Sync or Fork: Node-Level Synchronization Analysis of Blockchain

Qin Hu[1], Minghui Xu[2], Shengling Wang[3](✉), and Shaoyong Guo[4]

[1] Indiana University - Purdue University Indianapolis, Indianapolis, USA
qinhu@iu.edu
[2] The George Washington University, Washington, D.C., USA
mhxu@gwu.edu
[3] Beijing Normal University, Beijing, China
wangshengling@bnu.edu.cn
[4] Beijing University of Posts and Telecommunications, Beijing, China
syguo@bupt.edu.cn

Abstract. As the cornerstone of blockchain, block synchronization plays a vital role in maintaining the security. Without full blockchain synchronization, unexpected forks will emerge and thus providing a breeding ground for various malicious attacks. The state-of-the-art works mainly study the relationship between the propagation time and blockchain security at the systematic level, neglecting the fine-grained impact of peering nodes in blockchain networks. To conduct a node-level synchronization analysis, we take advantage of the large deviation theory and game theory to study the pull-based propagation from a microscopic perspective. We examine the blockchain synchronization in a bidirectional manner via investigating the impact of full nodes as responders and that of partial nodes as requesters. Based on that, we further reveal the most efficient path to speed up synchronization from full nodes and design the best synchronization request scheme based on the concept of correlated equilibrium for partial nodes. Extensive experimental results demonstrate the effectiveness of our analysis.

Keywords: Block synchronization · Large deviation theory · Game theory · Correlated equilibrium

1 Introduction

Since the appearance of Bitcoin [1], cryptocurrency as the killer application of blockchain piques substantial attention from the whole society to the underlying distributed ledger technology. Research on blockchain from all walks of life indicates its great potential and versatility. It is reported that the blockchain market size over the globe reaches \$3 billion in 2020 and is expected to surge to \$39.7 billion by 2025 [2].

As the infrastructure of blockchain systems, the peer-to-peer network consisting of peering nodes supports the most important operations of information dissemination and exchange, including both control and data messages.

To maintain the consistent recognition of the main chain, the synchronization of newly generated blocks among all nodes becomes extremely important. Otherwise, unexpected forks will emerge, which might be exploited by malicious clients to achieve various attacks, such as double spending and selfish mining, and can even lead to the breakdown of a blockchain system.

To enable the synchronization of blockchain, there exist five types of block propagation mechanisms [3], i.e., advertisement, header sending, unsolicited push, relay network, and push-advertisement hybrid. Focusing on the behaviors of nodes propagating block information, we can summarize them as pull-based and push-based. In the pull-based propagation, nodes with timely information of the blockchain, termed as full nodes, respond to the requests of updating block information from neighboring nodes, named as partial nodes, which can achieve block synchronization cost-efficiently in an on-demand manner. While in the push-based one, any node receiving the newly generated block automatically pushes this piece of information to neighbors, which can synchronize the blockchain network quickly but will cause unnecessary communication among nodes. Other works about blockchain synchronization mainly study the relationship between the propagation time and blockchain security at the systematic level [4–7], neglecting the fine-grained impact of peering nodes in blockchain networks.

In this paper, we study the blockchain synchronization in pull-based propagation from a microscopic perspective, using the large deviation theory and game theory to investigate different roles of peering nodes in synchronizing block information. This suggests the feature of *node-level* analysis of this work. Besides, our research is *bidirectional*, which captures the feature of impacts on block synchronization from two main types of nodes in the blockchain, i.e., the full node as the responder and the partial node as the requester. Specifically, we reveal clues about three critical questions: *How will the full node's response capability affect the synchronization? How to efficiently reduce its negative effect on synchronization? And how should the partial node to actively achieve the synchronization?*

In summary, our contributions in this work include the following three aspects:

- The impact of full nodes on synchronization is quantitatively characterized by the concept of response failure rate, which straightforwardly uncovers the synchronization probability of connected partial nodes.
- The negative impact of full nodes on synchronization can be fast eliminated via increasing the decay speed of the response failure rate, and the derived expression of the decay speed indicates that enlarging the response capacity related parameter is more efficiently than improving the response rate. This paves a clear path to facilitate synchronization from full nodes.
- The optimal synchronization scheme for the partial node is established based on the concept of correlated equilibrium, where a Node Synchronization (NS) problem is formulated to guarantee that the partial node can get synchronized without unnecessary cost or redundant response from full nodes.

