Applying Time Series Analysis and Neighbourhood Voting in a Decentralised Approach for Fault Detection and Classification in WSNs

Tuan Anh Nguyen, Doina Bucur, Marco Aiello
University of Groningen
Groningen, The Netherlands
{t.a.nguyen,d.bucur,m.aiello}@rug.nl

Kenji Tei
National Institute of Informatics
Tokyo, Japan
tei@nii.ac.jp

ABSTRACT

In pervasive computing environments, wireless sensor networks play an important infrastructure role, collecting reliable and accurate context information so that applications are able to provide services to users on demand. In such environments, sensors should be self-adaptive by taking correct decisions based on sensed data in real-time in a decentralised manner; however, sensed data is often faulty. We thus design a decentralised scheme for fault detection and classification in sensor data in which each sensor node does localised fault detection. A combination of neighbourhood voting and time series data analysis techniques are used to detect faults. We also study the comparative accuracy of both the union and the intersection of the two techniques. Then, detected faults are classified into known fault categories. An initial evaluation with SensorScope, an outdoor temperature dataset, confirms that our solution is able to detect and classify faulty readings into four fault types, namely, 1) random, 2) malfunction, 3) bias, and 4) drift with accuracy up to 95%. The results also show that, with the experimental dataset, the time series data analysis technique performs comparable well in most of the cases, whilst in some other cases the support from neighbourhood voting technique and histogram analysis helps our hybrid solution to successfully detects the faults of all types.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Distributed Systems—*Wireless sensor networks, Fault tolerance*

General Terms

Algorithms, Experimentation, Performance

Keywords

Wireless Sensor Networks, Decentralised Fault Tolerance, Online Sensory Data Fault Handling

1. INTRODUCTION

In pervasive computing environments such as smart cities, wireless sensor networks (WSNs) play an important role as the ideal infrastructure of context-aware systems, by providing sensors, actuators, and wireless communication. Thousands of sensors and actuators are deployed over the area of interest, e.g., the entire city, gathering necessary context information as well as providing services to users on demand and in real-time. In such environments, a WSN has to satisfy the following requirements:

- **Large-scale deployment:** Deployed sensor networks may contain hundreds or thousands of sensor nodes.
- **Accuracy and reliability:** WSNs monitor the phenomena of interest; it is crucial that the data reported should be as accurate as possible in order to provide the correct services to users at the correct time.
- **Real-time:** The feedback loops that represent a key feature of adaptive systems, allowing the system to respond to changes in the surrounding environments, should happen in real-time.
- **Self-responsibility:** To be self-adaptive and to provide services in a real-time manner, each of the sensors should be self-responsible for its own sensing data, as well as for its decisions, instead of being dependent on centralized data processing.

However, sensor data usually suffers from faults due to (1) tight resource constraints on sensor nodes, which are battery-powered and have limited memory and computational capacity, and also due to (2) the harsh environments where the sensors are deployed. Thus, to ensure the accuracy of sensor data in large-scale deployments, while still satisfying the real-time requirements of online sensory data fault correction, a *decentralised scheme for data fault detection and classification* is required, to be executed locally on each sensor node in the network. When designing a decentralised technique, the challenge is to process the maximal amount of data while achieving accuracy rates in fault detection comparable to centralized methods, and maintaining communication overhead, memory and computational costs low [20].

In this paper, motivated by this challenge of decentralisation, we present a hybrid approach for the detection and classification of sensor data faults. Each node handles its own data processing and decision-making locally in a distributed manner. In our proposed solution, a *neighbourhood voting* technique and the *AutoRegressive–Moving-Average (ARMA)* model for time-series data forecasting are used in combination to detect sensor data faults. A sensor reading is identified as faulty by comparing its value with 1) the value computed by neighbourhood voting, and 2) the value predicted, based on past local readings, by the ARMA time series forecasting model. The final decision may be based on either the intersection or the union of the two algorithms.

The neighbourhood voting technique is adopted as it does not require a priori knowledge of the phenomenon. Instead, this technique takes advantage of the possible redundancy in measurements of sensor readings in the WSN. The ARMA model is chosen because it has been shown to be a strong candidate for time series predictions both theoretically and experimentally [6]. More importantly, ARMA is considered a lightweight, because the parameters of the model can be adapted to the underlying time series in an online fashion, without the need to store large sets of past data [12].

For fault classification, in order to classify detected faults we use a fault model based on the frequency and continuity of fault occurrence, and on the observable and learnable patterns that faults leave in the data. This complete and consistent model allows us to classify sensor data faults into four fault types, namely, 1) random, 2) malfunction, 3) bias, and 4) drift.

The initial evaluation results with SensorScope [1], an outdoor temperature dataset, confirm that our solution is able to detect and classify faulty readings into the four fault categories, with an accuracy between 85% and 95%. The results also show that, with the experimental dataset, the time series data analysis technique performs comparable well in most of the cases, whilst in some other cases the support from neighbourhood voting technique and histogram analysis helps our hybrid solution to successfully detects the faults of all types. These results suggest that decentralised schemes for online sensory data fault handling are promising and should be investigated further.

