Consensus and Security in Canonchain

Lei Zhang*

March 15, 2019 Version 1.0

Abstract

In this paper, we build up the mathematical foundations of Canonchain, which utilizes directed cyclic graph (DAG) for storing transactions. We provide thorough and rigorous analysis on our consensus mechanism which depends on non-anonymous reputable entities, called witnesses. A novel scheme for replacing witnesses if needed is proposed. And the security of Canonchain network against malicious behavior is guaranteed.

1 Introduction

The concept of blockchain as an independent technology began to surge in 2015. Prior to this, it was known as the data structure of Bitcoin technology. In Nakamoto's white paper [1], the two words "block" and "chain" appear together, but it only refers to "a series of blocks." With the popularity of Bitcoin, we have classified technology and concepts in Bitcoin as blockchain 1.0. With Ethereum [2] running as a platform for distributed applications, people began to classify Ethereum as Blockchain 2.0. Now the market is vying for the fundamental structure for blockchain 3.0, and many people think that DAG technology is the best choice.

In traditional blockchain technology represented by Bitcoin and Ethereum, blocks and transactions are two separate concepts. The transaction is confirmed by the miners packing into the block, and the throughput of the transaction is limited by the block size and the block generation speed. Because there are too many variables involved in the blockchain design, a community can easily split up with different understandings of the blockchain. What's worse is that no one seems to agree with which values are correct and

*Author's contact information: leizha@ntlabs.io

thus cause the chain to fork. In the blockchain, the miners have the right to decide the content of the block. The profit-seeking behavior of the miners can easily lead to excessive concentration of power in the blockchain or voting rights, thus losing the discentralization characteristics. DAG-based digital currency was created to solve these problems. Compared to traditional blockchain technology, DAG technology has the following advantages: 1) High transaction speed and large throughput; 2) (Almost) no transaction fee and friendly to small payments; 3) No special miners required to participate; 4) Strong scalability.

The idea of using DAGs in the cryptocurrency space has been around for a while. DAGLabs has proposed a series of consensus protocols, such as GHOST [3], SPECTRE [4] and PHANTOM [5]. The general idea behind them is to utilize a DAG of blocks. Also the miners in the system still compete for transaction fees, and new tokens may be created by these miners. Instead, some cryptocurrencies depend on a DAG of individual transactions other than blocks. IOTA [6] and Byteball [7] are currently the most popular projects on the market. They all have the same advantages as DAG, but the design has its own merits, with different performance, complexity, reliability and security.

IOTA and Byteball have quite different design details in order to cater to different audiences. IOTA assigns a certain weight to each transaction, and the transaction is accessed through the power of work (PoW) mechanism, which can prevent forking by path of the maximum weight. Instead of utilizing PoW mechanism, Byteball prevents junk transactions by charging a small fee, and introduces votes from witnesses to determine correct transactions. Both IOTA and Byteball technologies have their own unique advantages. Byteball's unique features are undoubtedly its integrated private assets, offering smart contracts similar to Ethereum, and even further expanding the ability of these contracts to bet on sporting events or political elections. The uniqueness of IOTA undoubtedly includes no transaction fee, and it is the only technology that can be used as the backbone of the Internet of Things on a global scale.

Similar to Byteball, the consensus protocol in Canonchain relies on witnesses, which are non-anonymous reputable entities. However, the design and analysis in Byteball is quite heuristic. Our contributions in Canonchain are two folds. First, we provide thorough and rigorous analysis towards consensus and security in Canonchain network. Second, we propose a novel merit based witness replacement protocol, where a witness is kicked off the network by "virtual" consensus among users other than voting.

2 Definitions and Assumptions

Unlike Bitcoin and Ethereum, transactions in Canonchain are stored and organized in a DAG. In this section, we will describe key definitions and assumptions which will be used throughout this paper.

2.1 Witnesses

In Canonchain, we rely on some non-anonymous reputable people or companies who might have a long established reputation, or great benefits in keeping the network healthy. We call them witnesses. Each user has its own witness list, which is initially a copy of a witness pool $\mathcal{W}$ with N witnesses. Witnesses are expected to post transactions frequently and behave honestly. However, it is unreasonable to totally trust any single witness. In the case that a witness has lost his credibility or there are better witnesses, we propose a scheme in Canonchain which can gradually change witnesses by one witness a time without jeopardizing network consensus and security. Details on how to change witnesses will be elaborated in Section 4.

2.2 DAG

In Canonchain, each block represents one transaction, which contains references to previous blocks (called parents) through their hashes. Blocks and their parent-child links are the vertices and edges of the DAG, respectively. At any time, each node in the network would observe slightly different graph due to network delays. Let $\mathcal{G}_i(t)$ denote the graph node i has observed at time t. For any block B node i receives at time t, a validation procedure is performed before B can be added onto $\mathcal{G}_i(t)$. Two of those validation checks are of great importance to our consensus protocol. One is to make sure all B's parents already exist in the graph $\mathcal{G}_i(t)$. The other is to make sure that B's witness list, defined as the witness list of B's author when composing B, is identical to node i's witness list at time t. We will revisit this condition when we discuss our witness replacement protocol in Section 4.

In the following, we describe some key terms which will be used intensively in the subsequent sections. We drop the subscript i and use $\mathcal{G}(t)$ to represent a general graph at time t.

- Level: The level of a block refers to the length of the longest path in $\mathcal{G}(t)$ from this block to the genesis. Let $l(B)$ denote the level of block B.

- Witnessed Level: Suppose each block has a best parent determined by a certain rule. For any block B, start from B and travel back in time through best parent links, count witness-authored blocks (if the same witness is encountered more than once, he is not counted again), stop as soon as the K witnesses from B's witness list have been encountered. The witnessed level of B is defined as the maximum of the level of the block we stopped and the witnessed level of B's best parent. It assures that the witnessed level of a block is no less than the witnessed level of its best parent. We denote B's witnessed level by $wl(B)$.

