EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Condition Monitoring for Wireless Sensor Network-Based Automatic Weather Stations

Mary Nsabagwa [1,*], Julianne Sansa Otim [1], Roseline Nyongarwizi Akol [2], Grace Ninsiima[1], Robert Mwesigye[1], Maximus Byamukama[2] and Björn Pehrson[3]

[1] Department of Networks, Makerere University, Kampala, Uganda, mnsabagwa@cit.ac.ug, sansa@cit.ac.ug, graceolive8@gmail.com, mwesirob@gmail.com.
[2] Department of Electrical & Computer Engineering; rnakol@cedat.mak.ac.ug, maximus.byamukama@gmail.com
[3] KTH Royal Institute of Technology, Stockholm, Sweden; bjorn@pehrson.se

Abstract

Wireless Sensor Network (WSN)-based Automatic Weather Stations (AWSs) perform automatic collection and transmission of weather data. These AWSs face challenges, which lower their performance. Hence, a need for regular monitoring to reduce down time. We propose condition monitoring, comprised of a data receiver, analyser, problem classifier and reporter and visualizer, to mine data relationships, identify possible causes of problems and perform reporting of AWS status. The data receiver uses an M/M/1/k queuing model. We use Successive Pairwise REcord Differences (SPREDs) algorithm to compare arrival rates and packet content so as to establish sensor, node and AWS level performance. We also perform a hybrid of Grubb outlier detection and correlations amongst related variables for data validation. Problems take on one of four states. One connection can receive data at a rate as low as 1ms, without loss while problem identification especially in high density network is improved

Keywords: Automatic Weather Station (AWS), condition monitoring; queuing, Wireless Sensor Networks.

Received on 25 January 2018, accepted on 17 March 2018, published on 20 March 2018

doi: 10.4108/eai.20-12-2018.156083

1. Introduction

Automatic Weather Stations (AWSs) collect and transmit weather data without human intervention, enabling them to operate in remote areas. AWSs, which use Wireless Sensor Networks (WSNs) technology, in which distributed sensors collect varying parameters at predetermined intervals are the focus of this paper. While in remote deployments, WSNs face challenges such as coverage [1], packet loss and limited energy among others [2], which lower their health and life time. We use the term health to refer to the AWS's ability to perform its functions such as packet delivery rate and AWS availability among many other performance metrics. In order to ensure that the health of the AWSs is known at all times, there is need to perform condition monitoring to facilitate preventive maintenance and to lower downtime, hence lowering data losses. Condition monitoring has been proposed in applications such as railway [3], wind turbines [4], automotive industry [5] and in structural health monitoring [6]. No research has been performed on condition monitoring in AWSs. Moreover, AWSs have unique application requirements.

During condition monitoring, the monitoring entity may either perform active or passive monitoring. In active monitoring, the monitoring tool gets access to the network for data collection purposes as well as controlling the monitored device whereas in passive monitoring, only data collection is permitted. In order for the monitor to control the remote device, support protocols such as CoAP [7] may be used. Our focus is on passive monitoring, which receives weather data and analyzes it to only establish the

*Correspondence: mnsabagwa@cit.a.ug

health of the respective AWSs. Regardless of the type of monitoring used, the monitor should be furnished with data from which relationships are drawn. In our previous study of common AWS problems [8], AWSs face challenges such as energy exhaustion, packet dropping, inability of the gateway to transmit data and sensor node failure among others. While the received data is clearly structured, extracting knowledge on the AWS health requires performing data analytics.

The monitoring entity, while receiving data from the AWSs, may face challenges such as increased data volume and data transmission or arrival rates. The high data arrival rates may cause packet dropping, hence data loss at the receiving end. Therefore, the monitoring tool should be able to receive and process the growing number of data packets without loss as a result of buffer overflows. Queuing models provide solutions for the imbalance in arrival rates and processing speeds by allocating temporary storage to packets, using predefined procedures in order to avoid packet dropping. Based on the arrival method, service time distributions and number of servers, the queuing models are able to optimize metrics such as server utilization and reducing delays in the waiting time. Queuing has been used in monitoring Service Level Agreements (SLAs) processes [9], manufacturing [10], patient monitoring, video streaming [11], airport arrivals and departures [12] among others. Queuing can also be adopted by the data condition monitor, at the point of data reception, in order to avoid dropping of packets on arrival.

Once data has been received and stored, mining relationships in order to establish the health or performance of the AWS is done. Although the type of data may be known, in its raw form, conclusions about AWS performance is impossible at a glance. AWS performance is determined by metrics such as its availability, packet dropping, sensor degradation or any kind of deviation from what is considered normal behavior. These metrics may be provided by time series data, which provides information on normal behavior, hence forming a basis for identifying abnormal behavior or low performance. Using the time series data, anomaly detection through classification, clustering, association analysis, trend analysis and outlier analysis among others are possible [13]. Trend analysis of time-series data identifies significant increase or decrease in the magnitude of a variable and has been used in fields including energy [14], power [15], social media [16] and weather [17][18][19] using methods such as regression models, pattern mining, self-organizing map, fuzzy logic [20], graph-based methods [21], network anomaly detection [22] and others [23], Euclidean distance, k-nearest neighbors (KNNs), recurrences (REC) and support vector data description among others [24]. The data trends also provide insights into future performance of the monitored devices. Given the wide range of data types, characteristics and trends acquired by the AWS, it is impossible to apply one data mining technique, hence the need to use a hybrid of more than one mechanism. Furthermore, weather data trends vary with spatial distribution of the respective AWSs, hence a variation in readings at any given time.

Based on the above unique challenges, we propose condition monitoring located at a remote server and receiving data from an infinite number of AWSs. The proposed condition monitor consists of a data receiver, analyzer, problem classifier and reporter and visualizer shown in Fig. 1. Our contributions are as follows:

1. An M/M/1/k queuing algorithm, which is applied to each connection and generates parallel queues in order to handle high data arrival rates.
2. We propose a hybrid of three techniques to detect anomalies in data samples. The methods include:-
    - Successive Pairwise REcord Differences (SPREDs) clustering and classification technique
    - Observing data correlations
    - Using Grubb outlier detection
3. We complement available architectures with a reporting layer, which classifies problems in states for improved reporting.

