OPERA Chain: A DAG-based Lachesis Consensus Algorithm -Version 1.0-

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 analyze practical Byzantine fault tolerance (pBFT) and explain combining between consensus protocol and chain in pBFT by utilizing DAG structures with reachability. We propose Lachesis protocol to generate DAG structures using a cost function that can identify lazy participants 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 block as consensus. It indicates that Lachesis consensus algorithm can implement Byzantine fault tolerance. Based on Lachesis protocol and consensus algorithm, we can expect that DAG structures combine consensus protocol and chain in pBFT. In our algorithm, we design a flag table that has connection information of blocks in specific parts and check share information of some blocks. We can expect reduction of consensus steps than pBFT protocol for consensus by addressing the flag table since it can provide connection information of specific blocks. In this paper, we also provide proof of our consensus algorithm for safety and liveness of OPERA chain.

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

Contents

1. Introduction

With the appearance of Blockchain, many industrial fields expect commercialization of distributed ledger technology [18]. But delay of processing time for consensus and high commission obstruct the commercialization. The delay of processing time is concerned with the consensus algorithms [4, 7]. 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 [10]. It is equivalence to the proof of Byzantine fault tolerance in distributed database systems [1, 9]. Proof of Work (PoW) Blockchain requires lots of computation works to generate the blocks by participants [15]. 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) [17], Delegated Proof of Stake (DPoS) [11], and the algorithms using directed acyclic graph (DAG) [13]. 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 [16], Byteball [5], and Hashgraph [2]. 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.

We first explain possibility for efficient implementation of practical byzantine fault tolerance (pBFT) using DAG concept through analysis of pBFT and DAG structure [3]. In pBFT, consensus is reached through the protocol that a created block is shared with other participants and the share infor-

mation is shared with others again [8, 14]. After reaching consensus, the block is added to the participants' chains [3]. We propose OPERA chain that combined consensus protocol and chain in pBFT using DAG structures.

We also propose Lachesis protocol that can identify lazy participants and consensus algorithm on OPERA chain. We 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. Our consensus algorithm detects reachability between blocks using the flag table.

Distributed system in completely asynchronous communication cannot be reach Byzantine agreement since there exists unbounded message delays [6]. The Lachesis consensus algorithm differs from these systems that reach consensus in creating blocks. Lachesis protocol generates each block asynchronously and Lachesis algorithm makes a consensus by checking how many nodes know the blocks. Thus, OPERA chain can reach consensus.

We first describe motivation 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. In Chapter 6, previous approaches of DAG are explained. Finally, we conclude this paper along with the future appending works in Chapter 7. And, reference is introduced by Chapter 8.

1.1. Motivation

The Practical Byzantine Fault Tolerant (pBFT) is a protocol that allows all nodes to successfully reach an agreement for a block (information) when a Byzantine node exists [3]. Figure 1 illustrates the consensus method based on pBFT for adding blocks on the block chain. In pBFT, we need four processes to add blocks created by a node to the chain of all other nodes as follows,

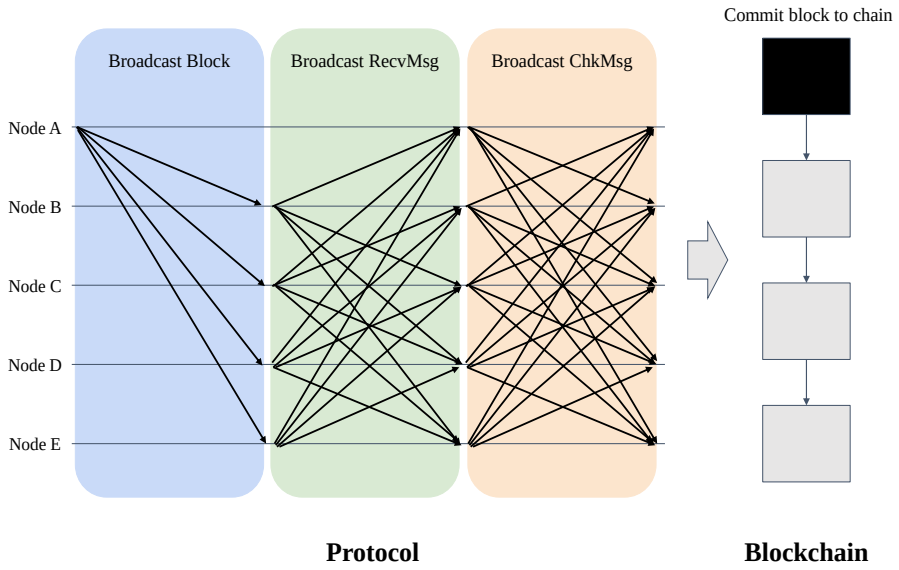

Figure 1: Consensus and Block Creation Process in pBFT

1. $BroadcastBlock$: The node that created the block broadcasts the block to other nodes.
2. $BroadcastRecvMsg$: Nodes that receive the block broadcast the message that the block has been received to the other nodes.
3. $BroadcastChkMsg$: If RecvMsg is received from other nodes with a quorum, the nodes that received RecvMsg verify the block for safety such as double spending problems. When the block is verified, it broadcasts a verification message (ChkMsg) to the other nodes.
4. $CommitBlocktoChain$: If the node that creates the block has received the ChkMsg from other nodes with a quorum, add the block to own chain. This process ensure that all nodes successfully reach an agreement.

We can integrate consensus protocol and chain in pBFT using DAG structure with path search. Whenever a new block occurs in a DAG, we check the node creating blocks in path from the new block to the blocks. And if the number of signatures of the blocks in the path exceeds the quorum covered by the pBFT, it means that the blocks accessible from path were shared among the participants who had exceeded the quorum. The check for the number of block signatures in path is equivalent to $BroadcastRecvMsg$ and

$BroadcastChkMsg$ in pBFT. pBFT adds the blocks to the chain through $BroadcastChkMsg$, however path search in DAG is a data structure in which the DAG itself is a chain and can check the consensus at the same time. Thus, consensus can be reached through a method similar to the broadcast process in pBFT [14].