- Block comparison: For any pair of blocks B_1 and B_2, we call B_2 is better than B_1 i.e., $B_1 \prec B_2$ if and only if B_2 has larger witnessed level, or larger level if B_1 and B_2 have the same witnessed level, or larger hash in the case that B_1 and B_2 have the same witnessed level and the same level. We denote this block ordering rule as $\mathcal{R}$.

- Best Parent: The best parent of a block is one of its parents, which is the best under rule $\mathcal{R}$. A block's best parent is always the best tip block of the graph under rule $\mathcal{R}$ when this block is composed. Here, tip blocks refer to blocks without any child.

- Main chain: The main chain of graph $\mathcal{G}(t)$ is defined as the path starting from the best tip block in $\mathcal{G}(t)$ under rule $\mathcal{R}$ to the genesis block through best parent links.

- Main chain index (MCI): We first define MCI for blocks that lie directly on the main chain. The genesis block has index 0, the next block on the main chain that is a child of genesis has index 1, and so on. By traveling forward along the main chain we assign indices to blocks that lie on the main chain. For any block that does not lie on the main chain, its MCI is assigned by the MCI of the block on the main chain that first includes it directly or indirectly.

- Stable block: Main chain for graph $\mathcal{G}(t)$ changes over time. However, they will converge to some block when tracing back in time. A block of $\mathcal{G}(t)$ is called a stable block if it is guaranteed to be contained in all future main chains of graphs $\mathcal{G}(\tau)$ for all $\tau \geq t$.

- Last stable block: The last stable block $L(t)$ of graph $\mathcal{G}(t)$ is defined as the stable block on the main chain with the largest MCI. And its MCI is called last stable MCI.

- Stable main chain: Stable main chain of graph $\mathcal{G}(t)$ is defined to be the path starting from the last stable block $L(t)$ to the genesis block through best parent links.

- Finalized MCI: When a block B is first included by some block on the stable main chain of graph $\mathcal{G}(t)$, its MCI will not be changed in the future. We call it the finalized MCI of block B.

Remark. The definitions of best parent and witnessed level above depend on each other. However, they can be built up recursively. The first block of each user has only one parent, the genesis block. Therefore, its best parent, level and witnessed level are the genesis, 1 and 0, respectively. For any other block, its best parent can be determined by the comparison of its parents under rule $\mathcal{R}$. And then its witnessed level can be computed through the best parent links.

Remark. Since the definitions of level, witnessed level and best parent of a block in a graph only depends on parent-child links, the following property holds: for two graphs $\mathcal{G}_0 \subseteq \mathcal{G}_1$, any block $B \in \mathcal{G}_0$ will have the same level, witnessed level and best parent in $\mathcal{G}_0$ as in $\mathcal{G}_1$. This is by the requirement that all B's parents exist in both $\mathcal{G}_0$ and $\mathcal{G}_1$ since $B \in \mathcal{G}_0$. This equivalence can be naturally extended to all quantities only depending on level, witnessed level and best parent.

2.3 Assumptions

The key assumptions used in our consensus protocols and technical discussions are as follows:

A1. Honest users should generate blocks serially. In other words, each honest user should include (directly or indirectly) all its previous blocks in every subsequent block.

A2. At most M witnesses are malicious, and M satisfies $M < 2K - N$.

A3. For honest nodes, i.e., all users running on these nodes are honest, the graph they eventually observe would be consistent with each other. That is to say, for any pair of honest nodes i and j, the graph $\mathcal{G}_i(t_i)$ node i observed at time t_i will be observed by node j at some time t_j, i.e., $\mathcal{G}_i(t_i) \subseteq \mathcal{G}_j(t_j)$.

3 Consensus in Canonchain

From the perspective of a single node in Canonchain network, the order of blocks can be established by the stable main chain of the graph it observes. Without loss of generality, we consider an arbitrary honest node, called node i. At any time t, the order of blocks whose MCI's are no greater than the MCI of $L_i(t)$ can be finalized. Therefore, the key to achieve consensus in Canonchain relies on the last stable block of local graph each node observes. The remainder of this section is organized as follows. The consistency of block ordering from each node's perspective is discussed in Section 3.1. We develop a sufficient condition in Section 3.2 under which a node can advance its last stable block. Based on this condition, Section 3.3 first describes a basic consensus algorithm, and then improves it by incorporating the idea of the last stable block from the viewpoint of a single block. The final consensus algorithm we present is implemented in Canonchain.

3.1 Consistency of Block Ordering

The prerequisite of consensus is to guarantee the consistent view of last stable blocks from different honest nodes in the network. In other words, the last stable block observed by one honest node will be eventually on the stable main chain of any other honest node in the network.

Proposition 1. *For any pair of honest nodes i and j, last stable block $L_i(t_i)$ observed by node i at time t_i will be on the stable main chain of $\mathcal{G}_j(t_j)$ for node j at some time t_j.*

Proof. The conclusion can be directly inferred from Assumption A3. and the definition of last stable block. According to Assumption A3., there exists a time t_j such that $\mathcal{G}_i(t_i) \subseteq \mathcal{G}_j(t_j)$, which implies that $\mathcal{G}_i(t_i) \subseteq \mathcal{G}_j(\tau)$ for all $\tau \geq t_j$. Since $L_i(t_i)$ is the last stable block of $\mathcal{G}_i(t_i)$, by the definition of last stable block it will be on the main chain of any graph which contains $\mathcal{G}_t(t_i)$. Specifically, $L_i(t_i)$ is on the main chain of $\mathcal{G}_j(\tau)$ for any $\tau \geq t_j$. Therefore, $L_i(t_i)$ is on the stable main chain of $\mathcal{G}_j(t_j)$. □

A direct corollary of Proposition 1 in the following shows that different honest nodes will have consistent finalized MCI for a block.

