Agnostic Diagnosis: Discovering Silent Failures in Wireless Sensor Networks

Xin Miao[1], Kebin Liu[1,2], Yuan He[1,2], Yunhao Liu[1], Dimitris Papadias[1]
[1]Department of CSE, Hong Kong University of Science and Technology, Hong Kong, China
[2]TNLIST, School of Software, Tsinghua University, Beijing, China
Email: {miao, kebin, heyuan, liu, dimitris}@cse.ust.hk

***Abstract*— In wireless sensor networks (WSNs), diagnosis is a crucial and challenging task due to the distributed nature and stringent resources. Most previous approaches are supervised, relying on a-priori knowledge of network faults. On the other hand, our experience with GreenOrbs, a long-term large-scale WSN system, reveals the need of diagnosis in an *agnostic* manner. Specifically, in addition to predefined faults (i.e., with known types and symptoms), *silent failures* that are unknown beforehand, account for a large fraction of network performance degradation. Currently, there is no effective solution for silent failures because they are often diverse and highly system-related. In this paper, we propose Agnostic Diagnosis (AD), an online lightweight failure detection approach. AD is motivated by the fact that the system metrics (e.g., radio-on time, number of packets transmitted) of GreenOrbs sensors usually exhibit certain correlation patterns. Violations of such patterns indicate potential silent failures. We accordingly design a *correlation graph*, which systematically characterizes internal correlations inside a node. Silent failures are discovered by tracking the changes and anomalies of *correlation graphs*. We implement AD on a working WSN consisting of 330 nodes. Our experimental results demonstrate the advantages of AD to discover silent failures, effectively expanding the capacity and scope of WSN diagnosis.**

I. INTRODUCTION

Wireless sensor networks (WSNs) have been widely used in many fields, such as environmental surveillance, emergency navigation, traffic monitoring, and industrial control [15][16][22]. WSNs are by nature error-prone and have unsatisfactory reliability, encountering various faults and failures during their operation. Consequently, diagnosis has drawn substantial attention in recent years as a method to enhance the applicability, reliability, and efficiency of WSNs.

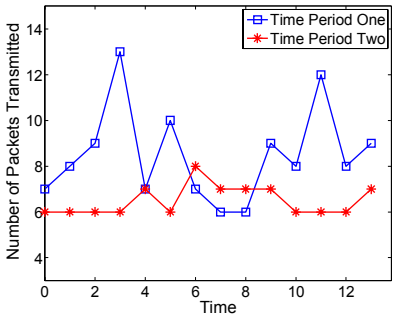
However, diagnosing WSNs is a challenging issue because, once a WSN is deployed, its inner conditions are not directly observable. Specifically, since many WSNs reside in harsh or remote environments, it is difficult to perform *in-situ* troubleshooting on the faulty nodes. Furthermore, the distributed nature and stringent resources of WSNs render it hard for a network operator to completely monitor the system's working status. Due to similar reasons, it is also infeasible to deploy management tools like SNMP, or other costly diagnostic modules, on the sensor nodes.

Many existing diagnostic approaches are supervised, i.e., they rely on either specific rules or inference models. An obvious drawback is that they are limited to faults with known types and symptoms, and hence, they cannot be easily generalized to different application scenarios. On the other hand, the interactions within the WSN and the causal dependencies between root causes and symptoms are usually unknown. As a result, silent failures remain undetected.

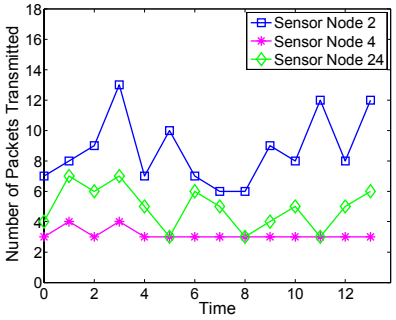
This paper is motivated by the need for long-term reliable operation of GreenOrbs, a large-scale WSN system in a forest [16]. Currently, GreenOrbs includes 330 nodes and has been in continuous operation for over eight months. During the deployment, we often observe system performance degradations, e.g., low packet delivery ratio. A portion of faulty nodes can be easily identified since they generate apparently abnormal system metrics (e.g., measurements that are clearly beyond the reasonable scope). The other faulty nodes, however, cannot be identified in this way. For instance, we adopt low-power listening mode so the radio is switched on only for receiving, sending, or idle listening. Consequently, the radio-on time should be closely correlated with the amount of traffic passing the node. We noted that during a five-minute period, a node kept its radio on for 47.5 seconds, transmitting only 3 packets in total. In the next five-minute period, it kept its radio on for 51.6 seconds and transmitted 550 packets. Any individual value of the metrics is not abnormal, but the correlation between radio-on time and number of transmitted packets clearly suggests inconsistency on that node.

A straightforward solution would be to develop a set of static correlation rules for identifying the faulty nodes. However, this is inapplicable for two reasons: (i) usually there is insufficient domain knowledge to enumerate all the rules; (ii) WSN deployment is often evolutional, so that the correlation rules change over time; both software upgrades and environment conditions may have a great impact on the correlations.

To overcome these problems, we propose Agnostic Diagnosis (AD), an online lightweight approach for WSNs. AD exploits the correlations among metrics of each sensor using a *correlation graph* that describes the latent status of the node. Such a *correlation graph* is updated periodically using the node's metrics. By mining through the *correlation graphs*, we identify the underlying rules of a normally running system, and detect abnormal correlations.



(a) Number of packets transmitted by node 2 in two time periods



(b) Number of packets transmitted by nodes 2, 4, and 24 in a same period

Figure 1. Examples of normal operation and node failure.

Our main contributions are summarized as follows:

- Unlike previous approaches, Agnostic Diagnosis relies on minimal a-priori knowledge and can be generalized to a wide variety of WSN applications.
- We propose the *correlation graph*, a compact structure that efficiently characterizes the internal correlations inside a node.
- We implement AD and evaluate it with traces from a 330-node GreenOrbs deployment. Case studies and statistics demonstrate the effectiveness of AD.

