`Consensus latency of PoW blockchains

Ke Wang & Hyong S. Kim
kewang1@andrew.cmu.edu, kim@ece.cmu.edu

***Abstract*—We analyze the consensus latency of Proof of Work (PoW) blockchains. A block *reaches consensus* if it is included in every miner's longest chain. We define consensus latency in this paper as the duration from when a block is mined to when that block first reaches consensus. Understanding consensus latency helps us reason about when to confirm blocks for a series of confirmation rules. We find that blocks could reach consensus fast in practical scenarios. For instance, in a propagation network with maximum network delay delta, a block could reach consensus in about 1.3 deltas on average. Contrary to common expectations, average consensus latency even decreases when mining becomes easier. We establish an analytical model to understand the relation between consensus latency and various system parameters in various scenarios. We also build a blockchain simulator to validate the accuracy of our analysis. Our estimation closely tracks the simulation results.**

Keywords—PoW blockchain, consensus, latency, analysis, simulation

I. INTRODUCTION

Blockchain is a consensus protocol that maintains a distributed ledger. It was first proposed in Bitcoin [1] and has been adopted by various applications [2–6] to reach consensus in a permissionless setting. Participants of a blockchain application rely on the confirmed blocks for consistent record-keeping. Existing applications adopt a simple rule for block confirmation, referred to as the *k-block* rule. A miner confirms a block if that block is in the miner's longest chain and at least k blocks have been mined on top of it. A confirmed block implies that all transactions inside that block are also confirmed. A miner could choose k depending on the value of the transactions inside the block. It is recommended to set k to 6 in Bitcoin [1] and 15 in Ethereum [7]. The *confirmation latency* of a block is the duration from when the block is mined to when it is confirmed. Once a block is confirmed, there is a non-zero probability, known as the *confirmation reversal probability*, that the confirmed block later gets reversed.

We define the *consensus state* of a block in this paper. A block *reaches consensus* or is in the *consensus state* if it is included in every miner's longest chain. A block reaching consensus is different from a block getting confirmed. When a block is confirmed according to the k-block rule, it may not be in the consensus state yet. Similar to a confirmed block getting reversed later, a block may leave the consensus state. This occurs when the adversary later releases a longer chain that does not contain this block. For blocks that ever reach consensus, we define the *consensus latency* as the time from when that block is mined to the time when it reaches consensus for the first time. In PoW blockchains, mining starts from a seed block, known as the *genesis block*. The genesis block is hardcoded into the protocol and is in the consensus state by definition. The *level* of a block b is its distance in terms of the number of blocks from the genesis block. A *pioneer block* is the first block known to any honest miner in a level. Non-pioneer blocks are also referred to as *fork blocks*. In an ideal scenario where there is no fork blocks, consensus latency is equal to the maximum network delay. By the time the pioneer block propagates to every miner, it reaches consensus. In practical scenarios where fork blocks exist, consensus latency could be much longer. The competition between the pioneer block and multiple fork blocks at the same level could only be resolved after higher level blocks are mined.

We analyze consensus latency in this paper. Understanding consensus latency is a first step to reason about when miners could confirm blocks for various confirmation rules. It is recently proposed [8] that a block should not be confirmed before the miner infers it to be in the consensus state. The corresponding confirmation rules have the advantage of lower confirmation reversal probability compared to existing k-block rule. Consensus latency thus denotes the minimum time before a block could be confirmed. Analysis of consensus latency helps us reason about confirmation latency.

We first analyze the consensus latency when no adversary is present. We start with a special network, the *exact* Δ*-delay network*. Any block takes exact Δ time before it is received by any other miner. We then analyze consensus latency in the *practical* Δ*-delay network*. The maximum propagation delay for a new block is still Δ, but majority miners receive the new block much sooner. Surprisingly, we find that consensus latency does not increase with block rate. As more fork blocks are mined with higher block rate, one would expect the average consensus latency to increase. Contrary to this expectation, we observe that the average consensus latency decreases with higher block rate. For instance, if the average block rate increases from 1.5 blocks per Δ to 3 blocks per Δ, the expected consensus latency decreases from 2.3Δ to 2.28Δ in the exact Δ-delay network. Similarly, the expected consensus latency decreases from 1.26Δ to 1.23Δ in the practical Δ-delay network. We identify two reasons that cause the decrease of consensus latency. First, higher block rate reduces interarrival time of pioneer blocks. When fork blocks exist at level l, no block at level l could reach consensus before the pioneer block at level $l+1$ is mined. Higher block rate reduces the interarrival time between the pioneer blocks at level l and $l+1$. Second, although on average more fork blocks exist when block rate increases, majority miners adopt either the pioneer block or the first fork block. We refer to additional fork blocks beyond the first fork block as *late fork blocks*. In the exact Δ-delay network, the pioneer block of a level always propagates faster than fork blocks at the same level. Most miners thus adopt the pioneer block. For a fork block f_b,

only the miner of that block receives f_b before it receives the pioneer block. In the practical Δ-delay network, by the time a late fork block is mined, majority of miners already receive either the pioneer block or the first fork block. Late fork blocks do not significantly increase the consensus latency. Based on these observations, we establish an analytical model to study the relationship between consensus latency and block rate. We prove an upper bound of the expected consensus latency in the exact Δ-delay network. We give a close estimation of the expected consensus latency in the practical Δ-delay network. We build a blockchain simulator to validate the accuracy of our analysis.

We then analyze the consensus latency in the presence of the adversary. A miner is *honest* if it follows the protocol. Otherwise, it is an *adversary*. A block is an *honest block* if mined by an honest miner. We assume the adversary performs malicious mining and controls the network at the same time. The adversary decides when an honest miner receives a block. The only constraint is that the delay from when an honest block is mined to when it is received by every honest miner is bounded by Δ. We first show that existing attacks such as selfish mining attack and double spend attack do not extend the consensus latency. We then analyze the case of irrational adversary with its only goal to extend the consensus latency. We design a *balance attack* under such circumstance. We demonstrate that when an irrational adversary is present, consensus latency could increase exponentially with block rate. We again validate our analysis through simulation.

Our analysis reveals that blocks could reach consensus quickly, especially in the common scenarios with fast network propagation and rational adversaries. Low consensus latency makes it feasible for consensus-aware confirmation rules, where a miner needs to first infer the block to reach consensus [8]. Blocks could reach consensus in as few as 1.23Δ. More importantly, consensus latency decreases as block rate increases. This enables us to configure a higher block rate for scalability purposes.

II. BACKGROUND

A. Blockchain and definitions of the analysis

In a blockchain system, each node maintains a linearly ordered log of transactions, known as the *ledger*. Transactions are grouped into *blocks* that are linked together to form a *blockchain*. A block points to its *parent block* through the *prevHash* field in the block header. We use f_b to denote the parent block of b. The prevHash field contains the hash of the entire previous block and uniquely identifies a single parent block. Block b is an *ancestor block* of b' if b is a parent block of b' or a parent block of an ancestor block of b'. In this case, we also say b' is a *descendent block* of b. A block b is both an ancestor block and a descendent block of itself.

A node that follows the protocol is called an *honest miner*. A block is an *honest block* if it is mined by an honest miner. Otherwise, it is a *malicious block*. A block is in the *consensus state* when it is included in all honest miners' blockchains. The *genesis block* is the first block of any blockchain. The genesis block is hard-coded in the protocol and thus in the consensus state by definition. The *length* of a blockchain is the number of non-genesis blocks in the blockchain. The *level* of a block is its distance in terms of the number of blocks from the genesis block. We say an honest block b at level l is a *pioneer block* if it is the first block at level l that is mined or received by an honest miner. Denote this pioneer block is first mined or received by an honest miner in round t_l.

We adopt a similar model as described in Dembo et al [9]. Let $\mathcal{T}(t)$ denote the tree formed by all mined blocks by time t. This includes all the blocks mined by the adversary and honest miners. Due to the network delay, different honest miners may have different partial views of the tree. Let $\mathcal{T}_i(t)$ denote the subtree of $\mathcal{T}(t)$ that belongs to ith honest miner m_i at time t. Let $C_i(t)$ denote the longest chain in $\mathcal{T}_i(t)$. $C_i(t)$ is interpreted as the longest chain in the local view of the ith honest miner. We say the system is in the *total-consensus state* at time t when all honest miners have the same longest blockchain. All $C_i(t)$s are the same at time t. Let $l_i(t)$ denote the length of $C_i(t)$ at time t. Let $l_{short}(t)$ denote the lowest $l_i(t)$ at time t. $l_{short}(t) = \min_{i=1,\dots,n} l_i(t)$.

B. Model and parameters

There are n honest miners with an equal mining power. We adopt the discrete-time model based on rounds. A *round* is the smallest time unit in our analysis. Each miner can compute one hash solution of the PoW puzzle per round. There are s rounds per second. s represents the physical capability of mining hardware. The network message delay between any two miners is bounded by Δ rounds. Denote p as the probability that a hash solution of the PoW puzzle succeeds. Denote η as the block generation rate of honest miners. We have $\eta = nsp$. The adversary can compute $\rho n/(1-\rho)$ hashes towards solving the PoW puzzle per round. The parameters of interest are:

- n: number of honest miners
- s: number of rounds per second
- Δ: maximum network delay in the unit of rounds.
- p: the probability of a hash solution succeeds.
- η: average block generation rate per second of honest miners, $\eta = nsp$.
- ρ: the ratio of malicious mining power over total mining power. Since there are n hashes computed by honest mining power, the adversary could compute $\rho n/(1-\rho)$ hashes per round.