Corollary 1. *For any pair of honest nodes i and j, they will have the same finalized MCI for any block B.*

Proof. Suppose block B has finalized MCI m for node i at time t_i, i.e., there is a block B_m on the stable main chain of $\mathcal{G}_i(t_i)$ that first includes B

directly or indirectly. According to Proposition 1, there exists some time t_j such that B_m is on the stable main chain of $\mathcal{G}_j(t_j)$. Since the stable main chain between B_m and the genesis block is the same for both $\mathcal{G}_i(t_i)$ and $\mathcal{G}_j(t_j)$, B_m is also the first block on the stable main chain of $\mathcal{G}_j(t_j)$ to include B. Therefore, B has the same finalized MCI m at node j. □

Given the last stable block $L_i(t)$, node i can finalize the ordering of blocks with finalized MCI, i.e., their MCI is no greater than $L_i(t)$'s MCI, under the following rules: 1) A block with higher finalized MCI ranks higher; 2) A block ranks higher than its ancestors with the same finalized MCI; and 3) for those with the same finalized MCI and non-inclusion relationship, a block with larger hash ranks higher. The finalized order of blocks will not change as the last stable block advances. The following proposition guarantees that different honest nodes will generate consistent block ordering.

Proposition 2. *For any pair of blocks B_1, B_2 and any pair of honest nodes i, j, the finalized order of B_1 and B_2 will be the same at node i and j.*

Proof. Without loss of generality, we assume B_1 ranks lower than B_2 at node i when node i finalizes their ordering. According to Corollary 1, B_1 and B_2 have the same finalized MCI at node j as at node i. Consider the following three cases which may cause B_1 to rank lower than B_2 at node i: 1) B_1's finalized MCI is smaller than B_2's finalized MCI; 2) B_1, B_2 have the same finalized MCI and B_2 includes B_1; 3) B_1, B_2 have the same finalized MCI, B_2 does not include B_1, and B_1's hash is smaller than B_2's. In either case, the condition still holds at node j. Therefore, B_1 will rank lower than B_2 when node j finalizes their order, which completes the proof. □

3.2 Advance of Stable Block

Given a block B in a graph $\mathcal{G}$, we start from B, traverse along best parent links and stop as soon as the majority of witnesses in B's witness list has been encountered. Let $S(B)$ and $\mathcal{W}(B)$ denote the block we stopped and the set of K witness we encountered, respectively. We call the witnessed level of $S(B)$ the minimum witnessed level of B, denoted as $wl_{min}(B)$, i.e., $wl_{min}(B) = wl\,(S(B))$.

For blocks B_0 and B_1, We use $B_1 \to B_0$ and $B_1 \xrightarrow{b} B_0$ to denote that B_1 includes B_0 through parent links and best parent links, respectively. The degenerated case of $B_0 = B_1$ is regarded true, i.e., $B_0 \to B_0$ and $B_0 \xrightarrow{b} B_0$. Consider the case $B_1 \xrightarrow{b} B_0$. Let $\mathcal{C}(B_0, B_1)$ denote the set of blocks on the chain from B_1 to B_0 through best parent links, which includes B_1 not B_0.

For any block B such that $B \notin \mathcal{C}(B_0, B_1), B \xrightarrow{b} B_0,$ and $B_1 \to B$, start from B and traverse through best parent links to find out all blocks authored by witnesses from $\mathcal{W}(B_1)$ (if the same witness is encountered more than once, he is not counted again), and stop as soon as $L = 2K - N - M$ witnesses from $\mathcal{W}(B_1)$ have been encountered. By Assumption A2., L is guaranteed to be positive. We denote the witness authored block we stopped at as $W(B)$. Define $\mathcal{S}(B_0, B_1)$ as the set of all such blocks. i.e.,

$$\mathcal{S}(B_0, B_1) = \left\{ W(B) \middle| B \notin \mathcal{C}(B_0, B_1), B \xrightarrow{b} B_0, \text{ and } B_1 \to B \right\}. \quad (1)$$

We call B_0 is advanceable with respect to B_1 if the following condition holds:

$$wl_{min}(B_1) > \max_{B \in \mathcal{S}(B_0, B_1)} l(B). \quad (2)$$

Recall that $l(B)$ represents the level of block B. In the following, we give a sufficient condition that a honest node can utilize to advance the last stable block along its main chain.

Theorem 1. *If a stable block B_0 of a graph $\mathcal{G}$ is advanceable with respect to another block B_1, the direct child of B_0 in $\mathcal{C}(B_0, B_1)$ is also a stable block of $\mathcal{G}$.*

Proof. Let B_0^* be B_0's direct child in $\mathcal{C}(B_0, B_1)$. It is equivalent to show that the main chain of any graph $\mathcal{G}^* \supseteq \mathcal{G}$ will contain B_0^*. We first show the following useful result.

Lemma 1. *For any two blocks B and B^* such that $B \notin \mathcal{C}(B_0, B_1), B \xrightarrow{b} B_0$, $B_1 \to B$, and $B^* \xrightarrow{b} B$, assume no block in $\mathcal{C}(B, B^*)$ is authored by any honest witness from $\mathcal{W}(B_1)$, the following condition holds:*

$$wl_{min}(B_1) > wl(B^*). \quad (3)$$

Proof of Lemma 1. Start from block B and traverse back in time through best parent links. $W(B)$ is the block we stop at as soon as $L = 2K - N - M$ witnesses from witness set $\mathcal{W}(B_1)$ have been encountered. Also start from block B^* and traverse back in time through best parent links. And $S(B^*)$ is the block we stop at as soon as K witnesses have been encountered. We argue that $W(B) \to S(B^*)$. In fact, $\mathcal{W}(B^*)$ will contain at most $N - K$ witnesses from $\mathcal{W} \setminus \mathcal{W}(B_1)$ and M malicious witnesses by Assumption A2.. Therefore, $\mathcal{W}(B^*)$ contains at least $K - (N - K) - M = L$ honest witnesses from witness set $\mathcal{W}(B_1)$. However, by the assumption in Lemma 1, there is no

block in $\mathcal{C}(B, B^*)$ authored by any honest witness from $\mathcal{W}(B_1)$. Thus, these L honest witnesses are all in $\mathcal{W}(B)$, which implies that $W(B) \to S(B^*)$. Therefore, we have

$$wl_{min}(B_1) \overset{(a)}{>} l(W(B)) \overset{(b)}{\geq} l(S(B^*)) \overset{(c)}{=} wl(B^*), \tag{4}$$

where (a) is by the fact that $W(B) \in \mathcal{S}(B_0, B_1)$ and (2), (b) is due to $W(B) \to S(B^*)$ and (c) is by the definition of witnessed level in the case that all blocks have the same witness list. It completes the proof of Lemma 1.

