FOROS

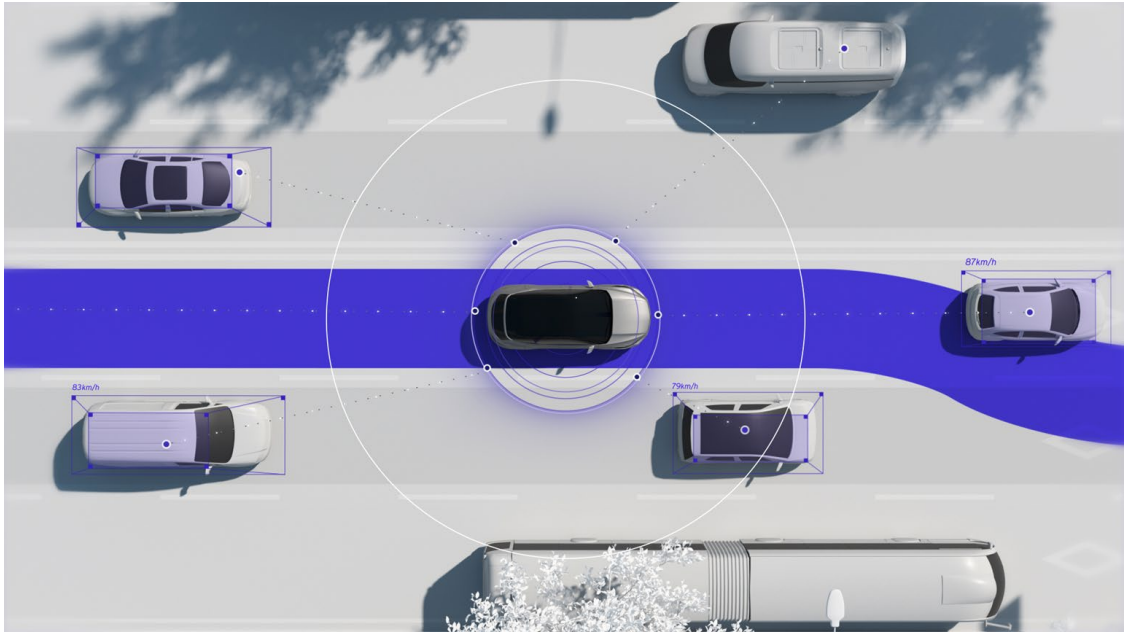
Failover ROS Framework

Wonguk Jeong (wonguk.jeong@42dot.ai)

Overview

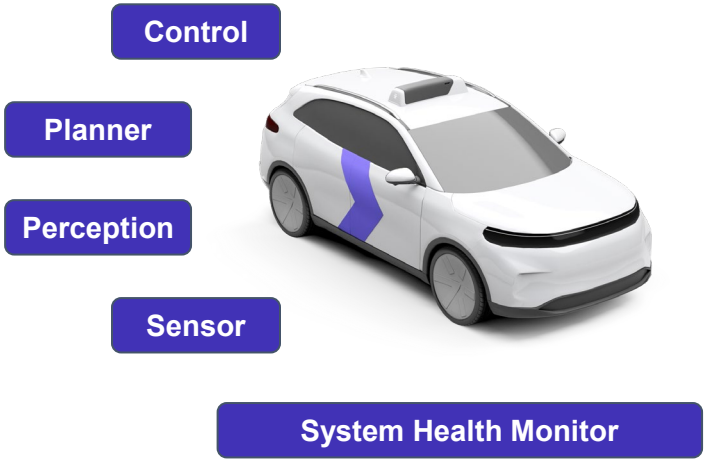
Motivation

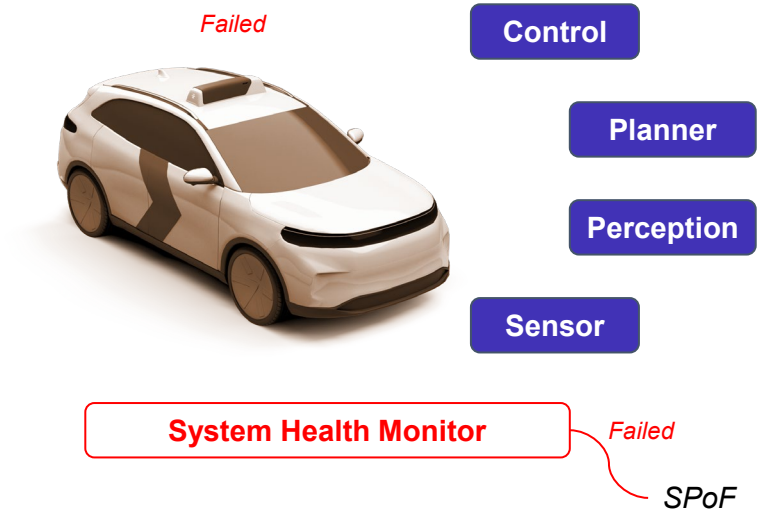
Reliability and **Safety** are essential for autonomous vehicles and robots.