All honest miners start mining from the genesis block. At the beginning of every round, a miner receives messages from the network. A message can be either new transactions from the clients or recently discovered blocks broadcast by other miners. If the miner receives a block that is in a chain longer than its local longest chain, the miner adopts the new block and starts mining on this new chain. The miner then includes any valid, outstanding transactions into a candidate block and computes the hash of the candidate block. If the hash satisfies the proof-of-work requirement, the candidate block is valid, and the miner mines a new block. The miner broadcasts this block to the entire network at the end of this round.

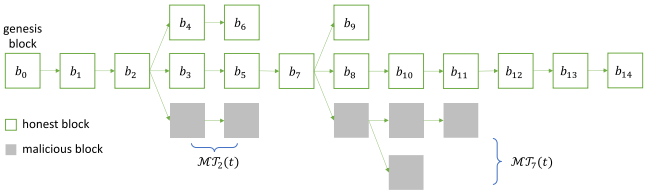

Figure 1. Two malicious subtrees in the block tree

C. Adversary

We assume a strong adversary with the following four capabilities to interfere with the honest miners:

- $\mathcal{A}_1$: The adversary has zero network delay from all the honest miners. The adversary could thus observe $\mathcal{T}(t)$ instantly. It arbitrarily chooses to mine on any block from $\mathcal{T}(t)$.
- $\mathcal{A}_2$: The adversary could arbitrarily delay the messages from an honest miner, but no more than Δ rounds.
- $\mathcal{A}_3$: The adversary could choose when to release its malicious blocks.
- $\mathcal{A}_4$: The adversary could force an honest miner to switch mining from its current longest chain to a different chain of equal length at any time. This may not be achievable in practice and is thus considered the worst attack scenario. For instance, in Bitcoin, miners keep the old chain in case of a tie. $\mathcal{A}_4$ gives an advantage to the adversary. This is in line with our goal to prove an upper bound of the confirmation reversal probability.

We organize different malicious blocks into *malicious subtrees*. We sort all honest blocks according to the time when they are mined. For the ith honest block b_i, define *malicious subtree* $\mathcal{MT}_i(t)$ as the subtree of $\mathcal{T}(t)$ that grows from block b_i and consists of all the malicious blocks that can be reached from b_i without going through another honest block. We say the malicious subtree $\mathcal{MT}_i(t)$ is *rooted* at the ith honest block. In the example in Figure 1, the malicious blocks form two subtrees: $\mathcal{MT}_2(t)$ and $\mathcal{MT}_7(t)$.

D. Consensus latency – system and miner perspective

There are two notions of consensus latency. From the system's perspective, consensus latency is defined as the duration from the time when the pioneer block at level l is mined, to the time a block at level l reaches consensus. Let $T_{\tau,sys}$ denote the expected consensus latency from the system perspective. From an individual miner's perspective, once it observes a block b at level l, it is important for him to know when he could confirm or discard that block. We thus define the miner-perspective consensus latency for a miner m_h. It is the latency from the time when miner m_h first observes a block b at level l, to the time when some block b' at level l reaches consensus. If b and b' is not the same block, the miner-perspective consensus latency could be interpreted as the latency before m_h could decide to discard block b. Let $T_{\tau,miner}$ denote the expected consensus latency from the miner's perspective. In both definitions, the end point of the measurement is the time when some block first reaches consensus. In practice, it is hard for a miner to know whether a block reaches consensus. One way for miner to infer consensus was thus proposed in [8]. The additional imposed latency of the inference protocol is bounded by 2Δ. We thus focus on the $T_{\tau,sys}$ in this paper.

We can show that $T_{\tau,miner} \leq T_{\tau,sys} \leq T_{\tau,miner} + \Delta$. By definition, the pioneer block at level l is mined in round t_l. Denote miner m_h mines or receives a block at level l for the first time in round $t_{h,l}$. We have $t_l \leq t_{h,l} \leq t_l + \Delta$. Denote a block b' at level l reaches consensus in round r. We have $T_{\tau,miner} = E[r - t_{h,l}] \leq E[r - t_l] = T_{\tau,sys}$. Similarly, $T_{\tau,sys} \leq T_{\tau,miner} + \Delta$. In the rest of the paper, we focus our analysis on $T_{\tau,sys}$.

III. Analysis of $T_{\tau,sys}$ with no adversary

A. $T_{\tau,sys}$ in the exact Δ-delay network

The exact Δ-delay network has been commonly cited as the worst network in terms of blockchain growth [10-12]. The *growth rate* of the blockchain held by an honest miner is defined as the number of blocks added to the chain per unit of time. The exact Δ-delay network is also leveraged by the adversary in selfish mining attacks [13] and double-spend attacks [12]. We thus first analyze $T_{\tau,sys}$ under this network model. We analyze $T_{\tau,sys}$ at any given level l_0. The definitions and analysis in the rest of this subsection are regarding level l_0.

Definitions

- *base block*: blocks at level l_0. Sort all base blocks by the time they are mined, and denote them as $bb_1, bb_2, \ldots$
- *class of block*: we classify all blocks at levels l_0 or higher based on the corresponding base blocks. A block b belongs to class i if it is bb_i or is a descendent block of bb_i.
- *class of a chain*: a blockchain belongs to class i if it contains the base block bb_i. If the chain does not contain any base block yet, that chain is said to be in class 0.
- *class of a miner*: a miner belongs to class i if its local longest chain C_i is in class C_i.

Analysis outline: When a pioneer block b at level $l \geq l_0$ is mined by m_h, denote it is in class x. If no base block reaches consensus yet, then the only base block that may possibly reach consensus before $t_l + \Delta$ is bb_x. This is because m_h is in class x until at least round $t_l + \Delta$. We say miner m_i is an *ally miner* in round t_l if the following two requirements are met.

- m_i is in class x.
- If m_i's chain at round t_l, $C_i(t_l)$, contains blocks up to level $h < l$, then the pioneer blocks at levels from $h + 1$ to l all belong to class x.

The probability for bb_x to reach consensus by $t_l + \Delta$ depends on the number of *ally miners* in round t_l. The mining of ally miners does not prevent bb_x from reaching consensus by round $t_l + \Delta$. Not all miners in class x are ally miners. Consider the example shown in Figure 2. There are five miners. The pioneer block b at level l is mined by m_1 in round t_l. Miner m_1, m_2 and m_3 currently belong to class x and the rest belong to class x'. Miner m_1 and m_2 are ally miners. Although m_3 is in class x, it only contains blocks up to level $l - 2$. As the pioneer block b' at level $l - 1$ is in class x', m_3 is not an ally miner.

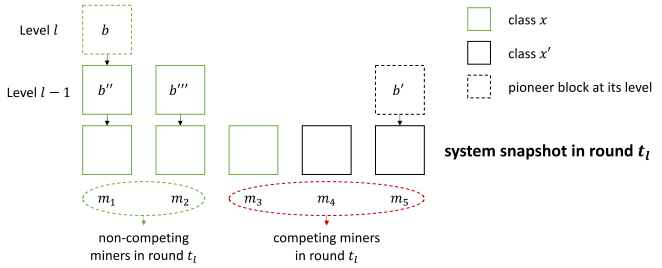

Figure 2. Miner m_3 is not an ally miner.

Indeed, if m_3 mines a block after it receives b' but before $t_l + \Delta$, bb_x does not reach consensus by round $t_l + \Delta$. We refer to the miners that are not ally miners as *competing miners*. An important property of the ally miners is proven in the following Lemma 1.

Lemma 1. When a pioneer block in class x is mined in round t_l, any ally miner in round t_l will always be in class x until at least round $t_l + \Delta$.

Proof. We prove by contradiction. Assume an ally miner m_α in class x is converted to another class $x' \neq x$ in round $t_l + \tau$ $(0 \leq \tau \leq \Delta)$. The only way such a conversion could happen is when m_α receives a block b' in class x' in round $t_l + \tau$. Since $t_l + \tau \leq t_l + \Delta$, m_α will not receive any block at level l or higher. Denote b' is at level l' and m_α's chain contains block up to level l_a in round t_l. We have $l_a < l' < l$. By definition of the ally miner, b' is not the pioneer block at level l'. Moreover, the pioneer block at level l', b'' is in class x. Since m_α receives b'' before b', it will not convert to class x' when it sees block b', contradicting the assumption.

Intuitively, bb_x is more likely to reach consensus if there are more ally miners in round t_l. We use the number of ally miners to describe the notion of *progress* towards a base block reaching consensus. We introduce the following definitions.

- $S_{u,i}$: The set of the system states where a pioneer block is just mined and there are i ally miners.
- T_i: When the system is in a state $s \in S_{u,i}$ in round t_l and no base block reaches consensus yet, let $E[T_{s\to FIN}]$ denote the expected latency from round t_l to the time when a base block reaches consensus. Let T_i denote the maximum expected latency among all the states in $S_{u,i}$, i.e., $T_i = \max_{s\in S_{u,i}} E[T_{s\to FIN}]$.

The following Lemma 2 formalizes our intuition.

Lemma 2. $\Delta \leq T_{n-1} \leq \cdots \leq T_1$.

The proof is given in the Appendix.

When the first base block bb_1 is mined in round t_{l_0}, bb_1 is in class 1. Only one miner (i.e., the miner that mines bb_1) belongs to class 1. By definition, the system is in a state in $S_{u,1}$. We have $T_{\tau,sys} \leq T_1$. We could obtain an upper bound of $T_{\tau,sys}$ by estimating T_1. We prove an upper bound of T_k in the rest of the section. We express T_k conditioning on different cases and establish a set of equations. Solving these equations leads to an accurate estimation of $T_{\tau,sys}$.