We first need to design a protocol to build a DAG to utilize the path search method. Also, a block search algorithm is required to authenticate $BroadcastRecvMsg$ and $BroadcastChkMsg$ processes in pBFT. Based on these two conditions, we can design a method that pBFT consensus process can be created in DAG creating blocks and allowing other blocks to reach consensus.

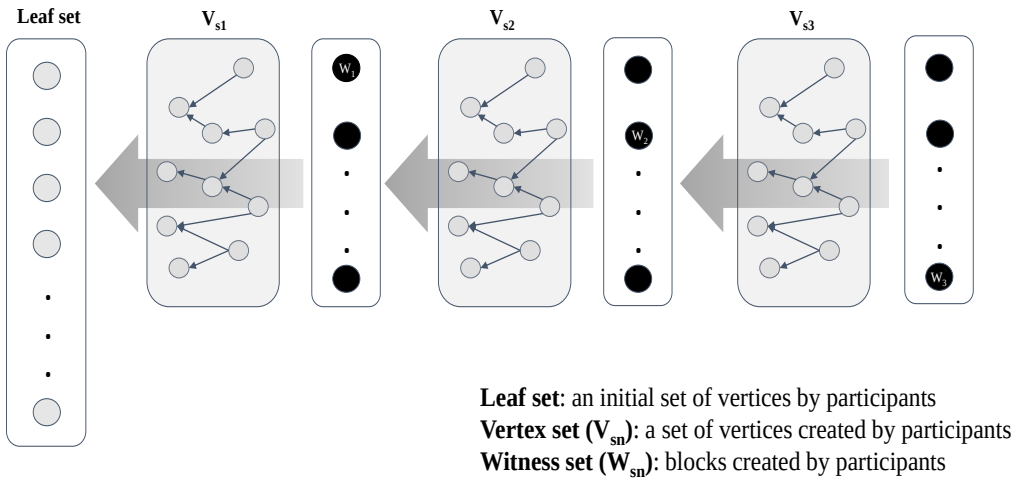

Figure 2: Consensus Method through Path Search in a DAG (combines chain with consensus process of pBFT)

Figure 2 shows a method for reaching consensus through the path search in the DAG described above. In Figure 2 $leafset$ represents a block initially created by network participants. V and $witnessset$ are a set of blocks connected to $leafset$. Thus, either V or one block of $witnessset$ has a path that can reach to any one block of $leafset$. w_1, which is part of the first $witnessset$, is the block where a quorum or more blocks exist on a path that reaches a block of $leafset$. This means that the blocks in $leafset$ are shared to the quorum participants and there are participants receiving $BroadcastRecvMsg$ sent from the quorum participants. Therefore, w_1

is the block that satisfies $Broadcastblock$ and $BroadcastRecvMsg$ processes at pBFT. In Figure 2, w_1 reaches $leafset$ through V_1 that the set of blocks in all paths from w_1 to a block of $leafset$. In this situation, if V_1 includes blocks created by more than a quorum of participants, w_1 includes $witnessset$. Likewise, w_2 is a block that can be reached for $witnessset$ including w_1 through blocks made by a quorum of participants.

We can recognize that blocks of leaf sets that could be reached by w_1 are shared with more than quorum participants through the presence of w_1 ($Broadcastblock$ and $BroadcastRecvMsg$ processes in pBFT). The existence of the w_2 shows that information of w_1 is shared with more than a quorum ($BroadcastChkMsg$ process). This is the same effect as the pBFT consensus processes and a path search in the DAG structure allows the chain to reach consensus. Therefore, it is important to identify the blocks that can satisfy the pBFT consensus process for path search method.

We suggest Lachesis protocol to construct DAG structures. In addition, for consensus in DAG, we also propose Lachesis consensus algorithm that can authenticate $BroadcastRecvMsg$ and $BroadcastChkMsg$ processes in pBFT.

1.2. Main Concepts

- **Event block** All nodes can create event blocks as time t. The structure of an event block includes the signature, generation time, transaction history, weight, 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.

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

- **Witness set** Witness set (W_s) is a set of witness and the cardinality of the set is $2n/3 < W_s \leq n$, where n is the number of all nodes.

- **Flag table** Flag table stores reachability from an event block to other Witness. Therefore, the sum of all reachabilities, namely all values in flag table, indicates the number of reacheabilities from an event block to other Witness.

- **Clotho** Clotho is some of Witness which satisfy that they are known in more than $2n/3$ nodes and more than $2n/3$ nodes know the information that they are known in nodes. Clotho 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 Clotho, based on the collected consensus time in parent witness set. When the consensus time reaches byzantine agreement, Clotho is confirmed as Atropos and used for time consensus ordering.

- **Weight** The first created event blocks have 1 as weight. And the weight of an event block excluding the first created event blocks is the sum of weights of parent event blocks. The weight used to determine the time order when the overlap of consensus time and the internal consensus time of each Atropos

- **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 3 shows an example of OPERA chain constructed through Lachesis protocol.

