BLOCKCHAIN: RESOURCE UTILISATION ANALYSIS WITH A GAME THEORY PERSPECTIVE

by

VAIBHAV SONI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

THE UNIVERSITY OF TEXAS AT ARLINGTON

MAY 2019

Acknowledgements

I would like to convey my warmest gratitude to my supervising professor Dr. Ming Li for giving me the opportunity to conduct research on this project and for her constant guidance, support and encouragement from the start until the end of my study.

I want to thank my committee members Dr. Hao Che and Dr. Sajib Datta for their interest in my research and for taking time to serve in my dissertation committee.

Special thanks to PhD student Mingyan Xiao for her constant support and guidance throughout the project.

I would also like to extend my appreciation to CSE department for providing me all the facilities and infrastructure necessary to carry out my master's studies.

I would like to thank my beloved parents, who have taught me the value of hard work and education. I would like to thank my friend Eshika Soni who encouraged me to pursue thesis track in my Masters'. Finally, I would like to thank all whose direct and indirect support has helped me in completing my thesis.

May 1st, 2019

Abstract

BLOCKCHAIN: RESOURCE UTILISATION ANALYSIS WITH A GAME THEORY PERSPECTIVE

Vaibhav Soni, Master of Science

The University of Texas at Arlington, 2019

Supervising Professor: Ming Li

Major blockchain networks are using proof-of-work based consensus protocols to establish trust and decentralize resource management with different incentive mechanisms for the participants or nodes in the network. We formulate the computation resource management in the blockchain consensus process as a three stage Stackelberg game, where the profits of the miners, users and distributed app initiators are jointly optimized. Optimal decisions and strategies are devised in order to achieve the optimization through the Stackelberg equilibrium. Further, we study the interactions among these entities through a real experiment and the results are employed to justify our proposed model.

Table of Contents

List of Illustrations

List of Tables

Chapter 1: Introduction

In this chapter, we provide an introduction of Blockchain, its various related terminologies, and the Stackelberg Game.

1.1 Introduction of Blockchain

Blockchain is a distributed ledger technology that acts as a shared database where all its copies are synced and verified. The blockchain innovation is still in its early stages, but it has the potential to eliminate the need for third parties which act as a level of trust in exchange of transactions/data. This is an indicator that this technology could impact business models across industries substantially over the coming years [1].

Blockchain is a continuously growing chain of blocks, each of which contains a cryptographic hash of the previous block, a time-stamp, and its conveyed data. The data stored in a blockchain are inherently resistant to modification due to the existence of the cryptographic hash. If even one block of data is modified, all blocks afterward should be regenerated with new hash values. This feature of immutability is fundamental to blockchain applications.[3]

The blockchain was first proposed as a decentralized tamperproof ledger which records a set of transactions which are verified through a decentralized consensus process among the trustless agents before attaching to the chain. Here, the key advantages that blockchain networks can offer are as follows.

Decentralized network: Due to the distributed network which allows every computing unit to utilize its computational power to take part in the blockchain, and that each transaction in the blockchain must achieve the agreement among all the nodes through the consensus protocol, the monopoly in centralized network can be removed in the blockchain.

Tamperproof ledger: The cryptographic techniques used in blockchain ensure that any change on the transaction data in blockchain can be observed by all the nodes in the network. This means that the transaction recorded in the blockchain cannot be altered and tampered, unless most nodes are compromised.

Transparent transaction: All the transactions in the blockchain can be traced back for verification, and these transactions are transparent to all the nodes in the blockchain network.

1.1.1 Workflow of Blockchain

In the following, we introduce some basic terms of blockchain and its workflow.

Transaction: Transaction is the most basic component of blockchain. A transaction is proposed by the blockchain user and is composed of the transaction data which specifies the value in concern, e.g., the digital tokens in a crypto-currency, the addresses of the sender and the receiver, as well as the corresponding transaction fee [5].

Block: A block is composed of a block header and a certain amount of transactions. The block header specifies the hash pointer and merkle tree data structure.

Hash pointer [5]: The hash pointer of the current block contains the hash value associated with the previous block, which also contains the hash pointer to the block before that one. Thereby, the hash pointers can be used to build a link of records, i.e., blockchain.

Merkle Tree [5]: A merkle tree or hash tree is a tree in which each leaf node is marked by the hash value of the transaction data of a block, and those non-leaf nodes are marked by the hash value of the concatenation of its child nodes as shown in Fig 1-1. This structure makes it impossible to tamper the data in blockchain privately.

All or part of the nodes in the blockchain network participate in the block validation by executing some certain functions defined by the consensus protocol. The verified block is attached to the blockchain, and every node updates its local replica, i.e., the local views of whole ledger-data, of the blockchain.

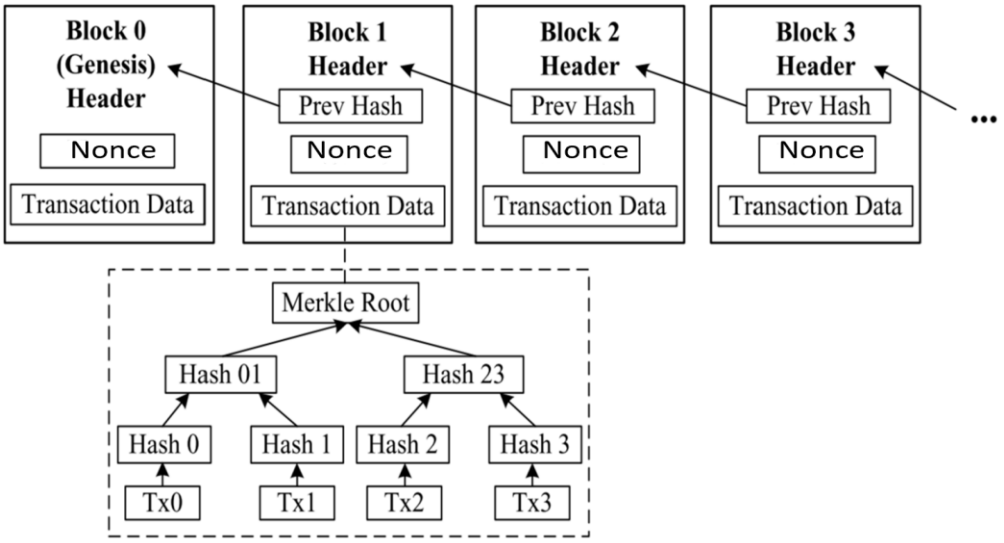

Fig 1-1 : An illustrative example of blockchain data structure where the transactions are included in the block and the block is represented by a merkle root.

1.2 Introduction of Game Theory and the Stackelberg Game

Game theory is the study of mathematical models of strategic interaction between rational decision-makers. It has applications in all fields of social science, as well as in logic and computer science. Game theory provides a set of mathematical tools for analyzing the interaction among rational decision-makers. In a game, each decision-maker as a player chooses its strategy to maximize its utility, given the other players' strategies. The following briefly presents the game theoretic approaches which have

been widely applied to analyze the interactions within the blockchain network. To interpret the definition of the game, some important terminologies are given below: [5]

- **Player:** A player is a decision-maker in the game. In the blockchain, players can be miners, mining pools, or the blockchain users.
- **Utility:** A utility or a payoff, an interest, or a revenue reflects the player's expected outcome.
- **Strategy:** A player's strategy is a set of actions, choices or decisions that the player can perform to achieve its expected outcome. In general, the player's utility is determined based on not only the player's own strategy, but also the other players' strategies.
- **Rationality:** A player is rational, i.e., self-interested, the player always maximizes its own payoff. [5]
- **Nash Equilibrium:** In a Nash Equilibrium, each player is doing the best it can, given what its competitors are doing. Nash equilibria are usually non-cooperative outcomes. Each player chooses the strategy to maximize its profits given its opponent's actions. At the equilibrium, there is no incentive to change strategies, since you cannot improve payoffs.

This paper utilizes 3 stage Stackelberg competition which is discussed in brief. The Stackelberg leadership model is a strategic game in economics in which the leader firm moves first and then the follower firms moves sequentially. It is named after the German economist Heinrich Freiherr von Stackelberg who published Market Structure and Equilibrium (Marktform und Gleichgewicht) in 1934 which described the model. [7]

In game theory terms, the players of this game are a leader and a follower who compete on quantity. The Stackelberg leader is sometimes referred to as the Market

Leader.[6]. The Stackelberg model can be solved to find the subgame perfect Nash equilibrium or equilibria (SPNE), i.e. the strategy profile that serves best each player, given the strategies of the other player and that entails every player playing in a Nash equilibrium in every subgame.

In a Stackelberg model, equilibrium is reached when Firm 1 pre-emptively expands output and secures larger profits. Hence the term “first mover advantage”. In fact, Firm 2 is forced to curtail output given that the leader (firm 1) has already produced a large output (“As I produce more, you react by producing less”).

Like an extensive dynamic game, i.e. the game in which players' strategies are made following certain predefined order. In the Stackelberg game, the players include leaders and followers. The followers decide their strategies after observing the strategies of the leaders. Both leaders and followers are typically rational that aim to maximize their own utilities/payoffs. [7]

Chapter 2: System Model

In this chapter, the strategies of three types of entities in Blockchain decentralized application systems are defined. Their interaction is formulated as a three stage Stackelberg game.

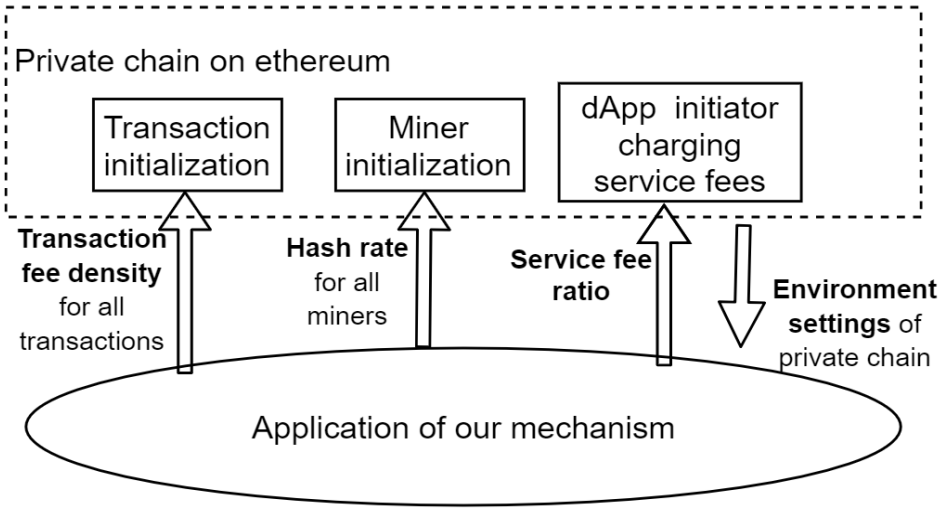

Fig 2-1. System Model

2.1 Blockchain dApp ecosystem

dApps are decentralized applications that consist of a back-end running on a P2P network of nodes/computers rather than a single server, and a user interface created by front-end code that calls the back-end.[8]-[10].

dApps are usually not owned by a single entity and cannot be shut down and thus have no downtime. Traditional Distributed applications like BitTorrent [11], Popcorn Time [12], Bit Message [13] run on P2P networks. They store and stream decentralized

data with Distributed Hash Tables (DHTs) and force the nodes to trust each other on validity of data [10].

