Sensor Faults: Detection Methods and Prevalence in Real-World Datasets

ABHISHEK B. SHARMA
Computer Science, University of Southern California, Los Angeles
LEANA GOLUBCHIK
Computer Science and Electrical Engineering-Systems,
University of Southern California, Los Angeles
and
RAMESH GOVINDAN
Computer Science, University of Southern California, Los Angeles

Various sensor network measurement studies have reported instances of transient faults in sensor readings. In this work, we seek to answer a simple question: How often are such faults observed in real deployments? We focus on three types of transient faults, caused by faulty sensor readings that appear abnormal. To understand the prevalence of such faults, we first explore and characterize four qualitatively different classes of fault detection methods. Rule-based methods leverage domain knowledge to develop heuristic rules for detecting and identifying faults. Estimation methods predict "normal" sensor behavior by leveraging sensor correlations, flagging anomalous sensor readings as faults. Time series analysis based methods start with an *a priori* model for sensor readings. A sensor measurement is compared against its predicted value computed using time series forecasting to determine if it is faulty. Learning-based methods infer a model for the "normal" sensor readings using training data, and then statistically detect and identify classes of faults.

We find that these four classes of methods sit at different points on the accuracy/robustness spectrum. Rule-based methods can be highly accurate, but their accuracy depends critically on the choice of parameters. Learning methods can be cumbersome to train, but can accurately detect and classify faults. Estimation methods are accurate, but cannot classify faults. Time series analysis based methods are more effective for detecting short duration faults than long duration ones, and incur more false positives than the other methods. We apply these techniques to four real-world sensor data sets and find that the prevalence of faults as well as their type varies with data sets. All four methods are qualitatively consistent in identifying sensor faults, lending credence to our observations. Our work is a first-step towards automated on-line fault detection and classification.

Categories and Subject Descriptors: B.8.1 [**Reliability, Testing and Fault Tolerance**]: Learning of models from data; C.4 [**Performance of Systems**]: Modeling Techniques, Reliability, availability, and serviceability

General Terms: Fault Detection
Additional Key Words and Phrases: Data Integrity, Statistical Techniques, Fault Prevalence, Sensor Networks

This research has been funded by the NSF DDDAS 0540420 grant. It has also been funded in part by the NSF Center for Embedded Networked Sensing Cooperative Agreement CCR-0120778.

1. INTRODUCTION

With the maturation of sensor network software, we are increasingly seeing longer-term deployments of wireless sensor networks in real mode settings. As a result, research attention is now turning towards drawing meaningful scientific inferences from the collected data [Tolle et al. 2005]. Before sensor networks can become effective replacements for existing scientific instruments, it is important to ensure the quality of the collected data. Already, several deployments have observed faulty sensor readings caused by incorrect hardware design or improper calibration, or by low battery levels [Ramanathan et al. 2006; Tolle et al. 2005; Werner-Allen et al. 2006].

Given these observations, and the realization that it will be impossible to always deploy a perfectly calibrated network of sensors, an important research direction for the future will be automated detection, classification, and root-cause analysis of sensor faults, as well as techniques that can automatically scrub collected sensor data to ensure high quality. A first step in this direction is an understanding of the prevalence of faulty sensor readings in existing real-world deployments. In this paper, we take such a step.

Sensor Data Faults. We start by focusing on a small set of sensor faults that have been observed in real deployments: single-sample spikes sensor readings (we call these SHORT faults, following [Ramanathan et al. 2006]), longer duration noisy readings (NOISE faults), and anomalous constant offset readings (CONSTANT faults).

The three fault types (SHORT faults, NOISE faults, and CONSTANT faults), that we focus on in this paper, cause the faulty sensor readings to deviate from the normal pattern exhibited by true (or non-faulty) sensor readings, and are derived from a "data centric" view of sensor faults [Ni et al. 2008]. These three fault types have been observed in several real-world deployments [Ramanathan et al. 2006; Tolle et al. 2005], and hence, it is important to understand their prevalence and develop automated techniques for detecting them. Given these three fault models, our paper makes the three contributions described below.

Before describing our contributions, we note that not all sensor data faults fall within the three fault categories considered in this paper. Ni et al. [Ni et al. 2008] provide examples of sensor data faults due to calibration errors that can persist during the entire deployment. For example, an *offset fault* due to calibration errors causes all the sensor readings to differ from the true value by a constant amount, but the sensor readings still exhibit normal patterns (e.g., a diurnal variation in case of ambient temperature).

Detection Methods. We first explore four qualitatively different techniques for detecting such faults from a trace of sensor readings. Our decision to consider four qualitatively different fault detection techniques is motivated by the following two factors. Firstly, our goal is to explore the space of fault detection techniques that are suitable for detecting the class of data faults – SHORT, NOISE, and CONSTANT – examined in this paper. Secondly, as one might expect, and as we shall see later in the paper, no single method is perfect for detecting the different types of faults we consider in this paper. Intuitively, then, it makes sense to explore the space of detection techniques to understand the trade-offs in detection accuracy versus the robustness to parameter choices and other design considerations. This is what we have attempted to do with respect to the methods that we have chosen, from among the existing general types of fault detection methods, and our choice of qualitatively different approaches exposes differences in the trade-offs.

All four methods follow a common framework for fault detection: they characterize the "normal" behavior of sensor readings, and identify significant deviations from this "nor-

mal" as faults. However, in order to facilitate a systematic exploration of the space of detection techniques, we choose/design these methods based on four different types/sources of information relevant for detecting the SHORT, NOISE and CONSTANT data faults. The four different classes of methods discussed in this paper are as follows.

—**Rule-based methods** leverage *domain knowledge about sensor readings* to develop heuristic rules/constraints that the sensor readings must satisfy.

—**Estimation methods** define"normal" sensor behavior by leveraging *spatial correlation in measurements at different sensors*.

—**Time series analysis based methods** leverage *temporal correlations in measurements collected by the same sensor* to estimate the parameters of an (a priori selected) model for these measurements. A sensor measurement is compared against its predicted value, computed using time series forecasting, to determine if it is faulty.

—**Learning-based methods** *infer a model for the normal and faulty sensor readings using training data*, and then statistically detect and identify classes of faults.

While all four methods are geared towards automated fault detection, our design is not fully automated. In particular, parameter selection (e.g., selecting the fault detection thresholds for Rule-based methods) using a combination of domain knowledge and heuristics (as summarized in Table I), requires manual intervention.

While our choice and our design of the four fault detection methods is targeted at SHORT, NOISE, and CONSTANT faults, we discuss extensions to the Estimation method, the Time series analysis based methods, and the learning-based methods that incorporate information from multiple sensors co-located on the same node (inter-sensor relationships) and/or information from sensors attached to different nodes (inter-node relationships) in Section 7. Leveraging these inter-node and inter-sensor relationships can be useful for detecting data faults not considered in this paper, for example, those caused by calibration errors [Ni et al. 2008].

Evaluation using injected faults. By artificially injecting faults of varying intensity into sensor data sets, we are able to study the detection performance of these methods. We find that these methods sit at different points on the accuracy/robustness spectrum. While rule-based methods can detect and classify faults, they can be sensitive to the choice of parameters. By contrast, the estimation method we study can tolerate errors in parameter choices (for example, errors in estimating the correlation between readings from different sensor nodes) but relies on spatial correlations and cannot classify faults. Our seasonal time series model based method exploits temporal correlations in sensor measurements to detect faults. It is more accurate and robust at detecting SHORT faults than longer duration NOISE faults, and incurs more false positives than the other methods. However, it can detect the onset of long duration NOISE and/or CONSTANT faults accurately; even in situations where all the sensor nodes suffer from faults simultaneously. Finally, our learning method (based on Hidden Markov Models) is cumbersome, partly because it requires training, but it can fairly accurately detect and classify faults. Furthermore, at low fault intensities, these techniques perform qualitatively differently: the learning method is able to detect more NOISE faults but with higher false positives, while the rule-based method detects more SHORT faults, with the estimation method's performance being intermediate. The time series forecasting based method is able to detect low intensity SHORT faults and short duration NOISE faults but incurs a high false positive rate. It has a low detection rate

for long duration NOISE faults. Motivated by the different performance of these methods, we also propose and evaluate hybrid detection techniques, which combine these methods in ways that can be used to reduce false positives or false negatives, whichever is more important for the application.

Evaluation on Real-World Datasets. Armed with this evaluation, we apply our detection methods (or, in some cases, a subset thereof) to four real-world data sets. The longest of our data sets spans six months, and the shortest spans one day. We examine the frequency of occurrence of faults in these real data sets, using a very simple metric: the fraction of faulty samples in a sensor trace. We find that faults are relatively infrequent: often, SHORT faults occur once in about two days in one of the data sets that we study, and NOISE faults are even less frequent. However, if a fault is not detected promptly, it can corrupt a significant fraction of samples – for one dataset, 15-35% of samples are affected by a combination of NOISE and CONSTANT faults across different nodes. We find no spatial or temporal correlation among faults. The different data sets exhibit different levels of faults: for example, in a six months-long dataset, less than 0.01% of the samples are affected by faults, while in another 3-month long dataset, close to 20% of the samples are affected. Finally, we find that our detection methods incur false positives and false negatives on these data sets, and hybrid methods are needed to eliminate one or the other.

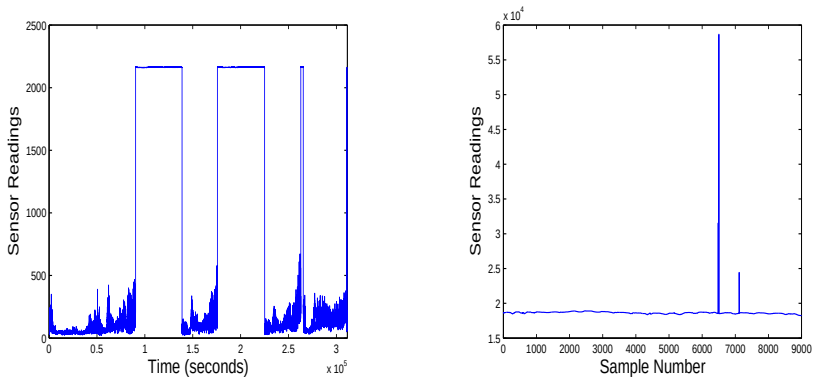
Our study informs the research on ensuring data quality. Even though we find that faults are relatively rare, they are not negligibly so, and careful attention needs to be paid to engineering the deployment and to analyzing the data. Furthermore, our detection methods could be used as part of an online fault diagnosis system, i.e., where corrective steps could be taken during the data collection process based on the diagnostic system's results.[1]

2. SENSOR FAULTS

In this section, we first visually depict some faults in sensor readings observed in real datasets. These examples are drawn from the same real-world datasets that we use to evaluate prevalence of sensor faults; we describe details about these datasets later in the paper. These examples give the reader visual intuition for the kinds of faults that occur in practice, and motivate the fault models we use in this paper.

