QoS-aware Fault Detection in Wireless Sensor Networks

A. De Paola, G. Lo Re, F. Milazzo, M. Ortolani

In International Journal of Distributed Sensor Networks, vol. 2013, Article ID 165732, 12 pages, 2013

NDS Research Group - Networking and Distributed Systems
University of Palermo, ITALY
http://www.dicgim.unipa.it/networks

QoS-aware Fault Detection in Wireless Sensor Networks

Alessandra De Paola, Giuseppe Lo Re, Fabrizio Milazzo and Marco Ortolani
{*alessandra.depaola, giuseppe.lore, fabrizio.milazzo, marco.ortolani*}@unipa.it
DICGIM - University of Palermo
Viale delle Scienze Edificio 6, Palermo, Italy

Abstract

Wireless sensor networks (WSNs) are a fundamental building block of many pervasive applications. Nevertheless the use of such technology rises new challenges regarding the development of reliable and fault-tolerant systems.

One of the most critical issues is the detection of corrupted readings amidst the huge amount of gathered sensory data. Indeed, such readings could significantly affect the Quality of Service (QoS) of the WSN, and thus it is highly desirable to automatically discard them. This issue is usually addressed through "fault detection" algorithms that classify readings by exploiting temporal and spatial correlations. Generally, these algorithms do not take into account QoS requirements other than the classification accuracy.

This paper proposes a fully distributed algorithm for detecting data faults, taking into account the response time besides the classification accuracy. We adopt Bayesian networks to perform classification of readings and Pareto optimization to allow QoS requirements to be simultaneously satisfied.

Our approach has been tested on a synthetic dataset in order to evaluate its behavior with respect to different values of QoS constraints. The experimental evaluation produced good results, showing that our algorithm is able to greatly reduce the response time at the cost of a small reduction in classification accuracy.

1 Introduction

Wireless sensor networks (WSNs) are a pervasive computing technology that is experiencing broad diffusion and use in many ICT applications both in the industrial and research fields [1]. The typical WSN is composed by several resource-constrained devices (sensor nodes) capable of sensing environmental quantities, such as light intensity, air humidity and temperature; even though their nature allow for significant flexibility, thanks to the possibility of performing small on-board computations and of cooperating with each other, such nodes are also prone to faults of different nature that could weaken their effectiveness as pervasive sensory tool.

Malfunctions may be due to several causes, such as harsh environmental conditions, resulting into hardware failures, power outages, producing incorrect sensory readings or possibly altering communications, or sensor miscalibrations, that have an effect on the ADC transducer; consequently, they may be classified according to the architectural level they affect the most, i.e. *Network* level, *Node* level, or *Sensor* level [2]. Network faults include loss of connectivity, routing loops, congestion and they affect the exchange of data among node, so they are perceived as link failures. Node faults are malfunctions of the main components of the sensor node, i.e. radio, CPU, battery and memory; they include unexpected resettings, meaningless values of sensed data and poor quality of transmissions, and they are perceived as link failures and data failures. Sensor faults only affect the quality of sensed data and are always perceived as data failures.

In order to understand the factors that may negatively affect the QoS provided by a WSN, it is important to capture the root cause of a fault; this often represents a challenge for the researcher and requires the adoption of artificial intelligence techniques [3, 4]. Research has traditionally focused on the task of *fault detection* and in this work we specifically address *data* faults, which occur when gathered data does not provide a reliable representation of the monitored physical phenomenon; in this case, transmission and processing of these data clearly constitute a waste of energy and time and the overall QoS gets worse. On the contrary, early detection of data faults helps

reducing the amount of sensory data to be processed by high-level systems, and in-network fault detection is in fact a crucial functionality for many WSN-based applications [5].

A high accuracy of the fault detection algorithm contributes to address the reliability requirements of WSN applications; however, this is not the only relevant requirement of such type of pervasive system: other significant requirements maybe a low communication latency, a long WSN lifetime obtained through energy consumption reduction, and so on.

To the best of our knowledge, works proposed in literature coping with the fault detection problem neglect other QoS requirements. Here we propose QoS-aware fault detection (QAFD) implemented as a distributed in-network algorithm, that has the aim of maximizing the accuracy of the fault detection but also of minimizing the response time for such task; we address the issue of combining these two different and contrasting goals by exploiting the *constrained Pareto optimization*. Each node is left free to decide whether to cooperate with its nearby nodes by exchanging its readings with them; the cooperation increases the accuracy of the detection, instead the non-cooperation reduces the response time. The fault detection is performed by running probabilistic inference on a Bayesian network distributed over the whole WSN; the distributed architecture makes the system highly reactive to changes in the monitored phenomena and allows to reduce the computational effort required to the single node. The Bayesian network structure is also adapted at runtime to reflect the choice of cooperating or not performed by sensor nodes; the more complex the structure, the higher the classification accuracy of the WSN; the simpler the structure, the lower the response time.

2 Related works

Fault detection is a widely studied topic in WSN research, but its correct definition differs from that generally adopted in other fields. According to the classical point of view in statistics and data mining, a data fault corresponds to a data pattern not complying with a well defined normal behavior [6]. This definition does not apply to WSNs, because in such context we are usually unaware of the ground truth, thus the data fault definition can be modified as a "data pattern not conforming to the expected behavior of the monitored quantity".

Many works in WSN literature extend such a definition, by providing a complete taxonomy of data faults occurring in wireless sensor nodes. The authors of [2] classify data faults as *temporal* or *spatial* anomalies; the former class includes high variability, frozen outputs and out-of-bound errors, the latter class includes calibration errors and a burst of values that differs from the average in the neighborhood. Temporal anomalies can be detected locally, by considering temporal series of data gathered by a single node, while the detection of spatial anomalies mandatorily requires the comparison of a single sensory measurement with data gathered by surrounding nodes. Also the authors of [7] propose a similar classification of data faults, by distinguishing among *short* (single or multiple values of out-of-bound readings), *noise* (data with high variance) and *constant* (frozen output); for the purpose of our work, we adopted the latter taxonomy.

Fault detection algorithms are usually classified with respect to two different perspectives: architectural and methodological [2]. The architectural perspective distinguishes two types of approaches: *Centralized* [8, 9], and *Distributed* [10, 11]. In a centralized architecture, data flow from sensor nodes to the base station that performs fault detection by analyzing the whole amount of received data; this approach is limited by the resources of the base station that constitutes a single point of failure. Moreover, faults are discovered only after their transmission, thus causing a waste of energy and time inside the WSN. In a distributed architecture, sensor nodes monitor their status and label their readings as corrupted or not; if a reading is classified as corrupted, it could be locally discarded. The classification accuracy of such approaches is usually lower than the centralized ones; nevertheless, the fault detection occurs earlier with respect to centralized approaches, since it is not necessary to wait for the complete data transmission toward the base station.