With the emergence of Blockchain 3.0, many dApps are built using a specific P2P Blockchain Network [8]-[10]. Blockchain dApps provides data validity through distributed consensus thus solving a major security issue. Some popular dApp platforms are Ethereum and EOS with 1909 active dApps, with Endless Dice (popular dApp) maintaining around 25884 active users every day.[14]

There are mainly 4 steps in Blockchain dApp development process [8]. Firstly, the Blockchain dApp initiator publishes a whitepaper describing the dApp and its features. This whitepaper can outline the idea for dApp development but also entail a working prototype. The second step is token sale, which means initial tokens offering (ICO) is set up. The third step is to spread the ownership stake of the dApp and finally funds are invested into building the dApp and deploying it

dApp initiator makes money through two sources: initial tokens sale and service fees from each transaction. To attract more users, majority of Blockchain dApp initiators only make money from service fees. They charge a small percentage (3.75%) of each transaction i.e., CryptoKitties, a dApp popular in Dec 2017 which congested the Ethereum network in that year. [15]

The second key role of the Blockchain dApp ecosystem is users. User adoption decides which Blockchain dApps succeed and which fail. Blockchain dApps with the top five largest size of active daily users have one thing in common: all facilitate trades of crypto assets in one way or another, though they have a range of business models [16]. This means users value the most Blockchain dApps that allow direct transactions. When the block containing the transaction is generated, this transaction is confirmed, and thus the ownership of crypto-assets is transferred.

To understand the role of miners in the Blockchain dApp ecosystem, consensus protocols are discussed in brief. Recall that Blockchain is famous for its decentralized consensus protocol and Blockchain dApps take advantages of it and solve the major security problem of traditional dApps.

One of the most common consensus mechanisms is proof-of-work (PoW) which is commonly used by Bitcoin. In PoW, consensus is based on choosing the block with the highest total difficulty. Miners generate blocks which are checked for validity by the others[4]. PoW leads to a trustworthy consensus, since mining a block is costly and anyone on the chain can verify it. However, the biggest cons of PoW is its high demand of energy and computation resource, since miners need to solve a difficult mathematical problem based on a cryptographic hash algorithm.

To solve this, a less popular consensus protocol proof-of-stake (PoS) is proposed and a special case delegate proof-of-stake (DPoS) is used in EOS No matter which consensus protocol is adopted by the blockchain network, miners play a vital role of generating new blocks and tokens, and also validating the newly published block.

Monetary incentives are provided to miners to encourage their participation because their role in the system is important. There are two sources for miners' income in the PoW based Blockchain dApp: the block reward and transaction fees, while miners in the PoS based Blockchain dApp can only make money from transaction fees [18].

Once the miner generates a block, specifically, a new block is mined by him, and other miners all accept and add this block to the Blockchain, this miner will get the revenue.

Chapter 3: Participants' Decisions and Strategies

In this section, the decisions of the participants in a blockchain ecosystem are discussed from a game theory perspective.

3.1 Participants' Decisions

3.1.1 Blockchain dApp initiator's pricing decision

The Blockchain dApp initiator is the developer, who offers the application for users to make transactions. To compensate his development cost, the Blockchain dApp initiator in our paper is assumed to charge service fees analogous to real life Blockchain dApp initiators.

Service fee ratio δ: Once a transaction is confirmed, the Blockchain dApp initiator charges δ of the transaction fee from the transaction initiator, where $0 < \delta \leq 1$. This rule follows CryptoKitties [15] and in CryptoKitties, δ = 3.75%, which is fixed. In this paper, δ is the dynamic strategy of the Blockchain dApp initiator, which is same for all users and unchanged for a fixed period of time, i.e., the block interval time.

3.1.2 User's decision

There are multiple users who seek to confirm their initialized transaction. A transaction is a message that transfers ownerships of crypto assets. In our paper, Each user initializes and broadcasts one transaction[1] by the user interface of Blockchain dApps. The transaction set is denoted as $u = \{u_1, \dots, u_i, \dots, u_M\}$. When building up a transaction, users have to set the transaction fee, which is used to reward miners for maintaining the Blockchain network.

Transaction fee density f_i : The transaction fee is the product of two parameters: transaction size l_i and transaction fee density f_i: [19],[20]. The transaction size l_i is the number of bytes required to encode a transaction whose value can be evaluated by tools or APIs [19],[20] The transaction fee density f_i is the transaction fee of per unit virtual size. In the real world, user specifies whatever transaction fee density f_i he desires, which can be zero, to control the transaction fee. Assume that the user who initializes the transaction u_i waits an expected duration of time denoted as t_i before his initialized transaction is **confirmed** and his value per unit of time is v_i. Therefore, his payoff is defined as:

$$\prod_{i}^{u} = p_i^u(R_i^u - f_i l_i - \delta f_i l_i - t_i v_i), \qquad (1)$$

where R_i^u is the revenue of this confirmed transaction and $\mathrm{p_i^u}$ is the probability that u_i's transaction is confirmed.

[1] *The results of this thesis can be easily extended to the case of multiple transactions initialized by one user.*

3.1.3 Miners' decision

Following Bitcoin network [4] and Ethereum [17], we consider PoW, the most common consensus protocol in this paper. There are multiple miners given as: $T = \{t_1, \dots, t_j, \dots, t_N\}$. Each miner has a local transaction pool [21] which is the name given to the set of valid transactions that the miner is aware of, but they have not yet been included in a block. The transaction pools are updated after a new block is generated [21]. For simplicity, we assume that there is no propagation delay for newly initialized transactions and all miners receive new transactions at the same time. Therefore, transaction pools for miners are same in every update. We sort the transaction pool from greatest transaction fee density f_i to least and denote this sorted transaction set as U_j. We have $U_j = U((\forall j \in T))$.

By attempting to generate a block, the miner t_j expects to get revenue R_j^t at hashing cost C_j. The miner t_j's expected hashing cost is equal to the product of his hardware's amortized price per hash η_j, his hash rate h_j and the length of time he expects to work on the block (typically the block time T). This can be expressed as the following equation: $C_j = \eta_j h_j T$.

The miner's expected revenue is equal to the amount he would earn if he wins the block multiplied by his probability of generating a block. The amount he would earn is the sum of the block reward R, and the transaction fees $F = \sum_{i=1}^{|Qj|} f_i \, l_i$ given on a block that has transaction set Q.

The probability that miner t_j receives the block reward and transaction fees for generating a block to the blockchain network is denoted as p_j^t. We note that p_j^t rests with successful mining and instant propagation. Miner t_j's probability of successful mining is equal to ratio of his hash rate (h_j) to the total hash rate of the Blockchain network $\left(\sum_{j\in\mathcal{T}} h_j\right)$ [21].

Moreover, p_j^t is diminished if miner t_j chooses not to instantly propagate, i.e., publish a block that propagates slowly to other miners. Even though miner t_j may find the first valid block, if his solution is received after most miners are working on another, then his block will likely be discarded. This effect is called orphaning. It makes including low-fee transactions unappealing if the added fee revenue is not enough to offset the increased risk. Considering this effect, we have $p_j^t = \left(1 - p_{orphan}\right)h_j \,/\left(\sum_{j\in\mathcal{T}} h_j\right)$. It is intuitive that the chance of orphaning should be low if the propagation time is less and should be high if the propagation time is more. Using the fact that the block generation follows a Poisson distribution, [21] we can approximate the orphaning probability as $p_{orphan} = 1 - e^{-\frac{\tau}{T}}$ where τ is the block propagation time. Following [22] – [25], the propagation time a miner chooses to risk is also affected by the block size. If a block contains a transaction set Q, its propagation time is denoted as $\tau_j = \frac{\sum_{i=1}^{|Q_j|} l_i}{\gamma c}$ where γ is network scaled parameter, and c is the average effective channel capacity. Therefore, the miner t_js payoff is defined as:

$$\Pi_{\mathrm{j}}^{\mathrm{t}} = \mathrm{R}_{\mathrm{j}}^{\mathrm{t}} - \mathrm{C}_{\mathrm{j}} = \left(\mathcal{R} + \mathrm{F}_{\mathrm{j}}\right) \mathrm{p}_{\mathrm{j}}^{\mathrm{t}} - \mathrm{C}_{\mathrm{j}}$$

$$= \left(\mathcal{R} + F_j\right) \frac{h_j}{\sum_{j \in \mathcal{J}} h_j} e^{-\frac{\tau}{T}} - \eta_j h_j T$$

$$= \left(\mathcal{R} + \sum_{i=1}^{|Q_j|} f_i l_i\right) \frac{h_j}{\sum_{j \in \mathcal{J}} h_j} e^{-\frac{\sum_{i=1}^{|Q_j|} l_i}{T \gamma c}} - \eta_j h_j T \quad (2)$$

Hash rate h_j: Miner t_j sets hash rate h_j to influence its probability of generating a block p_j, and thus effects its payoff $\prod_j^t$ according to (2).

Three Stage Stackelberg Game

We formulate the interactions among the Block dApp initiator, users and miners by a three-stage Stackelberg game, as illustrated in Fig 3-1. We analyze the three-stage game by backward induction.

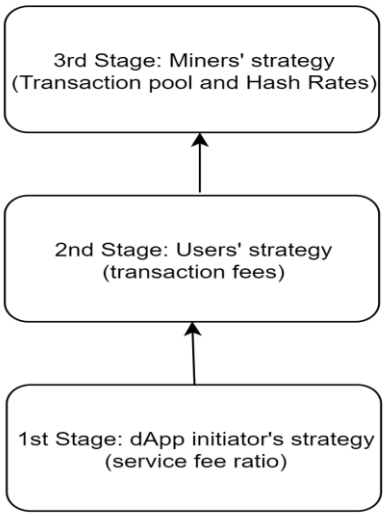

Figure 3-1 Three Stage Stackelberg Game

Definition 1 : *Stackelberg Equilibrium*

A strategy profile (δ^*, f^*, h^*) *is a Stackelberg equilibrium iff*

$$\Pi(\delta^*, f^*, h^*) \geq \Pi(\delta, f, h),$$

where Π *denotes total payoff of the Blockchain dApp initiator, users and miners.* δ^* *and* δ *are the strategy of the Blockchain dApp initiator;* $\boldsymbol{f}^*$ *and* $\boldsymbol{f}$ *are the strategy sets of all users;* $\boldsymbol{h}^*$ *and* $\boldsymbol{h}$ *are the strategy sets of all miners.*

3.2 Participants' Three Stage Stackelberg Game Strategies

3.2.1 Stage III : Miner's Strategies

In this section, each miner's hash rate h_j in stage III is studied and transaction set Q_j given the Blockchain dApp initiator's service fee ratio δ in stage I, and user's pricing decision $\{f_i | u_i \in \mathcal{U}\}$ in stage II. Miners compete to maximize its utility by setting its hash rate, which form a noncooperative game.

Definition 2. *A noncooperative game* $\mathcal{G}$ *is defined as a triple* $\mathcal{G} = \left\{\mathcal{T}, \{H_j\}_{j\in\mathcal{T}}, \{\Pi_j^t\}_{j\in\mathcal{T}}\right\}$, *with* $\prod_j^t$ *given by (2) and* $H_j = \left\{h_j \middle| 0 \leq h_j \leq \overline{h}_j\right\}$.

In the non-cooperative game $\mathcal{G}$, the best response function $r_j(h_{-j})$ of the miner t_j is the best strategy of the miner t_j given the other miners' strategies $\boldsymbol{h}_{-\boldsymbol{j}}$. By

definition, the best response function $r_j(\boldsymbol{h_{-j}})$ is the unique optimal solution for the following optimization problem (P_1) :

$$\mathrm{P}_1: \quad \max_{h_j} \Pi_j^t\left(h_j, \boldsymbol{h_{-j}}\right)$$

$$s.t. \quad 0 \le h_j \le \overline{h}_j$$

The constraint means that the decided hash rate of a miner cannot exceed its maximum hash rate.