This paper is an extension of an initial, work-in-progress contribution [13]. The remainder of this paper is organised as follows. In Section 2 we discuss related work on fault tolerance in WSNs. In Section 3 we discuss methodologies for fault categorisation, as well as our own approach for fault modelling. Section 4 details our proposed decentralised scheme for fault detection and classification. Experimental results with an outdoor temperature dataset of 10 sensors are shown and explained in Section 5, followed by conclusions and future work in Section 6.

2. RELATED WORK

Fault tolerance is a fundamental requirement to ensure the reliability and accuracy of sensor data. Fault tolerance approaches can be broadly divided into two major categories: 1) centralised, and 2) decentralised.

Centralised mechanisms take advantage of the computational capacity of a dedicated station for data analysis. Given these computational resources, automated and machine-learning approaches are usually employed: Hidden Markov Models (HMMs) are applied in [17], decision trees in [4], support vector machine classifiers in [8] However, the disadvantages of centralised approaches are that a) they require high computational resources for data processing, and b) the data to be processed (and the results of the processing) must be transmitted over the network, adding network overhead and time delays, and likely violating the real-time requirements. In addition, centralised techniques do not scale well to large distributed data streams. We aim at a solution that achieves comparable accuracy yet still satisfies the requirements of online sensor data processing.

Decentralised mechanisms specifically aim at real-time fault tolerance in streaming data, in which nodes handle their own faulty readings locally and flexibly, without the delays incurred by processing data at a remote station. Neighbourhood voting technique is extensively covered by literature, such as [9, 19, 10]. These techniques, however, have a shortcoming: the detection accuracy decreases rapidly when the number of neighbour nodes is small and the nodes' failure ratio is high. In this paper, neighbourhood voting technique is used in combination with the lightweight ARMA time series forecasting model, so that the overall accuracy of the fault detection is significantly enhanced.

Time series data models are applied in [16] and [18]. In [16], the authors use the autoregressive integrated moving average (ARIMA) model in a centralised scheme. This model is more complex and incurs more computation overhead, as well as requiring more historical data to be stored. In [18], the authors use a piecewise linear regression technique in a decentralised manner. In their work, the data is represented and compressed as a sequence of linear pieces that are fitted with linear least squares estimation. When a new data point deviates too much from the existing piece, a new piece is created. This method, however, does not take into account the correlations between different sensors.

All the above mentioned works are using a clustering-based architecture that is only partially decentralised, as the fault handling processing still relies on a cluster head for more computational capacity. In our paper, we aim at proposing a completely decentralised solution that can be run on each sensor node. The questions we are trying to answer is how to process as much data as possible while keeping the communication overhead, memory and computational cost low.

3. SENSOR DATA FAULT MODELLING

Sensor data may suffer from faults due to (1) tight resource constraints on sensor nodes, which are battery-powered and have limited memory and computational capacity, and also due to (2) the harsh environments where the sensors are deployed. As the first step of fault management, it is crucial to categorise faults. By comprehending the causes, effects, and especially the characteristics of each fault type, it is possible to propose suitable fault-tolerance mechanisms to detect, classify, and correct faults of each type.

3.1 Sensor Network Data Fault Types

Fault categorisation may vary with different points of view. Several existing fault taxonomies use different criteria, such as a fault cause, impact, or duration. One can also categorise faults based on the layer of the network stack where the fault occurs. For example, at the physical layer we may have *random noise*, *malfunctioning* or, most commonly, *calibration systematic errors* [5]. In terms of duration, faults can be classified as *permanent*, *intermittent* or *transient*.

Ni et. al. give extensive taxonomies of data faults that cover definition, cause, duration and impact of faults [14]. According to the authors, sensor network faults can be classified into two broad fault types: 1) *system faults* and 2) *data faults*. From a system-centric viewpoint, faults may be caused by *calibration*, *low battery*, *clipping*, or an *environment out of range* situation. On the other hand, data faults comprise *stuck-at*, *offset*, and *gain* faults. These three types of data faults are named *short*, *constant*, and *noise*, respectively, by Sharma et al. in [16].

3.2 Our Fault Modelling

Baljak et al. [3] propose a complete and consistent categorisation based on the **the frequency and continuity of fault occurrence** and on **observable and learnable patterns** that faults leave on the data. Their approach shares the same point of view with ours, thus we use these fault taxonomies in our research. We believe that this categorisation is flexible and applicable to a wide range of sensor readings. The underlying cause of the error does not affect this categorisation, which makes it possible to handle the faults based on their patterns of occurrence on each sensor node. Figure 1 presents our fault modelling and categorisation.