The remaining of this paper is organized as follows. In Sect. 2, we investigate the most related work on blockchain synchronization. Then we introduce the node-level synchronization model for both the full node and the partial node in Sect. 3, where the full node's response capability is further analyzed in Sect. 4 while the best synchronization request mechanism for the partial node is presented in Sect. 5. All theoretical analysis are evaluated in Sect. 6. And finally, we conclude the whole paper in Sect. 7.

2 Related Work

Similar to traditional distributed systems, there are three levels of synchrony of blockchain networks, namely synchronous, partially synchronous, and asynchronous. As the representative blockchain application, Bitcoin whitepaper [1] provides an initial analysis on its security against forks and double-spending attacks with an oversimplified model. Since 2015, Bitcoin consensus algorithm has been thoroughly investigated considering three levels of synchrony [8,9]. Garay *et al.* [8] formalize the Bitcoin consensus within a fully synchronous network. Persistence and liveliness are proved to be guaranteed hinging on the synchronous setting. Pass *et al.* [9] show that Bitcoin consensus satisfies consistency and liveliness in a partially synchronous network, but consistency cannot be satisfied in an asynchronous network.

As the most critical factor affecting blockchain synchronization, the propagation time of control messages and data messages is investigated to reveal how it affects blockchain security against various attacks, such as forks, double spending, and selfish mining, and how to mitigate the corresponding vulnerabilities. The propagation time is shown as the primary cause for blockchain forks [4]. In response, researchers propose three methods to speed up propagation: minimizing verification, pipelining block propagation, and increasing connectivity. Sompolinsky and Zohar [5] study the relation between higher transaction rate and the vulnerability to double-spending attacks, which shows that increasing block size and block generation rate can improve the throughput, but will increase the propagation time so that even weaker attackers can launch double-spending attacks. Besides, the selfish mining is investigated in a realistic setting where propagation time is taken into account [6], indicating that it becomes easier with increasing propagation delay. For PoS-based consensus, Kang *et al.* [7] propose a Stackelberg game based incentive mechanism to encourage miners to propagate blocks, enabling lower propagation delay and higher security level.

For the propagation mechanism in blockchain networks, five popular categories are summarized in [3], including advertisement, header sending, unsolicited push, relay network, and push-advertisement hybrid. Early on, the advertisement-based propagation is adopted by Bitcoin, which has a two-round message exchange procedure. Afterward, Bitcoin resorts to the header propagation to avoid using *inv* messages. In unsolicited push propagation, miners directly broadcast newly-mined blocks. The relay network, adopted by FIBRE [10], BloXroute [11], and Geeqchain [12], is to distribute relay nodes globally to which miners can connect to and exchange information at a high speed. However, relay nodes are criticized for introducing centralization to blockchain. Ethereum adopts push and advertisement hybrid propagation by which a node can automatically push messages to $\sqrt{n}$ nodes and advertises messages to neighboring nodes simultaneously [13].

In summary, existing works about blockchain synchronization focus on macroscopically investigating blockchain protocol to figure out the relationship between propagation time and security or propose new propagation mechanisms. However, in this paper, we study the blockchain synchronization from a microscopic and node-level perspective, using the large deviation theory and game theory to depict blockchain nodes precisely and investigate how nodes' capability affect synchronization.

3 System Model

In this paper, we assume that full nodes are homogeneous in terms of information request and response performance. Thus, we can shed light on the synchronization status of the whole blockchain system via studying the response capability of any specific full node. And all partial nodes are also assumed to be similar in terms of interacting with full nodes to get synchronized. As full nodes and partial nodes play different roles in blockchain synchronization, we introduce their models separately in the following.