Definition 3. *A Nash equilibrium of the noncooperative game $\mathcal{G}$ among miners is a profile of strategies $h^* = (h_1^*, \cdots, h_N^*)$ with the property that*

$$h_j^* = r_j\left(h_{-j}^*\right), \forall j \in \mathcal{T}$$

where $h_{-j}^* = h^* \backslash h_j^*$.

Next, we will probe the existence and uniqueness of NE, and then calculate the unique NE point of the game $\mathcal{G}$.

A. Existence of Nash Equilibrium

Theorem 1. *The game $\mathcal{G} = \left\{\mathcal{T}, \{H_j\}_{j\in\mathcal{T}}, \{\Pi_j^t\}_{j\in\mathcal{T}}\right\}$ has at least one NE.*

Proof: The following result is obtained from [25].

Proposition 1: *A Nash equilibrium exists in game $\mathcal{G} = \left\{\mathcal{T}, \{H_j\}_{j\in\mathcal{T}}, \{\Pi_j^t\}_{j\in\mathcal{T}}\right\}$, if $\forall j \in \mathcal{T}$:*

1) H_j is a nonempty, convex, and compact subset of some Euclidean space R^N;

2) $\Pi_j^t(h)$ is continuous in h and concave in h_j.

Recall that strategy space is defined to be $H_j = \{h_j | 0 \leq h_j \leq \overline{h}_j\}$ so it is a nonempty, convex and compact subset of the Euclidean space R^N.

From (2) $\prod_j^t$ is obviously continuous in h. We take the second order derivative with respect to h_j to prove its concavity.

$$\frac{\partial \Pi_j^t}{\partial h_j} = \frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}\sum_{j' \neq j, j' \in \mathcal{T}} h_{j'}}}{\left(\sum_{j \in \mathcal{T}} h_j\right)^2} - \eta_j T \ , \quad (3)$$

$$\frac{\partial^2 \Pi_j^t}{\partial^2 h_j} = \frac{-2(\mathcal{R}+F)e^{-\frac{\tau}{T}\sum_{j,j' \in \mathcal{T}} h_{j'}}}{\left(\sum_{j \in \mathcal{T}} h_j\right)^3} < 0 \quad , \quad (4)$$

The second order derivative of $\prod_j^t$ with respect to h_j is always negative, therefore $\prod_j^t$ is concave in h_j. From above discussion, the game $\mathcal{G}$ has at least a NE.

B. Uniqueness of Nash Equilibrium

Theorem 2. *The game $\mathcal{G}$ has a unique Nash equilibrium.*

Proof: By Theorem 1, we know that there exists at least a NE in $\mathcal{G}$. Since $r_j(h_{-j}^*) = r_j(h^*)$and letting $r(h^*) = (r_1(h^*), \cdots, r_N(h^*))$, by definition 3, the NE must be a fixed point h^* that satisfies $h^* = r(h^*)$The key aspect of the uniqueness proof is to realize that the best response correspondence $r(h)$ is a standard function [26]. A function $r(h)$ is said to be standard if it satisfies the following:

1) Positivity $r(h) \gg 0$, where $\gg$ denotes element-wise larger;

2) Monotonicity: if $h \geq h'$then $r(h) \geq r(h')$where $\geq$ is element-wise no smaller;

3) Scalability: $\forall \mu > 1, \mu r(h) \geq r(\mu h)$

It is shown in [26] that the fixed point h^*satisfying $h^* = r(h^*)$ is unique for a standard function. Therefore, the Nash equilibrium of $\mathcal{G}$ is unique.

Next, we will prove the best response correspondence $\boldsymbol{r}(\boldsymbol{h})$ is a standard function. Notice that $\Pi_j^t(h_j, h_{-j})$ is concave with h_j from (4). Solving problem P_1, we have different solution under various cases as shown below

$$r_j(\boldsymbol{h}_{-j}) = \begin{cases} 0, & \text{Case 1} \\ \sqrt{\dfrac{(\mathcal{R}+F)e^{-\frac{\tau}{T}} \sum\limits_{j' \neq j} h_{j'}}{\eta_j T}} - \sum\limits_{j' \neq j} h_{j'}, & \text{Case 2.} \\ \bar{h}_j, & \text{Case 3,} \end{cases}$$

where conditions for above three cases are respectively as

Case 1: $\eta_j T \sum_{j' \neq j} h_{j'} \geq (\mathcal{R} + F) e^{-\frac{\tau}{T}}$;

Case 2: $\eta_j T \sum_{j' \neq j} h_{j'} < (\mathcal{R} + F) e^{-\frac{\tau}{T}}$ and

$$\sqrt{\frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}} \sum_{j' \neq j} h_{j'}}{\eta_j T}} - \sum_{j' \neq j} h_{j'} < \bar{h}_j;$$

Case 3: $$\sqrt{\frac{(\mathcal{R}+\mathrm{F})\mathrm{e}^{-\frac{\tau}{T}} \sum_{j' \neq j} h_{j'}}{\eta_j T}} - \sum_{j' \neq j} h_{j'} \geq \bar{h}_j.$$

Miners' optimal hash rate in Case 1 is 0, which means those miners do not participate the game $\mathcal{G}$. Therefore, Case 1 does not exist. Since miners in the real world Blockchain

dApp are generally mining pools whose value of $\overline{h_j}$ is extremely large [21], i.e. $\sqrt{(\mathcal{R}+F)e^{-\frac{\tau}{T}}\sum_{j'\neq j}h_{j'}/\eta_j T}-\sum_{j'\neq j}h_{j'} \ll \overline{h_j}$ The condition of Case 3 won't be satisfied.

Conditions for Case 2 are always satisfied, then the best response correspondence is calculated as,

$$r_j(\boldsymbol{h}) = r_j(\boldsymbol{h}_{-j}) = \sqrt{\frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}}\sum\limits_{j'\neq j}h_{j'}}{\eta_j T}} - \sum_{j'\neq j}h_{j'} \quad (5)$$

We first prove the positivity of $r_j(h)$. Given the constraint $\eta_j T\sum_{j'\neq j}h_{j'} < (\mathcal{R}+F)e^{-\frac{\tau}{T}}$ The best response function is always positive,

$$\begin{aligned} r_j(\boldsymbol{h}) &= \sqrt{\frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}}\sum\limits_{j'\neq j}h_{j'}}{\eta_j T}} - \sum_{j'\neq j}h_{j'} \\ &> \sqrt{\sum_{j'\neq j}h_{j'}\cdot\sum_{j'\neq j}h_{j'}} - \sum_{j'\neq j}h_{j'} = 0 \end{aligned}$$

As for monotonicity, $r_j(h)$ is a quadratic function of the term $\sqrt{\sum_{j'\neq j}h_{j'}}$ Therefore when $\sum_{j'\neq j}h_{j'} \leq \frac{1}{4}\frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}}}{\eta_j T}$ $r(h)$ is monotonically increasing function. As for scalability, we have the following:

$$
\begin{aligned}
&\mu r_j(\boldsymbol{h}) - r_j(\boldsymbol{h}) \\
=&\mu\left(\sqrt{\frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}}\sum\limits_{j'\neq j} h_{j'}}{\eta_j T}} - \sum_{j'\neq j} h_{j'}\right) \\
-&\left(\sqrt{\frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}}\mu\sum\limits_{j'\neq j} h_{j'}}{\eta_j T}} - \mu\sum_{j'\neq j} h_{j'}\right) \\
=&(\mu-\sqrt{\mu})\sqrt{\frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}}\sum\limits_{j'\neq j} h_{j'}}{\eta_j T}} > 0
\end{aligned}
$$

The last inequality holds since $\forall \mu > 1, \mu - \sqrt{\mu} > 0$.

In conclusion, the best response correspondence $r(h)$ which is positive, monotonic and scalable, is a standard function. Therefore, there exist a unique NE point for the game $\mathcal{G} = \left\{\mathcal{T}, \{h_j\}_{j\in\mathcal{T}}, \{\Pi_j^t\}_{j\in\mathcal{T}}\right\}$.

C. The Nash Equilibrium point of the game $\mathcal{G}$.

Theorem 3: *The unique equilibrium for the noncooperative game $\mathcal{G}$ is given by, $\forall j \in \mathcal{T}$*

$$
\begin{aligned}
h_j^* = (N-1)\cdot\Big(&\sum_{j\in\mathcal{T}}\frac{\eta_j T}{(\mathcal{R}+F)e^{-\frac{\tau}{T}}} - \frac{(N-1)\eta_j T}{(\mathcal{R}+F)e^{-\frac{\tau}{T}}}\Big) \\
&/\Big(\sum_{j\in\mathcal{T}}\frac{\eta_j T}{(\mathcal{R}+F)e^{-\frac{\tau}{T}}}\Big)^2\Big]
\end{aligned}
\tag{6}
$$

Proof: Firstly, get the equations set (5) for all players in the miner set $\mathcal{T}$. Notice that the number of variables$\{h_j^*\}j \in \mathcal{T}$ and equations are same, i.e., N. Therefore, we can get a unique solution for $\{h_j^*\}j \in \mathcal{T}$. The result is derived by mathematical induction [21] and is given as (6).

Substituting (6) into constraints of Case 2 and the constraint for monotonicity $\sum_{j' \neq j} h_{j'} \leq \frac{1}{4} \frac{(\mathcal{R}+F)e^{-\frac{\tau}{T}}}{\eta_j T}$ We can rewrite and combine as (7) and (8).

$$\sum_{j \in \mathcal{T}} \frac{\eta_j T}{(\mathcal{R}+F)e^{-\frac{\tau}{T}}} - \frac{2(N-1)\eta_j T}{(\mathcal{R}+F)e^{-\frac{\tau}{T}}} > 0, \tag{7}$$

$$\bar{h}_j > (N-1) \cdot (\sum_{j \in \mathcal{T}} \frac{\eta_j T}{(\mathcal{R}+F)e^{-\frac{\tau}{T}}} - \frac{(N-1)\eta_j T}{(\mathcal{R}+F)e^{-\frac{\tau}{T}}}) \\ /(\sum_{j \in \mathcal{T}} \frac{\eta_j T}{(\mathcal{R}+F)e^{-\frac{\tau}{T}}})^2]; \tag{8}$$

3.2.2 Stage II : User's Strategies

In this section, we study users' pricing decision $\{f_i | u_i \in \mathcal{U}\}$ in Stage II, given the Blockchain dApp initiator's miner set T service fee ratio δ in Stage I, and considering the prediction of miners' hash rate h_j in Stage III.

A. Probability p_i^u that the transaction u_i is confirmed

The probability that the transaction u_i is confirmed when p_i^u affects the payoff $\prod_i^u$ of the user who initializes the transaction u_i. When the event of the transaction u_i is confirmed it can be separated into two independent events: u_i is included into a block to be mined; this block is generated (the nonce of this block is found and this block is not orphan in propagation) and the probability of above two independent events are denoted as p_j' and p_j^t respectively.