From a methodological point of view, it is possible to distinguish between *Threshold based* approaches [11, 12], and *Classification based* approaches [13, 14, 15]. The former class uses thresholds to perform fault detection; just as an example, consider the *majority voting schemes* and the *cluster based* method, where thresholds are used to separate readings in "good" and "corrupted". Such approaches are generally not much expensive in terms of computational and time resources, but the classification accuracy is highly dependent from the chosen threshold values, thus performances are quite unpredictable unless an extensive empirical tuning is performed in order to find the optimal threshold values. The latter class avoids the use of thresholds, thus no human intervention is required to correctly tune the classifier. As an example, in this categories we can found *Bayesian Networks* and *Neural Networks* approaches; in both cases, during an off-line learning phase, a model is automatically learnt from raw data, while an on-line algorithm exploits the learnt model in order to perform fault detection. Such methods are computationally and time expensive, but their classification accuracy is predictable, since it is determinted by the learning phase.

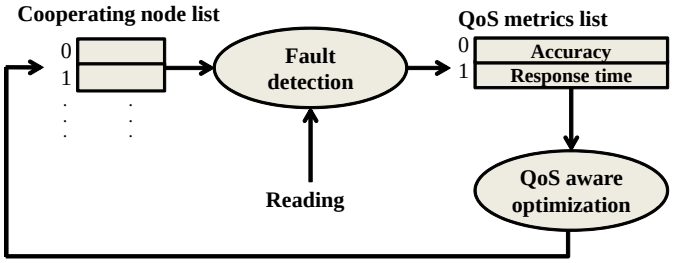

Figure 1: Block diagram of QAFD algorithm.

A great weakness for each of the described approaches is that they present a fixed computational complexity, strictly dependent from the adopted method; moreover the computational complexity highly affects the ideal upper bound of classification accuracy. As a consequence, in noisy scenarios, or when achieving a high classification accuracy is very important, it is suitable to adopt a more precise approach, even if it is more resource-demanding. On the contrary, in less dynamic scenarios, or when the focus is on the response time, it is more convenient to adopt a less expensive approach even if the classification accuracy is lower. To the best of our knowledge, none of the works presented in literature seems suitable to be adopted in every type of scenario, and with different priorities about application QoS requirements; in particular, none of them is able of tuning the classification accuracy and the computational complexity against the response time. In order to design a data-fault detection algorithm able to fuse different QoS requirements, we formulated such task as a multi-objective problem and we adopted Pareto optimization in order to deal with different and typically contrasting objective functions. The use of that theoretical framework is not totally new in the WSN research field, indeed it has been adopted for many different purposes. The authors of [16] define a set of metrics for evaluating network performances with respect to *reliability*, *lifetime* and *coverage*; the proposed experimental evaluation showed that the set of all possible solutions can be easily clustered and one of the identified cluster is Pareto optimal with respect to the considered qualitative dimensions. Finally, the authors of [17] cope with a multi-objective problem in an heterogeneous sensor network, and in that case the goal is to find the trade-off between the number of active sensor nodes and the network energy consumption.

3 Proposed approach

We propose a distributed algorithm for detecting corrupted sensory readings, by means of a Bayesian network distributed over sensor nodes. In such inference overlay network, the belief about the correctness of sensory readings flows thanks to communications among nodes. The use of a large Bayesian network allows for obtaining a high classification accuracy, but, at the same time, it might cause a high response time; for such reason, our system uses Bayesian networks with dynamic structure, in order to be able to adapt them according to the characteristics of the specific WSN deployment field. This adaptivity is obtained by allowing each sensor node to choose the set of neighbor nodes to cooperate with, and possibly also to choose to not cooperate at all. In the latter case, the sensor node tries to detect its own corrupted sensory readings by exploiting only temporal correlation among local measurements. On the contrary, if a node chooses to cooperate with its neighborhood, shared information is used as additional evidence, so that each node can also exploit spatial correlation for classifying its own readings. These independent and dynamic decisions drive the fault detection system toward the configuration that represents the best trade-off among all possibly contrasting application goals, such as high classification accuracy and low response time. Our distributed QoS-aware fault detection algorithm (QAFD) is composed by two main building blocks: a fault detection algorithm, based on Bayesian networks, and a QoS-aware optimization algorithm, periodically affecting the structure of the Bayesian networks. It is worth noting that both the fault detection and the QoS-aware optimization are locally performed in every sensor node. Figure 1 shows the interaction between these two logical blocks, within a single sensor node. The fault detection block takes as input the sensory reading to be classified and the set of neighbor nodes to cooperate with. Such block, beside classifying the latest sensory reading, produces a set of QoS indices, namely the classification accuracy and the response time. Such indices are then used by the QoS-aware optimization block for modifying the cooperating node list for the next step.

Figure 2 shows the dynamic arrangement of the overlay communication graph performed by the QAFD, in a simple running example. Let us suppose that at time $t = 0$ the overlay cooperation network includes all the sensor nodes in a single group, thus producing a very high classification accuracy. When the first optimization occurs ($t = T$), node 2 detects that there is some margin for reducing response time, thus it chooses to disconnect

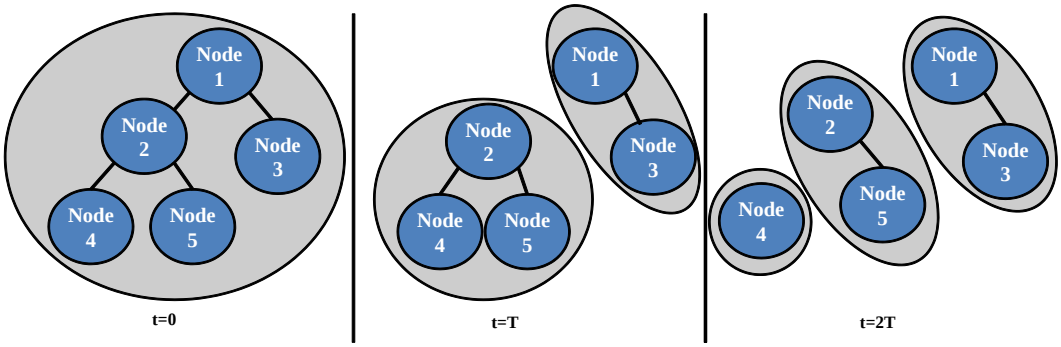

Figure 2: The dynamic arrangement of the overlay communication graph for five sensor nodes at different time instants.

from node 1 and it consequently modifies its cooperating node list $CN(2) = \{4, 5\}$. After T more steps, a new optimization occurs: because the classification accuracy is still sufficiently high, the node 2 chooses to disconnect also from node 4, thus its cooperating node list becomes $CN(2) = \{5\}$. The QAFD algorithm performs analogously to an online clustering, since the update of the cooperating node lists modifies the overlay communication graph. In order to ensure the convergence of the fault detection, we will assume for the rest of the paper that the underlying communication network is arranged as a tree, so that every cluster is still tree arranged.