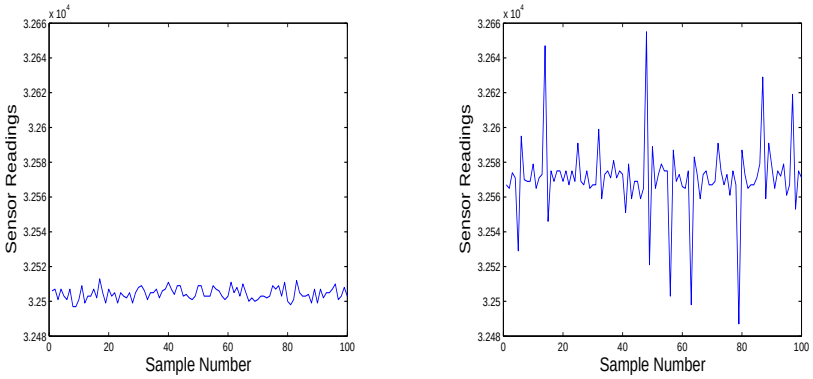
Figure 1(a) shows readings from a sensor reporting chlorophyll concentration measurements from a sensor network deployment on lake water. Due to faults in the analog-to-digital converter board the sensor starts reporting values $4-5$ times greater than the actual chlorophyll concentration. Similarly, in Figure 1(b), one of the samples reported by a humidity sensor is roughly 3 times the value of the rest of the samples, resulting in a noticeable spike in the plot. Finally, Figure 2 shows that the variance of the readings from an accelerometer attached to a MicaZ mote measuring ambient vibration increases when the voltage supplied to the accelerometer becomes low. In the absence of ground truth values (as is the case with the data shown in Figures 1 and 2), strictly speaking, the term fault

[1]**A note to the reviewer:** A preliminary version of this paper appeared in the Fourth Annual IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2007 [Sharma et al. 2007]. Apart from the material presented in the preliminary version, this manuscript describes and evaluates a time series forecasting based fault detection method, and analyzes a much larger dataset from the SensorScope deployment. The SensorScope dataset that we analyzed in [Sharma et al. 2007] consisted of readings collected over a month by 31 weather stations, whereas the SensorScope dataset analyzed in Section 5.1 [SensorScope 2006] consists of readings collected by 64 weather stations over 6 months.



(a) CONSTANT (b) SHORT

Fig. 1. Errors in sensor readings



(a) Sufficient voltage (6 volts) (b) Low voltage ($<$ 5 volts)

Fig. 2. NOISE fault: Increase in variance

refers to a deviation from the expected value. Hence, these data faults can also be thought of as *anomalies*.

The faults in sensor readings shown in these figures characterize the kind of faults we observed in the four datasets from wireless sensor network deployments that we analyze in this paper. We know of two other sensor network deployments [Tolle et al. 2005; Ramanathan et al. 2006] that have observed similar faults.

In this paper, we explore the following three fault models motivated by these examples:

(1) **CONSTANT**: The sensor reports a constant value for a large number of successive samples. The reported constant value is either very high or very low compared to the "normal" sensor readings (Figure 1(a)) and uncorrelated to the underlying physical phenomena.

(2) **SHORT**: A sharp change in the measured value between two successive data points (Figure 1(b)).

(3) **NOISE**: The variance of the sensor readings increases. Unlike SHORT faults that affect a single sample at a time, NOISE faults affect a number of successive samples

(see Figure 2).

SHORT and **NOISE** faults were first identified and characterized in [Ramanathan et al. 2006] but only for a single dataset. Ni et al. [Ni et al. 2008] categorize the three fault types defined above as representing the "data-centric" view of classifying faults, i.e. these fault types are defined in terms characteristics of the faulty data.

2.1 What causes sensor faults ?

While it is not always possible to ascertain the root cause for sensor faults, several system (hardware and software) faults have been known to result in sensor faults.

The typical hardware faults that have been observed to cause sensor faults include damaged sensors, short-circuited connections, low battery, and calibration errors. For the sensor faults shown in Figure 1(a), we were able to establish that they were caused due a fault in the analog-to-digital converter. Ramanathan et al. [Ramanathan et al. 2006] and Szewczyk et al. [Szewczyk et al. 2004] identified short circuit connections as the reason behind abnormally large or small sensor readings resembling SHORT or NOISE faults. Low battery voltage resulted in a combination of NOISE and CONSTANT faults at temperature sensors (see Figure 15) during the INTEL Lab, Berkeley deployment [INTEL 2004]).

A well-known root cause for sensor data faults is calibration problems [Ramanathan et al. 2006; Ni et al. 2008]. Calibration errors can corrupt the sensor measurements in different ways: (i) the measured value can differ from its true value by a constant amount (Offset fault), (ii) the rate of the measured data can differ from the true/expected rate (Gain fault), and (iii) the parameters associated with a sensor's original calibration formulas may change during a deployment (Drift fault). Calibration errors can affect all the samples collected during a deployment, and the faulty data may still exhibit normal patterns. For example, ambient temperature measurements affected by an Offset fault will still exhibit a diurnal pattern. Without the availability of ground truth values or a model for expected sensor behavior, detecting data faults due to calibration errors remains an open problem. In Section 7, we discuss extensions to our fault detection methods that can be used to automatically generate a model for expected sensor behavior by leveraging spatial correlation across sensor nodes. Bychkovskiy et al. [Bychkovskiy et al. 2003], and Balzano and Nowak [Balzano and Nowak 2007] exploit spatial correlation across sensor nodes to develop methods for online sensor calibration that can be used to recover from calibration errors during a deployment, once such an error is detected.

An example of software fault is given in [Ni et al. 2008] where Ni et al. identify instances of SHORT faults due to software errors during communication and data logging.

In this paper, our focus is on the prevalence of the SHORT, NOISE, and CONSTANT data faults and methods for detecting these faults. We do not attempt to precisely establish the *root cause* for these faults. The reader is referred to [Ni et al. 2008] for a detailed summary of known (system-level) root causes for sensor faults.

3. DETECTION METHODS

In this paper, we explore and characterize four qualitatively different detection methods for detecting SHORT, NOISE and CONSTANT faults. As discussed in Section 1, these methods leveraging different types/sources of information for fault detection. Rule-based methods – SHORT and NOISE rules – leverage *domain knowledge about sensor readings* to develop heuristic rules/constraints that the sensor readings must satisfy. The Lin-

ear Least-Squares Estimation (LLSE) based method defines "normal" sensor behavior by leveraging *spatial correlation in measurements at different sensors*. The autoregressive intergrated moving average (ARIMA) model based time series analysis methods leverage *temporal correlations in measurements collected by the same sensor* to estimate the parameters of an (a priori selected) model for these measurements. A sensor measurement is compared against its predicted value, computed using time series forecasting, to determine if it is faulty. Finally, the learning-based hidden markov model (HMM) method *infers a model for the normal and faulty sensor readings using training data*, and then statistically detect and identify classes of faults. These methods are described in detail in the rest of this section. Table I provides a summary of these methods (along with their variations and parameters).

3.1 Rule-based (Heuristic) Methods

Our first class of detection methods uses two intuitive heuristics for detecting and identifying the fault types described in Section 2.

NOISE Rule: Compute the standard deviation of sample readings within a window N. If it is above a certain threshold, the samples are corrupted by the NOISE fault.
To detect CONSTANT faults, we use a slightly modified NOISE rule where we classify the samples as corrupted by CONSTANT faults if the standard deviation is zero. The window size N can be in terms of time or number of samples. Clearly, the performance of this rule depends on the window size N and the threshold. Determining the best value for the window size N requires domain knowledge, in particular, a good understanding of the *normal* sensor readings. We discuss a heuristic for selecting the threshold value later in this section.

SHORT Rule: Compute the rate of change of the physical phenomenon being sensed (temperature, humidity etc.) between two successive samples. If the rate of change is above a threshold, it is an instance of a SHORT fault.

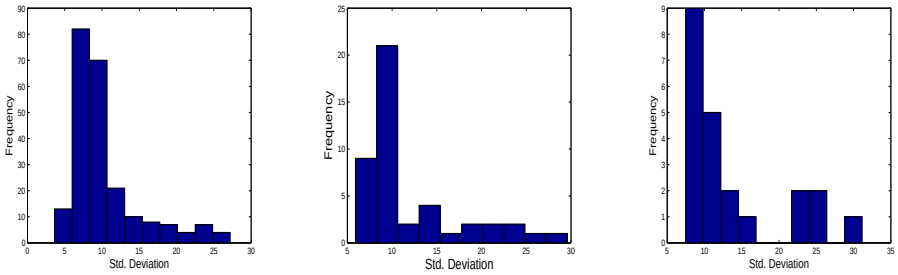
For well-understood physical phenomena like temperature, humidity etc., the thresholds for the NOISE and SHORT rules can be set based on domain knowledge. For example, [Ramanathan et al. 2006] uses feedback from domain scientists to set a threshold on the rate of change of chemical concentration in soil.

For automated threshold selection, [Ramanathan et al. 2006] proposes the following technique:

— **Histogram based method**: Divide the time series of sensor readings into groups of N samples. Plot the histogram of the standard deviations or the rate of change observed for these groups of N samples. Select one of the modes of the histogram as the threshold.

Clearly, if the histogram does not have a (distinct) mode, then the histogram based method will fail to select a good threshold. For the NOISE rule, the Histogram method for automated threshold selection will be most effective when, in the absence of faults, the histogram of standard deviations is uni-modal and sensor faults affect the measured values in such a way that the histogram becomes bi-modal. However, this approach is sensitive to the choice of N; the number of modes in the histogram of standard deviations depends on N. Figure 3 shows the effect of N on the number of modes in the histogram computed for sensor measurements taken from a real-world deployment [NAMOS 2005]. The measurements do not contain a sensor fault, but choosing a large value for N (500 or 1000) can

result in the Histogram method selecting an incorrect threshold. For example, choosing $N = 1000$ gives a multi-modal histogram (Figure 3.c); this would result in false positives, if we select one of the two modes greater than 20 as the fault detection threshold. As stated earlier, selecting the correct value for the parameter N requires a good understanding of the *normal* sensor readings. In particular, a domain expert would have to suggest that $N = 1000$ in our previous example was an unrealistic choice of parameter. In practice, one should also try a range of values for N to ensure that the samples flagged as faulty are not just an artefact of the value selected for N.



(a) N=100 (b) N=500 (c) N=1000

Fig. 3. Histogram Shape

3.2 An Estimation-Based Method

Is there a method that perhaps requires less domain knowledge in setting parameters? For physical phenomena like ambient temperature, light etc. that exhibit a diurnal pattern, statistical correlations between sensor measurements can be exploited to generate estimates for the sensed phenomenon based on the measurements of the same phenomenon at other sensors. Regardless of the cause of the statistical correlation, we can exploit the observed correlation in a reasonably dense sensor network deployment to detect anomalous sensor readings.

More concretely, suppose the temperature values reported by sensors s_1 and s_2 are correlated. Let $\hat{t_1}(t_2)$ be the estimate of temperature at s_1 based on the temperature t_2 reported by s_2. Let t_1 be the actual temperature value reported by s_1. If $|t_1 - \hat{t_1}| > \delta$, for some threshold δ, we classify the reported reading t_1 as erroneous. If the estimation technique is robust, in the absence of faults, the estimate error ($|t_1 - \hat{t_1}|$) would be small whereas a fault of the type SHORT or CONSTANT would cause the reported value to differ significantly from the estimate.

In this paper we consider the Linear Least-Squares Estimation (LLSE) method [Kailath 1977] as the estimation technique of choice. In scalar form, the LLSE equation is

$$\hat{t_1}(t_2) = m_{t_1} + \frac{\lambda_{t_1 t_2}}{\lambda_{t_2}}(t_2 - m_{t_2}) \tag{1}$$

where m_{t_1} and m_{t_2} are the average temperatures at s_1 and s_2, respectively. $\lambda_{t_1 t_2}$ is the covariance between the measurements reported by s_1 and s_2, and λ_{t_2} is the variance of the measurements reported by s_2.

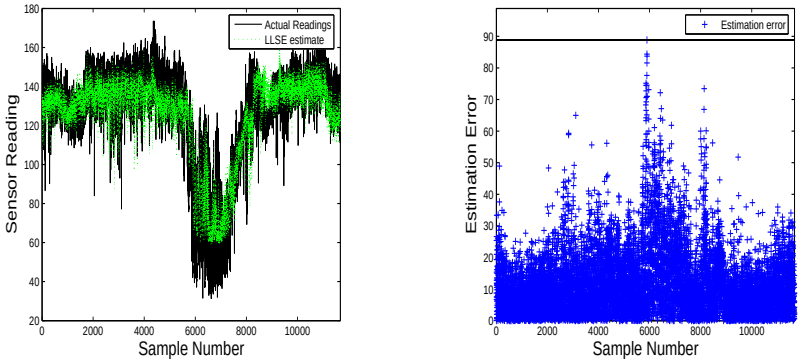
In the real-world, the value t_2 might itself be faulty. In such situations, we can estimate $\hat{t_1}$ based on measurements at more than one sensor using the LLSE equations for the vector case (a straight forward and well-known generalization of the scalar form equation).