3.1 Response Model of the Full Node

Considering that the requests of updating block information from partial nodes arrive at the full node randomly, we assume that this stochastic event is a Poisson process with arrival rate λ, which is inspired by the typical model of packet arrival process in communication networks [14]. It usually takes some time for the full node to respond and send out the latest block information since the node might be busy on handling other tasks, which can also be assumed as a Poisson process with response rate μ. To guarantee that the full node can finish responding to the requests from partial nodes most of the time, we assume $\mu > \lambda$. However, even with this condition, there might still exist some cases where the full node fails to respond.

To investigate this issue, we define the number of synchronization requests arrived at the full node and that the node can respond during time period $(t - 1, t)$ as a_t and r_t, respectively, where $t \in \mathbb{N}^*$. Then we can describe the request queue at the full node as

$$Q_t = (Q_{t-1} + a_t - r_t)^+,$$

where $(\cdot)^+$ denotes the positive part of the inside expression.

Next, we focus on the cumulative arrival and response process, denoted as $A_t = a_1 + \cdots + a_t$ and $R_t = r_1 + \cdots + r_t$, respectively. Thus, the length of the request queue until time t at the full node, defined as L_t, will be

$$L_t = A_t - R_t. \tag{1}$$

Generally speaking, since $\mu > \lambda$, one may expect that L_t would be negative, making it pointless with the definition of queue length. However, due to the randomness of the arrival and response process, the queue length can become positive, which may even overwhelm the response capability of the full node, leading to the failure of responding synchronization requests. To prepare for the worst case of response failure in blockchain, we focus on the maximum possible queue length at the full node when $t \to \infty$, which is defined as $\mathcal{L} = \sup_{t>0} L_t$, and further investigate the possibility of the request queue being over-length, i.e., $\mathcal{L} > \Gamma$, where Γ is defined as follows:

Definition 1 (Response capacity). *The response capacity Γ of the full node is the longest queue of synchronization requests that it can process without any failure.*

According to Definition 1, we can know that if $\mathcal{L} \leq \Gamma$, the fulll node can handle all synchronization requests successfully. But if $\mathcal{L} > \Gamma$, the request queue is too long for the full node to handle, which will make the partial nodes sending block synchronization requests fail to achieve the distributed consistency. To analyze this important event, we introduce the following definition:

Definition 2 (Response failure rate). *The response failure rate is the probability that the longest synchronization request queue arrived at the full node, i.e., $\mathcal{L}$, exceeds its response capacity Γ, denoted as $P(\mathcal{L} > \Gamma)$.*

With the help of $P(\mathcal{L} > \Gamma)$, we can capture the full node's failure of responding to block synchronization requests in a quantitative manner, which provides us a more straightforward clue about the synchronization status of the neighboring partial nodes. Based on this index, we can make adjustment or countermeasure in time to avoid unpredictable loss brought by the asynchronous blockchain information among partial nodes, which will be analyzed in Sect. 4.

3.2 Synchronization Model of the Partial Node

We assume that the number of partial nodes in a blockchain network is N, and each of them has direct access to multiple full nodes to obtain block synchronization information. Specifically, for any partial node, we denote the set of full nodes it has direct connections as $\mathcal{M} = \{M_i\},\ i \in \{1, \cdots, m\}$, where $m \in \mathbb{N}^*$ is the number of full nodes. And the above-defined response failure rate of these full nodes can be denoted as $P_i,\ i \in \{1, \cdots, m\}$.

For a cautious partial node, it may send the synchronization request to all connected full nodes so as to obtain a higher successful synchronization probability. Thus, the synchronization failure event can only happen to this partial node when all full nodes failed to respond with the latest block information, which means that the synchronization failure probability is $\prod_{i=1}^{m} P_i$. And accordingly, the successful synchronization probability of this partial node is $1 - \prod_{i=1}^{m} P_i$.

While in a more general case, a normal partial node might need to seriously consider where to send the synchronization request. First, sending the request costs communication resource, and thus generously sending the request to all available full nodes can bring too much burden on the resource consumption for the partial node. What's more, with the assumption that all full nodes have the same new information of the blockchain, it would be enough for the partial node to receive at least one response and thus other redundant responses become a waste. With this in mind, we can see that wisely sending the synchronization request is vital for the partial node, which will be elaborated in Sect. 5.

