Online Data Fault Detection For Wireless Sensor Networks - Case Study

Dima Hamdan [a,b], Oum-El-Kheir Aktouf[b], Ioannis Parissis[b], Bachar El Hassan[a], Abbas Hijazi[a]

a. *LASTRE Laboratory*, Lebanese University, Tripoli, Lebanon

b. *LCIS Laboratory*, INP- Grenoble University, Valence, France

***Abstract*— Sensor faults are the rule and not the exception in every WSN deployment. Sensors themselves may get stuck at a particular value or get partially disconnected and report noisy measurements. Sensor nodes may reboot unexpectedly or stop transmitting data. Software running on the sensor nodes may have bugs and may cause data loss. In this paper, we present an efficient approach for online data fault detection and its application to a case study from a real world dataset. Our approach exposes four types of data faults as they occur by locally applying five simple heuristic rules. By locally applying these rules, node will not need to exchange messages with neighboring ones and consequently to consume energy. Simulation results showed that around 19% of the total collected readings from a real world dataset were faulty.**

Keywords: *wireless sensor networks (WSN), data faults, fault detection, reliability, temporal correlation.*

I. INTRODUCTION

Wireless sensor networks (WSNs) consist of spatially distributed autonomous sensors used to monitor physical or environmental conditions, such as temperature or sound, and to cooperatively pass their data through the network to a main location. Sensors used in wireless sensor networks are state-of-the-art technology with the lowest possible price. The sensor measurements we get from these devices are therefore often noisy, incomplete and inaccurate.

By definition, a fault is an unpermitted deviation of at least one feature of the system from an acceptable condition [1]. Thus, it is an abnormal condition that may cause reduction in the performance of the WSN by decreasing the judgment accuracy of the base station and wasting much of the limited energy of the sensors. Therefore, the quality of the data received must be ensured. Sensor data integrity encompasses fault detection, fault localization, identification of root causes, and correcting/recovering from data faults. Our work focuses on data fault detection in sensor network. As far as we know, there have been few studies dealing with the detection of faulty readings in WSNs. This may be due to the fact that WSNs are still a new technology, and there are few datasets collected from the real world. These few studies though differ in their assumptions of the observed phenomena; they impose some requirements to identify anomalous data. Some approaches [2- 4] require a dense network where sensor node often turns to neighbors to further determine whether the data is normal while the node itself cannot decide. For instance, spatiotemporal correlations and sample redundancies are used to validate the data faults identified at local tiers [2], or the correlations between different attributes as well as spatiotemporal correlations which are first captured by probabilistic data models and then used to tolerate data loss and noise [3,4]. Besides the physical redundancy requirement, the efficiency of these solutions requires that the neighbors report correct readings which may not be always guaranteed. Other approaches [5-7] require resource-plentiful nodes to perform more complex computational tasks. Yao [5] identified anomalies by a local step on the nodes using temporal correlations then by an aggregation step using spatial correlations on more powerful node. Yuan [6] proposed an anomaly detection process that requires a powerful node to construct models for the aggregated data. Bing [7] applied the spatial correlation using a resource-plentiful node. In addition, some solutions [8, 9] are costly in communication terms. For example, a non-parametric and unsupervised method based on data exchanges among neighbors has been developed for outlier detection [8]. FTEQ [9] performs self-evaluation and cooperative evaluation schemes based on short and long terms spatiotemporal correlation to detect faulty data.

Therefore, we try to make an efficient contribution in this field by proposing an energy efficient fault detection approach. The basic idea behind the approach is to detect faults locally based on temporal correlation and by applying simple five heuristic rules. Based on this correlation, the method can perform well on different network topologies (dense and sparse), node capabilities and without incurring any message exchanges with neighbors. Based on the five rules, this method can incorporate the outlier detection approaches summarized above for applications that possess cross-attribute correlations. As a case study, we present the results of the application of our approach on real data collected by sensors designed for environmental monitoring.

The rest of the paper is organized as follows. Section 2 is dedicated to the description of the fault database and the different types of faults reported in the literature as well as our methodology used to detect each of these types of faults. Then, in section 3 we present and discuss the results obtained by applying our approach on real world dataset. In section 4 we evaluate the performance of our approach. Finally, a conclusion is drawn in section 5.

