LFT2

ICONLOOP

Abstract. In this paper, we present LFT2, a novel PBFT based consensus algorithm with a reduced 2-step consensus process. LFT2 is a leader-based Byzantine fault-tolerant replication protocol for a partially synchronous model. LFT2 successfully merges commit messages from Pre-Prepare, Prepare, Commit (3 stage) to Propose, Vote (2 stage), at the same time guaranteeing liveness and safety. LFT2 incorporates Candidate/Commit block mechanism to resolve security issues that could be raised in a simplified consensus process.

1. Introduction

Proof of Work (PoW) powered blockchains such as Bitcoin and Ethereum, once popularized the consensus algorithm and led the cryptocurrency industry with more than 90% of total market capitalization. PoW algorithm, however, deliberately consumes immense amounts of energy to attach higher costs to mining, arguably wasted energy only to instill higher risks to bad actors. This computationally expensive algorithm only achieves a somewhat secure network (still vulnerable to 51% attacks), with a slow block generation time and consensus finality. This makes PoW implausible for many applications such as financial sectors that require high performance to accommodate its services. Many blockchain projects since then have adopted PBFT as its choice of consensus algorithm, which is not only safer but also much more performant and computationally more efficient. One PBFT based example - Tendermint, which improved upon PBFT with optimization decisions such as removing View/Change to simplify faulty leader replacement, makes it more suitable for blockchain applications. PFBT-based algorithms, however, require large amounts of message exchanges to reach consensus. This is less ideal, in some cases, even critical for the systems to function as normal, such as systems with complex smart contract logic. We have also seen efforts to reduce message complexity from other PBFT-based protocols such as Casper FFG and Pipelined.

LFT2 is a lightweight consensus algorithm based on PBFT, tailored specifically for blockchain systems. LFT2 has reduced the 3-step consensus process from PBFT to 2 steps, effectively reducing communication messages needed to reach consensus. With a simplified messaging mechanism, network bottleneck is eased and block confirmation speed is improved. Additionally, LFT2 guarantees safety and liveness by incorporating the Candidate/Commit block mechanism in order to resolve security issues that could be raised in a simplified consensus process model.

Section 2 of this paper explains the LFT2 consensus algorithm and its process. Safety and liveness are proved in Section 3.

2. LFT2

LFT2 assumes that the network is partially synchronous which ensures bounded message transmission time between nodes through the gossip communication. Furthermore, LFT2 assumes that a node in a blockchain network has a duty of block generation, validation, possession, and that the node can generate a digital signature that can identify its own message. LFT2 guarantees safety and liveness when the number of entire nodes is more than or equal to $3f+1$ where f is the number of byzantine nodes. Therefore, in LFT2, a block is validated if $n-f$ number of nodes reach a consensus where there is a total n number of nodes participating in the consensus process and f is the maximum possible number of byzantine nodes. That is, a block is validated if it gets $n-f$ votes which form a quorum in the consensus process.

2.1 Glossaries

Term	Definition
Round	A part of the block consensus process. A serial number of a Round is incremented by 1 and set to 0 if the Node Set is changed
Node Set	All nodes in the consensus process including the Leader and Validators
Leader	The node that generates a new block and broadcasts it to the network
Validator	A node that validates the block suggested by the Leader
Propose	The event of the Leader creates a block and broadcasts it
Vote	The event of votes are casted and broadcasted to the network
Candidate Block	The block of the Round becomes a Candidate Block if the Round is successful
Commit Block	The Candidate Block becomes a Commit Block and written in the blockchain if the block that is connected to the Candidate Block reaches consensus

2.2 Assumptions and Rules

A system and nodes of LFT2 consensus algorithm operates on the assumptions that are explained in *2.2.1.* Furthermore, nodes that are participating in a block consensus follow t*he rules of 2.2.2.* Nodes that are not following the rules are regarded as byzantine nodes. LFT2 assumes that the maximum number of byzantine nodes in the network is *f.*

2.2.1 Assumptions

A Node participating in LFT2 consensus algorithm meet the following assumptions

- If a node gets a message, it sends the message to its adjacent nodes within a certain range of time(Δ) (gossip communication)
- There is a separate algorithm for Leader election and every node knows the order of Leader.
- A node has a local timer.
- All nodes share Crypto Suite in advance. A message that is sent to other nodes includes a signature.

A byzantine node which wants to stop or fork the network can perform the following actions.

- A byzantine node can delay a message or choose not to send it to other nodes.
- A byzantine node can send different messages to different nodes.
- A byzantine node cannot generate a signature of a different node.