How?

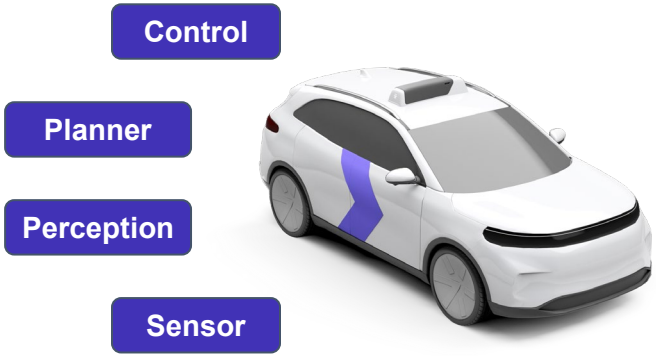
Mitigating **Single Points of Failure** (SPOFs)





So, How?

Enhancing high availability of safety - critical modules using **redundancy** *(= clustering)*



Active	System Health Monitor
Standby	System Health Monitor
Standby	System Health Monitor

System Health Monitor	*Failed*
System Health Monitor	*Active*
System Health Monitor	*Standby*

FOROS

github.com/42dot/foros

Failover ROS Framework

An open source ROS2 framework that can be used to provide **redundancy** for safety-critical nodes using a **RAFT consensus algorithm** with **minimal effort.**

Constraint

This framework can tolerate failures equal to the **cluster size minus quorum** .

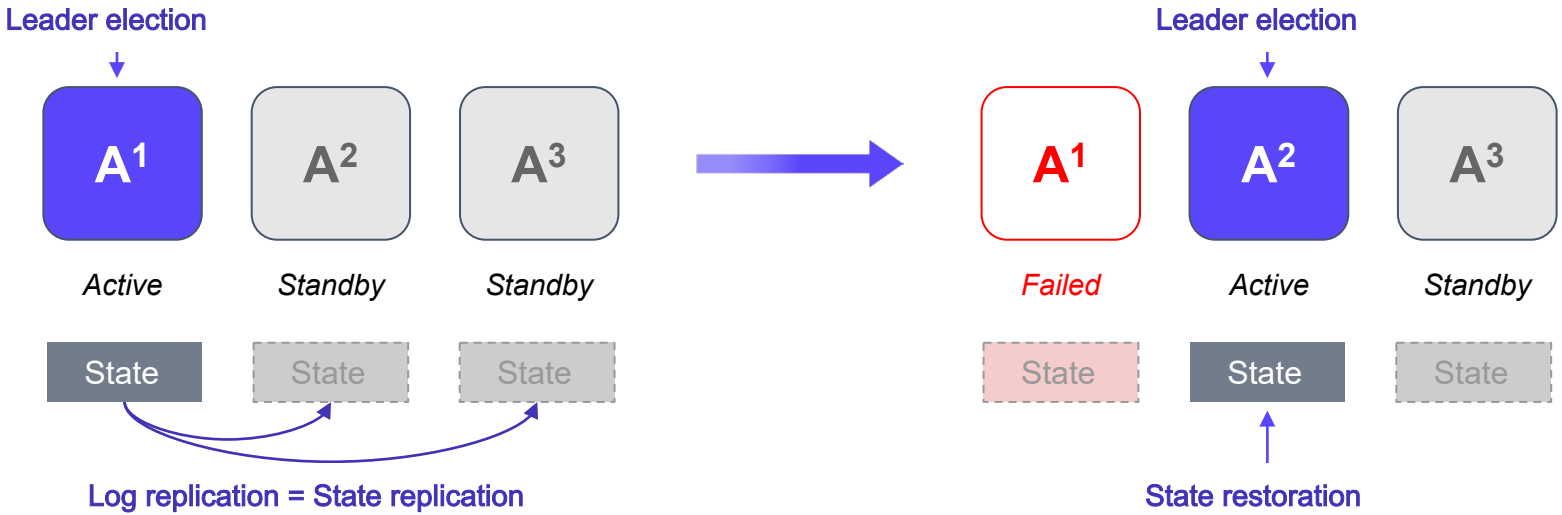
Cluster Size (N)	Quorum (Q = N / 2 + 1)	Number of fault tolerant nodes (N - Q)
1	1	0
2	2	0
3	2	1
4	3	1
5	3	2