The rest of the paper is organized as follows. Section 2 motivates the problem. Section 3 presents AD. Section 4 evaluates our design and Section 5 surveys related work. Finally, Section 6 concludes this paper.

II. MOTIVATION

Diagnosis is a fundamental task for long-term large-scale WSN systems. This section first introduces the basic information and application requirements of GreenOrbs, including several observations in the form of concrete examples. These observations reveal the existence of correlation patterns among the nodes' operational metrics and the feasibility of agonistic diagnosis.

A. GreenOrbs

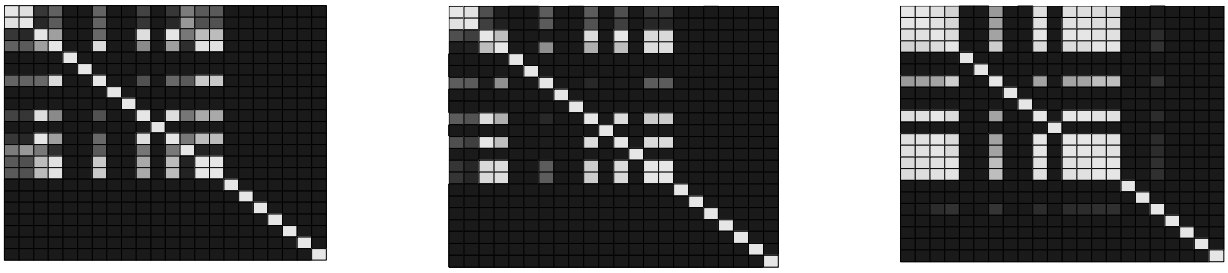
GreenOrbs is an ecological surveillance project deployed in a forest. It collects a group of sensory data such as temperature, humidity, illumination and carbon dioxide concentration to support various applications. Since the deployment is in a remote area, the overhead of *in-situ* debugging and troubleshooting is very high. Therefore, the diagnostic system is designed so that each node periodically transmits its current status to the sink. Specifically, we collect 22 types of metrics from each node that are classified into four categories: (1) timing metrics e.g. RadioOnTimeCounter, which denotes the accumulative radio-on time; (2) traffic metrics, e.g. TransmitCounter, which records the accumulative number of packets transmitted by a node; (3) task metrics, e.g. TaskExecCounter, which is the accumulative number of tasks executed; (4) other metrics such as ParentChangeCounter, which counts the number of parent changes.

The difficulty of diagnosis in GreenOrbs stems from the absence of a-priori knowledge on the possible faults and their symptoms, which renders general diagnostic rules inapplicable. Instead, we design a diagnostic system requiring minimal domain knowledge.

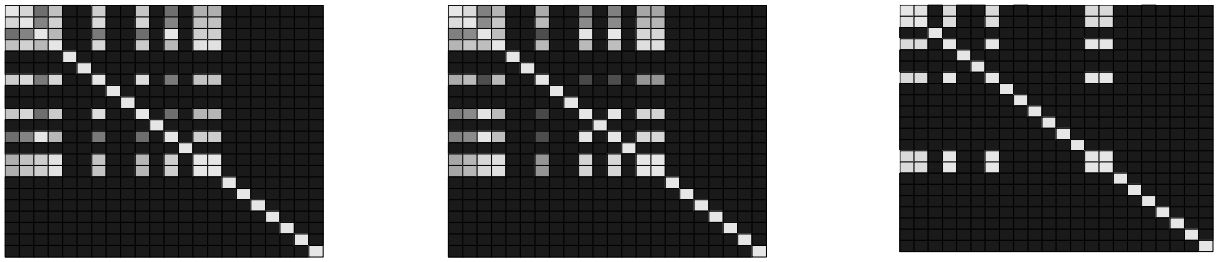
B. Observations

In the following, we demonstrate a set of interesting failure examples. Figure 1(a) plots the number of packets transmitted by a node every 15 minutes. During time period one, the value of this metric changes dramatically, possibly indicating that the node is faulty. However, a manual analysis of the whole trace shows that the node works well. On the other hand, even though the metric appears to be stable during period two, the node actually encounters routing loops (a fatal routing problem). This example demonstrates that the individual examination of metrics on the same sensor node may overlook silent failures, or may flag failures by mistake. Figure 1(b) shows another example, where the data curves correspond to the packet transmissions of different nodes during the same period. Even though the metrics exhibit different values and variations, all three nodes in fact perform well, but the correlation among them is relatively weak. This example suggests that even if considering multiple sensor nodes, an individual metric is insufficient to uncover failures.

We then conduct the correlation matrix of the 22 metrics and visualize them using gray-scale images (the details of the correlation matrix will be presented later). A square (i,j) in the image denotes the correlation score between metrics i and j. Figure 2(a)-(c) displays three images constructed from metrics of the same node. The node is normal in the first two periods, but faulty in the third one. Interestingly, the first two gray-scale images are very similar to each other, whereas they differ significantly from the third one. Analogous results are observed in the spatial domain as well. Figure 2 (d)-(f) shows the images of three nodes in the same period. The first two nodes are normal, while the third one is faulty. Observe that the first two images follow a similar pattern, which is clearly distinguishable from the last one.



(a) Healthy period 1 of sensor node 2 (b) Healthy period 2 of sensor node 2 (c) Faulty period of sensor node 2



(d) Healthy node 1 (e) Healthy node 2 (f) Faulty node

Figure 2. Visualizations of correlation matrices.

The difference of correlation patterns can be explained with the following example. There are two metrics called LoopCounter and SuccAckCounter. The former is the number of times the same packet passes through the current node, and the latter is the number of packets that are successfully transmitted by the current node. If the node is normal, the value of LoopCounter should be zero most of the time. Thus, these two metrics have weak correlation. If the node is out of order and produces "routing loops", the same packet passes multiple times. Eventually, LoopCounter dominates SuccAckCounter, and these two metrics are highly correlated, differentiating the faulty node from of the normal ones.

