OPERA Chain:
A DAG-based Lachesis Consensus Algorithm

Sang-Min Choi[a], Jiho Park[a], Kiyoung Jang[a], Hyunjoon Cheon[a], Yo-Sub Han[a], Byung-Ik Ahn[b]

[a]*Department of Computer Science, Yonsei University, 50 Yonsei-Ro, Seodaemun-Gu, Seoul 03722, Republic of Korea*
[b]*FANTOM, 10, Teheran-ro 20-gil, Gangnam-gu, Seoul, Republic of Korea*

Abstract

OPERA chain is a Directed Acyclic Graph (DAG) generated by Lachesis protocol and implemented for Byzantine fault tolerance. We propose Lachesis protocol using a cost function that can construct equational distribution in asynchronous network. We also design Lachesis consensus algorithm that can reach a consensus on the OPERA chain generated by Lachesis protocol. We consider 2/3 of all participants' agreement to a event as consensus. It indicates that Lachesis consensus algorithm can implement Byzantine fault tolerance. In our algorithm, we design a flag table that has connection information of blocks in specific parts and check share information of some blocks. Addressed the flag table, the consensus algorithm searches not whole paths but partial paths to detect share of each block for participants. We can expect reduction of time complexity for reaching consensus in our algorithm. In this paper, we provide proof of our consensus algorithm and response to expected attacks.

Keywords: Consensus algorithm, Byzantine fault tolerance, Lachesis protocol, Clotho, Atropos, Main chain

Contents

1. Introduction

With the appearance of Blockchain, many industrial fields expect commercialization of distributed ledger technology [11]. But delay of processing time for consensus and high commission obstruct the commercialization. The delay of processing time is concerned with the consensus algorithms. The consensus algorithm is that all members who participate a network can share the transactions and agree integrity of the shared transactions in distributed situations [2, 4]. It is equivalence to the proof of Byzantine fault tolerance in distributed database systems [2, 4]. Proof of Work (PoW) Blockchain requires lots of computation works to generate the blocks by participants [8]. It leads to not only delay of connecting blocks but necessity of high performance machines. The machines compute nonce values since PoW utilizes work levels to find nonce values as consensus. There are other consensus algorithms to compensate this computation concept. Some representative concepts are Proof of Stake (PoS) [10], Delegated Proof of Stake (DPoS) [5], and the algorithms using directed acyclic graph (DAG) [7]. PoS and DPoS use participants' stakes and delegated participants' stake to generate the blocks respectively. The DAG concepts address graph structures to decide consensus. In these concepts, block and connection are considered as node and edge, respectively. The purpose of DAG concepts is reduction of commision for network participants and improvement of the delay of consensus. The typical DAG concepts are Tangle [9], Byteball [3], and Hashgraph [1]. Tangle selects the blocks to connect network utilizing accumulated weight of nonce and Monte Carlo Markov Chain (MCMC). Thus, the consensus is decided through connection information and accumulated weight. Byteball generates main chain from DAG and reaches consensus through index information of the chain. Hashgraph connects each block from a member to other in random. Based on the connections, Hashgraph searches whether 2/3 members can reach each block or not and provides proof of Byzantine fault tolerance through the graph search. Each method does not require plenty of work such as PoW of Blockchain and reduces commissions since Tangle, Byteball, and Hashgraph do not require high computing power. Although each method has their own strengths, there are some improvement points. We propose Lachesis protocol that can construct equational distribution and consensus algorithm to improve previous methods. We also describe the proof of Byzantine fault tolerance for the graphs structured by Lachesis protocol. In Lachesis consensus algorithm, we design a flag table that has connection

information of blocks in specific parts. Our consensus algorithm searches not whole paths to detect share of each block for participants using the flag table. In our algorithm, we check share information of some blocks using the flag table and can expect reduction of time complexity.

We first describe previous approaches of DAG and our main concepts in the remaining of this section. In Chapter 2, we explain elements and actions of Lachesis protocol. We describe Lachesis consensus algorithm using flag table in Chapter 3. Proof of Byzantine fault tolerance is described in Chapter 4. In Chapter 5, we introduce response to attack in Lachesis protocol and consensus algorithm. Finally, we conclude this paper along with the future appending works in Chapter 6.

1.1. Previous Approaches of DAG

1.1.1. Tangle

IOTA [9] published a DAG-based technology called tangle. Tips concept was used to address scalability issues with the limitations of the Internet of Things. Also, a nonce by using weight level was composed to achieve the transaction consensus by setting the user's difficulty.
To solve the double spending problem and parasite attack, they used the Markov Chain Monte Carlo (MCMC) tip selection algorithm, which randomly selects the tips based on the size of the accumulated transaction weights. However, if a transaction conflicts with each other, there is still a need to examine all past transaction history to find it.

1.1.2. Byteball

Byteball [3] used an internal pay system called bytes. This is used to pay adding data to distributed database. Each storage unit is linked to each other that includes one or more hashes of earlier storage units. In particular, the consensus ordering is composed by selecting a single Main Chain, which is determined as a root consisting of the most witnesses.
A majority of witnesses detects the double-spend attempts through consensus time of Main Chain. The fee is charged according to the size of the bytes, and the list of all units should be searched and updated in the process of determining the witnesses.

1.1.3. RaiBlocks

RaiBlocks [6] has been developed to improve high frees and slow transaction processing, and it is a process of obtaining a consensus through the

balance weighted vote on conflicting transactions. Each node participating in the network becomes the principal and manages its data history locally. However, since RaiBlocks generate transactions in a similar way to an anti-spam tool of PoW, all nodes must communicate to crate transactions. Also, in terms of scalability, there is a need for steps to verify the entire history of transaction when a new node is added.

1.1.4. Hashgraph

Hashgraph [1] is an asynchronous DAG-based distributed ledger method. Each node is connected by its own ancestor and randomly communicates the event blocks through a gossip protocol. At this time, any famous node can be determined by the *see* and strong see at each round to reach an consensus quickly. They state that if more than 2/3 of the nodes reach consensus for an event, it will be assigned consensus position.