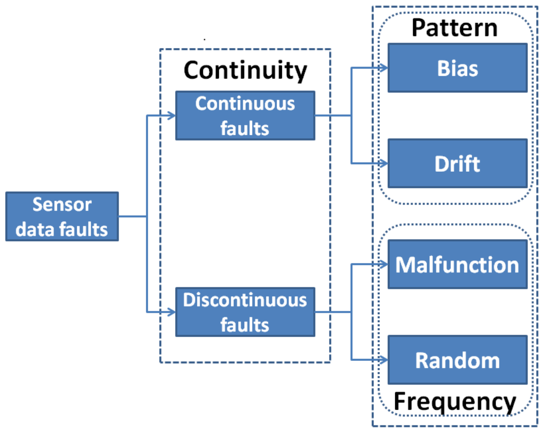
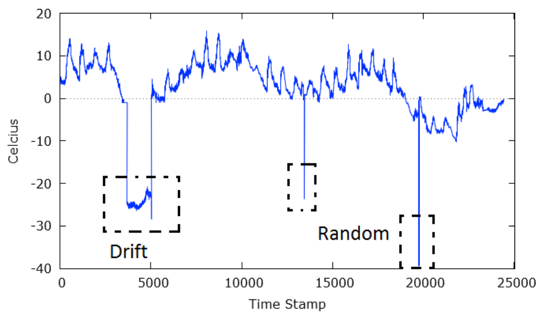


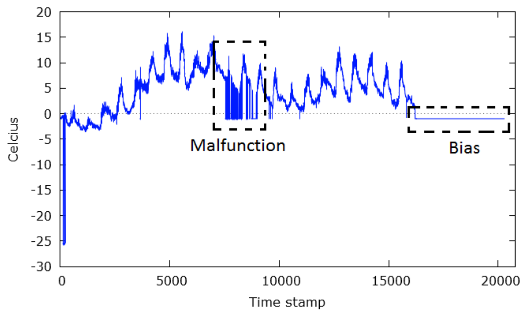
Figure 1: Sensor data fault modelling

We model data faults as following. Examples of these fault types are given in Figure 2.

- **Discontinuous** – Faults occur from time to time, and the occurrence of faults is discrete.
  - **Malfunction** – Faulty readings appear frequently. The frequency of the occurrences of faults is higher than a threshold τ.
  - **Random** – Faults appear randomly. The frequency of the occurrences of faulty readings is smaller than τ.
- **Continuous** – During the period under observation, a sensor returns constantly inaccurate readings, and it is possible to observe a pattern in the form of a function.
  - **Bias** – The function of the error is a constant. This can be a positive or a negative offset.
  - **Drift** – The deviation of data follows a learnable function, such as a polynomial change.



(a) Drift and Random faults at SensorScope 19



(b) Malfunction and Bias faults at SensorScope 29

Figure 2: Examples of random, malfunction, bias, and drift faults in SensorScope dataset

In addition, our approach towards fault categorisation and those of Ni et al. [14] and Sharma [16] do overlap. The fault types categorised by our approach can be mapped, as depicted in Table 3.2, into one fault or a combination of faults defined using the other approaches.

Table 1: Relationships between fault models

Baljak	**Ni**		**Sharma**
	Data-centric	**System-centric**	
Random	Outlier		Short
Random Malfunction	Spike	Connection/Hardware Low Battery	Short
Bias	Stuck-at	Clipping Connection/Hardware Low Battery	Constant
Drift	Noise	Low Batter Connection/Hardware Env. out of range	Noise
Bias Drift		Calibration	Noise

By understanding fault types, each node is able to handle its faulty readings appropriately, according to the type of fault. Random faults may be discarded as they contain no meaningful information. Malfunctioning nodes may be removed from the network. Bias and drift faults are more interesting: if detected accurately, a method may be designed to correct the faults in real-time.

4. FAULT HANDLING APPROACH

The overall fault-handling process is presented in Figure 3. In practice, faulty readings are detected and classified by the same algorithm; however, for clarity, in the figure we illustrate the detection and classification as two separate phases.

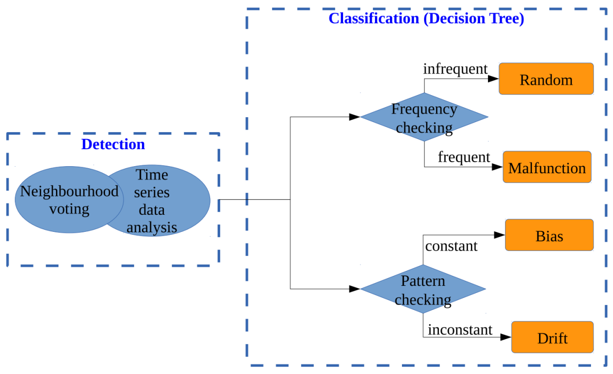


Figure 3: Fault handling process

Our solution uses a combination of neighbourhood voting and time series data analysis techniques. For fault classification, we take into account the duration and continuity of faults in sensor readings. In other words, we focus on how long and how often faults appear in the observation period. The classification algorithm is designed based on the fault model that we discussed in Section 3.2. In the following, we describe in more detail our hybrid solution, which can be implemented on each sensor node, providing the nodes with the ability to not only check the correctness of readings, but also to classify faults.

4.1 Hybrid Fault Detection

At the detection phase, a sensor node compares its current reading with 1) the value computed by neighbourhood voting, and 2) the expected value previously forecast by the time series data forecasting model. The correctness of the reading is decided based on either the intersection or the union of the two methods. The former technique decreases the rate of false positives, while the latter decreases the rate of false negatives. The result of the detection phase is the fault status of the readings examined.

