Sistemas Multinúcleo e Distribuídos
Mestrado em Engenharia Informática
Instituto Superior de Engenharia do Porto

Distributed Sensor Network System

Introduction

This project involves the development of a distributed sensor network where each sensor is represented by an Erlang Virtual Machine (that may run several processes), simulating periodic data collection (such as temperature, humidity, etc.). Sensors will communicate with a central server (another Erlang VM), which aggregates the data for further analysis. Some sensors may be far away of the central server and may use another intermediate sensor(s) to relay communication. The system's key features include its ability to detect sensor failures, adapt to those failures while maintaining functionality, and dynamically integrate new sensors. This project will not only simulate a real-world distributed system, since it is easy to deploy the developed software for example in Raspberry Pi SBC's, but also demonstrate Erlang's strengths in managing independent processes, handling errors, and maintaining system stability in case of failures.

System behavior

The system is designed to model the behavior of sensors in a distributed network, focusing on process communication, fault tolerance and scalability. The following aspects define how the system operates:

Independent Processes : Each sensor is implemented as an autonomous Erlang VM, which collects and transmits data at regular intervals.

Asynchronous Communication : The sensors communicate with the central server or other sensors asynchronously using message-passing, ensuring non-blocking interactions and smooth data transmission.

Data Aggregation : The central server gathers the data from the sensors and stores it for later analysis.

Scalability : The system must be designed to be scalable, meaning it can handle an increasing number of sensors without affecting overall performance, as each sensor operates independently.

Recovery and adaptation : The system should handle errors smoothly, minimizing their impact on the overall operation and ensuring continued functionality.

Error handling and Fault Tolerance

Fault tolerance is critical in distributed systems, and this project will explore Erlang's robust mechanisms for handling process failures and maintaining system functionality:

Failure Simulation : Sensors will be designed to simulate failures, such as stopping a process/VM. This feature allows the system to test its ability to manage errors and continue functioning.

Failure Detection : When a sensor fails, the central server or other sensors in the network should detect this failure and respond accordingly.

Self-Reconfiguration : After detecting a sensor failure, the system should automatically reconfigure itself, ensuring the remaining sensors continue to operate and send data without disrupting overall functionality.

Each sensor stores a list of nearby sensors that can be used to relay information to the central server if a direct connection is unavailable. In the event of an intermediate sensor failure, the system should be able to route the information through an alternative sensor, provided other available sensors exist.

Submission and presentation

No report is needed and only the submission of the client and server programs is necessary (deadline is 5^{th} of June). The presentation occurs from 9^{th} of June.

Grading

This assignment is to be solved individually and will assess the knowledge about distributed programming in the functional language Erlang. Grading criteria is described next.

Core Features (30%) : Ensure that sensors function as independent processes, communicate with the central server, and regularly send data.

Proper Data Storage (10%) : The server correctly stores data received from sensors using appropriate data structures for later analysis or access.

Failure Detection (25%) : Accurate detection of sensor failures, including logging, notifications, and any necessary status checks when a sensor stops responding.

Self-Reconfiguration (20%) : The system's ability to re-route data when a sensor fails, using alternative sensors for communication, ensuring continuous operation.

Modularity and Organization (15%) : Clear separation of concerns, use of well-structured functions or modules, and adherence to best coding practices for readability and maintainability.