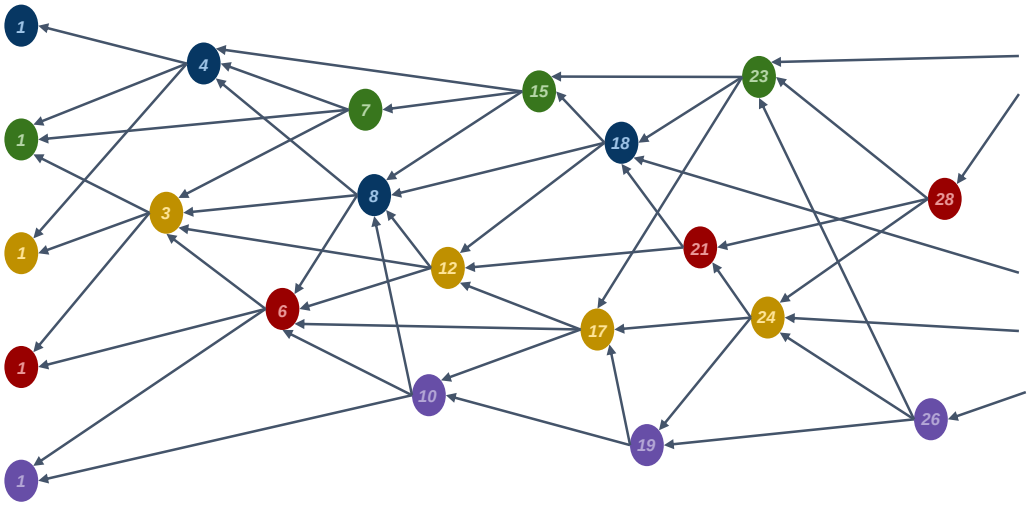

Figure 3: An Example of OPERA Chain

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 Witness, 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 3, a node runs Node selection algorithm. Node selection algorithm returns ID of another node to communicate with it. In line 4 and 5, the node requests synchronization and synchronizes OPERA chain. Line 6 runs Event block creation. In Event block creations, the node creates an event block and checks whether it is Witness. In line 7 and 8, Clotho selection and Atropos time consensus operate. They are for checking whether Witness can be

Clotho and assigning the consensus time and confirm as Atropos. The other part is replying synchronization of OPERA chain. In line 10 and 11, 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, B = Node selection algorithm()
4:      request sync to node A and B
5:      sync all known events by Lachesis protocol
6:      Event block creation
7:      Clotho 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 4 shows example of node structure component.

Each node has signature stamp, height vector, in-degree vector, flag table, Witness 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. The flag table is a n dimentional vector. If an event block e created by i^{th} node can reach j^{th} witness, then the j^{th} value in the flag table of e becomes 1 (otherwise 0). Each node only maintains the flag table of the top event block. Figure 4 is an example of component in node A. In figure 4, 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 4, 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. Witness hash list is data structure storing hash values of

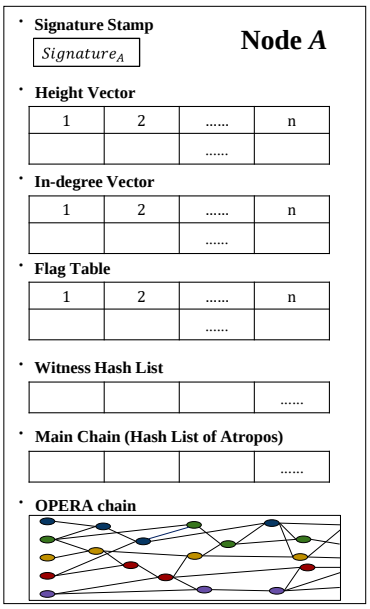

Figure 4: An Example of Node Structure

Witness. Main-chain is data structure storing hash values of Atropos. Main-chain is used to find out event blocks with complete consensus. The Witness, Clotho and Atropos selection algorithm are introduced in chapter 3.

2.2. Lachesis Protocol

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

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

We define Cost Function (C_F) for preventing the creation of lazy nodes. The lazy node is a node that has lower work portion in OPERA chain than others. When a node creates an event block, the node selects other nodes with low values by cost function and refers to the top event blocks of the reference nodes. 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 Witness 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.