4 Response Failure Analysis

In this section, we analyze the response failure rate $P(\mathcal{L} > \Gamma)$ in detail. We first focus on the derivation of its decay speed $I(x)$, based on which two critical factors impacting the systematic response are discussed.

We first let $\Gamma = lx$ with $x > 0$. Then according to the Cramér's theorem [15], for large l, there exists $P(\mathcal{L} > \Gamma) = P(\mathcal{L} > lx) \approx exp(-lI(x))$, which indicates that the probability of $\mathcal{L} > lx$ will decay exponentially with the rate $I(x)$ when $l \to \infty$. In detail, we have

$$\lim_{l\to\infty} \frac{1}{l} \log P(\mathcal{L} > lx) = -I(x), \tag{2}$$

where $I(x)$ is the rate function with the following expression

$$I(x) = \inf_{t>0} t\Phi^*(\frac{x}{t}). \tag{3}$$

According to the large deviation theory and the calculation process in [16], we can have the expression of $I(x)$ as:

$$I(x) = x \ln \frac{\mu}{\lambda}. \tag{4}$$

As we mentioned earlier, $I(x)$ reveals the decay speed of response failure rate $P(\mathcal{L} > \Gamma)$. In other words, the larger $I(x)$, the sharper decrease of $P(\mathcal{L} > \Gamma)$, and thus the more successful the block synchronization for the requested partial nodes. With this in mind, we desire to enlarge $I(x)$ as much as possible. On one hand, from the above expression of $I(x)$, one can tell that it is linearly increasing with the response capacity related parameter x when the ratio of the response rate μ to the arrival rate λ is fixed. On the other hand, if x is given, we can see $I(x)$ is logarithmically correlated to $\frac{\mu}{\lambda}$. Therefore, theoretically speaking, increasing x is more effective to improve $I(x)$ than increasing μ/λ, which will be numerically analyzed in Sect. 6.

Considering that x and $\frac{\mu}{\lambda}$ are two main factors impacting the value of $I(x)$, we study them further in the following. As x is based on the response capacity Γ

and the arrival rate λ is a system-wide parameter which cannot be adjusted, we mainly focus on Γ and μ since they are more controllable from the perspective of the full node. In the following, we investigate how to set Γ and μ to meet some specific system-performance requirements on response failure rate. To this end, we first denote a *response failure tolerance degree* as $\epsilon \in (0, 1]$, which acts as the constraint for the failure rate $P(\mathcal{L} > \Gamma)$. And then we introduce the following two definitions.

Definition 3 (Effective response capacity). *The effective response capacity $\Gamma^*(\epsilon)$ is the minimum capacity that the full node needs to provide to enforce that the response failure rate will never greater than ϵ, i.e.,*

$$\Gamma^*(\epsilon) = \min\{\Gamma : P(\mathcal{L} > \Gamma) \leq \epsilon\}.$$

Definition 4 (Effective response rate). *The effective response rate $\mu^*(\epsilon)$ is the minimum response rate requirement for the full node to guarantee that the response failure rate will never greater than ϵ, i.e.,*

$$\mu^*(\epsilon) = \min\{\mu : P(\mathcal{L} > \Gamma) \leq \epsilon\}.$$

Further, we have the following theorems to present the specific results of $\Gamma^*(\epsilon)$ and $\mu^*(\epsilon)$.

Theorem 1. *For $\epsilon \in (0, 1]$ and $\mu > \lambda$, we can calculate $\Gamma^*(\epsilon)$ as:*

$$\Gamma^*(\epsilon) = -\frac{\ln \epsilon}{\ln \frac{\mu}{\lambda}}.$$

Proof. As we mentioned at the beginning of this section, for $l \to \infty$, we have $P(\mathcal{L} > \Gamma) \approx e^{-lI(x)}$. Then it comes to $e^{-lI(x)} \leq \epsilon$ according to Definition 3. Besides, based on (4) and $\Gamma = lx$, we can prove that the value of $\Gamma^*(\epsilon)$ is $-\frac{\ln \epsilon}{\ln \frac{\mu}{\lambda}}$.