1.2. Main Concepts

- **Event block** all node can create event blocks as time t. The generated event block structure includes the signature, generation time, transaction history information, flag table information, and hash information to reference. The information of the referenced event block can be copied by each node.

- **Lachesis protocol** Lachesis protocol is rule to communicate between nodes. When each node creates event blocks, it determines which node chooses other node based on cost function. The cost function is applied to prevent the behavior of inactive nodes and distributes them randomly.

- **Share** Share is the relationship between nodes which have event blocks. If there is a path from an event block x to y, x share y. "x share y" means that a node creating x know event block y.

- **Supra-share** Based on Share concept, an event block x supra-share event block y when a node creating event block x knows that more than 2/3 of nodes know event block y.

- **Flag table** Flag table stores share information from a event block to each clotho, which is how many nodes share the Clotho. Flag table is utilized for assigning Atropos event block.

- **Clotho** Clotho is not only the first generated event blocks by each node, but event blocks that supra-share with more than two-thirds of previous Clotho. Every Clotho can be candidate of Decima and it has important roles for ensuring reliability between nodes.

- **Decima** Decima is some of Clotho which satisfy that they are supra-shared and more than two-thirds nodes know the information that they are supra-shared. Decima can be candidate of Atropos and it has important roles for ensuring consensus of event blocks.

- **Atropos** Atropos is assigned consensus time through the Lachesis consensus algorithm and is utilized for determining the order between event blocks. They have the ability to form Main-chain, and this allows for an attack response and time consensus ordering.

- **Reselection** To solve byzantine agreement problem, each node reselects consensus time for a Decima, based on the collected consensus time in previous generation. When the consensus time reaches byzantine agreement, Decima is confirmed as Atropos and used for time consensus ordering.

- **Main-Chain** Main-chain is the important role in OPERA chain. It is the set of the certain event blocks called Atropos. Thus, OPERA chain uses Main-chain to find rapidly ordering between event blocks. In OPERA chain, Each event blocks is assigned the proper consensus position.

2. Lachesis Protocol

The OPERA chain is one of the DAG-based consensus algorithm. In the OPERA chain, we call the participant member. The member participates the OPERA chain, however all of the members cannot create event block. Some of the members called node have authority of creating event blocks. The nodes are selected by for all nodes voting such as Delegated Proof of Stake (DPoS). Thus, the number of nodes is kept a fixed number. Figure 1 shows an example of OPERA chain constructed through Lachesis protocol.

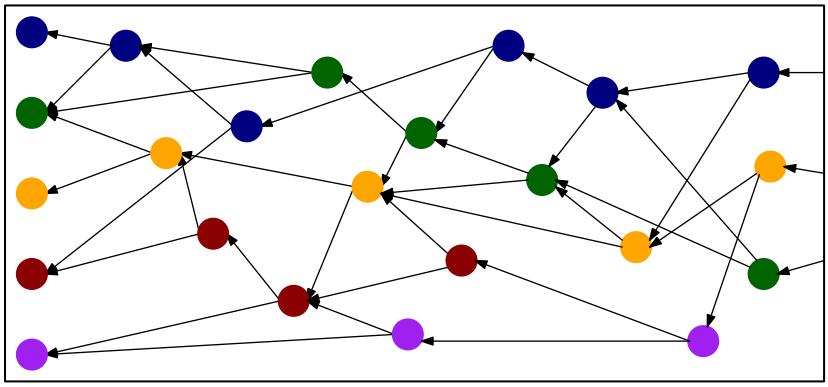

Figure 1: An Example of OPERA Chain

The OPERA chain is worked by nodes through Lachesis Protocol and Lachesis Consensus Algorithm. Lachesis Protocol is constructed on asynchronous communication between nodes in OPERA chain. Lachesis Protocol consists of event block including user information such as smart contract and edge between event blocks. In Lachesis Protocol, event blocks are created by a node after the node communicates information of OPERA chain with another node. Lachesis Consensus Algorithm is one of the consensus algorithm for solving the byzantine agreement problem. In Lachesis Consensus Algorithm, the OPERA chain uses Clotho and Atropos to finding consensus time for event blocks. Algorithm 1 shows the pseudo algorithm of OPERA chain. The algorithm consists of two parts and runs them in parallel. In a part, each node requests synchronization and creates an event block. In line 4, a node runs Node selection algorithm. Node selection algorithm returns ID of another node to communicate with it. In line 5 and 6, the node requests synchronization and synchronizes OPERA chain. Line 7 runs Event block creation. In Event block creations, the node creates an event block and checks whether it is Clotho. In line 8 and 9, Decima selection and Atropos time consensus operate. They are for checking whether Clotho can be Decima and assigning the consensus time and confirm as Atropos. The other part is replying synchronization of OPERA chain. In line 11 and 12, the node receives a request for synchronization and communicate and exchange OPERA chain with the communication partner.

The rest of this section introduces component and protocol algorithm of Lachesis Protocol. Lachesis Consensus Algorithm is introduced in detail at the next section.

Algorithm 1 Main Procedure

```
1: procedure MAIN PROCEDURE
2: loop:
3:     A = Node selection algorithm()
4:     request sync to a node A
5:     sync all known events by Lachesis protocol
6:     Event block creation
7:     Decima selection
8:     Atropos time consensus
9: loop:
10:    request sync from a node
11:    sync all known events by Lachesis protocol
```

2.1. Node Structure

In Lachesis Protocol, node is some of members with authority creating event blocks. Figure 2 shows example of node structure component.