The rest of this paper is organized as follows: Section 2 contains materials and methods we used for condition monitoring, section 3 gives details on the proof-of-concept experiment performed to test the proposed monitoring framework, section 4 presents results and we finally conclude in section 5.

2. Materials and methods

We designed a condition monitor for a network of AWSs based at the remote server to listen for incoming data from the AWS gateways, process it and store it in a database. Since processing is performed at the server with abundant resources, overhead resulting from the processing activities is considered negligible, except for buffer overflows. Once in the database, mining to deduce AWSs health and performance is done and results are classified into problems that are reported by the reporting layer. Figure 1 shows the architecture of the condition monitor, which comprises of four major components including:-

- Data receiver
- Data Analyser
- Problem Classifier

- Reporter & Visualizer

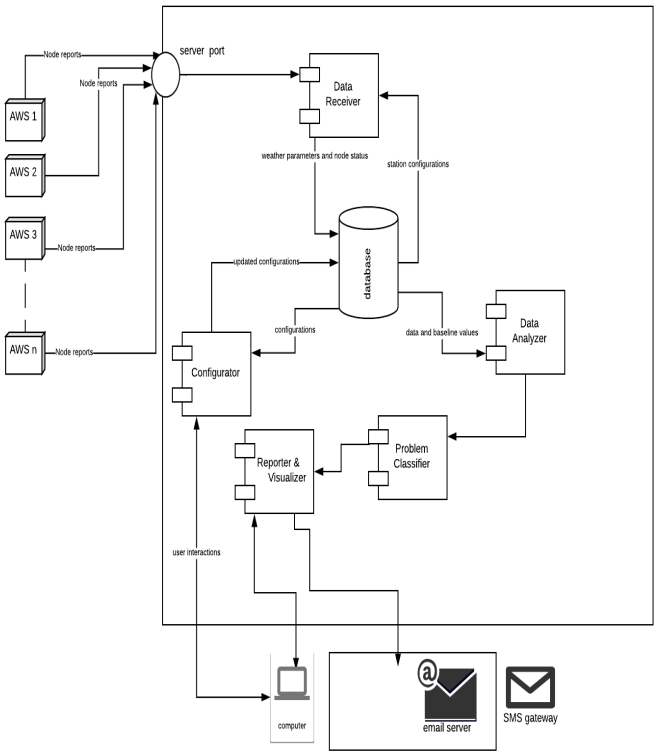

Figure 1. Architecture of AWS Condition Monitor running at the server

2.1. Data Receiver

The data receiver listens for incoming TCP connections, pre-processes received data and stores it in a database. The data receiver uses one TCP port to receive data / packets / reports from remote AWSs and creates a new connection for each AWS. The connections are maintained until data transmission from the respective AWS is complete. From this point on, we shall refer to what is received from AWSs as reports. The following are performed when a connection is established with the data receiver: - Create a data receiving thread if none exists for that connection / AWSs, receive and buffer the reports. The data receiving thread persists as long as there is incoming data via the same connection. The data receiving thread creates a child thread, also known as a data storing thread to extract reports from the buffer, process them and insert them into the database as shown in Figure 2.

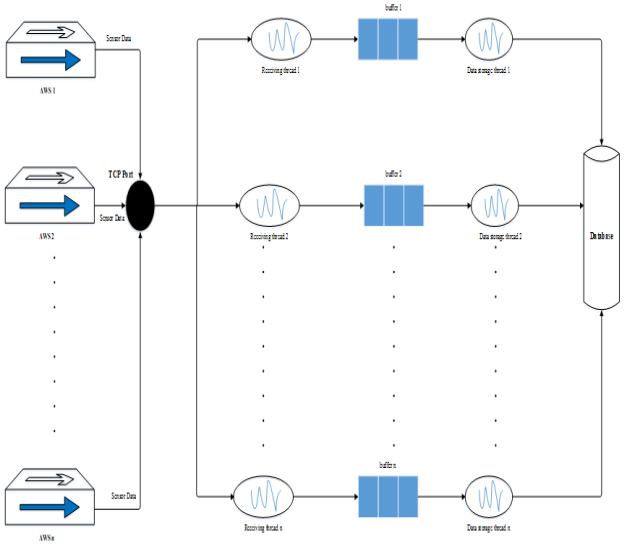

Figure 2. Architecture of the data receiver

A single connection may generate queues since the processing rate may be lower than the report arrival rate, hence the need for buffering. Buffering is a motivation for using a queuing model to handle the received reports and ensure that losses are avoided. The following are the characteristics;

i. Reports arrive at the server following a Poisson distribution. That is, server is unaware of how many possible reports it can pick ahead of time from the TCP port.

ii. Time taken to service a report is exponentially distributed. That is, reports arrive continuously and independently at a constant average rate.

iii. Inter-arrival time of reports and service time are independent of each other.

iv. Each AWS is associated with a single finite queue and there is no interaction between the AWSs.

The above data characteristics depict an M / M / 1 / k queuing model using a single server. Where:-

- M represents inter-arrival time of reports following an exponential distribution
- M represents processing time of reports following an exponential distribution
- 1- One server

- k - Finite buffer size

Since each connection gets a finite buffer, this queuing model therefore provides insights into metrics including average waiting, processing times of the reports and server utilization. Table 1 shows terms used in modeling the queuing system.