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
```

After the reference node is selected, each node communicates and shares 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

event block. The communication between nodes does not allow simultaneous communication with the same node.

3. Lachesis Consensus Algorithm

Lachesis Consensus Algorithm (LCA) is a process that examines whether event block is shared with 2n/3 nodes, where n is the numer of all nodes. Sharing an event block with 2n/3 nodes means that more than two-thirds of all nodes in OPERA chain knows the event block. Some event blocks are called Witness if the event blocks are linked to more than two-thirds previous Witness. By introducing Witness event blocks, we are able to track blocks that $2n/3$ of the network agree on. Instead of searching the path from an event block to other witness, based on the calcultion of the flag table, we can find reachability from an event block to other Witness. The Atropos blocks can then be used to form a chain, which we call the "Main-Chain". As we have used Atropos blocks which are derived from Witness blocks we can be sure that this chain of blocks is agreed upon by $2n/3$ of the network, thus we can be sure that at least $2n/3$ of nodes have come to consensus on this "Main-Chain". The main chain will be detailed further in the following section.

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 7 Atropos event blocks. The following sections describe the Witness selection algorithm, Clotho selection algorithm and Atropos consensus time selection algorithms required to create the Main-chain.

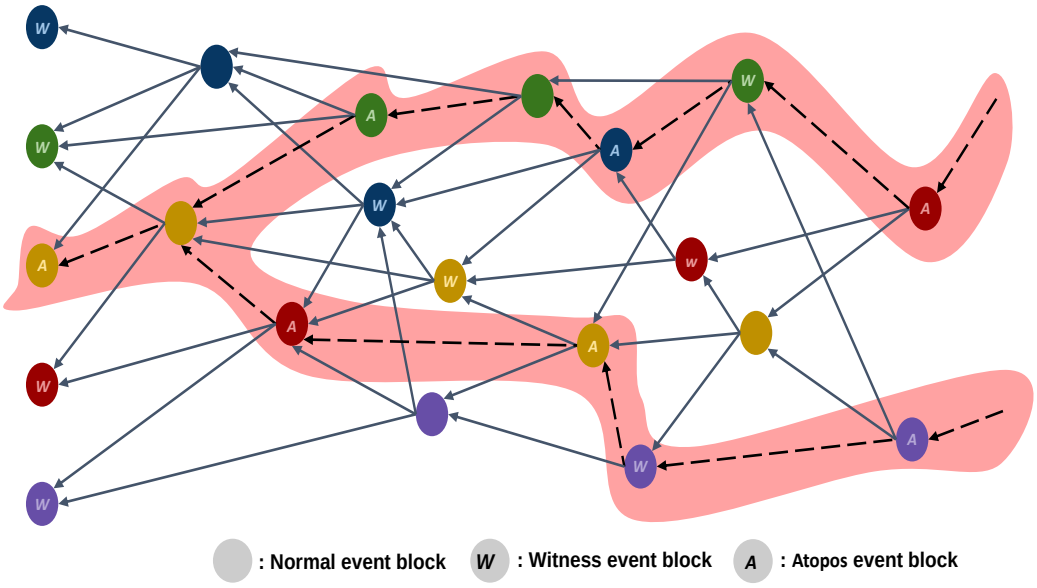

Figure 5: An Example of Main-chain

3.2. Witness Selection

All nodes can create event blocks and an event block can be Witness by satisfying some conditions. Not all event blocks can be Witness. We first regard the first created event blocks as Witness. The first created event blocks are composed of the first witness set W_{S1}. If there are total n nodes and these nodes create the event blocks, then the cardinality of the first witness set is n since all first event blocks are Witness. After that, if there is an event block e can reach 2n/3 witness, then e become Witness and belong to not W_{S1}, but other witness set. Because of this reason, excluding the first witness set, the range of cardinality of witness set W_{Sk} is $2n/3 < |W_{Sk}| \leq n$. We can check whether the new event block becomes Witness or not through flag table. Each node maintains flag table of the top event block. If the new event block is created, the event block has three hashs for previous three event blocks. We apply OR operation to the flag tables of previous event blocks. Figure 6 shows the example of calculation for flag table. In Figure 6 the event block W_1 is the recent created event block. We apply OR operation to the flag tables of three connected event blocks. The results of OR is the flag table of W_1 and we check whether the sum of row is more than 2n/3 or not. In this example, the sum of row is 4 and this value is more than

2n/3 (n=5). Thus, W_1 becomes Witness. The process of Witness selection algorithm is as follows.

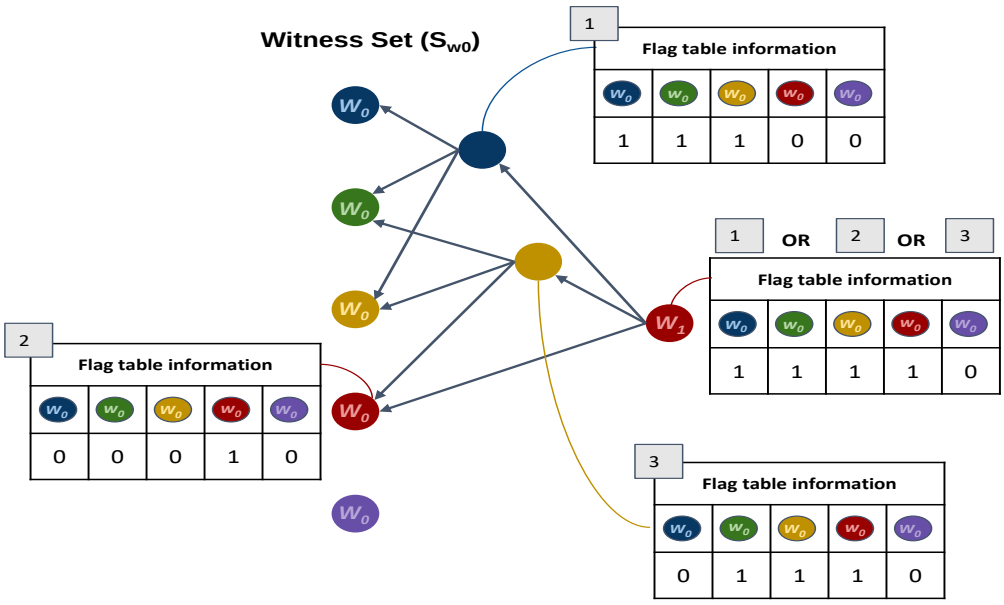

Figure 6: An Example of Flag Table Calculation

1. The first event blocks are considered as Witness.
2. When new event block is added in OPERA chain, we check whether the event block can be Witness using OR operation to the flag tables connected to new event block. If the sum of the flag table for the new event block is more than 2n/3, the new event block becomes Witness.
3. When new Witness appear on OPERA chain, some nodes update Witness hash list. If one of new event blocks becomes Witness, all nodes that share the new event block and add the hash value of the event block to their Witness hash list.
4. The new witness set is created if the cardinality of previous witness set W_{Sp} is more than 2n/3 and the new event block can reach 2n/3 witness in W_{Sp}.

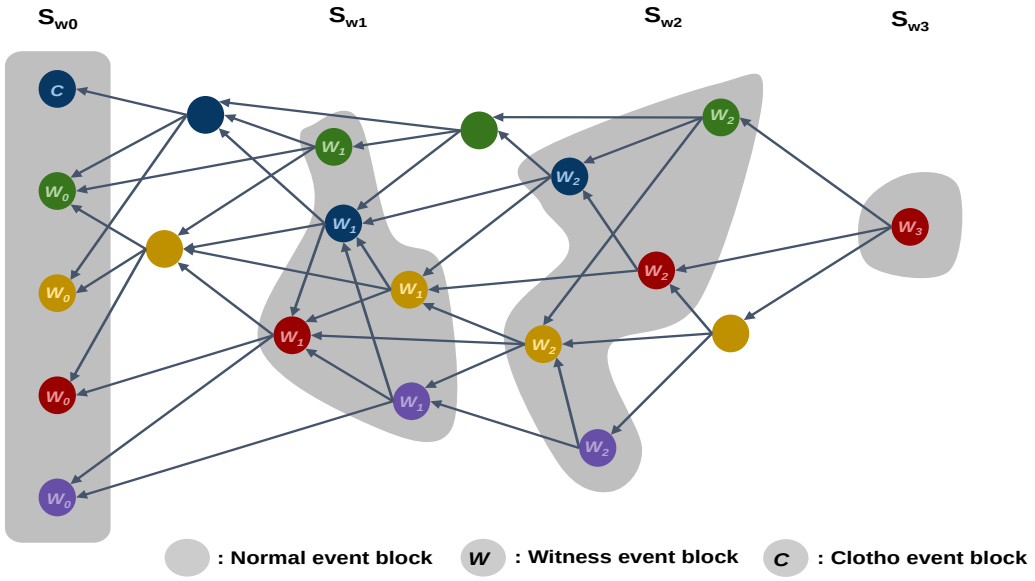

Figure 7: An Example of Clotho

3.3. Clotho Selection

Clotho selection algorithm checks whether Witness event blocks in the Witness hash list satisfy the Clotho condition. Clotho condition is that more than 2n/3 nodes know the Witness and an witness know the information. Clotho selection algorithm operates when an event block is nominated as Witness. If an Witness satisfies Clotho condition, the Witness become Clotho and make a candidate time for Atropos. After the Witness is nominated for Clotho, Atropos consensus time selection algorithm operates. Algorithm 3 shows the pseudo code of Clotho selection. The algorithm operates when each node creates Witness. Line 4 and 5 set $c.Clotho$ and $c.yes$ to *undecided* and 0 respectively. line 6-8 check whether Witness c' in 2^{nd} Witness set of W share c. In line 9-10, checks whether the number of Witness in 2^{nd} Witness set of W which share c is more than n/3 and declares the Witness c' to Clotho. The time complexity of Algorithm 3 is $O(n^2)$, where n is the number of nodes.

Figure 7 shows an example of Clotho. In Figure 7, a circle , a circle with w_i mark, and a circle with c mark indicates an event block, Witness, and Clotho respectively. If there are three other sets of Witness and there exists one Witness after recent clotho set, then one of Witness in the first Witness

Algorithm 3 Clotho Selection