This framework tolerates **fail-stop failure** but NOT **Byzantine failure**

- **Fail-stop failure** : the component stops operating.
- **Byzantine failure** : there is imperfect information on whether a component has failed

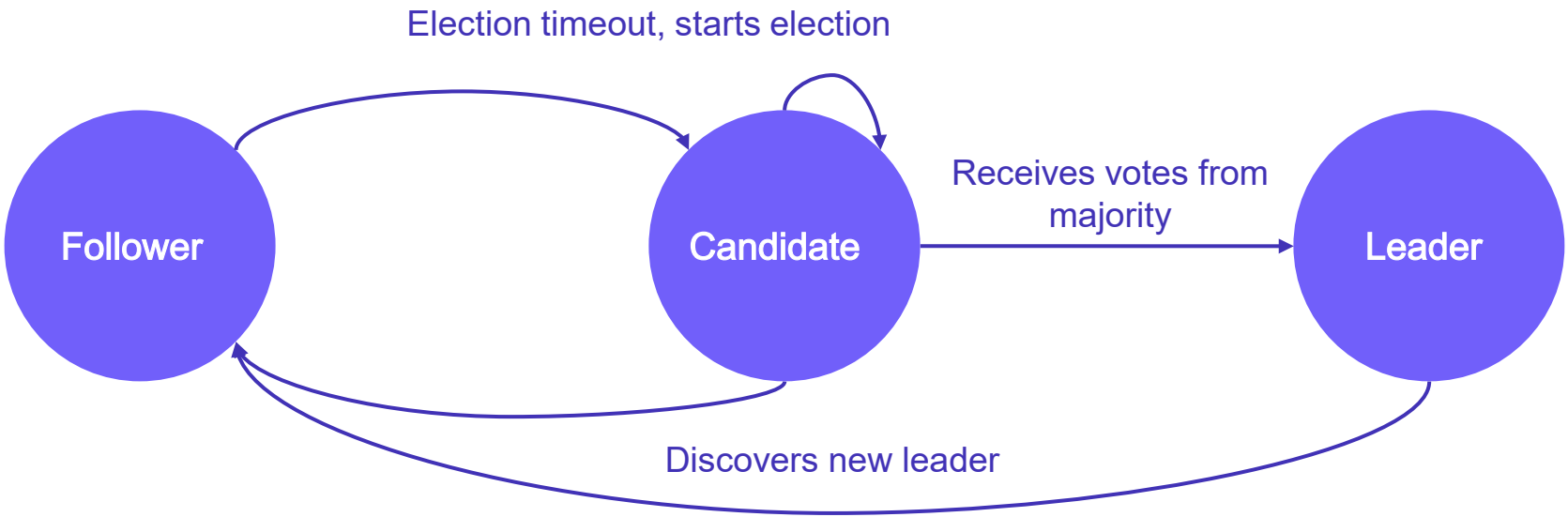
Core Features

- **Leader election** : determination of active nodes by election
- **Log replication** : consensus-based data storage. Mainly used for state replication
- **Inspector** : a tool for monitoring the status of clusters.



Leader Election

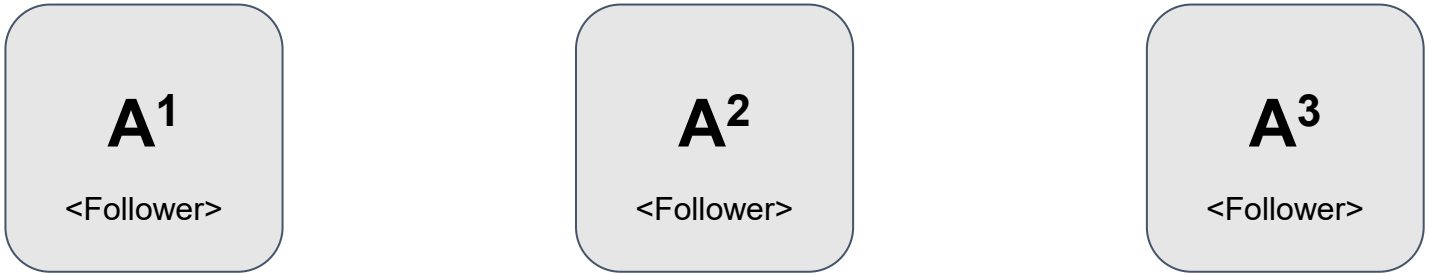
All nodes have one of the following states: **'Follower', 'Candidate'** , or **'Leader'**