Analysis of T_{n-1}: Denote miner m_1 mines a pioneer block at level $l \geq l_0$ in round t_l. There are in total $n-1$ ally miners including m_1 in round t_l. Denote they are in class x and the single competing miner is m_n. We express T_{n-1} conditioning on what happens in the next Δ rounds, $(t_l, t_l + \Delta]$.

1) In case CS_1, m_n does not mine a block at level l in these Δ rounds. Let p_1 denote the probability that CS_1 occurs.
2) In case $CS_{2,i}$ $(1 \leq i \leq \Delta)$, m_n mines a block at level l in these Δ rounds, and a pioneer block at level $l+1$ is mined in round $t_l + i$. We further divide $CS_{2,i}$ into two subcases.
   a) Case $CS_{2,i,1}$: the pioneer block at level $l+1$ is not mined by m_n. Let $p_{2,i}$ denote the probability of $CS_{2,i,1}$.
   b) Case $CS_{2,i,2}$: the pioneer block at level $l+1$ is mined by m_n. Let $q_{2,i}$ denote the probability of $CS_{2,i,2}$.
3) In case CS_3, m_n mines a block at level l in these Δ rounds, and no pioneer block at level $l+1$ is mined by round $t_l + \Delta$. Let p_3 denote the probability of CS_3.

In CS_1, the base block bb_x will reach consensus by round $t_l + \Delta$. In $CS_{2,i,1}$ and $CS_{2,i,2}$, the pioneer block at level $l+1$ is mined in round $t_l + i$. Thus $t_{l+1} = t_l + i$. In $CS_{2,i,1}$, all ally miners in round t_l are also ally miners in round t_{l+1}. The expected latency from round t_{l+1} to the time a base block reaches consensus is thus again bounded by T_{n-1}. In $CS_{2,i,2}$, there is only one ally miner in round t_{l+1}, i.e., miner m_n. All other miners belong to class x in round t_{l+1}. The expected latency from t_{l+1} to the time a base block reaches consensus is bounded by T_1. Finally, in CS_3, no pioneer block at level $l+1$ has been mined by round $t_l + \Delta$. In round $t_l + \Delta$, all miners' blockchains have the same length. There are $n-1$ miners in class x and one miner in a different class x'. It takes another $1/np$ rounds on average before the pioneer block at level $l+1$ is mined. The new pioneer block is in class x with probability $n - 1/n$ and in class x' with probability $1/n$. Therefore, the expected latency from round $t_l + \Delta$ to the time a base block reaches consensus is bounded by $1/np + ((n-1)/n)T_{n-1} + (1/n)T_1$. Combining the three cases, we have

$$T_{n-1} \leq p_1\Delta + \sum_{i=1}^{\Delta} p_{2,i}(i + T_{n-1}) + q_{2,i}(i + T_1) + p_3\left(\Delta + \frac{1}{np} + \frac{n-1}{n}T_{n-1} + \frac{1}{n}T_1\right) \quad (1)$$

We then bound the probabilities of each case. We prove a series of lemmas in the Appendix.

Lemma 3. $p_1 \geq (1-p)^\Delta$.

Lemma 4. $\sum_{j=1}^{i}(p_{2,j} + q_{2,j}) \geq (1-p_1)(1-(1-p)^i)$.

Lemma 5. $p_1 + \sum_{i=1}^{\Delta} p_{2,i} + p_3 \geq \sum_{i=1}^{\Delta} X_i + (1-p)^\Delta + \Delta p(1-p)^{2\Delta-1}$, where X_i is given by the following.

$$X_i = (1-p)^{i-1}p[(1-p)^i(1-(1-p)^{\Delta-i}) + ip(1-p)^{i-1}]$$

Lemma 6. $p_3 \leq \Delta p(1-p)^{2\Delta-1}$

With these lemmas, we could rearrange the above expression and compute the bound on T_{n-1} as in the following Lemma 7.

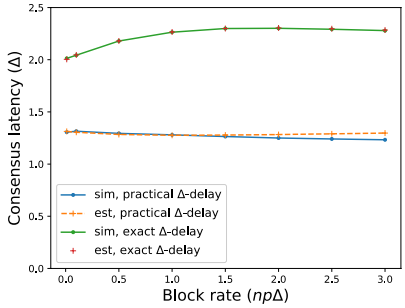

Figure 3. Simulation Results.

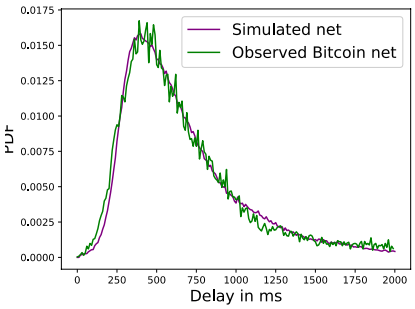

Figure 4. Block propagation in Bitcoin.

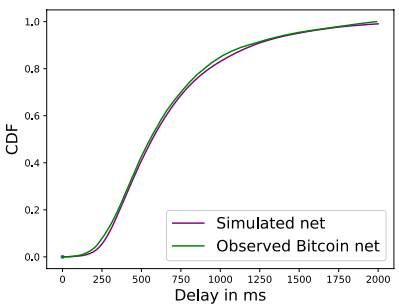

Figure 5. Simulated network

Lemma 7. $T_{n-1} \le AT_{n-1} + (1-(1-p)^{\Delta} - A)T_1 + B$, where A, B are coefficients that can be calculated given parameters n, p, Δ.

$$A = \sum_{i=1}^{\Delta} X_i + \Delta p(1-p)^{2\Delta-1}\frac{1}{np}$$

$$B = (1-p)^{\Delta}\Delta + (1-(1-p)^{\Delta})^2/p + \frac{1}{np}\Delta p(1-p)^{2\Delta-1}$$

Analysis of T_k: We apply the same approach to analyze T_k. Denote miner m_1 mines a pioneer block at level $l \ge l_0$ in round t_l. There are in total k ally miners including m_1 in round t_l. Denote ally miners are in class x. We express T_k conditioning on what happens in $(t_l, t_l + \Delta]$.

1) In case CS_1, none of the $n-k$ competing miners mines a block at level l in these Δ rounds. Let p_1 denote the probability that CS_1 occurs.

2) In case $CS_{2,i}$ $(1 \le i \le \Delta)$, at least one competing miner mines a block at level l in these Δ rounds, and a pioneer block at level $l+1$ is mined in round $t_l + i$. We further divide $CS_{2,i}$ into two subcases.

   a) Case $CS_{2,i,1}$: the pioneer block at level $l+1$ is mined by a miner that is an ally miner in t_l. Let $p_{2,i}$ denote the probability of $CS_{2,i,1}$.

   b) Case $CS_{2,i,2}$: the pioneer block at level $l+1$ is mined by a miner that is a competing miner in t_l. Let $q_{2,i}$ denote the probability of $CS_{2,i,2}$.

3) In case CS_3, at least one competing miner mines a block at level l in these Δ rounds, and no pioneer block at level $l+1$ is mined by round $t_l + \Delta$. Let p_3 denote the probability of CS_3.

In CS_1, the base block bb_x will reach consensus by round $t_l + \Delta$. In $CS_{2,i,1}$, a pioneer block at level $l+1$ is mined in round $t_{l+1} = t_l + i$; We further divide $CS_{2,i,1}$ into s $(1 \le s \le n-k)$ sub-cases. For subcase $CS_{2,i,1,s}$, s competing miners mine a block at level l from round $t_l + 1$ to $t_l + \Delta$. Let $p_{2,i,s}$ denote the probability of $CS_{2,i,1,s}$. We have $p_{2,i} = \sum_{s=1}^{n-k} p_{2,i,s}$. In $CS_{2,i,1,s}$, there are $n-s$ ally miners in round t_{l+1}. Similar to the reasoning in T_{n-1}, the expected latency from round t_{l+1} to the time some base block reaches consensus is no more than T_{n-s}.

In $CS_{2,i,2}$, the expected latency from t_{l+1} to the time some base block reaches consensus is no more than T_1. Finally, we also divide CS_3 into s $(1 \le s \le n-k)$ subcases. For subcase $CS_{3,s}$, s competing miners mine a block at level l from round $t_l + 1$ to $t_l + \Delta$, and no pioneer block at level $l+1$ is mined by round $t_l + \Delta$. Let $p_{3,s}$ denote the probability of $CS_{3,s}$. We have $p_3 = \sum_{s=1}^{n-k} p_{3,s}$. In $CS_{3,s}$, the expected latency from round $t_l + \Delta$ to the time a base block reaches consensus is bounded by $1/np + ((n-s)/n)T_{n-s} + (s/n)T_s$. Iterating through all cases, we have

$$T_k \le p_1\Delta + \sum_{s=1}^{n-k}\left[\sum_{i=1}^{\Delta} p_{2,i,s}(i + T_{n-s})\right] + \sum_{i=1}^{\Delta} q_{2,i}(i + T_1) + \sum_{s=1}^{n-k} p_{3,s}\left(\Delta + \frac{1}{np} + \frac{n-s}{n}T_{n-s} + \frac{s}{n}T_s\right) \quad (2)$$

We then bound the probabilities $p_1, p_{2,i,s}, q_{2,i}, p_{3,s}$. Similar to the analysis of T_{n-1}, we prove a series of lemmas in the Appendix. For simplicity and clarity, let $p_{\Delta} = (1-p)^{\Delta}$ and $p_{1b} = \Delta p(1-p)^{\Delta-1}$ denote the probability that a miner mines 0 or 1 block in Δ rounds.