3.1 Fault Detection

Fault detection is solved using distributed inference over a Bayesian network built on top of the cooperation network. The distributed inference algorithm takes as input a set of readings gathered by sensor nodes, and provides the classification of readings into corrupted and non corrupted ones. A first convergecast phase builds an initial estimate of the belief about reading classification, and a further broadcast phase refines such belief by adding information gathered over the whole cooperation network.

Bayesian networks are represented by a directed acyclic graph made up of random variables x_i, connected by directed links representing causal relations among variables. This model allows to take into account the probabilistic dependency between hidden random variables and observable random variables. The directed acyclic graph structure allows a simple computing of the likelihood function for the considered hidden variables. If $pa(x_i)$ denotes the set of parent variables for the random variable x_i, then the joint probability of the Bayesian network is:

$$p(x_1, ..., x_n) = \prod_i p(x_i | pa(x_i)). \tag{1}$$

In our system, each sensor node implements a small Bayesian network composed by one hidden variable c representing the (estimated) *class* of the current sensory reading and a set of observable variables representing the evidence probabilistically deriving from the true value of c. Observable variables can therefore considered as *features* related to the spatio-temporal correlations among readings.

Fault detection finds the combination of values for the hidden variable of each node, that maximizes its a posteriori probability, given the evidence. We demonstrated, in a previous work [18], that the Maximum a posteriori (MAP) approach is highly suitable to be implemented in WSNs, since it avoids the use of fixed thresholds and it is not too computationally expensive for sensor nodes.

Let $L = (l_1, ..., l_n)$ be the set of *local* observed variables, representing only the temporal correlation among the current and the past readings of a single node. If a node does not cooperate with its neighborhood, it can only exploit this evidence, and the structure of its Bayesian network becomes a *Naive Bayes classifier* [19], as shown in Figure 3(a).

Beside the local observed variables, the evidence variables include a set of *shared* observed variables, representing the spatial correlations among readings. If i and j are two cooperating nodes, then they share a set of variables $S_{i,j}$ that connect their two Naive Bayes classifiers by adding causal links from their hidden variables. The set of variables shared between two nodes i e j is a vectorial function of their last readings. Figure 3(b) shows two sensor nodes that linked their Naive Bayes classifiers through the use of shared variables; for the sake of simplicity, local and shared variables are grouped into the arrays of variables L and S.

There is a mapping between the current communication graph and the whole Bayesian network. The Bayesian network can be obtained from the communication graph by replacing each sensor node with a Naive Bayes structure and each communication link (i, j) with a shared variable $S_{i,j}$ and by adding two causal links from hidden variables c_i and c_j to the shared observable variable $S_{i,j}$. Figure 4 shows how the whole Bayesian network evolves in the previously considered running example. Dark shadowed ovals represent Naive Bayes classifiers implemented by each node, while light shadowed ovals represent cooperating clusters. It is possible notice that every pair of nodes belonging to the same cluster, is connected through the shared variables set.

The definition of the adopted fault detection method is completed by specifying the hidden and observed variables and how the MAP problem is solved.

Definition 1. *The hidden variable c takes values in the set* $C = \{\texttt{short}, \texttt{noise}, \texttt{constant}, \texttt{correct}\}$.

According to the taxonomy proposed in [7] we define: a `short` fault as a reading whose value is out of range for the monitored physical quantity; a `noise` fault as a burst of readings whose variance is higher than the environmental one, and finally a `constant` fault as a burst of readings that shows almost zero variations or, in other terms, that has a variance lower than the environmental one.

Definition 2. *Local observed variables are represented by a vectorial function of the last K readings of a single node and it is defined as* $L = \{l_1, l_2, l_3\} = \{\texttt{inner-gradient}, \texttt{repetitions}, \texttt{variance}\}$:

$$
\begin{aligned}
l_1 &= r_t - r_{t-1}, \\
l_2 &= \sum_{k=t}^{t-K+1} \mathbb{I}[|r_k - r_{k-1}| < \theta], \\
l_3 &= \sum_{k=t-K+1}^{t} \frac{(r_k)^2}{K} - \left(\sum_{k=t-K+1}^{t} \frac{r_k}{K} \right)^2,
\end{aligned}
\tag{2}
$$

where: r_t is the reading at time t; l_1 is the first derivative of r_t; l_2 is the number of consecutive readings, in a window of lenght K, that falls within a specific range θ and $\mathbb{I}[\cdot]$ is the indicator function (which is equal to 1 if the argument is true and 0 otherwise); l_3 is the variance of the last K readings.

Although the above set is not exhaustive and other features could be added to improve the classification accuracy, we found it is sufficient to recognize many temporal correlations among readings.

Definition 3. *Shared observed variables are represented by* $S_{i,j} = \{s_{i,j}\} = \{\texttt{outer-gradient}\}$:

$$
s_{i,j} = r_t^i - r_t^j, \tag{3}
$$

The shared variables set $S_{i,j}$ is, in principle, a vectorial function of the last K readings of sensor nodes i and j, but in practice we adopt a single function because it is sufficient to recognize spatial correlations; $s_{i,j}$ is simply the difference between the last reading sensed at node i and node j.

Definition 4. *The Maximium a Posteriori problem corresponds to finding the solution array* $c^* = (c_1^*, ..., c_n^*)$ *that maximizes the joint probability of the Bayesian network*:

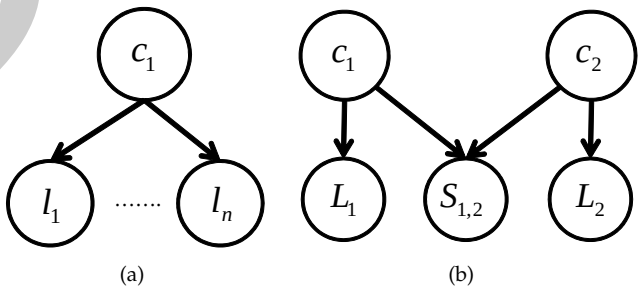

Figure 3: The Naive Bayes classifier, for a single node, including only local observed variables (a), and two Naive Bayes classifiers linked by shared observed variables (b).