In summary, proper diagnosis of WSNs, especially long-term large-scale systems like GreenOrbs, is crucial and challenging. This is partially due to resource constraints and hardness to track in-network status. Even a more important, but often overlooked fact is that the diagnostic capacity is restricted by our incomplete knowledge of possible failures. Our experience with GreenOrbs reveals that independent investigation of metrics is insufficient for capturing all the problematic nodes. On the other hand, exploiting correlations of metrics uncovers significantly more failures.

III. AGNOSTIC DIAGNOSIS

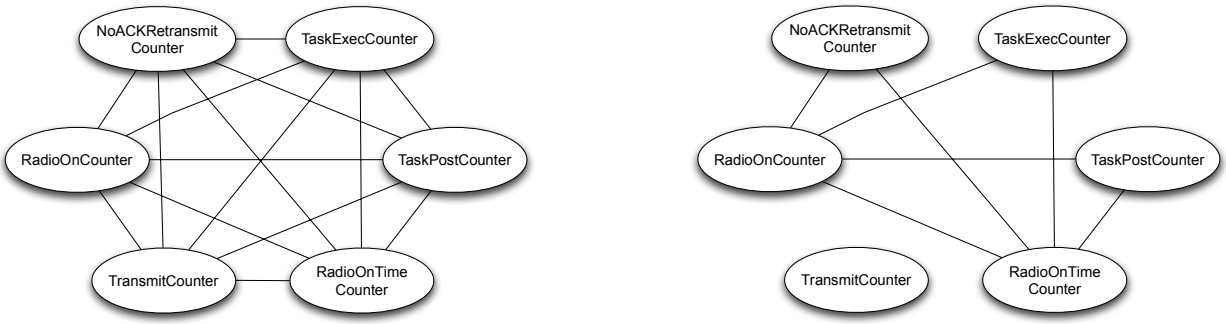
We assume a WSN consisting of N sensor nodes. At each time stamp t, a sensor s_i measures its working status, obtaining a status vector $S_{i,t}=(m_{1,t}, m_{2,t}, \ldots, m_{p,t})$, where p is the number of metrics and $m_{u,t}$ is the value of the u-th metric at time t, $1\leq u\leq p$. Table 1 summarizes the symbols used in this paper and their meanings.

A. Correlation Graphs

The *correlation graph* is a graph representing the pair-wise correlations of the metrics. It is constructed and maintained periodically, for each node in the WSN. Figure 3(a) illustrates the graph of a healthy node. Each edge has a weight, which denotes the correlation score between the corresponding metrics (weights of edges and some other vertices are omitted in our illustrations, for the sake of simplicity). In Figure 3(b), the missing correlation between TransmitCounter and RadioOnTimeCounter suggests that this node might be faulty.

Table 1. Definition of symbols

Symbols	Definition
N	Number of sensors
p	Number of metrics
w	Window size
s_i	Sensor node i, $1\leq i\leq N$
$S_{i,t}$	The p-dimensional status vector of sensor i at time t, $S_{i,t}=(m_{1,t}, m_{2,t}, \ldots, m_{p,t})$
$m_{u,t}$	The value of the u-th metric at time t, where $1\leq u\leq p$
$c_k(u,v)$	Correlation score of metrics u and v in time window k, where $1\leq u,v\leq p$
$CG_{i,k}$	*Correlation graph* of sensor i in window k



(a) Graph of a healthy node

(b) Graph of a faulty node

Figure 3. Correlation graphs of two nodes.

The correlation between metrics are evaluated in each time window k, which lasts from time $(k\text{-}1)*w+1$ to time $k*w$, i.e., $[(k\text{-}1)*w+1,\ k*w]$. Suppose $M_{u,k}=(m_{u,(k\text{-}1)*w+1},\ m_{u,(k\text{-}1)*w+2},\ \ldots,\ m_{u,k*w})$ and $M_{v,k}=(m_{v,(k\text{-}1)*w+1},\ m_{v,(k\text{-}1)*w+2},\ldots,\ m_{v,t*w})$ are the values of metrics u and v collected in time window k respectively. We define the correlation score between them using Pearson's product-moment coefficient:

$$c_k(u,v)=\frac{w\sum_{i=1}^{w} m_{u,(k-1)*w+i}m_{v,(k-1)*w+i}-\sum_{i=1}^{w} m_{u,(k-1)*w+i}\sum_{i=1}^{w} m_{v,(k-1)*w+i}}{\sigma_{u,k}\sigma_{v,k}}$$

where $\sigma_{u,k}$ and $\sigma_{v,k}$ are their standard covariance

$$\sigma_{u,k}=\sqrt{w\sum_{i=1}^{w} m_{u,(k-1)*w+i}^2-(\sum_{i=1}^{w} m_{u,(k-1)*w+i})^2}$$

$$\sigma_{v,k}=\sqrt{w\sum_{i=1}^{w} m_{v,(k-1)*w+i}^2-(\sum_{i=1}^{w} m_{v,(k-1)*w+i})^2}.$$

This correlation score falls within the range [-1,1]. The closer it is to either -1 or 1, the stronger the correlation between the variables is. As it approaches zero, the correlation decreases. Thus, we can construct a matrix where each element $c_k(u,v)$ is the correlation score between metric u and metric v:

$$Correlation\ Matrix=\begin{bmatrix} c_k(1,1) & c_k(1,2) & \ldots & c_k(1,p) \\ c_k(2,1) & c_k(2,2) & \ldots & c_k(2,p) \\ \ldots & \ldots & \ldots & \ldots \\ c_k(p,1) & c_k(p,1) & \ldots & c_k(p,p) \end{bmatrix}.$$

This matrix is symmetric, i.e., $c_k(u,v)=c_k(v,u)$. In addition, since each metric is perfectly correlated with itself, $c_k(u,u)=1$. The matrix is a representation of the *correlation graph*. In Figure 2, the visualization of correlation matrix is achieved by transforming correlation scores into gray-scale values in the range of [0,255]. Because the gray value of white is 255, lighter pixels correspond to more correlated metrics.