4.1.1 Neighbourhood Voting

The assumptions behind our neighbourhood voting technique are that 1) neighbourhood nodes are deployed closely together, thus the distance between them is a single-hop transmission, 2) the nodes monitor the same phenomenon, and 3) faults at each node develop unrelatedly. In a nutshell, neighbourhood voting technique expects that the majority of the sensors report the true value of the monitored phenomenon, thus a specific node can rely on its neighbours to check the validity of its readings. With neighbourhood voting, the system does not require a priori knowledge about the environment. Instead, it takes advantage of the redundancy in measurements of sensor readings.

Given a node S_i, its reading is denoted r_i, and the set of its neighbours is denoted by $Neighbour(S_i)$. The number of neighbours is denoted as $|Neighbour(S_i)|$. The neighbourhood voting technique used at each sensor node S_i consists of the following steps:

1. Collect the set of readings $R = r[1 \ldots |Neighbour(S_i)|]$ from all neighbours, excluding its own reading r_i.
2. Calculate the median of the group, $\mu = \{R\}_{\frac{1}{2}} = \tilde{r}$.
3. Calculate the difference between r_i and $\tilde{r}$,
$$D_{r_i\tilde{r}} = |r_i - \tilde{r}|$$
4. Compare the difference $D_{r_i\tilde{r}}$ with a threshold τ, that can be adjusted (usually set at $\tau = 0.2 * \tilde{r}$).
    - If $D_{r_i\tilde{r}} < \tau$ then r_i is a good reading,
    - If $D_{r_i\tilde{r}} \geq \tau$ then r_i is a faulty reading.

4.1.2 Time Series Analysis

Sensor measurements are the observations of a well-defined phenomena monitored. The observations are obtained periodically over time. In addition, the measurements exhibit a temporal correlation between consecutive observations. Thus, sensor data is a time series, and time series data analysis could be used for fault detection. The idea is to use a suitable time series forecasting model to predict the sensor reading at time t based on $t-k$ known previous readings. When the current observation at time t is available, it is compared against its predicted value to determine if it is faulty. In this paper, we choose the ARMA model for two reasons. First, the model has been shown to be a good candidates for time series prediction [6]. Second, ARMA is considered a lightweight technique, because the parameters of the model can be estimated using the recursive least square (RLS) algorithm [2], which allows to adapt the parameters of the underlying time series online, without the need to store large sets of past data.

AutoRegressive–Moving Average model.

The notation $ARMA(p, q)$ refers to the model with p autoregressive terms and q moving-average terms. This model contains the $AR(p)$ and $MA(q)$ models. We employ the $ARMA(p, q)$ model

$$\begin{aligned} X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} \\ + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - ... - \theta_q a_{t-q} \end{aligned} \quad (1)$$

In practice, the autoregressive and the moving average models of first order ($p = 1$), ($q = 1$), and of second order ($p = 2$), ($q = 2$) are of considerable practical importance [6]. Thus in our implementation, we use the $ARMA(2, 2)$ model of second order ($p = 2, q = 2$).

Parameter Estimation.

We use the exact maximum likelihood (ML) computational method [7] in order to estimate the two sets of parameters of the model, $\phi_1, \phi_2, ..., \phi_p$ and $-\theta_1, -\theta_2, ..., -\theta_q$, using training data. The parameter estimation phase is performed at a base station by using confirmed good readings from each sensor. After that, the ARMA model with determined parameters is implemented on sensor nodes. The correct estimation of these parameters can and should be done before the actual deployment of the sensors, in the sensor calibration phase.

At runtime, the base station that collects data from all sensors can re-perform the parameter estimation periodically for each sensor and can send updated parameters to the nodes. This way, the ARMA model at sensor nodes can adapt with the change in the phenomenon monitored.

Fault detection with L-step ahead prediction.

To detect faults in a sensor measurement time series, we first forecast the sensor readings at time $t+i$, $1 \leq i \leq L$ with $L > 1$, using measurements up to time t based on our *ARMA* model. We then compute the difference between actual sensor measurements at times $t+i$ and their predicted values, and flag the measurement as faulty if this difference is above a threshold δ, called *confidence interval* and usually set at 95% of the forecasted value. However, one should notice that the potential error in forecast grows with L [6].

Keeping the values of the last time-window of good readings.

Readings at time $t+i$ are forecast using measurements from time $t-i$ to time t. Therefore, if these measurements are faulty, the predicted values from t to $t+i$ are consequently faulty. To prevent this, at runtime we preserve the values of the last time window of good readings as the reference values for prediction. This means that, if measurements from t to $t+i$ are detected as faults, the readings from $t-i$ to t, that are detected as good readings, are used to predict values from $t+i$ to $t+2i$. Otherwise, readings from $t-i$ to t are discarded and replaced with the ones from t to $t+i$ for further prediction.

Alternatives to ARMA model.