Theorem 2. *For $\epsilon \in (0, 1]$ and $\mu > \lambda$, we can calculate $\mu^*(\epsilon)$ as:*

$$\mu^*(\epsilon) = \lambda e^{-\frac{\ln \epsilon}{\Gamma}}.$$

Proof. Similar to the proof of Theorem 1, due to $P(\mathcal{L} > \Gamma) \approx e^{-lI(x)} \leq \epsilon$, we can have $lx \ln \frac{\mu}{\lambda} \geq -\ln \epsilon$, which leads to the result of $\mu^*(\epsilon)$.

5 Correlated Equilibrium Based Node Synchronization Mechanism

As mentioned earlier, the synchronization of one certain partial node is collectively completed by the surrounding full nodes, which heavily depends on how many of them the partial node requests. In fact, each full node has a particular response capability with respect to the synchronization request, which is well captured by the response failure tolerance degree introduced in the above section, and it takes some cost for the partial node to send the synchronization request to a specific full node. For a reasonable and intelligent partial node, it is essential to work out an efficient and effective strategy to select the subset of full nodes as synchronization request targets. In other words, *given different* ϵ_i $(i \in \{1, \cdots, m\})$ *of all connected full nodes, how should the partial node make decisions on whether to send the blockchain synchronization request to each of them?*

To solve this problem, we first define that the decision strategy of the partial node is $\mathbf{p} = (p_1, \cdots, p_m)$ with $p_i \in \{0, 1\}$, where 0 (or 1) denotes not sending (or sending) the synchronization request to the full node M_i. From the perspective of the partial node, the ultimate goal of this decision is to guarantee that it can obtain the up-to-date information of the main chain from at least one full node. Thus, the profit of deciding whether to send the request to one specific full node M_i is jointly affected by the decisions of sending to other full nodes, which can be defined as

$$\phi_i(\mathbf{p}) = \frac{p_i(1 - \epsilon_i)}{\sum_{j=1}^{m} p_j(1 - \epsilon_i)}.$$

Note that in the case of $\mathbf{p} = \mathbf{0}$, we define $\phi_i(\mathbf{p}) = 0$.

With C_i denoting the cost of sending the request to M_i, we can define the utility of this decision as

$$U_i(\mathbf{p}) = \alpha_i \phi_i(\mathbf{p}) - p_i C_i, \tag{5}$$

where $\alpha_i > 0$ is a scalar parameter.

On one hand, as a utility-driven decision maker, the partial node desires to obtain an optimal utility for each individual decision about one specific full node, which is collectively affected by the decision vector $\mathbf{p}$ about all full nodes and can be described by the following game-theoretic concept named *correlated equilibrium.*

Definition 5 (Correlated equilibrium). *Denote the strategy space as* $\mathcal{V} = \{0, 1\}$ *with the size of* $V = 2$ *and a probability distribution over the space* $\mathcal{V}^m$ *as* $G(\mathbf{p})$. *Then* $G(\mathbf{p})$ *is a correlated equilibrium if and only if* $G(\mathbf{p})$ *makes that for any decision* $p_i, p_i' \in \mathcal{V}$, *there exists*

$$\sum_{\mathbf{p}_{-i} \in \mathcal{V}^{m-1}} G(p_i, \mathbf{p}_{-i}) \Big(U_i(p_i, \mathbf{p}_{-i}) - U_i(p_i', \mathbf{p}_{-i}) \Big) \geq 0,$$

where $\mathbf{p}_{-i} = (p_1, \cdots, p_{i-1}, p_{i+1}, \cdots, p_n)$ *denotes other decisions except for* p_i.

The above definition implies that under the correlated equilibrium $G(\mathbf{p})$, there is no motivation for the partial node to deviate from the strategy p_i about sending the request to M_i given other strategies $\mathbf{p}_{-i}$. In other words, the partial node can only obtain the maximized utility with respect to the individual decision via selecting p_i according to the decision vector $\mathbf{p}$ sampled from the correlated equilibrium $G(\mathbf{p})$. It is obvious that there may exist various correlated equilibria meeting the above-defined constraint.