```
1: procedure CLOTHO SELECTION
2:     if an Witness w in Witness_set W is created then
3:         for c ∈ 3^th parent Witness_Set of W do
4:             c.Clotho ← undecided
5:             c.yes ← 0
6:             for c' ∈ 2nd parent Witness_Set of W do
7:                 if c' share c then
8:                     c.yes += 1
9:             if c.yes > n/3 then
10:                c.Clotho ← yes
11:                c.time ← w.time_stamp
```

set become Clotho. Figure 7 shows this process in OPERA chain.

3.4. Atropos Consensus Time Selection

Atropos consensus time selection algorithm is the process in which the candidate time generated from Clotho selection is shared with more than two-thirds of all nodes and results in a consensus. After the Clotho is nominated, nodes check the candidate time of each node for the Clotho 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 Clotho as OPERA chain grows. The candidate time reaching the consensus is called Atropos consensus time. After deciding Atropos consensus time, Clotho 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.Clotho in ith witness_set W_Si
3:     c.consensus_time ← undecided
4:     for c' ∈ jth Witness_Set W_Sj do
5:         d ← j-i
6:         if d > 1 then
7:             if d is 2 then
8:                 c'.time ← c'.time_stamp
9:         else if d > 2 then
10:            s ← the set of Witness in (j − 1)th witness_set W_Sj−1 that c' can share
11:            t ← RESELECTION(s)
12:            k ← the number of Witness 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 in s
```

Algorithm 4 and 5 show pseudo code of Atropos consensus time selection and Consensus time reselection. In Algorithm 4, In line 5, d saves the deference of relationship between witness set of c and c'. Thus, line 7 means that c' is in $2th$ child witness set of c. Line 8, each Witness in W_{Sj} selects own timestamp as candidate time of c when they confirm Witness c as Cltoho. In line 10, 11, and 12, s, t, and k save the set of Witness that c' can share c, the result of $RESELECTION$ function, and the number of Witness in s having t. Line 14-18 is checking whether more than two-thirds of Witness in W_{Sj-1} select same candidate time. If two-thirds of Witness in W_{Sj-1} select same candidate time, the Witness c is assigned consensus time as t. Line 20 is minimum selection round. In minimum 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 Witness and output is reselected candidate time. Line 4-10 is classifying process how

many each candidate times is selected in the set of Witness S. In line 11-17, a candidate time which is the most selected and the smallest is selected. 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 Witness S
3:      Output: candidate time t
4:      D is dictionary data structure
5:      D.initialize()
6:      for Witness 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 of all nodes grows up into same shape. This allows each node to know the candidate time of other nodes based on its OPERA chain and reach a consensus agreement. The proof of that the agreement based on OPERA chain become agreement in action is shown in the section 4.

A consensus time order between event blocks is decided by Atropos consensus time selection, weight that event block has, and timestamp of event

block. A weight means the sum of parent event blocks. And, it used to order event blocks after determining the Atropos. In a LCA, the consensus order decision takes place in two cases. The first, the sequence of Atropos having the same consensus time. Second, the order of event blocks within the Atropos.

a. The sequence of Atropos having the same consensus time

The consensus order of the Atropos is used to determine that the sequence of Atropos have same moment in asynchronous networks. At that time, the priority of two Atropos can be determined based on the weight. Figure 8 shows an example determining the sequence of consensus between Atropos A_1 and A_2