Table 1. Terms used and their description

Term / symbol	Description & significance
λ	Mean arrival rate of the reports
$\bar{\lambda}$	The actual mean arrival rate of reports $\bar{\lambda} = \frac{\bar{Q}}{\bar{W}}$
π	Mean inter-arrival time; the time taken between two reports receptions
μ	Mean service / processing rate
$\bar{N}$	Mean / expected number of reports in the system
k	Buffer size (number of reports in a filled queue)
$\bar{Q}$	Number of reports in the queue, expressed as $\bar{Q} = \bar{\lambda} * \bar{S}$
$\bar{S}$	Mean / expected processing / service time, expressed as $\frac{1}{\mu}$
$\bar{W}$	Expected steady state time a report spends waiting in the queue.
$\bar{T}$	Expected / mean steady state time a report spends in the queuing system expressed as $\bar{T} = \bar{W} + \bar{S}$
ρ	Traffic intensity (load), expressed as $\rho = \lambda * \bar{S}$
α	Probability that the server is busy at any given time
P_n	The steady probability that there are n reports in the system including the one being processed

Performance of the system derived by the equations below and borrowed from [25]. The probability P_n given that there are *n* reports in the system awaiting service is given as

$$P_n = \begin{cases} \frac{(1-\rho)*\rho^n}{1-\rho^{(k+1)}} & if\ \lambda \neq \mu \\ \frac{1}{k+1} & if\ \lambda = \mu \end{cases} \quad For\ n = 0, 1, \ldots k \tag{1}$$

Mean arrival rate of reports is given by;

$$\bar{\lambda} = (1 - P\_k)\lambda \tag{2}$$

Where

$$P_k = \frac{\rho^k}{\sum_{i=0}^{K} \rho^i} \text{ for } k = 0, 1\ldots, k. \tag{3}$$

P_k is the probability that there are k reports in the system.

The system is only stable if $\rho > 0$ and when k is fixed. The system is unstable if the mean processing rate of reports in the system is less than their mean arrival rate. In this case, the buffer will be filled to capacity, leaving no space for incoming reports, hence incoming reports shall be lost.

If λ_r, λ_q and λ_s are the arrival rates of the reports at the receiving thread, queue and data storing threads, then the actual mean arrival rates, $\overline{\lambda_r}$, $\overline{\lambda_q}$ and $\overline{\lambda_s}$ respectively are;

$$\overline{\lambda_r} = (1 - P_k)\lambda_r \tag{4}$$

$$\overline{\lambda_q} = (1 - P_k)\lambda_q \tag{5}$$

$$\overline{\lambda_s} = (1 - P_k)\lambda_s \tag{6}$$

The expected number of reports in the queuing system at a given time including the one being processed at that time, $\bar{N}$ is given by

$$\overline{N} = \begin{cases} \frac{\rho - (k+1)*\rho^{k+2} + k\rho^{k+3}}{(1-\rho)(1-\rho^{(k+1)})} & if\ \lambda \neq \mu \\ \frac{k}{2} & if\ \lambda = \mu \end{cases} \tag{7}$$

Therefore; $\overline{Q} = \overline{N} - (1 - P_0)$, where P_0 is the probability that there are no reports in the system.

Therefore; $\overline{Q}$ is also given by;

$$\overline{Q} = \frac{\rho^2}{1-\rho} \tag{8}$$

$$where\ \rho = \frac{mean\ arrival\ rate\ of\ reports\ to\ the\ queue}{mean\ processing\ rate\ of\ reports\ in\ the\ queue}$$

$$\rho = \frac{\overline{\lambda_q}}{\mu_s} \tag{9}$$

$$\overline{Q} = \frac{\left(\frac{\overline{\lambda_q}}{\mu_s}\right)^2}{1-\frac{\overline{\lambda_q}}{\mu_s}} \tag{10}$$

$$\overline{Q} = \frac{\overline{\lambda_q}^2}{\mu_s(\mu_s - \overline{\lambda_q})} \tag{11}$$

From equation (11), the system is stable if $\boldsymbol{\mu_s} > \overline{\boldsymbol{\lambda_q}}$. That is, the denominator tends to zero when $\boldsymbol{\mu_s} = \overline{\boldsymbol{\lambda_q}}$

Mean waiting time of a report in the queue

From,

$$\overline{W} = \frac{\overline{Q}}{\overline{\lambda}} \quad (12)$$

$$\overline{W_q} = \frac{\overline{Q}}{\overline{\lambda_q}} = \frac{(1 - P_K)\,\lambda_q}{\mu_q\,\overline{\lambda_q}} \quad (13)$$

From Equation 13, average report waiting time before processing is given by;

$$\overline{W} = \overline{W_q} \quad (14)$$

Mean Service time for the reports $\bar{S}$

$$\bar{S} = \frac{1}{\mu} \quad (15)$$

At the receiving thread, mean service time (time from when the report arrives at the thread to the time when it is added to the queue), $\overline{S}_r$ is given by;

$$\overline{S}_r = \frac{1}{\mu_r} \quad (16)$$

At the data storing thread; time taken from when the report is de-queued to the time when data is saved to the database, $\overline{S}_s$ is given by;

$$\overline{S_s} = \frac{1}{\mu_s} \quad (17)$$

From equation 16 and 17, the average service time for a given report is given by;

$$\overline{S} = \frac{\overline{S}_s + \overline{S}_r}{2} \quad (18)$$

The mean steady state time a report spends in the system, both waiting in the queue and processing time is given by;

$$\overline{T} = \overline{W} + \overline{S} \quad (19)$$

Server utilization

Server utilization, α, is given by;

$$\alpha = (1 - P_K)\rho \quad (20)$$

The rate at which the receiving thread, buffer, and data storing threads utilize the server are $\alpha_r, \alpha_q, and\ \alpha_s$ respectively. Therefore, average server utilization is given by;

$$\alpha = \frac{\alpha_r + \alpha_q + \alpha_s}{3} \quad (21)$$

$$\alpha = \frac{(1 - P_K)\rho_r + (1 - P_K)\rho_q + (1 - P_K)}{3} \quad (22)$$

$$\alpha = \frac{(1 - P_K)(\rho_r + \rho_q + \rho_s)}{3} \quad (23)$$

Where;

$$\rho_x = \lambda_x * \overline{S_x} \quad for\ x \in \{r, q, s\} \quad (24)$$

Mean inter-arrival time, π

The average time between reception of two successive reports is given by:-

$$\pi = \frac{1}{\overline{\lambda_r}} \quad (25)$$

We need to ensure that average service time (equation 18) is less than the average inter-arrival time (equation 25) in order to reduce packet dropping at arrival.

2.2 Data Analyser and Classifier

The data analyser mines available and real-time reports for patterns and anomalies and as per a given AWS. Our previous work [8] provided the nature of data being mined and identified AWS problems. These are summarized into the three below.