In general, the information needed for applying the LLSE method may not be available a priori. In applying the LLSE method to a real-world dataset, we divide the dataset into a training set and a test set. We compute the mean and variance of sensor measurements, and the covariance between sensor measurements based on the training dataset and use them to detect faulty samples in the test dataset. This involves an assumption that, in the absence of faults or external perturbations, the physical phenomenon being sensed does not change dramatically between the time when the training and test samples were collected. We found this assumption to hold for many of the datasets we analyzed.

Threshold for fault detection: We set the threshold δ used for detecting faulty samples based on the LLSE estimation error for the training dataset. We use the following two heuristics for determining δ:

— **Maximum Error**: If the training data has no faulty samples, we can set δ to be the maximum estimation error for the training dataset, i.e. $\delta = \max\{|t_1 - \hat{t_1}| : t_1 \in TS\}$ where TS is set of all samples in the training dataset.

— **Confidence Limit**: In practice, the training dataset will have faults. If we can reasonably estimate, e.g., from historical information, the fraction of faulty samples in the training dataset, (say) p%, we can set δ to be the upper confidence limit of the $(1-p)$% confidence interval for the LLSE estimation errors on the training dataset.



(a) LLSE Estimate (b) Estimation Error

Fig. 4. LLSE on NAMOS dataset

Figure 4 is a visual demonstration of the feasibility of LLSE, derived from one of our datasets. It compares the LLSE estimate of sensor readings at a single node A based on the measurements reported by a neighboring node B, with the actual readings at A. The horizontal line in Figure 4(b) represents the threshold δ using the *Maximum Error* criterion. The actual sensor data had no SHORT faults and the LLSE method classified only one out of $11,678$ samples as faulty. We return to a more detailed evaluation of LLSE-based fault detection in later sections.

Finally, although we have described an estimation-based method that leverages spatial correlations, this method can equally well be applied by only leveraging temporal correlations at a single node. By extracting correlations induced by diurnal variations at a node, it might be possible to estimate readings, and thereby detect faults, at that same node. The method described next presents one approach for exploiting these temporal correlations for fault detection.

3.3 A Time Series analysis based Method

Physical phenomena such as temperature, ambient light, etc. exhibit a diurnal pattern. If these phenomena are measured periodically for a long time interval (as is the case with several sensor network deployments), the resulting time series of measurements by a sensor captures the diurnal pattern as well as other (shorter) time scale temporal correlations. These temporal correlations can be exploited to construct a model for the sensor measurements using time series analysis. Time series analysis is a popular technique for analyzing periodically collected data. For example, it is used by businesses to model and forecast demand for electricity, sale of airline tickets, etc. that exhibit temporal correlations like a diurnal and/or a seasonal pattern [Chatfield 2000].

In this paper, we use a multiplicative $(0,1,1)\mathrm{x}(0,1,1)_s$ seasonal ARIMA time series model for fault detection, where the parameter s captures the *periodic behavior* in the sensor measurement time series; for example, temperature measurements exhibit similarities with period $s = 24$ hours. The multiplicative seasonal model is widely used for modeling and forecasting of time series with periodicity [Box et al. 1994]. It can be written explicitly as,

$$z_t = z_{t-1} + z_{t-s} - z_{t-s-1} + a_t - \theta a_{t-1} - \Theta a_{t-s} + \theta\Theta a_{t-s-1} \tag{2}$$

where z_t is the sensor reading and a_t is a sample drawn from a white noise process at time t. Equation (2) shows how the model accounts for periodicity: z_t depends not only on the measurement at time $t-1$ but also on measurements made s time samples in the past, namely at times $t-s$ and $t-s-1$. For more details on seasonal models, we refer the interested reader to [Box et al. 1994].

Fault detection using forecasting: We used the implementation of maximum likelihood (ML) computational method (Chapter 7, [Box et al. 1994]) in SAS [SAS], a commonly used software package for time series analysis, to estimate the two parameters, θ and Θ, of the model using training data. To detect faults in a sensor measurement time series, we first forecast the sensor measurement at time t based on our model (using standard time series *forecasting* tools available in SAS). We then compute the difference between actual sensor measurement at time t and its predicted value, and flag the measurement as faulty if this difference is above a threshold δ.

We used two different durations of forecasting for fault detection.

— **One-step ahead**: We forecast the sensor measurement for time $t+1$, $\hat{z}_{t+1}$, based on the measurements up to time t. We then compare the measurement at time $t+1$, z_{t+1}, against its predicted value $\hat{z}_{t+1}$ to determine if its faulty.

— **L-step ahead**: Using measurements up to time t, we forecast the values for time $t+i$, $1 \leq i \leq L$ with $L > 1$. We then compare the actual measurements for time $t+i$, $1 \leq i \leq L$ against their forecast value. If the difference between the measured value and its

forecast for any sample is greater than δ, we flag that sample as faulty.

One-step ahead forecasting is more suited for detecting SHORT faults. The idea behind L-step ahead forecasting is to detect faults that last for long durations (for example, the fault shown in Figure 15). However, the potential error in our forecast grows with L (Chapter 5, [Box et al. 1994]). In order to control false positives due to erroneous forecast values, we restrict $L \leq s$.

Threshold for fault detection: We use two heuristics to determine the threshold δ for fault detection.

— **Forecast Confidence Interval**: For each sensor measurement in our "test" dataset, we compute both the forecast value and the 95% confidence interval for the forecast value. If a sensor measurement lies outside the 95% confidence interval for its forecast value, we flag that measurement as faulty.
Instead of using a fixed threshold δ for all the measurements, the confidence interval based heuristic adapts the threshold value dynamically for each measurement. If the confidence interval for the forecast value for a measurement is small (indicating that we have a high confidence that the forecast value represents the true value), then the threshold δ for that measurement is small. Similarly, if the confidence interval for the forecast value for a measurement is large, then the threshold for that measurement is large.

— **Forecast Error**: If the sensor measurement time series is monitored continuously for a long duration for the presence of faults, we can use the difference in forecast values and actual measurements observed in the past to set the threshold δ using the "Confidence Limit" heuristic described for LLSE based method in Section 3.2.

Alternatives to the ARIMA model: The choice of a time series model for sensor measurements is determined by the nature of the phenomenon being measured. For example, if the sensor measurements do not exhibit periodicity, an autoregressive (AR) or a moving average (MA) (or a combination of the AR and MA models called the autoregressive moving average (ARMA) model) would be more appropriate for time series analysis. The model that we use in this work is one of the simplest seasonal models available. It is possible that a more complex season model can be a better fit for the sensor measurement time series that we analyze in this paper. However, using a more complex model requires estimating more parameters, and generally, implies a more computationally intensive training phase requiring a larger training dataset. Our results with real-world datasets (Section 5) show that the model that we use in this paper is effective at detecting faults in a time series of temperature measurements. The issue of determining the *best-fit* time series model for modeling phenomena such as outdoor temperature and humidity is not the focus of our work.

3.4 A Learning-based Method

For phenomena that may not be spatio-temporally correlated, a learning-based method might be more appropriate. For example, if the pattern of "normal" sensor readings and the effect of sensor faults on the reported readings for a sensor are well understood, then we can use learning-based methods, for example Hidden Markov Models (HMMs) and neural networks, to construct a model for the measurements reported by that sensor. In this paper we chose HMMs because they are a reasonable representative of learning based methods

that can simultaneously detect and classify sensor faults. Determining the most effective learning based method is outside the scope of this paper.
A Hidden Markov Model (HMM) is characterized by the following.

—The number of states in the model, S.

—The set of possible measurements, O.

—For each state $s \in S$, the conditional probability of observing a measurement $o \in O$, $P\{o \mid s\}$.

—The state transition probabilities $A = \{a_{ij}\}$ where a_{ij} represents the probability of a transition to state j from state i.

—The initial state distribution $\pi = \{\pi_i\}$ where π_i is the probability that the HMM starts in state i.

Although the states of an HMM are hidden, we can attach some physical significance to them. For example, based on our characterization of faults in Section 2, for a sensor measuring ambient temperature, we can use a 5-state HMM with the states corresponding to day, night, SHORT faults, NOISE faults, and CONSTANT faults. Such an HMM can capture not only the diurnal pattern of temperature but also the distinct patterns in the reported values in the presence of faults.

For the basic HMM, the set of possible measurements O is discrete. In this paper, we use the basic HMM, and when modeling temperature, humidity, etc. we make the observation space discrete by clustering the sensor readings into bins. For example, we bin temperature measurements into bins of width 0.1°C. If the observation space is continuous (and cannot be made discrete), one can use a continuous density HMM (CDHMM), defined for a continuous observation space. However, a CDHMM is computationally more complex than a basic HMM. We chose a basic HMM over CDHMM because (a) we could make the observation space discrete without introducing significant rounding-off errors, and (b) we wanted to avoid the additional computational complexity involved in using a CDHMM (provided the basic HMM based method proved effective at detecting sensor faults). Our results with injected faults (Section 4) and real- world datasets (Section 5) demonstrate that our basic HMM based method is effective at detecting the types of sensor faults we consider in this paper.

Given values for $S, O, P\{o \mid S\}, A$, and π, and a sequence of sensor measurements $\{O_t\}$, the HMM can generate the most likely state S_t that resulted in observation O_t for each observation. If the state S_t associated with an observation O_t is a fault state (SHORT, NOISE, or CONSTANT), then we classify observation O_t as faulty. Thus, our HMM based method can detect as well as classify faults.

In order to estimate the parameters $S, O, P\{o \mid S\}, A$, and π of the HMM used for fault detection, we used a supervised learning technique. We injected faults into (fault-free) training dataset (using the techniques described in Section 4), labeled each sample as fault-free or faulty with a particular fault type, and used this labeled data for estimating the parameters of the HMM. For details on the techniques used for estimating the parameters of an HMM, and for generating the most likely sequence of states for a given sequence of observations, please refer to the tutorial by Rabiner [1989]. We used the implementation of HMMs provided in MATLAB [MATLAB].

3.5 Hybrid Methods

Finally, observe that we can use combinations of the Rule-based, LLSE, ARIMA and HMM methods to eliminate/reduce the false positives and negatives. In this paper, we study two such schemes:

— **Hybrid(U)**: Over two (or more) methods, this method identifies a sample as faulty if at least one of the methods identifies the sample as faulty. Thus, Hybrid(U) is intended for reducing false negatives (it may not eliminate them entirely, since all methods might fail to identify a faulty sample). However, it can suffer from false positives.

— **Hybrid(I)**: Over two (or more) methods, this method identifies a sample as faulty only if both (all) the methods identify the sample as faulty. Essentially, we take an intersection over the set of samples identified as faulty by different methods. Hybrid(I) is intended for reducing false positives (again, it may not eliminate them entirely) but suffers from false negatives.

Several other hybrid methods are possible. For example, Hybrid(U) can be easily modified so that results from different methods have different weights in determining if a measurement is faulty. This would be advantageous in situations where a particular method or heuristic is known to be better at detecting faults of a certain type.

Table I summarizes the methods (and their specific variations) described in this section. It also highlights the parameters associated with each method that must be determined – using expert knowledge, heuristics or empirically (trial and error) – in order to be able to use these methods for fault detection. For each parameter, we also specify, within brackets, the approaches used in this paper for determining their value.

