## Title
SYS 4: Designing High Integrity Embedded Networked Sensing Systems



## Authors
Nithya Ramanathan
Laura Balazano
Saurabh Ganeriwal
et al.

## Publication Date
2006

# Designing High Integrity Embedded Networked Sensing Systems

**Nithya Ramanathan, Laura Balazano, Saurabh Ganeriwal, Ram Kumar, Shane Markstrum, Roy Shea, Deborah Estrin, Eddie Kohler, Rupak Majumdar, Todd Millstein, Mani Srivastava**

## Introduction: Data integrity definition

### Sensor Networks in the Presence of Failures

- **Avoiding Failures**
  Check for software faults before deployment and calibrate sensors appropriately
- **Detecting Failures**
  Equip the nodes with outlier and fault detection abilities
- **Remediating Failures**
  Help the user address potential failures or validate anomalous data in-field
- **Malicious vs Non-Malicious Failures**
  They share a set of similar failure modes
  This work focuses on non-malicious failures

### Data and Platform Integrity

- **Platform Integrity**
  *Static checking of source code* removes faults before system deployment
  *Runtime fault detection* provides continued support in deployed systems
- **Data Integrity**
  *Reputation-based framework* to provide resiliency against malicious or non-malicious manipulation of the sensors.
  *Rule-based online fault detection* provides actions for the users to take to fix both network and sensor faults or validate sensor data

## Platform/System Integrity: Rapid fault identification, prevention, and isolation

### Static Checking of Source Code

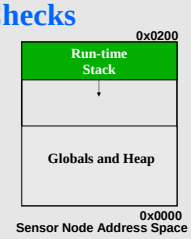
- Correct resource usage is difficult
  *Memory mismanagement* is a significant and serious source of errors on sensor nodes with no memory protection
- Develop simple and intuitive memory model
  1. Each block of *memory is under the control of exactly one program* at any time
  2. *Controlling program is responsible* for either tracking, freeing, or transferring ownership of the data
- Found *significant memory management errors* in both kernel and user SOS code using new analysis tool

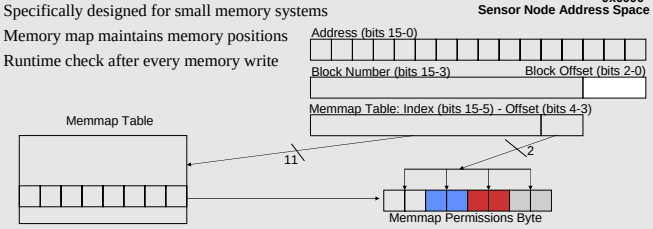
```
mod_op = (sos_module_op_t*) sys_msg_take_data(msg);
if(mod_op == NULL) return -ENOMEM;
if(mod_op->op == MODULE_OP_INSMOD) {
        ...
        ret = fetcher_request(sys_DFT_LOADER_PID, mod_op->mod_id,
                        mod_op->version, entohs(mod_op->size), msg->saddr);
        s->pend = mod_op;
        sys_led(LED_RED_TOGGLE);
        return SOS_OK;
}
return SOS_OK;
```

- Analysis is now *integrated into build system* of SOS

### Run-time Software Fault Checks

- Sensor networks use *single memory space* without hardware protection
  Memory space is shared by kernel, drivers and applications
  Buggy user application can *easily corrupt kernel*
- Create a single protection domain to protect the kernel
  Specifically designed for small memory systems
  Memory map maintains memory positions
  Runtime check after every memory write

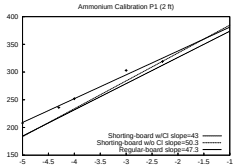


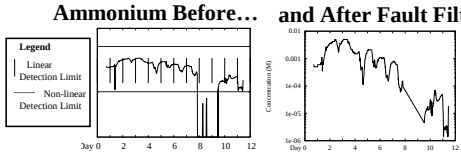

## Measurement and Data Integrity: Inconsistent measurements, faults, and calibration errors

### Sensor Calibration and Fault Remediation

- Sensor calibration should be done with the entire system to include bias introduced by hardware
- Sensor calibration should be done before and after the deployment.




**Ammonium Before… and After Fault Filtering**

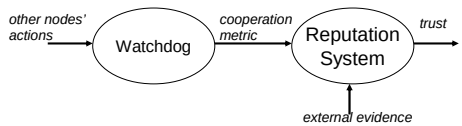



- Faulty data confounded the interesting phenomena.
- Based on data collected in Bangladesh, we designed a tool to detect calibration, orientation, bio-fouling, or sensor hardware faults in-field.
- This tool will suggest actions a user can take in the field to fix/validate problematic data. This increases the quantity of good data as compared to a post-facto removal of bad data.

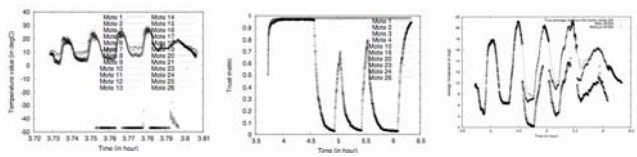
### Reputation based Framework for Sensor Nets

Data integrity is vulnerable to faulty and malicious behavior in all sensor network systems. In data collection systems, faults are indicators that sensor nodes are not providing useful information. In data fusion systems the consequences are more dire; the final value is easily affected by faulty values and the problems are no longer visibly obvious.

**Approach: Quantize the behavior of the node over time using a beta reputation system. Use this reputation as an inherent aspect in predicting the future behavior of the node.**



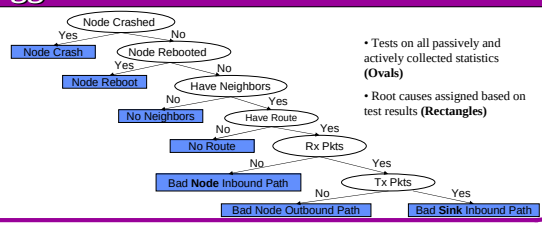

**Available as a middleware service on Motes in both SOS and TinyOS**



**Evaluation of RFSN over a temperature data set collected from James Reserve**

## Network Integrity: Sympathy as a sensor network debugger

- Sympathy aids the user in identifying and fixing network bugs in the field by:
- Collecting Metrics *such as neighbor table and number of packets received*
- Identifying Network Failures *using decision tree (right) to find root cause*
- Localize Failures *to determine why data is missing*




- Tests on all passively and actively collected statistics **(Ovals)**
- Root causes assigned based on test results **(Rectangles)**