978-1-4673-1291-2/12/$31.00 (c) 2012 IEEE

II. MATERIALS AND METHODS

A. Database Description

Our work is based on data collected by 54 battery-powered Mica2Dot wireless sensors designed for temperature and environmental monitoring. The sensors were deployed in the Intel Berkeley Research Lab [10]. Each sensor collected time stamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds from February 28th to April 5th 2004. During this period, a log of about 2.3 million readings was collected from these sensors. The reported readings consisted of 8 variables each: date, time, epoch, sensor id, temperature, humidity, light and voltage. Epoch is a monotonically increasing sequence number from each sensor. There are missing epochs in this data set. Sensor ids range from 1-54; data from some sensors may be missing or truncated. Figure 1 shows the data loss rate of temperature variable of each sensor. Temperature is expressed in degrees Celsius. Humidity is temperature corrected relative humidity ranging from 0-100%. Light is in Lux and voltage is expressed in volts.

The basis of our method is to detect sensor readings faults using a set of rules. In the literature, there exist four different rules for sensor fault detection. Some of these rules are based on the use of a predefined parameter. We first define each of the four types of the reported faults then we present our detection methods.

B. Data faults

1) Abrupt fault:

Abrupt fault is one of the most commonly seen faults in sensor data [11]. Data reported from many of the sensors showed a common pattern of a change between successive values [12]. However, due to the erratic hardware or to an environment event [13], a sudden change in value (Figure 2) may occur disturbing some expectation of continuity between successive values. We compute the rate of changes between consecutive readings to detect that a reading is anomalously different from the previous one. Let Δv, Δt and Δmax denote the variation of values, the variation of times and threshold for the rate of expected changes, respectively, the abrupt fault rule can be defined as:

$$\frac{\Delta v}{\Delta t} > \Delta \max \qquad (1)$$

2) Noise fault:

While noise is common and expected in sensor data, an unusually high amount of noise may be a sign of a sensor problem [11]. Unusually high noise may be due to a hardware failure or low batteries [11]. A noise fault (Figure 2) is characterized by a period during which the data values exhibit larger than normal variations [12]. The noise rule declares a fault whenever the standard deviation of set successive values exceeds a given threshold. Let σv , $\sigma \max$ denote the standard deviation of a set of N successive values, the maximum expected standard variation threshold and N is the window size, respectively. The noise fault rule can be defined as:

$$\sigma\vartheta > \sigma \max \qquad (2)$$

3) Stuck at fault*:*

Minimal instability is expected in a set of rapid, contiguous sensor values (Figure 2). Therefore, an unusually steady set of readings may be indicative of sensor failure. In this case, the reported value can be either very high or very low compared to the "normal" sensor readings [12]. The stuck at rule declares a fault whenever the standard deviation of a set of successive values stays below a certain threshold. Let σv, $\sigma \min$ denote the standard deviation of a set of N successive values and the minimum expected threshold, respectively; the stuck at fault rule can be defined as:

$$\sigma\vartheta < \sigma \min \qquad (3)$$

The value of ($\sigma \min$) may be obtained by analyzing the measurements in controlled laboratory conditions, setting it to be less than or equal to the minimum observed variability [14].

4) Out of range fault:

Each sensor type has a range in which the sensor values are valid. Application dependent thresholds for the maximum expected value and the minimum expected value are used. Herein, the out of range rule declares a fault whenever the mean of N rapid successive values exceeds the expected maximum and minimum thresholds. Letϑmax, ϑmin denote the maximum and minimum expected values, respectively; the out of range fault rules can be defined as:

$$\bar{\vartheta} > \vartheta max \qquad (4)$$

$$\bar{\vartheta} < \vartheta min \qquad (5)$$

C. Parameters determination

Traditionally, the appropriate values of parameters used in the various fault detection methods can be obtained from leveraging domain knowledge, a data sheet of the sensor, a model of other similar sensors, environmental understanding, or past behavior of the sensor in question [11]. In this paper, the various parameters used in our methods are obtained heuristically. We estimate the values of the parameters based on offline training phase (Figure 3). In order to find the optimal values of the parameters, we used data from the sensor 31 that reported the highest number of readings (Figure 1). Besides, this sensor looks like have a normal diurnal pattern in comparison to other sensors (Figure 4).