Method	Parameters
SHORT & NOISE rules	N: sample window size (domain knowledge), δ: threshold for fault detection (Histogram method)
Linear Least-Squares Estimation (LLSE)	δ: threshold for fault detection (Maximum training error, Confidence Limit)
ARIMA model	s: seasonality (domain knowledge)
ARIMA model: One-step	δ: threshold for fault detection (Forecast Confidence Interval, Forecast error)
ARIMA model: L-step	L: look-ahead forecast parameter (domain knowledge and empirically), δ: threshold for fault detection (Forecast Confidence Interval, Forecast error)
HMM	Number of states (domain knowledge)

Table I. Fault Detection Methods

4. EVALUATION: INJECTED FAULTS

Before we can evaluate the prevalence of faults in real-world datasets using the methods discussed in the previous section, we need to characterize the accuracy and robustness of these methods. To do this, we artificially injected faults of the types discussed in Section 2 into sensor measurements from a real-world dataset containing measurements for chlorophyll concentration in lake water [NAMOS 2005]. Ideally, before injecting faults, we should ensure that the dataset does not have any faulty samples. However, since we did

not have ground truth information about faults for this dataset, we relied on a combination of visual inspection, and feedback from the members of the NAMOS project to ensure (to the extent possible) that the dataset did not contain any faults.

This methodology has two advantages. Firstly, injecting faults into a dataset gives us an accurate "ground truth" that helps us better understand the performance of a detection method. Secondly, we are able to control the *intensity* of a fault and can thereby explore the limits of performance of each detection method as well as comparatively assess different schemes at low fault intensities. Many of the faults we have observed in existing real datasets are of relatively high intensity; even so, we believe it is important to understand the behavior of fault detection methods across a range of fault intensities, since it is unclear if faults in future datasets will continue to be as pronounced as those in today's datasets.

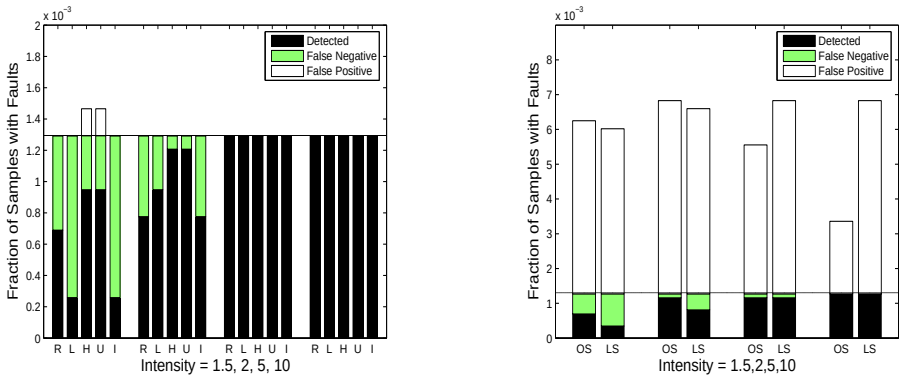
Sensor measurements for injecting faults. For evaluating the Rule-based methods, LLSE, HMM and the two hybrid methods, we inject NOISE faults into measurements of chlorophyll concentration in lake water collected by buoy number 106 during the NAMOS deployments in October, 2005 [NAMOS 2005]. Buoy number 106 collected a measurement every 8 seconds, and collect $22,600$ measurements in total. We use the samples collected during the first 24 hours (the first $11,000$ samples) as training data to train LLSE and the HMM. We inject Noise faults into the remaining samples, and we use these samples to test our methods. To train the HMM, we inject faults in the training data as well. These faults were of the same duration and intensity as the faults used for comparing different methods. We did not inject any faults in the training data for LLSE.

The samples from the NAMOS deployment did not provide enough training data to estimate the parameters of our ARIMA model, and hence, we use samples from the SensorScope deployment to evaluate the ARIMA model based method. We injected NOISE faults into temperature measurements collected by weather station 3 during the SensorScope deployment [SensorScope 2006]. This weather station took a temperature measurement every 30 seconds. We estimate the parameters of the ARIMA model using samples collected over 3 days (8640 total samples) by station 3. We inject NOISE faults into samples from another day (2880 samples collected over 24 hours), and use these samples to test our ARIMA model based methods.

Below, we discuss the detection performance of various methods for each type of fault. We describe how we generate faults in the corresponding subsections. We use three metrics to understand the performance of various methods: the number of faults detected, false negatives, and false positives. More specifically, we use the fraction of samples with faults as our metric, to have a more uniform depiction of results across the data sets. For the figures pertaining to this section and Section 5, the labels used for different detection methods are:**R**: Rule-based, **L**: LLSE, **H**: HMM, **OS**: ARIMA model based One-step ahead forecasting, **LS**: ARIMA model based L-step ahead forecasting, **U**:Hybrid(U), and **I**: Hybrid(I).

4.1 SHORT Faults

To inject SHORT faults, we picked a sample i and replaced the reported value v_i with $\hat{v_i} = v_i + f \times v_i$. The multiplicative factor f determines the intensity of the SHORT fault. We injected SHORT faults with intensity $f = \{1.5, 2, 5, 10\}$. Injecting SHORT faults in this manner (instead of just adding a constant value) does not require the knowledge of the range of "normal" sensor readings.



(a) SHORT Rule, LLSE, HMM, and Hybrid Methods (b) ARIMA Model

Fig. 5. Injected SHORT faults

Figures 5.a and 5.b depict the performance of our various methods for detecting SHORT faults of different intensities. The horizontal line in both the figures represents the actual fraction of samples with injected faults. The four sets of bar plots correspond to increasing intensity of SHORT faults (left to right).

The ARIMA model based methods incur significantly higher number of false positives compared to other methods. We plot their performance separately in Figure 5.b in order to be able to depict the performance of all the other methods with better clarity in Figure 5.a. For the Hybrid(U) and Hybrid(I), we include all the methods except the ARIMA model based methods. Including the ARIMA model based method in our hybrid methods would significantly increase the number of false positives for the Hybrid(U) method.

The SHORT rule and LLSE do not have any false positives; hence, the Hybrid(I) method exhibits no false positives (thus eliminating the false positives incurred by the HMM based method). However, for faults with low intensity ($f = 1.5, 2$), the SHORT rule as well as LLSE have significant false negatives, and hence, the Hybrid(I) method also has a high number of false negatives for these intensities.

The HMM method has fewer false negatives compared to SHORT rule and LLSE but it has false positives for the lowest intensity ($f = 1.5$) faults. While training the HMM for detecting SHORT faults, we observed that if the training data had a sufficient number of SHORT faults (on the order of 15 faults in 11000 samples), the intensity of the faults did not affect the performance of the HMM.

It is evident from Figure 5.a that Hybrid(U) performs like the method with more detections and Hybrid(I) performs like the method with less detections (while eliminating the false positives). However, in general this does not have to be the case, e.g., in the absence of false positives, Hybrid(U) could detect more faults than the best of the methods and Hybrid(I) could detect fewer faults than the worst of the methods (as illustrated on the real data sets in Section 5).

Our ARIMA model based methods do not perform as well as the other methods. Even though the One-step (OS) and the L-step (LS) ahead forecasting methods are able to detect most of the high intensity $f = \{5, 10\}$ faults, overall, they incur a significantly higher fraction of false positives than the other methods. However, comparing the performance of One-step ahead forecasting method against the L-step ahead forecasting with $L = 120$

samples shows that for all fault intensities, One-step ahead forecasting performs better than the L-step ahead forecasting, especially for faults with intensity $f = 10$. This is to be expected because fault detection with L-step ahead forecasting is better suited for detecting faults that affect more than one sample, for example NOISE and CONSTANT faults.

The choice of threshold used to detect a faulty sample governs the trade-off between false positives and false negatives; reducing the threshold would reduce the number of false negatives but increase the number of false positives. We select the threshold using the histogram method (with window size $N = 100$) for the Rule-based methods, the "Maximum Error" heuristic for LLSE, the "Forecast Error" heuristic for One-step ahead forecasting, and the "Forecast Confidence Interval" heuristic for L-step ahead forecasting.

4.2 NOISE Faults

To inject NOISE faults, we pick a set of successive samples W and add a random value drawn from a normal distribution, $N(0, \sigma^2)$, to each sample in W. We vary the intensity of NOISE faults by choosing different values for σ. The Low, Medium and High intensity of NOISE faults correspond to $0.5x$, $1.5x$, and $3x$ increase in standard deviation of the samples in W. Apart from varying the intensity of NOISE faults, we also vary their duration by considering different numbers of samples in W.

Duration of Noise faults. We inject NOISE faults of duration (number of samples in W) 3000 samples, 2000 samples, 1000 samples, and 100 samples for evaluating the NOISE rule, LLSE, HMM, and the two hybrid methods. Since these samples (from the NAMOS deployment) were collected at an interval of 8 seconds, in terms of time, these fault durations range from longer than 6 hours (for 3000 samples) to less than 15 minutes (for 100 samples).

For our ARIMA model based method, like in the case of injected SHORT faults, we use One-step ahead forecasting and L-step ahead ($L = 120$) forecasting for detecting the (injected) NOISE faults. In order to understand the impact of the parameter L on the performance of the L-step ahead forecasting based NOISE fault detection, we vary the duration of NOISE faults relative to $L = 120$. Hence, we inject NOISE faults of duration 720 samples, 120 samples, and 60 samples. Note that a fault affecting 720 samples lasts for 6 hours because the SensorScope deployment used a sampling interval of 30 seconds.

Figures 6 ($|W| = 3000$), 7 ($|W| = 2000$), 8 ($|W| = 1000$) and 9 ($|W| = 100$) show the performance of the NOISE rule, LLSE, HMM and the hybrid methods for NOISE faults with varying intensity and duration. For the ARIMA model, Figures 10 ($|W| = 60$), 11 ($|W| = 120$), and 12 ($|W| = 720$) show the performance of One-step and L-step ($L = 120$) ahead forecasting based fault detection. The horizontal line in each figure corresponds to the fraction of samples with faults.

4.2.1 *Impact of Fault Duration.* The impact of NOISE fault duration is most dramatic for the HMM method. For $|W| = 100$, regardless of the fault intensity, the number of faulty samples were not enough to train the HMM model. Hence, Figure (9) does not show results for HMM. For $|W| = 1000$ and low fault intensity, we again failed to train the HMM model. This is not very surprising because for short duration (e.g., $|W| = 100$) or low intensity faults, the data with injected faults is very similar to data without injected faults. For faults with medium and high intensity or faults with sufficiently long duration, e.g., $|W| \geq 1000$, performance of the HMM method is comparable to the NOISE rule and LLSE.