2.2.2 Rules

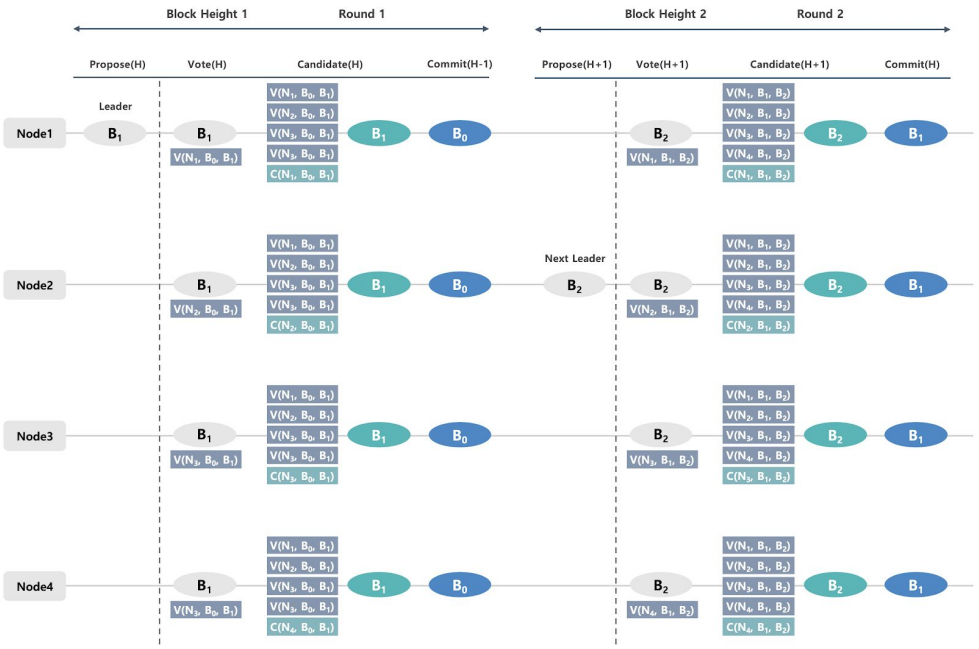
A non-faulty node follows the rules below:

- Rule 1. Leader Election
    - In each round, a designated node for the Round becomes a Leader.
    - Every node knows the order of the Leader.
- Rule 2. Timer
    - ProposeTimer
        - When the Propose stage gets started, ProposeTimer starts.
        - If the Leader does not create a block and deliver it to the network within the time tolerance, ProposeTimeout is called and Fail Vote gets started.
    - VoteTimer
        - In the Vote stage, if more than enough votes are aggregated to meet the quorum but consensus is not reached, VoteTimer starts running.
        - If consensus is made within the given time, VoteTimer is terminated.
        - If consensus is not made within the given time, VoteTimer starts running.
- Rule 3. Propose
    - The Leader makes one and only block for each round.
- Rule 4. Vote
    - A Validator votes for only one block that is generated by the Leader for each Round.

    - A Validator votes for a block that is connected to its own Candidate Block.
- Rule 5. Candidate
    - If a validator finds a block that is at higher Round than its current Candidate Block and meets a quorum, then the validator sets the block as its new Candidate Block.
    - If a validator finds a block that meets a quorum and is of higher height than its current Candidate Block, a node sets the block as its new Candidate Block.
- Rule 6. Commit
    - If a validator replaces its Candidate Block, the node Commits the previous Candidate Block.

2.3 Consensus Algorithm

This section describes the LFT2 consensus algorithm with an example of 4 nodes. Figure 1 illustrates the LFT2 consensus algorithm. The LFT2's events happen from left to right. The Events of different stages are divided by the dotted lines. Each node sends messages of each event to other nodes. The definitions of the notations in the illustrator are as follows.