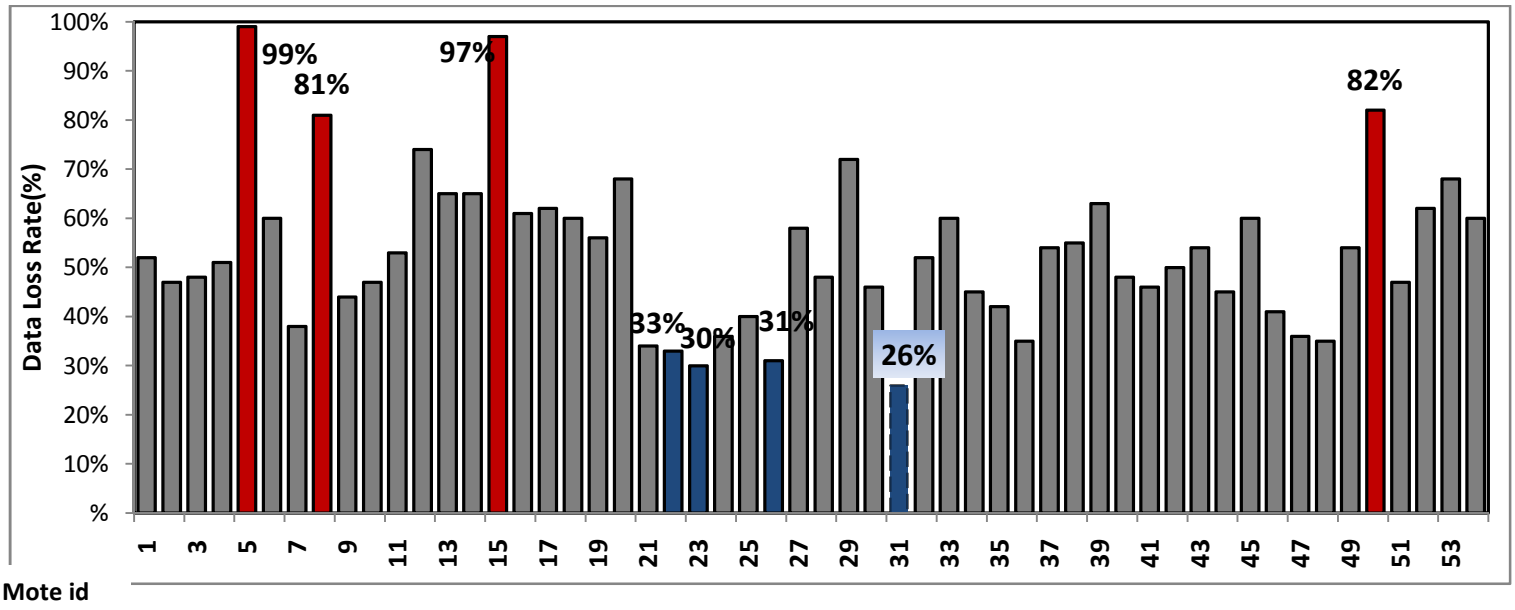
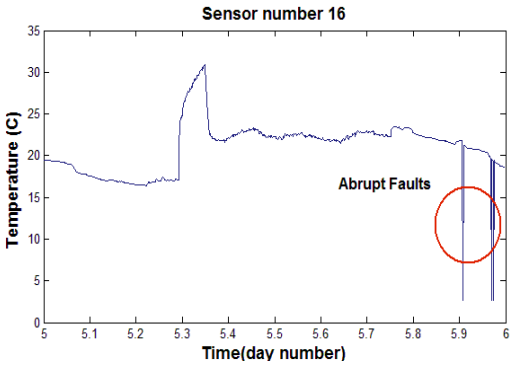
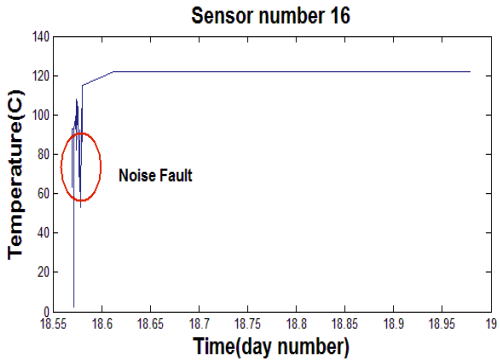
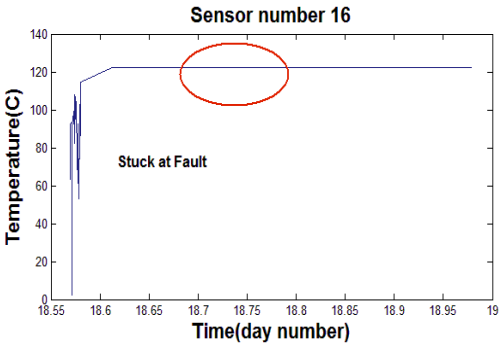