On the other hand, the partial node cares about the overall utility of all decisions about all surrounding full nodes since it reflects the general synchronization status of this partial node, which can be calculated as $\sum_{\mathbf{p}\in\mathcal{V}^m} G(\mathbf{p}) \sum_{i=1}^{m} U_i(\mathbf{p})$. Therefore, we can obtain the best correlated equilibrium for the partial node considering the global optimization goal, which is summarized as the following Node Synchronization (NS) problem.

NS Problem:

$$\max: \sum_{\mathbf{p}\in\mathcal{V}^m} G(\mathbf{p}) \sum_{i=1}^{m} U_i(\mathbf{p}) \tag{6}$$

$$\text{s.t.}: G(\mathbf{p}) \geq 0, \ \forall \mathbf{p} \in \mathcal{V}^m, \tag{7}$$

$$\sum_{\mathbf{p}\in\mathcal{V}^m} G(\mathbf{p}) = 1, \tag{8}$$

$$\sum_{\mathbf{p}_{-i}\in\mathcal{V}^{m-1}} G(p_i, \mathbf{p}_{-i})\Big(U_i(p_i, \mathbf{p}_{-i}) - U_i(p_i', \mathbf{p}_{-i})\Big) \geq 0, \forall p_i, p_i' \in \mathcal{V}. \tag{9}$$

Obviously, the above NS problem is an optimization problem with respect to the variable $G(\mathbf{p})$, where the optimization object (6) is to maximize the overall expected utility for all decisions, constraint (7) is a natural requirement for the probability distribution, constraint (8) refers to that the sum of all probability distribution is 1, and the last one (9) is directly obtained from the definition of correlated equilibrium to achieve individual utility maximization. Besides, via scrutinizing the NS problem, one can find that it is exactly a linear programming problem with respect to the probability distribution $G(\mathbf{p})$. In fact, there exist a lot of efficient algorithms to solve the linear programming problem with polynomial time complexity, such as interior point and simplex-based algorithms.

6 Experimental Evaluation

In this section, we first numerically analyze the key factor impacting the response failure rate $P(\mathcal{L} > \Gamma)$, i.e., the decay speed $I(x)$. Further, the proposed correlated equilibrium based node synchronization mechanism is validated to demonstrate its effectiveness. Specifically, all experiments are carried out using a laptop running with 2.7 GHz Dual-Core Intel Core i5 processor and 8 GB memory. And for the sake of statistical confidence, we report average values of all experimental results via repeating each experiment for 20 times.

6.1 Numerical Analysis of Response Failure Rate

We first plot $I(x)$ changing with the response capacity related parameter x and the response rate μ in Fig. 1. In particular, we use the difference between μ and λ, i.e., $\mu - \lambda$, to capture the impact of $\frac{\mu}{\lambda}$ in (4) on $I(x)$ for easy understanding. Specifically, we set $x \in [0, 1]$, $\lambda = 3$ and $\mu - \lambda \in [0, 10]$.