<State Machine>

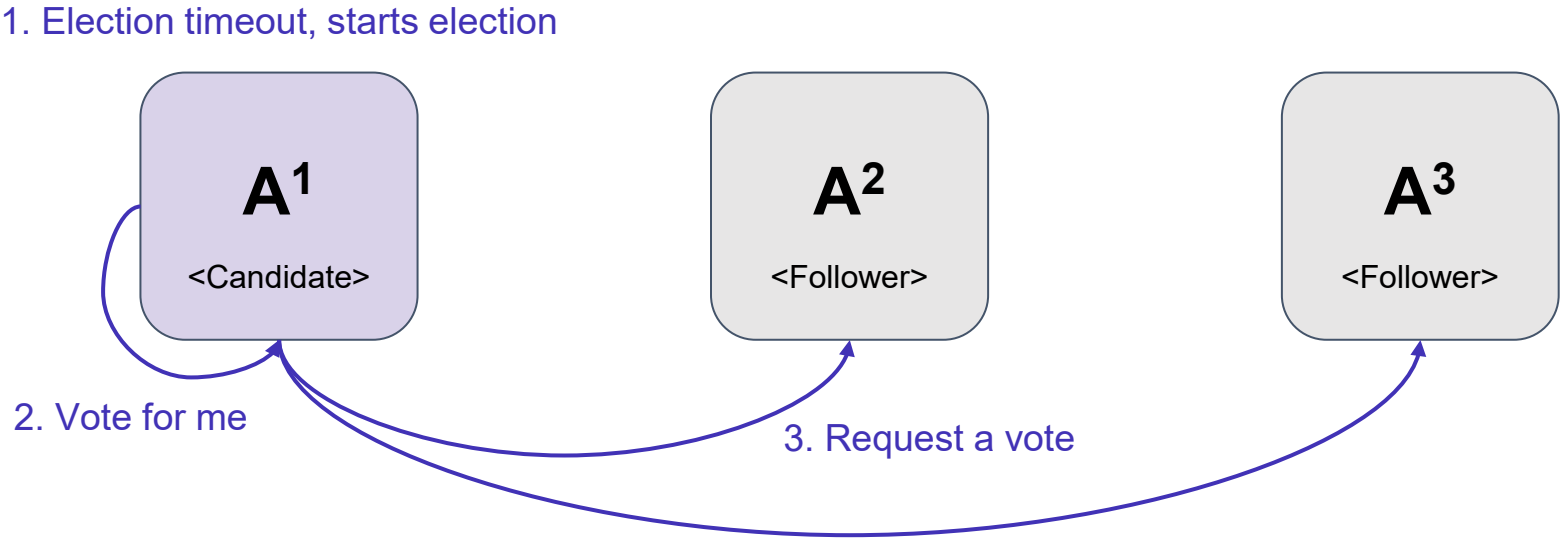
Leader Election

All nodes start in ‘ **Follower**’ state



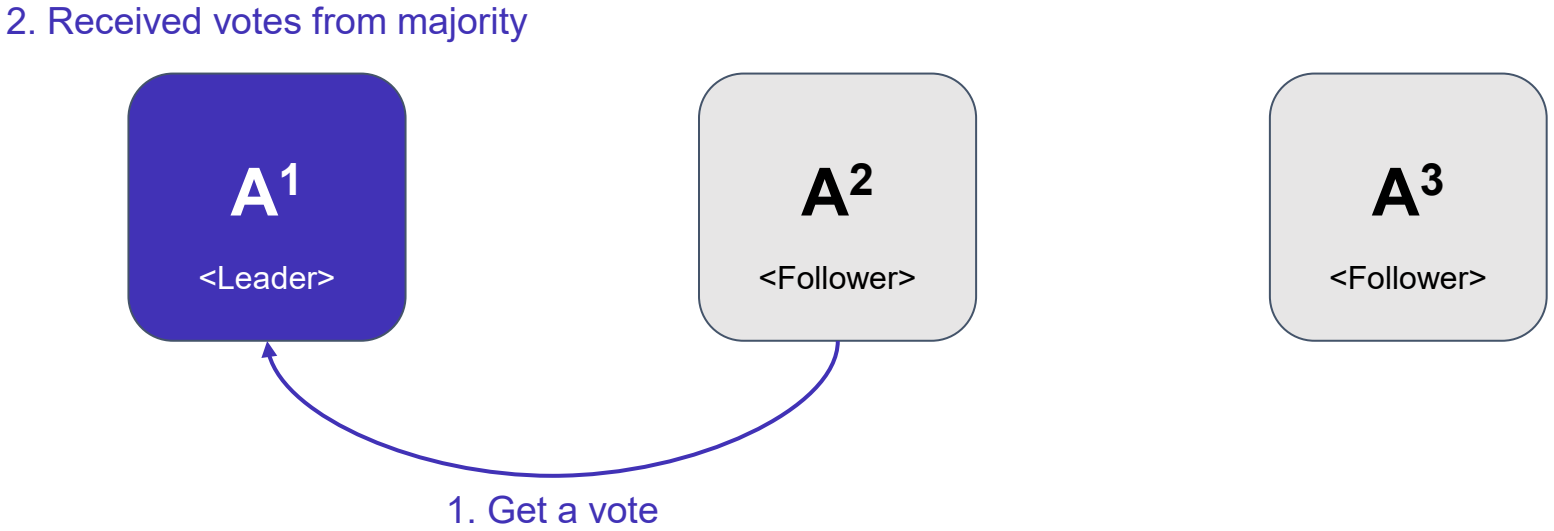
Leader Election

If a **'Follower'** does not receive a **'Leader'** heartbeat