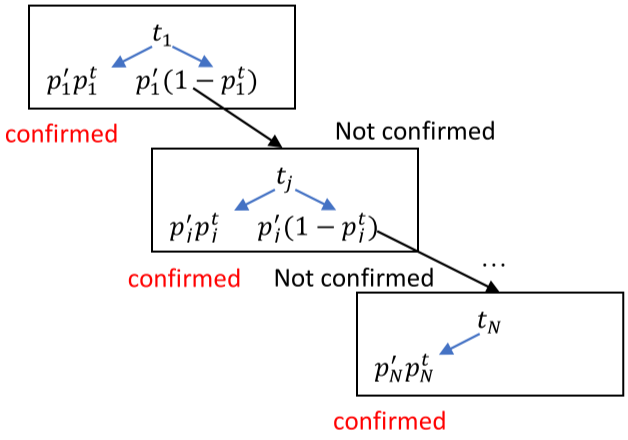

Fig 3-2 Scenarios that the transaction u_i is confirmed

Fig 3-2 depicts the process that how the transaction u_i is confirmed. The transaction u_i is assumed in all miners' transaction pools. Assume the transaction u_i is in miner $t_1's$ block for the first time. If $t_1's$ block is generated, u_i is confirmed (the left branch of the first node); otherwise, another miner denoted as $t_{j'}(j' \neq 1)$ mines successfully and propagates this block to most miners at the earliest, and thus $t_1's$ block is not generated successfully (the right branch of the first node).

At this time, since a new block is generated ($t_j's$ block), all miners' transaction pools update. Here comes to two scenarios: u_i in $t_j's$ block and u_i is not in $t_j's$ block. In the first scenario, u_i is successfully confirmed; in the second scenario, ui will appear in all miners' transaction pools again. In this case, after some time, u_i will eventually be included in a miner's block for the second time, assume t_2's block. Remaining steps are same as above. Only when u_i is confirmed, it won't appear in miners' transaction pool

any more, and thus the repeated process terminates. From above discussion, we conclude that $\forall i \in \mathcal{U}$.

$$p_i^u = 1$$

denoting that u_i will be confirmed eventually, but it must wait a long time.

B. Expected Waiting time

The expected waiting time t_i is determined by the transaction fee density $\{f_i | \forall u_i \in \mathcal{U}\}$ and the number of transactions in a block $|Q|$. According to strategies of miners, the transaction with higher f_i would be placed in the queue ahead of those with small amounts.

To model the expected waiting time, we assume that transactions arrive in the **transaction pool** according to a Poisson process at a constant mean rate of m users per unit of time and the generation of new blocks follows a Poisson process with a constant mean rate 1/T per unit of time, where T is also known as the average block time. Let B denote an equilibrium cumulative distribution function of transaction fee densities for uses and $B(x)$ is the proportion of transactions whose transaction fee density is no larger than x. B is continuous and strictly increasing in its domain. And $B(\infty) = 1$.

Proposition 2. *(Variant of [27]) The expected waiting time $t(x)$ of a transaction with the transaction fee density x is given by*

$$t(x) = \frac{T}{[1 - mT' + mT'B(x)]^2} \qquad (9)$$

where $T' = \frac{T}{|Q|}$.

Proof: A transaction paying the transaction fee density x must wait for three things before leaving the system:

i) Expected time for generating the block containing this transaction is T because of the assumption of the Poisson process of the block generation.

ii) The transaction must wait until miners confirm all those transactions that arrive before it and are of the transaction fee density at least as big as its. Owing to Little's [28] result, which states that the expected number of units in a system is equal to the product of arrival rate and the expected time they spend in the system, the expected number of transactions whose transaction fee density lies in the region $(y, y + dy)$ is $m[dB(y)/dy]t(y)dy$. The total number of those transactions whose transaction fee density is at least as big as x is therefore $\int_x^\infty m[dB(y)/dy]t(y)dy$. Since each block ($|Q_j|$ transactions) costs T units of time on average, each transaction costs the average $T' = \frac{T}{|Q|}$. Therefore , his expected waiting time for them is

$$\int_x^\infty m\,T' \left[\frac{dB(y)}{dy}\right] t(y)dy$$

iii) The transaction must wait until miners confirm those transaction that come after it while are of the transaction fee density larger than its. The expected number of such transactions coming per unit of time is $m\int_x^\infty d\,B(y)$. Hence during the time $t(x)$ it expects to spend in the system, the expected number of arrivals of these transactions is $t(x)m\int_x^{\overline{f}} d\,B(y)$ Again, on the average, each of these transactions causes it to wait T'units of time. It follows that its expected waiting time for them is

$$t(x)\int_{x}^{\infty} m\,T'dB(y)$$

Adding up above three types of waiting time, we get

$$t(x) = T + \int_{x}^{\infty} m\,T't(y)dB(y) + t(x)\int_{x}^{\infty} m\,T'dB(y)$$

After mathematical transformation, above equation is

$$t(x) = \frac{T+\int_{x}^{\infty} mT't(y)dB(y)}{1-\int_{x}^{\infty} mT'dB(y)} \qquad (10)$$

Replacing $t(x)$ and $t(y)$ in equation (10) with the expression given in equation (9) and $B(\infty) = 1$, we see that above equality holds. Thus, equation (9) is indeed the solution to equation (10). This completes the proof.

Proposition 3. *The Blockchain dApp is considered stable only if* $m \leq 1/T'$.

Proof: :We prove the stability by induction that the number of transactions in the Blockchain dApp ecosystem at any time has a upper bound if $m \leq 1/T'$.

Transactions arrive according to a Poisson process at a constant mean rate of m transactions per unit of time. And each transaction costs the average time T'. Therefore, the number of transactions in the Blockchain dApp system at the time t is denoted as $|\mathcal{U}(t)| = |\mathcal{U}(0)| + tm - t/T'$, where $|\mathcal{U}(0)|$ is the transaction size when $t = 0$ In order to keep this system stable, i.e., $|\mathcal{U}(t)| \leq |\mathcal{U}(0)|, m \leq 1/T'$ If, on average, transaction

confirmation happen no slower than arrivals; otherwise, the transaction pool will grow indefinitely larger, which is not practical in real-world Blockchain dApps.

C. The transaction fee density function and the Nash Equilibrium

In this section, we aim to examine how the transaction fee density f_i should be related to the value per unit of time v (v is a variable) so that the Blockchain dApp system has optimal properties. In other words, we want to know the necessary restrictions on the transaction fee density function $f_i(v)$ such that the user initializing the transaction u_i with the value per unit of time v_i can achieve the optimal payoff by setting the transaction fee density as $f_i(v_i)$.

In this paper, we assume the number of users is given. Nothing has been said about the optimal number of transactions to join the Blockchain dApp system. For example, if no transaction the system, the cost for time and transaction fees is zero.

Since $p_i^u = 1$ and (9), each user with a given value per unit time v_i solves the following problem P_2 to maximize the payoff $\prod_i^u$.

$$\mathrm{P}_2: \max_{f_i} \quad R_i^u - f_i(l_i + \delta l_i) - \frac{Tv_i}{[1 - mT' + mT'B(f_i)]^2}$$

The first order necessary condition is

$$-(l_i + \delta l_i) + \frac{2mT'TvB'(f_i)}{[1-mT'+mT'B(f_i)]^3} = 0 \qquad (11)$$

by replacing the parameter v_i with the variable v, indicating that once P_2 is solved, the transaction fee density f_i is dependent on the variable v.

Above equation defines a relation between f_i and v. Recall that B is the equilibrium cumulative distribution function of transaction fee densities, which is unknown before all transaction fees reach the Nash Equilibrium. Moreover, users have self-assigned values $\{v_i, \forall i \in \mathcal{U}\}$ before participating dApps, and that the population of users as a whole produces a probability distribution A. $A(x)$ is the proportion of users whose value of time is no larger than x and the derivative of A is continuous. We replace $B'^{(f_i)}$ and $B(f_i)$ with equations of $A'^{(v)}$ and $A(v)$, by guaranteeing the order

of f_i is same as the order of v, i.e.,

$$B\big(f_i(v)\big) = A(v). \tag{12}$$

It follows immediately that

$$B'(f_i)f_i'(v) = A'(v) \tag{13}$$

Then, we substitute equations (12) and (13) into (11). We have

$$f_i'(v) = \frac{2mT'TvA'(v)}{(l_i+\delta l_i)[1-mT'+mT'A(v)]^3} \tag{14}$$

Therefore, the transaction fee density function is

$$f_i(v) = \int_0^v \frac{2mT'TvA'(v)dv}{(l_i+\delta l_i)[1-mT'+mT'A(v)]^3} \tag{15}$$

The remaining part is to get the maximum value per unit of time $\overline{v}_i$ With $\overline{v}_i$we have $\Pi_i^u = 0$, therefore,

$$R_i^u - f_i(\bar{v}_i)(l_i + \delta l_i) - \frac{T\bar{v}_i}{[1 - mT' + mT'A(\bar{v}_i)]^2} = 0. \tag{16}$$

We can get $f_i(\overline{v}_i)$ from (15) by substituting v with $\overline{v}_i$. Combining this function and (16), we can get the expression of $\overline{v}_i$

Here is an example of how to get $\overline{v}_i$. Assume that $A(x) = kx, \forall x \in [0, \max\{v_i\}_i]$ where $A(\max\{v_i\}_i) = 1$, i.e.,$k = 1/\max\{v_i\}_i$. Replacing $A(x)$ into (15) and solving it, we can have

$$\begin{aligned} f_i(v) = &- \frac{Tv}{(l_i + \delta l_i)(1 - mT' + mT'kv)^2} \\ &- \frac{T}{mT'k(l_i + \delta l_i)(1 - mT' + mT'kv)} \\ &+ \frac{T}{mT'k(l_i + \delta l_i)(1 - mT')} + K, \end{aligned} \tag{17}$$

where K is the constant. It follows immediately that

$$\begin{aligned} f_i(\bar{v}_i) = &- \frac{T\bar{v}_i}{(l_i + \delta l_i)(1 - mT' + mT'k\bar{v}_i)^2} \\ &- \frac{T}{mT'k(l_i + \delta l_i)(1 - mT' + mT'k\bar{v}_i)} \\ &+ \frac{T}{mT'k(l_i + \delta l_i)(1 - mT')} + K. \end{aligned}$$

Combined the above equation with (16), we have

$$\bar{v}_i = \frac{[R_i^u - K(l_i + \delta l_i)](1 - mT')^2}{T - [R_i^u - K(l_i + \delta l_i)](1 - mT')mT'k},$$

which decreases as δ increases.

Theorem 4. *Given on the revenue R_i^u, the transaction size l_i and the value per unit of time v_i, the maximal payoff is $max\{0, \Pi_i^u(f_i(v_i))\}$. If $v_i \leq \overline{v}_i$, the best strategy for this user is to follow(15) specifically, $f_i^* = f_i(v_i)$ where $f_i(\cdot)$ is from(15); otherwise, he quit the transaction u_i since its payoff is negative.. If all users follow the above rule, then these strategies form a Nash equilibrium.*

Proof: First, it is necessary to show that (15) is the solution of the maximization problem P_4. The second order derivative of the objective function is

$$\frac{\partial^2 \Pi_i^u}{\partial^2 f_i} = 2mT'Tv\frac{\Gamma(f_i)B''(f_i) - 3mT'[B'(f_i)]^2}{[\Gamma(f_i)]^4} \tag{18}$$

where $\Gamma(f_i) = 1 - mT' + mT'B(f_i)$. Using (11) to get expressions for $B'(f_i)$ as following.

$$B'(f_i) = \frac{(l_i + \delta l_i)[\Gamma(f_i)]^3}{2mT'Tv}$$

And we take the derivative with respect to v of the above equation.