Now we prove Theorem 1 by contradiction. Assume that the main chain $\mathcal{C}^*$ of a graph $\mathcal{G}^* \supseteq \mathcal{G}$ does not contain B_0^*. Since B_0 is a stable block of $\mathcal{G}$ and $\mathcal{G} \subseteq \mathcal{G}^*$, $\mathcal{C}^*$ must contain B_0 by the definition of stable block. Let B_2 denote the tip block of $\mathcal{C}^*$. Start from B_2, traverse through best parent links till B_0, we stop at the first block that is included by B_1, denoted as B_3. The existence of B_3 is due to the fact that B_0 is included by B_1. Since $B_3 \notin \mathcal{C}(B_0, B_1), B_3 \overset{b}{\to} B_0,$ and $B_1 \to B_3$, it follows that $W(B_3) \in \mathcal{S}(B_0, B_1)$.

We first show that no block in $\mathcal{C}(B_3, B_2)$ is authored by any honest witness from $\mathcal{W}(B_1)$. It is proved by contradiction. Assume there are blocks in $\mathcal{C}(B_3, B_2)$ authored by honest witnesses from $\mathcal{W}(B_1)$. Among those, let B_4 denote the one with the smallest level. As shown in Fig. 1, let B_5 denote the block from the same witness as B_4, which is in the path from B_1 to $S(B_1)$ through best parent links. Since B_4 and B_5 come from the same honest witness, by Assumption A1., we have either $B_4 \to B_5$ or $B_5 \to B_4$. By the definition of B_3, which is the first block included by B_1 when traversing from B_2 through best parent links, we have $B_4 \to B_5$. Let B_6 and B_7 be direct parents of B_4 such that $B_4 \overset{b}{\to} B_6$ and $B_7 \to B_5$, respectively. By the definition of B_4, no block in $\mathcal{C}(B_3, B_6)$ is authored by any honest witness from $\mathcal{W}(B_1)$. Therefore, we have

$$wl(B_7) \overset{(a)}{\geq} wl(B_5) \overset{(b)}{\geq} wl(S(B_1)) \overset{(c)}{=} wl_{min}(B_1) \overset{(d)}{>} wl(B_6), \tag{5}$$

where (a) is due to $B_7 \to B_5$, (b) is due to $B_5 \to S(B_1)$, (c) is by the definition of minimum witnessed level, and (d) is by (3) in Lemma 1. It contradicts with the fact that B_6 is the best parent of B_4, which completes the proof that no block in $\mathcal{C}(B_3, B_2)$ is authored by any honest witness from $\mathcal{W}(B_1)$.

Therefore, similarly as in (5) we have

$$wl(B_1) \overset{(a)}{\geq} wl(S(B_1)) \overset{(b)}{=} wl_{min}(B_1) \overset{(c)}{>} wl(B_2), \tag{6}$$

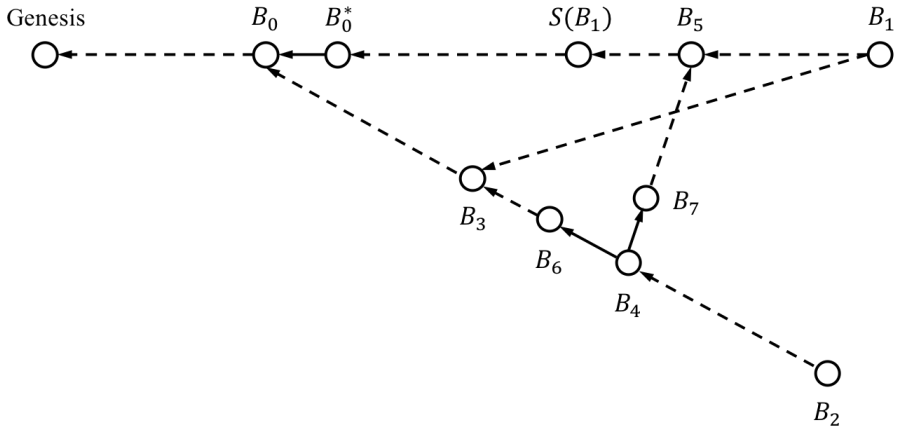

Figure 1: The case that B_4 and B_5 are from the same honest witness. Solid and dashed lines represent direct parent-child links and ancestor-descendant links, respectively.

where (a) is due to $B_1 \to S(B_1)$, (b) is by the definition of minimum witnessed level, and (c) is by (3) in Lemma 1. It contradicts the assumption that B_2 is the tip block of the main chain of $\mathcal{G}^*$, which finishes the proof of Theorem 1. □

3.3 Consensus Algorithms

Based on the results in previous two subsections, we have a basic algorithm given as Algorithm 1 to achieve consensus among different nodes in the network. The idea is to let each node advance its last stable block independently based on its local graph.