Lemma 8. ${p_1} \ge {p_{\Delta}}^{n-k}$.

Lemma 9. $\sum_{j=1}^{i}(p_{2,j} + q_{2,j}) \ge (1-p_1)(1-(1-p)^i)$.

Lemma 10.

$${p_1} + \sum_{i=1}^{\Delta} p_{2,i,1} + p_{3,1} \ge {p_{\Delta}}^{n-k} + \binom{n-k}{1}{p_{\Delta}}^{n-k-1}\left(\sum_{i=1}^{\Delta} X_{i,1} + p_{\Delta}p_{1b}\right) \quad (3)$$

and for any $2 \le t \le n-k$,

$$p_1 + \sum_{s=1}^{t}\sum_{i=1}^{\Delta} p_{2,i,s} + \sum_{s=1}^{t} p_{3,s} \ge {p_{\Delta}}^{n-k} + \sum_{s=1}^{t}\binom{n-k}{s}{p_{\Delta}}^{n-k-s}\left(\sum_{i=1}^{\Delta} X_{i,s} + p_{\Delta}{p_{1b}}^{s}\right) \quad (4)$$

where $X_{i,s}$ is given by the following.

$$X_{i,s} = (1-p)^{i-1}p[(1-p)^i\left(1-(1-p)^{\Delta-i}\right) + ip(1-p)^{i-1}]^s \quad (5)$$

Lemma 11. ${p_{3,n-k}} \le {p_{\Delta}}{p_{1b}}^{n-k}$, and for any $1 \le t \le n-k$,

$$\sum_{s=t}^{n-k} p_{3,s} \leq \sum_{s=t}^{n-k} {p_\Delta}^{n-k-s} {p_{1b}}^{s} \binom{n-k}{s} \quad (6)$$

$X_{i,s}$ in Equation 5 is a generalization of X_i in Lemma 5. Specifically, $X_{i,1} = X_i$. The proof of Lemma 8 and 9 is similar to Lemma 3 and 4. We further prove Lemma 10 and 11 in the Appendix. With these lemmas, we could compute the bound on T_k as given in the following.

$$T_k \leq \sum_{s=1}^{n-k} A_s T_{n-s} + \left(1 - {p_\Delta}^{n-k} - \sum_{s=1}^{n-k} A_s\right) T_1 + B + C \quad (7)$$

where A_s, B, C are coefficients that can be calculated given parameters n, p, Δ.

$$A_s = \binom{n-k}{s} {p_\Delta}^{n-k-s} \left[\sum_{i=1}^{\Delta} X_{i,s} + \frac{n-s}{n} p_\Delta p_{1b}\right]$$

$$B = (1-p_\Delta) \sum_{i=1}^{\Delta} (1-p)^{i-1} pi + [1-(1-p_\Delta)^2]\Delta$$

$$C = \frac{1}{np} \sum_{s=1}^{n-k} {p_\Delta}^{n-k-s} \binom{n-k}{s} {p_{1b}}^{s}$$

Combining Lemma 7 and 12, we have a set of $n-1$ equations consisting of $n-1$ variables, $T_1, T_2, \ldots, T_{n-1}$. Solving these equations gives us an upper bound of T_1.

To demonstrate our analysis is accurate, we build a blockchain simulator. We detail the mechanisms of the simulator in Section V. There are in total 10^4 honest miners. We configure the block mining difficulty such that $0.01 \leq np\Delta \leq 3$. I.e., up to 3 blocks are mined on average per Δ. In the current Bitcoin configuration, on average a block is mined every 10 minutes. If we take the conservative estimation of 10 seconds as maximum network delay, the block generation rate in Bitcoin is 0.017 blocks per Δ. Figure 3 shows both our analysis result and the simulation results of $T_{\tau,sys}$. The estimation closely tracks the simulation result. Our analysis only over-estimate $T_{\tau,sys}$ by up to 0.2% in the exact Δ-delay network.

B. $T_{\tau,sys}$ in the practical Δ-delay network

Empirical study observes that new blocks propagate in the network much faster [14, 15]. Majority of miners receive new blocks earlier than the maximum network delay. Figure 4 shows the block propagation pattern observed in Bitcoin network [15]. Given that anyone could join the Bitcoin network, it is impossible to know exactly when each node receives a new block. Instead, the authors in [15] leveraged Bitcoin's advertisement mechanism to infer when a node receives a block. In Bitcoin, after receiving a new block, the node sends an *INV* message to its neighbors, advertising that it receives the new block [16]. The authors deployed multiple *probing nodes* that connect to all publicly reachable Bitcoin nodes. The probing nodes do not relay blocks. They only record the timestamps when their neighbors send the INV messages. For each block, the first timestamp is used to approximate when the block is mined. The difference between the timestamp of a subsequent INV message and the first one is an estimate of the block delay to that node. Figure 4 shows the delay distribution of all miners. 86.2% of Bitcoin nodes receive the new block within 2 seconds. Note that miners are only a subset of all the Bitcoin nodes measured. Miners may also be connected through dedicated and reliable network for fast block propagation [14]. Therefore, the propagation should be faster within miners. Even if we take the estimation of $\Delta = 2$ seconds, Figure 4 shows that majority of miners still receive the new block much sooner, at around 0.5 seconds.

We model such a practical network using a propagation function $F(t)$. $F(t)$ denotes the number of miners having received the new block after t rounds since the block is mined. We have $F(0) = 1$, $F(t) < n$ for $0 \leq t < \Delta$ and $F(\Delta) = n$. The exact Δ-delay network is a special case and can be described as $F(t) = 1$ for $t < \Delta$ and $F(\Delta) = n$. By using $F(t)$, we make a simplifying assumption that the location of the miners does not impact block propagation. A new block propagates according to the same $F(t)$ wherever it is mined. Thus, we do not try to provide an upper bound on $T_{\tau,sys}$. Instead, we give a tight estimation of $T_{\tau,sys}$ under the practical Δ-delay network.

To simulate a network with the observed propagation pattern, we apply several heuristics to generate the topology in a trial-and-error manner. For instance, we modify the number of connections per node. The number of neighbors for each node follows a heavy-tail distribution. We also modify the per-hop latency for each link. Figure 5 shows the resulting simulated network. The CDF of block propagation delay closely resembles the observed pattern. The K-S measure between the two is 0.024. K-S measure is equal to 0 for identical CDFs and 1 for totally different CDFs [17].

We first simulate the blockchain system under the practical Δ-delay network and make two observations. First, even if fork blocks exist at a level, majority miners still receive the pioneer block first. Figure 6 shows the percentage of miners adopting the pioneer block when fork blocks exist. Even at higher block rate, approximately 76% of miners adopt the pioneer block earlier than fork blocks. Second, when multiple fork blocks exist, only a small number of miners adopt late fork blocks. A block is a *late fork block* if at least two blocks at the same level exist when it is mined. This is because by the time a late fork block is mined, most miners have either received the pioneer block or the first block.

We adopt a similar analysis as in the exact Δ-delay network. We relax the definitions of ally miners and competing miners in the context of the practical Δ-delay network. In round t_l, a miner is an *ally miner* if it is in the same class as the pioneer block at level l. Otherwise, it is a competing miner. The system is in a state $s \in {S'}_{u,i}$ in round t_l if there are in total i ally miners. Let $E[T_{s \to FIN}]$ denote the expected latency from the time when the system is in the state s to the time when some base block reaches consensus. Further define ${T'}_i$ as the maximum state expected latency among all states in ${S'}_{u,i}$. I.e., ${T'}_i = \max_{s \in {S'}_{u,i}} E[T_{s \to FIN}]$. We have $T_{\tau,sys} \leq {T'}_1$.

Analysis outline: Denote a pioneer block b at level $l \geq l_0$ is mined in round t_l. Further denote the pioneer block is in class x and there are k ally miners. We analyze ${T'}_k$ conditioning on what happens in $(t_l, t_l + \Delta]$. There are two cases.

- In case CS_1, none of the $n-k$ competing miners mines a fork block at level l. Let g_k denote the probability of CS_1.

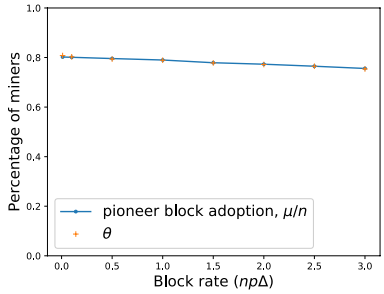

Figure 6. majority of miners adopt the pioneer block.

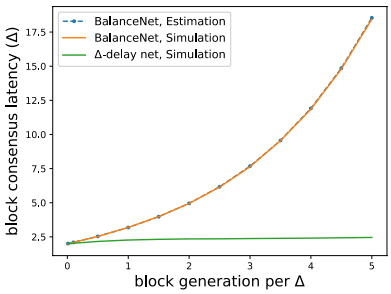

Figure 7. BalanceNet

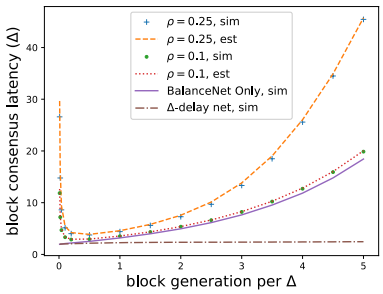

Figure 8. Balance Attack

- In case CS_2, at least one of the competing miners mine a fork block at level l.