(i) Insufficient power supplies, which cause nodes to shutdown, hence the inability to perform data collection and transmission
(ii) Data loss due to packet dropping, faulty sensors or node misconfiguration
(iii) Errors in the data collected

The proposed condition monitoring algorithms are based on data with smaller dimensions and limited number of data types. However, AWS data varies by type, acceptable data ranges due to spatial and temporal variations in sites of deployment and by parameter of interest. It is from that background that the data analyser performs mining using a hybrid of Grubb outlier detection [26], assessing correlations in data trends and using Successive Pairwise REcord Differences (SPREDs) to detect AWS problems. Before applying the methods, we first assessed relationships amongst AWS data to establish correlations, without which the tested data is considered

anomalous. The next subsections assess relationships amongst the AWS parameters, in order to provide input into the mining algorithms.

Power supply Behavior

Sensor node supply voltage (V_IN) and microcontroller voltage (V_MCU) maintain a constant level (Figure 3), regardless of the solar insolation levels. In the absence of solar insolation, the voltage levels should be kept within the same limits. Failure to maintain the voltage levels especially in limited or no solar insolation times implies that there is degradation of the energy systems, if the load is constant, hence a need for a replacement.

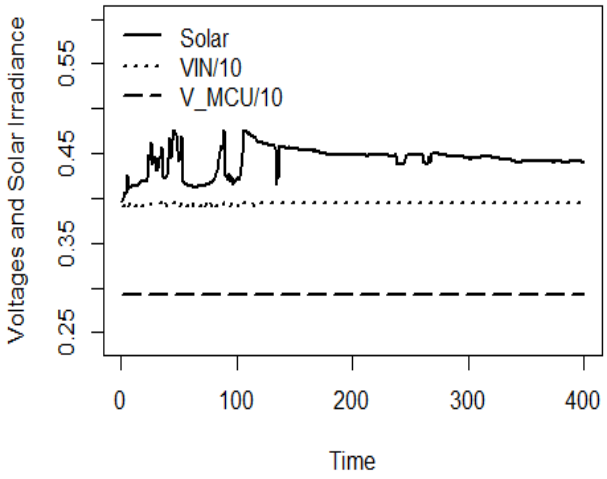

Figure 3. Input voltage, MCU voltage and Solar Insolation

Loss of Data

Data loss may be due to packet dropping, node misconfiguration, and sensor mechanical problems, AWS gateway or node shutdown due to power failures. Data loss can be discovered through analysing sequence numbers attached to the reports, observing inconsistencies in data transmission rates as well as comparing received data with historical data. RSSI and Link Quality Indicator (LQI) provide an indication of the quality of the link, which could be the cause of packet dropping.

Data Accuracy and Quality

For a given AWS, data accuracy may be assessed based on historical data, the expected patterns as per the configurations and data types for the received data. Additionally, data accuracy can be validated by comparing them with other weather parameters. Figure 4 shows a correlation matrix for the weather parameters. Soil and air temperature show a high positive correlation while temperature and relative humidity show a high negative correlation.

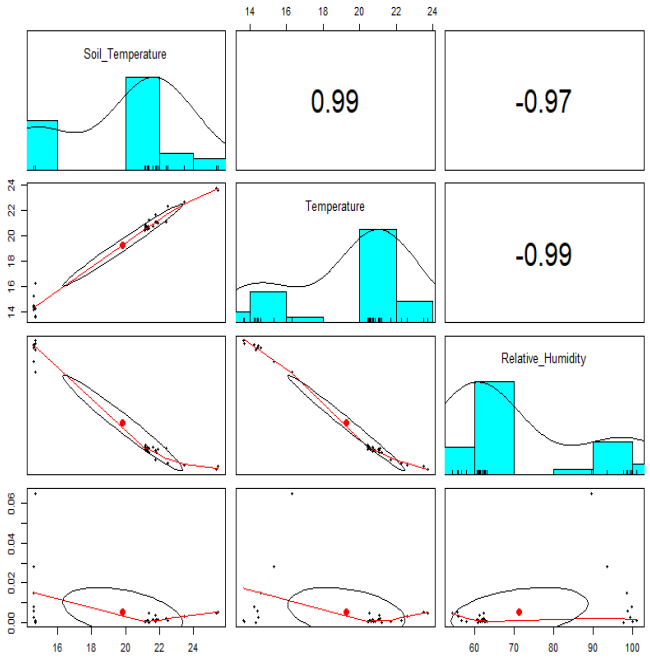

Figure 4. Scatterplot matrix showing correlation of a selected set of weather parameters

Increase in air temperature causes an increase in soil temperature and an increase in air temperature and soil temperature corresponds to a decrease in relative humidity. It is against these correlations that we conclude that sensor values have a problem if any deviation occurs. Based on the inverse proportion of temperature and relative humidity, we use the following to validate or invalidate the values

$T \propto 1/RH$

$$\Rightarrow \quad T = \frac{k}{RH} \tag{26}$$

$$\Rightarrow \quad k = T * RH \tag{27}$$

Using a sample of validated data, where T= 17.88 and RH = 93

$$\Rightarrow \quad k = 93 * 17.88 \ = 1662.84$$

Given either relative humidity or temperature, we apply the formula

$$\Rightarrow \quad T = \frac{1661.84}{RH}$$

Allowed values can have an error margin of ± 1, otherwise either relative humidity or temperature is reported suspicious. We also use Grubb outlier detection technique [26] to identify anomalies in data samples in order to flag them as suspicious. Grubb outlier detection checks if a significant difference exists between the expected and actual value using the equation

$$G = |xi - \bar{x}\ |S \tag{28}$$

At 95% confidence level, where

$\bar{x}$ is the sample mean of the data set
S is the standard deviation of the data set
x_i is the value in question

Lastly, in order to check the consistence in the rate of sensor, node or AWS level reporting, we propose an algorithm called *Successive Pairwise REcord Differences (SPREDs)*, which computes time differences between successive data packets and generates clusters representing the differences. In order to avoid lengthy computations, which should be regularly performed for each AWS, only a summary of the state at levels including AWS, node and sensor are maintained. At the node level, a list of model parameters that have been captured in the past is maintained. The list forms a basis of establishing anomalies for the future reports. A report showing reception time, number and list of unique time differences (clusters) between report intervals and a report interval change list/tracker is generated. The report interval change tracker gives the magnitude of the change, time the change occurred and whether it is an increase or decrease. The cluster list keeps cluster values and a count in each cluster. Using the two data structures (i.e. change tracker and cluster list), the analyser is able to identify information such as the cause of the change, what level of the AWS the problem is (sensor, AWS or node) including packet dropping, sensor faults and gateway failure to transmit. Classification of the problems is done using a decision table given in [8].