B. AD Framework

AD identifies anomalies in the temporal and spatial dimension. Specifically, *temporal detection* refers to sudden changes in the *correlation graph* of a node. *Spatial detection* discovers pattern inconsistencies using multiple nodes. If an anomaly is identified by both temporal and spatial detection, then it has a high chance of signifying a real problem.

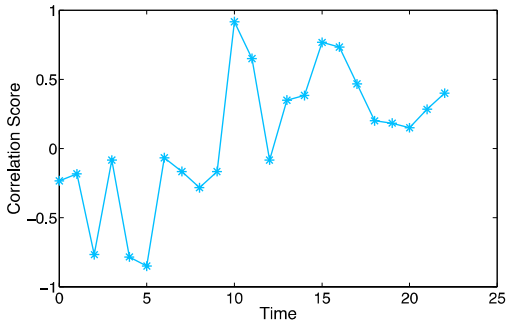
Temporal detection investigates the evolution of *correlation graphs* over time. In every window (e.g., $[1,\ldots,w]$, $[w+1,\ldots,2w]$), a *correlation graph* of sensor s_i is computed. If the system operates normally (i.e., no sensor nodes go out of order, and no events such as link failures, traffic congestions occur), the *correlation graphs* of the node should remain relatively stable. On the other hand, abrupt changes in consecutive graphs indicate silent failures.

Spatial detection investigates pattern differences in the graphs of nodes with similar characteristics. Although sensors are deployed in different areas and connected in an ad-hoc manner, they still share some intrinsic similarities. For instance, leaf nodes in the global routing tree only need to transmit data packets to their parents. The rest are intermediate nodes, which, in addition to their own data, forward packets originating in their sub-tree. Sensor nodes may share some similar patterns according to their roles in the network.

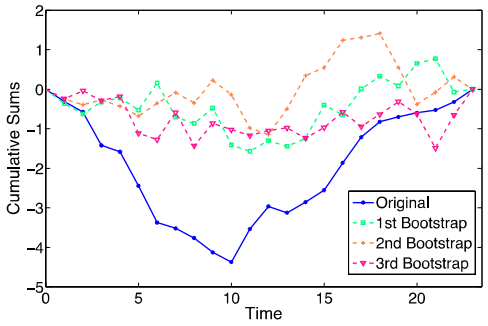
C. Temporal Detection

Given two consecutive *correlation graphs* $CG_{i,k}$ and $CG_{i,k+1}$ of the same node (for successive time windows), an abrupt change may occur at one or more edges. For example, suppose that two metrics RadioTimeOn and TransmitCounter are highly correlated in $CG_{i,k}$, but not in $CG_{i,k+1}$. Then, even if all the other edges are the same, the change in correlation is regarded as abrupt. Therefore, we focus on the timely identification of such changes in individual edges.

Problem 1. (Abrupt Change Time of *Correlation Graphs*) The abrupt change time of $CG(i)=\{CG_{i,1},\ CG_{i,2},\ \ldots,\ CG_{i,t},\ \ldots\}$ is the time when an abrupt change happens in time series $\{c_1(u,v),\ c_2(u,v),\ \ldots,\ c_t(u,v),\ \ldots\}$, where $c_t(u,v)$ is the correlation score between metrics u and v in time window t, $1\le u,v\le p$ and $u\ne v$.



(a) Original time series



(b) Original cumulative sums and cumulative sums after bootstrap

Figure 4. CUSUM example.

Given that GreenOrbs maintains p=22 metrics per sensor, for each *correlation graph* sequence $CG(i)$, there are totally $p\cdot(p\text{-}1)/2$=231 different time series. At the end of each time window, AD computes whether there is an abrupt change for any edge. If yes, it marks it as a change point of sensor s_i. The detection of abrupt change for each edge is modeled as a change point analysis problem. We adopt a classical CUSUM algorithm [17] to discover change points for time series. In our application, this includes three steps.

Step 1: Cumulative Sums Calculation. In this step, CUSUM charts are constructed based on the original time series. Let $\{c_1(u,v), c_2(u,v), \ldots, c_n(u,v)\}$ be the correlation scores of edge (u,v) from the first time window to the current one. The cumulative sums $CS_0, CS_1, CS_2, \ldots, CS_n$ are calculated as:

- $CS_0 = 0$;
- $CS_i = CS_{i-1} + c_i(u,v) - \sum_{i=1}^{n} c_i(u,v)/n$.

The intuition behind the cumulative sums is that if there are no abrupt changes in correlation scores, then the cumulative sums just shrink near zero. Supposing at the beginning all the scores are above the average, the term $c_i(u,v) - \sum_{i=1}^{n} c_i(u,v)/n$ is always larger than zero, causing cumulative sums CS_i to increase steadily. If $c_{k+1}(u,v)$ is an abrupt change, CS_{k+1} should be much smaller than CS_k. Thus, CS_k will dominate both the preceding and the subsequent cumulative sums. The change score is denoted as $CS_{diff}=max(CS_i)\text{-}min(CS_i)$.

Step 2: Bootstrap Analysis. Bootstrap analysis provides a way to test the significance of the change by mimicking the behavior of CUSUM if there are no change points. In this step, the time series $\{c_1(u,v),c_2(u,v),\ldots,c_n(u,v)\}$ are reordered randomly. Based on the random ordered time series, a new sequence of cumulative sums $CS'_0, CS'_1, CS'_2, \ldots, CS'_n$ are computed. The new change score is $CS'_{diff} = max(CS'_i)\text{-}min(CS'_i)$. After performing bootsraps M times, among which there are X times $CS_{diff}>CS'_{diff}$, the confidence level of CS_{diff} is calculated as X/M.