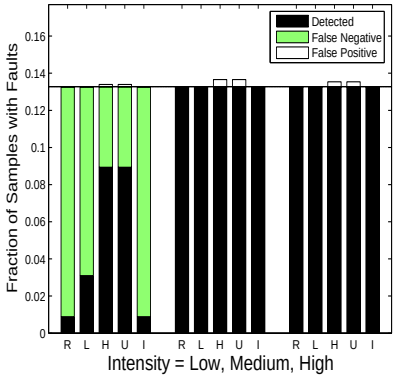


Fig. 6. NOISE Fault: 3000 samples

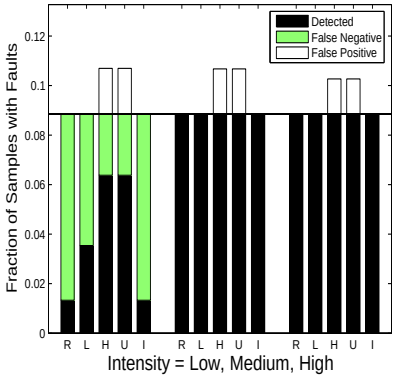


Fig. 7. NOISE Fault: 2000 samples

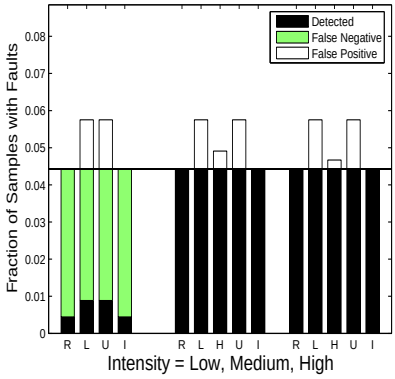


Fig. 8. NOISE Fault: 1000 samples

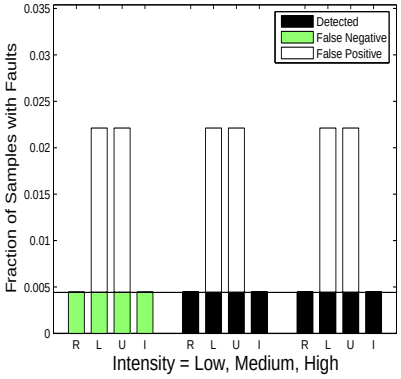


Fig. 9. NOISE Fault: 100 samples

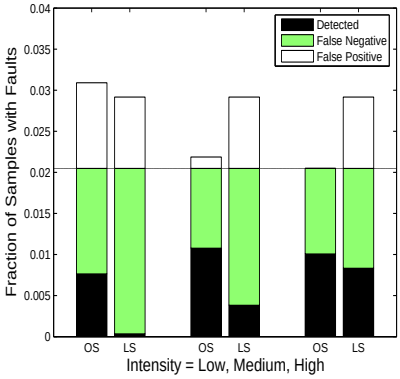


Fig. 10. NOISE, ARIMA: 60 samples

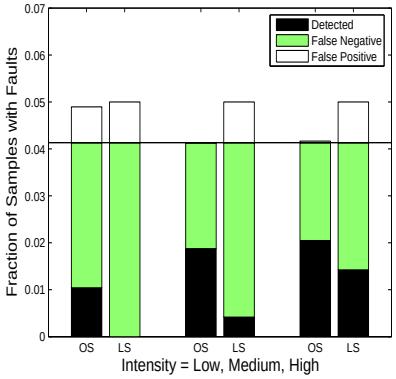


Fig. 11. NOISE, ARIMA: 120 samples

The NOISE rule and LLSE method are more robust to fault duration than HMM in the sense that we were able to derive model parameters for those cases. However, for

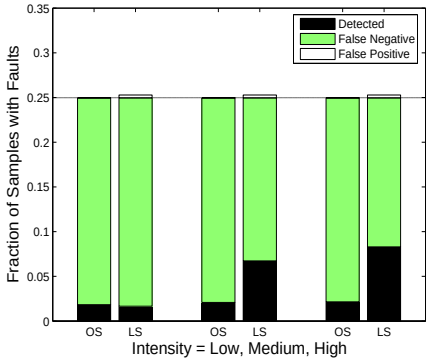


Fig. 12. NOISE, ARIMA: 720 samples

$|W| = 100$ and low fault intensity, both the methods fail to detect any of the samples with faults. The LLSE also has a significant number of false positives for $|W| = 100$ and fault intensity $0.5x$. The false positives were eliminated by the Hybrid(I) method.

For the ARIMA model based method, One-step ahead forecasting based method detects fewer faults as the fault duration increases. For example, for $|W| = 60$ samples and high fault intensity, One-step forecasting based method detects 50% of faults, but for $|W| = 720$ samples and high fault intensity, it detects only 9% of the faults. L-step ahead forecasting based method is more robust to increase in fault duration. It detects 41% and 33% of the high intensity faults when $|W| = 60$ samples and $|W| = 720$ samples, respectively, and hence, the degradation in its performance is not as severe as in case of the One-step ahead forecasting based method. It is also worth noting that for NOISE faults affecting $|W| = 60$ samples and $|W| = 120$ samples, regardless of the fault intensity, One-step ahead forecasting detects more faults than L-step ahead forecasting. However, for NOISE faults affecting $|W| = 720$ samples, L-step ahead forecasting detects more faults than One-step ahead forecasting. Hence, overall, the One-step ahead forecasting based method is more suited for detecting short-to-medium duration faults, whereas L-step ahead forecasting based method is need when the faults last for a long duration.

4.2.2 *Impact of Fault Intensity.* For medium and high intensity faults, there are no false negatives for the three methods–NOISE rule, LLSE and HMM. For low intensity faults, these three methods have significant false negatives. For fault duration and intensities for which the HMM training algorithm converged, the HMM method gives lower false negatives compared to the NOISE rule and LLSE. However, most of the time the HMM method gave more false positives. Hybrid methods are able to reduce the number of false positives and negatives, as intended. Like the other methods, the ARIMA model based One-step ahead and L-step ahead forecasting methods perform better (detect more faults and incur fewer false positives) as the fault intensity increases. High false negatives for low fault intensity arise because the measurements with injected faults are very similar to the measurements without faults.

Overall, however, the ARIMA model based method does not perform as well as the other three methods. For example, for NOISE faults lasting 6 hours (720 samples in case of the ARIMA model and 3000 samples for the other methods), even with high fault intensity, the

ARIMA model based method detects 33% of the faults whereas the other three methods can detect all the faults. The ARIMA model based method also incurs a higher rate of false positives compared to the other methods.

4.2.3 *ARIMA model–Impact of parameter L.* The performance results for the ARIMA model indicate that, for detecting very long duration faults, L-step ahead forecasting is better suited than the One-step ahead forecasting. However, for high intensity NOISE faults lasting for 6 hours (affecting 720 samples), by estimating the measurements 1 hour in advance (using L-step ahead forecasting with $L = 120$), we could detect only 33% of the faults. Will increasing the forecasting interval, L, help us detect more faults?

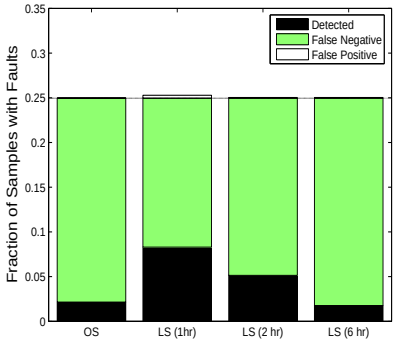


Fig. 13. High intensity, Long duration NOISE faults

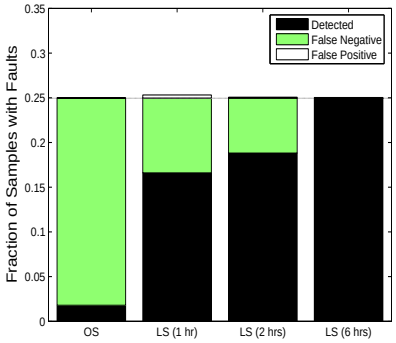


Fig. 14. CONSTANT and NOISE faults

Figure 13 shows the performance of L-step ahead forecasting technique for detecting high intensity NOISE faults with duration $|W| = 720$ samples. Contrary to our intuition, setting L equal to the fault duration (720 samples or 6 hours) detects less faults than $L = 120$ samples (1 hour). As we increase L, the uncertainty in the forecast value grows. Due to this increased uncertainty, we are forced to set the fault detection threshold δ to a large value (using the "Forecast Confidence Interval" heuristic defined in Section 3.3) to prevent large number of false positives. Even though the variance of the NOISE fault is high, for most of the samples, the change in the sensor measurement (due to the additive noise) is small compared to the threshold δ. Hence, fewer faults are detected for large value of L. For $L = 120$ samples, the uncertainty in the forecast value is smaller and hence, we can set a lower threshold value. By setting a lower threshold value, we are able to detect more faults because now the change in sensor measurements due to additive noise is larger than the threshold for a larger fraction of faulty samples. Thus, forecasting too far ahead into the future (large value of L) is not always beneficial. The benefits of using a large value of L to detect long duration faults can be outweighed by the increased error in forecast value for large L.

Does there exist a scenario for which increasing the value of L improves the performance of the ARIMA model based L-step ahead forecasting? In the same temperature measurement time series as the one used to obtain the results in Figure 13 (obtained from the SensorScope deployment), we inject a combination of high intensity NOISE and CONSTANT faults into 720 samples. The CONSTANT faults add a value of 20 to each sample.

Figure 14 shows the impact of increasing L on the number of faults detected. As we increase the value of L, we detect more faults and for L equal to the duration of faults, we detect all the faults. In this scenario, the faults are of such a high intensity that the increase in threshold δ (due to a larger value of L) is not large enough to cause false negatives. A similar combination of CONSTANT and NOISE faults occurred in the INTEL lab dataset [INTEL 2004], and for this dataset, increasing the value of L enabled us to detect more faults (refer to Figure 17).

The results shown in Figures 13 and 14 demonstrate that the *best* value of L for detecting long duration faults depends not only on the duration of the faults but also on the difference in magnitude between the faulty and normal sensor readings. For the real world datasets, contextual information about the phenomenon being sensed and the nature of faults are useful in determining the best value for L. For slowly varying measurements like ambient temperature, humidity, etc., if the long duration faults increase only the variance of the sensor measurements, limiting the forecast interval to a short duration (for example, an hour) works best. However, if the faults change both the mean and the variance of the sensor measurements, matching the forecast interval to the fault duration (to the extent possible) gives better performance. Apart from using this contextual information, we also relied on trial and error to determine the value of L that gave the best performance.

4.3 Other Hybrid methods

We noted earlier that in our evaluation with injected faults, Hybrid(U) performs like the method with more detections and Hybrid(I) performs like the method with less detections (while eliminating the false positives). As a result, for low intensity NOISE faults (see Figures 6 and 7), Hybrid(U) performs like the HMM, and Hybrid(I) performs like the NOISE rule.

Note that the LLSE method, apart from detecting faulty measurements, also provides an estimate of their correct value. We can leverage this fact to design a hybrid method that uses two different methods *in sequence* as follows.

—Use the LLSE method to identify (some of the) faulty measurements.

—Replace these measurements with their estimates from LLSE.

—Use this modified time series of measurements as input to another method.

We next evaluate whether the hybrid approach of using two different methods in sequence can detect more low intensity faults.

LLSE and HMM: We use a combination of LLSE and HMM to detect low intensity NOISE faults with duration $|W| = 3000$, and $|W| = 2000$ samples injected into readings from the NAMOS, October, 2005 deployment [NAMOS 2005]. These measurements are also used for the evaluation results shown in Figures 6 and 7. Table II shows a comparison between the LLSE, the HMM, and the combined LLSE→HMM method. Note that using the LLSE and the HMM methods *in sequence* helps us detect more faults than either of the two methods.

LLSE and ARIMA: Low intensity faults can also be detected using a combination of the LLSE and the ARIMA model based methods. Table III compares the performance of two *in sequence* hybrid methods (LLSE→ARIMA (One-step), and LLSE→ARIMA (L-step),

Duration # samples	LLSE	HMM	LLSE→HMM
2000	40	68	70
3000	23.3	71	74

Table II. Low Intensity NOISE faults, NAMOS, Percentage of faulty samples detected

L=120) against several other methods for detecting low intensity NOISE faults. The results in Table III are for the same sensor measurements as the ones used for Figures 10, 11, and 12.

We make three interesting observations based on the results shown in Table III. First, the combined LLSE→ARIMA (L-step) method outperforms the ARIMA (L-step) method, but its performance is comparable to the LLSE method. Hence, combining LLSE and ARIMA (L-step) does not provide a significant benefit over using the LLSE method. Second, for fault durations of 120 and 720 samples, the LLSE→ARIMA (One-step) hybrid method outperforms both the LLSE and the ARIMA (One-step) methods; with the hybrid method detecting 4.3% more faults compared to the best performing method (LLSE) for the fault duration equal to 720 samples. Third, for fault duration equal to 60 samples, the hybrid LLSE→ARIMA (One-step) method detects 11.9% fewer faults than the ARIMA (One-step) method, but it outperforms LLSE by a small margin. Thus, using two different methods *in sequence* may not always perform better than either of the two methods.