<Figure 1> An example of LFT2 Consensus Algorithm

- B : Block ID
- V(N, B', B) : Vote(Node Number, Candidate Block ID, Target Block ID)
- C(N, B', B) : Candidate(Node Number, Previous Candidate Block ID, Candidate Block ID)

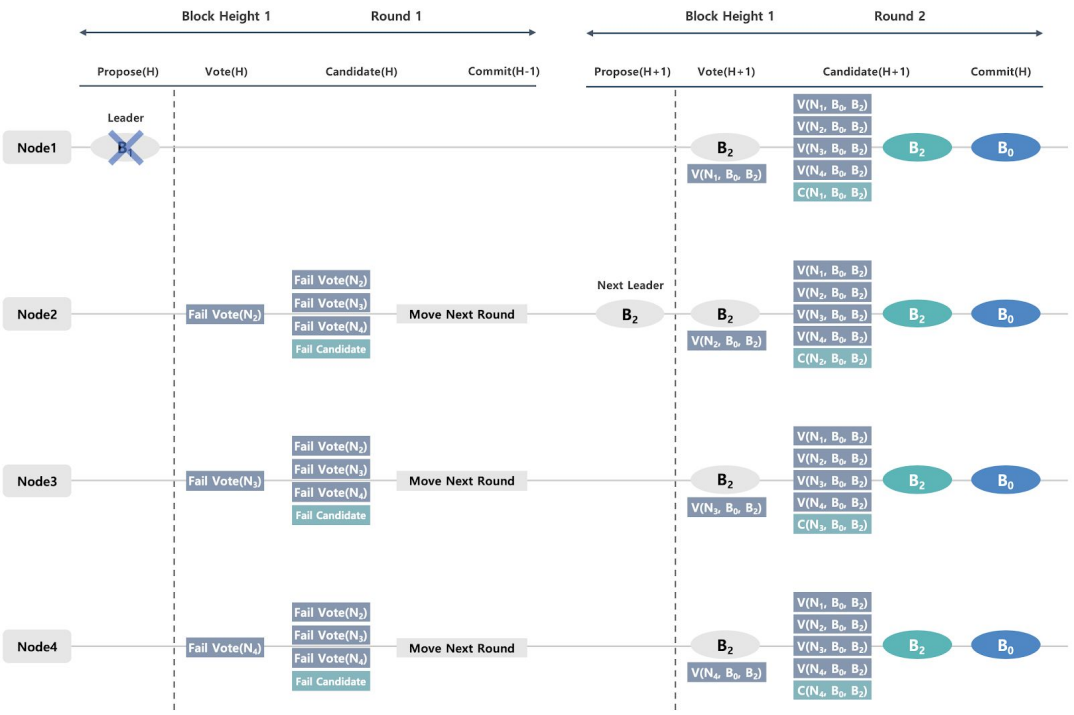
The stages can be differentiated by Round or the block height. Events taking place for each stage are as follows:

- **Propose**

    - As the new Round gets started, ProposeTimer is reset and starts running. And the Leader gets replaced by the next node in line of Leader that will create a next new block.
    - The Leader generates a block of new height which contains the information of transactions and the votes of Candidate Block.
    - The Leader sends the new block and its digital signature to other Validators.
- **Vote**
    - Validators receive the new block generated by the Leader.
    - Validators check whether the delivered block is generated by a valid Leader.
    - Validators check whether the block height and the hash of the previous block written in the block delivered is valid.
    - Validators check whether the delivered block is connected to Candidate Block
    - Validators make a vote message according to the results of the tests above and send it to Node Set.
- **Candidate**
    - A Node Set gets vote messages from Node Set except itself.
    - When Node Set gets more than enough votes to meet a quorum but consensus is not reached, VoteTimer starts running and it follows VoteTimer conditions of *Rule 2. Timer.*
    - Candidate Block information is not included in the next block and only each node knows it.
- **Commit**
    - When Node Set gets the same V(N, B',B) votes enough to form a quorum, It Commits the block B'.
    - It enters Propose stage in the next Round.

2.4 Round Advance

Round Advance represents the process of moving on to the next round even if a block is not generated correctly at a certain Round or the new block is rejected since the votes do not agree on the new block. Figure 2 illustrates the block consensus process where the Leader failed to generate a new block correctly in Round 1 which is a Round failure, but the process moves on to the next Round and a block gets accepted eventually.



<Figure 2> Round Advance (f=1)

The state information of each node in Figure 2 is as follows:

- **Round 1**
  - Node 1 (Leader) : Block generation failure
  - Node 2, 3, 4 (Validator) : Failure vote message generation / Normal operation
- **Round 2**
  - Node 2 (Leader) : Normal operation
  - Node 1, 3, 4 (Validator) : Normal operation

In a new Round for a block generation, ProposeTimer starts running. Then, Node 1, as a Leader, has the duty to make a block B1 of the new height with transactions collected and send it to other validators with its digital signature. However, Figure 2 shows the case when the leader could not generate a block B1 because of an error. Therefore, Node 1 fails to generate a new block and send it to the Validators within the time specified by ProposeTime. Hence, a Validator sends a fail vote message that says it did not receive block B1. VoteTimer does not start running since every node receives a quorum of failed votes. The current Round decides that the block consensus process in its Round is a failure and moves on to the next Round.