Step 3: Change Point Detection. Once the confidence level of CS_{diff} is higher than a predefined threshold (e.g., 90%), we say that an abrupt change happens in the current time series. We select the index k so that $CS_k=max|CS_i|$. Therefore, k is the last index before the abrupt change, and $c_{k+1}(u,v)$ is the change point.

In Figure 4, the original time series has an abrupt change in index 10. At the same position, the CUSUM curve encounters a sudden change, which is consistent with the original time series. However, in the CUSUM chats with random ordering, there is no sudden change, and all the curves tend to shrink near zero.

At the end of this step, the nodes whose *correlation graphs* change acutely are detected. However, they are not necessarily faulty, and are cross-validated by spatial detection.

D. Spatial Detection

Spatial detection takes snapshots of *correlation graphs* of all nodes in each time window, and groups similar ones together. Sensors whose *correlation graphs* diverge from the common patterns are considered suspicious.

Problem 2. (Spatial Detection) Given a set of nodes $s_1, s_2, \ldots, s_n$, as well as their *correlation graphs* in window t, i.e., $CG_{1,t}, CG_{2,t}, \ldots, CG_{n,t}$, divide them into K clusters with cluster centroids $C_1, C_2, \ldots, C_K$. The confidence level for a sensor node s_i to be suspicious is defined as:

$$\min_j(dist(CG_{i,t}, C_j)),$$

where $dist(CG_x, CG_y)$ is the binary distance function between two *correlation graphs*.

Considering the symmetry of the correlation matrix, we can take the upper triangular of it and transform it to a d-dimensional vector, where $d=p\cdot(p\text{-}1)/2$. Therefore, an intuitive solution is to apply K-Means clustering [19] directly and find the farthest nodes from the centers. However, this approach is complicated and inefficient due to the high dimensionality of *correlation graphs*. For high dimensionality, the concept of distance becomes meaningless. Specifically, the difference of the maximum and minimum distance converges to zero [24]:

$$\lim_{d\to\infty} \frac{\max(dist(CG_x, CG_y)) - \min(dist(CG_x, CG_y))}{\min(dist(CG_x, CG_y))} \to 0,$$

which causes the clustering results to be imprecise.

Another problem is the dependencies among dimensions. For instance, suppose metrics u, v, and w are highly correlated pair-wisely. As a result, the three dimensions $c(u,v)$, $c(u,w)$, $c(u,w)$ are very likely to be dependent. This suggests that the actual dimensionality of the *correlation graphs* is much smaller than d.

Following the idea of principle component analysis (PCA), our algorithm first projects *correlation graphs* to a low-dimensional space and then performs clustering on the projected data [18][20][21]. Let $X_1, X_2, ..., X_n$ denote the d-dimensional column vectors consisting of elements drawn from the upper triangles of $CG_{1,t}, CG_{2,t}, ..., CG_{n,t}$ respectively. Without loss of generality, assume X_i has zero mean. The original data matrix can be written as $X=[X_1, X_2, ..., X_n]$, which is of size $d \times n$. The goal of PCA is to maximize the variance of data after projection. Define the $m \times d$ projection matrix as $P=(u_1,u_2,...,u_m)^T$, where u_i is a d-dimensional unit column vector and m is the dimensionality of the subspace. Accordingly, the projected data of X_i is PX_i, which is an m-dimensional column vector in the low dimensional space, and the variance is:

$$\frac{1}{n}\sum_{i=1}^{n}(PX_i - P\overline{X})^2 = \frac{1}{n}\sum_{i=1}^{n}(PX_i)^2 = PSP^T$$

where $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i = (0\ 0\ ...\ 0)^T$ is the mean of projected data, and $S = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})(X_i - \overline{X})^T = \frac{1}{n}\sum_{i=1}^{n} X_i X_i^T$ is the covariance matrix.

We can rewrite the variance PSP^T as:

$$PSP^T = \begin{pmatrix} u_1^T \\ u_2^T \\ ... \\ u_m^T \end{pmatrix} S(\ u_1 \quad u_2 \quad ... \quad u_m \ ) = \sum_{i=1}^{m} u_i^T S u_i ,$$

which implies that maximizing PSP^T is equivalent to maximizing $u_i^T S u_i$ independently. Since u_i is a unit vector, a constraint is given by $u_i^T u_i = 1$. As a common strategy, a Lagrange multiplier can be added to find the maxima of the objective function subject to constraints. The Lagrange function is defined by $u_i^T S u_i + \lambda_i(1 - u_i^T u_i)$. When $Su_i = \lambda_i u_i$, this term has a stationary point, i.e., the partial derivatives are zero, which suggests that u_i is the eigenvector of S with eigenvalue λ_i. Hence $u_i^T S u_i = \lambda_i u_i^T u_i = \lambda_i$. If we take the largest eigenvalue and its corresponding eigenvector, $u_i^T S u_i$ is maximized. By induction, if we take the m largest eigenvalues and the corresponding eigenvectors, the variance PSP^T is maximized.

However, the commonly used methods to decompose eigenvectors of S (e.g., QR decomposition) have a computation cost of $O(d^3)$, which is not scalable with d. In our case, if we collect more status information, d increases in the form of a quadractic function of p, where p is the number of metrics. On the other hand, due to packet losses and the hardness of time synchronization, it is often difficult to ensure that in every time window, we can collect enough data to compute *correlation graphs* for each node. As a result, the number of *correlation graphs* to cluster is much smaller than, i.e., $n \ll d$. Based on this observation, our algorithm computes the eigenvectors of S in another way [20]. Substituting S in $Su_i = \lambda_i u_i$ using $S=n^{-1}XX^T$, we have

$$\frac{1}{n} XX^T u_i = \lambda_i u_i .$$

Then, after multiplying the equation by X^T in both sides, we obtain

$$\frac{1}{n} X^T X (X^T u_i) = \lambda_i (X^T u_i).$$

It implies that the vector $v_i = X^T u_i$ is the eigenvector of matrix $\frac{1}{n} X^T X$.