Duration # samples (time)	LLSE	ARIMA (One-step)	LLSE→ARIMA (One-step)	ARIMA (L-step)	LLSE→ARIMA (L-step)
60 (0.5 hr.)	23.7	37.3	25.4	1.7	23.7
120 (1 hr.)	23.5	25.2	26.9	0	23.5
720 (6 hrs.)	35.9	7.4	40.2	6.7	37.4

Table III. Low intensity NOISE faults, Percentage of faulty samples detected

Duration # samples (time)	LLSE	ARIMA (One-step)	LLSE→ARIMA (One-step)	ARIMA (L-step)	LLSE→ARIMA (L-step)
60 (0.5 hr)	9.6	1	2	0.87	0.97
120 (1 hr)	9.6	0.76	1.8	0.87	0.97
720 (6 hrs)	4.5	0	0.3	0.2	0.2

Table IV. False Positives as % of total # samples (2880)

False positives: Using two (or more) methods in sequence to detect faults can increase the number of false positives. Consider the case of using LLSE and ARIMA methods in sequence. In the first step, we replace the value of all the samples identified as faulty by their estimate given by the LLSE method. If the LLSE method has false positives, then we alter the values of these normal (not faulty) samples. In addition, if the estimates from LLSE are not good (differ significantly from "normal" sensor readings), these samples might be identified as faulty by the ARIMA method in the next step.

Table IV compares the false positive rate for the hybrid methods LLSE→ARIMA (One-step and L-step) against the ARIMA methods. For reference, we also show the false positive rate for the LLSE method. The increase in the false positive rate is more significant

for the LLSE→ARIMA (One-step) compared to LLSE→ARIMA (L-step). For both, the One-step and the L-step methods, we used the Forecast Confidence Interval heuristic (Section 3.3) to determine the threshold δ for fault detection. As discussed in Section 3.3, the confidence interval is smaller for One-step ahead forecasting (resulting in a smaller δ) compared to the L-step ahead forecasting. Hence, when used in sequence with the LLSE method, the ARIMA (One-step) method is more vulnerable to false positives (due to the false positives from LLSE in the first step) compared to the ARIMA (L-step) method.

In summary, the evaluation presented in Tables II, III, and IV show that if it is possible to obtain a good estimate of the correct value of an erroneous measurement, then using two methods in sequence can (possibly) detect more faults at the expense of a (slightly) higher false positive rate.

5. FAULTS IN REAL-WORLD DATA SETS

We analyze four datasets from real-world deployments – SensorScope [SensorScope 2006], Great Duck Island (GDI) [Mainwaring et al. 2002], INTEL Berkeley Lab [INTEL 2004], and NAMOS [NAMOS 2006] – for prevalence of faults in sensor traces. The sensor traces contain measurements for temperature, humidity, light, pressure, and chlorophyll concentration. All of these phenomena exhibit a diurnal pattern in the absence of outside perturbation or sensor faults.

Out of the four datasets, we were able to apply all four fault detection methods only to the SensorScope dataset. We could not apply one or more methods to the other datasets due to a variety of factors–for example, the NAMOS dataset did not have enough data for training. We discuss these factors in detail below. Table V provides a summary of the methods applied to each of the four datasets for fault detection.

Dataset	Rule-based	LLSE	HMM	ARIMA
SensorScope [SensorScope 2006]	✓	✓	✓	✓
INTEL Lab [INTEL 2004]	✓		✓	✓
GDI [Mainwaring et al. 2002]	✓	✓		
NAMOS [NAMOS 2006]	✓			

Table V. Real-world datasets and Detection methods

5.1 SensorScope

The SensorScope project is an ongoing outdoor sensor network deployment consisting of weather-stations with sensors for sensing several environmental quantities such as temperature, humidity, solar radiation, soil moisture, and so on [SensorScope 2006]. We analyzed the temperature measurements collected every 30 seconds over six months at 64 weather stations.

We did not have the ground truth regarding faulty samples for this dataset. Since this dataset is very large (more than 500,000 samples per weather station), we used a combination of visual inspection and Rule-based methods to identify samples with (very likely) faulty temperature values. The samples identified as faulty not only provide a ballpark estimate for the prevalence of faults, but also serve as a benchmark against which we compare the performance of other fault detection methods.

Using visual inspection and the SHORT rule, we identified approximately 0.01% of the total samples as affected by SHORT faults. There was significant variation in the

Method	Detected (% of total # faulty samples)	False Positive (% of total # samples)
HMM	35.3	< 0.01
LLSE	69.8	0.01
ARIMA (One-step)	96.7	0.02
ARIMA (L-step), L=120	76.7	3
Hybrid(I)	25	< 0.01
Hybrid(U)	98.9	3

Table VI. SensorScope: SHORT faults

prevalence of SHORT faults across individual weather stations – some stations did not have any faulty samples whereas one weather station (ID=39) had more than 0.07% of samples affected by SHORT faults. We did not find any instance of NOISE and CONSTANT faults. Table VI shows the performance of various methods. The "Detected" and "False Positive" percentages are computed by aggregating the number of faulty samples and the false positives, respectively, over all the weather stations.

Based on the results shown in Table VI, we make the following three observations. First, the ARIMA (One-step) method performs the best; it detected 96.9% of the faulty samples and incurred few false positives. Second, our evaluation with injected SHORT faults (Section 4.1) showed that the ARIMA (One-step) method is better suited than the ARIMA (L-step) method for detecting SHORT faults. This observation is confirmed by the relative performance of the One-step and the L-step ahead forecasting methods for detecting the SHORT faults in the SensorScope dataset. Third, the HMM and the LLSE methods detect less faults than the two ARIMA methods but they incur fewer false positives.

5.2 INTEL Lab, Berkeley data set

54 Mica2Dot motes with temperature, humidity and light sensors were deployed in the Intel Berkeley Research Lab between February 28th and April 5th, 2004 [INTEL 2004]. In this paper, we present the results on the prevalence of faults in the temperature readings (sampled on average once every 30 seconds).

This dataset exhibited a combination of NOISE and CONSTANT faults. Each sensor also reported the voltage values along with the samples. Inspection of these voltage values showed that the faulty samples were well correlated with the last few days of the deployment when the lithium ion cells supplying power to the motes were unable to supply the voltage required by the sensors for correct operation.

The faulty samples were contiguous in time (Figure 15). We applied the NOISE rule, the HMM method (using a simple 2-state HMM model), and the ARIMA One-step and L-step methods to detect the faulty samples. Interestingly, for this dataset, we could not apply the LLSE method. NOISE faults across various nodes were correlated, since all the nodes ran out of battery power at approximately the same time. This breaks an important assumption underlying the LLSE technique, that faults at different sensors are uncorrelated.

Figure 16 shows the fraction of the total temperature samples (collected by all the motes) with faults, and the performance of the NOISE rule, the HMM and the hybrid methods at detecting these faults. We present the performance results for the ARIMA model separately in Figure 17 for clarity. Both the NOISE rule and HMM have some false negatives while the HMM also has some false positives. For this data set, we could eliminate all the false positives using Hybrid(I) with NOISE rule and HMM. However, combining the NOISE

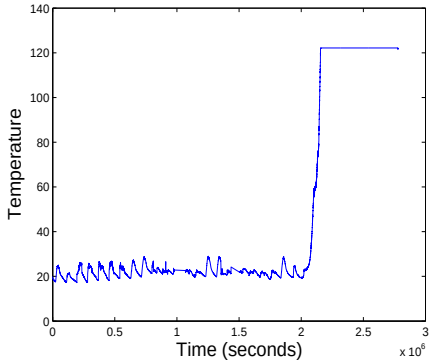


Fig. 15. Intel data set: NOISE faults

rule and HMM for Hybrid(I) incurred more false negatives.

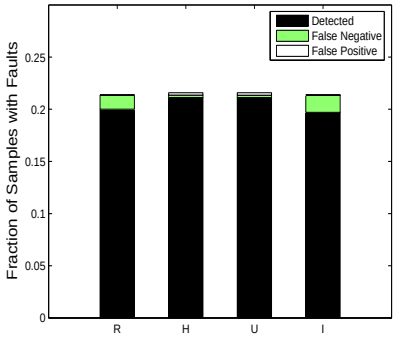


Fig. 16. Intel data set: Prevalence of NOISE faults

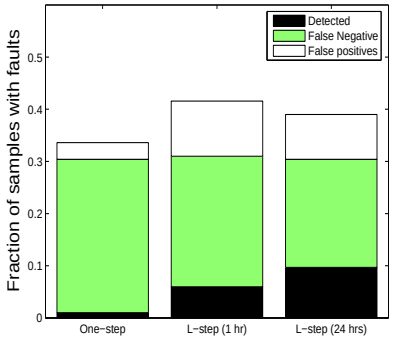


Fig. 17. Intel data set: ARIMA methods

The fact that this dataset contained sensor measurements collected over two months enabled us to apply our time series method for fault detection. We set the periodicity parameter $s = 2880$ because samples were collected every 30 seconds and hence, 2880 samples were collected per day. For each sensor mote, we used measurements collected over the first 10 days to estimate the parameters of the time series model. The size of the training dataset was influenced by two factors: (a) the periodicity parameter s, and (b) missing samples. Since our model differences the time series twice (the differences are $(z_t - z_{t-1}) - (z_{t-s} - z_{t-s-1})$, refer to equation (2)), we discard $s+1 = 2881$ samples (i.e., measurements collected over a day). The average yield for INTEL data set was 50%; i.e. only 50% of the samples collected every 30 seconds at a sensor were delivered at the base-station. Due to large number of missing samples per day, we had to include measurements from more days in the training dataset, in order to have sufficient data for training the ARIMA model.

Figure 17 shows the performance of the time series based method with three different forecasting scenarios: One-step ahead, L-step ahead with $L = 120$ samples (i.e. 1 hour) and L-step ahead with $L = 2880$ samples (i.e. 24 hours). Maximum number of faults are detected with L-step ahead forecast with $L = 2880$. This is so because the duration of the NOISE and CONSTANT faults in the INTEL dataset (Figure 15) was on the order of days, and hence L-step ahead forecasting with a large L is needed to detect these long duration faults. Even for $L = 2880$ samples, we could not detect two-thirds of the faults. However, our method was able to detect 90% of the faulty samples reported during the first day on which the faults started. This shows that our time series based method can detect long duration faults provided that their duration is shorter than the time series periodicity.

Finally, in this data set, surprisingly there were few instances of SHORT faults. A total of 6 faults were observed for the entire duration of the experiment (Table VII). All of these faults were detected by the HMM method, LLSE, the SHORT rule, and the ARIMA model based method.

ID	# Faults	Total # Samples
2	1	46915
4	1	43793
14	1	31804
16	2	34600
17	1	33786

Table VII. Intel Lab: SHORT Faults, Temperature

5.3 Great Duck Island (GDI) data set

We looked at data collected using 30 weather motes on the Great Duck Island over a period of 3 months [Mainwaring et al. 2002]. Attached to each mote were temperature, light, and pressure sensors, and these were sampled once every 5 mins. Of the 30 motes, the data set contained sampled readings from the entire duration of the deployment for only 15 motes. In this section, we present our findings on the prevalence of faults in the readings for these 15 motes.

The predominant fault in the readings was of the type SHORT. We applied the SHORT rule, the LLSE method and Hybrid(I) to detect SHORT faults in light, humidity, and pressure sensor readings. We could not apply the ARIMA and the HMM methods to this dataset due to a large number of missing (not recorded) observations. Each of the 15 motes that we considered had more than 60% of the samples missing for at least 18 days, and more than 10% of the samples missing for at least 56 days. For an ARIMA model with periodicity s, in order to predict the sensor reading at time t, we need sensor readings at times $t-1$, $t-s$, and $t-s-1$ (refer to Equation (2), Section 3.3). If any of these three readings are missing, then the ARIMA model cannot predict the reading at time t. For the ARIMA methods, one way to tackle the problem of one (or more) missing readings needed to predict the value at time t can be to use estimates of these missing readings (obtained from the ARIMA or LLSE method). This works if we can be confident that these estimates are not erroneous. This approach worked for the INTEL dataset, but did not for the GDI dataset, mainly because we could not obtain estimates for all the missing samples. Similarly, the large fraction of missing samples also prevented the training phase of the HMM method from converging, and hence, we could not use the HMM method on this dataset.