The choice of a time series model for sensor measurements is determined by the nature of the phenomenon being measured. Our experimental results with a real-world dataset (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 modelling different phenomena is the focus of our future work.

4.2 Fault Classification Algorithm

Once a sensor node exhibits faulty measurements, the classification process checks the frequency and continuity of the occurrence of faults in order to identify the fault type, i.e., either random, malfunction, bias, or drift. One should notice that, while a sensor reading is checked for correctness immediately, i.e., precisely at the time when the reading is observed, the classification process runs periodically, i.e., after every T readings, in order to be able to check the frequency and continuity of the fault occurrence. The number of readings T, and thus the time interval for fault classification, is adjustable to meet the real-time requirement. The classification process, illustrated in Figure 3, is now clarified in Algorithm 1.

The algorithm takes, as its inputs, the readings $R[1..T]$ within the checked interval, together with the state (i.e., good or faulty) of the readings. First, the occurrences of faults $|\varepsilon_i|$ in R are computed. Next, the algorithm checks the continuity and the frequency of the fault occurrences $|\varepsilon_i|$ in order to classify the type of the fault found in the observed interval into one of four fault types (random, malfunction, bias, drift).

Algorithm 1 Fault Classification

```
Input: 1) $R[1..T]$: vector of $T$ sensor readings
       2) $S[1..T]$: vector of the faulty state of $R[1..T]$
Output: C: fault type of sensor node in the interval
 1: Compute the occurrences of faults $|\varepsilon_i|$ in $R$
 2: check the continuity
 3: if $\varepsilon_i$ is discrete then
 4:   Check the frequency
 5:   if $|\varepsilon_i| \geq \tau$ then
 6:     $C = Malfunction$
 7:   else
 8:     $C = Random$
 9:   end if
10: end if
11: if $\varepsilon_i$ is continuous then
12:   Check the fault function $\varepsilon_i$
13:   if $\varepsilon_i = const$ then
14:     $C = Bias$
15:   else
16:     $C = Drift$
17:   end if
18: end if
19: return $C$
```

Real-time fault handling.

To achieve real-time fault detection and classification, we run the fault handling process periodically, say every T minutes. For example, if $T = 30$ minutes, we first collect new sensor readings for half an hour and then perform anomaly detection using the framework described above. The fault detection interval, T, controls the trade-off between real-time fault handling and resource consumption.

Histogram analysis to check the distribution of ε_i.

A histogram represents the frequency of occurrence by classes of data [15]. In our case, a histogram will show how faulty readings are distributed. In order to check if the fault type is bias, we can check the histogram of the readings $R[1..T]$. If the frequency of occurrence of fewer than c classes of data is higher than τ, we say that the fault is bias, otherwise the fault type is drift. In practice, c usually is 2 or 3, while τ is assigned by 0.8.

5. PRELIMINARY EVALUATION RESULTS

5.1 Dataset

We conduct preliminary analysis on the well known sensor dataset SensorScope [1]. In the SensorScope project, large networks of sensors are deployed to collect environmental data such as temperature, humidity, and solar radiation. In this paper, we use temperature readings collected from 23 sensors deployed in the Grand St. Bernard pass between Switzerland and Italy in 2007. Each sensor collected samples every two minutes for 43 days. In what follows, we select 10 sensor nodes which suffer from various faulty readings of more than one type. The 10 analysed nodes are 2, 6, 7, 9, 15, 17, 18, 19, 20, and 29. 13 other sensor nodes are not selected for analysis because their readings are almost correct, with the exception of only few data points which exhibit random faults. We denote the time series data of each of these 10 sensors as *SensorScope nodeID*. Figure 4 visually illustrates various faults of all four types that are exhibited in the time series data of the 10 sensors.