Instead of the last stable block which is expressed with respect to the whole graph a node possesses, one can define the last stable block in view of a specific block. To be precise, for block B in graph $\mathcal{G}$, let $\mathcal{G}(B)$ denote the subgraph of $\mathcal{G}$, whose blocks are included by B directly or indirectly. A block is called the last stable block in view of B or B's last stable block if it is the block with the largest level which is guaranteed to be contained in the main chain of any graph containing $\mathcal{G}(B)$. This "local view" stable block represents what a node thinks about the stable part of its local graph with information only up to B. Therefore, by receiving a block, a node can obtain the block author's option regarding which part of the graph is stable, and compare it with its own vision. Furthermore, a node can build up a chain by starting from a block, tracing back in time till the genesis block,

Algorithm 1 Basic Consensus Algorithm

Input: Local graph $\mathcal{G}_i = \{Genesis\}$ for node i.
Initialization:
Set the last stable block L_i of $\mathcal{G}_i$ to the genesis block.
Main iterations:
for all block B composed or received by node i **do**
  Perform validation procedure on B to determine whether B can be added to graph $\mathcal{G}_i$.
  if B cannot be added to $\mathcal{G}_i$ **then**
    Reject block B.
  else if B cannot be decided to add to $\mathcal{G}_i$ **then**
    Add block B in a queue for future consideration.
  else
    Add B and its parent links to graph $\mathcal{G}_i$.
    Find out the main chain of $\mathcal{G}_i$ denoted as $\mathcal{C}_i$ and its tip block B_i.
    while L_i is advanceable with respect to B_i **do**
      Set L_i to be the child of L_i on $\mathcal{C}_i$.
    end while
    Finalize the MCI's for blocks included by L_i.
  end if
end for
Output: Linear ordering of all blocks with finalized MCI.

and going from one block to its last stable block per step. Such a chain can considerably accelerate the process of synchronization for light clients, as well as the inquiry about a certain block without exhaustive search through the whole graph.

Now each block contains information about the last stable block from its viewpoint. Let $B_{lsb}(B)$ denote the information in B regarding B's last stable block. In order to reduce the degrees of freedom of adversaries, upon receiving a block B, a node is required to make sure such information is correct, i.e., $B_{lsb}(B)$ is indeed the last stable block in view of B. In Proposition 3, we develop an authentication scheme and show that it ensures the correctness of the last stable block information included in every block.

Proposition 3. *The last stable block information included in all blocks in graph $\mathcal{G}$ is valid if the following conditions hold for any block B in $\mathcal{G}$:*

1. $B_{lsb}(B) \xrightarrow{b} B_{lsb}(B^*)$, *where B^* is B's best parent;*

2. *For any block in* $\mathcal{C}(B_{lsb}(B^*), B_{lsb}(B)) \cup \{B_{lsb}(B^*)\}$ *excluding* $B_{lsb}(B)$*, it is advanceable with respect to* B*;*

3. $B_{lsb}(B)$ *is not advanceable with respect to* B*.*

Proof. We prove by induction on $\mathcal{G}$. The problem is trivial for the case that $\mathcal{G}$ only contains the genesis block. Suppose the conclusion is true for some graph $\mathcal{G}_0$. We want to show that if one more block B is added to $\mathcal{G}_0$ and the conditions in Proposition 3 hold, the information in B regarding B's last stable block is correct, i.e., $B_{lsb}(B)$ is indeed the last stable block in view of B.

Denote B's best parent in $\mathcal{G}_0$ as B^*. By the assumption that the proposition holds for $\mathcal{G}_0$, $B_{lsb}(B^*)$ is a stable block of $\mathcal{G}(B^*)$. It follows that $B_{lsb}(B^*)$ is also a stable block of $\mathcal{G}(B)$ due to $\mathcal{G}(B^*) \subseteq \mathcal{G}(B)$. Assume $B_{lsb}(B)$ satisfies the conditions in the statement of the proposition. By iteratively applying Theorem 1 to all blocks in $\mathcal{C}(B_{lsb}(B^*), B_{lsb}(B)) \cup \{B_{lsb}(B^*)\}$, from $B_{lsb}(B^*)$ to $B_{lsb}(B)$'s best parent, $B_{lsb}(B)$ is a stable block in $\mathcal{G}(B)$. Furthermore, it is assumed that $B_{lsb}(B)$ is not advanceable with respect to B. Therefore, $B_{lsb}(B)$ is the last stable block of $\mathcal{G}(B)$, which completes the proof of Proposition 3. □

A byproduct of the above scheme is that whenever a node has checked the correctness of the last stable block information in a block, it can utilize it to advance the last stable block of its local graph, which is stated by Proposition 4 in the following.

Proposition 4. *If an honest node successfully adds a block* B *to its local graph* $\mathcal{G}$*, the last stable block of the new graph either remains unchanged or becomes* $B_{lsb}(B)$*.*

Proof. Let $\mathcal{G}^*$ denote the graph by adding block B to graph $\mathcal{G}$. We define the last stable block of graph $\mathcal{G}$ and $\mathcal{G}^*$ as L and L^*, respectively. The tip block of the main chain of $\mathcal{G}^*$ is denoted as B^*. Consider the following two cases.

- B^* and B are not the same block: Since B is the only block contained in $\mathcal{G}^*$ but not in $\mathcal{G}$, we have $B^* \in \mathcal{G}$. It follows that the main chains of $\mathcal{G}^*$ and $\mathcal{G}$ are the same, which implies that the last stable block of $\mathcal{G}^*$ is the same as that of $\mathcal{G}$, i.e., $L^* = L$.

- B^* is the same as B: By the definition of the B's last stable block, $B_{lsb}(B)$ is a stable block in $\mathcal{G}(B)$. It follows that $B_{lsb}(B)$ is also a

stable block in $\mathcal{G}^*$ due to $\mathcal{G}(B) \subseteq \mathcal{G}^*$. Since $B_{lsb}(B)$ is not advanceable with respect to B, we have the last stable block of $\mathcal{G}^*$ is the same as $B_{lsb}(B)$, i.e., $L^* = B_{lsb}(B)$.

The proof of Proposition 4 is completed. □

A direct corollary of Proposition 4 is as follows.