Algorithm 1: Successive Pairwise REcord Differences (SPREDs)

```
1: aws_sensor_list, missed_node_count =0
2:  if exists( new_aws_record)
3:  reference_parameters =
get_reference_parameter_list
4:        while(exists more tokens in model_tokens list)
5:            if(reference_parameter not found in new list)
6:                record_sensor_level_miss(time,
parameter)
7:       node_record_time_diff = last_record_time –
previous_record_time
8:      if(node_record_time_diff not equal to previous
diff)
9:            record_change(old_ddiff, new_diff, time)
10:
update_cluster_list(cluster_name,cluster_count)
11:    if(missed_node_count equals
len(aws_sensor_list))
12:        record_aws_miss(aws_name)
```

2.3 Visualizer and reporter

Reporting using either SMS, emails or web visualization is necessary because it informs stakeholders of the occurrence of a problem. It is important to determine when and how often to report persistent problems. Furthermore, a reported problem should only be re-reported if it has persisted for a maximum period of time. All problems both fixed and ignored are archived so as to generate a dataset that can guide in preventive maintenance. The reporter also gives information such as when node data was last received, details for last received packet, number of reports sent for a particular problem and time that has elapsed since problem was last reported. Figure 5 shows the state transitions of problems.

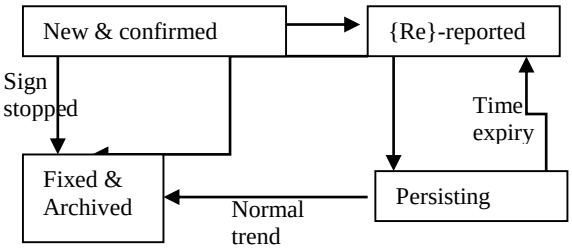

Figure 5. State transition diagram of the problems

The states in Figure 5 are explained as follows;
New & confirmed problem: There is supporting evidence of a new problem.
Reported/re-reporting problem: problem has been reported once and if it still persists, there is a chance of reporting it again
Persisting problem: The number of times a problem has been reported has exceeded the threshold and reporting of the same has stopped
Fixed & archived problem: Problem has been resolved and information stored for future reference.

The amount of data lost due to problems is calculated as below, depending on the level of loss, after at time T, at data rates of time t, where t<<T.

$$Sensor_{loss} = \frac{received}{expected} * 100 \quad (29)$$

$$Node_{loss} = \text{sensor1}_{\text{level}_{\text{loss}}} + sensor2_{loss} + \ldots sensorn\ loss \quad (30)$$

$$AWS_{loss} = node1_{loss} + node2_{loss} + \cdots + noden_{loss} \quad (31)$$

In cases where any of the levels of loss are 100%, the loss should be reported as an availability problem via either SMS or email. Node-level loss that is less than 50% is due to packet dropping and if in the next time T, a similar inconsistency occurs, packet dropping report should be sent via email and SMS. Re-reporting a problem should be done via SMS or email after a day from when the first case was reported. At the third and last time of reporting the same problem, the problem becomes persistent and no more messages are sent on the same. During the problem state transitions, the web visualizations should report details of each of the problems at all levels, indicating when the problem was first reported, current state if not yet fixed and how many times it has been reported. Also indicated, is the frequency of a given problem based on archived problems. The number of SMSs and emails on a specific

problem is controlled to enable drawing attention of the concerned persons to the messages.

3. Proof-of-concept Experiment

As a proof-of-concept, we set up an AWS consisting of 3 RSS2 wireless sensor nodes[27] and a gateway comprising of a sink node and raspberry pi. The gateway, consisting of a sink node and rasp berry pi, was placed in an office, approximately 80 feet from the AWS stand on which three sender nodes were installed (Figure 6). Sensors were placed 2m, 10m and close to the ground (gnd) and their data collected and processed using the RSS2 nodes to which they were connected. The sensor nodes were attached to the 10m metallic pole. Table 2 shows details of sensors and nodes used. In addition to transmitting weather parameters, each node transmits its input voltage, micro controller voltage and MAC address for proper identification using IEEE 802.15.4 protocol[28]. Nodes are configured to send data packets to the gateway after 1 minute and 15 seconds.

On receiving data packets, the sink node appends radio link information including RSSI and LQI to each packet, before sending them to a remote server or repository. At the repository, all packets are received via a single TCP port, processed and stored in a database. Real-time processing is performed on both stored and incoming data to establish performance of the remote AWSs.

Table 2. Weather parameters and nodes used

Node name	Weather parameter	Measurement mechanism
2m	Temperature and Relative Humidity	Capacitive humidity readings and band gap temperature readings
gnd	Rain, Soil moisture, Soil temperature	Interrupts and voltage,
10m	Wind Speed, Wind Direction, Solar insolation	Interrupts
sink	Pressure	Piezo-resistivity