Algorithm 1: Fault Detection Algorithm

Data: i, node id; CN, the cooperation network; R, the last W sensed readings.
Result: c^*, the optimal class label; p_{err} the probability of error for the chosen label; lat the latency (# of hops) experienced for computing c^*.

Convergecast
begin
  Compute local features using Eq. 2;
  Compute local belief using Eq. 5;
  for $j \in CN$ **do**
    Send the last reading $last(R)$ to j;
  end
  for $j \in CN$ **do**
    Receive the last reading of j;
  end
  Compute shared features using Eq. 3;
  if *i is a leaf node* **then**
    Compute the parent's reading belief $\Phi(c_j)$ using Eq. 6;
  else
    Receive $\Phi(c_i)$ from children in CN;
    Compute $\Phi(c_j)$ using Eq. 6;
  end
  if *i is not a root node* **then**
    Send $\Phi(c_j)$ to parent j;
  else
    Compute optimal class label c^* using Eq. 8;
  end
end

Broadcast
begin
  if *i is the root node* **then**
    for $j \in CN$ **do**
      Send c^* as optimal class label;
      Send 0 as latency;
    end
  else
    Receive c^* from parent node;
    Receive latency lat from parent node;
    Compute optimal class label c^* using Eq. 9;
    Compute p_{corr} using Eq. 10;
    Compute $p_{err} = 1 - p_{corr}$;
    if *i is not a leaf* **then**
      for $j \in CN$ **do**
        Send c^* as optimal class label;
        Send $lat + 2$ as latency;
      end
    end
  end
end

Output
return c^*, p_{err}, lat;

$$p(c_1, ..., c_n | L_1, .., L_n, S_{1,2}, ..., S_{n-1,n}) = \\ = \prod_{\substack{i \\ j \in CN(i)}} p(L_i|c_i) p(S_{i,j}|c_i, c_j) p(c_i), \quad (4)$$

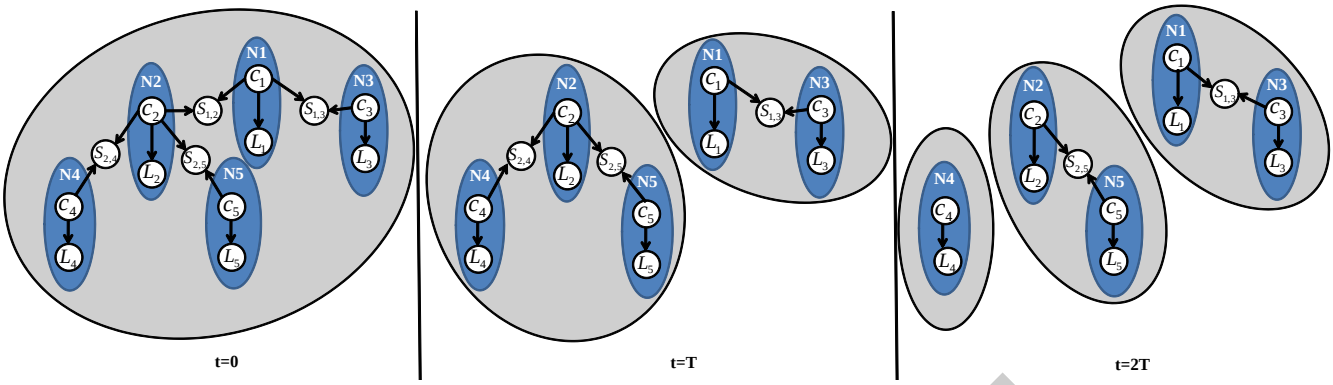

Figure 4: Evolution of the Bayesian network for a a simple network composed by five sensor nodes.

where (i, j) are pairs of cooperating nodes, and $CN(i)$ is the cooperating node list of node i. The above equation is obtained by applying Equation 1 to the particular form of Bayesian network induced by QAFD algorithm; it represents the joint probability of $(c_1, c_2, ..., c_n)$ of a single cluster of nodes and with slight variations could be extended for the whole WSN.

In order to solve this problem, we adapted the well-known "max-product" inference algorithm [20] to Bayesian networks. The pseudocode for the QAFD fault detection algorithm is shown in Algorithm 1. It includes two main phases: (i) a convergecast phase, estimating the probability distribution over all possible classes for each sensed reading, and (ii) a broadcast phase, determining the final class labeling. Since the communication network is arranged as a tree, we assume that any sensor node is aware of being a leaf, a root or an intermediate node. In order to take into account the latency of the algorithm, we also included a variable that tracks the number of hops (between any node and the root of the cluster) required for computing the class label.

The convergecast phase starts whenever fresh readings are to be classified. Each sensor node of the cluster computes the local features L_i as in Equation 2 and the *local belief* $\Lambda(c_i)$ as:

$$\Lambda(c_i) = p(c_i)p(L_i|c_i). \tag{5}$$

Such quantity represents the belief about the class of the sensory reading at node i, only considering local evidence. Successively, sensor nodes exchange their own readings with their direct neighbors, so they can compute also shared features as in Equation 3.

Each node i, with the exception of root node, sends a message to its parent j containing the *parent's reading belief* computed as follows:

$$\Phi(c_j)_{i\to j} = \max_{c_i} \phi(c_i, c_j). \tag{6}$$

The matrix $\phi(c_i, c_j)$ represents the joint belief about the classes of the readings of nodes i and j; such quantity takes into account *local belief*, *shared features*, and belief messages received from children nodes:

$$\phi(c_i, c_j) = p(S_{i,j}|c_i, c_j)\Lambda(c_i) \prod_{z\in CN(i)/j} \Phi(c_i)_{z\to i}, \tag{7}$$

where the productory is replaced by 1 if i has no children.

Root node r terminates the convergecast phase and computes its optimal class assignment as:

$$c_r^* = \underset{c_r}{argmax}\ \Lambda(c_r) \prod_{z\in CN(r)} \Phi(c_r)_{z\to r}. \tag{8}$$

Then, the root node starts the broadcast phase, during which optimal classes c^* are propagated over the tree. Every non-root node i receives the label c_j^* from its parent j and computes its own optimal class as:

$$c_i^* = \underset{c_i}{argmax}\ \phi(c_i, c_j^*), \tag{9}$$

and then propagates such quantity to its children. This phase ends at leaf nodes and produces the solution of the MAP problem.

