Distributed Sensor Network Simulation with Leader Election and Fault Tolerance

Project Team Member: Likitha Muralidhar
Project Team: 16

Project Overview:

- Simulate a network of embedded sensor nodes purely in software.
- Each node runs as a separate thread/process and communicates over sockets.
- Nodes generate random sensor readings (temperature, humidity, etc.) periodically.
- A base station node collects sensor readings.
- Implement leader election (Raft algorithm) to handle base station failures.
- Demonstrate distributed coordination, fault detection, and message passing.

Goals:

- Implement a distributed system on a small scale that mimics embedded sensor networks.
- Demonstrate leader election when the base station fails.
- Show fault tolerance and reliable communication between nodes.
- Provide clear evaluation to verify correctness of message passing and leader election.

Detailed Designed:

Architecture:

- Sensor nodes → generate data → send to base station
- Base station → collects data, maintains current leader info
- Nodes monitor base station health → trigger leader election if failure detected

Communication:

- Use TCP sockets for messaging between server nodes
- Heartbeat messages for failure detection

Leader Election:

- Implement Raft consensus for leader selection
- Node with highest ID becomes new base station when failure occurs

Assumptions:

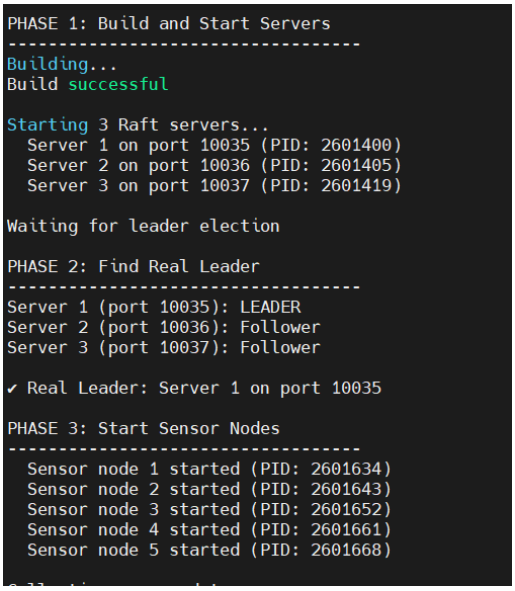
- Sensor nodes are homogeneous (same capabilities).
- Base station may fail randomly; nodes can detect failure within a timeout period.
- Network communication is reliable within simulated environment.
- Single cluster of nodes (number configurable).

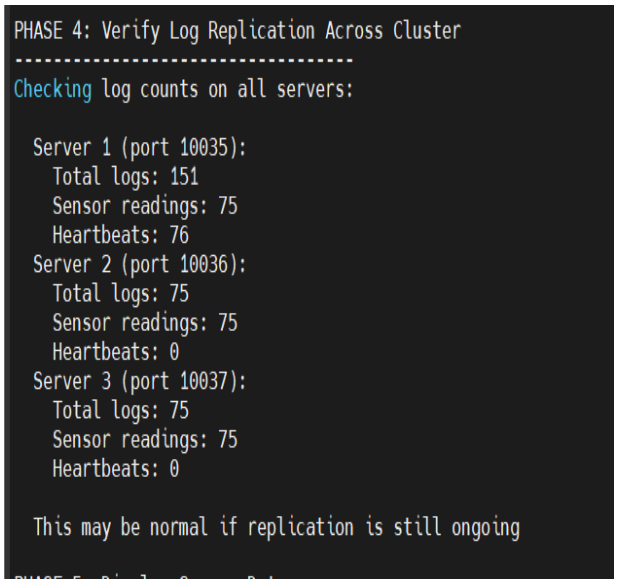
Modules:

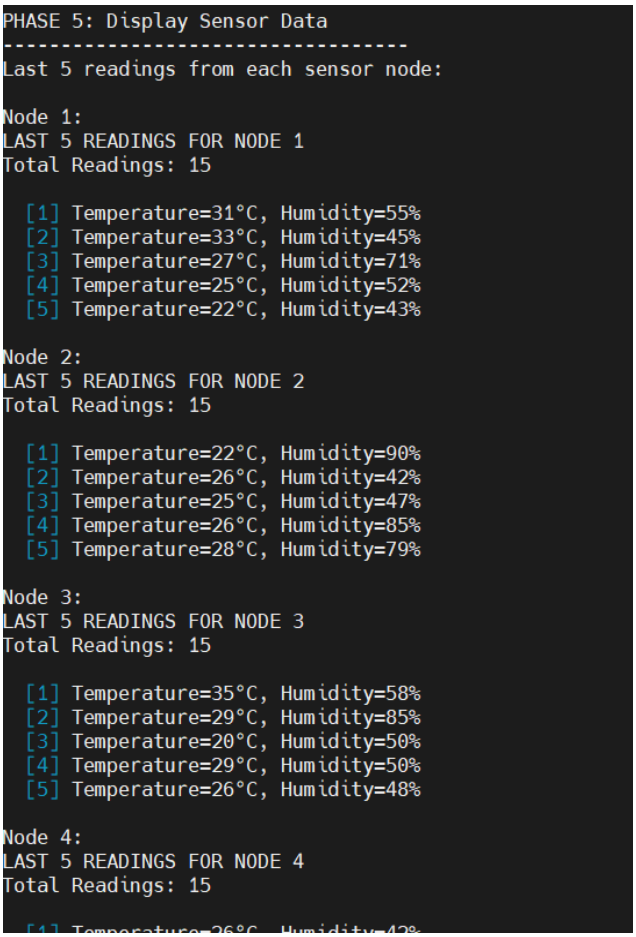
- Node Module: Generates sensor data, Sends readings to base station, Sends/receives heartbeat messages
- Base Station Module: Collects sensor readings, Maintains node status, Detects failures
- Leader Election Module: Handles leader election when base station fails, Implements Raft logic
- Communication Module: Handles socket message passing, Provides heartbeat and sensor reading channels
- Query Module: Inspect the latest sensor readings generated by each node. Validate that RAFT servers hold consistent replicated state