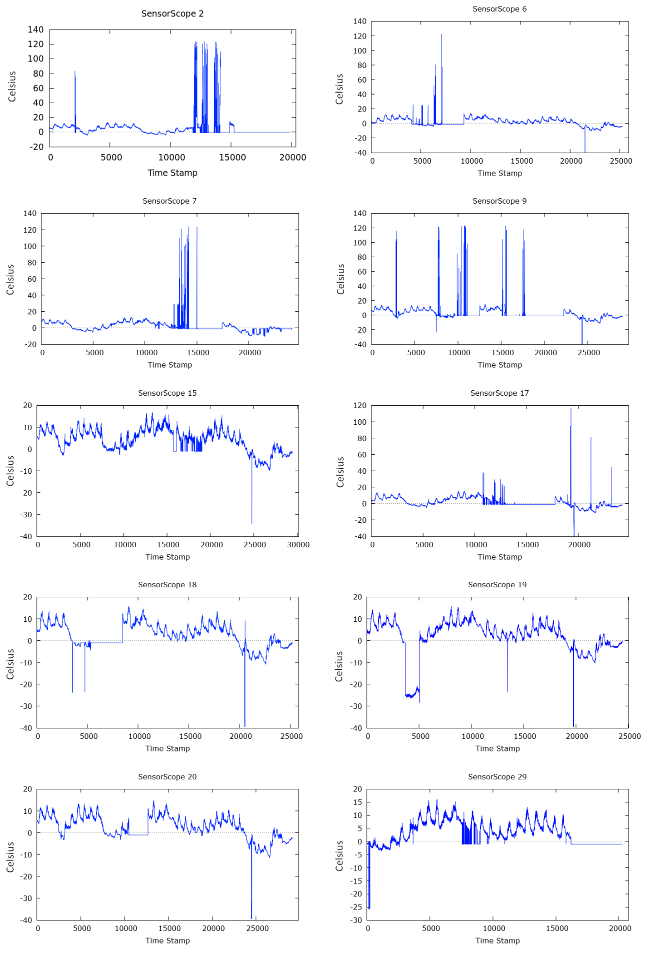


Figure 4: Actual readings from 10 experimental sensor nodes

Ground truth of faults in the dataset.
To the best of our knowledge, the dataset does not provide fault annotation. Thus, we visually inspect the SensorScope time series to identify the characteristics of correct readings, as well as those of random, malfunction, bias, and drift faults. Then, to obtain the ground truth, we first run a script to roughly annotate the data; afterwards, we manually double check to ensure that the ground truth is as precise as possible. Our way of building ground truth is similar to and consistent with current practice, such as [18, 11], for datasets that lack ground truth.

Our observations show that discrete faults (i.e., random and malfunction) occur widely in sensor data, even at the early stages of sensor deployment. On the other hand, continuous faults (bias and drift) do not appear that frequently and in some cases, bias faults occur in the late stages of sensor deployment and the faults remain until sensor nodes physically malfunction or die.

Checking interval.
A checking interval consists of T readings. As SensorScope sensors collect samples every two minutes, we set the checking interval to $T = 30$, which means that readings are checked against faults every hour. We focus on detecting whether the sensor data within that period is faulty and what the fault type is, rather than trying to identify every single fault.

5.2 Experimental Results

We evaluate the accuracy of our solution using the dataset and the ground truth identification described above. We use

1. FP - the number of false positives,
2. TP - the number of true positives, and
3. the overall success rate for each fault type

$$SR = \frac{\sum_{FaultType} TP}{\sum_{FaultType} Faults}$$

as our metrics. Specifically, the results in Tables 2 to 5 are presented as follows – x/y indicates that x out of y faults are detected correctly (corresponding to y true positives), and we also indicate separately the number of corresponding false positives. Note that a continuous fault (bias or drift) may consist of many consecutive data points; we focus on detecting these events rather than on identifying every faulty data point within the event.

Table 2 and Table 3 show the results obtained with neighbourhood voting (NV) and time series analysis (TS), respectively. The first impression is that both neighbourhood voting and time series analysis achieve good detection accuracy over the SensorScope dataset for all four types of faults. The success rates achieved range between 80% (time series analysis for drift faults) and 92.6% (time series analysis for bias faults).

Neighbourhood voting has accurate detection results because, with our experimental dataset, the number of neighbours is sufficiently large, and the nodes' failure ratio is low. Nevertheless, neighbourhood voting does worse than time series analysis, because in some cases several neighbours of a node also have faulty readings, especially in the late state of the sensor's lifetime. On the other hand, experimental results suggest that time series analysis, the ARMA model in particular, is not able to detect bias and drift faults in some cases. However, with support from neighbourhood voting and from histogram confirmation, our hybrid solution successfully detects fault events of these types.

The results obtained from the $NV \cap TS$ intersection are shown in Table 4, while Table 5 reflects the results obtained with the $NV \cup TS$ union. Clearly, the $NV \cap TS$ helps to decrease false positives and computational overhead of the method, while the success rate reduces slightly. Vice versa, the $NV \cup TS$ decreases false negatives, and thus increases the success rates and the overhead, and increases fault positives. This conclusion is summarised in Table 6, and is also illustrated in Figure 5.

6. CONCLUSIONS AND FUTURE WORK

Sensor data may suffer from faults. To ensure the accuracy and reliability of sensor data in large-scale deployments,

Table 2: Results of neighbourhood voting

Sensor node	Random		Malfunction		Bias		Drift	
	TP	FP	TP	FP	TP	FP	TP	FP
SensorScope 2	4/4	0	42/53	1	180/200	2	3/5	0
SensorScope 6	2/4	1	4/4	1	67/70	1	0/0	0
SensorScope 7	5/8	1	15/18	1	91/103	3	4/5	0
SensorScope 9	5/5	1	14/17	3	133/148	7	0/0	0
SensorScope 15	1/1	0	75/81	0	4/5	0	0/0	0
SensorScope 17	4/5	0	69/76	1	137/157	2	15/19	0
SensorScope 18	4/4	0	13/15	1	18/18	1	13/16	1
SensorScope 19	4/4	0	2/2	0	0/0	0	39/46	5
SensorScope 20	2/3	0	4/6	0	68/73	0	0/0	0
SensorScope 29	7/7	0	46/51	0	125/134	0	0/0	0
Total	38/45	3	284/323	8	823/908	16	74/91	6
SR (%)	**84.4**		**87.9**		**90.6**		**81.3**	