In CS_1, the base block in class x will reach consensus by $t_l + \Delta$. In CS_2, no base block will reach consensus before round t_{l+1}. Denote eventually μ_k miners adopt the pioneer block b at level l and the rest adopt fork blocks. Let ϑ_k denote the probability that the pioneer block at level $l+1$ is also in class x. $\vartheta_k$is approximately μ_k/n. If $t_{l+1} > t_l + \Delta$, every miner has equal chance to mine the pioneer block at level $l+1$. ϑ_k is thus μ_k/n. If $t_{l+1} \leq t_l + \Delta$, depending on when each miner starts to mine on level $l+1$, the probability of mining the pioneer block is slightly different across different miners. However, Figure 6 demonstrates that overall μ_k/n is a good estimation of ϑ_k. As block rate varies from 0.01 to 3 blocks per Δ, μ_k/n is off from ϑ_k by at most 0.2%. We could then approximate T'_k using the following:

$$T'_k = g_k\Delta + (1-g_k)\{E[t_{l+1} - t_l | CS_2] + \frac{\mu_k}{n}T'_{\mu_k} + \frac{n-\mu_k}{n}T'_{n-\mu_k}\} \quad (8)$$

Equation 8 applies to any $1 \leq k < n$. Assuming g_k, μ_k and $E[t_{l+1} - t_l | CS_2]$ are known, we obtain a set of $n-1$ equations with $n-1$ unknown variables $T'_1, \ldots T'_{n-1}$. Solving these equations gives us an estimation of T'_1. In the next subsections, we estimate g_k, μ_k and $E[t_{l+1} - t_l | CS_2]$ respectively.

Estimate g_k: By round $t_l + i$, $F(i)$ miners have received the pioneer block at level l. Among the rest $n - F(i)$ miners, k are ally miners. On average, number of hashes computed towards a fork block in round i is $(n - F(i))\,(n-k)/n$. The probability that none of these hashes succeeds is $(1-p)^{(n-F(i))(n-k)/n}$. Iterating through all rounds in $(t_l, t_l + \Delta]$, we have

$$g_k \approx (1-p)^{\sum_{i=1}^{\Delta}(n-F(i))(n-k)/n} \quad (9)$$

Estimate μ_k: Number of miners adopting the pioneer block b depends on when the first fork block is mined. We condition our analysis on when the first fork block at level l is mined. Let $\mu_{k,j}$ denote the number of miners adopting the pioneer block when the first fork block is mined in round $t_l + j$. By round $t_l + j - 1$, $F(j-1)$ miners have already received the pioneer block. In any round $t_l + i$ where $i \geq j$, the number of miners that newly receive b is $f[i] = F(i) - F(i-1)$. Among these miners, some have already received the fork block and will thus not switch to accept the pioneer block. By round $t_l + i$, the fork block has already propagated to $F(i-j)$ miners. We could estimate the number of miners that newly receive and adopt the pioneer block in round $t_l + i$ as $f[i] \cdot (n - F[i-j])/n$. Therefore, we have

$$\mu_{k,j} = F[j-1] + \sum_{i=j}^{\Delta} f[i] \cdot (n - F[i-j])/n \quad (10)$$

Equation 10 only considers the impact of the first fork block. According to our observation in the simulation, late fork blocks are only adopted by a small number of miners. We find our approximation reasonable. Let $p_{k,j}$ denote the probability of the first fork block being mined in round $t_l + j$, we have

$$p_{k,j} = (1-p)^{\sum_{i=1}^{j-1}(n-F(i))(n-k)/n} \cdot \left[1 - (1-p)^{(n-F(j))(n-k)/n}\right] \quad (11)$$

Finally, μ_k could be calculated as

$$\mu_k = \frac{1}{1-g_k}\sum_{j=1}^{\Delta} p_{k,j}\mu_{k,j} \quad (12)$$

We estimate $E[t_{l+1} - t_l | CS_2]$ in a similar fashion. We condition when on fork blocks being mined. Finally, we substitute g_k, μ_k and $E[t_{l+1} - t_l | CS_2]$ into Equation 8. Solving these equations gives us an estimation of T'_1. Figure 3 shows the estimation and simulation results in red curves. Our estimation of $T_{\tau,sys}$ closely tracks the simulation results, with error up to 6% in the practical Δ-delay network. The result shows that as block rate increase, $T_{\tau,sys}$ even slightly decreases. The increase of block rate leads to the decrease in $E[t_{l+1} - t_l | CS_2]$. Meanwhile, higher block rate results in more late fork blocks, which have a limited impact on the consensus. Overall, the consensus latency decreases.

IV. Analysis of $T_{\tau,sys}$ with Adversary

We first discuss the impact of several existing attacks on the consensus latency, $T_{\tau,sys}$. We show that selfish mining attack and double-spend attack (also referred to as the private-chain attack in [12] do not increase $T_{\tau,sys}$. We then present a new attack aiming to extend $T_{\tau,sys}$.

Both the selfish mining attack and the double-spend attack assume the exact Δ-delay network [12, 13]. In the selfish mining attack, the adversary selectively releases its blocks to gain a revenue larger than its ratio of mining power. The adversary maintains a private chain that is longer than or equal to the longest chain of all honest miners. Honest miners mine on a shorter, public chain. The adversary only releases a malicious

block if the honest chains catch up with that block. Honest miners thus frequently work on blocks that are destined to be discarded in the future. As shown in Figure 4, the adversary mines a block b_{mal} in round r_l at level l. No honest block at level l has been mined by round r_1. The pioneer block b_l at level l is mined in round r_2. The adversary postpones broadcasting b_{mal} until round $r_2 + \Delta$, when every honest miner is going to receive b_l. Honest miners thus waste their mining power during period $[r_1, r_2 + \Delta]$. Given the adversary ability Adv_4, the adversary could also make the miner of the pioneer block switch to b_{mal}. b_{mal} reaches consensus. Since the intention of the selfish mining attack is not to extend block consensus latency, $T_{\tau,sys}$ is the same as analyzed in Section III.

In the double-spend attack, the adversary aims to make an honest miner reverse a confirmed block. To do this, the adversary minimizes the growth rate of the honest chains by enforcing the exact Δ-delay network. After a block is confirmed, the adversary keeps mining its private chain. Once the malicious chain is longer than the longest chain of an honest miner, the adversary broadcasts the malicious chain to that honest miner. The previous confirmed block is then discarded. The release of the malicious blocks does not impact the consensus latency. $T_{\tau,sys}$ still remains the same as analyzed in Section III.

We present a new attack, the *balance attack* that aims to extend $T_{\tau,sys}$. The motivation of the balance attack is that no block could get confirmed during the attack. The blockchain system effectively incurs a denial of service attack. The balance attack extends the consensus latency combining two methods. We describe in Section IV.A how the adversary enforces a new malicious network, *BalanceNet*, to delay a block from reaching consensus. Built on the BalanceNet, we describe how the adversary conducts its own malicious mining to further delay block consensus in Section IV.B.

A. BalanceNet

BalanceNet is inspired by the following observation in the exact Δ-delay network. Denote a pioneer block b at level l is mined by m_α. Suppose only one fork block b' at level l is mined by miner m_β in the next Δ rounds. In the exact Δ-delay network, by round $t_l + \Delta$, all miners except m_β will adopt b at level l. Only m_β adopts b'. The probability for either b or b' to reach consensus at round $t_{l+1} + \Delta$ is

$$pe_1 = \frac{1}{n}(1-p)^{(n-1)\Delta} + \frac{n-1}{n}(1-p)^{\Delta} \quad (13)$$

In general, if by round $t_l + \Delta$, k miners adopt b and $n-k$ miners adopt b', the probability for either b or b' to reach consensus by round $t_{l+1} + \Delta$ is

$$pe_k = \frac{k}{n}(1-p)^{(n-k)\Delta} + \frac{n-k}{n}(1-p)^{k\Delta} \quad (14)$$

pe_k is maximal when $k = n/2$. The name of the BalanceNet thus indicates this even distribution of all blocks at the same level. Formally, in the BalanceNet, the adversary delivers blocks in the following manner.

- For any l, the pioneer block at level l is delivered to all honest miners in round $t_l + \Delta$.
- If multiple blocks at level l have been mined by round $t_l + \Delta$, by adjusting the delivery order within these blocks, the adversary makes sure each block at level l is adopted by the same number of honest miners.

$T_{\tau,sys}$ in the BalanceNet: As in Section III, we analyze $T_{\tau,sys}$ at any given level l_0. We use the same definitions of base blocks and classes. We focus on the case when $p\Delta \ll 1$. With this assumption, we could simplify the analysis by assuming that the pioneer block at level $l+1$ is not mined before round $t_l + \Delta$. No block at level l is delivered to any miner before round $t_l + \Delta$. Only a few miners who mine a block at level l themselves could start mining on level $l+1$ before round $t_l + \Delta$. Since $p\Delta \ll 1$, the probability that these miners indeed mine a block at level $l+1$ before round $t_l + \Delta$ is close to 0. Therefore, every miner has the same length of chain in round $t_l + \Delta$.