Figure 1-Data loss rate for each mote. The mote 31 has the least missing data





(b)



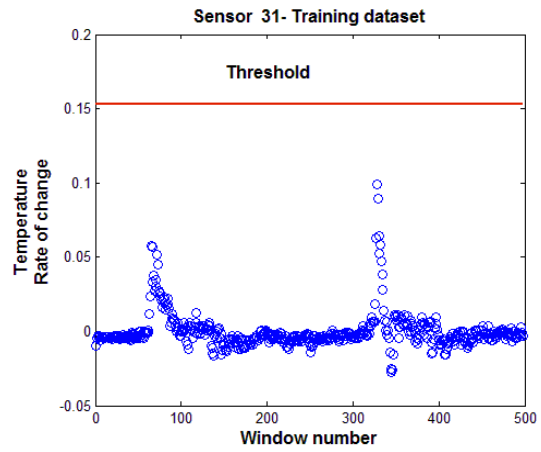
(c)

Figure 2: Data Fault types Examples. Abrupt fault is shown in (a), noise fault is shown in (b), stuck at fault is shown in (c)

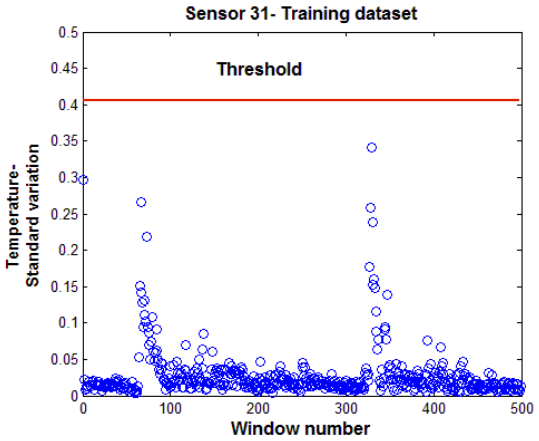
With the least missing data, we assume that sensor 31 is the most reliable among the sensors of the network and data reported from this sensor may be used in order to compute the values required in our detection techniques. We used the first two days data points for the training phase. We used data points at the beginning of the deployment as the battery state is likely to be good. The parameter values are shown in Table I**.** The min and max values are selected based on the database description given by the Intel lab [10] and the observation during the two days of training.

III. RESULTS AND DISCUSSION

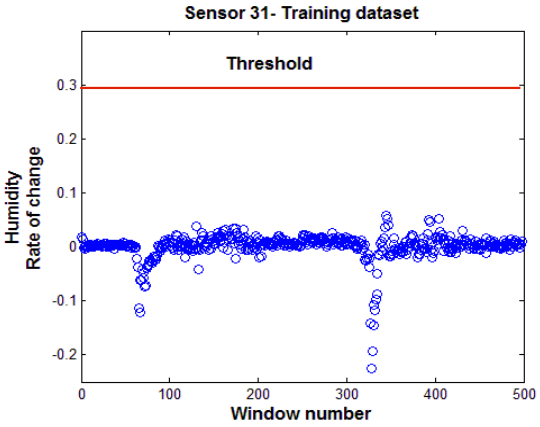
In order to detect the four types of data faults, five heuristic rules were applied to a real sensor data set. We can notice that data showed a combination of the four types of faults as shown in Table II. In average, 19% of the (collected) temperatures samples were detected as faulty, 17% of the humidity samples were detected as faulty, while 21 % of the collected light samples were detected as faulty. The total faulty samples rate was about 19% of the total collected samples. As shown in TABLE II, we notice that the ABRUPT fault type occurs in the lowest percentage (1%) compared to the other data fault types. The STUCK AT and OUT OF RANGE occur in the highest percentage (14%). We notice that the percentage of STUCK AT fault is close to that of the OUT OF RANGE fault. This is due to the fact that the sensors are stuck at OUT OF RANGE values (temperature: 120, humidity:-3.9, light: 0.92 and others). Inspection of the voltage values showed that the faulty samples were well correlated with the last few days of the deployment when supplying power to the motes were low (less than 1.4) as shown in Figure 5. To distinguish fairly the healthy and the faulty motes in presence of the high percentage of data loss (53%), we calculated a ratio that we called useful data rate (UDR).