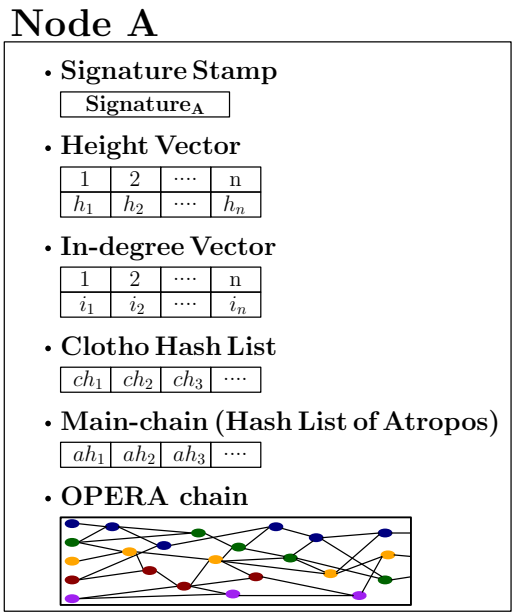

Figure 2: An Example of Node Structure

Each node has signature stamp, height vector, in-degree vector, Clotho hash list, and Main-chain. Signature stamp is data structure for storing hash

value that indicate the most recently created event block by the node. We call the most recently created event block into top event block. FigureFigure 2 is an example of component in node A. In figure 2, signature A indicates hash value of top event block. Each value in height vector is the number of event blocks created by other nodes respectively. In figure 2, h_i is the number of event blocks created by i^{th} node. Each value in in-degree vector is the number of edges from other event blocks created by other nodes to top event block. Clotho hash list is data structure storing hash values of Clotho. Main-chain is data structure storing hash values of Atropos. Main-chain is used to find event blocks with complete consensus. Clotho and Atropos selection algorithm is introduced in chapter 3.

2.2. Event Block Structure

In this section, we introduce event block structure. An event block includes own hash value and event information which each nodes generates. Hash value is used for detecting and preventing data forgery.

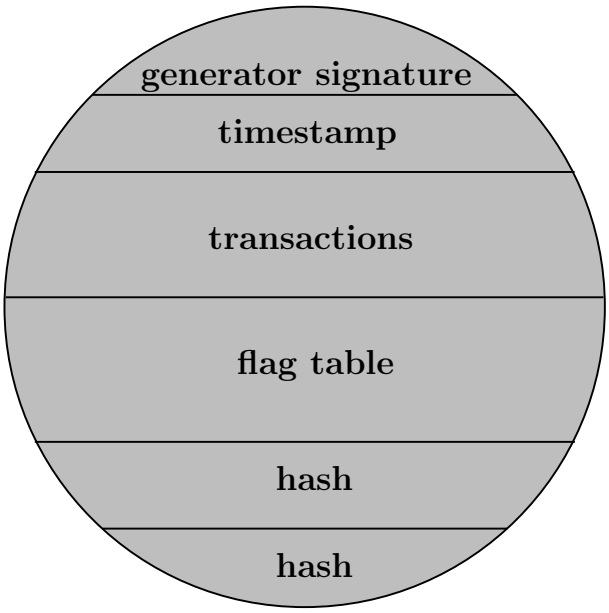

Figure 3: A Structure of Event Block

Figure 3 shows the event block component used in OPERA chain. An event block includes signature indicating who create the event block, timestamp indicating when the event block is created, transaction indicating what

the event block stores, flag table indicating how many nodes share previous event blocks and two hash values indicating what the event block refer to.

2.3. Lachesis Protocol

Lachesis protocol works when nodes create event block. For creating event block, each event block refers to two other event block. Reference means that event block store hash value of other event block. In Lachesis protocol, event block refer to two other event blocks under the certain conditions as follows,

1. The reference event blocks are top event blocks.
2. One of two reference event blocks is own top event block.

We define Cost Function (C_F) for preventing the creation of isolated nodes and ensuring the even distribution of connections between nodes. When a node creates an event block, the node selects another node with low cost function and refer to top event block of the reference node. An equation (1) of C_F is as follows,

$$C_F = I/H \tag{1}$$

where I and H denote certain values of in-degree vector and height vector respectively. If the number of nodes with the lowest C_F is more than two, one of the nodes is selected as random. If we select the node that have high H, we can expect high possibility to create Clotho because the high H indicates that the communication frequency of the node had more opportunities than others with low H. Algorithm 2 shows the selecting algorithm for reference node. The algorithm operates for each node to select communication partner from other nodes. Line 4 and 5 set min_cost and S_{ref} to initial state. Line 7 calculates cost function c_f for each node. In line 8, 9, and 10, find minimum value of cost function and set min_cost and S_{ref} to c_f and ID of each node respectively. Line 11 and 12 append ID of each node to S_{ref} if min_cost equals c_f. Finally, line 13 selects randomly one node ID from S_{ref} as communication partner. The time complexity of Algorithm 2 is $O(n)$, where n is the number of nodes.

After the reference node is selected, two nodes communicate and share information that is all event blocks known by them. A node creates an event block by referring to the top event block of the reference node. The Lachesis protocol works and communicates asynchronously. This allows a node to create an event block asynchronously even when another node creates an

Algorithm 2 Node Selection

```
1: procedure NODE SELECTION
2:     Input: Height Vector H, In-degree Vector I
3:     Output: reference node ref
4:     min_cost ← INF
5:     s_ref ← None
6:     for k ∈ Node_Set do
7:         c_f ← I_k / H_k
8:         if min_cost > c_f then
9:             min_cost ← c_f
10:            s_ref ← k
11:        else if min_cost equal c_f then
12:            s_ref ← s_ref ∪ k
13:    ref ← random select in s_ref
```

event block. The communication between nodes does not allow simultaneous communication with the same node. Figure 4 shows the process of Lachesis protocol.