The performance of the QAFD fault detection algorithm can be evaluated through the probability of correct classification p^i_{corr}, computed as:

$$p^i_{corr} = \phi(c^*_i, c^*_j). \tag{10}$$

It is also worth pointing out that the conditional probability tables, namely $p(L_i|c_i)$ and $p(S_{i,j}|c_i, c_j)$, and the class priors $p(c_i)$, are computed by an off-line supervised learning, based on sensor readings recorded during one day and manually labeled, through a simple frequentist approach.

3.2 QoS-aware Optimization

The fault detection algorithm is performed starting from a given structure of the communication network, defined in terms of clusters. The classification accuracy and the response time highly depend on the size of network clusters. In order to find the optimal cluster structure, we propose a dynamic and distributed algorithm that charges sensor nodes with determining neighbor nodes to cooperate with, on the basis of QoS indices associated with different configurations. The main goal is to find the network configuration representing the best trade-off among several application-driven QoS goals and constraints.

This QoS-aware optimization is performed through Pareto optimization, that allows to consider multiple objective functions, possibly contrasting and with non-comparable units of measurement.

Our system can be seen as a complex system where each agent interacts with the environment by taking decisions $d \in D$ at each time step of their lifetime. Before taking any decision, an agent can evaluate its goodness by means of a set of QoS metrics $m_d = (q_1, ..., q_n)$. In order to allow an agent to choose the better decision to be taken, it is required to define an order relation over the space of m_d; the Pareto dominance is the order relation we chose to adopt. Let m_{d_1} and m_{d_2} be the quality metrics arrays respectively for the decisions d_1 and d_2, then m_{d_1} *Pareto dominates* m_{d_2} ($m_{d_1} \preceq m_{d_2}$) if each component k of m_{d_1} is better then the corresponding component of m_{d_2}. Such definition, in a minimization problem, corresponds to the following equation:

$$m_{d_1} \preceq m_{d_2} \Leftrightarrow \left\{\forall k = 0, \ldots, n => m_{d_1}(k) \leqslant m_{d_2}(k)\right\}. \tag{11}$$

A Pareto optimal decision is defined as a decision that is not dominated by any other decision:

$$d^* = \{d_i \in D : \forall d_j \in D, d_j \neq d_i \Rightarrow m_{d_i} \preceq m_{d_j}\}. \tag{12}$$

The set of Pareto optimal solutions constitutes the Pareto optimal front.

In addition to multiple objective functions, a real application may be characterized by a set of constraints about QoS requirements. We take into account these constraints by representing them as points in the QoS metric space, namely $v = (v_1, ..., v_n)$. A decision d is said to be *admissible* if and only if its quality metric array $m_d \preceq v$. This further check allows to eliminate possible Pareto optimal solutions that break at least one QoS constraint.

In our approach, each sensor node i is an agent of the overall system (the WSN), and it has to periodically choose the subset of neighbor nodes to cooperate with. If $N(i)$ is its neighborhood set, then the output of the Pareto optimization is the decision d^* that drives the cooperating node list, $CN(i) \subseteq N(i)$, to the combination corresponding to the optimal value of the metrics array m_{d^*} that also satisfies the constraints array v.

The following of this section specify all the components of our QoS-aware optimization.

Definition 5. *The decision d for node i is defined as a pair of values $(action, j)$, where $action \in \{connect, disconnect, do\text{-}nothing\}$, and $j \in N(i)$.*

Each decision of the node i corresponds to a single atomic action affecting its cooperating node list, $CN(i)$. In particular i can choose to *connect* to or *disconnect* from one of its neighbors, or either to do nothing; this latter decision is taken when the current cooperating node list is Pareto optimal, and thus $CN(i)$ is left unchanged.

Definition 6. *The QoS metrics vector corresponding to a decision d is defined as $m_d = (q_{err}, q_{lat})$:*

$$\begin{aligned} q_{err}(t) &= \frac{\sum_{j \in CN(i)} \sum_{\tilde{t}=t-T+1}^{t} p^j_{err}(\tilde{t})}{|CN(i)| \cdot T}; \\ q_{lat}(t) &= \begin{cases} \max\limits_{j \in CN(i)} \{lat^j(t) + 2\} & \text{if } |CN(i)| > 0 \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{13}$$

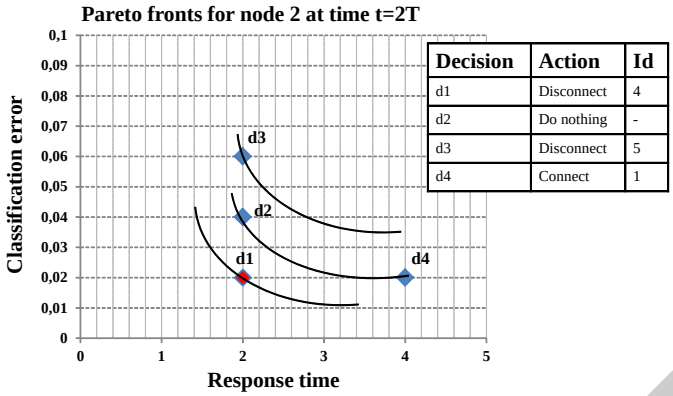

Figure 5: Example of a two dimensional Pareto optimization; the x axis represents the response time, while the y axis is the classification error.

Such metrics are used for predicting the goodness of each possible decision d. The first metric, q_{err}, represents the average classification error of the nodes belonging to the chosen configuration of $CN(i)$ over a time window of length T; clearly, in order to compute such value, any node j of the cluster simply stores the values of p^j_{err} for its last T readings.

In order to define the second metric, it is necessary to consider that the response time of the inference algorithm is proportional to the number of hops between i and the furthest node in its cluster; in the worst case i is a leaf of the tree representing the cluster topology, so the response time for node i will be at most equal to the maximum value of the response time ($lat^j(t)$) for nodes in $CN(i)$, plus the two messages used for broadcast and convergecast on the link of i.

The satisfiability of a decision is verified by checking that the corresponding metrics vector is not dominated by the constraints, that is $(q_{err}, q_{lat}) \preceq (v_{err}, v_{lat})$. Such constraints are used as imposed upper bound for the values of the QoS metrics and they should be manually chosen by the application programmer.

In summary, node i evaluates the impact of changing its cooperating node list by forecasting the values of the future QoS metrics for the decisions of connecting or disconnetting from each node $j \in CN(i)$. Resulting solutions are filtered by cutting-off all those that are not admissible with respect to the specified constraints. The filtered solutions are ordered making use of definition 11 and the optimal front is found. If several solutions belong to the Pareto optimal front, a random decision among them is selected. Finally, it is also worth noting that it is possible that no admissible solutions are found after filtering; this occurs whenever the Pareto optimal front is placed beyond the constraints polyhedron. To cope with this situation, the constraints are relaxed by making all QoS metrics unbounded and then letting the QoS-aware optimization algorithm choose the Pareto optimal solution that is considered as the "less unsatisfatory" one. The pseudocode for the described optimization algorithm is summarized in Algorithm 2.