Table 3: Results of time series analysis

Sensor node	Random		Malfunction		Bias		Drift	
	TP	FP	TP	FP	TP	FP	TP	FP
SensorScope 2	4/4	0	48/53	2	183/200	2	3/5	0
SensorScope 6	3/4	1	4/4	0	69/70	2	0/0	0
SensorScope 7	7/8	1	14/18	2	96/103	6	4/5	1
SensorScope 9	5/5	1	15/17	4	137/148	9	0/0	0
SensorScope 15	1/1	0	76/81	0	4/5	0	0/0	0
SensorScope 17	4/5	1	71/76	3	143/157	3	14/19	0
SensorScope 18	4/4	0	14/15	2	14/18	0	11/16	1
SensorScope 19	4/4	0	2/2	0	0/0	0	41/46	5
SensorScope 20	2/3	0	5/6	0	73/73	0	0/0	0
SensorScope 29	7/7	0	49/51	1	122/134	2	0/0	0
Total	41/45	4	298/323	14	841/908	24	73/90	7
SR (%)	**91.1**		**92.3**		**92.6**		**80.2**	

while still satisfying the realtime requirements of any online data fault correction mechanism, we contribute a decentralised scheme for fault detection and classification, suitable for embedded implementation at each sensor node. As the first step of designing an online fault-handling framework, we propose a lightweight decentralised scheme for fault detection and classification.

Our proposed solution applies a combination of both a neighbourhood voting mechanism and time series data analysis to detect sensor data faults. A sensor reading is compared with not just the value computed with a neighbourhood voting technique, but also is double checked with the value previously forecast by the ARMA time series data analysis model. Then, the detected faulty readings are classified based on the frequency and continuity of fault occurrence and the observable and learnable patterns that faults

Table 4: Results of the intersection $NV \cap TS$

Sensor node	Random		Malfunction		Bias		Drift	
	TP	FP	TP	FP	TP	FP	TP	FP
SensorScope 2	4/4	0	40/53	0	180/200	0	3/5	0
SensorScope 6	2/4	1	4/4	0	67/70	1	0/0	0
SensorScope 7	5/8	1	13/18	0	89/103	2	4/5	0
SensorScope 9	5/5	0	14/17	1	130/148	3	0/0	0
SensorScope 15	1/1	0	75/81	0	4/5	0	0/0	0
SensorScope 17	4/5	0	68/76	1	135/157	1	14/19	0
SensorScope 18	4/4	0	13/15	1	14/18	0	11/16	1
SensorScope 19	2/3	0	2/2	0	0/0	0	38/46	4
SensorScope 20	7/7	0	4/6	0	68/73	0	0/0	0
SensorScope 29	38/45	2	45/51	0	120/134	0	0/0	0
Total	38/45	2	278/323	3	807/908	7	70/91	5
SR (%)	**84.4**		**86.1**		**88.9**		**76.9**	

Table 5: Results of the union $NV \cup TS$

Sensor node	Random		Malfunction		Bias		Drift	
	TP	FP	TP	FP	TP	FP	TP	FP
SensorScope 2	4/4	0	50/53	0	189/200	0	3/5	0
SensorScope 6	3/4	1	4/4	1	69/70	2	0/0	0
SensorScope 7	7/8	1	16/18	3	98/103	7	4/5	1
SensorScope 9	5/5	2	15/17	6	140/148	13	0/0	0
SensorScope 15	1/1	0	76/81	0	4/5	0	0/0	0
SensorScope 17	4/5	1	72/76	3	145/157	1	15/19	0
SensorScope 18	4/4	0	14/15	2	18/18	1	13/16	1
SensorScope 19	4/4	0	2/2	0	0/0	0	42/46	6
SensorScope 20	2/3	0	5/6	0	73/73	1	0/0	0
SensorScope 29	7/7	0	50/51	1	127/134	2	0/0	0
Total	41/45	5	304/323	16	863/908	26	77/90	8
SR (%)	**91.1**		**94.1**		**95.0**		**84.6**	

Table 6: Overall success rates and fault positives

Method	Fault type							
	Random		Malfunction		Bias		Drift	
	SR	FP	SR	FP	SR	FP	SR	FP
NV	84.4	3	87.9	8	90.1	16	81.3	6
TS	91.1	4	92.2	14	92.7	24	80.2	7
Intersection	84.4	2	86.1	3	88.9	7	76.9	5
Union	91.1	5	94.1	16	95.0	26	84.6	8

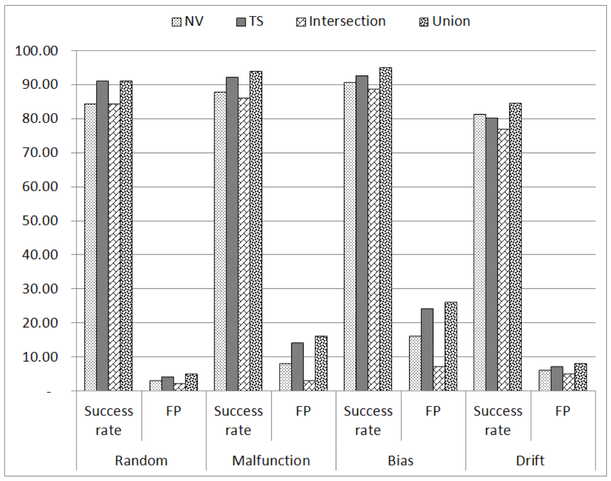


Figure 5: Experimental results

leave in the data. In addition, histogram analysis also checks the distribution of faults, distinguishing bias and drift faults. In our solution, the ARMA time series forecasting model is chosen as it shows potential to be a lightweight mechanism, satisfying the requirements of keeping the communication overhead, memory and computational cost low.