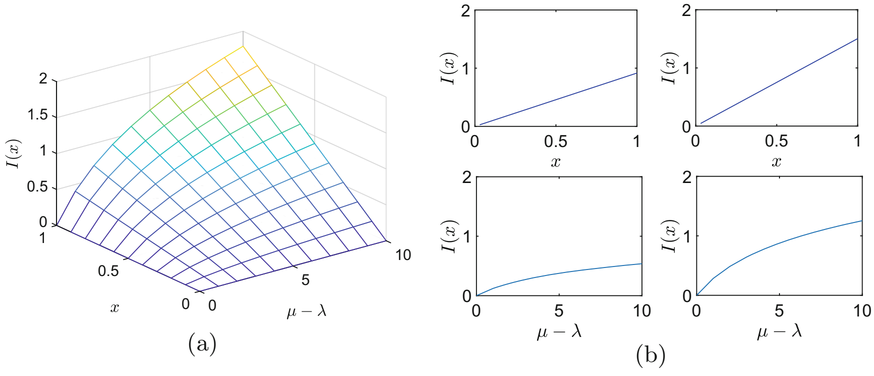


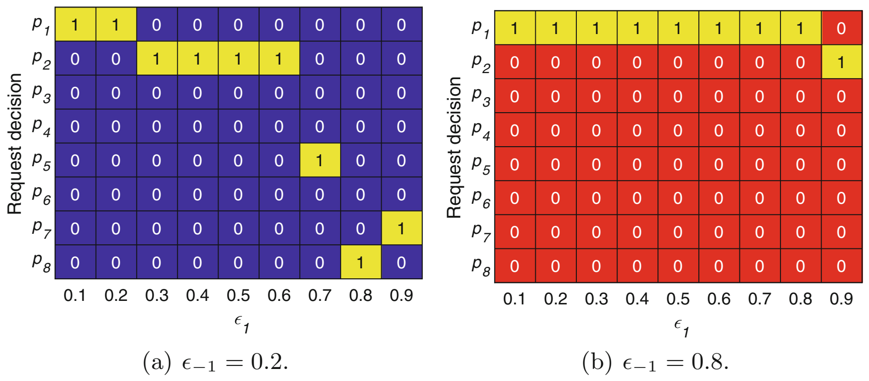
Fig. 1. Decay speed of the synchronization failure rate.

It can be seen that $I(x)$ increases with the larger x and $\mu - \lambda$, which means that we can achieve a higher decay speed for the response failure rate via either improving the response capacity $\Gamma = lx$ or enhancing the full node's response rate μ given a specific arrival rate λ. Besides, via comparing the first and second lines of subfigures in Fig. 1(b), one can tell that the decay speed has different changing trends with respect to x and $\mu - \lambda$, where the increasing x can lead to linear change while the increase of $\mu - \lambda$ can only bring logarithmic variation. Thus, we can conclude that raising the response capacity can achieve a lower response failure rate more efficiently.

6.2 Evaluation of Node Synchronization Mechanism

Next, we explore the effectiveness of our proposed node synchronization mechanism in Sect. 5. In detail, we take the case of $m = 8$ as an example and focus on the decision of sending the synchronization request to the full node M_1 who has a varying response failure tolerance degree $\epsilon_1 \in (0, 1)$ with an interval of 0.1. Other parameters are set as $\alpha_i = 10, C_i = 5$. The request sending decisions are reported in Fig. 2 with two representative cases, where the response failure tolerance degrees of all other full nodes, i.e., M_2 to M_m, are the same and fixed as $\epsilon_{-1} = 0.2$ and 0.8. It is obvious that the request decision vectors in two cases are very different. In the case of $\epsilon_{-1} = 0.2$ in Fig. 2(a), p_1 is 1 until ϵ_1 is larger than others, which means that sending the request to M_i is a good choice until its response failure rate is higher than others. And similarly, when $\epsilon_{-1} = 0.8$ as shown in Fig. 2(b), p_1 keeps to be 1 except for $\epsilon_1 = 0.9$ which is larger than response failure rates of other full nodes.

Finally, we examine the maximized total utility in the NS problem and evaluate its performance under the impacts of the scalar parameter α_i and the cost C_i. Here the number of full nodes is still set to $m = 8$. The experimental results are reported in Fig. 3. From Fig. 3(a), one can see that the maximized total utility keeps the same as zero until $\alpha_i = 5$, which is because we set $C_i = 5$ in this experiment and the profit $\phi_i \in [0, 1]$. This means that only when the profit parameter



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
p_1	1	1	0	0	0	0	0	0	0
p_2	0	0	1	1	1	1	0	0	0
p_3	0	0	0	0	0	0	0	0	0
p_4	0	0	0	0	0	0	0	0	0
p_5	0	0	0	0	0	0	1	0	0
p_6	0	0	0	0	0	0	0	0	0
p_7	0	0	0	0	0	0	0	0	1
p_8	0	0	0	0	0	0	0	1	0

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
p_1	1	1	1	1	1	1	1	1	0
p_2	0	0	0	0	0	0	0	0	1
p_3	0	0	0	0	0	0	0	0	0
p_4	0	0	0	0	0	0	0	0	0
p_5	0	0	0	0	0	0	0	0	0
p_6	0	0	0	0	0	0	0	0	0
p_7	0	0	0	0	0	0	0	0	0
p_8	0	0	0	0	0	0	0	0	0

Fig. 2. Synchronization request decision of the partial node.