Moreover, the projected data matrix which lives in a m-dimensional space can be directly computed based on v_i. Denote the data matrix after projection as $Z=[Z_1, Z_2, ..., Z_n]$, where Z_i is the corresponding data point of X_i after projection, i.e., $Z_i = PX_i$, and

$$Z = [\ Z_1 \quad Z_2 \quad ... \quad Z_n \ ] = \begin{bmatrix} u_1^T X_1 & u_1^T X_2 & ... & u_1^T X_n \\ u_2^T X_1 & u_2^T X_2 & ... & u_2^T X_n \\ ... & ... & ... & ... \\ u_m^T X_1 & u_m^T X_2 & ... & u_m^T X_n \end{bmatrix}.$$

Since $v_i = X^T u_i = \begin{pmatrix} X_1^T u_i & X_2^T u_i & ... & X_n^T u_i \end{pmatrix}^T$, we have

$$Z = (\ v_1 \quad v_2 \quad ... \quad v_m \ )^T .$$

This implies that the data matrix after projection Z consists of eigenvectors of matrix $\frac{1}{n} X^T X$. Since matrix $X^T X$ is of size $n \times n$, the computation cost of its eigenvalue decomposition process is $O(n^3)$ rather than $O(d^3)$.

After projection, we perform K-Means clustering on the columns of projected data matrix Z. The confidence of Z_i to be suspicious is proportional to its distance to the nearest centroid. Note that K-Means clustering needs the number of clusters K beforehand. In our evaluation, we found that $K = 3$ works well when we set $m = 15$. A number of improvements such as QT clustering [23] have been proposed to remove this constraint. Another issue worth mentioning is that *correlation graphs* are constructed only in each time window, which yields a trade-off between detection delay and false alarm rate. If we set a long window, each node transmits a large amount of status data within the window, and the false alarm rate is low, at the cost of a long detection delay. On the contrary, if we set a short window, the detection delay is low, but more false alarms might be generated.

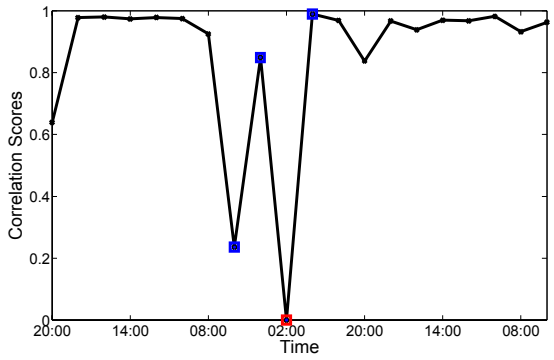


Figure 5. Correlation scores of TransmitCounter and ReceiveCounter.

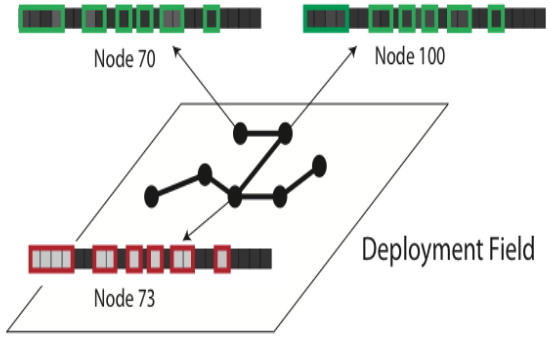


Figure 6. Rows of correlation matrices for nodes 70, 73 and 100.

IV. EVALUATION

We first present several case studies to show the effectiveness of Agnostic Diagnosis. Then, we study the relationship between the *correlation graph* changes and system performance. Finally, we statistically assess the results of our evaluation.

A. Case Studies

We use three months of data collected from 330 nodes in GreenOrbs. Using AD, we have identified four types of failures, some of which can be detected by the abnormal values of metrics, while others are silent ones.

The first type is called *ingress drops*, which is detected by the correlation between ReceiveCounter and TransmitCounter. Since the data rate of a sensor is fixed (i.e., three packets every fifteen minutes), the number of packets transmitted by a healthy node should be very close to the number of packets it receives from its children in the routing tree. However, if ingress drops occur, a portion of the incoming packets is dropped. Consequently, a change point in the time series of correlation score between ReceiveCounter and TransmitCounter suggests an ingress drop.

Figure 5 illustrates the correlation scores of these two metrics over time. Since they are always positively correlated, the correlation scores are larger than 0. In the diagram, there are four change points detected by temporal detection algorithm. Each of them is marked with a square, and the change point detected at 02:00 is the most significant one, i.e., with the highest confidence. An inspection of the raw data reveals that at that time, this sensor received packets from its children in the routing tree but did not forward all of them to the sink, causing ingress drops. However, if we investigate ReceiveCounter and TransmitCounter separately, this failure cannot be detected since the values of both metrics are within the normal range.

The second type is *routing failures*. A typical symptom is a node changing its parent too frequently. Figure 6 depicts one row of the correlation matrix for three sensors. Each pixel in the row stands for the correlation score between ParentChangeCounter and another metric. Similar to Figure 2, lighter pixels correspond to higher correlation scores. Node 73 has a quite different pattern compared to the others. As an example, consider the second pixel in the row, which corresponds to the correlation between ParentChangeCounter and RadioOnCounter. For healthy nodes 70 and 100, the two metrics are not correlated, as they signify the number of times a node turns on the radio and the number of times it changes its parent node in the routing tree, respectively. For node 73, the correlation score is so high that almost every time it turns on its radio, a parent change happens. Again, if we merely focus on the values of the two metrics independently, such failures cannot be found.