The initial evaluation results with SensorScope [1], an outdoor temperature dataset, confirm that our solution is able to detect and classify faulty readings into four types, namely, 1) random, 2) malfunction, 3) bias, 4) drift with accuracy ranging between 85% and 95%. These promising results suggest to investigate further the decentralised scheme for online sensor data fault handling.

This approach is in its initial phase, thus there are numerous suggestions that could address many open issues and help improve our approach: more evaluations of the accuracy of the method using other benchmark datasets, time series data estimations to propose a suitable model to detect faults in other types of sensor data (e.g., light). Most importantly, we intend to realise our system on actual sensors in a real environment in order to evaluate the computation and communication costs of our proposed solution.

7. ACKNOWLEDGEMENTS

The work is supported by the FP7-ICT-2013-EU-Japan, Collaborative project ClouT, EU FP7 Grant number 608641; NICT management number 167. Tuan Anh Nguyen thanks the Honiden lab at the National Institute of Informatics, Tokyo, Japan for supporting him during his internship during which he conducted the reported research.

8. REFERENCES

[1] Sensorscope project: `http://sensorscope.epfl.ch/`.

[2] S. T. Alexander. *Adaptive Signal Processing: Theory and Applications*. Springer-Verlag New York, Inc., 1986.

[3] V. Baljak, T. Kenji, and S. Honiden. Faults in Sensory Readings: Classification and Model Learning. *Sensors & Transducers*, 18:177–187, 2013.

[4] V. Baljak, K. Tei, and S. Honiden. Classification of faults in sensor readings with statistical pattern recognition. In *SENSORCOMM 2012, The Sixth International Conference on Sensor Technologies and Applications*, pages 270–276, 2012.

[5] V. Baljak, K. Tei, and S. Honiden. Fault classification and model learning from sensory Readings – Framework for fault tolerance in wireless sensor networks. In *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 408–413, 2013.

[6] G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley.com, 2013.

[7] Y. Bresler and A. Macovski. Exact maximum likelihood parameter estimation of superimposed exponential signals in noise. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34(5):1081–1089, 1986.

[8] A. Bulut, A. K. Singh, P. Shin, T. Fountain, H. Jasso, L. Yan, and A. Elgamal. Real-time nondestructive structural health monitoring using support vector machines and wavelets. In *Nondestructive Evaulation for Health Monitoring and Diagnostics*, pages 180–189. International Society for Optics and Photonics, 2005.

[9] J. Chen, S. Kher, and A. Somani. Distributed fault detection of wireless sensor networks. In *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 65–72. ACM, 2006.

[10] P. Jiang. A new method for node fault detection in wireless sensor networks. *Sensors*, 9(2):1282–1294, 2009.

[11] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 219–230. ACM, 2004.

[12] Y.-A. Le Borgne, S. Santini, and G. Bontempi. Adaptive Model Selection for Time Series Prediction in Wireless Sensor Networks. *Signal Processing*, 87(12):3010–3020, 2007.

[13] T. A. Nguyen, M. Aiello, and K. Tei. A Decentralized Scheme for Fault Detection and Classification in WSNs . In *Proceedings of The 1st IEEE International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA 2013, Work in Progress session)*, 2013. To appear.

[14] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava. Sensor Network Data Fault Types. *ACM Transactions on Sensor Networks (TOSN)*, 5(3):25, 2009.

[15] K. Pearson. Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material. *Philosophical Transactions of the Royal Society of London. A*, 186:343–414, 1895.

[16] A. B. Sharma, L. Golubchik, and R. Govindan. Sensor Faults: Detection Methods and Prevalence in Real-World Datasets. *ACM Transactions on Sensor Networks (TOSN)*, 6(3):23, 2010.

[17] E. U. Warriach, M. Aiello, and K. Tei. A Machine Learning Approach for Identifying and Classifying Faults in Wireless Sensor Network. In *15th IEEE International Conference onComputational Science and Engineering*, pages 618–625. IEEE, 2012.

[18] Y. Yao, A. Sharma, L. Golubchik, and R. Govindan. Online anomaly detection for sensor systems: A simple and efficient approach. *Performance Evaluation*, 67(11):1059–1075, 2010.

[19] S.-J. Yim and Y.-H. Choi. An adaptive fault-tolerant event detection scheme for wireless sensor networks. *Sensors*, 10(3):2332–2347, 2010.

[20] Y. Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE*, 12(2):159–170, 2010.