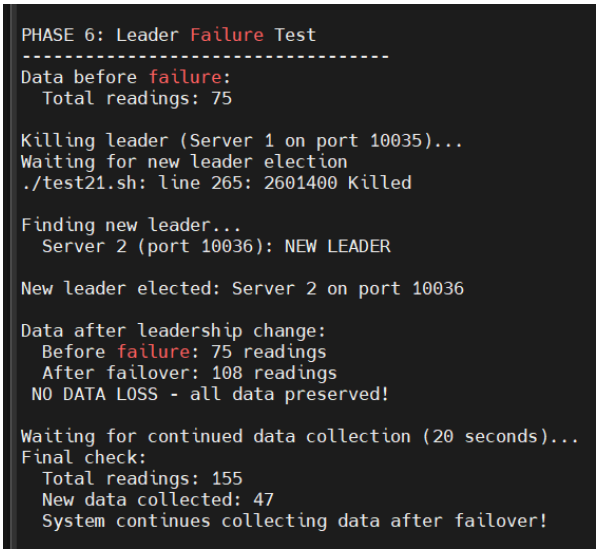
Evaluation:

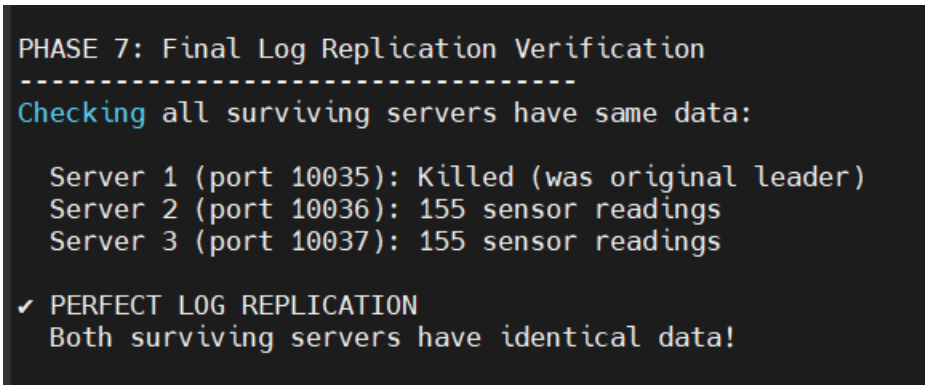
The system was evaluated using both manual testing procedures and an automated end-to-end test script. Manual evaluation allowed fine-grained inspection of leader behavior, log replication, and state-machine updates, while the automated script executed the full workflow—including server startup, sensor node simulation, leader failover, and log verification—in a consistent and repeatable manner. Using both methods ensured a comprehensive assessment of system correctness, fault tolerance, and overall stability.











1. Log Replication Prior to Leader Failure

The system was evaluated for its ability to consistently replicate sensor data across all RAFT servers before any failure occurred. This verified that the leader correctly propagated incoming log entries to all followers and that the replicated state machine remained synchronized across the cluster.

2. Leader Failure Handling

A controlled leader crash was performed to assess the system's fault-tolerance and recovery behavior. The evaluation focused on how quickly the remaining servers detected the failure, initiated a new election, and selected a new leader to maintain system availability.

3. End-to-End Sensor Data Ingestion

Data messages generated by each sensor node were monitored to ensure that all HEARTBEAT and DATA entries were successfully appended to the leader's log, replicated to followers, and applied to the state machine without loss or corruption.

4. Log Consistency After Leader Change

Following the leader failure and election of a new leader, the complete log history on all surviving servers was examined. This step validated whether the new leader retained a consistent and correct log and whether followers maintained log matching properties defined by the RAFT protocol after the failover transition.

Unachieved / Limitations:

Limited sensor data: More sensor data could be added, apart from temperature and Humidity

Testing Instruction :

There are two ways

1. Run the test script , Complete Evaluation can be done
   ./test.sh
2. Manual confirmantion – Follow the below instructions
   Compile:

   make clean && make

   Start RAFT servers: (Multiple servers) (peer1 = IP:PORT)

   ./server <port> <peer1,peer2> <id>

   Start sensor nodes:

   ./node <leader_ip> <node_id><server ports>

   Query

   ./query <leader_ip><leader_port> 1 - Print the metrics

   ./query <leader_ip><leader_port> 2 1 - Prints the data received from Node 1 (only the last five)

Contribution:

Implemented RAFT election, replication, and state machine logic.

Debugged replication issues involving multi-token log entries.

Created experiment automation and evaluated system on cluster.

Analyzed results and documented system behavior under failure.