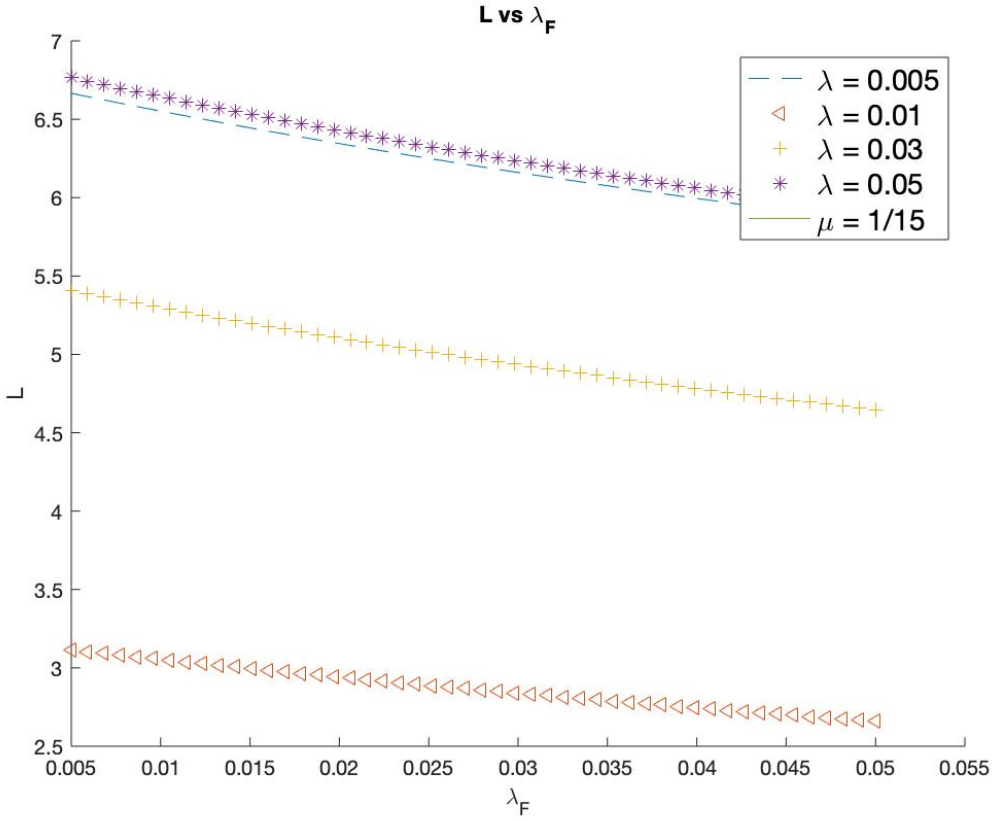


Figure 19. Average number of customers in system (L) vs Rate of unsuccessful arrival (λ_F)

It is observed in Figure 19 L declines as λ_F increases such that as more number of transactions (or slots) fail to arrive within the required realtime deadline, less number of transactions will be accommodated. Also, it is observed that L picks up as λ increases as expected. Notice that the intervals in between the plots are spaced narrower at higher λ as expected. The following graph plots L_Q versus λ_F, for various λ, a μ (at 1/15), and n (at 10) in Figure 20.

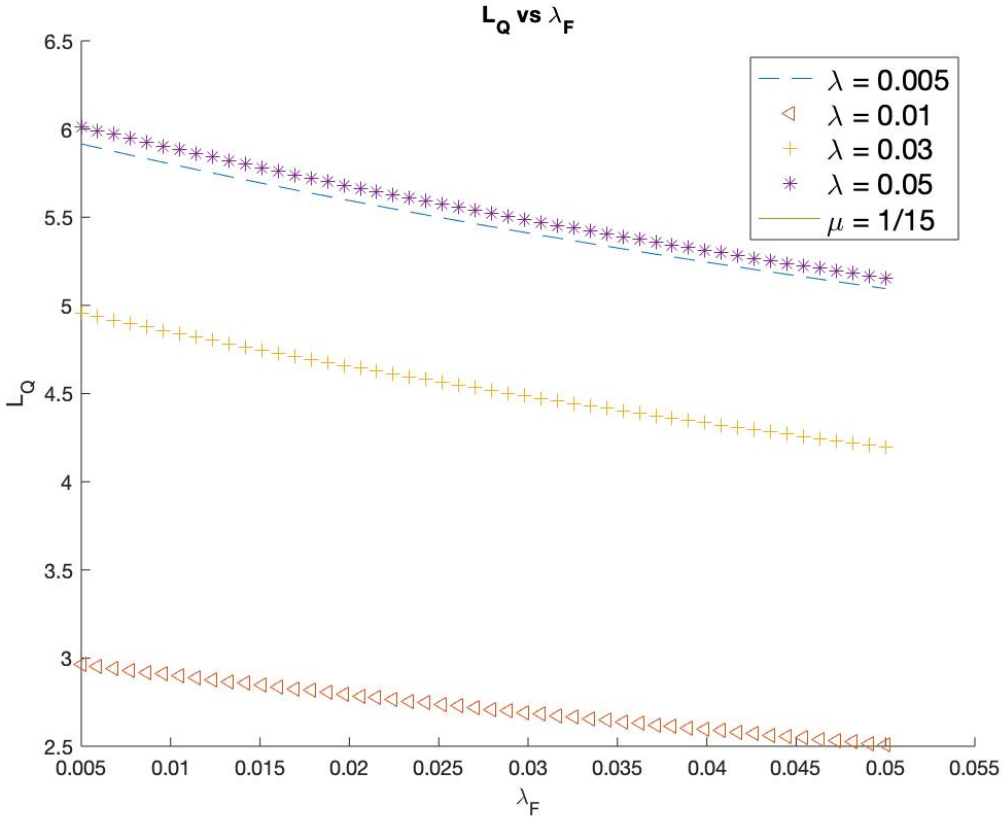


Figure 20. Average number of customers in queue (L_Q) vs Rate of unsuccessful arrival (λ_F)

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

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

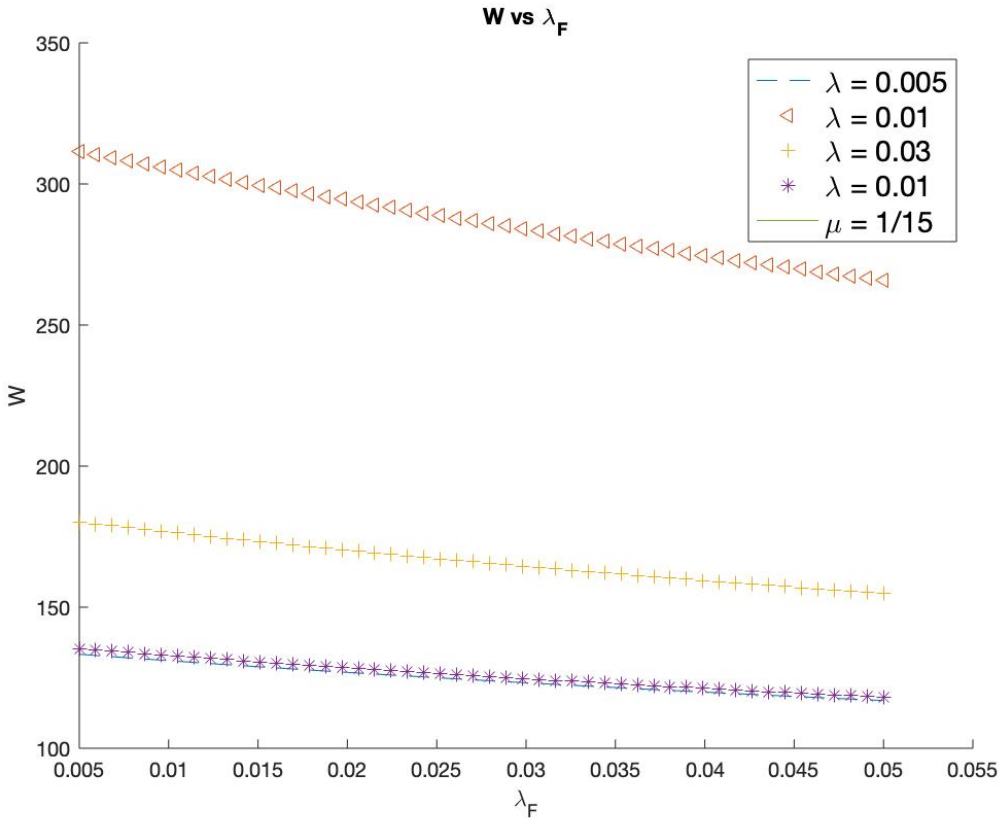


Figure 21. Average amount of time in system (W) vs Rate of unsuccessful arrival (λ_F)

Figures 21 and 22 plot the average amount of time in the system/queue with respect to rate of unsuccessful arrival. It is observed that W declines as λ_F increases as more number of transactions (or slots) fail to arrive within the required realtime deadline, less amount of time on average each transaction (or slot) will be waiting in the block prior to posting. Also, it is observed that W picks up as λ increases as expected. Notice that the intervals in between the plots are spaced narrower at higher λ as expected.

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

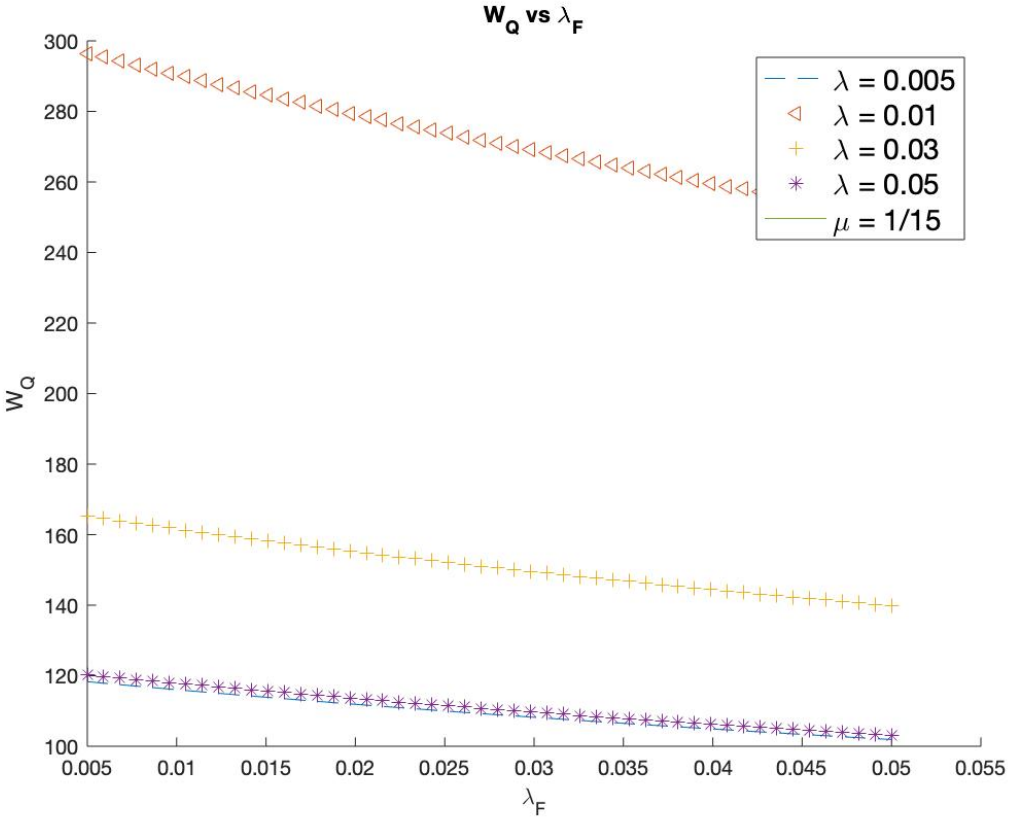


Figure 22. Average amount of time in queue (W_Q) vs Rate of unsuccessful arrival (λ_F)

It is observed in Figure 22 W_Q declines as λ_F increases as more number of transactions (or slots) fail to arrive within the required realtime deadline, less amount of time on average each transaction (or slot) will be waiting in the block prior to posting. Also, it is observed that W picks up as λ increases as expected. Notice that the intervals in between the plots are spaced narrower at higher λ as expected.

IMPLEMENTATION

This section details on the different transaction-sampling algorithms that were put to test by implementing the real-time model using the Ethereum open source. The flow diagram below shows the flow of control related to the mining and transaction sampling algorithm.

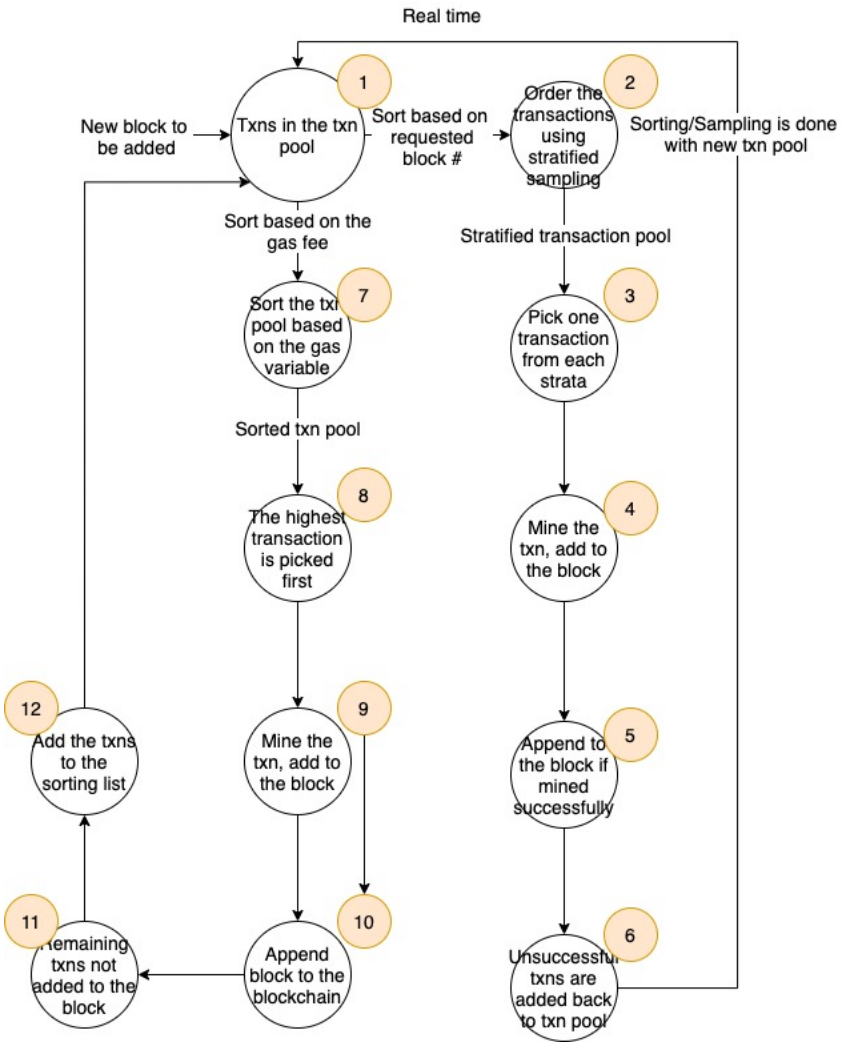


Figure 23. Realtime flowchart

To analyze those different transaction-sampling algorithms, 4 transactions are posted with different gas fees and at different times are demonstrated in Figures 24, 25, 26 and 27.