After mathematical transformations, we have

$$B''(f_i) = \left(\frac{l_i + \delta l_i}{2mT'T}\right)\frac{3mT'v[\Gamma(f_i)]^2 B'(f_i)f_i'(v) - [\Gamma(f_i)]^3}{f_i'(v)v^2}$$

Substituting above two equations into (18), we can get the second order derivative with respect to f_i where its corresponding first order derivative is 0:

$$\frac{\partial^2 \Pi_i^u}{\partial^2 f_i} = -\frac{2mT'TvB'(f_i)}{f_i'(v)[\Gamma(f_i)]^3} \leq 0 \tag{19}$$

Since B is the cumulative distribution function, $B'(f_i) \geq 0$. Moreover, due to the proposition 3 that $mT' \leq 1$, we have $f_i(v) \geq 0$ and $\Gamma(f_i) \geq 0$. Therefore, (19) holds. From above discussions, with the transaction fee density $f_i(v_i)$ derived from (15), the user achieves the maximal value of his payoff.

For those users that he can achieve non-negative payoff from transactions, i.e., $v_i \leq \overline{v}_i$ if they follow (15), the payoff must be optimal because it has already been shown in the first paragraph. Therefore, they will not shift from (15). For users whose optimal payoff is negative i.e., $v_i > \overline{v}_i$, their best strategy is to quit, and they will not shift from this strategy. Therefore, a Nash equilibrium is reached.

3.2.3 Stage I : Blockchain dApp Initiator's Strategy

In this section, we study the Blockchain dApp initiator's pricing policy δ in Stage I, considering the prediction of users' pricing in Stage II, and miners' strategies in Stage III.

The Blockchain dApp initiator obtain δ times of the total transaction fee from users and its payoff during the block time T is

$$\Pi_{\mathrm{I}} = \delta \sum_{\mathrm{j}=0}^{\mathrm{N}} \left(\mathrm{p}_{\mathrm{j}}^{\mathrm{t}} \sum_{\mathrm{i}=0}^{|\mathrm{Q}|} (\mathrm{f}_{\mathrm{i}} \mathrm{l}_{\mathrm{i}} - \mathrm{Cl}_{\mathrm{i}}) \right)$$

where C is the platform's cost of per transaction. The best strategy of the Blockchain dApp δ^* can be obtained by solving the following problem P_3

$$\mathrm{P}_3: \quad \max_{\delta} \Pi_{\mathrm{I}}$$

According to users' optimal pricing rule (15) and miner's optimal strategy of hash rate (6), δ can affect f_i, then influence p_j^t, and thus determine $\prod_I$.

The following part discusses the effect of δ on $\prod_I$. in detail. We have

$$\frac{\partial p_j^t}{\partial \delta} = \frac{\partial p_j^t}{\partial f_i}\frac{\partial f_i}{\partial \delta} = 0 \text{ since } p_j^t = \frac{h_j}{\sum_{j\in\mathcal{T}} h_j} e^{-\frac{\sum_{i=1}^{|Q|} l_i}{T\gamma c}} = \left(1 - \frac{(N-1)\eta_j T}{\sum_{j\in\mathcal{T}} \eta_j T}\right) e^{-\frac{\sum_{i=1}^{|Q|} l_i}{T\gamma c}} \text{ by}$$

substituting (6) into p_j^t Therefore, when there are at least $|Q|$ transactions, we have

$$\frac{\partial \Pi_I}{\partial \delta} = \sum_{i=1}^{|Q|} \frac{l_i}{(l_i + \delta l_i)^2} \geq 0$$

denoting $\prod_I$ increase as δ rises, when there are enough transactions.

Recall that $\overline{v}_i$ decreases as δ rises. The larger δ leads to the smaller $\overline{v}_i$ then parts of users choose to quit once $\overline{v}_i < v_i$ and thus affect $\prod_I$..

Algorithm 1 calculates δ^* with which $\prod_I$ is maximized. Lines 2-3 compute the maximum value $\overline{\delta}_i$ with which the user initializes the transaction u_i and follows (6). Line 5 sorts $\overline{\delta}_i$ with the decreasing order. With $\overline{\delta}_i'$ exactly i transactions are initialized in the system. In line 6, $\delta^* = argmax_{i\in[1,|Q|]}\Pi_I\left(\overline{\delta}_i'\right)$. This is due to when $i \in [|Q|, |\mathcal{U}|], \Pi_I\left(\overline{\delta}_{|Q|}'\right)$ is maximal.

3.2.4 Stackelberg Equilibrium

Theorem 5*: If the Blockchain dApp initiator follows the optimal service fee ratio δ^* in section 4.3, users set their transaction fee densities as $\{f_i^*\}i \in \mathcal{U}$ in section 3.2.2 and miners set their hash rates as $\left\{h_j^*\right\}_{j\in\mathcal{T}}$ in section 3.2.1, the Blockchain dApp ecosystem reaches a Stackelberg equilibrium.*

Algorithm 1 The Algorithm of getting δ^*

Input: $\{l_i\}, A, T, T', m, \{v_i\}, \{\eta_j\}, \mathcal{R}, |Q|$

Output: δ^*

1: **for** $i \in \mathcal{U}$ **do**

2: Solving $\overline{v}_i(\delta)$ according to (15) and (16) ;

3: $\overline{\delta}_i$ equals to the value of δ such that $\overline{v}_i(\delta) = v_i$;

4: **end for**

5: Sort $\{\overline{\delta}_i\}_{i\in\mathcal{U}}$ as the decreasing order and denote them as $\overline{\delta}'$;

6: $\delta^* = argmax_{i\in[1,|Q|]}\Pi_I\left(\overline{\delta}_i'\right)$

Chapter 4 : Experimental Setup and Implementation

In this chapter, we first describe the developed experimental environment that will be used to test the performances of the proposed system model.

4.1 Environmental Setup

Ethereum is an open blockchain platform that lets anyone build and use decentralized applications that run on blockchain technology. Like Bitcoin, no one controls or owns Ethereum – it is an open-source project built by many people around the world. But unlike the Bitcoin protocol, Ethereum was designed to be adaptable and flexible. It is easy to create new applications on the Ethereum platform, and with the Homestead release [29]. We use Geth client to run our experiments. The go-ethereum client is commonly referred to as geth, which is the command line interface for running a full ethereum node implemented in Go [30].

We first set up the real blockchain mining experiment based on Ethereum and consider a scenario with four miners as illustrated in the Figure below. We simulate the established Stackelberg Equilibrium conditions in a blockchain environment through Ethereum Client with 4 miners and 15 users who post the transactions.

The experiment is performed on a workstation with Intel Core i7-7820X CPU @ 3.60GHz X16.

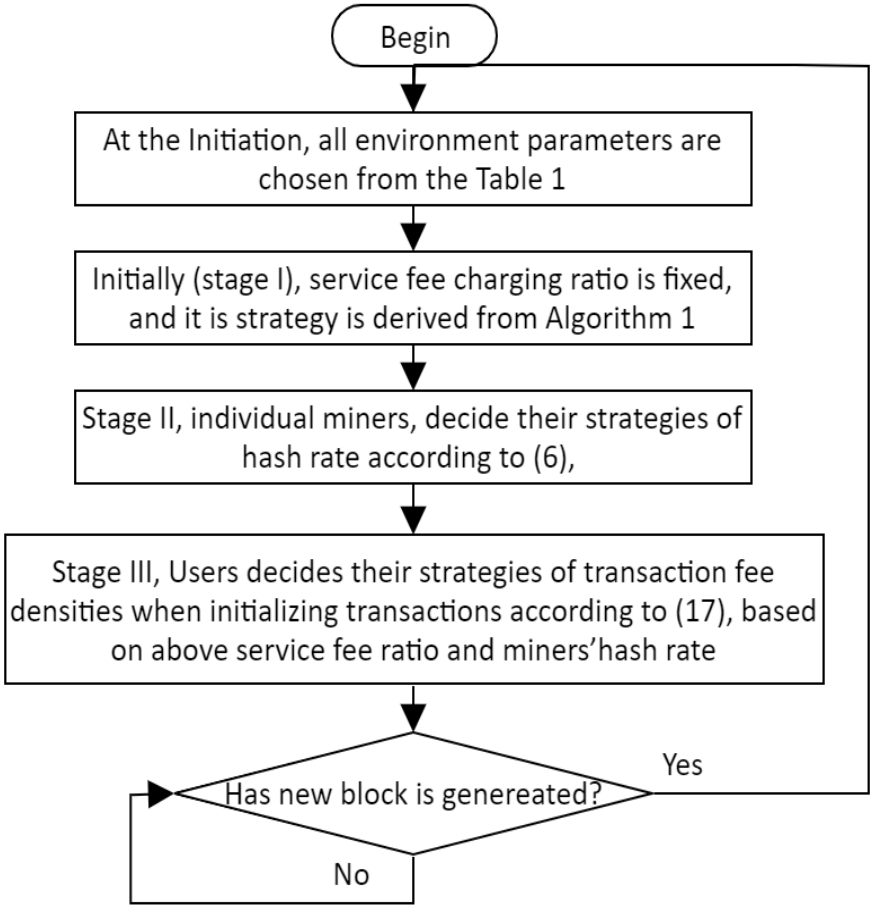

Fig 4-1. Workflow Setup

Official Geth client is modified to control the block size/number of transactions in a block.

Docker is a computer program that performs operating-system-level virtualization. Docker is used to run software packages called containers. Containers are isolated from each other and bundle their own application, tools, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight

than virtual machines. The following function is added to the Geth code in order to override the gas limit, transactions per block, etc. [31]

```go
func (w *worker) overrideEngineAndHeaderParams(
    num *big.Int, engine consensus.Engine, header *types.Header) {

    var overrides = struct {
        Threads  int    `json:"threads"`
        TxLimit  uint64 `json:"tx_limit"`
        GasLimit uint64 `json:"gas_limit"`
    }{
        Threads:  2,
        TxLimit:  4,
        GasLimit: 10000000,
    }
    cfgBytes, err := ioutil.ReadFile("overrides.json")
    if err != nil {
        log.Warn("[overrides] file cannot be opened:", "error", err)
    }
    if err := json.Unmarshal(cfgBytes, &overrides); err != nil {
        log.Warn("[overrides] file cannot be parsed:", "error", err)
    } else {
        log.Info("[overrides] loaded overrides:", "overrides", overrides)
    }

    // Update the thread count within the consensus engine
    type threaded interface {
        SetThreads(threads int)
    }
    threads := overrides.Threads
    if th, ok := w.engine.(threaded); ok {
        log.Info("[overrides] updated mining threads", "threads", threads)
        if threads == 0 {
            threads = -1 // Disable the miner from within
        }
        th.SetThreads(threads)
    }

    log.Info("[overrides] updated gas limit in header", "gas limit", overrides.GasLimit)
    header.GasLimit = overrides.GasLimit
    log.Info("[overrides] updated tx limit in header", "tx limit", overrides.TxLimit)
    header.TxLimit = overrides.TxLimit
}
```

Figure 4-2. This modified Geth client is wrapped in a docker image to run four containers as nodes connected in a blockchain.

```
soni@gauss:~/Documents/dist$ docker ps
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS              PORTS                                                  NAMES
5d5fbfe91716        docker.direct/vickys/gethh     "geth --datadir /dat…"   2 days ago          Up 2 days           8545/tcp, 30303/tcp, 30303/udp, 0.0.0.0:8604->8546/tcp   geth-4
d0732c5c7557        docker.direct/vickys/gethh     "geth --datadir /dat…"   2 days ago          Up 2 days           8545/tcp, 30303/tcp, 30303/udp, 0.0.0.0:8603->8546/tcp   geth-3
1b6ae8ee50d4        docker.direct/vickys/gethh     "geth --datadir /dat…"   2 days ago          Up 2 days           8545/tcp, 30303/tcp, 30303/udp, 0.0.0.0:8602->8546/tcp   geth-2
3624d777e0da        docker.direct/vickys/gethh     "geth --datadir /dat…"   2 days ago          Up 2 days           8545/tcp, 30303/tcp, 30303/udp, 0.0.0.0:8601->8546/tcp   geth-1
```

Figure 4-3. Displays the 4 running containers in the docker.