The third type is *link failures*. The detection of link failures exploits the correlation between RetransmitCounter and other metrics such as RadioOnTimeCounter. Figure 7 illustrates the time series of these two metrics. Initially, the metrics are weakly correlated, since a node may retransmit some packets when its radio is on, but it does not always happen. Starting at 8:30, the correlation increases, indicating that a large portion of radio-on time is used to retransmit packets. Thus, the transmission from this node to its parent suffers from low link quality. On the other hand, it is difficult to simply set a threshold for the ratio of retransmitted packets because it may change depending on the time and the location. For instance, a threshold equal to 0.5 may be acceptable in an indoor environment where links are easily interfered, but too low in an open space where link quality is very good.

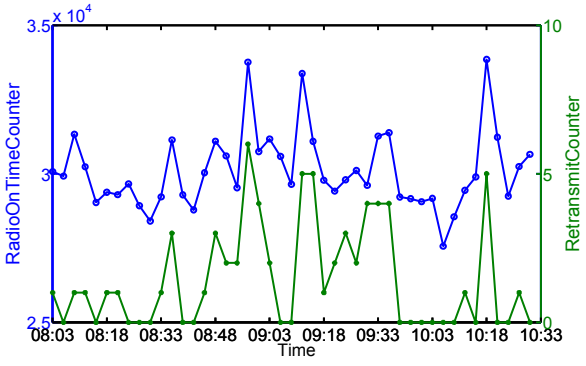
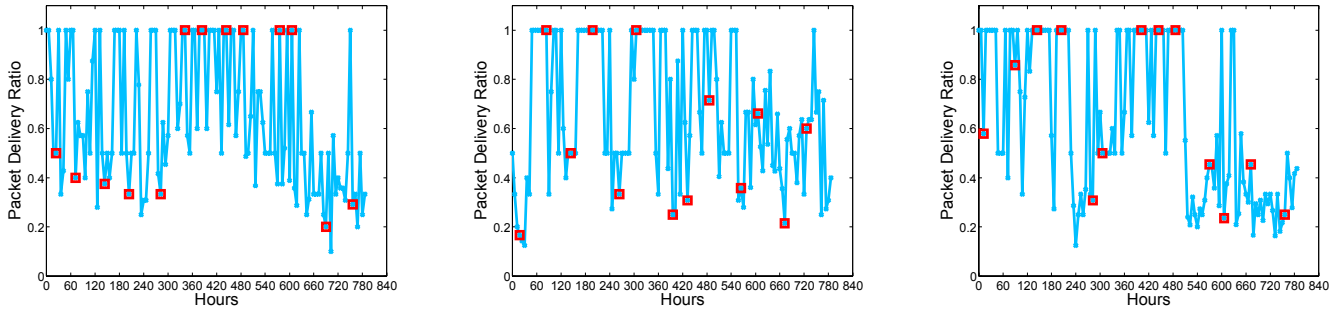


Figure 7. Time series of RadioOnTimeCounter and RetransmitCounter.



(a) Node 25 (b) Node 50 (c) Node 60

Figure 8. Packet delivery ratios of three nodes over time.

The fourth kind refers to *node failures* that may be caused by software or hardware problems. The correlation between TaskPostCounter and TaskExecuteCounter can be employed to uncover the internal status of programs. Generally speaking, if the node is healthy, the number of tasks performed (i.e., TaskExecuteCounter) should be highly correlated with the number of tasks posted (i.e., TaskPostCounter). However, if bugs occur, this correlation tends to be weak. We do not address hardware failures because they can be readily detected through a simple rule: when a node crashes, none of its neighbors can know its existence any more. Thus, if the node does not appear in the neighborhood of nearby sensors, it is highly possible that it has crashed.

B. *Effectiveness*

To evaluate the effectiveness of our agnostic approach we investigate data manually and compare them with the results of our system.

Temporal Detection: We depict both packet delivery ratios (PDR) and temporal detection results in Figure 8. PDR is defined as the ratio between the amount of packets received by the sink and that sent by the source node. For each square, its x-axis value corresponds to the time when the temporal detection phase reports change points, while its y-axis value corresponds to the packet delivery ratio at that time.

When failures occur to a sensor or its links, a possible symptom is that the node suffers from low packet delivery ratio. Local minima in the PDR curve are captured by the squares, suggesting that low PDRs correspond to the time instants when temporal detection discovers change points in the node's correlation graphs. However, some local minima are not captured by squares, possibly because they do not necessarily signify failures. For example, if node *s* is healthy but its parent is faulty, it also causes node *s*'s PDR to encounter a local minimum. Moreover, if the sensor recovers from a temporary failure, temporal detection also captures these time instants, i.e., the red squares that correspond to local maxima of PDR curve.

Spatial Detection: To validate the effectiveness of spatial detection, we randomly pick 30 faulty nodes that are identified in this phase. Specifically, 23 of them are flagged because some failures take place, 5 are false alarms, and the other 2 nodes exhibit a sudden increase in the number of packets received, leading to different correlation scores between PacketReceiveCounter and other metrics. However, after manually inspecting the raw data, we believe that these two nodes are healthy since the number of packets transmitted by them increases accordingly. The involved faults include packet loops, frequent parent changes, link failures, mismatch between radio-on time and packets transmitted/received, ingress failures, and stack overflows. These failures are not independent because several types of failures may occur in the same sensor node simultaneously. Packet loops is the most frequent type of failures.

C. *Traffic Overhead*

Since our diagnostic approach is a proactive one, it incurs additional traffic overhead. In our implementation, the packets storing these metrics are sent back to the base station along with the sensing data packets. Every fifteen minutes, all the status information can be packed in one packet, which is reasonable considering the benefits of diagnosis.