Definition 7. *The constraints array is defined as* $v = (v_{err}, v_{lat})$.

Figure 5 shows, in our running example, the Pareto optimization performed by the sensor node 2 at time $2T$ for the classification error and the response time. Node 2 can choose among four different decisions, on the basis of its current cooperating list ($CN(2) = \{4, 5\}$); among such decisions, d_1 is the Pareto optimal one because its QoS metrics are lower than those of other decisions. It is also worth noting that decisions are grouped into *Pareto fronts*; d_2 and d_4 belong to the same Pareto front, so they are Pareto equivalent.

In conclusion, QAFD algorithm is composed by two main blocks: a fault detection block activated for each sensory reading, and a Pareto optimization modifying the communication graph over which the fault detection is performed, occurring only every T time steps. The QAFD is intended as a fully distributed algorithm where each sensor node runs the same program whose pseudocode is summarized in Algorithm 3.

4 Experimental results

This section describes the experimental evaluation of our QAFD algorithm, with the aim of proving that our approach allows sensor nodes to adapt their behavior with respect to the imposed QoS requirements and the dynamics

Algorithm 2: QoS-aware Optimization Algorithm

Data: M_d, array of QoS metrics for all admissible decisions.
Result: d^*, the Pareto Optimal decision.

Pareto Optimal Front Generation
begin
  for $i \in M_d$ **do**
    mark i as pareto optimal;
    for $j \in M_d, j \neq i$ **do**
      if $M_d(j) \prec M_d(i)$ **then**
        mark $M_d(i)$ as non-optimal;
      end
    end
    if *i is marked as optimal* **then**
      add i to Pareto Optimal front;
    end
  end
end

Pareto Optimal decision
begin
  if *Many solutions belongs to Pareto Optimal front* **then**
    set d^* to a random i of the Pareto Optimal front;
  else
    set d^* as the unique i of the Pareto Optimal front;
  end
end

Output
return d^*;

of the real deployment scenario.

We simulated a tree-arranged WSN, characterized by depth 16 and branch factor 3, and composed by 100 sensor nodes gathering temperature measurements every 30 seconds for two days. In order to build the simulated dataset we used 10 Mica2Dot sensor nodes as real generators of readings, and then we modified this basic dataset by adding 10 different Gaussian noise signals $\mathcal{N}(0, \sigma_E^2)$, with $\sigma_E^2 = 0.02$, to the sensory signal gathered by each real sensor node. Data faults have been simulated by corrupting the obtained dataset, according to fault definitions proposed in [7]:

- *Short:* $\tilde{r}(t) = g \times r(t)$
- *Noise:* $\tilde{r}(t) = r(t) + \mathcal{N}(0, \sigma_N^2)$
- *Constant:* $\tilde{r}(t_0 + i) = r(t_0) + \mathcal{N}(0, \sigma_\theta^2), i \in \{1, .., k\}$,

where $g = 1.5$ is a gain constant, $\sigma_N^2 = 3$ and $\sigma_\theta^2 = 10^{-9}$ are Gaussian noises, t_0 is a random time instant where a constant fault starts and $K = 10$ is its duration.

Different scenario dynamics have been simulated by generating three different corrupted datasets where the amount of data faults corresponds respectively to 20%, 30% and 40% of total number of readings. In each dataset, different classes of faults were mixed in equal parts. For each dataset, we used the first day of readings as training set to learn conditional probability tables for the Bayesian network, while the second day was used as the test set.

4.1 Comparison with benchmarks

The performances of QAFD algorithm, with different QoS constraints, were compared against two benchmarks, corresponding to the max-product inference algorithm over two different static network topologies. The first topology corresponds to a fully connected network, i.e. a single cluster where 100% of nodes cooperate; here the average response time is static and corresponds to 28 hops, and the classification accuracy achieves its upper bound. The second one corresponds to a network where none of the nodes cooperate, i.e. the number of clusters is equal to the

Algorithm 3: QAFD Algorithm

Data: t, the current time; m, the current quality metrics; R, the last K readings; CN, the current cooperating node list; v, the array of constraints.

Result: c^*, the most probable class for the last reading; p_{err}, the probability of error; lat, the response time; CN_{new}, the next cooperating node list.

Fault detection
begin
  Run *Fault detection (CN,R)* $\rightarrow c^*, p_{err}, lat$;
end

Pareto optimization
begin
  if *t is multiple of T* **then**
    for $d \in D$ **do**
      Compute QoS metrics m_d using Equation 13;
      Accept d only if $m_d \preceq v$;
    end
    if *no admissible solutions found* **then**
      Accept all d as admissible;
    end
    Run *QoS-aware Optimization*$(m_{d_1}, m_{d_2}, ..., m_{d_n}) \rightarrow d^*$;
    Update CN according to the decision d^*;
  end
end

Output
return c^*, p_{err}, lat, CN;

number of nodes; in this benchmark the average response time is equal to 0 and the classification accuracy drops to its lower bound.

We want to show that the adaptive behavior of QAFD makes it able to meet the imposed constraints, while paying a small cost in terms of the possibly unconstrained metrics; at the same time, although the static topologies achieve better performance with respect to one QoS requirement, they dramatically worsen the other one. We adopted two different set of QoS constraints, namely:

$$\begin{aligned} v_1 &= (v_{err} = 0.06, v_{lat} = unconstrained), \\ v_2 &= (v_{err} = unconstrained, v_{lat} = 2). \end{aligned} \tag{14}$$

Sensor nodes run the Pareto optimization every $T = 100$ readings.

Performances are evaluated by using the two considered QoS metrics, namely the classification accuracy (a value between 0 and 1) and the response time (measured in numbers of hops), averaged over time windows of size T and over all the networks nodes.

Figure 6 compares the performances of the four considered configurations in three different scenarios. As expected, the "100% cooperating node" configuration always achieves the best classification accuracy and the worst response time in every scenario, while the "0% cooperating node" configuration is the best for response time and the worst for the classification accuracy.

Analyzing the performance of QAFD with a constraint over the classification accuracy (v_1), it is possible to note that such metric approaches the constraint for every scenario, paying an increasing cost in response time as the corruption percentage increases. Moreover, while the classification accuracy is quite close to the "100% benchmark", in comparison, its response time obtains a significant reduction.