In the next Round, ProposeTimer starts running like it did in the previous Round. Then Node 2, the new Leader for the Round, generates a new block within the given time and sends it to all Validators with its digital signature. Each Node 1, 3, and 4 checks the block B1, which is delivered from the Leader, respectively and decides that B1 valid and sends an okay vote to all Validators. Since each node receives the quorum of okay votes hence consensus is reached, VoteTimer does not start running, and the Candidate Block B0 in the previous Round gets replaced by B_1 in the current Round. Then, block B0 becomes a Commit Block and is written into the blockchain.

As illustrated above, LFT2 can reach a consensus even if the Leader fails to send block B_1 within a given time in Round 1, because Round Advance keeps the process moving forward so that consensus on block B2 is reached on the next Round without network error.

3. Proof of Consensus Algorithm

In order to prove a Byzantine consensus algorithm, it must be shown that a network fork does not get triggered by less than *f* number of byzantine nodes, and consensus will eventually be made in the network [2,10,1]. Therefore, we prove safety of LFT2 which guarantees no fork in the partially synchronous network and liveness of LFT2 without an error. Then, it is proven that LFT2 provides consensus finality which ensures that once consensus is reached in a blockchain, it is unchangeable.

Theorem 1. Let B(prev_block, height) be a block of height and its previous block is prev_block. In the LFT2 consensus algorithm process, different blocks B(B0, H) and B'(B0, H) that are of the same height are not committed together.

Proof:

Suppose that B(B_0, H) and B'(B_0, H) are different two blocks connected to B0 and they are committed together. Then two blocks B''(B, H + 1) and B'''(B', H + 1) of higher heights connected to B and B' respectively become Candidate Blocks by *Rule 5. Candidate Block* If B''(B, H + 1) becomes a Candidate Block, more than *n-f* number of nodes have set prev_block B as the Candidate Block. Likely if B'''(B', H + 1) becomes a Candidate Block, more than n-f number of nodes set B', prev_block of B''', as the Candidate Block. Therefore, in the network consisting of *3f + 1* nodes, there exist *2f + 1* nodes that have set B as the Candidate Block and another *2f + 1* nodes that have set B' as the Candidate Block. Thus there are *4f + 2* nodes that have vote rights. Even if *f* number of byzantine nodes vote twice, the total number of votes that can be generated is *3f + 1 + f*, that is, *4f + 1* which contradicts the assumption that two different blocks B(B0, H) and B'(B0, H), both connected to B0, are committed. Therefore, different blocks of the same height cannot be generated in the LFT2 consensus algorithm.

3.1 Liveness Proof

Liveness of a consensus algorithm is a property that consensus of a network will eventually be reached, never left being impossible to reach a consensus forever. In this section, in order to prove that LFT2 guarantees liveness in the network with less than *f* number of byzantine nodes, we show that a Round never gets trapped in the endless waiting state and will eventually be completed, and then we show that every round will eventually reach a consensus.

Theorem 2. If every node is in the same Round, consensus is reached in the Round.

Proof:

When a non-faulty node enters a new Round, ProposeTimer starts running and pro or con vote begins regardless of whether a new block is generated by the Leader. If a quorum of votes are delivered, the process can move on to the next Round whether consensus is reached or not

thanks to VoteTimer. Therefore, even if less than *f* number of faulty nodes participate in the consensus, the consensus in the Round never stops.

Theorem 3. Let Δ be the network transmission time in the network through gossip communication. If a non-faulty node enters a new Round at time *t*, all non-faulty nodes enter the new round within $t + \Delta$.

Proof:

Suppose non-faulty Node *n* received a quorum of votes at time *t'*. Then non-faulty nodes can receive a quorum of votes within $t' + \Delta$ through gossip communication.

If Node *n* reached a consensus and moved to the next Round at *t*, all non-faulty nodes move to the next Round within $t + \Delta$. If the process moved on to the next Round because the previous Round timed out, then *t* is *t'* + VoteTimeout. Then, since all non-faulty nodes received a quorum of votes at $t' + \Delta$, the process moves on to the next Round within $t + \Delta$, that is, *t'* + VoteTimeout + Δ.

Theorem 4. Let Δ be the network transmission time through gossip communication. Assume that block generation and block validation take 0 time. If $2\Delta <$ ProposeTimeout, all nodes in a Round with a non-faulty Leader receive the new block generated by the Leader.

