When a node elects itself, it timestamps this election with the current value of its logical clock. This is a big problem when trying to determine whether an event happened before a given point in time. Logical clocks, however, only capture the happens-before relationship and have no correlation to actual time. Therefore, we decided to use the idea of logical clocks: timestamping every message with the current logical clock value and updating the logical clock upon receipt of a message.

Algorithm Overview

- **Main goal:** After topology changes stop, maintain a leader-oriented directed acyclic graph in each connected component.
- When a node becomes a sink (loses all of its outgoing links), it starts a new reference level (RL) – a wave in search of a path to the leader. This new RL is reflected back to the leader, and the search stops.
- If it finds a path to the leader, the search stops.
- Once the RL reaches a dead end, the wave is reflected back.
- When a node receives a reflected RL from all of its neighbors, it means that it is disconnected from the leader and, thus, it elects itself. When a node elects itself, it timestamps this election with the current value of its logical clock.
- As the wave propagates, via messages sent by the nodes, all logical clocks along the way are updated.

Logical Clocks

- Logical clocks: non-negative integer values (initially 0).
- Nodes timestamp all messages they send with the value of their logical clock.
- The logical clock of a node is updated to 1 more than the maximum of its current logical clock value and the timestamp of the received message.

Example Algorithm Execution

- **Leader-oriented DAG**
  - Leader-oriented DAG
  - Start new reference level (timestamp with the current logical clock value)
  - Reflect reference level
  - Elect Self (timestamp with the current logical clock value)
  - Correctness Proof
    - **Main goal:** Show that eventually the algorithm terminates correctly, forming a leader-oriented directed acyclic graph in each connected component.
    - After the last topology change, no node elects itself an infinite number of times.
    - Most of the properties used in this proof hold for both logical and perfect clocks.
    - Eventually, there are no messages in transit.
    - Every node has an accurate view of its neighbors’ heights.
  - Future Work
    - **Stability proof** (ongoing work): Under specific circumstances, a leader is not elected unnecessarily. Instead, nodes that are still in the same connected component as the leader try to find an alternative path to it.
    - Better characterization of situations in which the leader is not elected unnecessarily.