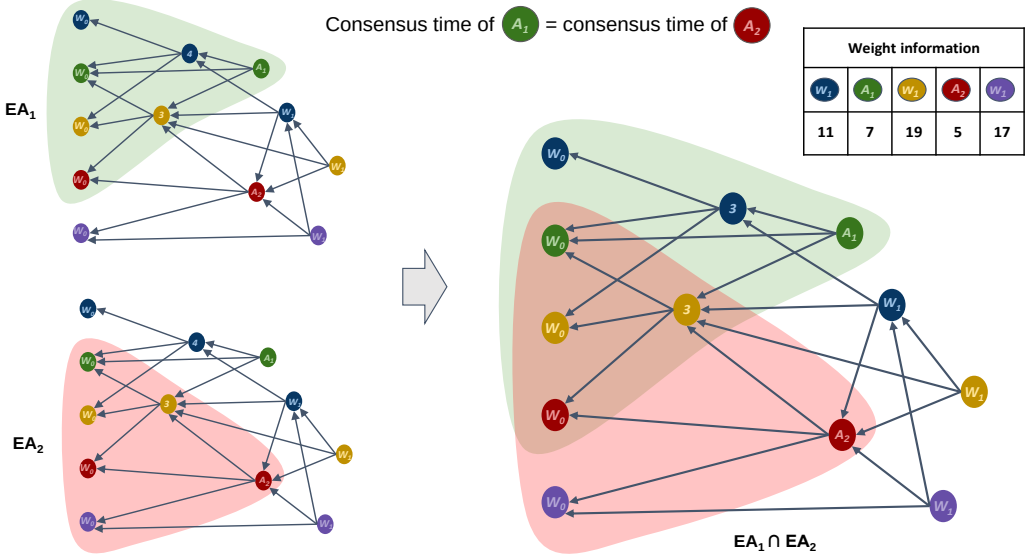

Figure 8: An Example of determining the sequence of consensus

In Figure 8, A_1 and A_2 present that the event blocks are determined as Atropos. It means that they will have each consensus time respectively. When the two event blocks have same consensus time, the priorities of both event blocks should be determined. This is not only meaningful in determining order of between A_1 and A_2, but the event blocks included path of any witness (EA) that can be reached from the Atropos. At the same time, it

can determine which event blocks belong to intersection of EA of any Atropos. In figure 8, EA_1 presents a collection of event blocks belonging to all path reachable from Atropos A_1 to Witness. And, EA_2 is a set of event blocks that can be obtained from Atropos A_2. At this moment, there may be areas of intersection $(EA_1 \cap EA_2)$ that have similar times. In a LCA, a weight is used to prioritize A_1 and A_2, then the relevant event blocks can belong to determined Atropos areas. In Figure 8, A_1 and A_2 have weights 7 and 4, respectively. Here, Atropos with a small weight is determined by a priority. The smaller weight, the earlier created the event blocks will be. It means that the event block that has the smaller weight than others early satisfies witness conditions. In Figure 8, since the weight of A_2 is smaller than A_1, the consensus order of A_2 with high priority is determined. Thus, the intersection areas $(EA_1 \cap EA_2)$ can be belong to area of A_2.

b. The order of event blocks within the Atropos

The order of event blocks in the Atropos means the determination of consensus order in the event blocks that any Atropos can reach. Figure 9 shows an example of sequencing of event blocks within the Atropos.

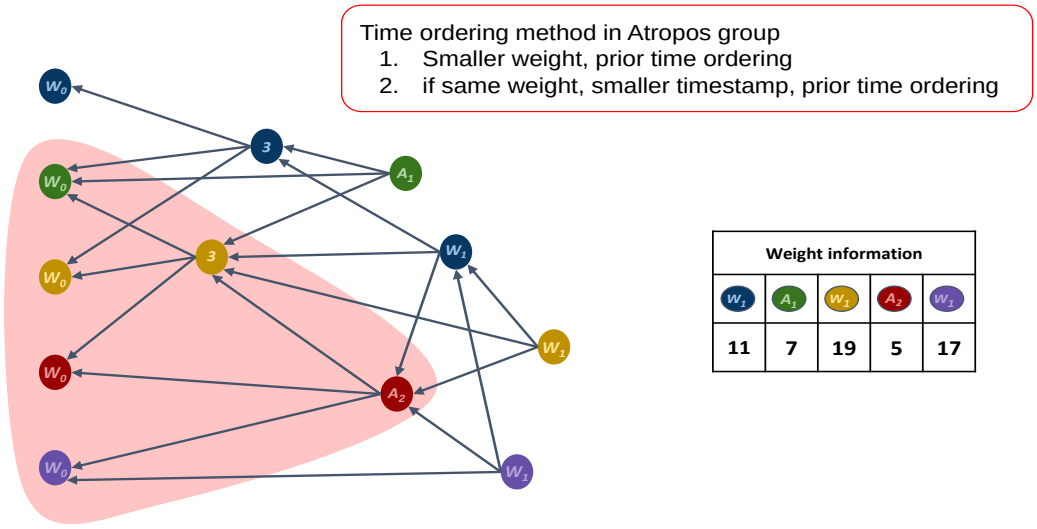

Figure 9: An Example of ordering of event blocks within the Atropos

Figure 10 shows the part of OPERA chain in which the final consensus order is determined based on weight. The number of represented by each

event block is a weight value.