1. Sorted (e.g., the normal case as in decreasing order): The transactions in the transaction pool are sorted and then picked by miners. It is shown in Figure 24 that the transactions with highest gas fees are posted in earlier blocks.

Figure 24. Sorted mining order transaction postings

Random: The transactions from the transaction pool are sampled at random irrespective of the gas fee and arrival time of the transactions. This gives equal probability to all of the transactions in

the transaction pool. The randomness of the transaction sampling by the miners is shown below for the same quantity of transactions posted as earlier. Figure 25 shows the ordering of the transactions in the block by the random algorithm is different from the one by the sorted (normal) algorithm.

Figure 25. Random mining order transaction postings

Sorted (increasing order unlike the normal sorted): Similar to the sorted (normal) algorithm presented earlier in which the transactions are sorted in decreasing order, yet it sorts the transactions in increasing order of the variable taken into consideration. In this example, the transactions are sorted based on gas fees so that the higher the gas fee goes, the later the transactions are posted onto the block. Figure 26. depicts the scenario where in higher gas fee transactions are posted in later blocks.

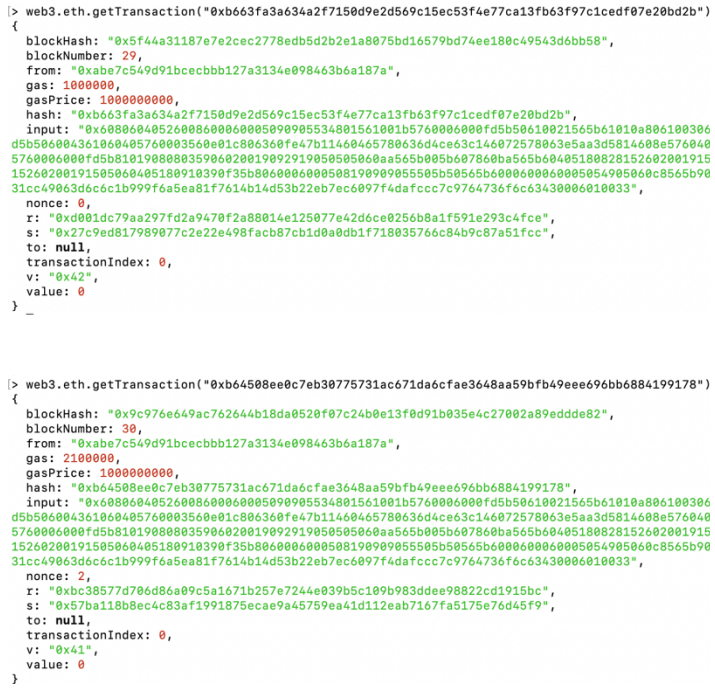


Figure 26. Sorted mining (increasing) order transaction postings

Stratified: In the stratified sampling algorithm, the transactions are stratified based on a variable and one transaction is sampled from each group. In this demonstration, the transactions are grouped based on the gas fee and one transaction is sampled from each group. The following screenshots in Figure 27 demonstrate the stratified algorithm where the transactions below 30000 of gas are posted in one block while the transactions above a certain range are posted in another block.

All the different types of sampling techniques that were proposed were implemented using the Ethereum open source.

Figure 27. Stratified mining order transaction postings

CHAPTER VI

HYBRID CHAIN

There are few disadvantages of a public blockchain and permissioned blockchain as well. To mitigate the individual disadvantages of these two types of blockchain, a hybrid chain is proposed. The proposed hybrid chain model supposedly undermines the disadvantages in both the different types of blockchain by combining the two together to form a new system a transaction could be posted in both a public and permissioned blockchain depending on the type of the transaction. A theoretical approach on the feasibility of the Hybrid chain is developed using a Markovian queueing model on top of the single-tuple model [32, 34] to create a double-tuple model. The proposed double-tuple VBASBS queueing model for the hybrid chain has a total of $(n+1)^2$ states i.e., from $P_{0,0}$ to $P_{n,n}$, unlike in a single-tuple queuing model which is known to hold $(n+1)$ states. The states in the double-tuple VBASBS model are defined as follows:

(i, j): i the number of slots in the blockchain-0, j the number of slots in blockchain-1

The state transition probabilities are defined as follows:

$t\big((i,j) \rightarrow (i,k)\big) = (k-j)\lambda_0$, where $j < k$ and $k < n$.

$t\big((i,n) \rightarrow (i,0)\big) = \mu_0$

$t\big((n,j) \rightarrow (0,j)\big) = \mu_1$

$t\big((i,j) \rightarrow (k,j)\big) = (k-i)\lambda_1$, where $i < k$ and $k < n$

The transitions for the state model are given by

$$t\big((i,j) \rightarrow (i,k)\big) = (k-j)\,\lambda_0 \tag{43}$$

where $j < k$ and $k \leq n$.

$$t\big((i,n) \rightarrow (i,0)\big) = \mu_0 \tag{44}$$

$$t\big((i,j) \rightarrow (k,j)\big) = (k-i)\lambda_1 \tag{45}$$

$$t\big((n,j) \rightarrow (0,j)\big) = \mu_1 \tag{46}$$

where $i < k$ and $k \leq n$.

The total number of states in the entire model are $(n+1)^2$, which are denoted as $P_{i,j}$, where $0 \leq i,j \leq n$. The balance equations for the proposed state transition model are shown below. The steady state equations are obtained using $Steady\ state\ probability\ *$

$\sum(all\ outgoing\ transitions) = \sum(steady\ satte\ probabilities * incoming\ transition)$

STEADY STATE PROBABILITIES

Equation 47 shows the steady state equation for $P_{0,0}$ as follows.

$$P_{0,0}\big((\lambda_0 + 2\lambda_0 + 3\lambda_0 + \cdots + n\lambda_0) + (\lambda_1 + 2\lambda_1 + 3\lambda_1 + \cdots + n\lambda_1)\big) = \mu_0 P_{0,n} + \mu_1 P_{n,0} \quad (47)$$

$$P_{0,0}\big(\lambda_0(1 + 2 + 3 + \cdots + n) + \lambda_1(1 + 2 + 3 + \cdots + n)\big) = \mu_0 P_{0,n} + \mu_1 P_{n,0}$$

$$P_{0,0}\big((1 + 2 + 3 + \cdots + n)(\lambda_1 + \lambda_0)\big) = \mu_0 P_{0,n} + \mu_1 P_{n,0}$$

$$P_{0,0}\left(\frac{n(n+1)}{2}(\lambda_1 + \lambda_0)\right) = \mu_0 P_{0,n} + \mu_1 P_{n,0}$$

$$t\big((n,j) \to (0,j)\big) = \mu_1$$

Steady state equation for $P_{0,1}$ is given in Equation 48 as follows.

$$P_{0,1}\big((\lambda_0 + 2\lambda_0 + 3\lambda_0 + \cdots + (n-1)\lambda_0) + (\lambda_1 + 2\lambda_1 + 3\lambda_1 + \cdots + n\lambda_1)\big) = \mu_1 P_{n,1} + \lambda_0 P_{0,0}$$

$$P_{0,1}\left(\lambda_0\big(1 + 2 + 3 + \cdots + (n-1)\big) + \lambda_1(1 + 2 + 3 + \cdots + n)\right) = \mu_1 P_{n,1} + \lambda_0 P_{0,0}$$

$$P_{0,1}\left(\lambda_0\left(\frac{n(n-1)}{2}\right) + \lambda_1\left(\frac{n(n+1)}{2}\right)\right) = \mu_1 P_{n,1} + \lambda_0 P_{0,0} \quad (48)$$

The steady state equations shown below are for the first row of the states in the state model, which can be generalized as follows.

$$P_{0,i}\big((\lambda_0 + 2\lambda_0 + 3\lambda_0 + \cdots + (n-i)\lambda_0) + (\lambda_1 + 2\lambda_1 + 3\lambda_1 + \cdots + n\lambda_1)\big) = \mu_1 P_{n,i}$$

$$P_{0,i}\left(\lambda_0\big(1 + 2 + 3 + \cdots + (n-i)\big) + \lambda_1(1 + 2 + 3 + \cdots + n)\right) = \mu_1 P_{n,i} + \lambda_0 P_{0,i-1} + 2\lambda_0 P_{0,i-2} + \cdots + i\lambda_0 P_{0,0}$$

$$P_{0,i}\left(\lambda_0\left(\frac{(n-i)(n-i+1)}{2}\right) + \lambda_1\left(\frac{n(n+1)}{2}\right)\right) = \mu_1 P_{n,i} + \lambda_0 P_{0,i-1} + 2\lambda_0 P_{0,i-2} + \cdots + P_{0,0} i\lambda_0$$

The steady state equation for the last value state in the first row, which is $P_{0,n}$is given as follows:

$$P_{0,n}\big((\lambda_1 + 2\lambda_1 + 3\lambda_1 + \cdots + n\lambda_1)\big) = \mu_1 P_{n,n} + \lambda_0 P_{0,n-1} + 2\lambda_0 P_{0,n-2} + \cdots + n\lambda_0 P_{0,0}$$

$$P_{0,n}\big(\lambda_1(1 + 2 + 3 + \cdots + n)\big) = \mu_1 P_{n,n} + \lambda_0 P_{0,n-1} + 2\lambda_0 P_{0,n-2} + \cdots + n\lambda_0 P_{0,0}$$

$$P_{0,n}\lambda_1\left(\frac{n(n+1)}{2}\right) = \mu_1 P_{n,n} + \lambda_0 P_{0,n-1} + 2\lambda_0 P_{0,n-2} + \cdots + n\lambda_0 P_{0,0}$$

The steady state equations for the second row ($P_{1,0}$ – $P_{1,n}$), are shown below, starting with $P_{1,0}$:

$$P_{1,0}\big((\lambda_0 + 2\lambda_0 + 3\lambda_0 + 4\lambda_0 + \cdots + (n-1)\lambda_0 + n\lambda_0) + (\lambda_1 + 2\lambda_1 + 3\lambda_1 + 4\lambda_1 + \cdots + (n-2)\lambda_1 + (n-1)\lambda_1)\big) = \mu_0 P_{0,n} + \mu_1 P_{0,0}$$

$$P_{1,0}\left(\lambda_0(1 + 2 + 3 + 4 + \cdots + (n-1) + n) + \lambda_1\big(1 + 2 + 3 + 4 + \cdots + (n-2) + (n-1)\big)\right) = \mu_0 P_{0,n} + \mu_1 P_{0,0}$$

$$P_{1,0}\left(\lambda_0\left(\frac{n(n+1)}{2}\right) + \lambda_1\left(\frac{n(n-1)}{2}\right)\right) = \mu_0 P_{0,n} + \mu_1 P_{0,0}$$

$$P_{1,0}\left(\lambda_0\left(\frac{n(n+1)}{2}\right) + \lambda_1\left(\frac{n(n-1)}{2}\right)\right) = \mu_0 P_{0,n} + \mu_1 P_{0,0}$$

The generalized steady state equation for any state in row 2 are shown in below mentioned Equations:

$$P_{1,i}\big((\lambda_0 + 2\lambda_0 + 3\lambda_0 + \cdots + (n-i)\lambda_0) + (\lambda_1 + 2\lambda_1 + 3\lambda_1 + \cdots + (n-1)\lambda_1)\big) = \mu_1 P_{0,i} + \lambda_0 P_{0,i-1} + 2\lambda_0 P_{0,i-2} + \cdots + i\lambda_0 P_{0,0}$$

$$P_{1,1}\left(\lambda_0\big(1 + 2 + 3 + \cdots + (n-i)\big) + \lambda_1\big(1 + 2 + 3 + \cdots + (n-1)\big)\right) = \mu_1 P_{0,i} + \lambda_0 P_{0,i-1} + 2\lambda_0 P_{0,i-2} + \cdots + i\lambda_0 P_{0,0}$$

$$P_{1,1}\left(\lambda_0\left(\frac{(n-i)(n-i+1)}{2}\right)+\lambda_1\left(\frac{n(n-1)}{2}\right)\right)$$
$$= \mu_1 P_{0,i} + \lambda_0 P_{0,i-1} + 2\lambda_0 P_{0,i-2} + \cdots + i\lambda_0 P_{0,0}$$

$$P_{0,1}\left(\lambda_0\left(\frac{n(n+1)}{2}\right)+\lambda_1\left(\frac{n(n-1)}{2}\right)\right) = \mu_1 P_{0,i} + \lambda_0 P_{0,i-1} + 2\lambda_0 P_{0,i-2} + \cdots + i\lambda_0 P_{0,0}$$

Given n rows and columns in the state model, there will be a total of $(n+1)^2$ equations. The final equation is where all the steady state probabilities sum up to one.

$$\sum_{i=0}^{n}\sum_{j=0}^{n} P_{i,j} = 1$$

The coefficients of all the equations are calculated based on the value of row and column values of each state. The coefficients are then input to Matlab in the form of a $((n+1)^2+1)$ by $(n+1)^2$, which represents matrix A in equation $Ax = B$. 'x' represents the steady state values matrix of dimensions $(n+1)^2$ by 1, which are $P_{0,0}, P_{0,1} \ldots P_{0,n}, P_{1,0}, \ldots P_{1,n} \ldots P_{n,n}$. The matrix '$B$' is the product of '$A$' and '$x$', which is of dimensions $((n+1)^2+1)$ by 1. The entire column of 'B' is zeros except for the very last part. The equations are then solved using the linsolve function in Matlab [14]. The results of the linsolve function are the steady state probabilities and the values are used to obtain the values of L_{Q_0} , L_{Q_1}, W_{Q_0}, W_{Q_1}, L_0, L_1, W_0, W_1, by which a few performances, such as the transaction execution time (e.g., W_{Q_0}, W_{Q_1}, W_0, W_1) and the block spatial requirement (e.g., L_{Q_0}, L_{Q_1},L_0, L_1), are analyzed under various assumptions on the transaction arrival and service (i.e., block posting process) through extensive numerical methods on Matlab. The values of specific variables obtained from solving the steady state equations, are plotted against different values of n, with a total of 8 graphs shown in Figures 1 - 8.