Figure 18 shows the overall prevalence (computed by aggregating results from all the 15 nodes) of SHORT faults for different sensors in the GDI data set. The Hybrid(I) technique eliminates any false positives reported by the SHORT rule or the LLSE method. The intensity of SHORT faults was high enough to detect them by visual inspection of the entire sensor readings timeseries. This ground-truth is included for reference in the figure under the label **V**.

It is evident from Figure 18 that SHORT faults are relatively infrequent. They are most prevalent in the light sensor (approximately 1 fault every 2000 samples). Figure 19 shows the distribution of SHORT faults in light sensor readings across various nodes. We did not observe any discernible pattern in the prevalence of these faults across different sensor nodes;

In this data set, NOISE faults were infrequent. Only two nodes had NOISE faults with a duration of about 100 samples. The NOISE rule detected it, but the LLSE method failed primarily because its parameters had been optimized for SHORT faults.

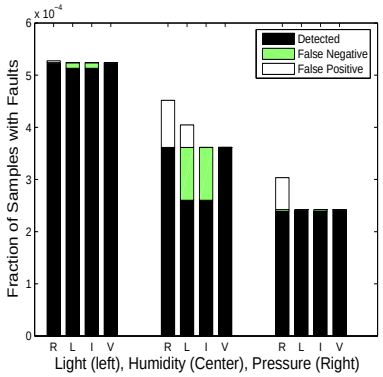


Fig. 18. SHORT Faults in GDI data set

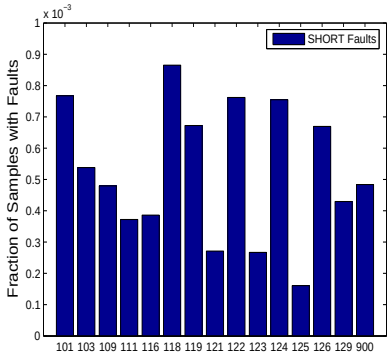


Fig. 19. SHORT Faults: Light Sensor

5.4 NAMOS data set

Nine buoys with temperature and chlorophyll concentration sensors (fluorimeters) were deployed in Lake Fulmor, James Reserve for over 24 hours in August, 2006 [NAMOS 2006]. Each sensor was sampled every 10 seconds. We analyzed the measurements from chlorophyll sensors for the prevalence of faults.

The predominant fault was a combination of NOISE and CONSTANT caused by hardware faults in the ADC (Analog-to-Digital Converter) board. Figure 1.a shows the measurements reported by buoy 103. We applied the NOISE Rule to detect samples with errors. Figure 20 shows the fraction of samples corrupted by faults. The sensors at 4 buoys were affected by the ADC board fault and in the worst case, at buoy 103, 35% of the reported values were erroneous. We could not apply the LLSE, the HMM and the ARIMA model based method because there was not enough data to train the models (data was collected for 24 hours only).[2]

[2]The NAMOS dataset used in Section 4 is from a different deployment done in October 2005 [NAMOS 2005]. This deployment consisted of only 4 buoys collecting data over 48 hours. We did not find any instances of faults

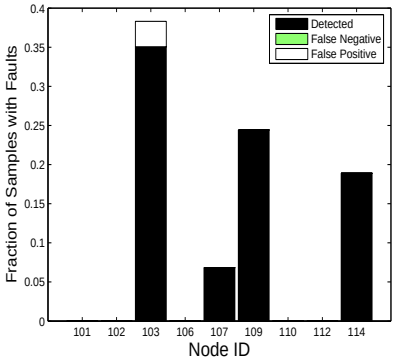


Fig. 20. NAMOS data set: NOISE/CONSTANT faults

6. RELATED WORK

Sensor data integrity encompassing fault detection, fault localization, identification of root causes, and correcting/recovering from data faults is an active area of research. In this section, we discuss several pieces of work that are closely related to our "data-centric" fault detection approach. These techniques are similar to one (or more) of the four classes of methods explored in this paper, and we point these out when discussing the corresponding work. However, none of these methods can be applied *without modification*, to fault detection. For example, some of these techniques are designed with a specific sensing task or type of sensor in mind (e.g., [Elnahrawy and Nath 2003] focusses on improving the accuracy of aggregate queries over sensor readings). When discussing each related work, we explain why it cannot be used directly for comparison. However, it may be possible to use some of these methods for sensor fault detection under different assumptions — for example, when information about the probability distribution of the true sensor readings, the characteristics of the noise corrupting sensor data, etc. is available.

Two recent papers [Khoussainova et al. 2006; Jeffery et al. 2006] have proposed a declarative approach to erroneous data detection and cleaning. StreamClean [Khoussainova et al. 2006] provides a simple declarative language for expressing integrity constraints on the input data. Samples violating these constraints are considered faulty. StreamClean uses a probabilistic approach based on entropy maximization to estimate the correct value of an erroneous sample. The evaluation in [Khoussainova et al. 2006] is geared towards a preliminary feasibility study and does not use any real world data sets. Extensible Sensor stream Processing (ESP) framework [Jeffery et al. 2006] provides support for specifying the algorithms used for detecting and cleaning erroneous samples using declarative queries. This approach works best when the types of faults that can occur and the methods to correct them are known a priori. The ESP framework is evaluated using the INTEL Lab data set [INTEL 2004] and a data set from an indoor RFID network deployment. These two declarative approaches are similar to our Rule-based methods. For example, we can think of the integrity constraints in StreamClean as (a combination of) rules. Similarly, specifying an algorithm for detecting erroneous samples within the ESP framework is equivalent

in the dataset from the October 2005 deployment and hence do not discuss it here.

to deciding which Rule-based method to apply, as both these decisions require knowledge of the data fault types.

Koushanfar et al. propose a real-time fault detection procedure that exploits multi sensor data fusion [Koushanfar et al. 2003]. Given measurements of the same source(s) by n sensors, the data fusion is performed (n+1) times, once with measurements from all the sensors; in the rest of the iterations the data from exactly one sensor is excluded. Measurements from a sensor are classified as faulty if excluding them improves the consistency of the data fusion results significantly. Simulations and data from a small indoor sensor network are used for evaluation in [Koushanfar et al. 2003]. Data from real world deployments are not used. This approach is similar to our HMM model based fault detection because it requires a sensor data fusion model. However, this approach cannot be used for fault detection in applications such as volcano monitoring [Werner-Allen et al. 2006] and monitoring of chlorophyll concentration in lake water [NAMOS 2006] where sensor data fusion functions are not easy to define without consulting a domain expert; however the HMM based method can be.

Elnahrawy et al. propose a Bayesian approach for cleaning and querying noisy sensors [Elnahrawy and Nath 2003]. The focus in [Elnahrawy and Nath 2003] is to improve the accuracy of aggregate queries over sensor readings (for example, SUM, AVG, COUNT etc.) rather than detect data faults. Using a Bayesian approach requires knowledge of the probability distribution of the true sensor readings and the characteristics of the noise process corrupting the true readings. The evaluation in [Elnahrawy and Nath 2003] does not use any real world datasets. In terms of the prior knowledge and the models required, the Bayesian approach in [Elnahrawy and Nath 2003] is similar to the HMM based method we evaluated in this paper. It is difficult to apply this method to the real world datasets considered in this paper due to the lack of prior information about the probability distribution for true sensor readings and the noise characteristics for each sensor, especially for the NAMOS dataset because the variation of chlorophyll concentration in lake water is not a well understood phenomenon. Even for well-understood phenomena like ambient temperature, humidity etc. (INTEL and SensorScope datasets), in the absence of ground truth values and contextual information related to the sensor calibration, determining the probability of the true sensor readings is difficult. In some cases, prior information related to true sensor readings can be obtained by calibrating and testing the sensors extensively in a controlled environment before a deployment, as done by Ramanathan et al. [Ramanathan et al. 2006] for a ground water monitoring deployment. Elnahrawy et al. also assume that the noise corrupting the sensor readings follows a Gaussian distribution. This assumption does not hold for the SHORT and the CONSTANT faults in general.

Tulone et al. model the temperature measurements in the INTEL Lab data set [INTEL 2004] using an autoregressive (AR) time series model [Tulone and Madden 2006]. They use a procedure similar to our One-step ahead forecasting based fault detection to detect faults/outliers similar to SHORT faults. However, the AR model based fault detection technique used in [Tulone and Madden 2006] is not suitable for detecting long duration faults (for example, NOISE and CONSTANT fault types) or short duration faults that occur frequently. This is because the autoregressive model captures only short time scale trends. For example, the AR model used in [Tulone and Madden 2006] is trained using samples collected over an hour (at a rate of a sample every 30 seconds) and hence, it captures temporal trends/correlations that are on order of a few minutes. For such an AR model, a

long duration NOISE fault lasting a couple of hours (or a short duration fault that occurs frequently) will be treated as a change in underlying data distribution, and the AR model will be retrained to fit the faulty data. Thus, it will fail to identify a majority of the faulty samples. Our seasonal ARIMA model based fault detection method can detect both the short duration and the long duration faults, and hence, subsumes the fault detection capabilities of an AR model. However, estimating the parameters for a seasonal ARIMA model is more complex computationally than estimating the parameters for an AR model. At the very least, we need temperature measurements collected over $2-3$ days to train our seasonal ARIMA model, whereas measurements collected over only an hour are enough to train the AR model used in [Tulone and Madden 2006].

Several papers on real-world sensor network deployments [Tolle et al. 2005; Ramanathan et al. 2006; Werner-Allen et al. 2006; Mainwaring et al. 2002] present results on meaningful inferences drawn from the collected data. However, to the best of our knowledge, only [Tolle et al. 2005; Werner-Allen et al. 2006; Ramanathan et al. 2006] do a detailed analysis of the collected data. The aim of [Ramanathan et al. 2006] is to do root cause analysis using Rule-based methods for on-line detection and remediation of sensor faults for a specific type of sensor network monitoring the presence of arsenic in groundwater. The SHORT and NOISE detection rules analyzed in this paper were proposed in [Ramanathan et al. 2006]. Werner et al. compare the fidelity of data collected using a sensor network monitoring volcanic activity to the data collected using traditional equipment used for monitoring volcanoes [Werner-Allen et al. 2006]. Finally, Tolle et al. examine spatiotemporal patterns in micro-climate on a single redwood tree [Tolle et al. 2005]. While these publications thoroughly analyze their respective data sets, examining fault prevalence and/or developing a generic sensor data fault detection approach was not an explicit goal. Our work presents a thorough analysis of four different real-world data sets. Looking at different datasets also enables us to characterize the accuracy and robustness of four qualitatively different detection methods.

7. DISCUSSION AND FUTURE WORK

In this section, we briefly discuss two issues that are relevant to data fault detection – outliers and event detection as well as utility of faulty samples; however, these are not a focus of our work. We also discuss interesting extensions to the fault detection methods evaluated in this paper that can be useful for detecting other types of faults.

Outliers and Events. Sensor measurements can deviate from their expected values due to an unexpected event or without any known causes (outliers), especially in the context of environmental monitoring. The fault detection methods discussed in this paper are likely to flag these measurements as faulty. However, discarding these samples as faulty may result in loss of important information related to an unknown event or outlier [Gupchup et al. 2008]. Before discarding/filtering out the faulty samples, we can try to extract some information from these samples. For example, if the samples flagged as faulty do not match any of the known fault models, it is possible that they are due to an unknown event or are simply outliers. In such a situation, contextual information about the sensors and the phenomenon being monitored can help us decide whether these samples are due to an unknown event. If so, using the flagged samples, we can try to generate a *signature* for the event using statistical and learning techniques other than those we have presented in this

paper. Having a *signature* can help us detect occurrences of the same event in the future.