Corollary 2. *When an honest node successfully composes a block B and adds it to its local graph $\mathcal{G}$, the last stable block of the new graph becomes $B_{lsb}(B)$.*

Proof. By the definition of best parent, B's best parent is the tip block of the main chain of $\mathcal{G}$. And it is easy see that B becomes the tip block of the main chain of the new graph, denoted as B^*. It follows that the last stable block of the new graph becomes $B_{lsb}(B)$ by following the proof of Proposition 4 for the case of $B = B^*$. □

Based on the basic consensus algorithm described in Algorithm 1, we can apply the idea of stable block in view of a single block and results in Proposition 3 and 4 to generate Algorithm 2. It is implemented in Canonchain to achieve consensus.

4 Witness Replacement

Witnesses in Canonchain are reputable users with real world identities, and users who name them expect them to act honestly. However, it is unreasonable to totally trust any single witness. We need to change the witness list of each node if some witness has lost his credibility or there are just better candidates. This section is devoted to describe in detail the witness replacement scheme implemented in Canonchain.

4.1 Merit Based Scheme

Witness based Canonchain runs iteratively with two major phases. In the first phase, which we call the witness stable phase, the witness list of each node contains the same witnesses (e.g. 12 witnesses). In the second phase, which is called the witness replacement phase, one witness will be added or removed from each node's witness list. Afterwards the system goes back to the witness stable phase. The key idea behind adding or removing one witness from the witness list is to utilize merit based criteria towards witnesses, which measure how well each witness has performed. In the following, we

Algorithm 2 Canonchain Consensus Algorithm

Input: Local graph $\mathcal{G}_i = \{Genesis\}$ for node i.
Initialization:
Set the last stable block L_i of $\mathcal{G}_i$ to the genesis block.
Main iterations:
for all block B composed or received by node i **do**
  Perform validation procedure on B to determine whether B can be added to graph $\mathcal{G}_i$.
  if B cannot be added to $\mathcal{G}_i$ **then**
    Reject block B.
  else if B cannot be decided to add to $\mathcal{G}_i$ **then**
    Add block B in a queue for future consideration.
  else
    Check whether the information inside B regarding B's last stable block is correct.
    if the information is erroneous **then**
      Reject block B.
    else
      Check the MCI of the last stable block of $\mathcal{G}_i$, denoted as m.
      Add B and its parent links to graph $\mathcal{G}_i$.
      if the MCI of B's last stable block is greater than m **then**
        Set L_i to be B's last stable block.
      end if
      Finalize the MCI's for blocks included by L_i.
    end if
  end if
end for
Output: Linear ordering of all blocks with finalized MCI's.

propose an innovative merit based scheme that each node can apply in a fully discentralized manner to achieve consensus on the performance of witnesses.

According to Proposition 1, it is guaranteed that different nodes have consistent view of last stable blocks. Therefore, whenever two nodes have the same last stable block, they have identical subgraph structure as well as finalized ordering for those blocks included by the last stable block directly or indirectly. We rely on this property to design our merit based criteria on witness performance. To be concrete, suppose the last stable MCI's at two nodes both reach beyond 100. Let B denote the stable block with MCI = 100. The subgraph consisting of all blocks included by B denoted

as $\mathcal{G}(B)$ is the same for both nodes, hence any measure which fully depends on $\mathcal{G}(B)$. For example, each node can evaluate that how many blocks a witness has composed in $\mathcal{G}(B)$, how many of those are non-serial and how many of those contribute to the main chain construction and the advance of last stable block, etc. These statistics obtained by both nodes are exactly the same. Therefore, every node in the network can independently track the witness performance as its last stable block advances.

The trigger for each node to transit from the witness stable phase to the witness replacement phase for adding one witness is that it observes some degradation on witness performance, e.g., the statistics of some witness performance falls below a predetermined threshold. In such event, one additional witness will be added into each node's witness list. The new candidate can be selected through campaign among those who either have genuine interests in caring for the network health or who have earned a good reputation in Canonchain related activities, e.g., it could be based on proof-of-stake (PoS). Time to update the witness list is different for each node, which happens when its last stable MCI increases by a fixed number from where the witness replacement phase is triggered. After the witness lists of all nodes have been augmented, Canonchain network goes back to the witness stable phase.

Things are similar for removing one witness from each node's witness list. Each node collects the statistics of witness performance based on its local graph in the witness stable phase. After a period of time, it enters the witness replacement phase for removing one witness from its witness list, based on its own evaluation of witness performance. Again, time to remove one witness is different for each node, which is triggered when its last stable MCI increases by a fixed number from where the new witness was added to its witness list. Although, each node individually determines when to remove one witness from its witness list, they can achieve consensus on which witness needs to be removed. After the removal of one witness, Canonchain network goes back to the witness stable phase.

4.2 Transition Phase

Our proposed merit based scheme is fully discentralized. Each node can apply it with respect to its local graph by tracking its last stable block and measuring witness performance using all information fully determined in view of the last stable block. Specifically, each node independently tracks witness performance before its last stable MCI advances to m_1 from m_0. After that a new witness will be selected and added to the witness list or

one witness is determined to be removed from the witness list. Since the reason to add or remove one witness is due to the performance of existing witnesses, of which all nodes in the network have the same metrics, a sensible node would choose to update its witness list in order to keep the network more healthy and robust. Ideally, each node will finish updating its witness list by adding or removing one witness before its last stable MCI reaches m_2. The gap between m_1 and m_2 is given for the community to reach consensus and nodes to update their respective witness lists. After that, the system switches back to the witness stable phase, which technically means that all blocks which include m_2 directly or indirectly will have the same witness lists.

However, this is not going to naturally happen in reality. We need to design some transition phase for the system to smoothly go back to the witness stable phase. The issue is that a node cannot guarantee that the witness list of any block it receives is identical to its owns. There are two major reasons. One is due to some malicious behavior, where some node does not update its witness list even though its last stable MCI has passed m_2. For example, the witness being removed does not give in in order to keep his profits. The other reason is because of the network delay. Different nodes will arrive at last sable MCI m_2 at different times. It results in the fact that when one node's last stable MCI reaches m_2, it can receive a block from another node whose last stable MCI has not arrived at m_2 yet, and vice versa.

