Fault-Detection Using Heartbeat Tactic

This homework makes you familiar with design and implementations of architectural tactics. An important goal of this assignment is to emphasize that designing architecture is important but implementing the design, testing and maintaining the design decisions are critical.

Objectives:

You are to implement the "Heartbeat" tactic for fault detection. The implementation is to be a minimum prototype of the tactic rather than full implementation of a system. Even so, consider issues of "critical strength" implementation that could be applied to large systems. For example, multiple threads in a single JavaScript process is not appropriate. Use "industrial strength" technology and design choices that can be applied system-wide.

Please consider the following items:

1. **Select a key *software functionality* in your autonomous self-driving vehicle case study**. This functionality is needed for an availability quality attribute concrete scenario. For instance, a critical software functionality can be "Obstacle Detection". Software of a Self-Driving Car needs to actively monitor a module that implements this function to identify whether it is running or has crashed or is not responding. It cannot be a single-processor solution because a given processor may have failed, but failure is not an option for the system as a whole.

2. **Develop a critical process (with minimum functionality)**. Implement the main functionalities of your critical module. For instance, have a few methods implementing an obstacle detection feature. This needs to be an approximation of the functionality not a full implementation.

3. **Design a Non-deterministic failure in this process which makes it crash**. Please create a realistic issue that makes your process not being responsive. DO NOT use process.exit and similar commands.

4. **Implement Heartbeat to monitor the process**

5. Your heartbeat implementation should have all the required fault detection features.

Rule 1: Do not embed a failure in a static "if" statement. The failure must be random and it must cause the process to crash. Avoid making the process simply sleep.

Rule 2: Implement send/receive/monitoring functions on **different processes** and preferably different processors on a network.

Deliverables:

- UML Class and Sequence diagrams illustrating tactic behavior.
- Compilable and runnable source code and executables.

- Read me file including guidance on how to compile and run the code and a list of language libraries or frameworks used.
- Additional documentation as necessary.