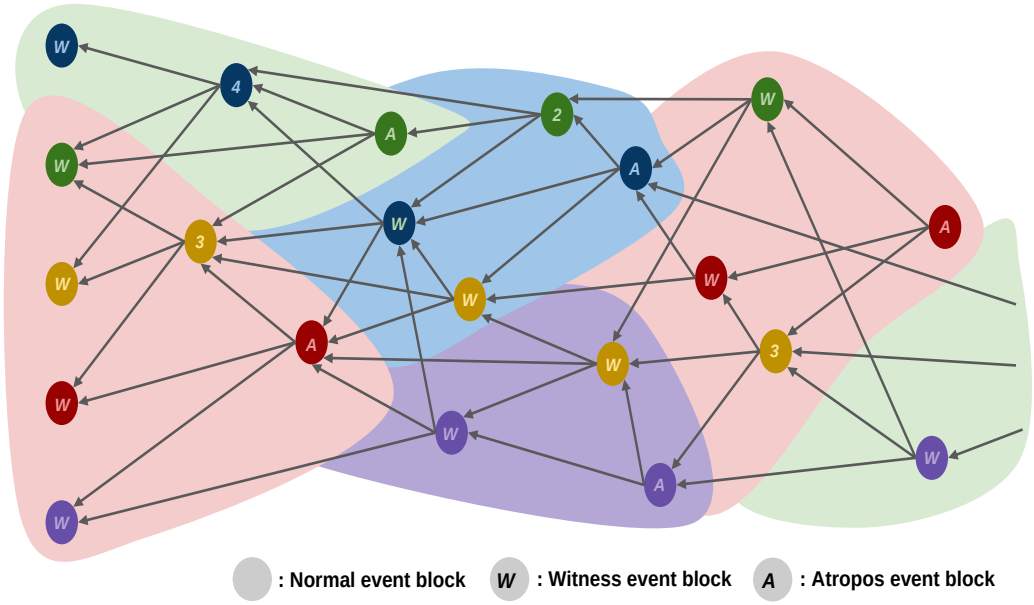

Figure 10: An Example of time ordering of event blocks in OPERA chain

In the above conditions of a and b, if weights of Atropos are equal, or event blocks within Atropos have the same weight, it can determine a priority as the timestamp. Then, if timestamp is also equal, the size of hash value determines the priority.

4. Proof of Lachesis Consensus Algorithm

In this section, we provides proof of liveness and safety in our OPERA chain and shows the Byzantine Fault Tolerance of OPERA chain. To shows the Byzantine Fault Tolerance, we need to an assumptions that more than two-thirds of participant nodes are reliable node. From the assumption, we provide some definitions and Lemmas and eventually show the Byzantine Fault Tolerance.

4.1. Preliminaries

Let $G = (V, E)$ denote directed acyclic graph (DAG). V is a set of vertices and E is a set of edges. DAG is a directed graph with no cycle. Namely, in

DAG, there is no path that source and destination at the same vertex. A path is a sequence P of vertices (v_1, v_2, ..., $v_{(k-1)}$, v_k) that uses no edge more than once. A vertex v_p is parent of v_c if there is a path form v_p to v_c and the length of path is 1. A vertex v_a is ancestor of v_c if there is a path form v_p to v_c and the length of path is more than equal to 1.

4.2. Proof of Byzantine Fault Tolerance for Lachesis Consensus Algorithm

Definition 4.1 (node)**.** *The machines participate the OPERA chain and have authority to create event blocks. the total number of Node is n.*

Definition 4.2 (event block)**.** *In OPERA chain, we call a vertex an event block.*

Definition 4.3 (self parent)**.** *An event block e_s is self parent of an event block e_c if e_s is parent of e_c and both event blocks have same signatures.*

Definition 4.4 (self ancestor)**.** *An event block e_a is self ancestor of an event block e_c if e_a is ancestor of e_c and both event blocks have same signatures.*

Definition 4.5 (share)**.** *An event block e_x share an event block e_y if there is a path from e_x to e_y.*

Definition 4.6 (Witness)**.** *The first created event blocks become witness or an event block e that can reach witnesses more than 2n/3 become witness.*

Definition 4.7 (Witness set)**.** *All first event blocks are elements of witness set W_1 ($|W_1| = n$). And the witness set W_k is a set of witnesses such that $w_i \in W_k$ cannot reach more than 2n/3 other witnesses in W_k ($k > 1$).*

Definition 4.8 (child-parent relationship of witness set)**.** *If there is witness set W_k and an event block e that can reach more than 2n/3 witnesses in W_k, e becomes the element of witness set W_l ($k \neq l$). In this case, W_l and W_k are child-parent relationship (W_l is child of W_k and W_k is parent of W_l).*

Definition 4.9 (Clotho)**.** *An witness w_a in witness set W_a is Clotho if more than 2n/3 witnesses in witness set W_p (W_p is the first child witness set of W_a) share w_a and the witnesses in W_p are shared by more than n/3 witnesses in W_q (W_q is the first child witness set of W_p) and there is an witness w_k in the fourth child set of W_a.*

Definition 4.10 (Atropos)**.** *Clotho become Atropos if the consensus time of Clotho is confirmed.*

Proposition 4.1. *More than 2n/3 witness in W_p are shared from more than n/3 witness in witness set W_g (W_g is the child witness of W_p).*

Proof. The number of witness in each witness set is more than 2n/3. Since a witness in W_g share more than 2n/3 witness in W_p, the number of share from W_g to W_p is $(2n/3)^2$. Thus, more than 2n/3 witness in W_p are shared from more than n/3 witness in W_g. □

Proposition 4.2. *If a witness in W_p is widespread in more than n/3 nodes in W_g, the witness becomes widespread in more than 2n/3 nodes.*

Proof. Because a witness in W_p is widespread in more than n/3 nodes in W_g, witness in witness set W_t (W_t is child witness set of W_g) must share the witness in W_p. a witness in W_a know that the witness in W_p become widespread in more than 2n/3 nodes, since witness in W_a share more than 2n/3 witness in W_t. □

Lemma 4.3 (Sharing)**.** *If a witness w_a in witness set W_a is created, more than 2n/3 witness in witness set W_p (W_p is third parent witness set of W_a) become widespread in more than 2n/3 nodes and the node creating the witness in W_a knows that widespread information.*

Proof. Based on propositions 4.1 and 4.2, more than 2n/3 witness in W_p become widespread in more than 2n/3 nodes and the node creating the witness in W_a know that widespread information. □

Lemma 4.4 (Fork)**.** *If the pair of event blocks (x, y) is fork, then the OPERA chain can structurally detect the fork before a witness sharing one of the fork is nominated as Clotho.*

Proof. Suppose that a node creates two event blocks (x, y) and the event blocks are fork. To create two Clotho that shares one of the pair respectively, the pair of event blocks must be spread in more than 2n/3 nodes. Thus, more than n/3 nodes must know the two fork event blocks. It means that all nodes must know the existence of the fork before two Clotho is nominated. For the reason, if there exist fork event blocks, then the OPERA chain can structurally detect the fork before a witness sharing one of the fork is nominated as Clotho. □