Analogously, QAFD with a constraint over the response time (v_2) always meets such requirement by paying a small decrease in the classification accuracy as the corruption percentage increases. Moreover, at the cost of a small increase of response time with respect to the "0% benchmark", the classification accuracy is remarkably higher.

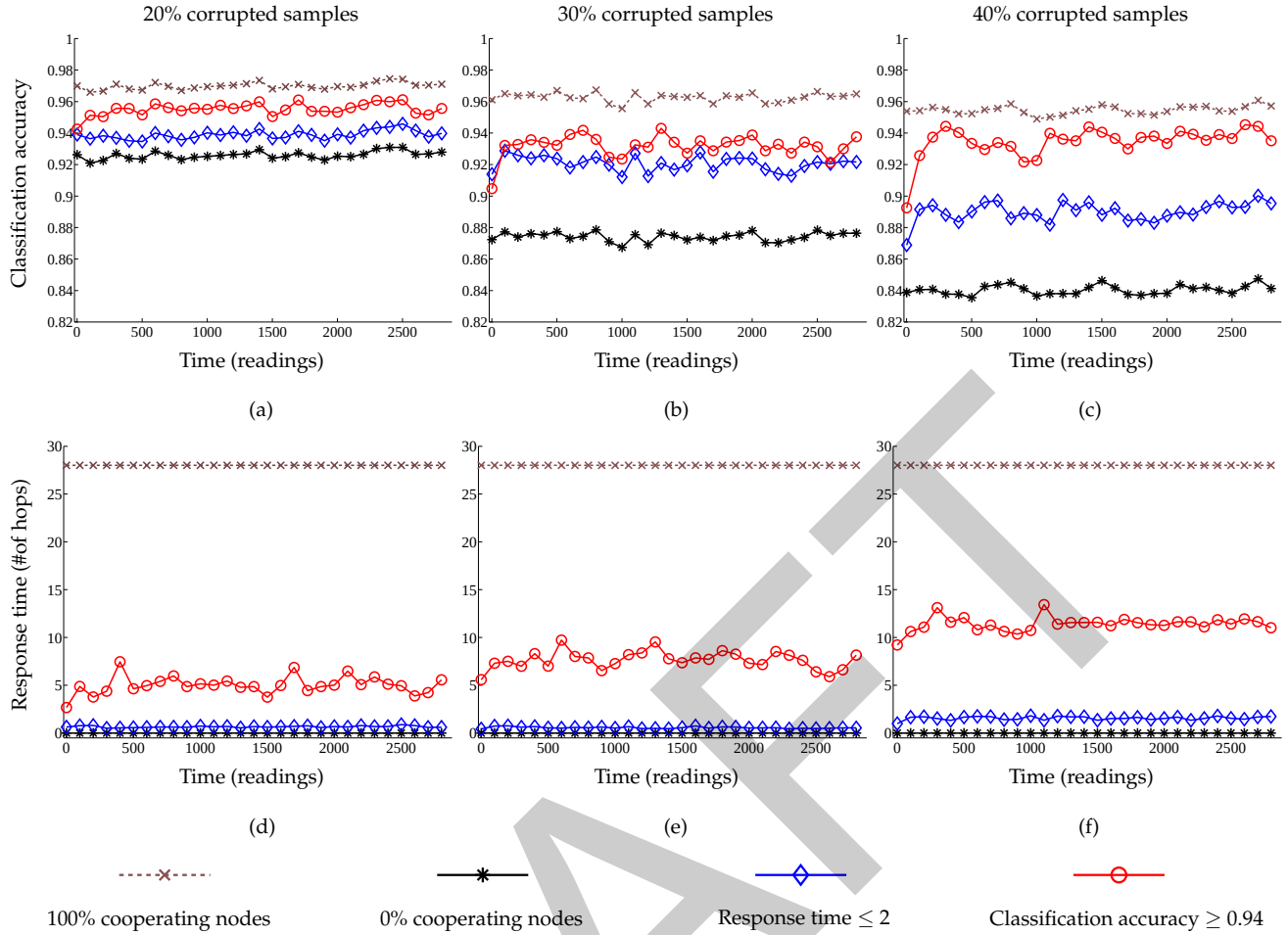

Figure 6: The performance of QAFD for two different constraints compared to those of two fixed network topologies while the amount of corrupted samples spans from 20 to 40 percent.

4.2 Adaptivity of QAFD to different scenarios

We aim to demonstrate the adaptability of QAFD to different scenarios, namely to different percentages of corruption in sensory data. This adaptability corresponds to tuning the unconstrained QoS metrics in order to satisfy the constrained ones.

Figures 7 and 8 show this property by comparing the performances of QAFD, respectively with constraints v_1 and v_2, for the three considered scenarios. In particular, Figure 7 shows how QAFD behaves when a constraint over the classification accuracy is imposed; as shown, QAFD rapidly adapts the network communication graph, thus increasing the response time for higher percentage of corruption. Analogously, Figure 8 shows that, as the amount of corrupted samples increases, QAFD with a constraint over the response time is able to tune the classification accuracy by decreasing it after few time steps.

Such results show that QAFD is able to find the correct trade-off among the QoS metrics, satisfying the imposed constraints and taking implicitly into account also the dynamics of the real deployment scenario.

4.3 Relationship between response time and classification accuracy

Obtained experimental results allows a qualitative analysis of the relationship between response time and classification accuracy in three different scenarios, as shown in Figure 9. The plots shown were obtained by averaging values obtained in the four settings described in previous experiments.

Trends are quite similar for all the three scenarios: the minimum value of the classification accuracy is obviously achieved for singleton clusters; on the contrary, a fully connected network gets the maximum of the classification accuracy.

A small increase of the allowed response time, for small cluster size, corresponds to great increase of the classifi-

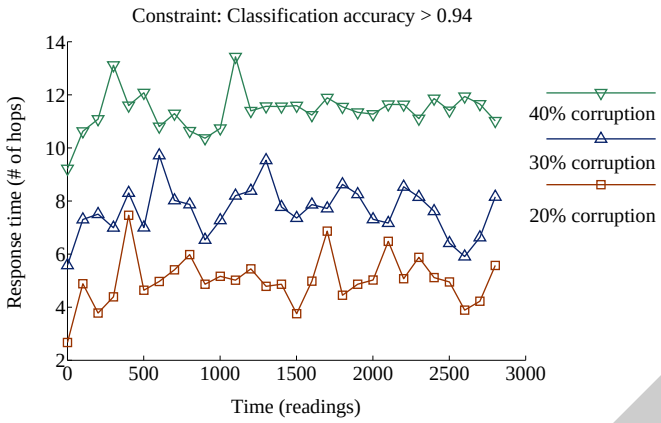

Figure 7: A comparison among the average response time for different amounts of corruption with constraint on classification accuracy.