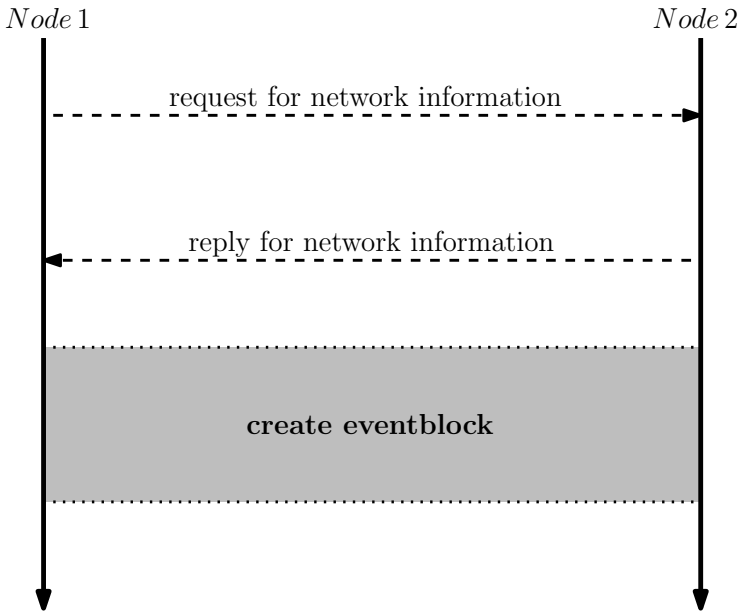

Figure 4: The Process of Lachesis Protocol

3. Lachesis Consensus Algorithm

Lachesis Consensus Algorithm (LCA) is a process that examines whether event block is shared with supra-majority nodes. Sharing an event block with supra-majority nodes means that more than two-thirds of all nodes in OPERA chain knows the event block. For the supra- majority sharing, some event blocks are called Clotho if the event blocks are linked to more than two-thirds previous Clotho shared with supra-majority nodes. When an event block is determined to Clotho, the Clotho includes flag table that shows how many nodes know previous Clotho. Using flag table, we select Decima and Atropos from Clotho by Decima selection algorithm and Atropos consensus time selection algorithm. These algorithms check whether the Atropos is satisfied with byzantine fault tolerance. The rest of this section shows flag table construction algorithm, Clotho selection algorithm, Decima selection algorithm, and Atropos consensus time selection algorithm.

3.1. Main-chain

In LCA, Main-chain is the important role for ordering the event blocks. Main-chain is the virtual path to connect between the Atropos. Each event blocks can search own consensus position by checking the nearest Atropos event block. Each node has information for Main-chain and is able to search consensus position of event blocks. Assigning and searching consensus position are introduced in the consensus time selection section. Figure 5 shows the example of Main-chain composed of 4 Atropos event blocks. The following sections describe the Clotho selection algorithm, Decima selection algorithm and Atropos consensus time selection algorithms required to create the Main-chain.

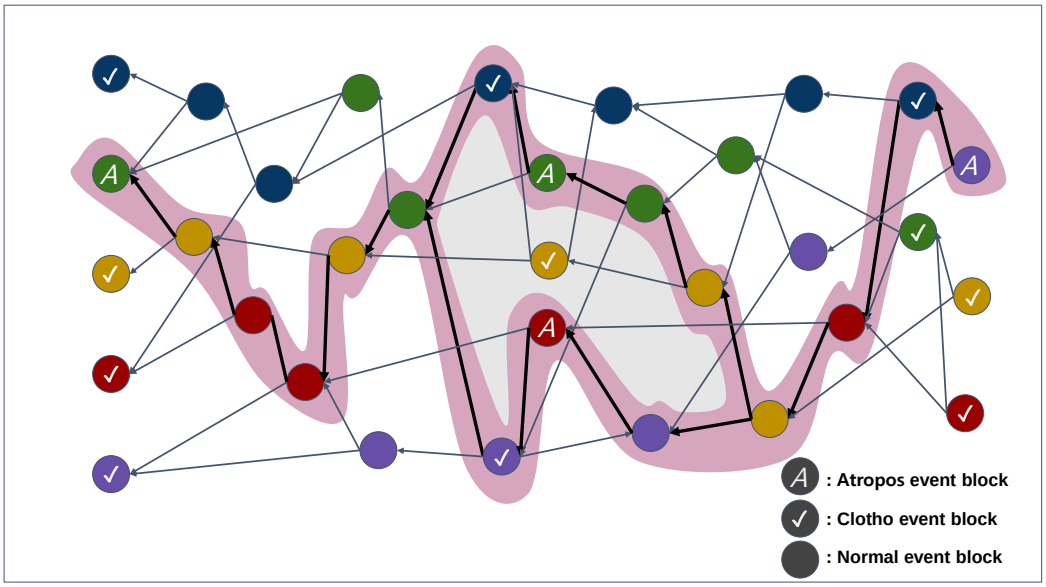

Figure 5: An Example of Main-chain

3.2. Clotho Selection

All nodes create event blocks by itself, and the first event blocks (or genesis) are regarded as Clotho. Each event block can refer to more than two-thirds previous Clotho shared with supra-majority nodes. Each node has a Clotho hash list that is a space to preserve the hash values. The flag table is consists of bits that support determining the Atropos, and Clotho only has its flag table information for each generation. The process of Clotho selection algorithm is as follows.

1. The first event blocks which means all event blocks in genesis generation by each node are considered Clotho. The Clotho has a flag table.
2. When new event block is added in OPERA chain, we check whether the event block can be Clotho using depth first search (DFS). If the event block supra-shares more than two-thirds of all previous Clotho event blocks, the new event block becomes Clotho.
3. All Clotho include connection information in flag table. If one of new event blocks becomes Clotho, the Clotho bit in the event block has true value and fills the accumulated share information with previous Clotho set to own flag table.

4. When new Clotho appear on OPERA chain, some nodes update Clotho hash list. If one of new event blocks becomes Clotho, all nodes that share the new event block add the hash value of the event block to their Clotho hash list.
5. The generation is increased by 1 with appearance of new Clotho.