In the BalanceNet, blocks could only reach consensus in round $t_l + \Delta$ for some level l. The system is in the *k-branch state* if honest miners belong to k different classes in round $t_l + \Delta$. Let E_k denote the expected latency from $t_l + \Delta$ to the time when any base block reaches consensus. If $k = 1$, the block at level l_0 has already reached consensus. We have $E_1 = 0$. Initially, the system is in the k-branch state in round $t_{l_0} + \Delta$ if $k-1$ out of the $n-1$ competing miners mine at least a block during $(t_{l_0}, t_{l_0} + \Delta]$. Denote this probability as p_k. We have

$$p_k = \binom{n-1}{k-1}(1-p_\Delta)^{k-1}{p_\Delta}^{n-k} \quad (15)$$

$T_{\tau,sys}$ can be expressed as

$$T_{\tau,sys} = \Delta + \sum_{k=1}^{n} p_k E_k \quad (16)$$

We calculate E_k. By definition of E_2, miners belong to two classes x and x' in round $t_l + \Delta$ for some $l \geq l_0$. Half of the miners belong to class x, while the others belong to class x'. Without loss of generality, assume the pioneer block at level $l+1$ is mined by a miner in class x. The probability that none of the miners in class x' mines a block during $(t_{l+1}, t_{l+1} + \Delta]$ is $\varepsilon_2 = (1-p)^{n\Delta/2}$. Thus, by round $t_{l+1} + \Delta$, with probability ε_2, base block bb_x reaches consensus. With probability $1-\varepsilon_2$, the system is still in the 2-branch state. We could recursively express E_2 as:

$$E_2 = E[t_{l+1} - t_l] + (1-\varepsilon_2)E_2 \quad (17)$$

In the BalanceNet, $E[t_{l+1} - t_l] = \Delta + 1/np$. Solving Equation 17 yields E_2. We generalize the above approach to iteratively compute E_k from E_2. This gives us $T_{\tau,sys}$ in the BalanceNet. To demonstrate that our analysis is accurate, we simulate the BalanceNet under various settings. There are 10^4 honest miners. $\Delta = 10^{12}$. We vary the block mining difficulty such that $0.01 \leq np\Delta \leq 3$. Figure 7 shows our analysis closely tracks the simulation results. Even with no malicious mining, the adversary could greatly increase $T_{\tau,sys}$ in the BalanceNet compared to the exact Δ-delay network.

B. Full balance attack

Adversary strategy: The adversary could combine its own malicious mining with the BalanceNet to further increase $T_{\tau,sys}$.

The attack starts when every honest miner's longest chain contains a block at level $l_0 - 1$, and no miner has a block at level l_0. In the case $l_0 = 1$, the attack starts in round 0. We say a *consensus event at level* l occurs if a pioneer block in class x is mined in round t_l and no miner from a different class other than x mines any block during $(t_l, t_l + \Delta]$. If there is no malicious mining, the base block bb_x reaches consensus when a consensus event at level l occurs in round $t_l + \Delta$. However, with malicious mining, the adversary could release a malicious block at level l if it has already mined such a block. The release of the malicious block *breaks the consensus event*. The adversary delivers its malicious block at level l in a similar fashion as it does when there are multiple honest blocks. Half of the honest miners adopt the malicious block at level l, and the rest adopt the honest block at level l. The system is again in the 2-branch state in the end of round $t_l + \Delta$.

After the adversary releases its malicious block at level l, it keeps mining its malicious chain. All malicious blocks starting from level $l + 1$, if any, are still kept in private. The adversary repeats the previous action every time a consensus event occurs. Every time the adversary releases its blocks, the system is back in the 2-branch state.

$T_{\tau,sys}$ in the Balance Attack: There are two cases when the first consensus event at level $l \geq l_0$ occurs in round $t_l + \Delta$.

1) In case CS_1, the adversary has not mined a block at level l by round $t_l + \Delta$. A base block reaches consensus by $t_l + \Delta$. $T_{\tau,sys} \leq t_l + \Delta - t_{l_0}$. Denote the probability of CS_1 as p_1.
2) In case CS_2, the adversary has mined a block at level l by round $t_l + \Delta$. It releases its malicious blocks so that the system is in the 2-branch state in the end of round $t_l + \Delta$. The adversary randomly chooses from one of the two classes x and x' to keep mining. Denote the adversary chooses class x. The next consensus event occurs in round $t_{l'} + \Delta$. There are further two subcases.
    a) In case $CS_{2,a}$, by round $t_{l'} + \Delta$, the only honest block at level l' is in class x' and the adversary has mined a block at level l'. The adversary could again release a malicious block at level l'. The system is back in the 2-branch state in the end of round $t_{l'} + \Delta$.
    b) In case $CS_{2,b}$, the only honest block at level l' is in class x, or the adversary has not mined a malicious block at level l'. The base block bb_x then reaches consensus by round $t_{l'} + \Delta$.

There is one difference between the initial consensus event and all the subsequent consensus events. When the initial consensus event at level l occurs, the adversary could break the consensus event as long as it has mined a block at level l. When a subsequent consensus event occur at level l', even if the adversary has mined a block at level l', with probability 1/2, the adversary is in the same class as the only honest block at level l'. With probability at least 1/2, a base block will reach consensus after a non-initial consensus event occurs.

We condition our analysis on when the initial consensus event occurs. We first estimate the probability that the initial consensus event occurs at level $l_0 + i - 1$ $(i \geq 1)$. We then calculate the probability that the adversary has already mined a malicious block at level $l_0 + i - 1$ by round $t_{l_0+i-1} + \Delta$. Initially any miner who mines a block during $(t_{l_0}, t_{l_0} + \Delta]$ will be in a different class. Let $p_{n,i}$ denote the probability that the first consensus event occurs in round $t_{l_0+i-1} + \Delta$ $(i \geq 1)$. We have $p_{n,1} = {p_\Delta}^{n-1}$. More generally, if the system is in the k-branch state in round $t_l + \Delta$ for some level l, let $p_{k,i}$ $(2 \leq k \leq n, i \geq 1)$ denote the probability that the next consensus event occurs in round $t_{l+i-1} + \Delta$. Let $\varepsilon_k = (1-p)^{n\Delta/k}$ denote the probability that none of a group of n/k miners mines any block in Δ rounds. We have $p_{k,1} = {\varepsilon_k}^{k-1}$. We could recursively calculate $p_{k,i}$ for $i \geq 2$ and $2 \leq k \leq n$, as follows.

$$p_{k,i} = \sum_{s=2}^{k} q_{k,s} p_{s,i-1} \tag{18}$$

$q_{k,s}$ is the probability that $s-1$ out of $k-1$ groups mine at least a block during $(t_{l+1}, t_{l+1} + \Delta]$. We then calculate the probability that the adversary mines at least i blocks by round $t_{l_0+i-1} + \Delta$. Let $\alpha_{i,k}$ denote the probability that adversary mines k blocks during $(t_{l_0-1} + \Delta, t_{l_0+i-1} + \Delta]$. By round $t_{l_0+i-1} + \Delta$, with probability $\sum_{k=1}^{i-1} \alpha_{i,k}$, a base block will reach consensus. If the adversary mines $i + j$ $(j \geq 0)$ blocks, let Γ_j denote the expected time from round $t_{l_0+i-1} + \Delta$ to the time when a base block reaches consensus. We could express $T_{\tau,sys}$ as follows.

$$T_{\tau,sys} = (\Delta + \frac{1}{np}) \cdot [\sum_{i=1}^{\infty} p_{n,i}(i + \sum_{j=i}^{\infty} \alpha_{i,j}\Gamma_j)] - \frac{1}{np} \tag{19}$$

We now calculate $\alpha_{i,j}$ and Γ_j. The duration of the period $(t_{l_0-1} + \Delta, t_{l_0+i-1} + \Delta]$ can be expressed as $D = i\Delta + \sum_{s=1}^{i} W_s$, where W_s is a geometric random variable with average $1/np$ .To efficiently calculate $\alpha_{i,j}$, we leverage the approximation of large numbers that when $p \ll 1$, the geometric distribution with average $1/np$ can be approximated as the exponential distribution with the same average. Thus $\sum_{s=1}^{i} W_s$ follows the Erlang distribution with parameter i and $\lambda = np$. Let $\lambda_a = \rho np$ denote the rate of malicious mining. We have

$$\alpha_{i,j} = \int_{t=0}^{+\infty} \lambda^i \frac{t^{i-1}[\lambda_a(t + i\Delta)]^j}{i!\, j!} e^{-\lambda t - \lambda_a(t+i\Delta)} dt \tag{20}$$

Rearranging Equation 20, we have

$$\alpha_{i,j} = e^{-\lambda_a i\Delta} \frac{\lambda^i {\lambda_a}^j}{i!\, j!} \mathcal{X}_{i,j} \tag{21}$$

where $\mathcal{X}_{i,j} = \int_{t=0}^{+\infty} t^{i-1}(t + i\Delta)^j\, e^{-(\lambda+\lambda_a)t} dt$. We compute $\mathcal{X}_{i,j}$ by fixing $i = \delta$. Let $\mathcal{Y}_{z,j} = \int_{t=0}^{+\infty} t^{z-1}(t + \delta\Delta)^j\, e^{-(\lambda+\lambda_a)t} dt$. We have $\mathcal{X}_{i,j} = \mathcal{Y}_{i,j}$ and for $1 \leq z \leq i$,

$$\mathcal{Y}_{z,j} = \frac{z-1}{\lambda + \lambda_a} \mathcal{Y}_{z-1,j} + \frac{j}{\lambda + \lambda_a} \mathcal{Y}_{z,j-1}$$

and

$$\mathcal{Y}_{1,j} = \frac{\Delta^j}{\lambda + \lambda_a} + \frac{j}{\lambda + \lambda_a} \mathcal{Y}_{1,j-1}$$