Variable L_{Q_0}, which is the average number of transactions in the public chain, is obtained by using the formula in Equation 28. Figures 1 and 2 demonstrate the validity of the proposed VBASBS double-tuple queueing model. Under the assumptions on the arrival rates (λ_0/λ_1) and service times (μ_0/μ_1), these two figures show a monotonically increasing trends of the average number of slots in a block as a representation of the number of transactions in both public and permissioned chain in a normalized manner as expected. 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 $P_{i,j}$ represents a transaction with i number of slots in the permissioned chain and j number of slots in the public chain and its steady state probability represents the normalized likelihood of the number of transactions of size $(i + j)$. As a result, the almost linearity is achieved in the Figures 1 and 2. The graphs shown might look exactly linear, but the data points show slight irregularity in the linearity. For example, the data points for n = 15,16,17 are 13.6213, 14.6208 and 15.6203, which indicate very slight change in the linearity.

ANALYSIS

Observations drawn from the simulation results in Figures 1 and 2 are such that as the size of the block increases, the average number of slots in the mined transactions to be posted in the block increases. The higher the number of slots, the more linear increase in the values of both L_{Q_0} and L_{Q_1}, as the value of n contributes to determining the final value of L_Q. 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 in both permissioned and public to be posted as soon as the block hits n, which does not lose any generality from the stand point 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 has very little impact on the growth rate of L_{Q_0} and L_{Q_1}, as all three plots drawn against different values of λ_0/λ_1 have the same consistency and linearity.

$$L_{Q_0} = \sum_{i=0}^{n}\sum_{j=0}^{n}(i)P_{i,j}$$

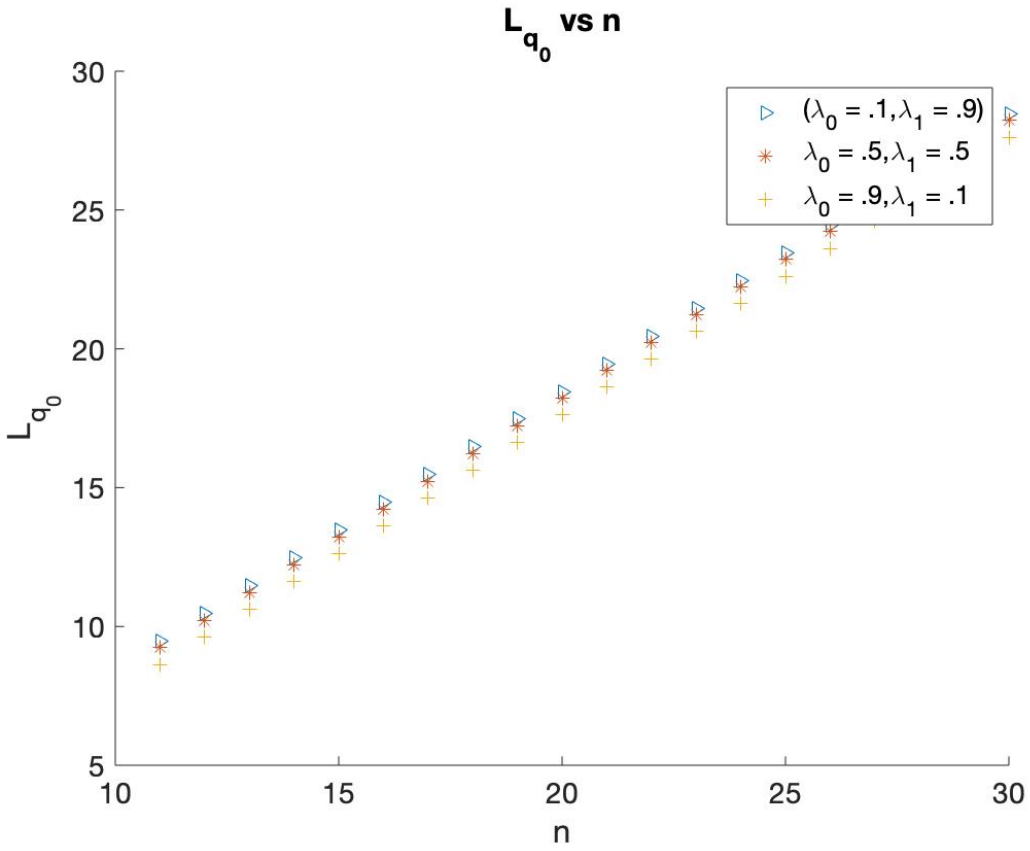


Figure 1: L_{q_1} vs n

The value of L_{Q_1}, which is the average number of transactions (or slots) in the permission-less chain, is obtained by using the following formula.

$$L_{Q_1} = \sum_{i=0}^{n}\sum_{j=0}^{n}(j)P_{i,j}$$

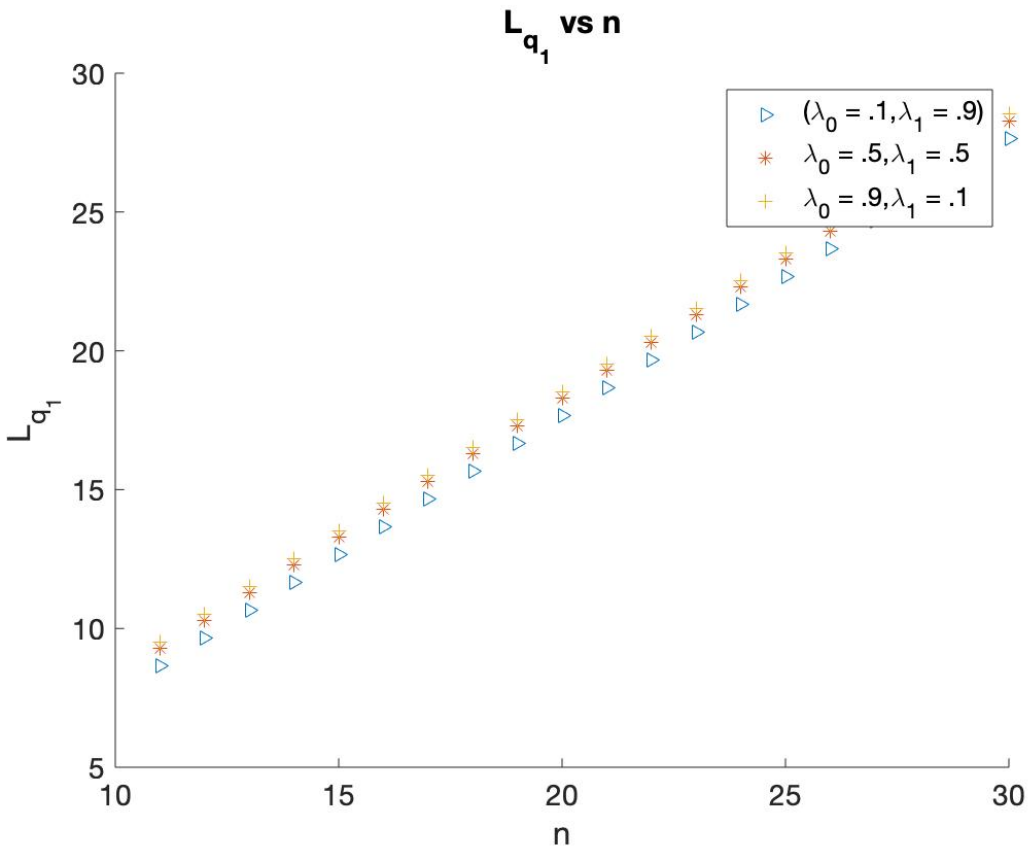


Figure 2: L_{q_1} vs n

As the proposed VBASBS double-tuple queueing model has been validated in the simulation shown in Figures 1 and 2 without loss of intuition, Figures 3 and 4 demonstrate the average number of transactions in the transaction pool waiting for mining selection and are determined using formulae in Equations 30 and 31. The only difference between L_{Q_0}/ L_{Q_1} and L_0 / L_1 being $\frac{1}{\mu_0}$ / $\frac{1}{\mu_1}$ respectively, which is the reason why the Figures 1, 2 and Figures 3, 4 share a similar linearity pattern. The values of L_0 / L_1 are slightly higher than the values of L_{Q_0}/ L_{Q_1}, which is as expected because the values of μ_0 / μ_1 are ranged between 0 and 1.

