Chapter 4. Algorithm Steps of Locus Chain Consensus Algorithm

▪ The validity of transactions independent of Consensus Algorithm

In Locus Chain, the validity of each transaction could be verified without the Consensus Algorithm in most cases. By the nature of Locus Chain's AWTC ledger structure, only the account owner (who possesses the account's secret key) can issue and add transactions to the account's transaction chain. Every transaction added is a valid and good transaction, unless the account with the secret key performs Byzantine behaviors purposely.

Byzantine account may create double-spending situations by issuing two or more different transactions (double-spending transactions) with the same index number. However, the second transaction is dropped by network nodes if the first transaction is properly propagated before the second transaction. For example, we assume that about 10 second is enough time for propagating a network transaction over a shard. If the time interval between two double-spending transactions is long enough to propagate the first transaction over the whole shard (say, about 10 seconds), or double-spending transactions are not issued in the same round, the second transaction is dropped by each network node, and only the first one becomes valid.

The problem becomes complex if two or more transactions are issued simultaneously. It is hard (or not feasible) to select one transaction uniquely before executing the consensus algorithm. However, each node may detect simultaneously issued transactions by comparing receiving timestamps of double-spending transactions. If a node received two or more transactions of same index number in a relatively short period (say again, 10 seconds), then the node may mark the transactions as conflict and stop processing those transactions until the consensus algorithm resolves the conflict.

In summary, each transaction is immediately valid, without the result of the consensus algorithm, unless the secret key owner explicitly conducts Byzantine double-spending activities. Locus Chain's consensus algorithm resolves the double-spending transactions when they exist. Furthermore, the consensus algorithm builds additional stability for Locus Chain Network operation, including Sharding and Pruning.

▪ Steps of Locus Chain Consensus Algorithm

The core idea of Locus Chain's Consensus Algorithm is based on a property of gossip network, which distributes information over network nodes fairly and repeatedly. This property makes network nodes eventually have the same information received. Our algorithm is about finding out the information received by the majority of network nodes.

Our consensus algorithm can be roughly described as follows; first, the state of the received information is proposed by nodes as several digested hash numbers. Then the proposed hash numbers are counted, compared, and selected as the official state of received information.

Our algorithm is executed on a per-Round basis. The algorithm is consists of four steps; “Committee Election Step,” “Round State Propose Step,” “Round State Selection Vote Step,” “Round State Commitment Step.”

The role of each step is as follows.

- **Step 1: Committee Election**

  Not all nodes in shards participate in the consensus algorithm. The algorithm is supposed to be processed by a certain number of node groups, to balance and optimize network and computational resources.

  Two groups of nodes (or two committees) are needed to run the algorithm. The first group of committee nodes is ‘Proposer Committee,’ which proposes Round State Value. Round State Value is a digested, representative hash value of received transactions during a round. The second group is ‘Voter Committee,’ which verifies, counts, and votes for proposed Round State Values. Both committees are newly elected for each round.

  Committee member nodes are elected using reproducible pseudo-random numbers that can be calculated using disclosed information. For example, A Verifiable Random Function (VRF) using disclosed ledger information is a good fit for this purpose.

  When a new round begins, each node in a shard checks whether elected as a committee member or not, by calculating its own random value for the round. Each elected nodes broadcast ‘Election Message,’ which contain the node’s identity and information can be used to verify the calculated random number. All nodes in the shard listen for Election Messages to find out the number of committee members elected.

  Each shard and rounds have its own desired size (cardinality) of committee nodes, but the random nature of the election process makes the actual number different for each election. The actual size of the committee could be known by gathering election messages.

  Elected committee nodes are rewarded with incentives if they faithfully participate in the consensus algorithm.