Utility of faulty samples. Applications using a sensor network typically combine data from several sensors with different sensing modalities and spatiotemporal scales (commonly referred to as *data fusion*) in order to extract the relevant information. Depending on the fault type and intensity, a faulty sample may still provide useful information. During the data fusion process, we can assign a lower weight/importance to faulty samples compared to the clean/non-faulty samples [Zahedi et al. 2008]. In the case of data faults due to calibration errors, we can correct faulty samples if we can determine the proper calibration formula. Thus, it is not advisable to *always* discard/filter out the fault samples.

Next we discuss several enhancements to the methods presented in this paper. These enhancements can possibly not only improve the accuracy of these methods, but also enhance their applicability to other fault types. We plan to evaluate these enhanced methods as part of our future work.

Enhancements to the proposed methods. As mentioned in Section 3.2, the vector version of the LLSE method can incorporate spatial correlation across sensors attached to different nodes in computing the estimates for sensor values. Spatial correlations can provide a *global* view of the data collected by the network. In the absence of ground truth values, using the spatial correlation information from multiple sensors can result in a higher fidelity model or better estimates for sensor data, and hence, more accurate and robust fault detection. We expect the vector version of the LLSE method to perform better than the scalar version for fault detection that relies on leveraging inter-node relationships – for example, detecting faults due to calibration errors.

The ARIMA model based methods can be expanded to take measurements from different sensor(s) attached to same and/or different node as input while forecasting the expected sensor reading (refer to Chapter 11, Section 5 in [Box et al. 1994]). Such enhancements can enable the ARIMA model based methods to leverage spatial correlations (or inter-node relationships) as well as inter-sensor relationships, in addition to the temporal correlations. For example, in applying the ARIMA model based methods to a time series of temperature measurements, we can use measurements from the humidity sensor as an input. Temperature and humidity variations are known to be strongly correlated, and an ARIMA model that combines the two can perform better in situations where inter-sensor relationships (across different sensing modalities) are needed for fault detection. However, this enhanced ARIMA model is more complex and we need to estimate more model parameters. This increased complexity may result in a more computationally intensive training phase requiring a larger training dataset. Similarly, we can enhance our HMM based method to take measurements from other sensors as input [Bengio and Frasconi 1995].

8. SUMMARY AND CONCLUSIONS

In this paper, we focused on a simple question: How often are the sensor data fault types – SHORT, NOISE, and CONSTANT – observed in real deployments? To answer this question, we first explored and characterized four qualitatively different classes of fault detection methods (Rule-based, LLSE, time series forecasting, and HMMs) and then applied them to real world datasets. Several other methods based on Bayesian filters, neural networks, etc. can be used for sensor fault detection. However, the four methods discussed

in this paper are representatives of the larger class of these alternate techniques. Hence, an analysis of the four methods with injected faults presented in Section 4, not only demonstrates the differences, in terms of accuracy and robustness, between these methods, but can also help make an informed opinion about the efficacy of several other methods for sensor fault detection.

Fault type	SensorScope	INTEL Lab	GDI	NAMOS
SHORT	Infrequent (less than 0.01% samples affected), high intensity	Infrequent (only 5 nodes affected)	Infrequent, high intensity	None
NOISE, CONSTANT	None	Frequent (20-25% samples affected), spatiotemporally correlated	Infrequent	Frequent (15-35% samples affected)

Table VIII. Datasets: Prevalence of faults

We now summarize our main findings. Table VIII summarizes the fault prevalence for the different datasets. The prevalence of faults was lowest in the SensorScope dataset (less than 0.01% samples were affect by faults). However, in the INTEL Lab and NAMOS datasets a significant percentage (between 15-35%) of samples were affected by a combination of NOISE and CONSTANT faults. Such a high percentage of erroneous samples highlights the importance of automated, on-line sensor fault detection. In the GDI data set SHORT faults occurred once in two days but the faulty sensor values were often orders of magnitude higher than the correct value. Except for the INTEL Lab dataset, we found no spatial or temporal correlations among faults. In that dataset, the faults across various nodes were correlated because all the nodes ran out of battery power at approximately the same time.

Table IX and X summarize the evaluation results with injected faults. As discussed in Section 4, most of the methods work well for high and medium intensity SHORT faults, and high intensity and long duration NOISE faults. However, their performance is severely degraded in case of low intensity and/or short duration faults. In particular, the HMM and the ARIMA (L-step) methods are not suited for detecting short duration NOISE faults, while the LLSE and the ARIMA methods perform poorly at detecting low intensity SHORT faults. The observation that low intensity faults are harder to detect is not entirely unexpected. Our "data-centric" approach to fault detection is most effective

Method	High/Medium fault intensity	Low fault intensity	False positives
SHORT rule	works well	detects at least 50% of faults	No
LLSE	works well	performs poorly	No
HMM	works well	detects more faults than SHORT rule and LLSE	Yes (for low fault intensity)
ARIMA (One-step)	works well	detects at least 50% of faults	Yes (high rate of false positives)
ARIMA (L-step)	works well	performs poorly	Yes (high rate of false positives)

Table IX. Detection methods: Performance on injected SHORT faults

Method	High/Medium fault intensity	Low fault intensity	Fault duration	False positives
NOISE rule	works well	performs poorly	robust to changes in fault duration	No
LLSE	works well	performs poorly	more suited for long duration faults	Yes (for low intensity and/or short duration faults)
HMM	works well	detects more faults than NOISE rule and LLSE	not suited for low intensity and short duration faults	Yes (high false positive rate)
ARIMA (One-step)	No significant impact on performance		Avg. performance for short duration faults; not suited for long duration faults	Yes
ARIMA (L-step)	reasonable performance	performs poorly	reasonable performance for long duration faults; not suited for short duration faults	Yes

Table X. Detection methods: Performance on injected NOISE faults

when the faulty samples differ significantly from normal sensor readings. This is not always the case for low intensity faults. Two important observations can be made based on the evaluation of the fault detection methods with injected faults: (i) the four classes of methods sit at different points on the accuracy/robustness spectrum, and no single method is perfect for detecting the different types of faults; (ii) hybrid methods can help eliminate false positives or false negatives, and using two methods in sequence can detect more low intensity faults at the expense of slightly higher false positives.

The fault detection methods performed well on the real world datasets except for the ARIMA model based methods when used on the INTEL Lab, Berkeley dataset. This is because most of the datasets experienced high intensity faults.

Even though we analyzed most of the publicly available real world sensor datasets for faults, it is hard to make general statements about sensor faults in real world deployments based on just four datasets. However, our results raise awareness of the prevalence and severity of the problem of data corruption and can inform future deployments. Overall, we believe that our work opens up new research directions in automated high-confidence fault detection, fault classification, data rectification, and so on. More sophisticated statistical and learning techniques than those we have presented can be brought to bear on this crucial area.

REFERENCES

BALZANO, L. AND NOWAK, R. 2007. Blind Calibration in Sensor Networks. In *Proceedings of the Intenational Conference on Information Processing in Sensor Networks (IPSN)*.

BENGIO, Y. AND FRASCONI, P. 1995. An Input Output HMM Architecture. In *Proceedings of the Neural Information Processing Systems Conference (NIPS)*.

BOX, G. E. P., JENKINS, G. M., AND REINSEN, G. C. 1994. *Time Series Analysis: Forecasting and Control, 3rd Edition*. Prentice Hall.

BYCHKOVSKIY, V., MEGERIAN, S., ESTRIN, D., AND POTKONJAK, M. 2003. A Collaborative Approach to In-Place Sensor Calibration. In *Proceedings of the 2nd Intenational Workshop on Information Processing in Sensor Networks (IPSN)*.

CHATFIELD, C. 2000. *Time Series Forecasting*. Chapman and Hall/CRC Press.

ELNAHRAWY, E. AND NATH, B. 2003. Cleaning and Querying Noisy Sensors. In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*.

GUPCHUP, J., SHARMA, A., TERZIS, A., BURNS, R., AND SZALAY, A. 2008. The Perils of Detecting Measurement Faults in Environmental Monitoring Networks. In *Proceedings of the ProSense Special Session and International Workshop on Wireless Sensor Network Deployments (WiDeploy), held at DCOSS*.

INTEL. 2004. The Intel Lab Data. Data set available at: `http://berkeley.intel-research.net/labdata/`.

JEFFERY, S. R., ALONSO, G., FRANKLIN, M. J., HONG, W., AND WIDOM, J. 2006. Declarative Support for Sensor Data Cleaning. In *Proceedings of the International Conference on Pervasive Computing*.

KAILATH, T., Ed. 1977. *Linear Least-Squares Estimation*. Hutchison & Ross, Stroudsburg, PA.

KHOUSSAINOVA, N., BALAZINSKA, M., AND SUCIU, D. 2006. Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints. In *Proceedings of the ACM Workshop on Data Engineering and Mobile Access (MobiDE)*.

KOUSHANFAR, F., POTKONJAK, M., AND SANGIOVAMMI-VINCENTELLI, A. 2003. On-line Fault Detection of Sensor Measurements. In *IEEE Sensors*.

MAINWARING, A., POLASTRE, J., SZEWCZYK, R., AND ANDERSON, D. C. J. 2002. Wireless Sensor Networks for Habitat Monitoring . In *the ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*.

MATLAB. `http://www.mathworks.com/`.

NAMOS. 2005. NAMOS: Networked Aquatic Microbial Observing System. Data set available at: `http://robotics.usc.edu/~namos/data/jr_oct/web/`.

NAMOS. 2006. NAMOS: Networked Aquatic Microbial Observing System. Data set available at: `http://robotics.usc.edu/~namos/data/jr_aug_06/`.

NI, K., RAMANATHAN, N., CHEHADE, M., BALZANO, L., NAIR, S., ZAHEDI, S., POTTIE, G., HANSEN, M., AND SRIVASTAVA., M. 2008. Sensor Network Data Fault Types. *Transactions on Sensor Networks*. to appear.

RABINER, L. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of IEEE 77(2)*, 257–286.

RAMANATHAN, N., BALZANO, L., BURT, M., ESTRIN, D., KOHLER, E., HARMON, T., HARVEY, C., JAY, J., ROTHENBERG, S., AND SRIVASTAVA, M. 2006. Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks. Tech. Rep. 62, CENS. April.

RAMANATHAN, N., SCHOELLHAMMER, T., ESTRIN, D., HANSEN, M., HARMON, T., KOHLER, E., AND SRIVASTAVA., M. 2006. The Final Frontier: Embedding Networked Sensors in the Soil. Tech. Rep. 68, CENS. November.

SAS. The SAS Forecasting Software. `http://www.sas.com/technologies/analytics/forecasting/index.html`.

SENSORSCOPE. 2006. The SensorScope Lausanne Urban Canopy Experiment (LUCE) Project. Data set available at: `http://sensorscope.epfl.ch/index.php/LUCE`.

SHARMA, A. B., GOLUBCHIK, L., AND GOVINDAN, R. 2007. On the Prevalence of Sensor Faults in Real-World Deployments. In *Proceedings of the IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*.

SZEWCZYK, R., POLASTRE, J., MAINWARING, A., AND CULLER, D. 2004. Lessons from a sensor network expedition. In *Proceedings of the First European Workshop on Sensor Networks (EWSN)*.

TOLLE, G., POLASTRE, J., SZEWCZYK, R., CULLER, D., TURNER, N., TU, K., BURGESS, S., DAWSON, T., BUONADONNA, P., GAY, D., AND HONG, W. 2005. A Macroscope in the Redwoods. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*. ACM Press, New York, NY, USA, 51–63.

TULONE, D. AND MADDEN, S. 2006. PAQ: Time series forecasting for approximate query answering in sensor networks. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*.

WERNER-ALLEN, G., LORINCZ, K., JOHNSON, J., LEES, J., AND WELSH, M. 2006. Fidelity and Yield in a Volcano Monitoring Sensor Network. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*.

ZAHEDI, S., SZCZODRAK, M., JI, P., MYLARASWAMY, D., SRIVASTAVA, M. B., AND YOUNG, R. 2008. Tiered Architecture for On-Line Detection, Isolation and Repair of Faults in Wireless Sensor Networks. In *Proceedings of the MILCOM*.