Let m_3 denote the MCI of the first block on stable main chain whose last stable block's MCI exceeds m_2. Before a node's last stable MCI reaches m_2, assume its witness list is updated from $\mathcal{L}$ to $\mathcal{L}^*$ by adding or removing one witness. For the case of removing one witness, each node will be aware of $\mathcal{L}^*$ by looking into the witness performance based on its local graph when its last stable MCI reaches m_1. However, things are different for the case of adding one witness. This is because a node does not know which witness will be added at the time its last stable MCI reaches m_1. But a node can figure out what $\mathcal{L}^*$ is without updating. For example, a node can keep track of the new witness included in the witness list of blocks authored by witnesses in $\mathcal{L}$. If more than M of the new witnesses are the same, it will be the actual one added to $\mathcal{L}$ to generate $\mathcal{L}^*$. Now we propose the following rules for a node to determine how to deal with a block B it receives whose last stable block has MCI m, where $m_0 \leq m \leq m_3$. Here we assume that B has passed the validation procedure excluding the one regarding witness list.

R1. If $m_0 \leq m \leq m_1$, B is accepted if its witness list is $\mathcal{L}$, and rejected

otherwise;

R2. If $m_1 < m \leq m_2$, if the node has not figured out what $\mathcal{L}^*$ is, B will be put into a queue for future revisit; or B is accepted if its witness list is $\mathcal{L}$ or $\mathcal{L}^*$, and rejected otherwise.

R3. If $m_2 < m \leq m_3$, B is accepted if its witness list is $\mathcal{L}^*$, and rejected otherwise;

Technically, the transition phase is defined to be the process of advancing the last stable MCI from m_2 to m_3. The following Proposition 5 assures that the system will go back to the witness stable phase afterwards.

Proposition 5. *For any node in the network, let $\mathcal{L}^*, B^*$ denote the updated witness list and last stable block when its last stable MCI reaches m_3, respectively. Any block it successfully receives which includes B^* will have the same witness list $\mathcal{L}^*$.*

Proof. For any block B which satisfies $B \to B^*$, we have $B_{lsb}(B) \to B_{lsb}(B^*)$. In fact, $B_{lsb}(B^*)$ is a stable block in $\mathcal{G}(B^*)$. It follows that $B_{lsb}(B^*)$ is also a stable block in $\mathcal{G}(B)$ due to $\mathcal{G}(B^*) \subseteq \mathcal{G}(B)$. Thus, $B_{lsb}(B) \to B_{lsb}(B^*)$ holds since $B_{lsb}(B)$ is the last stable block in $\mathcal{G}(B)$. By the definition of B^*, the MCI of $B_{lsb}(B^*)$ is larger than m_2. Hence, the MCI of $B_{lsb}(B)$ is also larger than m_2. By R3. in our designed protocol, if B can be accepted, its witness list should be identical to $\mathcal{L}^*$, which completes the proof of Proposition 5. □

4.3 Witness Replacement Protocol

We summarize our proposed witness replacement protocol discussed in previous subsections in Fig. 2. It only depicts one cycle of three phases from the viewpoint of an arbitrary honest node in the network. The horizontal line in Fig. 2 represents the advancement of the last stable MCI. In the following, we describe in detail what to fulfill inside each of the three phases.

P1. The witness stable phase is between MCI m_0 and m_1. During this phase, each node collects statistics to measure witness performance, which triggers the witness replacement phase at MCI m_1.

P2. The witness replacement phase is between MCI m_1 and m_2. During this phase, one witness will be added or removed from each node's witness list. Which witness to add is achieved by consensus among the community, e.g., it could be based on PoS; whilst which witness to

remove can be determined by each node independently, which acts as a virtual consensus among the community. The update on the witness list is mandatory in order to keep the network healthy and robust.

P3. The transition phases happens between MCI m_2 and m_3. It technically guarantees that the network will go back to the witness stable phase after m_3. And m_3 works as a new m_0 for the next cycle.

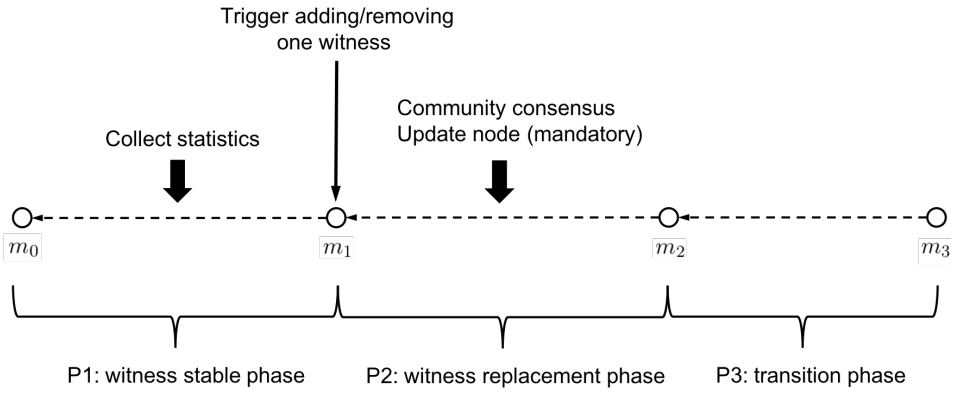

Figure 2: Three-phase witness replacement protocol in Canonchain

We need to make sure that all honest nodes in the network have consistent view on a block, i.e., a block is either rejected properly by all honest nodes or accepted and included into their respective local graphs. It is guaranteed by Proposition 6 in the following.

