Simulation Framework for QoI Characterization of Sensor Networks in the Presence of Faults

Marcin Szczodrak[+], Sadaf Zahedi[*], Ping Ji[+], Dinkar Mylaraswamy[**], Mani Srivastava[*] and Robert Young[***]

[+]City University of New York, NY USA, *mszczodrak@gmail.com, pji@jjay.cuny.edu*
[*]University of California, Los Angeles, CA USA, {*szahedi, mbs*}*@ee.ucla.edu*
[**]Honeywell, MN USA *dinkar.a.mylaraswamy@honeywell.com*
[***]MoD, United Kingdom *riyoung@dstl.gov.uk*

***Abstract*—The quality of information delivered by a sensor network depends heavily on the integrity of data produced by the sensor nodes themselves. As such, the ability to model a sensor networking application scenario in the presence of faults that affect sensor data integrity is quite crucial. In this paper we describe a sensor network simulation framework that provides high fidelity modeling of sensor faults, and enables users to study end-to-end quality of information. Our sensor fault simulator is based on the popular, open-source simulator OMNet++ and the associated Castalia package for modeling basic sensor network behaviors. The framework provides the ability to model various elements of a complete application scenario including node deployment, physical process, sensor faults, fault detection, and data fusion. This paper will describe the architecture of the simulation platform, the novel sensor fault model for modeling a variety of fault types and progressive failure behaviors in a parameterized fashion, and the enhancements to Castalia with integrated functionalities for fault detection.**

I. Introduction

A key goal of the NIS ITA is to develop a quantifiable description of the quality of sensor-obtained information in support of decision making. Fundamental to this is an ability to understand the impact of faults, miscalibrations, environmental effects and other problems that afflict the integrity of data returned by sensors upon the overall Quality of Information (QoI) yielded by the sensor networks after fusing data from different sensors. In order to gain such insight, we developed a Sensor Fault Simulator (SFSim) with the ability to model the impact of faulty sensor behavior on the Quality of Information (QoI) for sensor networks. Our goal with this tool is to let the user explore the performance and QoI in an application scenario (i) under ideal conditions, (ii) in the presence of various realistic sensor faults, and (iii) in the presence of fault detection and repair processing.

To meet the goal of SFSim to inject faults and study their impact on application QoI, we decided to use as the foundation Castalia, an open-source sensor network simulation package [2]. Castalia in turn sits on top of OMNet++ [1], an open-source free-for-research discrete event network simulator. Castalia models the three components of a sensor network, namely: wireless networking, physical world sensing, and sensor nodes. Nodes in Castalia interact with a wireless channel via a protocol stack, and with one or more physical process, one for each sensing modality. Furthermore, the platform characteristics are modeled via a Resource Manager, which keeps track of relevant hardware state such as battery, memory, CPU, and time. A user interested in modeling an application scenario would draw upon a library of network protocol, wireless channel, and physical process models, and write a customized application module that would sense, compute, and communicate. While Castalia is adequate for modeling communication aspects of sensor networks, it lacks the ability to model the complex physical processes that underlie the application scenarios to be considered within the ITA. Before this work Castalia had no provision for modeling application scenarios in the presence of sensor faults, nor any support for fault detection and data repair functions within the sensing module.

II. Design of the Sensor Fault Simulator - SFSim

There were two possible ways of implementing the functionality we desired within SFSim. One approach would be to incorporate all the functionality in Castalia itself: including descriptions of new physical processes, models of faults, and fault management framework. The problem with this approach is the enormous amount of changes needed in Castalia, and the difficulty of implementing sophisticated mathematical models of sensors within Castalias C++ environment. Instead, the alternative approach that we adopted was to keep physical process and fault models outside Castalia in external tools written using a suitable package (e.g. Matlab, Octave, R), and generating trace files for play out by Castalia. Adopting the second approach allows us to use the best tools for each subcomponent and has greatly simplified the development of the overall simulation environment.

Figure 1 shows the simulation framework that we are creating, a prototype version of which is functional. The user describes the various aspects of an application scenario using simple text files, which are processed by a Matlab-based Scenario Generation Tool. The aspects of the scenario that this tool takes as input include:

1. Sources: this input describes the sources that generate physical signals to be detected by the sensors. For each source both a trajectory in space, and temporal evolution of various physical signals emitted by it are described. A way-point model has been instigated.

2. Sensors: this input describes the location of the various sensor nodes, the signal modalities it can detect, and the minimum detection threshold for each modality. Currently sensors are assumed omni-directional, although we anticipate to model directionally sensitive sensors in the future.