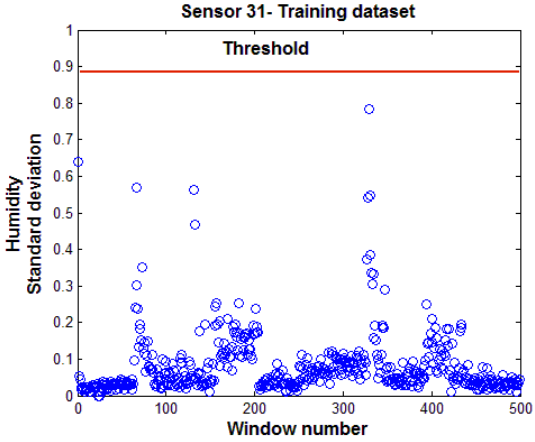
(a)



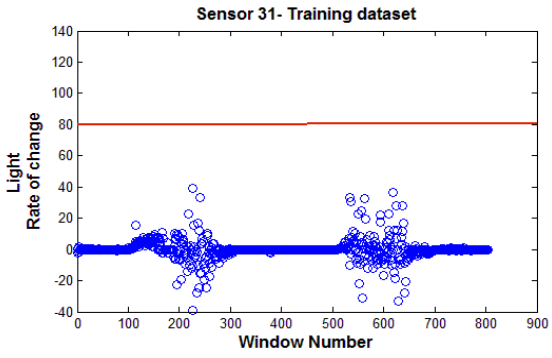
(b)



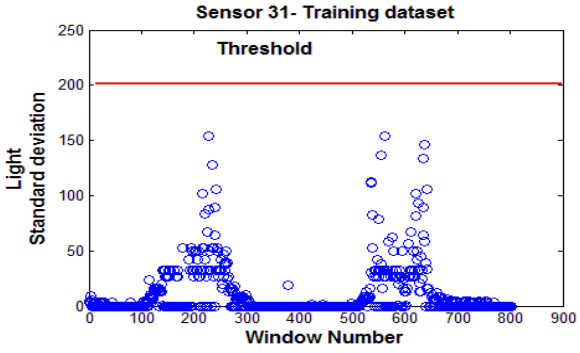
(c)



(d)



(e)



(f)

Figure 3 – Parameters determination for the different variables: (a, b) Temperature (c, d) Humidity and (e, f) Light sensors

This ratio took into consideration the data loss rate (DLR) and the data faulty rate (DFR) of each mote. Thus, the (UDR) of each mote can be defined as:

$$UDR_i^t = (100 - DFR_i^t) * (100 - DLR_i^t) \qquad (6)$$

Where UDR_i^t, $DFR_i^t and\ \ DLR_i^t$ denote the useful data rate, the data loss rate and the data faults rate, respectively, of the mote i for the sensor type t. The (UDR_i)of the mote i, can be defined as:

$$UDR_i = \frac{\left(UDR_i^{temperature} + UDR_i^{humidity} + UDR_i^{light}\right)}{3} \qquad (7)$$

In Figure 6, we notice that the sensors (31, 22, 21, 48 and 47) are the most healthy (UDR>55%) motes whose both the DLR and the DFR are low. In contrast to the (5, 15, 8, 12 and 29) motes whose the DLR and/or DFR are high (UDR < 25%).