Proof:

If a non-faulty node enters a new Round at t, all nodes enter the new Round within $t + \Delta$ by *Theorem 3*. That is, the predesignated node that would be the Leader of the new Round enters the new Round from the current Round within $t + \Delta$. Therefore, every node can receive the new block within $t + 2\cdot\Delta$. Since $2\cdot\Delta <$ ProposeTimeout, the first Node that enters the new Round can receive a new block before ProposeTimeout.

Hence, by *Theorem 4*, if a non-faulty Leader generates a new block, all nodes receive the new block before the ProposeTimeout event takes place.

Theorem 5. Let Δ be the network transmission time in the network through gossip communication. Assume that block generation and block validation is not considered. If $2\cdot\Delta <$ ProposeTimeout, $2\cdot\Delta <$ VoteTimeout, then a Round with a non-faulty Leader eventually reaches a consensus.

Proof:

As shown in Theorem 4, all non-faulty nodes receives a new block before ProposeTimeout happens. In the process, a byzantine node may reduce VoteTimer to keep consensus from being reached. If Leader broadcasts a new block at *t*, every node receives the new block within $t + \Delta$. And votes of all non-faulty nodes are sent to all non-faulty node within $t + 2\Delta$. Even if *f* number of byzantine nodes invoke VoteTimeout at *t* to nodes with no transmission delays, since *t* + VoteTimeout is after $t + 2\Delta$, the Round reaches a consensus before VoteTimeout happens.

It is shown that if non-byzantine node is the Leader then the Round eventually reaches a consensus by *Theorem 5*, and even if the leader is a byzantine node, the Round is completed eventually too by *Theorem 2*. Therefore, LFT2 guarantees that consensus is always reached even if a byzantine node is the Leader and the network stops.

4. Conclusion

Blockchain provides safe transaction platforms without a Trusted Third Party being involved and provides an immutable database by reaching a consensus on an open network. The safety of a blockchain and its application varies depending on a consensus algorithm that the chain uses. For example, if a blockchain uses a consensus algorithm that guarantees consensus finality, then it only takes one or two blocks to show that a block actually exists in the blockchain even though only designated representatives participate in the consensus.

LFT2 is a lightweight 2-step consensus algorithm that is designed in such a way to merge PBFT's Commit message transmission process with the Block Propose of the next Round. LFT2 guarantees safety and liveness by making a Validator only able to vote for blocks that are connected to its Candidate Block. LFT2 provides consensus finality, and it reduces the PBFT Algorithm's message transmission process by one step. By reducing the message transmission process, the burden on the network for consensus process becomes lighter. The time needed for consensus agreement will be decreased, hence increase the performance of the transaction throughput. Additionally, the need for synchronizing local time has been a problem for previous 2-step consensus algorithms such as EOS Pipelined BFT, however LFT2 does not need to synchronize local time even though LFT2 assumes the partial synchronous network. However, LFT2 might be slower than other algorithms that are based on 1-step(Round) consensus due to the property that LFT2's block consensus is not reached in the current Round but in the next Round.

References

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

[2] M. Castro, B. Liskov, et al., "Practical byzantine fault tolerance", 1999.

[3] J. Kwon, "Tendermint: Consensus without mining", 2014.

[4] E. Buchman, "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains", 2016.

[5] E. Buchman, J. Kwon, Z. Milosevic, "The latest gossip on BFT consensus", 2018.

[6] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance" 1987.

[7] Vitalik Buterin and Virgil Griffith, "Casper the Friendly Finality Gadget", 2019.

[8] Vitalik Buterin, Daniel Reijsbergen, Stefanos Leonardos, Georgios Piliouras, "Incentives in Ethereum's Hybrid Casper Protocol", 2019.

[9] BitFury Group, "Proof of Stake versus Proof of Work", 2015.

[10] C. Dwork, N. Lynch, L. Stockmeyer, "Consensus in the Presence of Partial Synchrony", 1988.

[11] C. Cachin, M. Vukolic, "Blockchain Consensus Protocols in the Wild", 2017.

[12] Daniel Larimer, “DPOS BFT - Pipelined Byzantine Fault Tolerance”, https://medium.com/eosio/dpos-bft-pipelined-byzantine-fault-tolerance-8a0634a270ba, 2018. [13] Digiconomist, “Bitcoin energy consumption index.”, 2018.

[14] T. Swanson, “How much electricity is consumed by Bitcoin, Bitcoin Cash, Ethereum, Litecoin, and Monero?”, 2018.