$$L_0 = \lambda_0 W_0 = L_{Q_0} + \frac{1}{\mu_0}$$

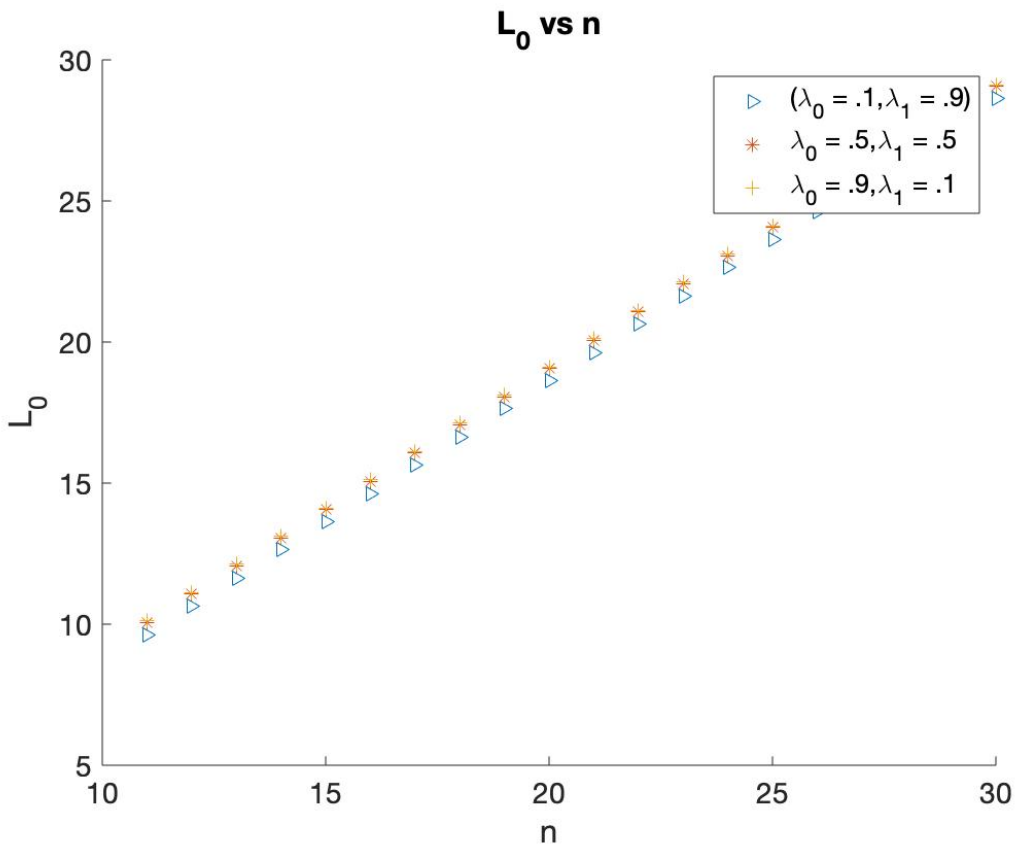


Figure 3. L_0 vs n

$$L_1 = \lambda_1 W_1 = L_{Q_1} + \frac{1}{\mu_1}$$

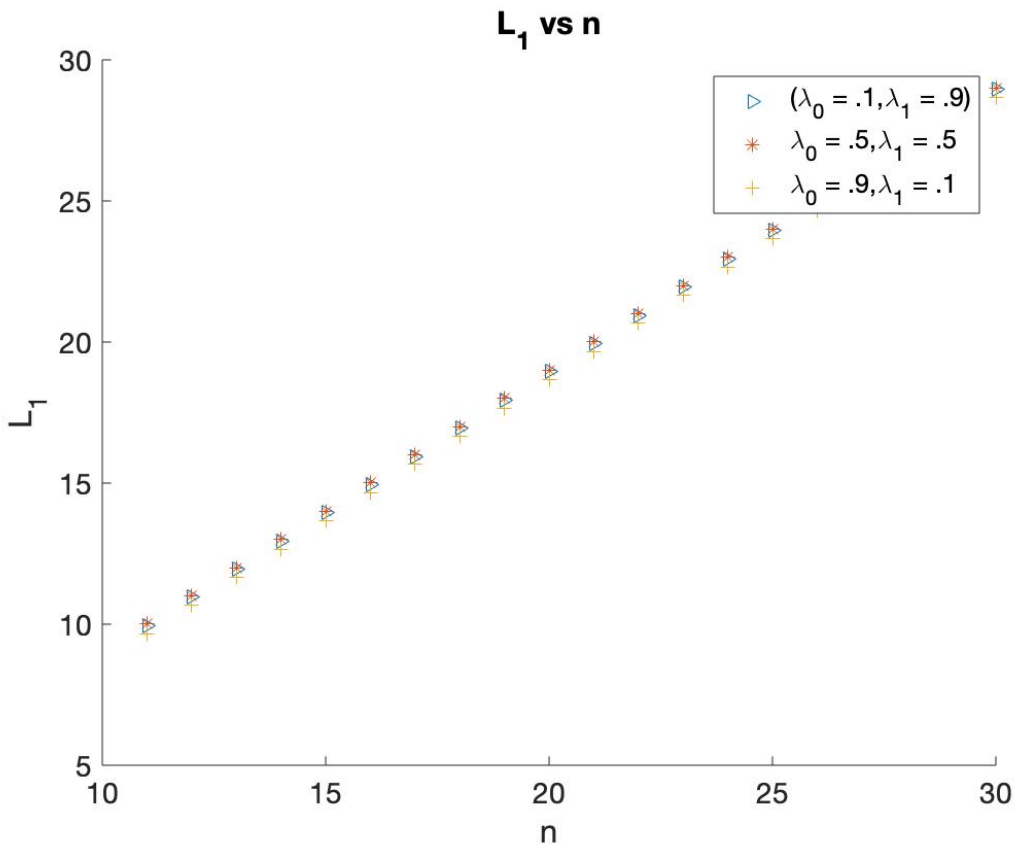


Figure 4. L_1 vs n

As mentioned for the previous figures, the curves in Figures 3 and 4 are also not perfectly linear. Figures 5 and 6 demonstrate the average waiting time of the mined transactions (or slots) for posting on the block, which is proportional to L_{Q_0} in a monotonic manner. It is looked into that for a given size of the block, the waiting time increases as the arrival rate decreases and also as the value of n increases. Both the figures demonstrate that as the value of $\lambda_0/\ \lambda_1$ is reduced, the rate at which the waiting time increases is sped up significantly. This is observed in Figure 5, where the topmost curve has a higher slope and with a lower value of λ_0, and in Figure 6, where the topmost curve has the highest slope but has the lowest value of λ_1, which is used to calculate for that particular graph.

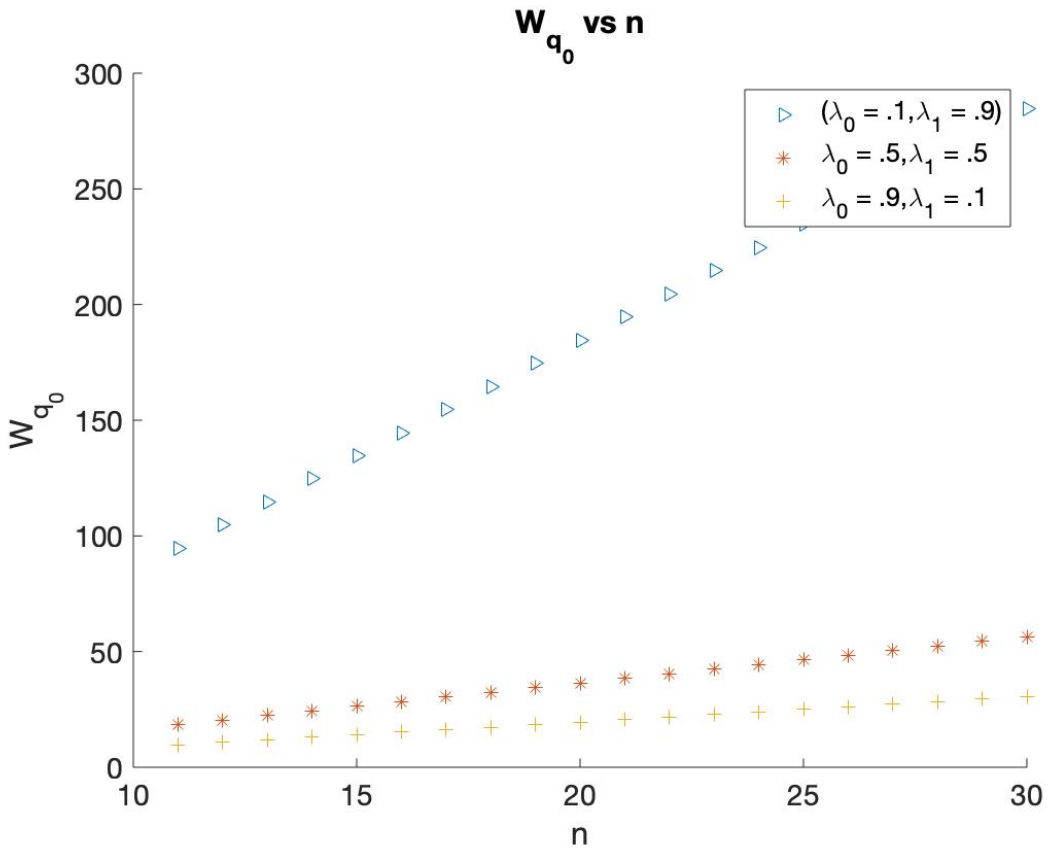


Figure 5. W_{Q_0} vs n

$$W_{Q_0} = \frac{L_{Q_0}}{\lambda_0}$$

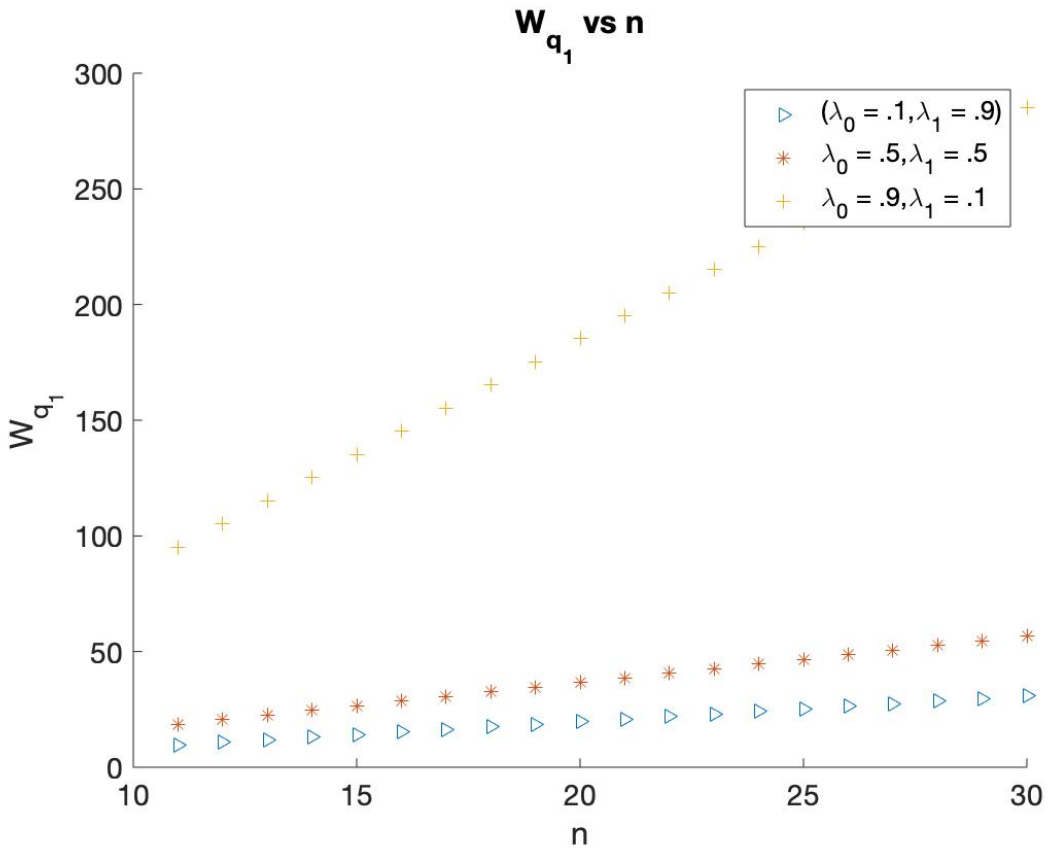


Figure 6. W_{Q_1} vs n

$$W_{Q_1} = \frac{L_{Q_1}}{\lambda_1}$$

Figures 7 and 8 demonstrate the waiting time of the pending transactions (in terms of slots) in the transactions pool for the mining selection for the block in both permission-less and permissioned chains. The waiting time in Figure 7 is observed to be the summation of W_{Q_0} with the reciprocal of μ_0, and likewise, the waiting time observed in Figure 8 is the summation of W_{Q_1} and the reciprocal of μ_1. Thereby, the following graphs of W_0 and W_1 have a slight increase in values compared to W_{Q_0} and W_{Q_1}.

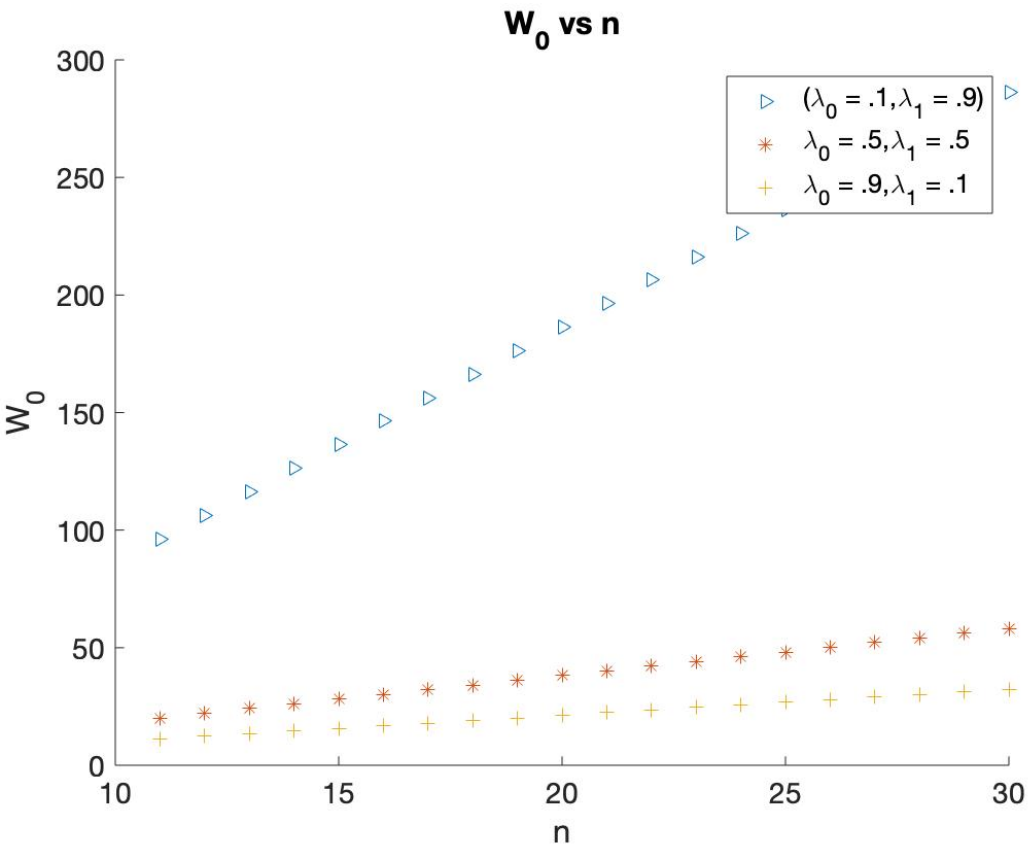


Figure 7. W_0 vs n

$$W_0 = W_{Q_0} + \frac{1}{\mu_0}$$

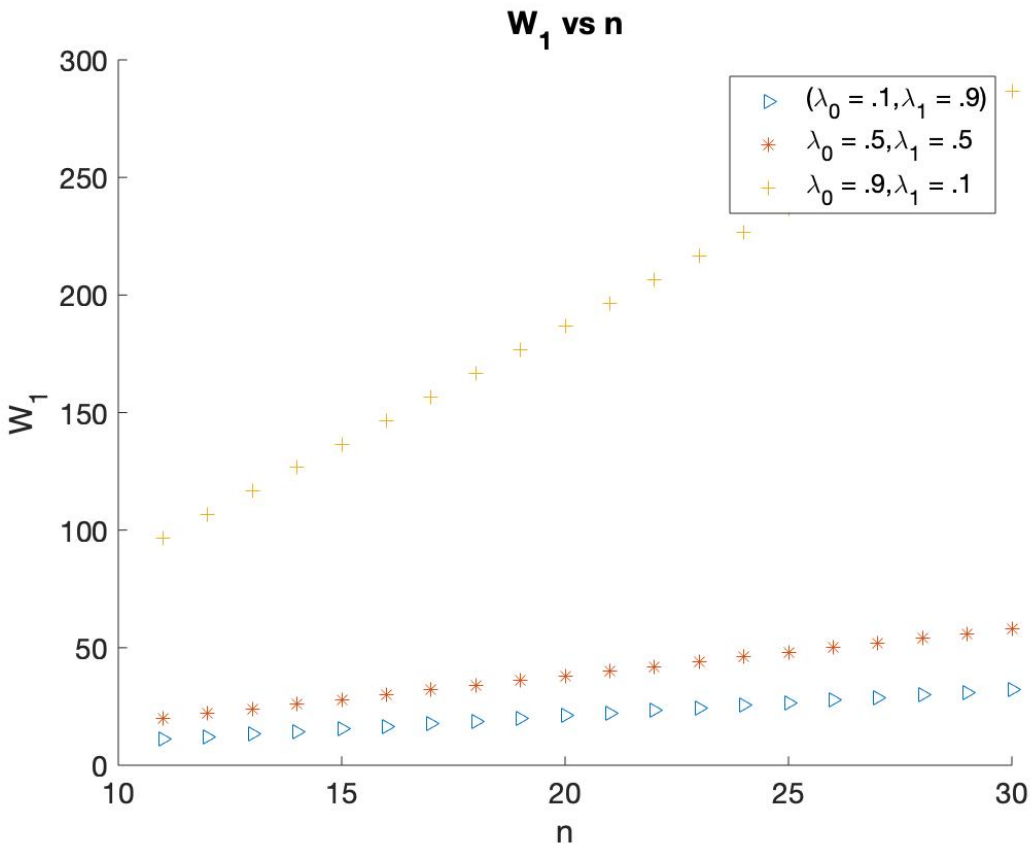


Figure 8. W_1 vs n

$$W_1 = W_{Q_1} + \frac{1}{\mu_1}$$

Another model related to the hybrid chain is proposed which focuses on the dependability of the state the transaction is in. The hybrid chain proposed in this research is an attempt to model and design a dependable interface between heterogeneous chains, in particular, between the private and the non-private chains such as between Hyperledger and Ethereum main net. The dependability in the hybrid chain is primarily determined by the likelihood for a transaction declared to be private to stay private, or vice versa. A set of random variables (e.g., the rate for a transaction to be declared private or public, respectively; the rate for a transaction, no matter which state it is currently in, to switch its state to private or public and stays in the same state in a

dependable manner, respectively; and the rate for a transaction, no matter which state it is currently in, to switch its state to private or public yet switches to the other state in an undependable manner, respectively) are identified in order to track the state transitions of the transaction with respect to the dependability. The ultimate goal of the proposed hybrid chain is to establish a theoretical foundation to realize a dependable interoperability across heterogeneous blockchains and thereby realizing a seamless inter-chain and multi-chain synergy. Another proposed hybrid chain model consisting of the states expressed from the standpoint of a transaction's life cycle in steady state. Note that the dependability in the following context is defined by the state of a transaction 100% reliable and secured as desired and defined, or undependable otherwise.