Proposition 6. *For any block B, it is either accepted or rejected properly by all honest nodes in the network.*

Proof. We consider the cases that B has passed validation procedure excluding the one regarding witness list, and it has correct information about its last stable block. Because otherwise, the error will be detected at any honest node, which results in the rejection of B. Let m deonote B's last stable MCI. Without loss of generality, we assume $m_0 \leq m \leq m_3$, where m_0 till m_3 are defined as before. Also let $\mathcal{L}$ and $\mathcal{L}^*$ denote the witness list before and after the update, respectively. Consider the following scenarios:

- If $m_0 \leq m \leq m_1$, according to Corollary 2, the last stable MCI of B's author has not yet reached m_1 when composing B. It implies that B's author has not updated its witness list at that time, i.e., B's witness

list should be $\mathcal{L}$. Therefore, B will be accepted by any honest node if B's witness list is L, and rejected otherwise.

- If $m_1 < m \leq m_2$, the last stable MCI of B's author has reached beyond m_1 by Corollary 2. It follows that B's authors has entered the witness replacement phase. If B's witness list is $\mathcal{L}$ or $\mathcal{L}^*$, it will be accepted by any honest node as soon as the node figures out what the updated witness list $\mathcal{L}^*$ is. Otherwise, B will be rejected.

- If $m_2 < m \leq m_3$, the last stable MCI of B's author has reached beyond m_2 according to Corollary 2. Therefore, if B's witness list is $\mathcal{L}^*$, it will be accepted any honest node. Otherwise, it will be rejected by any honest node. And the rejection is justified because the last stable MCI of B's author has exceeded m_2 but it has not updated the witness list yet.

Therefore, any honest node will have the same decision towards B, which completes the proof of Proposition 6. □

4.4 Revisit to Advance of Stable Block

In Section 3.2, we developed a sufficient condition under which an honest node can advance its last stable block. But the technical proof for Theorem 1 is only valid when all nodes have the same witness list. In this section, we will discuss on how to extend the results in Section 3.2 for advance of last stable block in one cycle of our witness replacement protocol.

Let N denote the number of witnesses in the witness stable phase. And M and K have the same definitions as previous. We assume $M < 2K-N-2$. Based on our protocol in Fig. 2, when determining whether a block B_0 is advanceable with respect to another block B_1, we only need to consider the case that all blocks of interest have witness list L or L^*. Here, L^* is the witness list by adding or removing one witness from L. For any block B such that $B \notin \mathcal{C}(B_0, B_1), B \xrightarrow{b} B_0$, and $B_1 \to B$, we redefine $W(B)$ in Section 3.2 with a different $L = 2K - N - M - 2$. We still call B_0 is advanceable with respect to B_1 if condition (2) holds. Lemma 2 in the following is an replacement of Lemma 1, such that Theorem 1 is applicable in the case of witness replacement.

Lemma 2. *For any two blocks B and B^* such that $B \notin \mathcal{C}(B_0, B_1), B \xrightarrow{b} B_0$, $B_1 \to B$, and $B^* \xrightarrow{b} B$, assume no block in $\mathcal{C}(B, B^*)$ is authored by any*

honest witness from $\mathcal{W}(B_1)$, the following condition holds:

$$wl_{min}(B_1) > wl(B^*). \tag{7}$$

Proof. Start from block B and traverse back in time through best parent links. $W(B)$ is the block we stop at as soon as $L = 2K-N-M-2$ witnesses from witness set $\mathcal{W}(B_1)$ have been encountered. Also start from block B^* and traverse back in time through best parent links. And $S(B^*)$ is the block we stop at as soon as K witnesses have been encountered. However, we do not always have $wl(B^*) = l(S(B^*))$ since the witness list of B^* could be one short of the witness list of B^*'s best parent. By the definition of witnessed level, we have

$$wl(B^*) = l(S(B'))), \tag{8}$$

where $B^* \xrightarrow{b} B'$. Denote $\mathcal{S}$ as the set of all distinct witnesses from B^* to B' through best parent links, which are also from B^*'s witness list. The cardinality of $\mathcal{S}$ is either K or $K-1$.

We argue that $W(B) \to S(B')$. In fact, $\mathcal{S}$ will contain at most $N+1-K$ witnesses outside $\mathcal{W}(B_1)$ and M malicious witnesses by Assumption A2.. Therefore, $\mathcal{S}$ contains at least $K-1-(N+1-K)-M = L$ honest witnesses from witness set $\mathcal{W}(B_1)$. However, by the assumption in Lemma 2, there is no block in $\mathcal{C}(B, B^*)$ authored by any honest witness from $\mathcal{W}(B_1)$. Thus, these L honest witnesses are all in $\mathcal{W}(B)$, which implies that $W(B) \to S(B')$. Therefore, we have

$$wl_{min}(B_1) \overset{(a)}{>} l(W(B)) \overset{(b)}{\geq} l(S(B')) \overset{(c)}{=} wl(B^*), \tag{9}$$

where (a) is by the fact that $W(B) \in \mathcal{S}(B_0, B_1)$ and (2), (b) is due to $W(B) \to S(B')$ and (c) is by (8). It completes the proof of Lemma 2. □

Acknowledgement

The author would like to thank Xiang Gao, Dr. Sichao Yang and Dr. Chong Li for their helpful discussions during the draft of this paper.

References

[1] Satosh Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. `http://www.bitcoin.org/bitcoin.pdf`.

[2] Vitalik Buterin. Ethereum whitepaper. https://github.com/ethereum/wiki/wiki/White-Paper.

[3] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. Lecture Notes in Computer Science, pages 507–527. Springer, 2015.

[4] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol, 2016. https://eprint.iacr.org/2016/1159.

[5] Yonatan Sompolinsky and Aviv Zohar. Phantom: A scalable blockdag protocol, 2018. https://eprint.iacr.org/2018/104.

[6] Serguei Popov. The tangle. https://iota.org/IOTA_Whitepaper.pdf.

[7] Anton Churyumov. Byteball: A decentralized system for storage and transfer of value. https://byteball.org/Byteball.pdf.