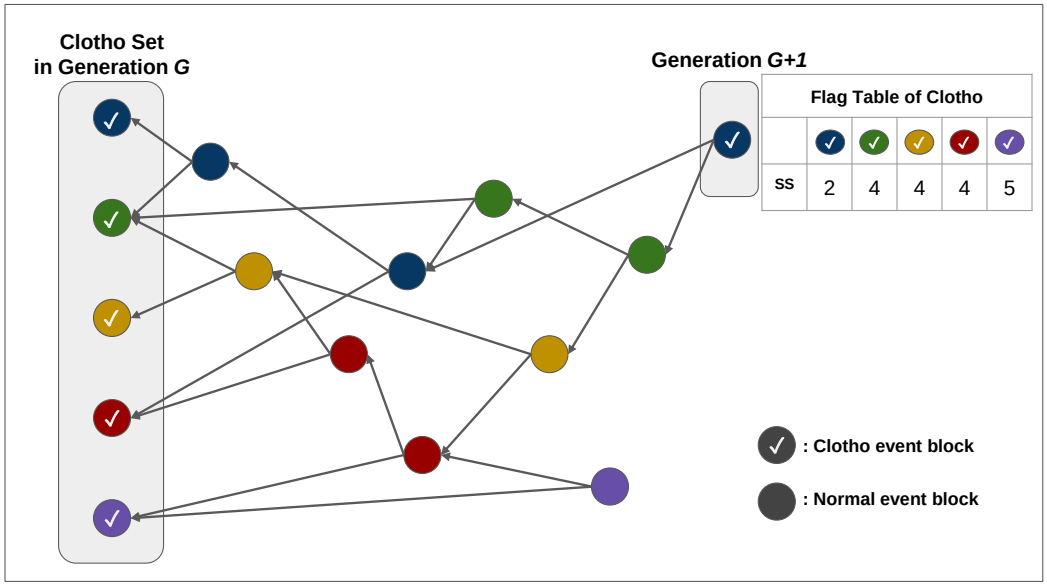

Figure 6: An Example of Clotho

Figure 6 shows an example of flag table of a Clotho. In Figure 6, a circle indicates an event block and a circle with $\surd$ mark means a Clotho. There are five participant nodes in this example and event blocks are classified into different colors according to nodes that generate event blocks. We can show the five Clotho event blocks in generation G. When an event block is added to OPERA chain, depth first search runs to check whether the event block is Clotho. The event block in generation $G+1$ in Figure 6 is the first Clotho of $G+1$ and can have flag table. Flag table is $2 \times n$ matrix where n is the number of nodes. According to this structure, the flag table in Figure 6 has 2×5 matrix. The first row indicates the hash value of previous Clotho created by participant nodes and second row means the number of nodes from the event block to previous Clotho. We can consider the number of nodes on the paths as the share. For example, the flag table value of first

Clotho is 2. It means that the event blocks on all paths from the Clotho in $G+1$ to the first Clotho in G are generated by two nodes. In the same way, we can compute each value in flag table.

3.3. Decima Selection

Decima selection algorithm checks whether Clotho event blocks in the Clotho hash list satisfy the Decima condition. Decima condition means whether the Clotho is supra-shared and more than two-thirds nodes know the information that the Clotho is supra-shared. Decima selection algorithm operates when a Clotho is created. If a Clotho satisfy Decima condition, the Clotho is Decima and the candidate time for Atropos is time when a Clotho satisfy the condition. After the Clotho is nominated for Decima, Atropos consensus time selection algorithm operates. Algorithm 3 shows the pseudo code of Decima selection. The algorithm operates when each node creates Clotho. Line 4 and 5 set *c.Decima* and *c.yes* to *undecided* and 0 respectively. line 6 and 7 check whether Clotho c' in G_{i+1} supra-shares Clotho c and *c.yes* adds 1 if a Clotho c' supara-shares Clotho c. In line 8 and 9, checks whether the number of Clotho in g+1 which supra-share Clotho c is more than 2n/3 and declares the Clotho c' to Decima. The time complexity of Algorithm 3 is $O(n^2)$, where n is the number of nodes.

Algorithm 3 Decima Selection

```
1: procedure DECIMA SELECTION
2:     for c ∈ Clotho_Set in Generation(G_i) do
3:         c.Decima ← undecided
4:         c.yes ← 0
5:         for c ∈ Clotho_Set in Generation(G_{i+1}) do
6:             if SS_c in c' > 2n/3 then
7:                 c.yes += 1
8:         if c.yes > 2n/3 then
9:             c.Decima ← yes
```

3.4. Atropos Consensus Time Selection

Atropos consensus time selection algorithm is the process in which the candidate time generated from Decima selection is shared with more than two-thirds of all nodes and results in a consensus. After the Decima is nominated, nodes check the candidate time of each node for the Decima and check

a condition that the number of nodes which have same value of candidate time is more than two-thirds of all nodes. If the condition is not satisfied, each nodes reselects candidate time from some candidate time which the node collects. By reselection process, each node reaches time consensus for candidate time of Decima as OPERA chain grows. The candidate time reaching the consensus is called Atropos consensus time. After deciding Atropos consensus time, Decima is nominated to Atropos and each node stores the hash value of Atropos and Atropos consensus time in Main-Chain. The Main-chain is used for time order between event blocks. The proof of Atropos consensus time selection is shown in the section 4.

Algorithm 4 Atropos Consensus Time Selection

```
1:  procedure ATROPOS CONSENSUS TIME SELECTION
2:      Input: c.Decima in Generation(G_i)
3:      c.consensus_time ← undecided
4:      for c' ∈ Clotho_Set in Generation(G_j) do
5:          d ← j-i
6:          if d > 1 then
7:              if d is 2 then
8:                  c'.time ← c'.time_stamp
9:              else
10:                 s ← the set of Clotho in j-1 that c' can supra-shares
11:                 t ← RESELECTION(s)
12:                 k ← the number of Clotho having t in s
13:                 if d mod h > 0 then
14:                     if k > 2n/3 then
15:                         c.consensus_time ← t
16:                         c'.time ← t
17:                     else
18:                         c'.time ← t
19:                 else
20:                     c'.time ← the minimum value of candidate time in s
```

Algorithm 4 and 5 show pseudo code of Atropos consensus time selection and Consensus time reselection. In Algorithm 4, line 6 means that c' has bigger generation than c. In line 6, d saves the deference of generation. Line 8 and 9, each Clotho in j selects own timestamp as candidate time when they confirm Clotho c as Atropos. In line 11, 12, and 13, s, t, and k save the set of

Clotho that c' has supra-share, the result of $RESELECTION$ function, and the number of Clotho in $j-1$ having t. Line 15-19 is checking whether more than two-thirds of Clotho in $j-1$ select same candidate time. If two-thirds of Clotho in $j-1$ select same candidate time, the Clotho c is assigned consensus time. Line 21 is minimal value selection round. In minimal value selection round minimum value of candidate time is selected to reach byzantine agreement. Algorithm 5 operates in the middle of Algorithm 4. In Algorithm 5, input is the set of Clotho and output is reselected candidate time. Line 4-10 is classifying process how many each candidate times is selected in the set of Clotho. In line 11-17, a candidate time which is the most selected and the smallest is reselected. The time complexity of Algorithm 5 is $O(n)$ where n is the number of nodes. Since Algorithm 4 includes Algorithm 5, the time complexity of Algorithm 4 is $O(n^2)$ where n is the number of nodes.