for a certain period of time, it is changed to **'Candidate'** and an election is held



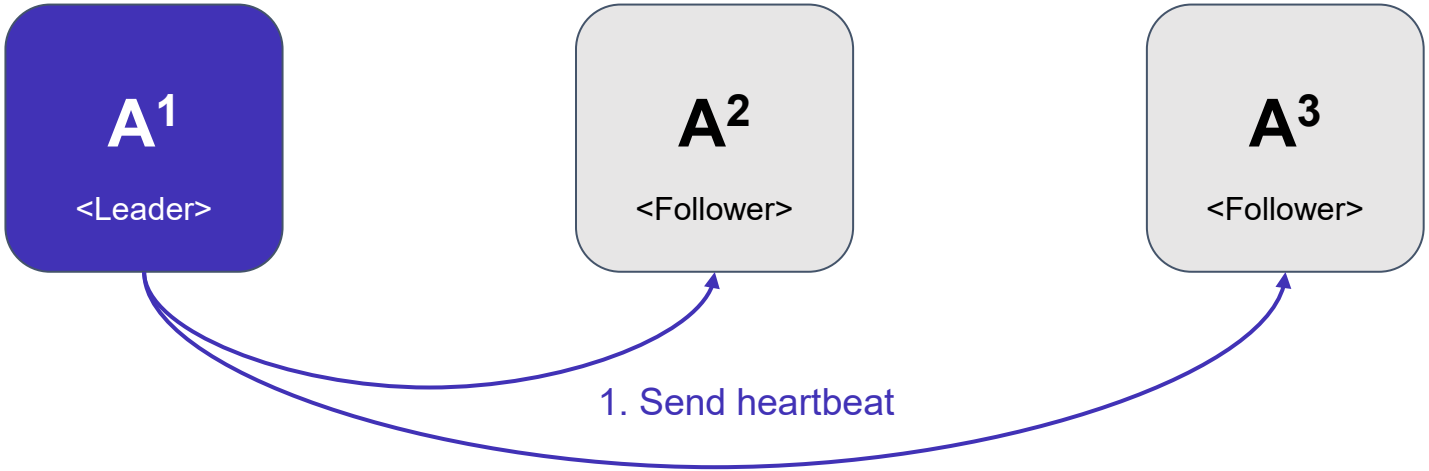
Leader Election

When a **‘Candidate’** receives a majority of the votes, it becomes the **‘Leader’** .