Theorem 4.5. *Each OPERA chain of all nodes grows up into same shape.*

Proof. Suppose that OPERA chains of nodes A and B has different shape (or structure). For any two nodes A and B, there is two event blocks x and y which are in both $OPERA(A)$ and $OPERA(B)$, and path between x and y in $OPERA(A)$ is not equal to that in $OPERA(B)$. In OPERA chain, a node communicates with another and synchronize both OPERA chain of them to create an event block. Thus, for two event blocks included in both OPERA chains, their path in one of OPERA chains should be equal to that in the other. Hence, in order to have different path between two event blocks in OPERA chain of two nodes, one of them is an attacker and it forks an event block. Based on Lemma 4.4, if an attacker forks an event block, OPERA chain detects and rectifies it before new witness is generated. It contradicts our assumption. Therefore, two nodes have consistent OPERA chain. □

Lemma 4.6. *For any witness set W, all nodes nominate same witness into Clotho.*

Proof. Based on Theorem 4.5, each OPERA chain nominates witness into Clotho by flag table. Because flag table consist of the information of structure in OPERA chain, if all nodes have OPERA chain with same shape, information in flag table must be equal. Thus, all nodes nominate same witness into Clotho since OPERA chain of all nodes has same shape. □

Lemma 4.7. *In reselection algorithm, for any Clotho, a witness in OPERA chain selects the same consensus time candidate.*

Proof. Based on Theorem 4.5, witness selects consensus time based on select information of previous witness. However, reselection algorithm is based on path search. Thus, if all nodes have OPERA chain with the same partial shape, a witness in OPERA chain selects the same consensus time candidate by reselection algorithm. □

Theorem 4.8. *Lachesis consensus algorithm guarantees to reach consensus for the consensus time.*

Proof. For any witness set W, time consensus algorithm checks whether there is a value selected by 2n/3 of nodes. However, each node selects one of values collected from parent witness set W_p by the time consensus and reselection algorithm. Through the reselection algorithm in time consensus algorithm, each node selects maximum frequency and minimum values. After some of

witness set, each node may select same consensus time candidate by reselection algorithm. However, there are possibility that consensus time candidate does not reach consensus. To solve the problem, time consensus algorithm includes minimal selection round per next h^{th} child witness set. In minimal value selection algorithm, each node selects minimum value among values collected from parent witness set. Thus, the consensus time reaches a consensus by time consensus algorithm. □

Theorem 4.9. *If the number of reliable node is more than 2n/3, an event block x created by reliable is assigned consensus time, and other event blocks created after event block x has consensus time t must be assigned later consensus time than t.*

Proof. Because reliable nodes in OPERA chain try to create continuously event blocks by communicating with every other node, reliable node will share the event block x. Based on Proposition 4.1, if a witness y in witness set W share event block x and more than n/3 witness in witness set W_c (W_c is the child witness set of W) share the witness y, the witness will be nominated as Clotho and Atropos some time. Then event block x and witness y will be assigned consensus time t. For an event block, assigning consensus time means that the event block is known by more than 2n/3 nodes. Therefore, malicious node cannot modify event blocks after the event blocks are assigned consensus time. After event block x has consensus time t, it is not possible to later discover a new even block z that will be assigned earlier consensus time than the consensus time y. To be assigned earlier consensus time than y, event block z must satisfy two condition. first condition is that a witness w in W shares z. and the second is that more than n/3 witness in W_c share w. If the two condition is satisfied, z can be assigned earlier consensus time than y. Malicious nodes make an event block satisfying first condition like parasite chain. However, the second condition cannot be satisfied because more than 2n/3 witness in W_c already created when x is assigned t. Therefore after an event block is assigned consensus time, new event blocks cannot be assigned earlier consensus time than 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. Previous Approaches of DAG

6.1. Tangle

IOTA [16] 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.

6.2. Byteball

Byteball [5] 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.

6.3. RaiBlocks

RaiBlocks [12] 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.

6.4. Hashgraph

Hashgraph [2] 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.

7. Conclusion

In order to realize the distributed ledger technology, we have proposed a new asynchronous DAG-based consensus algorithm. The consensus algorithm is named a Lachesis consensus algorithm (LCA) and is the structure of consensus between event blocks through Lachesis protocol. Especially, the LCA checks reachabilities from an event block to Witness using flag table in a DAG structure that provides proof of the process of satisfying the consensus of pBFT. To solve the Byzantine fault tolerant problem, the elected nodes form the network by applying a DPoS like method. To ensure the distribution of participating nodes, the Lachesis protocol used the cost function. And, for efficient and quick selection process, Witness and Clotho are elected based on the flag table. Then, Atropos is selected by Weight after

time consensus ordering. In this process, all paths search need not be calculated every moment. The flag table can efficient calculate share information without path search algorithm at the certain moment. In terms of time complexity, the LCA performs more effective calculation for consensus based on flag table than path searching algorithm.

Based on Lachesis protocol and consensus algorithm, OPERA chain can protect malicious attacks such as fork, double spending, parasite chain, and network controls. These protections guarantee the safety of OPERA chain. We can also verify existence of Atropos with OPERA chain. It means that OPERA chain reaches consensus and guarantee the liveness.

Finally, the time ordering ensures guarantee by weight value on the flag table. Based on these properties, the LCA provides a fair, transparent, and effective consensus algorithm.

8. Reference

[1] J. Aspnes. Randomized protocols for asynchronous consensus. *Distributed Computing*, 16(2-3):165–175, 2003.

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

[3] 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.

[4] J. Chen and S. Micali. Algorand. *arXiv preprint arXiv:1607.01341*, 2016.

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

[6] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[7] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

[8] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong. Zyzzyva: speculative byzantine fault tolerance. *ACM SIGOPS Operating Systems Review*, 41(6):45–58, 2007.

[9] L. Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

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

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

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

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

[14] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42. ACM, 2016.

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

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

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

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