Algorithm 5 Consensus Time Reselection

```
1:  function RESELECTION
2:      Input: the set of Clotho S
3:      Output: candidate time t
4:      D is dictionary data structure
5:      D.initialize()
6:      for Clotho c ∈ S do
7:          if c.time in D.keys() then
8:              D[c.time] ← D[c.time] + 1
9:          else
10:             D[c.time] ← 1
11:     max_value ← 0
12:     t ← infinite
13:     for key k, value v ∈ D do
14:         if max_value < v then
15:             max_value] ← v
16:         else if max_value == v && t > k then
17:             t ← k
18:     return t
```

In the above paragraph, Atropos Consensus Time Selection expressed as if each node reach consensus agreement by exchanging candidate time with each other. However, when each node communicates with each other

through the Lachesis protocol, the OPERA-chain that each node has is consistent with OPERA-chain in other node. This allows each node to know the candidate time of other nodes based on its OPERA-chain and reach a consensus agreement. We call this virtual agreement. The proof of that virtual agreement and actual agreement reaches same result is shown in the section 4.
A consensus time order between event blocks is decided by Atropos consensus time selection and timestamp of event block. Atropos consensus time is the smallest consensus time among the Atropos sharing the event block. Deciding time order between two event blocks, the event block with a smaller Atropos consensus time is determined to the forward position. If the event blocks have same value of Atropos consensus time, then the event block with a smaller timestamp is determined to the forward position. For rapidly finding the Atropos consensus time, each node uses the Main-chain. Figure 7 shows the example for consensus time ordering.

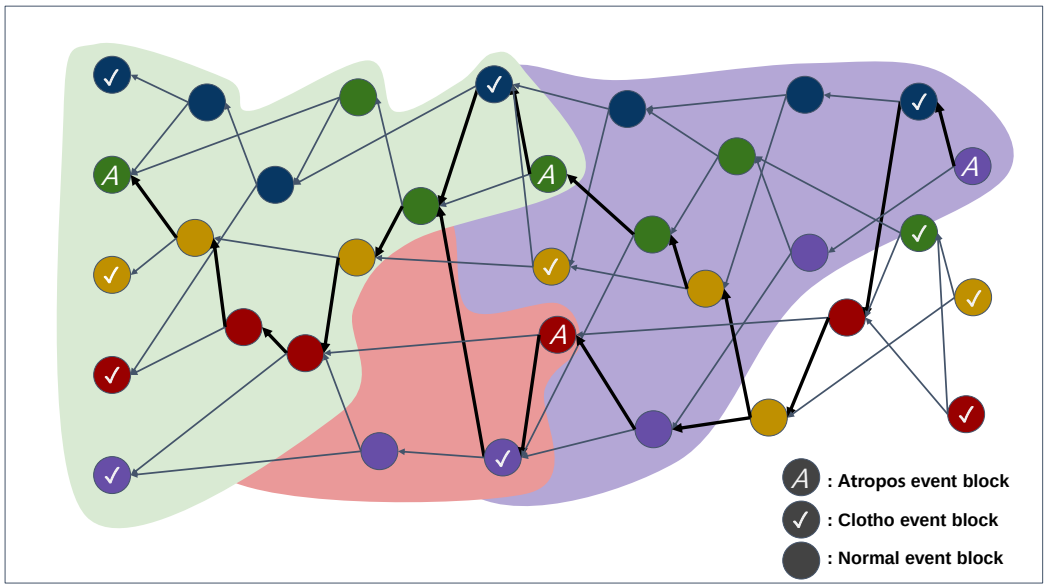

Figure 7: An Example of Time Ordering

4. Proof of Lachesis Consensus Algorithm

This section provides a number of useful definitions, followed by several proofs, building up from the Lemma (Lemma 1) to the Byzantine Fault

Tolerance Theorem (theorem 3). In the proofs it is assumed that there are n nodes (n > 1), more than 2n/3 of which are reliable node. It is also assumed that the digital signatures and cryptographic hashes are secure, so signatures cannot be forged, signed messages cannot be changed without detection, and hash collisions can never be found.

Definition 4.1 (node)**.** *n is the number of participated nodes.*

Definition 4.2 (OPERA Chain)**.** *$OPERA(A)$ is a directed ayclic graph (DAG) that belongs to node A.*

Definition 4.3 (event block set)**.** *EB is a set of all event blocks in OPERA chain. EB_i is a set of all event blocks created by node i. n is the number of participated nodes.*

Definition 4.4 (signature)**.** *$sig(x)$ is a signature of the node (node ID) that creating an event block x.*

Definition 4.5 (path)**.** *$P(h, k)$ is a set of event blocks from event block h to k including event blocks h and k. if there is no path between event block h and k, $P(h, k) = \emptyset$.*

Definition 4.6 (parent)**.** *If an event block x have has hash value of another event block y, then y is parent of x. If $sig(x) = sig(y)$, an event block y is self-parent of event block x and $sig(x) \neq sig(y)$, an event block y is other-parent of event block x.*

Definition 4.7 (ancestor)**.** *An event block y is ancestor of an event block x if $P(x, y) \ni k$, where $\exists k \ni EB$ and $k \neq x$ and $k \neq y$. If $sig(x) = sig(y)$, an event block y is self-ancestor of event block x and $sig(x) \neq sig(y)$, an event block y is other-ancestor of event block x.*