P_{vv} : the state in which a transaction (or a sequence of transactions as part of a batch of computations to be carried out as a whole, and of which an instance of a transaction of concern at an instant of time) is declared private (v) and stays private (v) in a dependable manner. Notice that implicitly this state tracks the dependability of a transaction along with P_{bb} likewise, and the undependability tracked by P_{vb} and P_{bv} to be described in the following. Therefore, the dependability in this research concerns across private (e.g., Hyperledger) and public (non-private, e.g., Ethereum), and is the probability for an instance of transaction at an instant of time to be declared (i.e., either code or data or both) private (or public) and to stay private (or public), respectively.

P_{vb} : the state in which a transaction is declared private (v) but switches public (b) in an undependable manner. The undependability in this research also concerns across private (e.g., Hyperledger) and public (non-private, e.g., Ethereum), and is the probability for an instance of transaction at an instant of time to be declared (i.e., either code or data or both) private (or public) and to stay private (or public), respectively. The private chain is managed with permission in general and the public chain without, and hence the undependability of concern, in other words, is

that a transaction that is supposed to be accessed and executed with permission, has got compromised or failed functionally ending up being accessed and executed without permission, or vice versa.

P_{bv} : the state in which a transaction is declared public (b) and switches private (v) in an undependable manner. In practice, P_{vb} might seem to be more of devastation than P_{bv}, however, in fact, P_{bv} could also be a devastation such that any code or data that were supposed to be accessed and executed without a permission unexpectedly and at the worst time, has got blocked requiring a permission due to any compromise or functional failure.

P_{bb} : the state in which a transaction is declared public (b) and stays public (b) in a dependable manner. Note that across the border between the private and the public, an interface needs to be placed in order to provide a seamless context switching back and forth from private to public or vice versa such that a private transaction supposed to switch its declaration to public prior to entering the public domain, or vice versa, in order to ensure the dependability, which is clearly distinguished from the context switching due to any undependable causes.

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

q_v, q_b: the rate for a transaction to be in a private state (v) and public (b) at an instant of time, respectively. Note that these random variables can take an effect no matter which state the transaction is currently in, instead, what matters is which state the transaction currently is attempting to switch into.

q_{vv}: the rate for a transaction to maintain its state within private (vv). This is a random variable required for a dependable state transition from the private state and stay in the private state.

q_{bb}: the rate for a transaction to stay its state within public (bb). This is a random variable required for a dependable state transition from the public state and stay in the public state.

q_{vb}: the rate for a transaction to switch its state from private to public (vb). This is a random variable required for an undependable state transition from the private state to the public state.

q_{bv}: the rate for a transaction to switch its state from public to private (bv). This is a random variable required for an undependable state transition from the public state to the private state.

The state transition probabilities (i.e., $t(P_i, P_j)$: state transition probability from P_i to P_j) are defined as follows.

$t(P_{vv}, P_{vv}) = q_v q_{vv}$, this state transition tracks and implies the sustainability of the dependability of a private-transaction by staying in the private state such that if a transaction is currently dependable in the private state (i.e., at the rate of q_v) and it continues to stay in the private state (i.e., at the rate of q_{vv}), which is itself considered to be the most critical requirement of dependability in the specific context of the hybrid chain in this research; then it contributes to the steady state probability of P_{vv} in a positive manner. Note that q_v and q_{vv} are assumed to be two independent random variables.

$t(P_{vv}, P_{vb}) = q_v q_{vb}$, this state transition tracks and implies the rate to switch the state of a transaction from the private state to the public state in an undependable manner such that if a transaction is currently dependable in the private state (i.e., at the rate of q_v) yet it switches its state to the public (i.e., at the rate of q_{vb}) which is not the state transition as commanded, in other words, in an undependable manner; then it contributes to the steady state probability of P_{vb} in a positive manner while to P_{vv} in a negative manner. Note that q_v and q_{vb} are assumed to be two independent random variables.

$t(P_{vv}, P_{bv}) = q_b q_{bv}$, this state transition tracks and implies the rate to switch state of a transaction from the public state to the private state in an undependable manner such that if a transaction is currently dependable in the public state (i.e., at the rate of q_b) yet it switches its

state to the private (i.e., at the rate of q_{bv}) which is not the state transition as commanded, in other words, in an undependable manner; then it contributes to the steady state probability of P_{bv} in a positive manner while to P_{vv} in a negative manner. Note that q_b and q_{bv} are assumed to be two independent random variables.

$t(P_{vv}, P_{bb}) = q_b q_{bb}$, this state transition tracks and implies the rate to switch the state of a transaction from the private state to the public state in a dependable manner (note that in fact, it does not matter which state the transaction is in but which state the transaction is attempting to be in) such that a transaction was dependable in the private state and then commanded to switch its state to the public no matter which state the transaction was in (i.e., at the rate of q_v or q_b, and q_b in this particular state transition) and it continues to stay in the public state (i.e., at the rate of q_{bb}); then it contributes to the steady state probability of P_{bb} in a positive manner while to P_{vv} in a negative manner. Note that q_b and q_{bb} are assumed to be two independent random variables.

$t(P_{bv}, P_{vv}) = q_v q_{vv}$, this state transition tracks and implies the rate to switch the state of a transaction from an undependable private state (e.g., P_{bv} in this particular state transition) to the dependable private state (e.g., P_{vv} in this particular state transition) such that a transaction was in an undependable private state and then commanded to switch its state to the dependable private state no matter which state the transaction was in (i.e., at the rate of q_v) and it continues to stay in the private state (i.e., at the rate of q_{vv}); then it contributes to the steady state probability of P_{vv} in a positive manner while to P_{bv} in a negative manner. Note that q_v and q_{vv} are assumed to be two independent random variables.

$t(P_{bv}, P_{vb}) = q_v q_{vb}$, this state transition tracks and implies the rate to switch the state of a transaction from an undependable private state (e.g., P_{bv} in this particular state transition) to the undependable public state (e.g., P_{vb} in this particular state transition) such that a transaction was in an undependable private state and then commanded to switch its state to the private state (i.e.,

at the rate of q_v) yet it switches to the public state in an undependable manner (i.e., at the rate of q_{vb}); then it contributes to the steady state probability of P_{vb} in a positive manner while to P_{bv} in a negative manner. Note that q_v and q_{vv} are assumed to be two independent random variables.

$t(P_{bv}, P_{bv}) = q_b q_{bv}$, this state transition tracks and implies the rate for a transaction to stay in an undependable private state (e.g., P_{bv} in this particular state transition) such that a transaction was commanded to be in the public state (i.e., at the rate of q_b) yet it switches to the private state in an undependable manner (i.e., at the rate of q_{bv}); then it contributes to the steady state probability of P_{bv} in a positive manner in a self-transition. Note that q_b and q_{bv} are assumed to be two independent random variables.

$t(P_{bv}, P_{bb}) = q_b q_{bb}$, this state transition tracks and implies the rate to switch the state of a transaction from an undependable private state (e.g., P_{bv} in this particular state transition) to the dependable public state (e.g., P_{bb} in this particular state transition) such that a transaction was in an undependable private state and then commanded to switch its state to the dependable private state (i.e., at the rate of q_b) and it continues to stay in the public state (i.e., at the rate of q_{bb}); then it contributes to the steady state probability of P_{bb} in a positive manner while to P_{bv} in a negative manner. Note that q_b and q_{bb} are assumed to be two independent random variables.

$t(P_{vb}, P_{vv}) = q_v q_{vv}$, this state transition tracks and implies the rate to switch the state of a transaction from an undependable public state (e.g., P_{vb} in this particular state transition) to the dependable private state (e.g., P_{vv} in this particular state transition) such that a transaction was in an undependable public state and then commanded to switch its state to the dependable private state (i.e., at the rate of q_v) and it continues to stay in the private state (i.e., at the rate of q_{vv}); then it contributes to the steady state probability of P_{vv} in a positive manner while to P_{vb} in a negative manner. Note that q_v and q_{vv} are assumed to be two independent random variables.

$t(P_{vb}, P_{vb}) = q_v q_{vb}$, this state transition tracks and implies the rate for a transaction to stay in an undependable public state (e.g., P_{vb} in this particular state transition) such that a transaction was commanded to be in the private state (i.e., at the rate of q_v) yet it switches to the public state in an undependable manner (i.e., at the rate of q_{vb}); then it contributes to the steady state probability of P_{vb} in a positive manner in a self-transition. Note that q_v and q_{vb} are assumed to be two independent random variables.

$t(P_{vb}, P_{bv}) = q_b q_{bv}$, this state transition tracks and implies the rate to switch the state of a transaction from an undependable public state (e.g., P_{vb} in this particular state transition) to the undependable private state (e.g., P_{bv} in this particular state transition) such that a transaction was in an undependable public state and then commanded to switch its state to the public state (i.e., at the rate of q_b) yet it switches to the private state in an undependable manner (i.e., at the rate of q_{bv}); then it contributes to the steady state probability of P_{bv} in a positive manner while to P_{vb} in a negative manner. Note that q_b and q_{bv} are assumed to be two independent random variables.

$t(P_{vb}, P_{bb}) = q_b q_{bb}$, this state transition tracks and implies the rate to switch the state of a transaction from an undependable public state (e.g., P_{vb} in this particular state transition) to the dependable public state (e.g., P_{bb} in this particular state transition) such that a transaction was in an undependable public state and then commanded to switch its state to the dependable public state (i.e., at the rate of q_b) and it continues to stay in the public state (i.e., at the rate of q_{bb}); then it contributes to the steady state probability of P_{bb} in a positive manner while to P_{vb} in a negative manner. Note that q_b and q_{bb} are assumed to be two independent random variables.

$t(P_{bb}, P_{vv}) = q_v q_{vv}$, this state transition tracks and implies the rate to switch the state of a transaction from the public state to the private state in a dependable manner such that a transaction was dependable in the public state and then commanded to switch its state to the private state (i.e., at the rate of q_v) and it continues to stay in the private state (i.e., at the rate of

q_{vv}); then it contributes to the steady state probability of P_{vv} in a positive manner while to P_{bb} in a negative manner. Note that q_v and q_{vv} are assumed to be two independent random variables.

$t(P_{bb}, P_{vb}) = q_v q_{vb}$, this state transition tracks and implies the rate to switch the state of a transaction from the dependable public state (e.g., P_{bb} in this particular state transition) to the public state in an undependable manner (e.g., P_{vb} in this particular state transition) such that a transaction was dependable in the public state and then commanded to switch its state to the private state (i.e., at the rate of q_v) yet it switches to the public state in an undependable manner (i.e., at the rate of q_{vb}); then it contributes to the steady state probability of P_{vb} in a positive manner while to P_{bb} in a negative manner. Note that q_v and q_{vb} are assumed to be two independent random variables.

$t(P_{bb}, P_{bv}) = q_b q_{bv}$, this state transition tracks and implies the rate to switch the state of a transaction from the dependable public state (e.g., P_{bb} in this particular state transition) to the private state in an undependable manner (e.g., P_{bv} in this particular state transition) such that a transaction was dependable in the public state and then commanded to stay its state in the public state (i.e., at the rate of q_b) yet it switches to the private state in an undependable manner (i.e., at the rate of q_{bv}); then it contributes to the steady state probability of P_{bv} in a positive manner while to P_{bb} in a negative manner. Note that q_b and q_{bv} are assumed to be two independent random variables.