The boundary case is $\mathcal{Y}_{1,0} = 1/(\lambda + \lambda_a)$. Once we obtain $\mathcal{Y}_{i,j}$ we have $\mathcal{X}_{i,j} = \mathcal{Y}_{i,j}$ and obtain $\alpha_{i,j}$. Finally, we analyze Γ_j. We start with Γ_0. After the adversary breaks a consensus event, the system is in the 2-branch state. With probability $p_{2,i}$, the next consensus event occurs after another i levels are mined. Let $\Gamma_s = 0$ when $s < 0$. We have

$$\Gamma_0 = \sum_{i=1}^{\infty} p_{2,i}\{i + \frac{1}{2}\sum_{j=0}^{\infty} \alpha_{i,j}\Gamma_{j-i}\} \quad (22)$$

In general,

$$\Gamma_k = \sum_{i=1}^{\infty} p_{2,i}\{i + \frac{1}{2}\sum_{j=0}^{\infty} \alpha_{i,j}\Gamma_{k+j-i}\} \quad (23)$$

To numerically calculate Γ_k, we make two observations. First, in a stable blockchain system, the honest chain growth rate is higher than the malicious chain growth rate. $\alpha_{i,j}$ decreases with j. We pick a sufficiently large η and assume $\alpha_{i,j} = 0$ when $j \geq i + \eta$. Second, since the adversary could break each subsequent consensus event with probability at most ½, we could show that Γ_k is bounded. Let $\mathcal{J}$ denote the expected time from the start of the attack to the time when the first consensus event occurs. Assume every time a subsequent consensus event occurs, with probability ½ a base block reaches consensus. This leads to an upper bound of Γ_k. We have $\Gamma_k \leq \mathcal{J} + 0.5\Gamma_k$ and $\Gamma_k \leq 2\mathcal{J}$. We thus approximate that $\Gamma_k = 2\mathcal{J}$ when $k \geq \eta$. Equation 22 and 23 give us a set of η equations containing η unknown variables, $\Gamma_0, \Gamma_1, \ldots, \Gamma_{\eta-1}$. Solving these equations yields Γ_k. In practice, we choose η sufficiently large such that the estimation error converges to 0. Finally, we obtain $T_{\tau,sys}$ by substituting Γ_k and $\alpha_{i,j}$ into Equation 19. Figure 8 compares $T_{\tau,sys}$ in four different scenarios, the BalanceNet only, the BalanceNet combined with different malicious mining power, and the exact Δ -delay network. $T_{\tau,sys}$ is further extended as the malicious mining power increases. When $np\Delta = 5.0$, the balance attack results in a 18.5 increase of $T_{\tau,sys}$, from 2.28Δ to 45Δ.

V. Blockchain Simulator

We build a discrete time blockchain simulator to validate the accuracy of our analysis. The simulator proceeds based on rounds. We simulate block mining as follows. The simulator processes every event in a FIFO order. There are two types of events. A block-mined event represents a miner mining a new block. A block-received event represents a miner receiving a new block. Processing an event can generate a set of new events. For example, when processing a block-mined event for miner m_h, the simulator generates $n - 1$ block-received events. Each of these events represents the newly mined block being received by an honest miner. The timing of these events depends on the network delay. The simulator also generates a new block-mined event. This event represents when this miner mines its next block. It occurs T rounds in the future. T follows a geometric distribution with average $1/p$.

VI. Conclusion

We propose that a block should not be confirmed before it reaches consensus. We study the consensus latency in this paper. We analyze the consensus latency in various scenarios, both with and without the adversary. We find that blocks could reach consensus quickly when there is no adversary, in around twice the maximum network delay. The more fork blocks are generated, the longer it takes a block to reach consensus. However, somewhat contrary to our intuition, the impact of fork blocks is limited. Even when there are 3 blocks mined by network delay Δ, the expected consensus latency is still as low as 2.3Δ.

We find that existing attacks such as the selfish mining attack and double spend attack do not impact the consensus latency. We further show that the adversary could greatly increase the consensus latency if it could totally control the network. We describe an adversarial network called BalanceNet. The consensus latency could increase to $\sim 20\Delta$ in the BalanceNet. We further describe a new attack called the Balance Attack that combines malicious mining with the BalanceNet. Consensus latency is extended to $\sim 45\Delta$ when the adversary possesses 25% of honest mining power. Our analysis of consensus latency helps us better understand the consensus behavior of PoW blockchains.

Appendix

A. *Proof of Lemma 2*

It is easy to see $T_k \geq \Delta$. A block could only reach consensus after it propagates to every honest miner. We now show that $T_{k+1} \leq T_k$ for any $1 \leq k \leq n-2$. The idea of the proof is that given any state $s \in S_{u,k+1}$, we construct a state $s' \in S_{u,k}$ and show $E[T_{s\to FIN}] \leq E[T_{s'\to FIN}]$.

Consider any state $s \in S_{u,k+1}$. There are $k+1$ ally miners in round t_l. Denote the ally miners are in class x. Further assume miner m_1's chain contains the pioneer block at level l. Pick any ally miner m_β other than m_1 and changes its class to a new class x'. This could be done through assigning a new "imaginary" blockchain to m_β. The new chain has the same length as m_β's old chain. Each block in the new chain at level l_0 or higher is only adopted by m_β and no other miner. Consequently, no other miner is in class x'. Let s' denote the resulting system state. There are k ally miners in s', i.e., all the ally miners in s except m_β are also ally miners in s'. Thus, we have $s' \in S_{u,k}$.

To show $E[T_{s\to FIN}] \leq E[T_{s'\to FIN}]$, we consider the random process of block mining of all miners, denoted as $\{X(t,\omega), t \in T, \omega \in \Omega\}$. This process consists of a series of random variables indexed by time t, each representing which of the n miners mines a block in round t. We say a *sample path* of this process $\{X(t,\omega_0), t \in T\}$ for a fixed outcome $\omega_0 \in \Omega$ is a function that maps the time t to an n-dimension vector $(\theta_1, \ldots, \theta_n)$ where $\theta_i = 1$ if m_i mines a block in round t, and $\theta_i = 0$ otherwise. $T_{s\to FIN}(\omega_0)$ denotes the consensus latency under sample path ω_0. We show that for any sample path ω_0, if $T_{s'\to FIN}(\omega_0)$ is finite, then $T_{s\to FIN}(\omega_0)$ is also finite and $T_{s\to FIN}(\omega_0) \leq T_{s'\to FIN}(\omega_0)$. Denote a base block b reaches consensus in round t'. I.e., $T_{s'\to FIN}(\omega_0) = t' - t_l$. If block b does not belong to class x or class x', then assigning m_β a new chain does not impact consensus. We have $T_{s\to FIN}(\omega_0) = T_{s'\to FIN}(\omega_0)$. If block b belongs to class x or class x', $T_{s\to FIN}(\omega_0) \leq T_{s'\to FIN}(\omega_0)$. Combining all sample paths, we have

$E[T_{s\to FIN}] \le E[T_{s'\to FIN}]$. Since this applies to any state $s \in S_{u,k+1}$, we have $T_{k+1} \le T_k$.

B. Proof of Lemma 3

Since m_n's chain contains blocks up to level $l-1$ in round t_l, m_n not mining any block during $(t_l, t_l+\Delta]$ implies that it will not mine any block at level l. Therefore $p_1 \ge (1-p)^\Delta$.

C. Proof of Lemma 4

The probability that m_1 mines at least one block during $(t_l, t_l+i]$ is $1-(1-p)^i$. m_1 mining a block during $(t_l, t_l+i]$ implies that the pioneer block at level $l+1$ must have been mined by round t_l+i. On the other hand, $1-p_1$ is the probability that m_n indeed mines a block at level l by round $t_l+\Delta$. Therefore, the RHS in Lemma 4 represents the probability of the events that belong to $\bigcup_{j=1}^{i} CS_{2,j}$.

D. Proof of Lemma 5

For each X_i, $(1-p)^{i-1}p$ is the probability that miner m_1 does not mine any block during $(t_l, t_l+i-1]$ and mines a block in round t_l+i. The second part $(1-p)^i(1-(1-p)^{\Delta-i}) + ip(1-p)^j$ is the probability that miner m_n mines at most one block during $(t_l, t_l+i]$ and at least one block during $(t_l, t_l+\Delta]$. $\sum_{i=1}^{\Delta} X_i$ represents events that belong to either $CS_{2,i,1}$ $(1 \le i \le \Delta)$ or CS_1;

The second term in the RHS of Lemma 5, $(1-p)^\Delta$, is the probability that m_n does not mine any block in these Δ rounds. It represents events that belong to CS_1; The third term, $\Delta p(1-p)^{2\Delta-1}$, is the probability that m_n mines one block and m_1 does not mine any block during $(t_l, t_l+\Delta]$. It represents events that belong to either CS_1, $CS_{2,i,1}$ or CS_3. On the other hand, the three terms in the RHS of Lemma 5 represent the probability of three disjoint sets of events. By the probability rule of the sum over disjoint sets, Lemma 5 is proven.

E. Proof of Lemma 6

Case CS_3 implies the following two events occur.

- E_1: miner m_1 does not mine any block during $(t_l, t_l+\Delta]$.
- E_2: miner m_n mines a block at level l, but does not mine any block at level $l+1$ during $(t_l, t_l+\Delta]$.