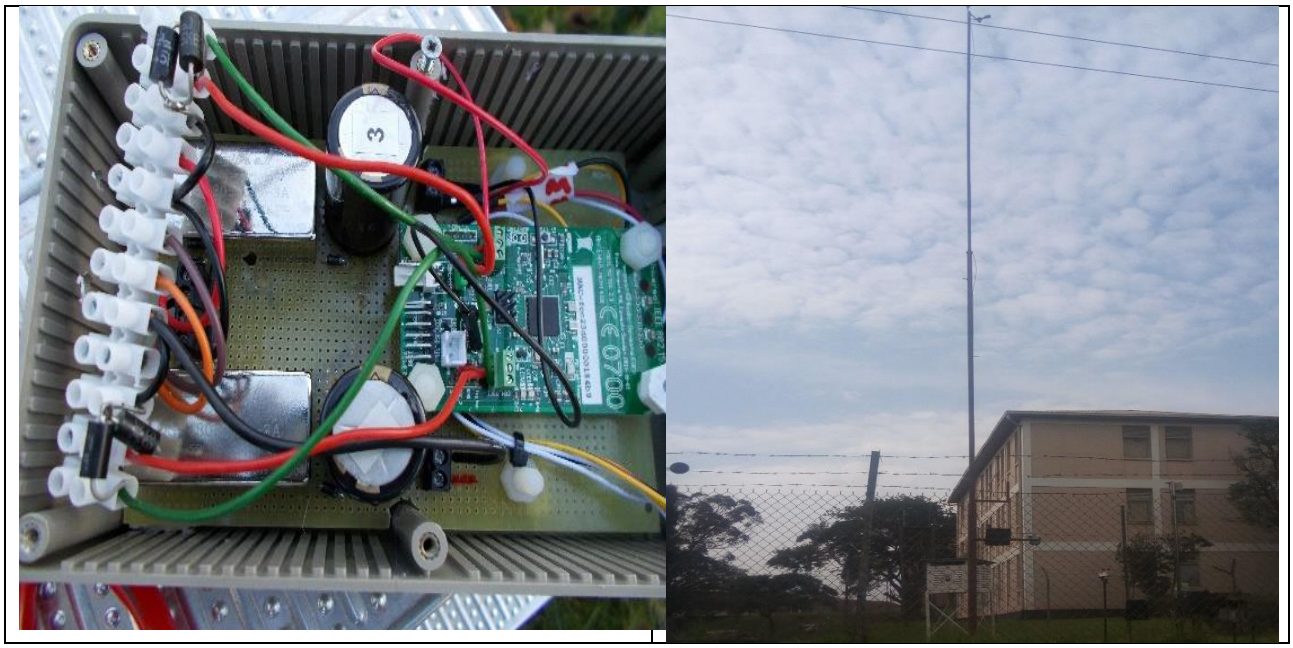

Figure 6. Left: Contents of a sensor node; Right: Deployment site for the AWS, showing the 10m stand on which three sensor nodes (2m, 10m and gnd) are placed. The gateway is placed in the building, close to the window on second floor, approximately 80 feet from the AWS pole.

4. Results

In this section, we present results of the various components of the AWS condition monitor.

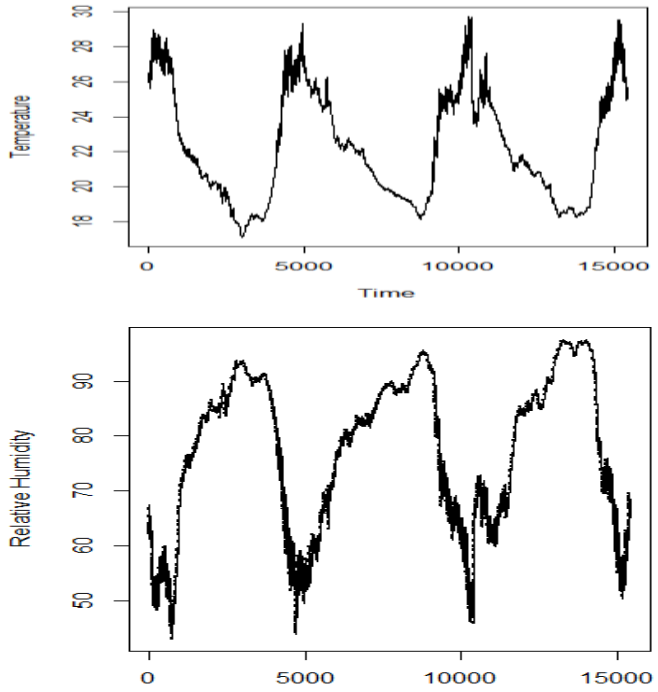

Figure 7. Web visualization of relative humidity and temperature over time to assist in real-time observations of the correlations of the two parameters.

The observations from the two graphs can lead to a conclusion as to whether any one of the two parameter values is erroneous.

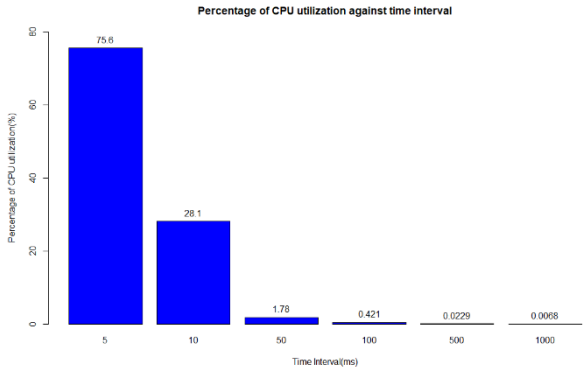

Figure 8. Percentage CPU Utilization for different time intervals

The CPU utilization is 75.6%, 28.1% and 1.78 for arrival rates of 5ms, 10ms and 50ms respectively. As the report intervals/arrival rates (ms) increases, CPU utilization increases because the time that CPU is made busy increases due to increased load. The same CPU utilization is experienced across the different parallel connections, hence a higher CPU utilization for all AWSs combined.

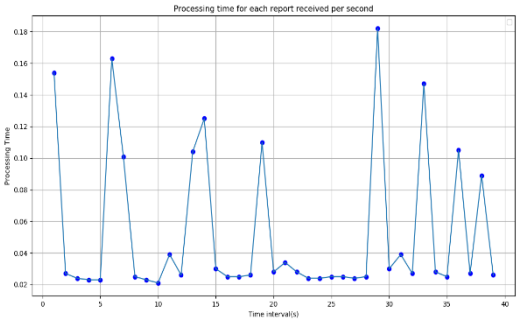

Figure 9. Variation of processing time in clock counts for the 1 second interval data.