$t(P_{bb}, P_{bb}) = q_b q_{bb}$, this state transition tracks and implies the sustainability of the dependability of a public-transaction by staying in the public state such that if a transaction is currently dependable in the public state (i.e., at the rate of q_b) and it continues to stay in the public state (i.e., at the rate of q_{bb}); then it contributes to the steady state probability of P_{bb} in a positive manner. Note that q_b and q_{bb} are assumed to be two independent random variables. The steady state equations for the slim chain model are as follows.

$$P_{vv}(q_v q_{vv} + q_v q_{vb} + q_b q_{bv} + q_b q_{bb}) = P_{vv} Q_{vv} = (P_{vb} + P_{bv} + P_{bb}) q_v q_{vv}$$

$$P_{vb}(q_v q_{vb} + q_v q_{vv} + q_b q_{bb} + q_b q_{bv}) = P_{vb} Q_{vb} = (P_{vv} + P_{bv} + P_{bb}) q_v q_{vb}$$

$$P_{bv}(q_b q_{bv} + q_v q_{vv} + q_v q_{vb} + q_b q_{bb}) = P_{bv} Q_{bv} = (P_{vv} + P_{vb} + P_{bb}) q_b q_{bv}$$

$$P_{bb}(q_b q_{bb} + q_v q_{vv} + q_v q_{vb} + q_b q_{bv}) = P_{bb} Q_{bb} = (P_{vv} + P_{vb} + P_{bv}) q_b q_{bb}$$

$$P_{vv} + P_{vb} + P_{bv} + P_{bb} = 1$$

The solutions for the above steady state equations for the slim chain model are as follows.

$$P_{vb} = \frac{1}{Q''' + 1 + Q'''Q'' + Q'' + Q''' + Q' + Q'} = Q''''$$

$$P_{vv} = Q''''Q'''$$

$$P_{bv} = Q''''Q'''Q'' + Q''''Q''$$

$$P_{bb} = Q''''Q'''Q' + Q''''Q'$$

$$Q''' = \frac{(1 + Q'' + Q')q_v q_{vv}}{Q_{vv} - Q'' q_v q_{vv} - Q' q_v q_{vv}}$$

$$Q'' = \frac{q_b q_{bv}}{Q_{bv}} + Q'$$

$$Q' = \frac{\left(1 + \frac{q_b q_{bv}}{Q_{bv}}\right)\left(\frac{q_b q_{bb}}{Q_{bb}}\right)}{1 - \frac{q_b q_{bv}}{Q_{bv}} \frac{q_b q_{bb}}{Q_{bb}}}$$

Based on the solutions of the proposed hybrid chain model, the following four graphs are plotted in order to demonstrate the expected dependability [4] (i.e., P_{vv}) versus various yet primary variables such as q_v and q_{vv}; and another expected dependability (i.e., P_{bb}) versus the variables

such as q_b and q_{bb}. Note that, in fact, the overall dependability of interest is the sum of P_{vv} and P_{bb} in the proposed hybrid model.

The graphs on P_{vv}versus q_v shown in Figure 9 are plotted in order to demonstrate the effect of q_v of a transaction on the steady state probability for the transaction to be in state P_{vv} at high or medium (e.g., .9 or .5, respectively) of q_b, at high or medium (e.g., .9 or .5, respectively) of q_{vv} and q_{bb}, and at medium or low (e.g., .5 or .1, respectively) of q_{vb} and q_{bv}, where the plots with $q_b = .9, q_{vv}, q_{bb} = .9,\ q_{vb}, q_{bv} = .1\ (i.e.,\ (.9, .9, .1)), q_b = .9, q_{vv}, q_{bb} = .5,\ q_{vb}, q_{bv} = .5\ (i.e.,\ (.9, .5, .5)), q_b = .5, q_{vv}, q_{bb} = .9,\ q_{vb}, q_{bv} = .1\ \ (i.e., (.5, .9, .1)),\ and\ q_b = .5,\ q_{vv}, q_{bb} = .5,\ q_{vb}, q_{bv} = .5\ (i.e.,\ (.5, .5, .5))$, reveal that P_{vv} increases as q_v increases across any combination of values of other variables, i.e., $q_b, q_{vv}, q_{bb}, q_{vb}, q_{bv}$ as expected; in comparison between (.9,.9,.1) and (.5, .9, .1), dropping $q_b$from high (.9) to medium (.5) did not degrade P_{vv} as much throughout the entire range of q_v, and further notice that in the range of low q_v (e.g., .0 to .2), (.9,.9,.1) even outperformed (.5, .9, .1) under the proposed hybrid chain model; likewise, in comparison between $(.9, .5, .5)$ and $(.5, .5, .5)$, dropping $q_b$from high (.9) to medium (.5) did even improve P_{vv} slightly throughout the entire range of q_v except when $(.9, .5, .5)$ and $(.5, .5, .5)$ briefly glide closely on each other without reversing their P_{vv} values; also, it has to be admitted that the spike ups of (.5, .9, .1) and $(.5, .5, .5)$ are not expected and will be looked into for verification both theoretically and experimentally;

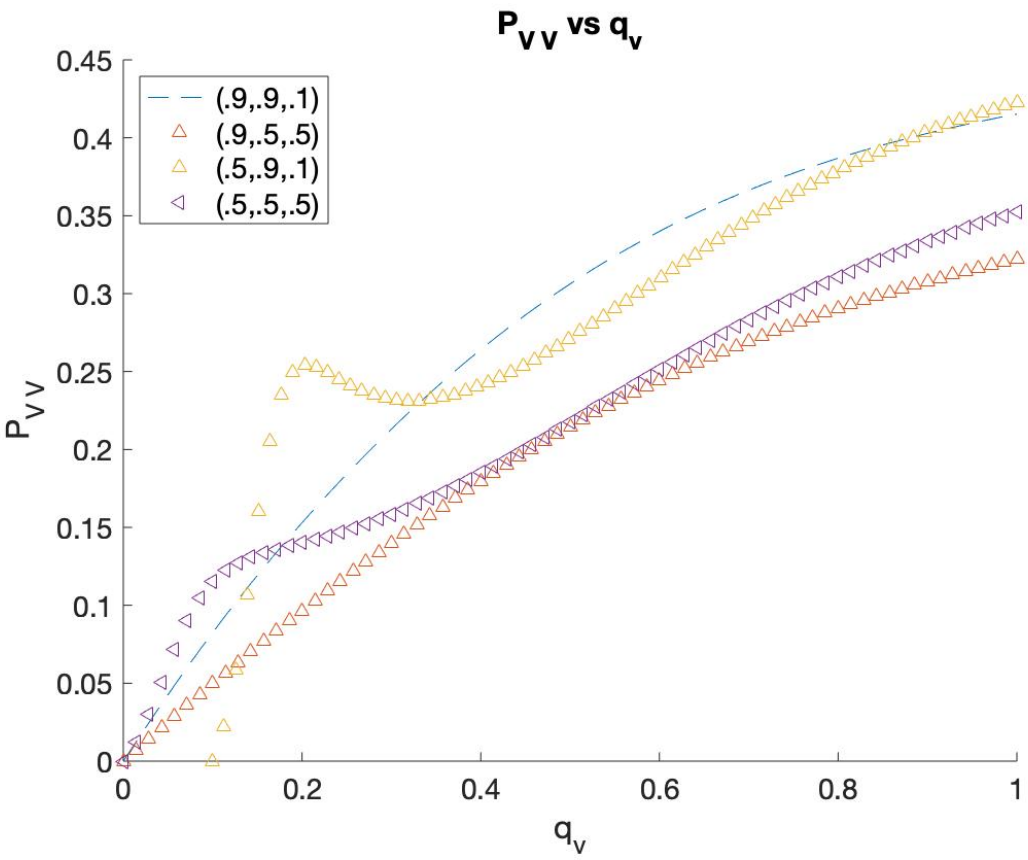


Figure 9. P_{vv}versus q_v

P_{vv}versus q_{vv} below are plotted in order to demonstrate the effect of q_v of a transaction on the steady state probability for the transaction to be in state P_{vv} at high or medium (e.g., .9 or .5, respectively) of q_v and q_b, at high or medium (e.g., .9 or .5, respectively) of q_{bb}, and at medium or low (e.g., .5 or .1, respectively) of q_{vb} and q_{bv}, where the plots with $q_v, q_b = .9, q_{bb} = .9,$ $q_{vb}, q_{bv} = .1\ (i.e.,\ (.9,.9,.1)), q_v, q_b = .9, q_{bb} = .5,\ q_{vb}, q_{bv} = .5\ (i.e.,\ (.9,.5,.5)), q_v, q_b =$ $.5, q_{bb} = .9,\ q_{vb}, q_{bv} = .1\ \ (i.e.,(.5,.9,.1)),\ and\ q_v, q_b = .5,\ \ q_{bb} = .5,\ q_{vb}, q_{bv} =$ $.5\ (i.e.,\ (.5,.5,.5)),$ reveal that P_{vv} increases as q_{vv} increases across any combination of values of other variables, i.e., $q_v, q_b, q_{bb}, q_{vb}, q_{bv}$ as expected; in comparison between (.9,.9,.1) and (.5, .9, .1), dropping q_v, q_b from high (.9) to medium (.5) did degrade P_{vv} as expected throughout the entire range of q_{vv} and widened towards 1.0; likewise, in comparison between $(.9,.5,.5)$ and

$(.5, .5, .5)$, dropping q_v, q_b from high (.9) to medium (.5) did degrade P_{vv} as expected throughout the entire range of q_{vv} and widened towards .5 then attempting to narrow back towards 1.0;

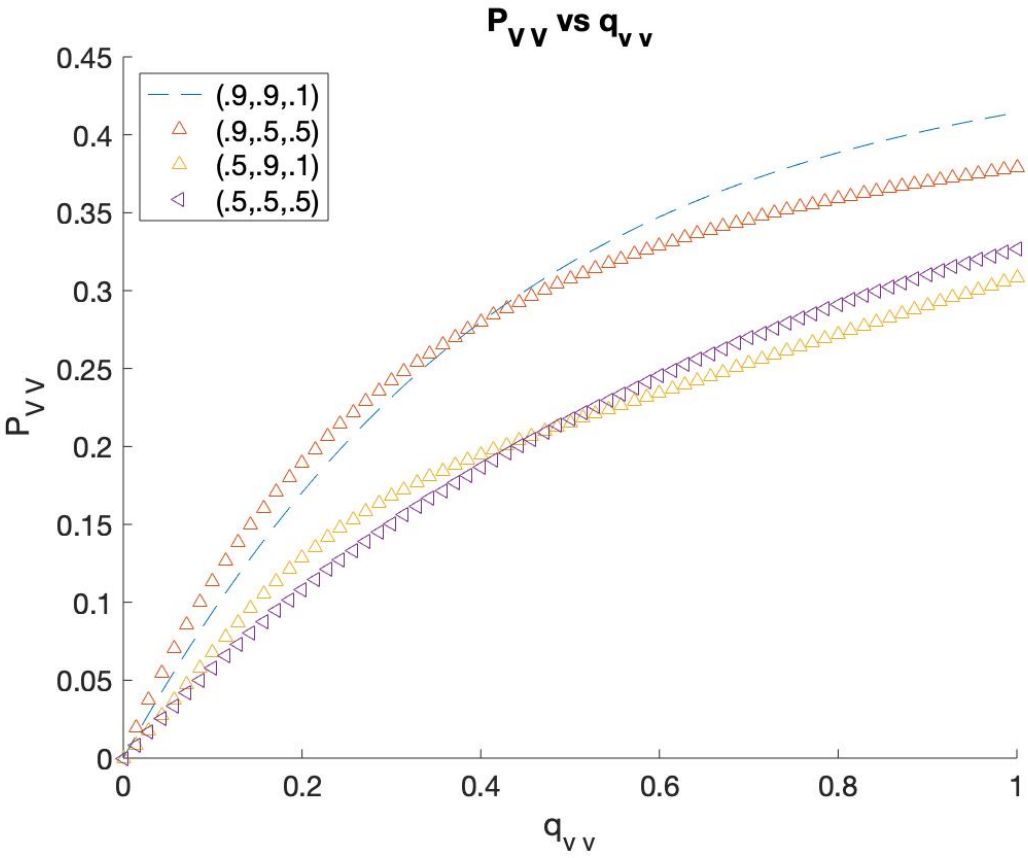


Figure 10. P_{vv}versus q_{vv}

P_{bb}versus q_b below are plotted in order to demonstrate the effect of q_v of a transaction on the steady state probability for the transaction to be in state P_{bb} at high or medium (e.g., .9 or .5, respectively) of q_v, at high or medium (e.g., .9 or .5, respectively) of q_{vv} and q_{bb}, and at medium or low (e.g., .5 or .1, respectively) of q_{vb} and q_{bv}, where the plots with $q_v = .9, q_{vv}, q_{bb} = .9$, $q_{vb}, q_{bv} = .1\ (i.e.,\ (.9, .9, .1)), q_v = .9, q_{vv}, q_{bb} = .5,\ q_{vb}, q_{bv} = .5\ (i.e.,\ (.9, .5, .5)), q_v =$ $.5, q_{vv}, q_{bb} = .9,\ q_{vb}, q_{bv} = .1\ \ (i.e., (.5, .9, .1)),\ and\ q_v = .5,\ \ q_{vv}, q_{bb} = .5,\ q_{vb}, q_{bv} =$ $.5\ (i.e.,\ (.5, .5, .5))$, reveal that reveal that P_{bb} increases as q_b increases in the early or low range (up to .1 to .2) then turns downward throughout across any combination of values of other

variables, i.e., $q_b, q_{vv}, q_{bb}, q_{vb}, q_{bv}$, and those concavities will be looked into for verification both theoretically and experimentally since they appear to be scientifically significant potentially to provide a foundation for a theoretical optimality of the proposed hybrid chain; in comparison between (.9,.9,.1) and (.5, .9, .1), dropping q_b from high (.9) to medium (.5) did never degrade P_{bb} throughout the entire range of q_b, instead, P_{bb} was improved and widening towards around .2 then narrows back throughout towards 1.0., which will trigger a serious investigation both theoretically and experimentally to explain this pattern of dependability behavior of the proposed hybrid chain; likewise, in comparison between (.9,.5,.5) and (.5, .5, .5), similar pattern of P_{bb} with respect to q_b was observed with the concavity formed around the slightly earlier or lower q_b;