Leader Election

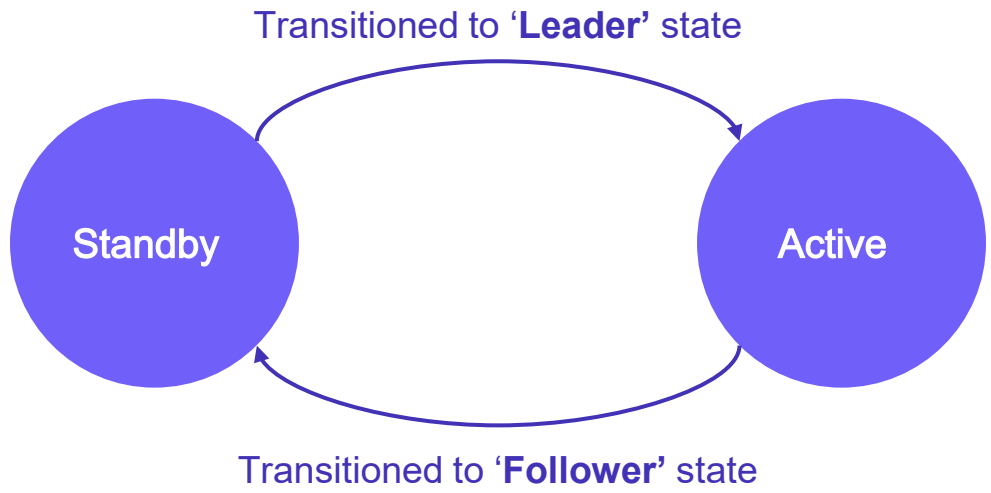
The **'Leader'** periodically sends heartbeats to prevent elections for new leaders.



Leader Election

The complex leader election process is all handled within the FOROS framework.

Developers only need to consider ‘ **Standby** ’ and ‘**Active** ’ states.



Filter the messages below :

- Published topics
- Received service requests

Leader Election : How to Use

Simple! Use *ClusterNode* class instead of *rclcpp ::Node*.

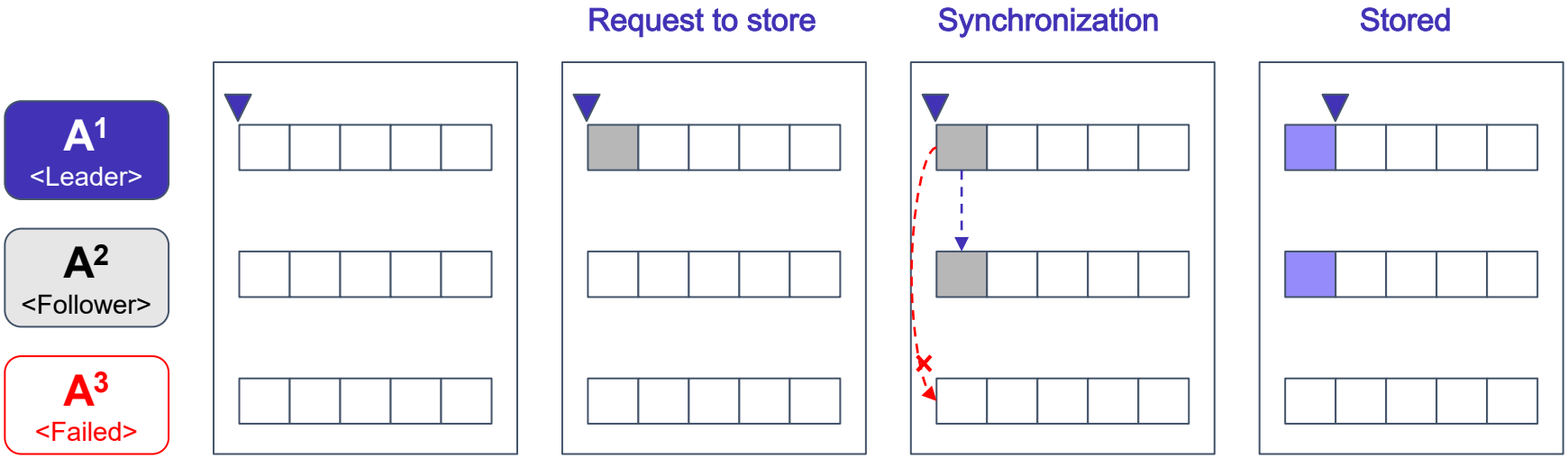
```
auto node = akit::failover::foros::ClusterNode::make_shared(
    "Test_cluster",                          // Cluster Name
    0,                                       // Node ID
    std::initializer_list<uint32_t>{0, 1, 2}  // Node IDs in the given cluster
);
```