- **Step 2: Round State Propose**

  Each round has fixed start time and end time. When certain time is passed after a round’s end, the Proposer Committee proposes Round State Values. Each node in the Proposer Committee independently calculates its own Round State Value by digesting transactions

received during the round

Currently, Round State Value is the top-level hash value of a Merkle-Patricia Trie (MPT). Each proposer committee node creates an MPT by inserting transactions into the MPT. MPT is a kind of Radix Trie. Like other tree/tries, an MPT is generated by adding elements one by one. The MPT is free of the insertion order. In other words, the same MPT will be created regardless of the order of the inserted elements. Also, each trie-node of an MPT contains a representative hash value. Each proposer committee node creates its MPT by adding all incoming transactions in real-time. The calculated final value of the MPT's root node will be used as the Round State Value for the committee node.

When Round State Value is calculated, the committee node broadcasts a Round State Propose Message (or just Propose Message) signed with its secret key. The message contains information including Round State Value and the number of transactions

- **Step 3: Round State Selection Vote**

  After the Round State Propose Step, the shard is flooded with Round State Propose Messages as many as the number of Proposer Committee nodes. Round State Selection Vote is the process of selecting one proposed round state which mostly suits for the round. Each Voter Committee nodes listens for Propose Messages. If Propose Message of a certain condition is received, or enough number of Propose Messages are received, the node selects a Propose Message which suits for predefined criteria like follows.

  - ✓ Criteria 1: Select the proposal that Round State Value is the same as the voter node's own Round State Value.
  - ✓ Criteria 2: If criteria 1 is not satisfied, select a proposal that its transaction count close to the node's own received transaction count.
  - ✓ Criteria 3: If multiple proposals are matched, compare the numerical value of Round State Value and the random election value of the Proposer Node, to choose one.

  The voter node selects one proposal, then composes and broadcasts a Selection Vote Message signed with its secret key.

- **Step 4: Round State Commitment**

  Each voter committee nodes listens and counts for Selection Vote Messages issued in previous step.

  If one single Round State Value collects over 2/3 of all Selection Votes, the Round State Value is regarded as "confirmed" for the state of the round. When a Voter Committee node

observes a Round State Value having 2/3+1 votes of Voter Committee size, then marks the Round State Value as confirmed, then composes and shares a “Round State Confirmation Message” for confirmed Round State Value. The confirmed Round State Value will have multiple signs of several voter nodes. Signing algorithms like Schnorr Signature are used to optimize the signing process.

All nodes in the shard listen for Confirmation Messages. A node finalizes the round as confirmed when the node receives enough number of signs for a Confirmation Message. Each Voter Committee Node will issue a “Confirmation Failure Message” (or Null Confirmation Message) when no Round State Value collects 2/3+1 of votes. If a node receives enough number of Failure Messages, then the round is regarded as a failure to reach an agreement. The failed round will be resolved by further actions.

▪ Round State Chain

Transactions of Locus Chain are placed in AWTC structure and managed by its owner accounts. However, the result of the consensus algorithm does not belong to any particular account. Locus Chain uses a special type of global transaction chain, named “Round State Chain,” to manage information delivered by the consensus process.

Round State Chain contains global information for each round, like MPT hash value of ledger state, number of included transactions, information regarding Proposer Committee and Voter Committee, and hash for the previous round state. This information is contained actually in the confirmed round state and stored almost as-is on the Round State Chain.

▪ Clock Timing of the Consensus Algorithm

Each step of Locus Chain’s consensus algorithm is executed in sync with certain time schedules. Currently, each round of Locus Chain is about 2 minutes in length, which means all steps should be completed in two minutes. Start time and timeout of each step are decided accordingly. Also, the size of shards and committees are adjusted regarding performance goals.

The first step (“committee election step”) begins simultaneously with the round. However, the second step is delayed to a certain time after the end of the round. The delay ensures the propagation of transactions over the shard. Currently, the delay is about one minute long.

Because a time clock is critical for our algorithm, we assume that each network node has a clock accurate enough, with a maximum of several seconds of error. If required, a clock-matching protocol is used to detect clock errors.

In the next chapter, we will see how failed agreements are recovered.