V. RELATED WORKS

Diagnosis of wireless sensor networks has been tackled from various perspectives. The first type aims at software debugging. Clairvoyant [1] is a GDB-like source-level tool that provides a suit of standard debugging commands such as break, set, watch and backtrace. Declarative Tracepoint [2] supports a declarative, SQL-like language, allowing developers to insert a set of action-associated rules to applications at runtime. DustMiner [3] employs a front-end to collect runtime event logs and a back-end to perform frequent pattern mining to uncover root causes of failures and performance anomalies. Our approach differs in that we do not aim at debugging the source code. Instead, our goal is to discover the silent failures caused by nodes, ambient environment, and protocols.

The second type of diagnosis is achieved through rule-based mining, or specific inference models. Sympathy [4] periodically collects general system metrics such as nodes' next hops and neighbors. Decision trees are then used to analyze root causes of node failures. PAD [8] leverages a packet marking strategy for constructing and maintaining the inference

model. PowerTracing [9] employs a special power meter and HMM to identify patterns of power consumption. Our work addresses the problem when there is little domain knowledge. Besides, since no specific models are assumed in our system, it has wider applicability.

There are also works closely related to diagnosis [10][13]. For example, [11] and [12] focus on node failures. LiveNet uses passive monitoring to reconstruct dynamics of live sensor networks. PD2 [14] pinpoints the root causes of application performance problems. SNMS [5] proposes a network management system designed for wireless sensor networks. Both MinRoute [6] and EmStar [7] visualize the operational sensor network.

VI. CONCLUSIONS

Most existing methods rely on pre-defined rules to discover faults with known types and symptoms. On the other hand, our experience with a large-scale sensor network reveals that diagnosis should be agnostic so as to discover silent failures. This paper presents Agnostic Diagnosis, a novel approach that relies on minimum domain knowledge. AD explores the correlation between system metrics using a two-stage cross validation scheme to detect silent failures. Our evaluation, based on a dataset collected from 330 nodes over several months, demonstrates that the two stages indeed capture system performance degradation and discover potential failures and network events.

ACKNOWLEDGEMENT

This work is supported in part by National Basic Research Program of China (973 Program) under Grant No. 2011CB302705, and NSFC/RGC Joint Research Scheme N HKUST602/08.

REFERENCE

[1] J. Yang, M. L. Soffa, L. Selavo, et al., "Clairvoyant: A Comprehensive Source-Level Debugger for Wireless Sensor Networks," In Proc. of ACM SenSys, 2007.

[2] Q. Cao, T. Abdelzaher, J. Stankovic, et al., "Declarative Tracepoints: A Programmable and Application Independent Debugging System for Wireless Sensor Networks," In Proc. of ACM SenSys, 2008.

[3] M. M. H. Khan, H. K. Le, H. Ahmadi, et al., "Dustminer: Troubleshooting Interactive Complexity Bugs in Sensor Networks," In Proc. of ACM SenSys, 2008.

[4] N. Ramanathan, K. Chang, L. Girod, et al., "Sympathy for the Sensor Network Debugger," In Proc. of ACM SenSys, 2005.

[5] P. Dutta, M. Grimmer, A. Arora, et al., "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," presented at the Proceedings of the 4th international symposium on Information processing in sensor networks, Los Angeles, California, 2005.

[6] L. Girod, J. Elson, A. Cerpa, et al., "EmStar: A Software Environment for Developing and Deploying Wireless Sensor Networks," In Proc. of USENIX Annual Technical Conference, 2004.

[7] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," presented at the Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA, 2003.

[8] K. Liu, M. Li, Y. Liu, et al., "Passive Diagnosis for Wireless Sensor Networks," In Proc. of ACM SenSys, 2008.

[9] M. M. H. Khan, H. K. Le, M. LeMay, et al., "Diagnostic powertracing for sensor node failure analysis," presented at the Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, Stockholm, Sweden, 2010.

[10] M. M. H. Khan, L. Luo, C. Huang, et al., "SNTS: Sensor Network Troubleshooting Suite," In Proc. of DCOSS, 2007.

[11] S. Guo, Z. Zhong, and T. He, "FIND: Faulty Node Detection for Wireless Sensor Networks," In Proc. of ACM SenSys, 2009.

[12] R. Rajagopal, X. Nguyen, S. C. Ergen, et al., "Distributed Online Simultaneous Fault Detection for Multiple Sensors," In Proc. of IEEE/ACM IPSN, 2008.

[13] B.-r. Chen, G. Peterson, G. Mainland, *et al.*, "LiveNet: Using Passive Monitoring to Reconstruct Sensor Network Dynamics," In Proc. of *IEEE/ACM DCOSS*, 2008.

[14] Z. Chen and K. G. Shin, "Post-Deployment Performance Debugging in Wireless Sensor Networks," In Proc. of IEEE RTSS, 2009.

[15] M. Li, Y. Liu, J. Wang, et al., "Sensor Network Navigation without Locations," In Proc. of IEEE INFOCOM, 2009.

[16] L. Mo, Y. He, Y. Liu, et al., "Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest," In Proc. of ACM SenSys, 2009.

[17] M. Basseville and I. Nikiforov, Detection of abrupt changes: theory and application, 1993.

[18] I. Jolliffe, Principal component analysis: Springer verlag, 2002.

[19] J. MacQueen, Some methods for classification and analysis of multivariate observations vol. 1: California, USA, 1967.

[20] C. Bishop, "Pattern Recognition and Machine Learning (Information Science and Statistics)," 2006.

[21] H. Zha, X. He, C. Ding, et al., “Spectral Relaxation for K-means Clustering”, In Proc. of NIPS, 2001.

[22] G. Werner-Allen, et al., "Lance: optimizing high-resolution signal collection in wireless sensor networks," 2008, pp. 169-182.

[23] Heyer, L.J., et al. “Exploring Expression Data: Identification and Analysis of Coexpressed Genes”. Genome Research, 9:1106-1115 (1999).

[24] K. Jonathan, J. Goldstein, R. Ramakrishnan, et al., "When Is" Nearest Neighbor" Meaningful?," In Proc. of ICDT, 1999.