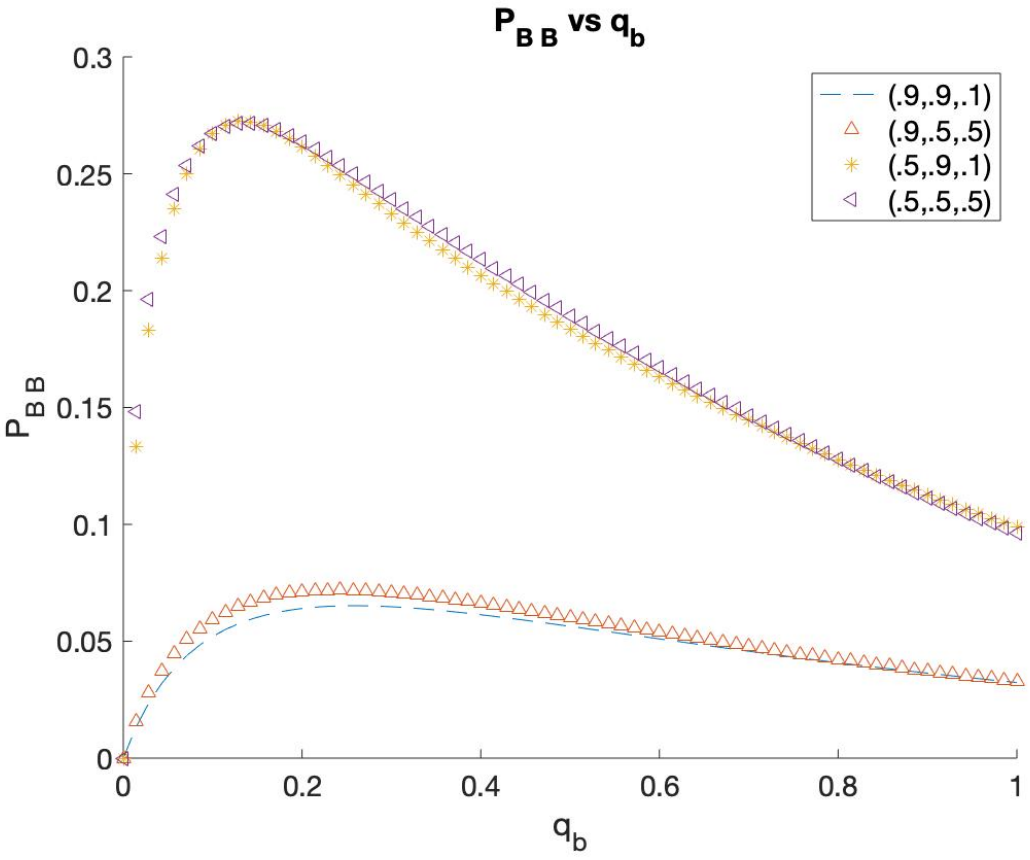


Figure 11. P_{bb} versus q_b

and P_{vv}versus q_{bb} below are plotted in order to demonstrate the effect of q_v of a transaction on the steady state probability for the transaction to be in state P_{vv} at high or medium (e.g., .9 or .5, respectively) of q_v and q_b, at high or medium (e.g., .9 or .5, respectively) of q_{vv}, and at medium or low (e.g., .5 or .1, respectively) of q_{vb} and q_{bv}, where the plots with $q_v, q_b = .9, q_{vv} = .9, q_{vb}, q_{bv} = .1\ (i.e.,\ (.9,.9,.1)), q_v, q_b = .9, q_{vv} = .5,\ q_{vb}, q_{bv} = .5\ (i.e.,\ (.9,.5,.5)), q_v, q_b = .5, q_{vv} = .9,\ q_{vb}, q_{bv} = .1\ \ (i.e.,(.5,.9,.1)),\ and\ q_v, q_b = .5,\ \ q_{vv} = .5,\ q_{vb}, q_{bv} = .5\ (i.e.,\ (.5,.5,.5))$, reveal that reveal that reveal that P_{bb} increases as q_{bb} increases in the early or low range (up to .2 to .3) then turns downward throughout across any combination of values of other variables, i.e., $q_v, q_b, q_{vv}, q_{vb}, q_{bv}$, and those concavities will be looked into for verification as well both theoretically and experimentally; in comparison between (.9,.9,.1) and (.5, .9, .1), dropping $q_v, q_b$from high (.9) to medium (.5) did in fact improved P_{bb} in a significant manner throughout the entire range of q_{bb} finding the widest gap in between at around .3; likewise, in comparison between (.9,.5,.5) and (.5, .5, .5), similar pattern of P_{bb} with respect to q_{bb} was observed with the concavity formed around the slightly delayed or higher q_{bb}.

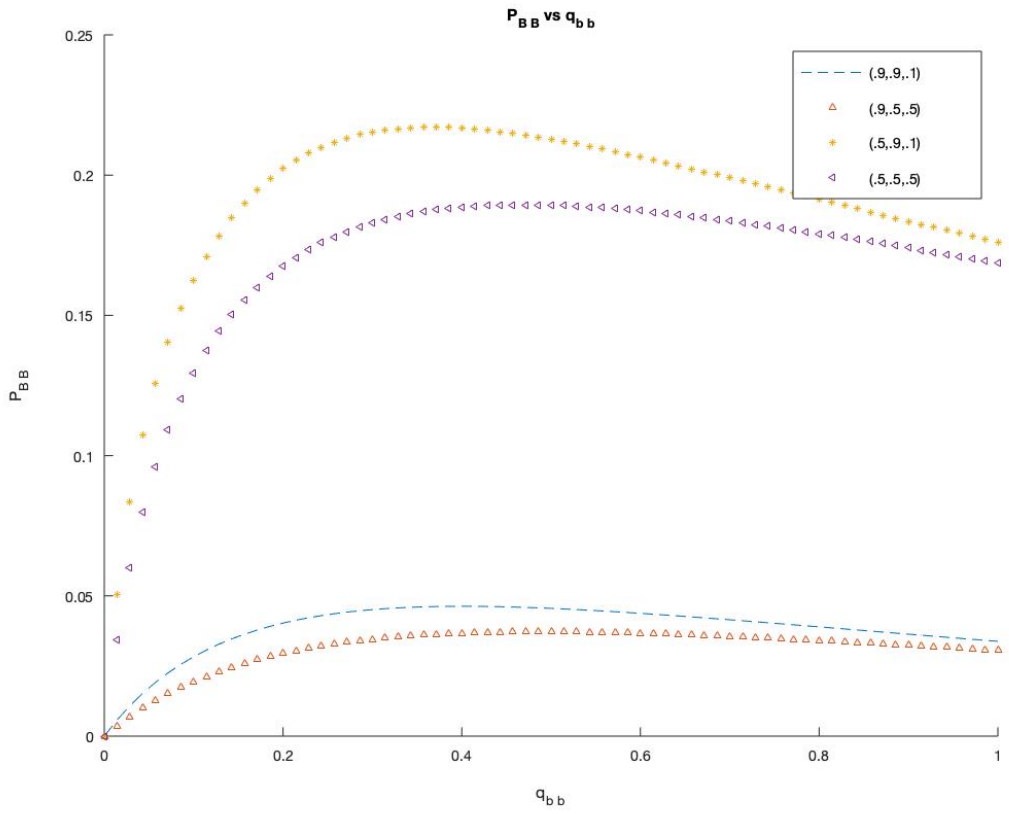


Figure 12. P_{vv}versus q_{bb}

The hybrid chain is being proposed with a similar architecture compared to that of slim chain. However, there will be two major differences between them such that IPFS in slim chain will be replaced with other blockchains that will have compatibility with the current blockchain (e.g., a paired hybrid chains of private and main nets); the transaction to be added to the blockchain will be holding a special string variable to determine which blockchain the transaction is being added; and if any errors are encountered while adding to the secondary blockchain, the transaction will stay on the secondary blockchain rather than coming back to the primary blockchain unlike the slim chain, where the transaction from IPFS will be returned back to the blockchain in case of any error.

IMPLEMENTATION

This section covers the basic control flow of the proposed hybrid mode and Figure 9 shows the detailed flowchart of the proposed hybrid model that was put into simulation using Ethereum to replicate a permission-less blockchain and Hyperledger to replicate for permissioned blockchain. Each major change in the control flow is numbered and described in detail.

1. The transaction to be added to the blockchain, the source of this transaction is either from a mobile application / HTML request from the user. This section is also responsible to determined which blockchain the transaction will be added to, public or permissioned.
2. After it is determined to be added to the secondary blockchain like, Ethereum or Bitcoin, the transaction details are sent to the appropriate blockchain for further processing.
3. If the user that has initiated the transaction does not have a public ID in the secondary blockchain, a new one is created on his behalf.
4. The transaction details are added to the secondary blockchain (mostly monetary transactions).
5. Any errors on processing this transaction are handled internally in the secondary blockchains, without the primary blockchain waiting for an acknowledgement.
6. Upon successful transaction posting in the secondary blockchain, the hashcode is returned back to the sever, which then also adds the hashcode to the appropriate user that initiated the transaction.
7. Hash code returned from the secondary blockchain is received at this time of the flow.
8. The hash code is then added to the primary blockchain also to keep a track for later purposes.
9. The transaction (along with the hashcode from the secondary blockchain) is then added to the primary blockchain.

10. If the transaction (mostly a user modification) is intended to be posted to primary blockchain (permissioned), then the transaction is directly placed in the permissioned given that the transactions satisfies all for its validity.

A crypto payment system is developed as a testbed to demonstrate the effectiveness and efficiency of the proposed hybrid chain such that a basic and simple demo has been built across both HTML and mobile (Android) application, wherein the user has the choice to make a transaction. The transaction is then moved to the primary or secondary blockchain depending on the type of transaction initiated by the user. The developed front-end application consisted of:

1. Making a payment in terms of Ether (crypto currency for Ethereum), resulting in a transaction to be initiated in Ethereum (public/secondary blockchain).
2. Modifying user personal information, which would have a transaction recorded in the primary (permissioned) blockchain upon successful request.

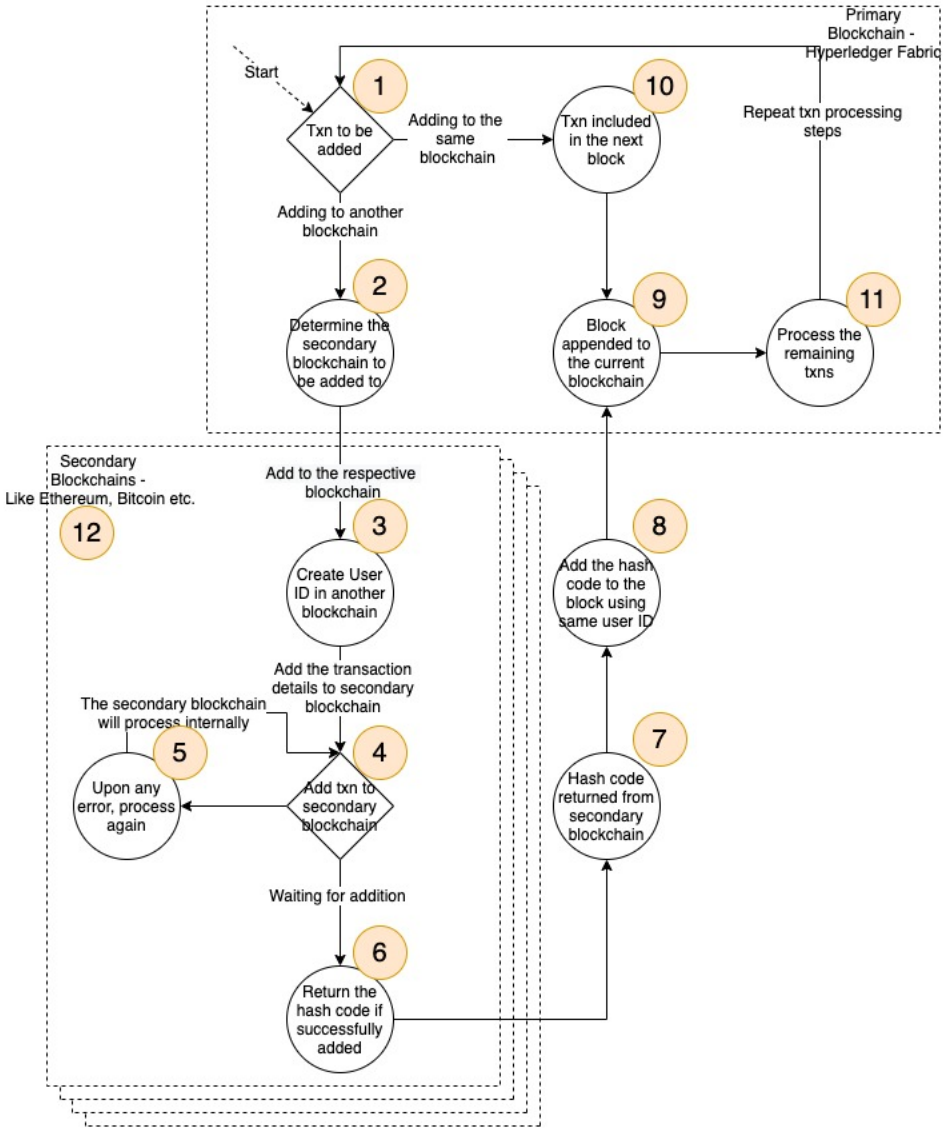


Figure 13. Hybrid chain flowchart

The following is a practical implementation of the hybrid chain using Ethereum for public blockchain and Hyperledger for permissioned blockchain:

Figure 14 describes the details of the HTML page that the user will be interacting to make a payment or modify information.

Figure 14. HTML enabled page to demonstrate a payment

Figure 15 details the backend information after the user as made a payment request from the HTML page, while Figure 16 shows the user information stored in Hyperledger's backend.

Figure 15. Details of the user logged in backend

Figure 16. User information in Permissioned chain

Figure 17. Payment recorded by user in backend

Figure 18. The payment transaction in Ethereum console

Figures 17 and 18 show the transaction details of the user payment recorded in the backend as well as in Ethereum. The amount of .8741Ether is posted from HTML through the user account and the same amount is reflected in the transaction value as well. The ether is transferred to the smart contract in Ethereum rather than to another public address.