α_i is larger than the cost, can the partial node obtain a positive overall utility. While within Fig. 3(b), it is shown that the maximized utility first increases with C_i and then decreases when C_i is too large. This is because with a lower C_i, the partial node can still obtain a better utility via strategically making the request decision; while when the cost is too high, even the best decision cannot compensate the high resource consumption in request sending process.

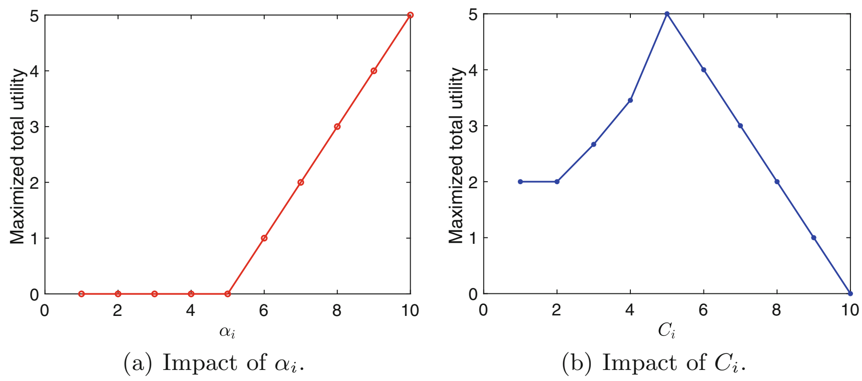


Fig. 3. Maximized total utility of the partial node changing with α_i and C_i.

7 Conclusion

In this paper, we take advantage of the large deviation theory and game theory to study the blockchain synchronization in the pull-based propagation from a microscopic perspective. To be specific, we investigate the fine-grained impacts of peering nodes in synchronizing block information at the node level. On one hand, the full node as the synchronization responder is analyzed based on the queuing model, which reveals the most efficient path to speed up synchronization via increasing the response capacity. On the other hand, the partial node is inspected as the requester, where the best synchronization request scheme is designed using the concept of correlated equilibrium. Extensive experiments are conducted to demonstrate the effectiveness of our analysis.

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Technical report, Manubot (2019)
2. Blockchain market. https://www.marketsandmarkets.com/Market-Reports/blockchain-technology-market-90100890.html. Accessed 30 May 2020
3. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 3–16 (2016)
4. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: IEEE P2P 2013 Proceedings, pp. 1–10. IEEE (2013)
5. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in bitcoin. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 507–527. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47854-7_32
6. Göbel, J., Keeler, H.P., Krzesinski, A.E., Taylor, P.G.: Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay. Perform. Eval. **104**, 23–41 (2016)
7. Kang, J., Xiong, Z., Niyato, D., Wang, P., Ye, D., Kim, D.I.: Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks. IEEE Wirel. Commun. Lett. **8**(1), 157–160 (2018)
8. Garay, J., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: analysis and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 281–310. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_10
9. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 643–673. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_22
10. Bicoin relay network. https://github.com/bitcoinfibre/bitcoinfibre. Accessed 30 May 2020
11. Klarman, U., Basu, S., Kuzmanovic, A., Sirer, E.G.: bloXroute: a scalable trustless blockchain distribution network whitepaper. IEEE Internet Things J. (2018)
12. Conley, J.P.: The Geeq project white paper (2018)
13. Wüst, K., Gervais, A.: Ethereum eclipse attacks. Technical report, ETH Zurich (2016)
14. Cao, J., Cleveland, W., Lin, D., Sun, D.: Internet traffic tends to poisson and independent as the load increases. Technical report, Technical report, Bell Labs (2001)
15. Ganesh, A.J., O'Connell, N., Wischik, D.J.: Big Queues. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-39889-9
16. Wang, S., Wang, C., Hu, Q.: Corking by forking: Vulnerability analysis of blockchain. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 829–837. IEEE (2019)