IV. PERFORMANCE EVALUATION

The exposed approach has been evaluated using the TinyOS 2.x [15]. TinyOS is a free and open source component-based operating system and platform targeting wireless sensor networks (WSNs). What motivates to use TinyOS is its acceptance in WSN community and its portability to many hardware platforms. Then, the performance of our solution can be easily studied using the TOSSIM tool [16]. TOSSIM is a TinyOS simulation tool which simulates WSN physical and link layer features accurately. In order to be a generic solution, we chose to implement our approach as a configurable service under the application layer. Service parameters ($\Delta max, \sigma min, \sigma max, \vartheta min, \vartheta max$) can be set and adjusted to be available to a myriad of applications and scenarios. The required memory on every node on the network for TinyOS implementation on different platforms is: 13330 bytes in ROM and 1107 bytes in RAM (Micaz), 12912 bytes in ROM and 1181 bytes in RAM (TelosB), 12889 bytes in ROM and 1118 bytes in RAM (Tinynode). Another important performance parameter is the energy consumption. One of the main advantages of our approach is its efficiency in term of energy consumption where sensor does not need to exchange messages to detect the data faults. So it is a lightweight solution since network overhead consumes three times more energy than computations [17]. However, the dataset training itself may contain faults. In this case, it couldn't reflect correctly the ground truth data model parameters. Therefore, to improve the result accuracy in presence of a high percentage of data loss (53%), the variation in time is taken into account between successive samples.

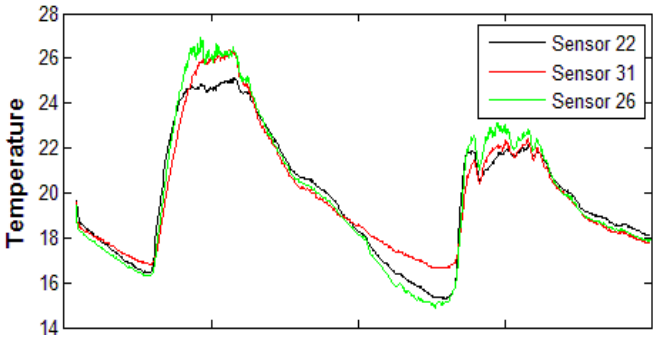


Figure 4 – Data models for the Sensor 22, 26, 31 in the first two days of deployment

SENSOR TYPE	RATE OF CHANGE $\frac{\Delta v}{\Delta t}$	MAXIMUM STANDARD DEVIATION σ max	MINIMUM STANDARD DEVIATION σ min	MAXIMUM EXPECTED VALUE ϑmax	MINIMUM EXPECTED VALUE ϑmin
T	0.15	0.4	0.003	50	-10
H	0.3	0.9	0	100	0
L	40	100	0	2000	1

Table I - Parameter values (T: Temperature, H: Humidity, L: Light)

SENSOR TYPE	ABRUPT	NOISE	STUCK AT	OUT OF RANGE	% DATA FAULTS (DISTINCT SAMPLES)
T	1%	2%	17%	17%	20%
H	0.5%	1%	15%	13%	17%
L	1%	2%	18%	10%	21%
Average	1%	1.5%	14%	14%	19%

Table II – Data Fault rates per sensor type and per data fault type (T: Temperature, H: Humidity, L: Light)

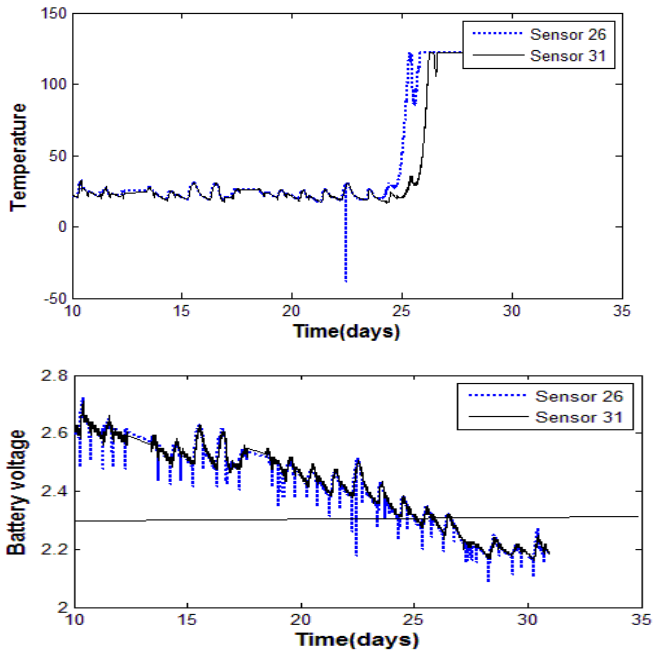


Figure 5- Temperature readings and battery voltages from sensors 26 and 31. The horizontal line provides an approximate voltage level at which both sensors begin to fail.