Register state transition callbacks using *register_on_activated* , *register_on_standby*

```
node->register_on_activated([&]() { RCLCPP_INFO(logger, "activated"); });
node->register_on_standby([&]() { RCLCPP_INFO(logger, "standby"); });
```

Log Replication

When the ‘**Leader**’ requests to store data, it requests data synchronization from other nodes and succeeds when more than half of the nodes are synchronized



Log Replication : How to Use

Use *commit_command* to request to store data

```
node->commit_command(
   akit::failover::foros::Command::make_shared(std::initializer_list<uint8_t>{
       1}),
   [&](akit::failover::foros::CommandCommitResponseSharedFuture
           response_future) {
     auto response = response_future.get();
     if (response->result() == true) {
       RCLCPP_INFO(logger, "commit completed");
     } else {
       RCLCPP_ERROR(logger, "commit failed");
     }
   });
```

Log Replication : How to Use

Use *get_commands_size* , *get_command* to get stored data.

```
int len = node->get_commands_size();
auto command = get_command(len - 1);
```

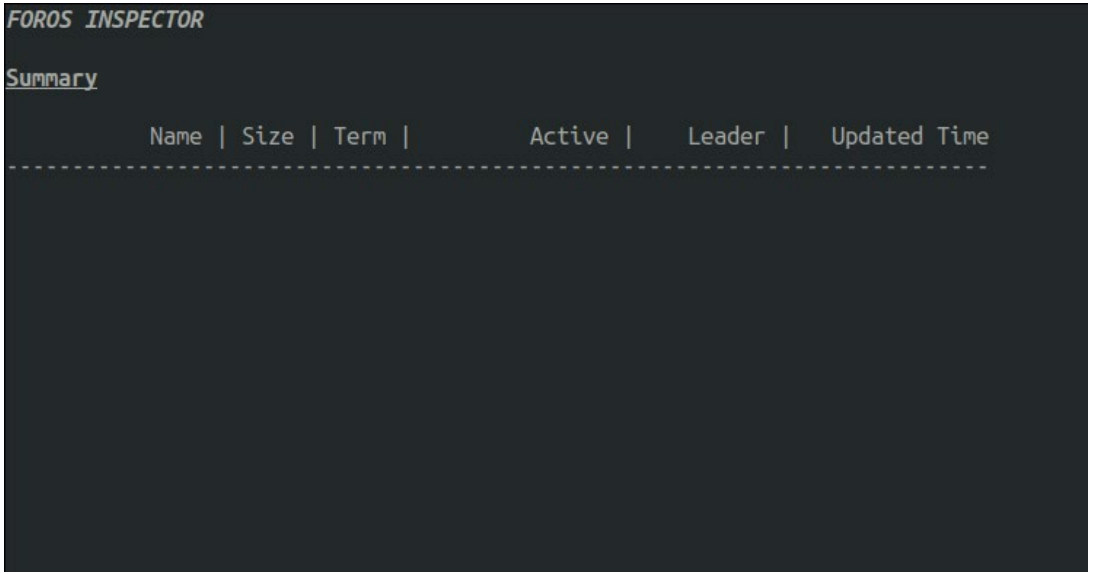
Use *register_on_committed* , *register_on_reverted* to register commit/revert callback.

```
node->register_on_committed(
   [&](int64_t id, akit::failover::foros::Command::SharedPtr command) {
     RCLCPP_INFO(logger, "command commited : %ld, %d", id, command->data()[0]);
   });

node->register_on_reverted([&](int64_t id) {
 RCLCPP_INFO(logger, "command reverted until : %ld", id);
});
```

Inspector

Visualize active cluster information and node information in the cluster with TUI



Summary : Cluster Information

Name	Cluster Name
Size	Cluster Size
Term	Election No.
Active	Active Node IDs
Leader	Leader ID

Details : Node Information

Node ID	Node ID
State	State
Term	Election No.
Voted For	ID
Data Size	Stored Data Size
Size	Cluster Size

Links

- FOROS Github : https://github.com/42dot/foros
- FOROS Wiki : https://github.com/42dot/foros/wiki
- RAFT : https://raft.github.io/
- RAFT Paper : https://raft.github.io/raft.pdf
- 42dot : https://42dot.ai/

Q&A