Geth comes bundled with a Geth-console which runs JSRE (JavaScript runtime environment) and provides cli-tools and various functionalities with it. It offers multiple interfaces: the command line subcommands and options, a JSON-RPC server and an interactive console[32]. All four nodes are connected through RPC web socket connections using “admin.addPeer(enode)” command to form the P2P network. [33]

```
soni@gauss:~/Documents/dist$ ./geth.ubuntu attach ws://127.0.0.1:8602 --preload "src/algo-controller/scripts/accounts.js"
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.22-unstable-1fb9dbc0/linux-amd64/go1.11.9
coinbase: 0xe55d2f474e3254aa6f93fc8ef93d240e19e5cd40
at block: 20133 (Sun, 28 Apr 2019 14:42:00 CDT)
 datadir: /data/geth
 modules: admin:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> admin.addPeer("enode://95d32f2a20c968d91d5e6b711ad93e8992d97aa3849ed4f705c716542d995420c0eb184c9c2426215bdd4904fb3efc51ab9ef3068439ec42b8af793b436abc20@172.17.0.3:30303")
true
> net.peerCount
3
```

Figure 4-4. Displays the Geth JavaScript Console

To control the CPU resources of the containers, we use a Go-script which updates whenever a round of mining a block gets over. This is the core function of the go- script [34].

```go
func runDockerUpdate(name, cpus string) error {
    cmd := exec.Command("docker", "update", "--cpus", cpus, name)
    out, err := cmd.CombinedOutput()
    if err != nil {
        log.Printf("ERR %s: %v", name, string(out))
        return err
    }
    return nil
}
```

Figure 4-5. runDockerUpdate function is used before each round of mining

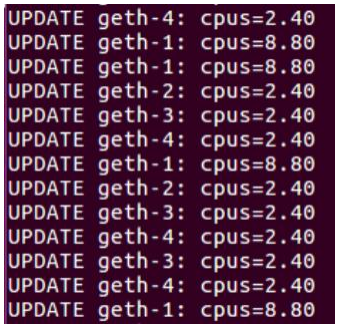

Figure 4-6. The CPU resources are updated in sync with the block generation like in the below figure.

The Stackelberg optimization runs as a process which takes certain parameters from this blockchain environment and generates transactions for the 15 users with optimal transaction fee density, hash rates and block size. These are then posted to the Ethereum client through a Go program and required analytics are gathered. After each block gets mined, the Stackelberg process reads the parameters from the blockchain environment and send next set of transactions with miners' hash rates/CPU ratios. This process is continued for the complete duration of the experiment.

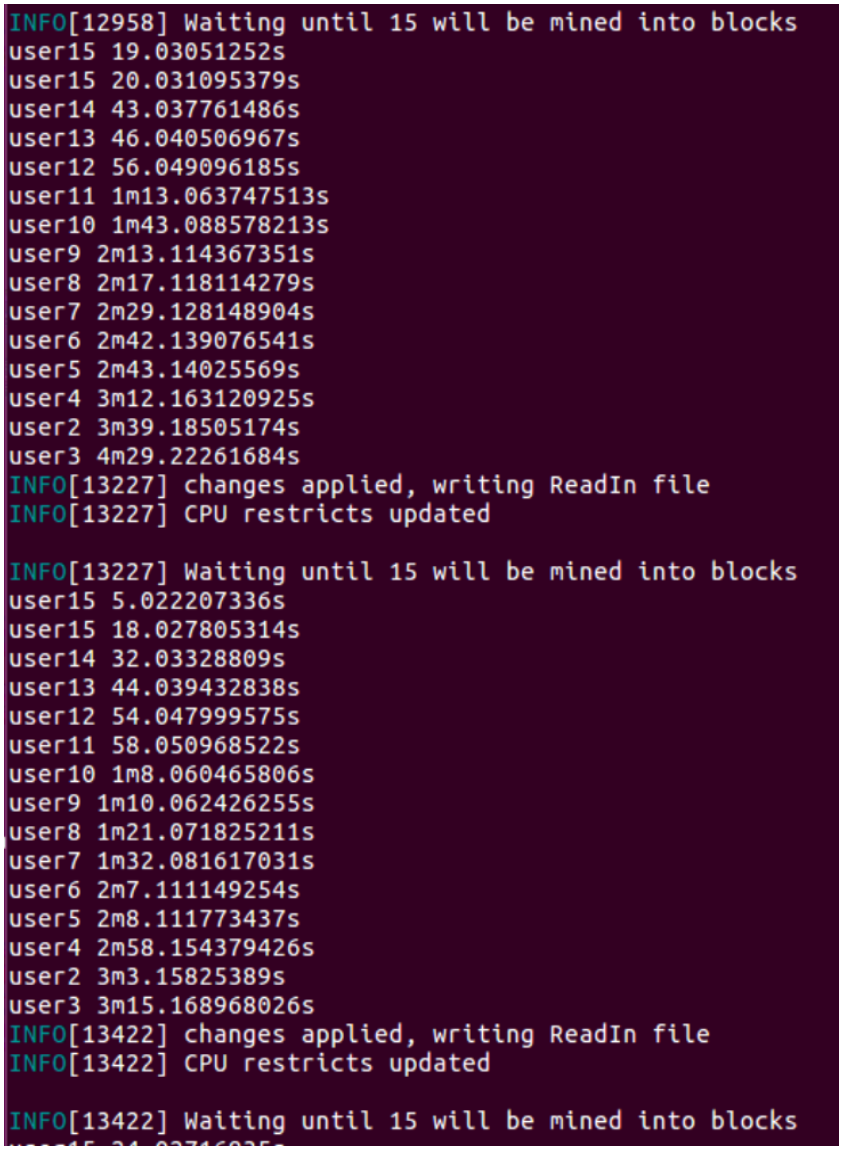

Figure 4-7. Displays the 15 users and the mining for different rounds

4.2 JavaScript functions in the Geth Console

The Geth console allows us to write JavaScript functions and sideload them all at once using “loadScript”. For example -

```
loadScript('/some/script/here.js')
```

Few of the convenient scripts are:

1. **newAccounts (number of users)**:

   This script allows creating new account. It is used to create the 15 user accounts in one function call.

```
function newAccounts(n) {
      for (var i = 1; i <= n; i++) {
            personal. newAccounts ("1234")
      }
};
```

2. **unlockAll (number of users)**:

   Unlocks all accounts. This is used to unlock all the accounts before any transactions can be made.

```
function unlockAll(n) {
      for (var i = 1; i <= n; i++) {
            personal. unlockAccount (eth. accounts[i-1],
"1234")
      }
};
```

3. **allBalances ():**

   Checks the balances of all accounts. All accounts and their ether balance is displayed.

```
function allBalances () {
    var totalBal = 0;
    for (var acctNum in eth. accounts) {
        var acct = eth. accounts[acctNum];
        var acctBal = web3.fromWei(eth. getBalance(acct),
"ether");
        totalBal += parseFloat(acctBal);
        console.log (“eth. accounts [" + acctNum + "]: \t"
+ acct + " \tbalance: " + acctBal + " ether");
    }
    console.log (“Total balance: " + totalBal + " ether");
};
```

4. **getFreq(lowerlimit, upperlimit)**:

   This is used to get the number of blocks mined by each user. During the mining process, many empty blocks do get generated, so the above function is used to filter only the blocks with transactions for each miner.

```
function getFreq(input1, input2){
    var m1 = 0;
    var m2 =0 ;
    var m3 =0;
    var m4 =0;
    miner_freq = new Array();
```

```
    for (var i=input1; input1<=input2; input1++){
        var transLen =
eth.getBalance(input1).transactions.length
        if (transLen>0){
            var ff = eth.getBlock(input1).miner;
            miner_freq.push(ff);
        }
      var len = miner_freq.length;
    for(var j=0;j < len; j++){
        if(miner_freq[j] ==
"0x7a34bc752c01209e71cf6b7b8264c27f9e70e84d")
        m1 = m1 + 1;
        if(miner_freq[j] ==
"0xe55d2f474e3254aa6f93fc8ef93d240e19e5cd40")
        m2 = m2 + 1;
        if(miner_freq[j] ==
"0xf91b66e17e7ef2df0fcb53fbee850e0ace4222aa")
        m3 = m3 + 1;
        if(miner_freq[j] ==
"0x8283176e72d7878466ca7b57a61044b8cc586374")
        m4 = m4 + 1;
}
console.log("miner1 "+m1);
console.log("miner2 "+m2);
console.log("miner3 "+m3);
```

```
console.log("miner4 "+m4);
};
```

4.2.1 Auxiliary functions in the Geth Console

Additional details for a few auxiliary functions used with geth console are:

I. Eth.blocknumber - gets the latest block on the node

II. Txpool.status – gets the status of the transaction pool on the node

III. Miner.getHashRate() – gets the current hash rate of the miner

IV. Eth.getBlock(blockNumber) – gets the content of the block

V. Net.peerCount – number of peers node is connected to.

VI. Admin.nodeInfo.enode – gets the enode string of the node

Parameters	Value	Description
M	15	The number of users
N	4	The number of miners
γ	2.5	The network scale-related parameter
k	0.2	The coefficient of $A(x) = kx$
T	2	The block time
$\|Q\|$	15	The number of transactions in each block
m	15	The initialized transaction number per unit of time
$\mathcal{R}$	1	The block reward
c	1	The average effective channel capacity
C	1	The dApp initiator's cost per unit of the transaction size
η_j	Uniformly distributed from $[0.0002, 0.0004]$	Miner t_j's hardware's amortized price per hash
v_i	Uniformly distributed from $[0.01, 0.05]$	User u_i's value per unit of time
R_i^u	Uniformly distributed from $[2, 5]$	Revenue of u_i if his initialized transaction is confirmed

Table 4-1 System Parameters used in the experiment

Chapter 5 : Analysis and Evaluation

This chapter provides various evaluation results and their discussion.

5.1 Hash Rate Ratio of miners vs probability of successful mining

We use 11 different experimental settings, with miner 1's hash rate ratio increasing from 0.01 to 0.91, while other miners' hash rate ratios are decreasing from 0.33 to 0.03. 100 nonempty blocks are mined for one experiment settings and we repeat 10 times.