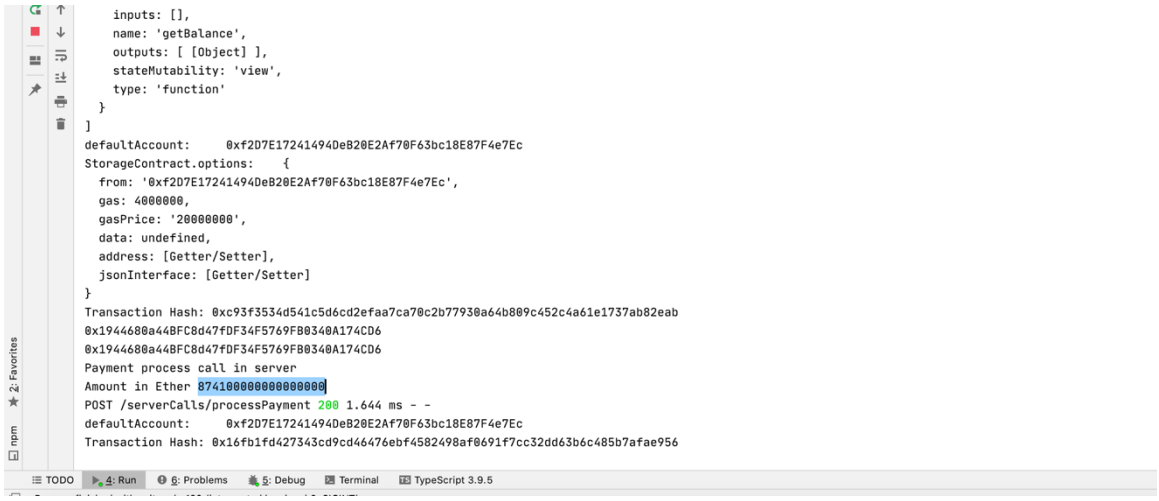


Figure 19. The same amount recorded in backend

The payment amount captured in the backend of the server and shown on the HTML page and the one posted on Ethereum are equal and is shown in Figure 19.

CONCLUSION

The proposed research has established the capability to balance the load of computation across on- and off-chains in order to take advantage of the synergistic cost and performance benefits from both on- and off-chains in a dependable manner along with the proposed dependability monitoring system based on the novel checkpoint-and-rollback protocol in order to coordinate the, otherwise, intrinsically centralized off-chain transactions of concern in a decentralized manner mitigating the potential loss of dependability otherwise; a dependable crypto computing capability under stringent real-time requirement in order to address and resolve the speed issue of the otherwise saturating blockchain performance today or in the near future; a slim chain in order to address and resolve the scalability issue of blockchain-based crypto computing; and a hybrid chain in order to address and resolve a need for transactions across private and main nets in particular; and various rigorous yet novel dependability models have been built and demonstrated to establish a sound theoretical foundation ultimately for optimal, dependable and high performance crypto computing.

A dependability model for each proposed crypto solution has been identified and defined along with various performance variables, and has ultimately provided a theoretical yet practical understanding of each crypto solution. A practical solution has been developed as well for each of the four different crypto solutions using Ethereum and Hyperledger as two of the bases for public and permissioned blockchains. A theoretical and a practical model has been developed to evaluate the dependability of the crypto system in a quantitative manner with respect to the proposed checkpoint and rollback algorithm, slim chain, real-time and hybrid-chain models. This research has presented a work on how to assure the dependability of a crypto system built across on and off the blockchain by using the proposed checkpoint and rollback algorithm in an adaptive manner, proposed an implementation of a new storage mechanism in blockchain facilitated by IPFS known as slim-chain, has proposed an analytical approach how to design and realize a crypto computing (Ethereum blockchain-based) under stringent real-time requirement and proposed a hybrid chain based on the one as proposed in [34], and a variable bulk arrival and static bulk service (VBASBS) of double-tuple queueing model also has been developed to provide a quantitative tool to measure its performance as the dependability model.

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System.

[2] Chris Dannen. Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners

[3] Thomas Renner, Johannes Muller, Odej Kao, Endolith. A Blockchain based Framework to Enhance Data Retention in Cloud Storages 26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (2018).

[4] Diego Vegros, Jaime Saenz. Peer-To-Peer Networks and Internet Policies New York: Nova Science Publishers, Inc., 2010Author, F.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 12. Publisher, Location (2010)

[5] Uwe M. Borghoff Catalogue of Distributed File/Operating Systems Berlin Heidelberg: Springer-Verlag, (1992)

[6] Vitalik Buterin. A Next Generation Smart Contract & Decentralized Application Platform, Ethereum White Paper

[7] Israel Koren and C. Mani Krishna Fault-Tolerant Systems. San Francisco, CA: Morgan Kaufmann Publishers, (2007)

[8] Pouria Pirzadeh, Michael Carey, Till Westmann, A performance study of big data analytics platforms IEEE International Conference on Big Data, 2017

[9] Mrunal Sogodekar, Shikha Pandey, Isha Tupkari, Amit Manekar Big data analytics: hadoop and tools IEEE Bombay Section Symposium, 2016

[10] Sheldon M. Ross Introduction to Probability Models 11 Edition (Elsevier, Academic Press, Taiwan, 2014).

[11] Barry W. Johnson. Design and Analysis of Fault Tolerant Digital Systems (Addison-Wesley Publishing Company, 1986

[12] Morgen E. Peck Blockchains: How They Work and Why Theyll Change the World

[13] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler The Hadoop Distributed File System

[14] H. Subramanian Decentralized Blockchain-Based Electronic Marketplaces

[15] Dejan Vujii, Dijana Jagodi, Sinia Rani, Blockchain technology, bitcoin, and Ethereum: A brief overview

[16] Sara Rouhani, Ralph Deters, "Performance Analysis of Ethereum Transactions in private blockchain" 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)

[17] Mrs. Anamika Chauhan , Om Prakash Malviya , Madhav Verma, Tejinder Singh Mor, "Blockchain and Scalability", 2018 IEEE International Conference on Software Quality, Reliability and Security Companion

[18] Hiroki Kuzuno, Christian Karam "Blockchain explorer: An analytical process and investigation environment for bitcoin" 2017 APWG Symposium on Electronic Crime Research (eCrime).

[19] Tiana Laurence, Blockchain For Dummies-For Dummies

[20] Ethereum: A Super Decentralized Generalized transaction ledger, http://gavwood.com/paper.pdf

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

[22] Morgen E. Peck (September 2017) Blockchains: How They Work and Why They'll Change the World. Last accessed: March 2019. [Online].

[23] Abhilash Kancharla, Jongho Seol, Nicole Park, Indy Park, Nohpill Park; Dependable Industrial Crypto Computing, 28th International Symposium on Industrial Electronics (IEEE-ISIE 2019) during June 11-14, 2019 in Vancouver, Canada

[24] Marcello Cinque, Christian Esposito “How to Assess the Dependability of Applications on Top of the Blockchain: Novel Research Challenges” 2018 14th European Dependable Computing Conference (EDCC)

[25] F. Lombardi, L. Aniello, S. De Angelis, A. Margheri, V. Sassone” A Blockchain-based Infrastructure for Reliable and Cost-effective IoT-aided Smart Grids

[26] Qi Zhang, Petr Novotny, Salman Baset, Donna Dillenberger, Artem Barger, Yacov Manevich “LedgerGuard: Improving Blockchain Ledger Dependability” https://www.researchgate.net/publication/324939637_LedgerGuard_Improving_Blockchain_Ledger_DependabilityLast Accessed: April 2019 [Online]

[27] Jongho Seol, Abhilash Kancharla, Nicole Park, Nohpill Park, 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.

[28] How to Check the Reliability of a Blockchain Project [Online] Last Accessed: March 2019 https://coinjournal.net/sponsored-story-how-to-check-the-reliability-of-a-blockchain-project/

[29] Abhilash Kancharla, Nohpill Park; A Realtime Crypto Computing and Block-Dependability, IEEE SC2 2019 - Kaohsiung, Taiwan, Nov. 18-21, 2019, The 9th IEEE International Symposium on Cloud and Service Computing, Kaohsiung, Taiwan, Nov. 18-21, 2019

[30] Abhilash Kancharla, Seol Jongho, Nohpill Park, Hyeyoung Kim; Slim Chain and Dependability, The 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI 2020)

[31] Abhilash Kancharla, Nohpill Park, Zuqiang Ke, Hyeyoung Kim; Hybrid Chain and Dependability, The 2nd ACM International Symposium on Blockchain andSecure Critical Infrastructure (BSCI 2020)

[32] Seol Jongho, Abhilash Kancharla, Nohpill Park, Hyeyoung Kim, Zuqiang Ke; A Variable Bulk Arrival and Static Bulk Service Queueing Model for Blockchain, The 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI 2020)

[33] Abhilash Kancharla, Nohpill Park, Indy Park; Transaction Sampling Algorithms for Realtime Crypto Block Dependability, International Journal of Big Data and Intelligence

[34] Abhilash Kancharla, Nohpill Park, Hyeyoung Kim; A Realtime Chain and Variable Bulk Arrival and Variable Bulk Service (VBAVBS) Model with λ_F, MDPI, Applied Science, Electrical, Electronics and Communications Engineering

APPENDIX A

The pseudo code used for checkpoint and rollback algorithm is as below:

```
Initialize current_checkpoint_interval;

while(1){

      if((new_checkpoint_t - last_checkpoint_t) <=
      current_checkpoint_interval){

            current_checkpoint_interval = new_checkpoint_t -
      last_checkpoint_t;

      }

      else{

            if(dependable_count >= threshold_count))

                  ++current_checkpoint_interval;

      }

}
```

APPENDIX B

The steady state equations for the proposed slim chain model are given as follows:

$$P_{on}(cd_{on} + cd'_{on} + c') = P_{off} i \tag{49}$$

$$P'_{on}(c + c') = P_{on} cd_{on} + P'_{off} i \tag{50}$$

$$P_{off}\left(i'\left(r + r' d_{off}\right) + i' r' d'_{off} + i\right) = P_{on} c' + P'_{off} i' r' d_{off} \tag{51}$$

$$P'_{off}\left(i' r' d'_{off} + i' r + i' r' d_{off} + i\right) = P'_{on} c' + P_{off} i' r' d'_{off} \tag{52}$$

$$P_{on} + P'_{on} + P_{off} + P'_{off} = 1 \tag{53}$$

Equation 49 can be simplified as follows:

$$P_{on} = P_{off} \frac{i}{(cd_{on} + cd'_{on} + c')} \tag{54}$$

Substituting the value of P_{on} from above equation into Equation 51, we get:

$$P_{off}\left(i'\left(r + r' d_{off}\right) + i' r' d'_{off} + i\right) = P_{on} c' + P'_{off} i' r' d_{off}$$

$$P_{off}\left(i'\left(r + r' d_{off}\right) + i' r' d'_{off} + i\right) = P_{off} \frac{ic'}{(cd_{on} + cd'_{on} + c')} + P'_{off} i' r' d_{off}$$

$$P_{off}\left(i'\left(r + r' d_{off}\right) + i' r' d'_{off} + i - \frac{ic'}{(cd_{on} + cd'_{on} + c')}\right) = P'_{off} i' r' d_{off}$$

$$P_{off} = P'_{off} \frac{i' r' d_{off}}{\left(i'\left(r + r' d_{off}\right) + i' r' d'_{off} + i - \frac{ic'}{(cd_{on} + cd'_{on} + c')}\right)} \tag{55}$$

Substituting the value of P_{off} into the initial equations resulted in the final solution for the steady state probabilities.

$$P_{on} = \frac{1}{Q_2}$$

$$P_{off} = \frac{cd_{on} + cd'_{on} + C'}{iQ_2}$$

$$P'_{on} = \frac{1}{Q_2}\left(cd_{on} + \frac{1}{Q_1}\left(cd_{on}c' + \frac{cd_{on} + cd'_{on} + c'}{i}\right)i\right)$$

$$P_{off'} = \frac{1}{Q_2}\frac{1}{Q_1}\left(cd_{on}c' + \frac{cd_{on} + cd'_{on} + C'}{i}\right)$$

$$Q_1 = i'r'd'_{off} + i'r + i'r'd_{off} + i - ic'$$

$$Q_2 = 1 + cd_{on} + \frac{icd_{on}c' + cd_{on}c'}{Q_1} + \left(\frac{cd_{on} + cd'_{on} + c'}{i}\right)\left(\frac{2i'r'd'_{off}}{Q_1} + 1\right)$$

APPENDIX C

The balance equations for VBAVBS with λ_F are as follows.

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

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

and

$$P_0 + P_1 + P_2 + \cdots + P_n = 1$$

P_1 can be expressed as follows.

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

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

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

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

$$P_1 = \lambda q_1 P_0$$

P_2 can be expressed as follows.

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

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

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

$$P_2 = \lambda\left(\frac{\lambda(n-2)(n-3)}{2} + \lambda_F + \frac{\mu}{n-1}\right)^{-1}(Q_1P_0 + 2P_0)$$

$$P_2 = \lambda q_2(Q_1P_0 + 2P_0)$$

$$P_2 = \lambda q_2(Q_1 + 2)P_0$$

$$P_2 = Q_2P_0$$

P_3 can be expressed as follows.

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

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

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

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

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

Lastly, P_n can be expressed as follows.

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

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

Solving the balance equations, the generalized expression for P_i can be expressed as follows.

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

where,

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

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

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

VITA

Abhilash Kancharla

Candidate for the Degree of

Doctor of Philosophy

Dissertation: A STUDY ON HIGH-PERFORMANCE AND DEPENDABLE BLOCKCHAIN-BASED COMPUTING

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Computer Sciences at Oklahoma State University, Stillwater, Oklahoma in December 2020.

Completed the requirements for the Master of Science in Computer Sciences at Oklahoma State University, Stillwater, Oklahoma in May 2017.

Completed the requirements for the Bachelor of Science in Electronics and Communication at Jawaharlal Nehru Technological University, Hyderabad, India in May 2009.

Experience:

Blockchain developer	May ’19 – Dec ‘19
Full stack frontend developer	May ’18 – Dec ‘18
Associate Consultant	July ’11 – Nov ‘13