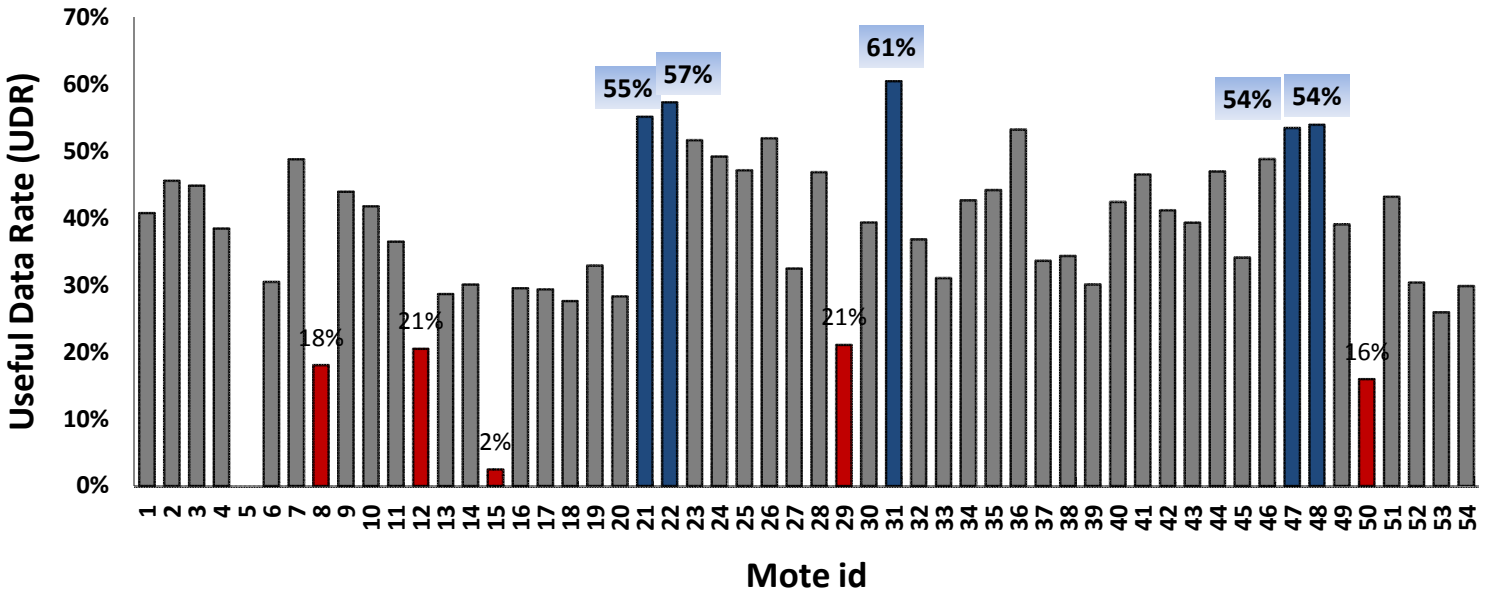


Figure 6- Useful data rate for each mote

V. CONCLUSION AND FUTURE WORK

In this paper, an efficient data fault detection method has been applied to a real world dataset. The method focuses on four types of data faults which are common in WSN deployments [12]. Our work is divided into three phases: Understanding of the real-world data, estimating parameters phase and testing phase. The first point that emerges is that, a data analysis phase was necessary to understand the temporal correlation between the readings of the same sensor, diurnal pattern of each sensor type, the data loss, etc. Second, the parameters for each sensor type are estimated based on a training data process for one confident sensor in the network. This sensor has the lowest data loss rate in comparison to other sensors. Then, the values of the method parameters were evaluated with a certain tolerance value. The last phase was the fault detection phase. Our detection method is rule based. Each mote can apply these rules online to detect abnormal data without incurring any communication cost. Although, non-complex techniques and a small training dataset were used, our method demonstrated good performance on detecting the different types of sensor faults. The results showed that about 19% of the total collected samples were faulty. Future work will explore the remedial action techniques to clean efficiently the faulty samples.