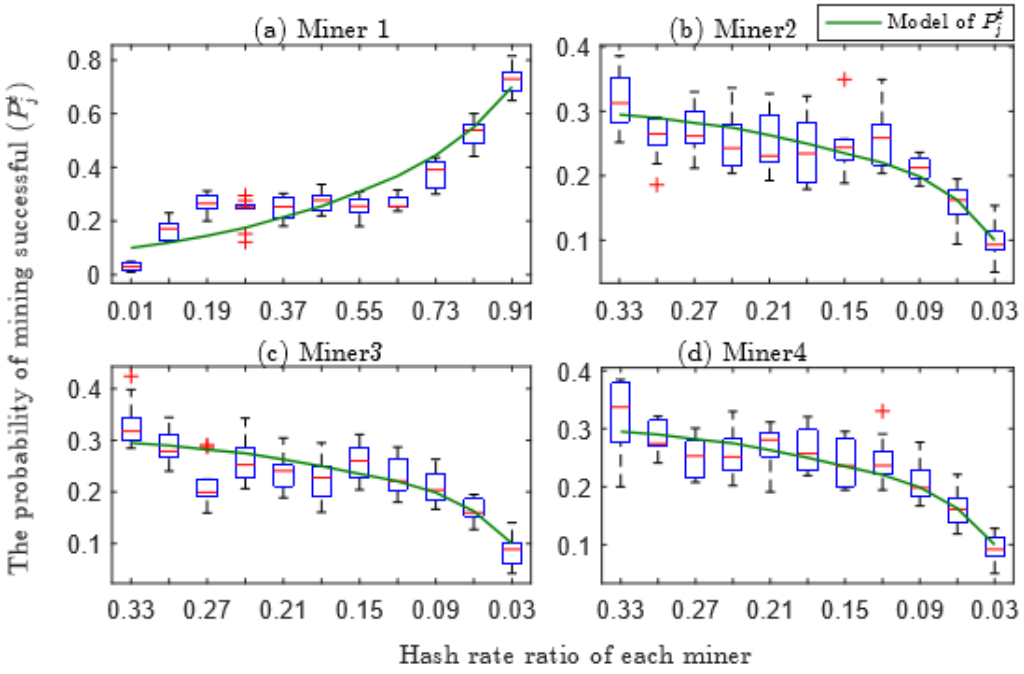

Figure 5-1. Hash Rate Ratio of Miner 1 (a), Miner 2 (b), Miner 3 (c) and Miner 4 (d) vs the probability of successful mining p_j^t

The comparison of the real experimental results and our proposed analytical model is show in Figure 5-1. As expected, there is not much difference between the real results and our analytical model. For example, in Figure 2(a), when hash rate ratio of miner 1 is 0.01, its experimental probability of mining successfully p_j^t is about 0.05. As the

hash rate ratio of miner 1 increases to 0.91, p_j^t rises to 0.75. The line of the model p_j^t shows a similar trend. It increases from 0.1 to 0.7 with the hash rate ratio increasing from 0.01 to 0.91. Figure 5-1(b), 5-1(c) and 5-1(d) also show an increasing trend of p_j^t as the hash rate ratio increases, both in experimental results and the model. This is because the probability that the miner successfully mines the block is directly proportional to its relative computing power when the block sizes are identical. Similarly, the trend of miner 2, miner 3 and miner 4 are identical, since their hash rate ratios are same.

5.2 Transaction fee density (ether) for Transaction 1 and Transaction 14 vs the waiting time

To validate our proposed model of waiting time (9), we set 21 different experiment settings, with transaction 1's transaction fee density f_1 increasing from 0 to 5 ethers with the interval 0.25, while other 14 transactions' transaction fee density are uniformly distributed among [0, 5]. The number of transactions in each block is 4. We conduct 6 times for each experiment setting.

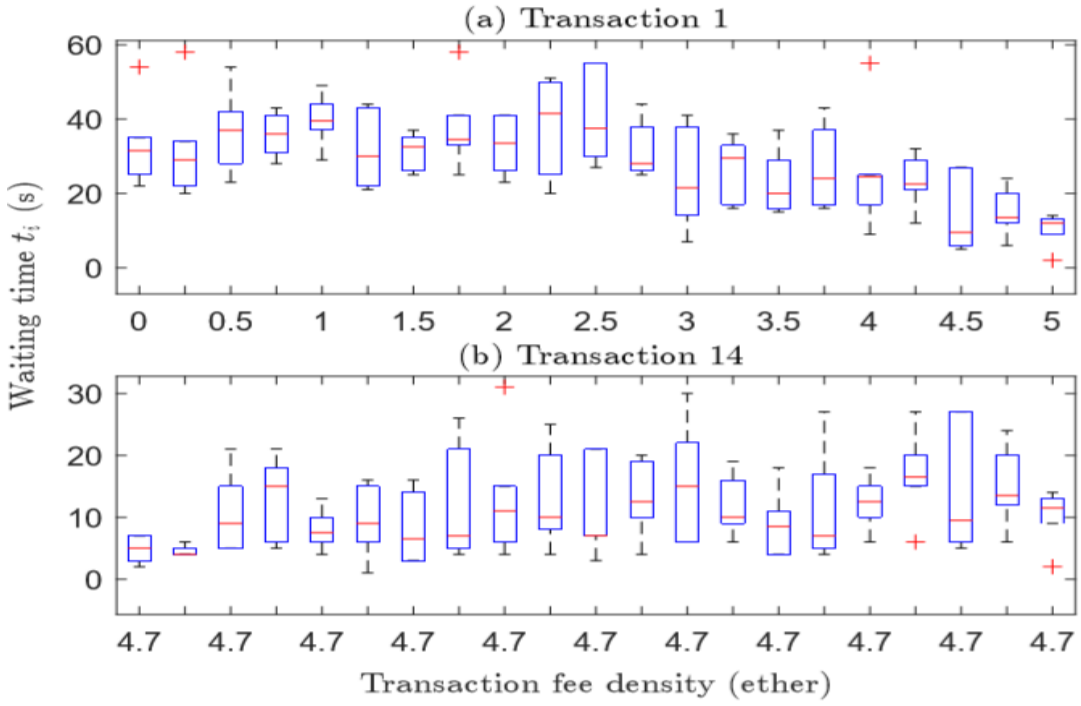

Figure 5-2 Transaction fee density (ether) for Transaction 1 and 14 vs the waiting time t_i in seconds

The comparison of experimental results of the waiting time among transaction 1 and another randomly selected one: transaction 14 is depicted in Fig 5-2.

We notice that the waiting time of transaction 1 is about 30s, even though its transaction fee density is 0. This is because we set the mining difficulty as 0x00 when setting up the ethernet chain. Also, we can see a decreasing trend of f_1 in Figure 5-2(a) as expected, from 30s to 15s which is because transaction 1's transaction fee density rises, and thus the portion that is larger than f_1 decreases. There is an increasing trend of f_{14} in Figure 5-2(b), from 7.5s to 10s. This is because transaction 1's fee density is increasing, and thus the portion that is larger than f_{14} rises.

5.3 Comparison between real experiment and model for transaction fee density vs waiting time

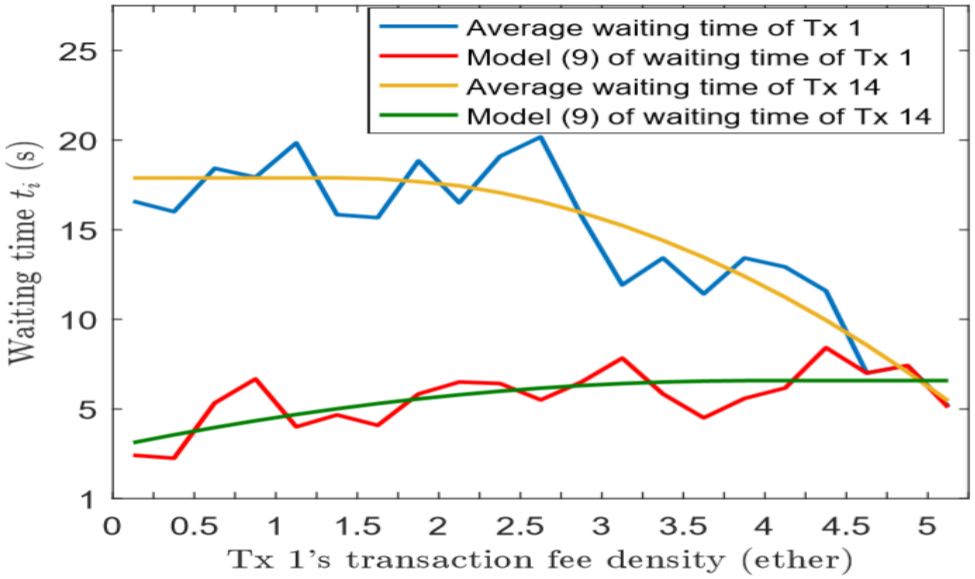

Figure 5-3. Transaction fee density of Transaction 1 and 14 vs waiting time t_i for the model and real experiment.

We compare experimental results of the average waiting time with the model of (9) in Figure 5-3. We can see there is not much difference between the real results and our analytical model, even though the average waiting time fluctuates a lot. For example, the average waiting of transaction 1 is decreasing from 18.2s to 7s, as transaction 1's fee density increases from 0 to 5, while other transactions' fee densities is kept same. This is due to waiting time of a transaction is directly related to the portion that is larger than its transaction fee density.

5.4 Numerical Results

To illustrate the impacts of different parameters from the proposed model on the performance. System parameters are set as Table 1. Notice that some of these parameters are varied according to the evaluation scenarios.

1) *The impact of the number of transactions in each block* $|Q|$:

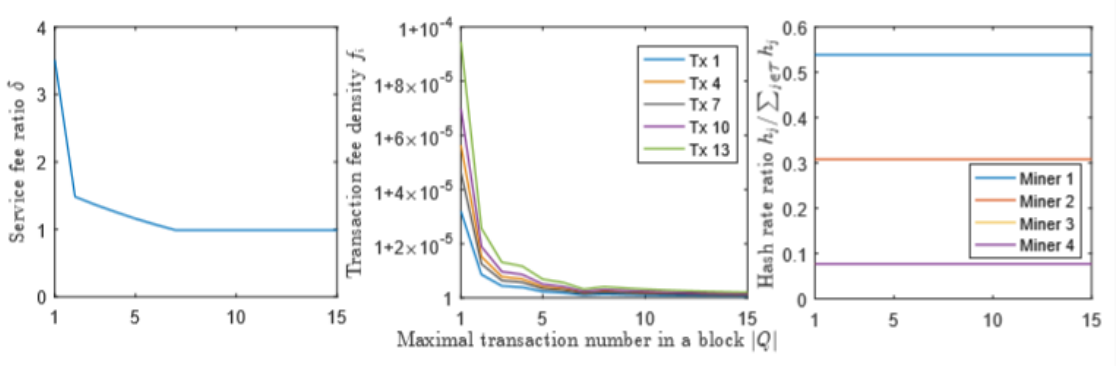

Figure 5-4. Three parties' strategies vs no. of transactions in each block

From Figure 5-4, we find that the service fee ratio δ decreases with the increase of the number of transactions in each block. For example, when $|Q|$ equals to 1, its service fee ratio δ is 3.59. As $|Q|$ increases to 15, δ decreased to 0.95. This is because the dApp initiator's optimal revenue from one block is fixed (other system parameter is fixed). When the number in a block rises, the initiator can still achieve this optimal revenue through lowering its service fee ratio.

Next, we see that for transaction fee density there is a decreasing trend. This is because when the number of transactions in a block is larger, the competition

among transactions is reduced. In this way, users do not need to set high transaction fee density to rise its competitive power.

Next, we see that the hash rate ratio is not affected by the number of transactions. For example, miner 1's hash rate ratio is 0.52, miner 2's hash rate ratio is 0.31, the hash rate ratio of miner 3 and miner 4 are same, i.e., 0.08. This is because the block size for all miners is identical, i.e., $|Q|$, $\eta_1 = 2 \times 10^{-4}, \eta_2 = 3 \times 10^{-4}, \eta_3 = \eta_4 = 4 \times 10^{-4}$ .In this scenario, miners' hash rate ratio is directly related with η_j.