Definition 4.8 (genesis)**.** *The initial event block created by a node is called genesis. The event block as genesis has no parents.*

Definition 4.9 (reliable node)**.** *In DPoS approach, an elected node among all members can include trusted nodes and attackers. However, a node that assumes no malicious behavior between these nodes as a reliable node.*

Definition 4.10 (fork)**.** *The pair of event blocks (x, y) is a fork, if $sig(x) = sig(y)$, $P(x, y) = \emptyset$ and $P(y, x) = \emptyset$.*

Definition 4.11 (create event block)**.** *A reliable node tries to create continuingly event blocks with every other node (with hashes of the latest self-parent and other-parent), share the information with every other before creating event block.*

Definition 4.12 (share)**.** *A node A share event block y of a node B, if* $P(a, y) \neq \emptyset$ *,where* $EB_A \ni a, EB_B \ni y$*.*

Definition 4.13 (supra-share)**.** *An event block x can supra-share an event block y if* $P(x, y) \supset S$*, where S is the set of nodes such that* $|S|$ *is more than 2/3 nodes and for* $S \ni \forall k$*,* $P(k, y) \neq \emptyset$*.*

Definition 4.14 (Clotho)**.** *In a generation* $g + 1$*, any event blocks satisfy supra-share for the set of Clotho in g, where they can satisfy more than 2n/3 nodes.*

Definition 4.15 (flag table)**.** *For a Clotho c, the flag table* $V = (v_1, v_2, \ldots, v_n)$ *is a vector in* $\mathbb{Z}^*$ *and* v_i *indicates the number of nodes which appear on* $P(c, y)$*, where y is a one of previous Clotho.*

Definition 4.16 (generation)**.** *The generation created number of a Clotho x is defined to be* $g+i$*, where g is the highest generation number of the ancestor Clotho of x (or 1 if it has no parents), and i is defined to be 1 if x can know more than 2n/3 Clotho in g (or 0 if it cannot).*

Definition 4.17 (Atropos)**.** *When a Clotho in generation* $g + 2$ *is created, a Clotho of node i in g can be confirmed as Atropos if there is a set C of Clotho in* $g+1$ *such that* $|C| > 2n/3$ *and for clotho in C ,* v_i *of its flag table is larger than* $2n/3$*.*

Definition 4.18 (consensus position)**.** *An order between all event blocks by utilizing the determined Atropos group and the assigned consensus time of each Atropos is called consensus position.*

Definition 4.19 (consistency)**.** *OPERA chain in node A and B are consistent if for two event block x and y included in both* $OPERA(A)$ *and* $OPERA(B)$*,* $P(x, y)$ *in A is equal to* $P(x, y)$ *in B.*

Lemma 4.1 (fork lemma)**.** *If the pair of event blocks* (x, y) *is fork, then the OPERA chain can structurally detect the fork.*

Proof. suppose that a node creates two event blocks (x, y) and the event blocks is fork. By the Definition 4.10, reliable node tries to create continuously event blocks with every other node. Thus, two reliable nodes A and B be able to have an event block x and y respectively. When the nodes create an event block with each other, the nodes have the pair of event blocks (x, y). Thus, the nodes detect that the pair of event blocks (x, y) is fork. □

Lemma 4.2. *If two nodes are selected, they have consistent OPERA chain.*

Proof. If node A and B has no consistent OPERA chain, for two event blocks x and y included in both $OPERA(A)$ and $OPERA(B)$, $P(x, y)$ in $OPERA(A)$ is not equal to $P(x, y)$ in $OPERA(B)$. In OPERA chain, node A communicates with node B and synchronize both $OPERA(A)$ and $OPERA(B)$ to create an event block. Thus, for two event blocks included in both $OPERA(A)$ and $OPERA(B)$, their path in $OPERA(A)$ should be equal to that in $OPERA(B)$. Hence, in order to have different path between two event blocks in $OPERA(A)$ and $OPERA(B)$, one of them is an attacker and it forks an event block. Based on fork lemma, if an attacker forks an event block, OPERA chain detects and rectifies it before new Clotho is generated. Therefore, two nodes have consistent OPERA chain. □

Theorem 4.3 (Atropos confirmation)**.** *Each Clotho event block x is confirmed as Atropos using flag tables in posterior Clotho event blocks.*

Proof. By Lemma 4.2 and 4.1, all nodes have consistent OPERA chain and the OPERA chain can detect fork, respectively. All nodes that have consistent OPERA chain also have consistent Clotho. Each node has same results computed through flag table because each node can reach consistent flag tables. Thus, an event block x in Clotho at g can be confirmed by other posterior Clotho at $g + 1$, and all nodes can share the confirmation. □

Lemma 4.4. *If $OPERA(A)$ and (B) are consistent, and the time consensus algorithm running on A shows that an event block in generation g created by node a_1 sends a value v_A to other node b_1 in $g+1$, and the algorithm running on B shows that an event block in g by member a_1 sends a value v_B to an event block by node b_1 in $g + 1$, then $v_A = v_B$.*

Proof. The time consensus algorithm only sends a value v from event block x to event block y if y can supra-share x. It is not possible for consistent OPERA chain to have two events that are forks, by Fork lemma. Therefore,

the two value (v_A, v_B) must be coming from the same event block x in both $OPERA(A)$ and $OPERA(B)$. the value is calculated purely as OPERA chain structure, so the two OPERA chain must have on the value, and $v_A = v_B$. □

Theorem 4.5 (multi-value consensus)**.** *For any Multi-value question, Lachesis consensus algorithm guarantees to reach consensus.*

Proof. For any generation g, time consensus algorithm checks whether there is a value sharing supra-majority of all nodes have (2/3 of all nodes). If the condition is satisfied, multi-value byzantine consensus problem reaches an agreement. However, each node selects one of values collecting previous generation by the time consensus algorithm and sends the selected value to next generation. Through the reselection behavior in time consensus algorithm, multi-value byzantine agreement problem can be changed to 2-value byzantine agreement problem as the OPERA chain grows. The 2-value byzantine consensus problem cannot reach an agreement with low probability. For the reason, time consensus algorithm includes minimal value selection round per next h generation. In minimal value selection algorithm, each node selects minimum value among values which the nodes have. Thus, multi-value byzantine consensus problem eventually reaches an agreement. □