The probability of E_1 is $(1-p)^\Delta$. Denote m_n's chain contains blocks up to level $h < l$ in round t_l. E_2 is then equivalent to m_n mining $l-h$ blocks during $(t_l, t_l+\Delta]$, which occurs with probability $\binom{\Delta}{l-h} p^{l-h}(1-p)^{\Delta-l+h}$. It can be shown that when $p\Delta \ll 1$, this probability is maximized when $h = l-1$. Lemma 6 is then proven.

F. Proof of Lemma 7-11

Due to page limit, proofs of Lemma 7-11 could be found in the extended online version [18].

G. Proof of Lemma 7

Let $M = p_1 + \sum_{i=1}^{\Delta} p_{2,i} + p_3$. We rearrange the RHS of Equation 1 as follows.

$$
\begin{aligned}
&T_{n-1}\Big[\sum_{i=1}^{\Delta} p_{2,i} + \frac{n-1}{n}p_3\Big] + T_1\Big[\sum_{i=1}^{\Delta} q_{2,i} + \frac{1}{n}p_3\Big] \\
&= (1 - M + \frac{1}{n}p_3)T_1 + (1 - p_1 - (1 - M + \frac{1}{n}p_3))T_{n-1} \\
&= (1 - M + \frac{1}{n}p_3)(T_1 - T_{n-1}) + (1-p_1)T_{n-1} \qquad (29)
\end{aligned}
$$

By Lemma 2, $T_1 - T_{n-1} \ge 0$. By Lemma 6, $p_3 \le \Delta p(1-p)^{2\Delta-1}$. By Lemma 5, $M \ge \sum_{i=1}^{\Delta} X_i + (1-p)^\Delta + \Delta p(1-p)^{2\Delta-1}$. We have

$$
\begin{aligned}
&RHS\ of\ Eq\ 29 \\
&\le (1 - A - (1-p)^\Delta)(T_1 - T_{n-1}) + (1-p_1)T_{n-1} \qquad (30)
\end{aligned}
$$

where $A = \sum_{i=1}^{\Delta} X_i + \Delta p(1-p)^{2\Delta-1}\frac{n-1}{n}$. On the other hand, by Lemma 4, $\sum_{j=1}^{i}(p_{2,j} + q_{2,j}) \ge (1-p_1)(1-(1-p)^i)$. Thus,

$$
\begin{aligned}
&(p_1 + p_3)\Delta + \sum_{j=1}^{\Delta}\big(p_{2,j} + q_{2,j}\big)\, j \\
&= \Delta - \sum_{j=1}^{\Delta}\big(p_{2,j} + q_{2,j}\big)\,(\Delta - j) \\
&= \Delta - \sum_{i=1}^{\Delta-1}\sum_{j=1}^{i} p_{2,j} + q_{2,j} \\
&\le \Delta - (1-p_1)\left(\Delta - 1 - \sum_{i=1}^{\Delta-1}(1-p)^i\right) \\
&\le \Delta - (1-p_1)(\Delta - (1-(1-p)^\Delta)/p) \qquad (31)
\end{aligned}
$$

Substitute Equation 30 and 31 into the RHS of Equation 1, We have

$$
\begin{aligned}
&T_{n-1} \le (1 - A - (1-p)^\Delta)(T_1 - T_{n-1}) + \Delta \\
&+(1-p_1)(T_{n-1} - \Delta + (1-(1-p)^\Delta)/p + p_3/np) \qquad (32)
\end{aligned}
$$

By Lemma 2, $T_{n-1} \ge \Delta$. By Lemma 4, $p_1 \ge (1-p)^\Delta$. By Lemma 6, $p_3 \le \Delta p(1-p)^{2\Delta-1}$. Substituting these into Equation 32, Lemma 7 is proven.

H. Proof of Lemma 10

We start with the base case, Equation 3. ${p_\Delta}^{n-k}$ is the probability that none of the $n-k$ competing miners mines a block during $(t_l, t_l+\Delta]$. ${p_\Delta}^{n-k}$ represents the events that belong to CS_1. For each $X_{i,1}$, $(1-p)^{i-1}p$ is the probability that miner m_1 does not mine any block in the first $i-1$ rounds and mines a block in round t_l+i. The second term, $(1-p)^i(1-(1-p)^{\Delta-i}) + ip(1-p)^{i-1}$ is the probability that a competing miner mines at most one block by round t_l+i and at least one block by round $t_l+\Delta$. Therefore $\sum_{i=1}^{\Delta} X_{i,1}$ represents the events that belong to either $CS_{2,i,1}$ $(1 \le i \le \Delta)$ or CS_1; On the other hand, $p_\Delta p_{1b}$ is the probability that m_1 does not mine any block in these Δ rounds, and the only competing miner who mines a block at level l mines 1 block during $(t_l, t_l+\Delta]$. $p_\Delta p_{1b}$ represents the events that belong to either $CS_{3,1}$ or CS_1; Finally, there are $n-k$ different ways of choosing which competing miner mines a block at level l. By the probability rule of the sum over disjoint sets, Equation 3 is proven.

Equation 4 could be proved in a similar way. The only difference is that there are $n-k$ different ways of picking out s competing miners who mine a block at level l. $\sum_{i=1}^{\Delta} X_{i,1}$ thus describes the events that belong to either $CS_{2,j,t}$ $(1 \le j \le \Delta, 1 \le t \le s)$ or CS_1. The third term, $\Delta p(1-p)^{2\Delta-1}$, is the probability that m_1 does not mine any block and m_n mines one block during $(t_l, t_l + \Delta]$. Therefore, the three terms in the RHS of Equation 4 denote the probability of three disjoint sets of events. Furthermore, $\sum_{i=1}^{\Delta} X_{i,1}$ represents the events that belong to either $CS_{2,i,1}$ $(1 \le i \le \Delta)$, or CS_1; $(1-p)^{\Delta}$ represents the events that belong to CS_1; and $\Delta p(1-p)^{2\Delta-1}$ represents the events that belong to either CS_1, $CS_{2,i,1}$ $(1 \le i \le \Delta)$, or CS_3. By the probability rule of sum over disjoint sets, Lemma 10 is proven.

I. Proof of Lemma 11

Case $CS_{3,n-k}$ implies the following two events occur.

- E_1: miner m_1 does not mine any block during $(t_l, t_l + \Delta]$.
- E_2: all $n-k$ competing miners mine a block at level l, but none of them mines a block at level $l+1$ during $(t_l, t_l + \Delta]$.

The probability of E_1 is $(1-p)^{\Delta}$. For any single competing miner, denote its chain contains blocks up to level $h < l$ in round t_l. The probability that it mines $l-h$ blocks during $(t_l, t_l + \Delta]$ is $\binom{\Delta}{l-h} p^{l-h}(1-p)^{\Delta-l+h}$. Similar to the proof of Lemma 6, when $p\Delta \ll 1$, this probability is maximized when $h = l-1$. We have $prob[E_2] \le {p_{1b}}^{n-k}$. Therefore $p_{3,n-k} \le p_{\Delta}{p_{1b}}^{n-k}$. We could apply the same reasoning for Equation 6. Lemma 11 is then proven.

REFERENCES

[1] Nakamoto, S., 2019. Bitcoin: A peer-to-peer electronic cash system. Manubot.

[2] Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K. and Njilla, L., 2017, May. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 468-477). IEEE.

[3] Mettler, M., 2016, September. Blockchain technology in healthcare: The revolution starts here. In 2016 IEEE 18th international conference on e-health networking, applications and services (Healthcom) (pp. 1-3). IEEE.

[4] Su, S., Wang, K. and Kim, H.S., 2018, July. SmartSupply: Smart contract based validation for supply chain blockchain. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 988-993). IEEE.

[5] Wang, K., Zhang, Z. and Kim, H.S., 2018, July. ReviewChain: Smart contract based review system with multi-blockchain gateway. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 1521-1526). IEEE.

[6] Reyna, A., Martín, C., Chen, J., Soler, E. and Díaz, M., 2018. On blockchain and its integration with IoT. Challenges and opportunities. Future generation computer systems, 88, pp.173-190.

[7] Wood, G., 2014. Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014), pp.1-32.

[8] How to confirm blocks in PoW blockchains. Under review.

[9] Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. 2020. Everything is a Race and Nakamoto Always Wins. arXiv preprint arXiv:2005.10484 (2020).

[10] Garay, J., Kiayias, A. and Leonardos, N., 2015, April. The bitcoin backbone protocol: Analysis and applications. In Annual international conference on the theory and applications of cryptographic techniques (pp. 281-310). Springer, Berlin, Heidelberg.

[11] Pass, R., Seeman, L. and Shelat, A., 2017, April. Analysis of the blockchain protocol in asynchronous networks. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 643-673). Springer, Cham.

[12] Kiffer, L., Rajaraman, R. and Shelat, A., 2018, October. A better method to analyze blockchain consistency. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 729-744).

[13] Eyal, I. and Sirer, E.G., 2014, March. Majority is not enough: Bitcoin mining is vulnerable. In International conference on financial cryptography and data security (pp. 436-454). Springer, Berlin, Heidelberg.

[14] Gencer, A.E., Basu, S., Eyal, I., Van Renesse, R. and Sirer, E.G., 2018, February. Decentralization in bitcoin and ethereum networks. In International Conference on Financial Cryptography and Data Security (pp. 439-457). Springer, Berlin, Heidelberg.

[15] https://dsn.kastel.kit.edu/bitcoin/

[16] https://en.bitcoin.it/wiki/Protocol_documentation#inv

[17] https://en.wikipedia.org/wiki/Kolmogorov–Smirnov_test.