2) *The impact of the transactions' arriving speed m:*

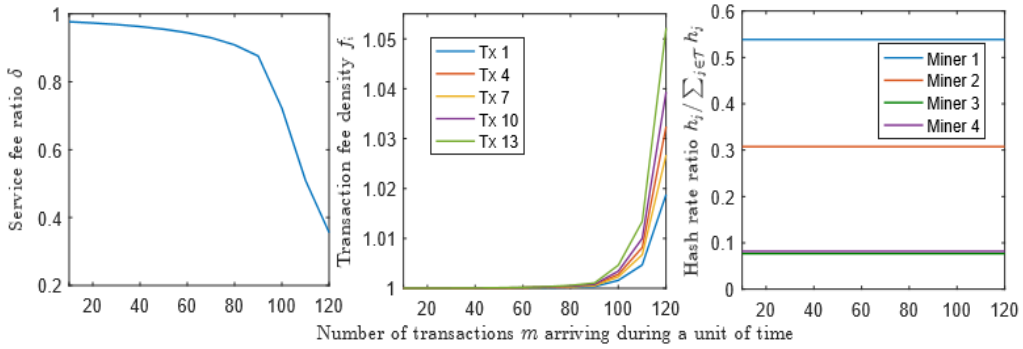

Figure 5-5. Three parties' strategies vs transaction's arriving speed m

We evaluate the impacts brought by the number of arriving transactions in a unit of time to three parties' strategies, and the results are shown in Figure 5-5.

We find that the service fee ratio δ decreases with the increase of the number of arriving transactions in unit of time. For example, when m equals to 10, its service fee ratio δ is 0.96. As m increases to 120, δ decreased to 0.35. This is because more transactions arrive in the dApp system, the dApp initiator can get the

optimal revenue through setting lower service fee, given that the dApp initiator's optimal revenue from one block is fixed (other system parameter is fixed).

Next, we see that there is an increasing trend of transaction fee density. This is because when the number of transactions in a block is larger, the competition among transactions is increased. In this way, users need to set high transaction fee density to raise their competitive power.

Next, we see that the hash rate ratio is not affected by the number of transactions. For example, miner 1's hash rate ratio is 0.52, miner 2's hash rate ratio is 0.31, the hash rate ratio of miner 3 and miner 4 are same, i.e., 0.08. The reason for this is same as mentioned above.

3) *The impact of the block time T*:

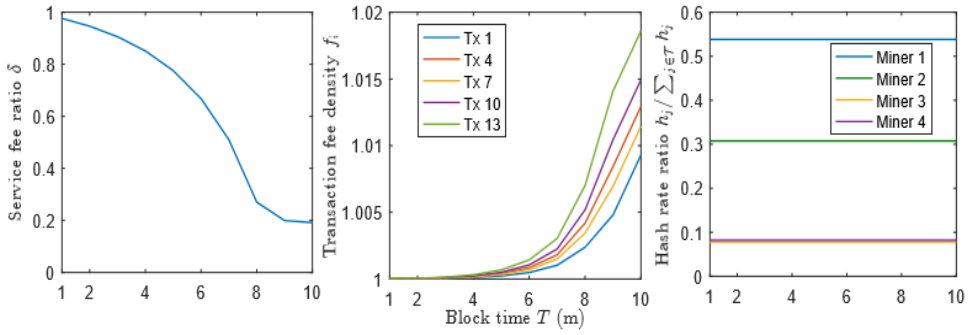

Figure 5-6. Three parties' strategies vs block time

We evaluate the impacts brought by the block time T to three parties' strategies, and the results are shown in Figure 5-6.

We find that the service fee ratio δ decreases with the increase of the number of arriving transactions in unit of time. For example, when T equals to 1, its service fee ratio δ is 0.98. As T increases to 10, δ decreased to 0.21. This is due to the fact the block time is longer, the revenue of users for a confirmed block is less

(the waiting cost is larger because the cost is positive which is related with T). To attract more users (transactions), service fee ratio should be smaller.

Next, we see that there is an increasing trend of transaction fee density. This is because the competition among transactions is increased since more transactions arrives during a block time. In this way, users need to set high transaction fee density to raise their competitive power.

The hash rate ratio is not affected by the number of transactions. For example, miner 1's hash rate ratio is 0.52, miner 2's hash rate ratio is 0.31, the hash rate ratio of miner 3 and miner 4 are same, i.e., 0.08. The reason for this is same as mentioned above.

4) *The impact of the block reward T*:

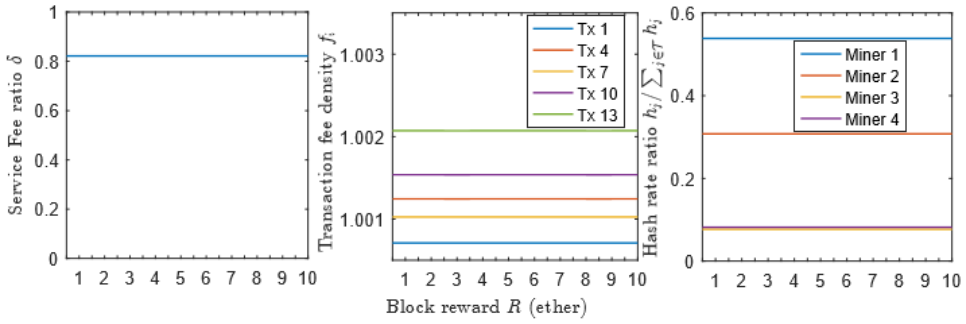

Figure 5-7. Three parties' strategies vs block reward

We evaluate the impacts brought by the block reward T to three parties' strategies, and the results are shown in Figure 5-7.

We find that three parties' strategies are not related to the block reward. For example, service fee ratio δ keeps 0.82 with the block reward varying from 1 to 10. The transaction fee density is 0.9995, 1.0013, 1.001, 1.0016, 1.0021 respectively for transaction 1, transaction 4, transaction 7, transaction 10 and transaction 13.

miner 1's hash rate ratio is 0.52, miner 2's hash rate ratio is 0.31, the hash rate ratio of miner 3 and miner 4 are same, i.e., 0.08. This is because users' and the dApp initiator's utility function has no relationship with the block reward and miners' hash rate ratio is only related with their cost per hash rate η_j in our mechanism.

Chapter 6 : Conclusion

In this thesis, we have investigated the utility-based strategy choice instruction, for supporting dApp to work efficiently in proof-of-work based public blockchain networks. We have adopted the three-stage Stackelberg game model to jointly study the utility maximization of the dApp initiator, users and the miners. Through backward induction, we have derived the unique Nash equilibrium point of the game. The existence and uniqueness of the Stackelberg equilibrium has been proved analytically. We have performed extensive experiments to validate the proposed analytical model. Moreover, we have conducted numerical simulations to evaluate the network performance, which provide insights for the dApp initiator to choose suitable system parameters.

Chapter 7: References

[1] Seppälä, J., 2016. The role of trust in understanding the effects of blockchain on business models.

[2]https://brage.bibsys.no/xmlui/bitstream/handle/11250/2472245/17527_FULLTEXT.pdf?sequence=1&isAllowed=y

[3] https://ieeexplore.ieee.org/document/8466786

[4] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Self-published Paper, May 2008, [Online]. https://bitcoin.org/bitcoin.pdf.

[5] Ziyao Liu, Nguyen Cong Luong, et al,”A Survey on Applications of Game Theory in Blockchain”, Mar 2019

[6] H. von Stackelberg, Market Structure and Equilibrium: 1st Edition Translation into English, Bazin, Urch & Hill, Springer 2011, XIV, 134 p

[7] https://en.wikipedia.org/wiki/Stackelberg_competition

[8] (2018) Blockchian dapps. [Online]. Available: https://blockchainhub. net/decentralized-applications-dapps/

[9] (2018) Decentralized application wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Decentralized application

[10] S. Raval, Decentralized Applications: Harnessing Bitcoin’s Blockchain Technology. ” O’Reilly Media, Inc.”, 2016.

[11] (2018) Bittorrent. [Online]. Available: https://www.bittorrent.com/

[12] (2018) Popcorn time. [Online]. Available: https://popcorn-time.to/

[13] (2018) Bitmessage. [Online]. Available: https://bitmessage.org/wiki/ Main Page

[14] (2018) State of the dapps. [Online]. Available: https://www.stateofthedapps.com/

[15] (2018) Whitepaper of cryptokitties. [Online]. Available: https://drive. google.com/file/d/1soo-eAaJHzhw XhFGMJp3VNcQoM43byS/view

[16] (2018) Top 5 ethereum dapps. [Online]. Available: https://www. coindesk.com/top-5-ethereum-dapps-daily-active-users

[17] (2018) Mining in ethereum. [Online]. Available: https: //ethereum-homestead.readthedocs.io/en/latest/mining.html

[18] (2018) Proof of work vs proof of stake. [Online]. Available: Available: https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/

[19] (2018) Transaction size. [Online]. Available: https://bitzuma.com/posts/ making-sense-of-bitcoin-transaction-fees/

[20] (2018) Mining in ethereum. [Online]. Available: https://ethereum-homestead.readthedocs.io/en/latest/mining.html

[21] R. Bowden∗, H.P. Keeler∗, A.E. Krzesinski† and P.G. Taylor∗ : “Block arrivals in the Bitcoin blockchain”,Jan 2018

[22] J. Kang, Z. Xiong, D. Niyato, P. Wang, D. Ye, and D. I. Kim, “Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks,” IEEE Wireless Communications Letters, 2018.

[23] P. R. Rizun, “A transaction fee market exists without a block size limit,”
Block Size Limit Debate Working Paper, 2015.

[24] Y. Jiao, P. Wang, D. Niyato, and K. Suankaewmanee, “Auction mechanisms in cloud/fog computing resource allocation for public blockchain networks,” arXiv preprint arXiv:1804.09961, 2018.

[25] G. Debreu, “A social equilibrium existence theorem,” Proceedings ofthe National Academy of Sciences, vol. 38, no. 10, pp. 886–893, October 1952

[26] R. D. Yates et al., “A framework for uplink power control in cellular radio systems,” IEEE Journal on selected areas in communications, vol. 13, no. 7, pp. 1341–1347, May 1995.

[27] F. T. Lui, “An equilibrium queuing model of bribery,” Journal ofpolitical economy, vol. 93, no. 4, pp. 760–781, Aug 1985.

[28] J. D. C. Little, “A proof for the queuing formula l = λw,” Operations

Res, vol. 9, no. 7, pp. 383–387, May 1961.

[29] http://ethdocs.org/en/latest/introduction/what-is-ethereum.html.

[30] http://ethdocs.org/en/latest/ethereum-clients/go-ethereum/index.html#go-ethereum

[31] https://en.wikipedia.org/wiki/Docker_(software)

[32] https://www.ethereum.org/cli

[33] https://github.com/ethereum/go-ethereum/wiki/Connecting-to-the-network

[34] https://docs.docker.com/config/containers/resource_constraints/

Chapter 8: Biographical Information

Vaibhav Soni was born in Rajasthan, India in 1994. He had received his B.E in Computer Science and Engineering from PES- School of Engineering, Visveswaraya Technological University, Bangalore, India in 2017. He graduates from The University of Texas at Arlington, Arlington, Texas in May 2019 with Master of Science in Computer Science. He has worked as a Software Developer Intern at ClearBlade Inc. Austin, Texas and aspires to work at a leading tech company as a Software Engineer.