REFERENCES

[1] R. Isermann . Fault-Diagnosis Systems, "An introduction from Fault Detection to Fault Tolerance". Springer-Verlag, Berlin Heidelberg, 2006.

[2] S. Zahedi, M. Szczodrak, P. Ji, D. Mylaraswamy, M. Srivastava, R. Young, "Tiered architecture for on-line detection, isolation and repair of faults in wireless sensor networks". Proceedings of IEEE Military Communications Conference (MILCOM), 2008.

[3] A. Deshpande, C. Guestrin, S. Madden, "Using probabilistic models for data management in acquisitional environments". Proceedings of Second Biennial Conference on Innovative Data Systems Research (CIDR), 2005.

[4] B. Sheng, Q. Li, W. Mao, W. Jin, "Outlier detection in sensor networks". Proceedings of the 8th ACM international symposium on Mobile ad hoc etworking and computing (MobiHoc), 2007, p. 219-228.

[5] Yuan Yao, Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan. "Online Anomaly Detection for Sensor Systems: a Simple and Efficient Approach". In Proceedings of the IFIP WG 7.3 International Symposium on Computer Performance, Modeling, Measurements and Evaluation (Performance), Nov, 2010.

[6] J.Yuan, H. Zhou and H.Chen. "Subjective Logic-Based Anomaly Detection Framework in Wireless Sensor Networks". Hindawi Publishing Corporation International Journal of Distributed Sensor Networks.Volume 2012, Article ID 482191, 13 pages.

[7] Y.Bing Feng et al, "Fault Detection of WSN Based on Spatial Correlation". Applied Mechanics and Materials Journal, Vols. 58-60, pp 1504-1510, June 2011.

[8] J. Branch, B. Szymanski, C. Giannella, R. Wolff, H. Kargupta, "In-network outlier detection in wireless sensor networks. Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS), 2006, p. 51-58.

[9] Rongbo Zhu, "Efficient Fault-Tolerant Event Query Algorithm in Distributed Wireless Sensor Networks". International Journal of Distributed Sensor Networks Volume 2010..

[10] Url: http://db.csail.mit.edu/labdata/labdata.html.

[11] Kevin Ni , Nithya Ramanathan , Mohamed Nabil Hajj Chehade , Laura Balzano , Sheela Nair , Sadaf Zahedi , Eddie Kohler , Greg Pottie , Mark Hansen , Mani Srivastava, "Sensor network data fault types".ACM Transactions on Sensor Networks (TOSN), v.5 n.3, p.1-29, May 2009.

[12] Abhishek B. Sharma, Leana Gollubich, Ramesh Govindan," Sensor Faults: Detection Methods and Prevalence in Real-World Datasets". ACM Transactions on Sensor Networks (TOSN) Volume 6 Issue 3, June 2010.

[13] Ramanathan, N., Balzano, L., Burt, M., Estrin, D., Kohler, E., Harmon, T., Harvey, C., Jay, J.,Rothenberg S., Srivastana, M.. "Rapid Deployment with Confidence: Calibration and Fault Detection in Environmental Sensor Networks". Tech. Rep. 62, CENS, April 2006.

[14] I. Urteaga, K. Barnhart, and Q. Han." Redflag a run-time distributed, flexible, lightweight, and generic fault detection service for data-driven wireless sensor applications". In IEEE PerCom, 2009.

[15] TinyOS (2007), Url : http://www.tinyos.net/.

[16] P. Levis, N. Lee, M. Welsh, D. Culler, "Tossim: Accurate and scalable simulation of entire tinyos applications," in Proceedings of the First ACM Conference on Embedded Networked Sensor Systems, SenSys, 2003.

[17] H. Karl, A. Willis," Protocols and Architectures for Wireless Sensor Networks", John Wiley and Sons, 2005.