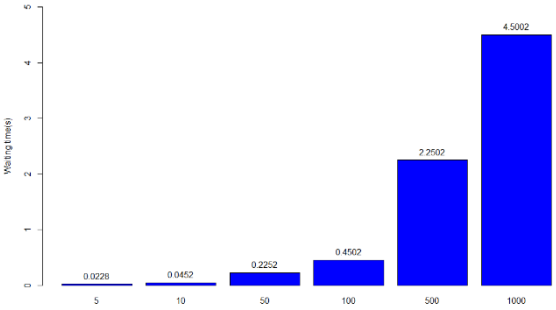

Figure 10. Average waiting time for different transmission intervals

The processing time is independent of the report arrival rate and varies from 0.02 to 0.18 clocks. This variation is as a result of the average size and content of the received reports. Reports that take the least amount of processing time are discarded on reception because they fail to match the required set guidelines while some reports consume processing time above 0.4 clocks depending on the size of the string. This implies that the storage process consumes approximately 3 to 4 times the amount of time consumed by the pre-processing time. Our algorithm proposes initial pre-processing in order to cut down on amount of time for storing invalid data. Other levels of processing can be done after storing the data to avoid dropping of reports from the queues.

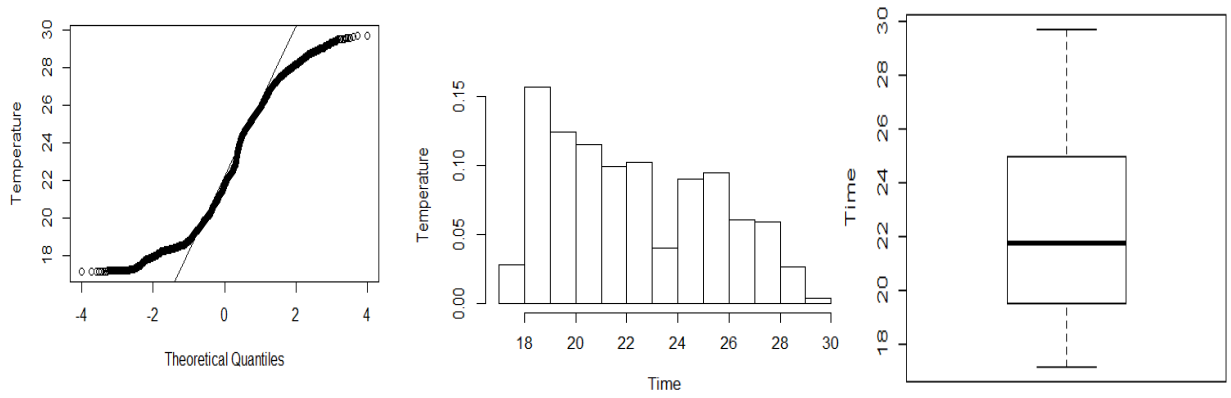

Figure 11. **Left**: Normal plot, **middle**: histogram and **right**: whisker plot for temperature values over a period from 17th to 20th August 2018, with a reporting interval of 15 seconds.

The distribution of temperature data indicates a normal distribution with the majority of readings appearing between 18 and 24. The assessment discovered no outliers in data values and gradual increases and decreases are observed. Unless rainfall readings are observed, which can cause a gradual decrease, temperature changes should be gradual.

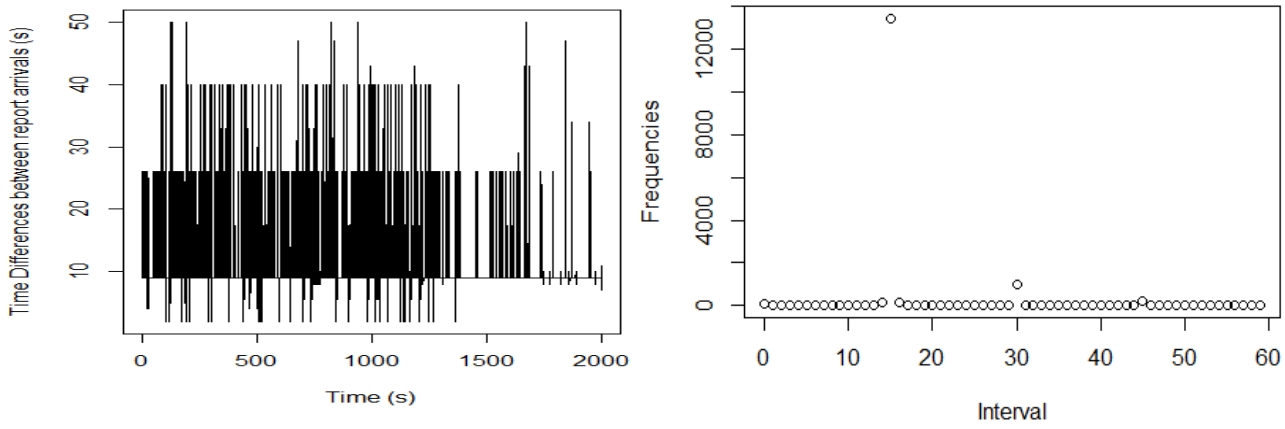

Figure 12. **Left**: Differences in reporting intervals (clusters) for the 10m node. **Right**: The number of clusters formed from reporting interval differences

While there is variation in the reporting interval, 15 seconds interval has the highest frequency (13471 times), followed by 30 seconds, which is 997 times given a total of 15527 records. This implies that the expected and right interval is 15 seconds. The rest are reported as packet drops and if data is not received after a prolonged time, for this case 1 hour, the node failure is reported.

5. Conclusions

Monitoring performance of WSN-based AWS is an important task that should be performed in order to facilitate preventive maintenance and reduce AWS downtime. The monitoring process becomes more complex as the network of AWSs being monitored grows variations in AWS data. Furthermore, receiving big volumes of data centrally becomes more challenging as the data arrival and processing rates vary, causing data packets to drop. We have proposed an architecture for monitoring a network of AWSs distributed over a wide area, consisting of a data receiver which, receives and stores data at rates as low as 1ms using M/M/1/k queuing model. The data receiver is able to perform an infinite number of parallel connections at the same time, hence facilitating reception of packets from many AWSs at the same time. In order to detect

anomalies, we have proposed a hybrid of outlier detection in numerical data and assessment of correlations in data trends and using Successive Pairwise REcord Differences (SPREDs) to provide sensor, node and AWS-level anomalies. The identified problems are reported using SMS, email and web visualizations, indicating the problem type, state of the problem and time the problem has lasted. Reporting of observed problems is done only three times in order to reduce chances of ignoring the messages. In future, we shall evaluate the performance of the listener in terms of receiving multiple reports at the same time, a new gateway design recently designed to regulate power consumption amongst the sensor nodes.