3. Terrain: this input describes the characteristic of the terrain in terms of how the physical signals of a particular modality propagate in space over time. Currently we support

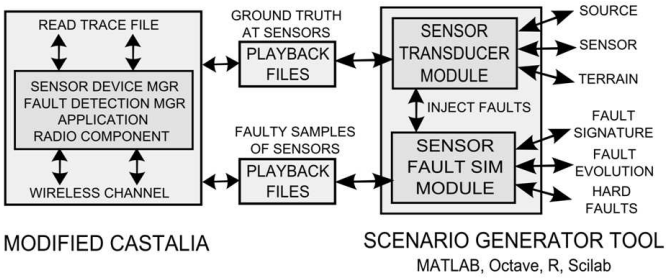

Fig. 1. Scenario Simulation Environment with Fault Injection

a propagation model with signal strength decaying as inverse of a distance power.

4. Fault Signature: this input specifies parameters of a generic statistical fault model that we have designed to capture many different types of faults encountered in real life.

5. Fault Evolution: this input specifies how the faults evolve in time, using a progressive Markov model.

6. Hard Faults: this input specifies hard faults that affect a sensing modality.

Using these inputs, the external tool generates trace files that list for each sensor node the temporal evolution of the signal detected at the location of the sensor node. Two types of trace files can be generated. The first have no faults injected and the traces represent the ground truth at the sensors based purely on the measurement process. The second have faults injected according to the fault model specified.

III. Simulation Implementation

For simulating the impact of sensor faults upon network behaviour, we have developed two key enhancements to Castalia. The first is a special physical process that reads the sensor value trace files produced by our scenario generator tool. To do this we have created a ReadTraceFilePhysicalProcess module, which loads sensor traces generated by the external scenario generator tool to Castalia. The traces taken to Castalia may include only *ground truth* sensor values or contain various *faults* drawn from the library of fault models that we designed in the scenario generator. By this, we have assumed that sensor faults are independent with the network application.

The second enhancement is a configurable fault detector module that sits between the Sensor Device Manager and the Application Module. This annotates and transforms the sensor value time series according to user-specified fault detection and fault repair methods. Over time we expect a rich library of univariate sensor fault detection and fault repair algorithms for the user to select from. At the present time we have implemented simple algorithms including the basic functions such as generating moving average values for smoothing.

We also implement a two-tier fault detection framework within Castalia, in which *local sensor nodes* (i.e., *local tier*) are associated with *sink nodes* (i.e., *global tier*) where the sensor data and fault information from local nodes are processed at a central fusion point. Figure 2 depicts the fusion node linkage intended within the two-tier framework.

We have modeled simple applications that utilize the two-tier structure implemented in Castalia, in which we assume a

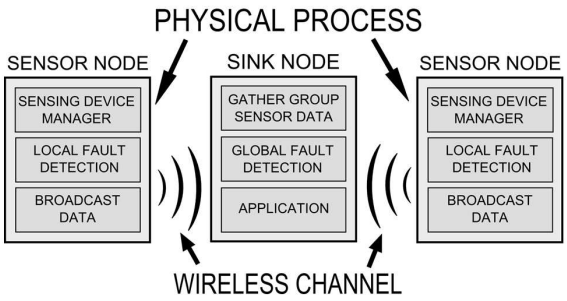

Fig. 2. Coordination between Two Tiers of Sensor Nodes

battle-ground scenario where acoustic and thermal sensors are deployed. The Application module of local nodes first receives data passed from the Local Fault Detection manager, which may or may not have applied a local fault detection method, and transmits this data to the sink nodes. The sink nodes then aggregate the gathered data, apply Global-information based Fault Detection, and cleanse the data as necessary. The scenarios that we have explored include: 1. Computing the average and maximum values of a group of sensors; and 2. Detecting if a transient event has occurred, and then determine the location of the event. The following pseudo-code illustrates the procedures followed by the sink nodes.

```
1. Data Aggregation
2. Fault Detection
3. Tag faulty data points as suspicious
4. Switch (Application Function)
5.   Case1: Get Average
6.      Compute Avg of non-suspicious data
7.   Case2: Get Max value
8.      Compute Max of non-suspicious data
9.   Case3: Event Localization
10.     If event occur
11.        Locate event using all data points
```

The next step of this research is to develop complete scenario evaluations using the sensor fault simulator. We will evaluate the scenarios under three conditions: 1. none of the sensors are faulty, 2. some of the sensors are faulty but there is no fault checking and 3. fault detection modules are present in both local and global tiers to check for fault occurrence. These future studies will demonstrate the worth of fault detection and the results will be presented in a subsequent paper.

IV. Acknowledgment

Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defense and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of 0 or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] Omnet++ - discrete event simulation system. http://omnetpp.org.
[2] Online resource of castalia simulator. http://castalia.npc.nicta.com.au/.