Lemma 4.6. *If $OPERA(A)$ and $OPERA(B)$ are consistent, and A decides a value v for multi-value byzantine agreement election in generation g and B has not decided prior to g, then B will decide same value in $g + 2$ or earlier.*

Proof. In multi-value byzantine consensus problem, $OPERA(A)$ decides a value v, which means that some Clotho in generation g received value v from a set of nodes S that contains more than 2n/3 nodes. Because value selection is consistent (by lemma 4.4), all other generation g event blocks in A and B will receive values from more than 2n/3 nodes, a majority of whom will also be in S, because two subsets of size greater than 2n/3 drawn from a set of size n must each have a majority of their elements in common with the other. Therefore, every generation g Clotho in both A and B will have value v. every event blocks in A and B in that generation will receive unanimous value v and will decide v for multi-value byzantine agreement election in generation $g + 1$. However, if $g + 1$ is minimal value selection, the consensus time cannot be confirmed and is confirmed in $g + 2$. □

Lemma 4.7. *For any generation g, if two Atropos a_1 and a_2 is confirmed respectively, nodes know the two Atropos and can assign consensus time to them.*

Proof. By the Clotho definition, each Clotho supra-shares more than 2n/3 previous Clotho. For the reason, when two of Clotho in $g+1$ are selected, they must supra-share same Clotho which are more than one-thirds of Clotho in g. This means that more than n/3 Clotho in $g+1$ share both Atropos a_1 and a_2 if the nodes that confirm Atropos a_1 and a_2 are reliable nodes. With the Clotho definition and previous explanation, all Clotho in $g+2$ share both Atropos a_1 and a_2. Thus, nodes know the two Atropos at $g+2$ or earlier. When the two Atropos have consensus time, each nodes can assign consensus position to the Atropos. □

Theorem 4.8 (Byzantine fault tolerance)**.** *Each event block x created by a reliable node will eventually be assigned a consensus position in the total order of events.*

Proof. All reliable nodes will eventually learn of event block x created by a reliable node, by the definition of reliable node. Therefore, there will be a generation where an Atropos is descendant of x. Then x is assigned a consensus group of the Atropos, and a consensus time which the Atropos is confirmed, and its consensus group will be fixed. Furthermore, it is not possible to later discover a new event block y that will come before x in the consensus order. Because, to come earlier in the consensus history, the y would have to be included in that consensus group by the Atropos or earlier consensus group. That would mean that the Atropos shares y. But once the consensus group is created, there is no way to discover new ancestors for them in the future. Furthermore, it isn't possible for a generation to gain new Atropos in the future. Any new Clotho in g that is discovered in the future will not be an ancestor of the known Clotho (of which there are more than 2n/3) in $g+1$. Therefore, once an event block is assigned a place in the total order, it will never change its position, neither by swapping with another known event block, nor by new event blocks being discovered later and being inserted before it. □

5. Response to Attacks

Like all other decentralized blockchain technologies, OPERA chain will likely be subject to attacks by attackers which aim to gain financial profit to

damage the system. Here we describe several possible attack scenarios and how the OPERA chain intends to take preventive measures.

5.1. Sybil attack

Generally, blockchains rely on consensus mechanism such as Proof-of-Work (PoW) to limit their vulnerability to Sybil attacks. In this situation, attackers can assume multiple identities, so the number of faulty processes appears to be larger than it is. If the number of attacker is less than 1/3 nodes, OPERA chain is kept up whatever the attacker do. Since OPERA chain can operate when two-thirds of all nodes do reliably. In addition, as the node operation method of OPERA chain will use a method similar to Delegated Proof of Stake (DPoS), the outcome through the voting system will be intended to accurately identify attackers. No more participation can be made except for the node initially involved by voting.

5.2. Parasite chain attack

In a DAG-based protocol, a Parasite chain attack can be made with a malicious purpose, attempting connection by making it look like a legitimate event block, and it can cause Double-spending problems. In order to defend this method, our event block confirmation cannot structurally make a fork. In addition, event blocks in parasite chain cannot be assigned since event blocks is shared in more than 2/3 of all nodes to be assigned.

6. Conclusion

In order to realize the distributed ledger technology, a new asynchronous DAG-based consensus algorithm was proposed. Its name is a Lachesis protocol and Lachesis consensus algorithm (LCA) of OPERA chain. To solve the Byzantine fault tolerant problem, the elected nodes form the network by applying a DPoS method. To ensure the distribution of participating nodes, the Lachesis protocol used the cost function. And, for efficient and quick selection process, Clotho and Atropos are elected based on the flag table. In this process, all paths need not be calculated every moment. The LCA can be calculated based on the flag table for selecting Clotho and Atropos. In terms of time complexity, the existing path calculation results in an $O(n^3)$. However, the LCA performs effective calculation for consensus based on flag table. It can also defend to any double-spending problem based on the main

chain by consisting of Atropos. Finally, the time ordering ensures guarantee by converting multi-value to 2-value consensus problem. Based on these properties, the LCA provides a fair, transparent, and effective consensus algorithm.

[1] L. Baird. Hashgraph consensus: fair, fast, byzantine fault tolerance. Technical report, 2016.

[2] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association.

[3] A. Churyumov. Byteball: A decentralized system for storage and transfer of value, 2016.

[4] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[5] D. Larimer. Delegated proof-of-stake (dpos), 2014.

[6] C. LeMahieu. Raiblocks: A feeless distributed cryptocurrency network, 2017.

[7] S. D. Lerner. Dagcoin, 2015.

[8] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[9] S. Popov. The tangle, 2017.

[10] S. N. Sunny King. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake, 2012.

[11] M. Swan. *Blockchain: Blueprint for a new economy.* O'Reilly Media, 2015.