Acknowledgments: We acknowledge the financial support of NORAD (Agreement number UGA-13/0018). We are also grateful to the meteorological services of Uganda, Tanzania, Norway and South Sudan for the technical support given. We are also grateful to Dr. Julius Omona and Ms. Florence Bageya for hosting the gateway and Lawrence Ssanyu for proofreading this document. Special thanks go to the former interns of WIMEA-ICT lab, Makerere University including Robert Kasumba, Joshua Muhumuza, Eugene Tumwesigye Owak and Stephen Byarugaba for the technical input.

References

[1] C. Zhu, C. Zheng, L. Shu, and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks," Journal of Network and Computer Applications. p. 8, 2012.

[2] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: A survey on recent developments and potential synergies," J. Supercomput., p. 5, 2014.

[3] V. J. Hodge, S. O. Keefe, M. Weeks, and A. Moulds, "Wireless Sensor Networks for Condition Monitoring in the Railway Industry: A Survey," IEEE Trans. Intell. Transp. Syst., 2015.

[4] W. Qiao and D. Lu, "A Survey on Wind Turbine Condition Monitoring and Fault Diagnosis - Part I: Components and Subsystems," IEEE Trans. Ind. Electron., 2015.

[5] Y. Lei, J. Lin, M. J. Zuo, and Z. He, "Condition monitoring and fault diagnosis of planetary gearboxes: A review," Measurement: Journal of the International Measurement Confederation. 2014.

[6] and J. W. Bhuiyan, Md Zakirul Alam, Guojun Wang, Jiannong Cao, "Deploying wireless sensor networks with fault-tolerance for structural health monitoring," IEEE Trans. Comput., vol. 64, no. 2, 2015.

[7] Z. Shelby, K. Hartke, and C. Bormann, "Constrained Application Protocol(CoAP)," CoRE Work. Gr., 2013.

[8] J. S. O. Mary Nsabagwa, Isaac Mugume, Robert Kasumba, Joshua Muhumuza, Steven Byarugaba, Eugene Tumwesigye, "Condition Monitoring and Reporting Framework for Wireless Sensor Network-based Automatic Weather Stations," in 2018 IST-Africa Week Conference (IST-Africa), 2018.

[9] G. Cicotti, L. Coppolino, S. D. Antonio, and L. Romano, "Runtime Model Checking for SLA Compliance Monitoring and QoS Prediction," J. Wirel. Mob. Networks, Ubiquitous Comput. Dependable Appl., 2015.

[10] and T. O. Ryota Suzuki, Shigeru Kohmoto, "Non-intrusive Condition Monitoring for Manufacturing Systems," in 2017 25th European Signal Processing Conference (EUSIPCO), 2017.

[11] H. Tang, L. Chen, Y. Wang, N. Wang, and X. Li, "Stalling assessment for wireless online video streams via ISP traffic monitoring," in IEEE Wireless Communications and Networking Conference, WCNC, 2017.

[12] A. Jacquillat, A. R. Odoni, and M. D. Webster, "Dynamic Control of Runway Configurations and of Arrival and Departure Service Rates at JFK Airport Under Stochastic Queue Conditions," Transp. Sci., 2017.

[13] F. Chen, P. Deng, J. Wan, D. Zhang, A. V. Vasilakos, and X. Rong, "Data mining for the internet of things: Literature review and challenges," International Journal of Distributed Sensor Networks. 2015.

[14] M. Gorawski, A. Gorawska, and K. Pasterak, "The TUBE algorithm: Discovering trends in time series for the early detection of fuel leaks from underground storage tanks," Expert Syst. Appl., 2017.

[15] L. C. Shen, Lijuan and Cassottana, Beatrice and Tang, "Statistical trend tests for resilience of power systems," Reliab. Eng. \& Syst. Saf., vol. 177, 2018.

[16] L. A. Trucolo, Caio Cesar and Digiampietri, "Improving trend analysis using social network features," J. Brazilian Comput. Soc., vol. 23, 2017.

[17] N. N. Karmeshu Supervisor Frederick Scatena, "Trend Detection in Annual Temperature & Precipitation using the Mann Kendall Test – A Case Study to Assess Climate Change on Select States in the Northeastern United States," Mausam, 2015.

[18] P. Jain, X. Wang, and M. D. Flannigan, "Trend analysis of fire season length and extreme fire weather in North America between 1979 and 2015," Int. J. Wildl. Fire, 2017.

[19] R. K. Aggarwal, P. K. Mahajan, Y. S. Negi, and S. K. Bhardwaj, "Trend Analysis of Weather Parameters and People Perception in Kullu District of Western Himalayan Region," Environ. Ecol. Res., 2015.

[20] S. B. and S. A. Amro Al-Radaideh, A. R. Al-Ali, "A wireless sensor network monitoring system for highway bridges," in In Electrical and Information Technologies (ICEIT), 2015 International Conference on (pp. 121-122). IEEE, 2015.

[21] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," Data Min. Knowl. Discov., 2015.

[22] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," Journal of Network and Computer Applications. 2016.

[23] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," in Procedia Computer Science, 2015.

[24] M. Flach et al., "Multivariate Anomaly Detection for Earth Observations: A Comparison of Algorithms and Feature Extraction Techniques," Earth Syst. Dyn. Discuss., 2016.

[25] F. E. Grubbs, "Procedures for Detecting Outlying Observations in Samples," Technometrics, 1969.

[26] "Radio sensors." [Online]. Available: radio-sensors.com.

[27] IEEE Standard, "IEEE standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-Part 15.4 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.15.4-2006, pp. 1–26, 2006.