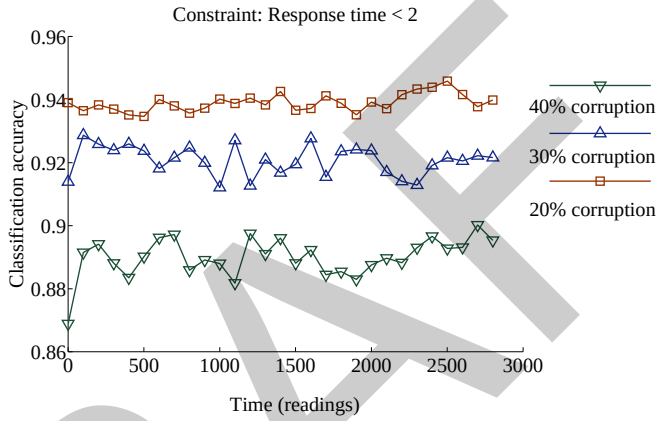

Figure 8: A comparison among the average classification accuracy for different amounts of corruption with constraint on response time.

cation accuracy. This behavior is more visible when the corruption amounts to 20%; in this scenario a response time proportional to 5 hops corresponds to a classification accuracy very close to its maximum value.

The three plots tend to asymptotically reach their maximum value as the response time increases. Such evidence allows us to deduce that farther nodes have a small influence on the choice of the correct class label; this behavior is strictly related to the characteristic of the monitored physical quantity (e.g. the temperature), and simply means that spatial correlation decreases as the distance between nodes increases.

5 Conclusion

This paper proposed a distributed fault detection algorithm for WSN, able to find the optimal trade-off among different and possibly contrasting QoS requirements. Even if we considered only classification accuracy and response time, many other QoS requirements could be considered, provided that the corresponding metric is defined.

The fault detection algorithm we proposed performs a probabilistic inference on Bayesian networks distributed over the wireless sensor nodes. A QoS-aware Pareto optimization algorithm allows to adapt the Bayesian network structure according to the considered QoS metrics.

Experimental results showed the capability of our system to dynamically tune its behavior, also according to different dynamics of the real deployment scenario, and to optimize QoS metrics if compared with static approaches.

In a future work we are going to include metrics related to network energy consumption and link quality, that are other relevant QoS requirement of WSN applications.

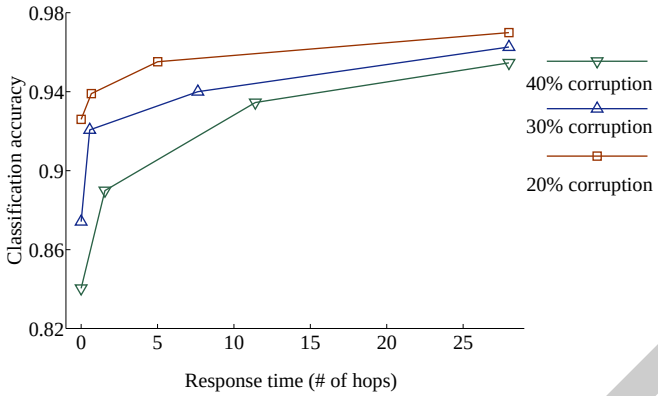

Figure 9: Average response time and accuracy for the three considered scenarios.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: a Survey," *Computer Networks*, vol. 38, no. 4, pp. 393– 422, 2002.

[2] R. Jurdak, X. Wang, O. Obst, and P. Valencia, "Wireless Sensor Network Anomalies: Diagnosis and Detection Strategies," in *Intelligence-Based Systems Engineering*, ser. Intelligent Systems Reference Library. Springer Berlin Heidelberg, 2011, vol. 10, pp. 309–325.

[3] S. Gaglio, L. Gatani, G. Lo Re, and A. Urso, "A logical Architecture for Active Network Management," *Journal of Network and Systems Management*, vol. 14, no. 1, pp. 127–146, 2006.

[4] A. De Paola, S. Fiduccia, S. Gaglio, L. Gatani, G. Lo Re, A. Pizzitola, M. Ortolani, P. Storniolo, and A. Urso, "Rule Based Reasoning for Network Management," in *Proceedings of the Seventh International Workshop on Computer Architecture for Machine Perception (CAMP 2005)*, 2005, pp. 25–30.

[5] A. De Paola, S. Gaglio, G. Lo Re, and M. Ortolani, "Sensor9k: A testbed for designing and experimenting with WSN-based ambient intelligence applications," *Pervasive and Mobile Computing*, vol. 8, no. 3, pp. 448–466, 2012.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[7] A. Sharma, L. Golubchik, and R. Govindan, "Sensor faults: Detection methods and prevalence in real-world datasets," *ACM Transactions on Sensor Networks*, vol. 6, no. 3, pp. 23:1–23:39, 2010.

[8] K. Ni and G. Pottie, "Bayesian Selection of Non-faulty Sensors," in *IEEE International Symposium on Information Theory (ISIT 2007)*, 2007, pp. 616–620.

[9] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, "On-line fault detection of sensor measurements," in *Proceedings of IEEE Sensors*, vol. 2, 2003, pp. 974–979.

[10] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng, "Localized Outlying and Boundary Data Detection in Sensor Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1145–1157, 2007.

[11] J. Chen, S. Kher, and A. Somani, "Distributed Fault Detection of Wireless Sensor Networks," in *Proceedings of the 2006 Workshop on Dependability issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS '06)*, 2006, pp. 65–72.

[12] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, "Distributed anomaly detection in wireless sensor networks," in *Proceedings of the 10th IEEE Singapore International Conference on Communication systems (ICCS 2006)*, 2006, pp. 1–5.

[13] X. Wang, J. Lizier, O. Obst, M. Prokopenko, and P. Wang, "Spatiotemporal Anomaly Detection in Gas Monitoring Sensor Networks," in *Proceedings of the 5th European conference on Wireless sensor networks*, 2008, pp. 90–105.

[14] M. Markou and S. Singh, "Novelty detection: a review–part 2: Neural Network Based Approaches," *Signal Processing*, vol. 83, no. 12, pp. 2499–2521, 2003.

[15] P. Yang, Q. Zhu, and X. Zhong, "Subtractive Clustering Based RBF Neural Network Model for Outlier Detection," *Journal of Computers*, vol. 4, no. 8, pp. 755–762, 2009.

[16] R. Hoes, T. Basten, C. Tham, M. Geilen, and H. Corporaal, "Quality-of-Service Trade-Off Analysis for Wireless Sensor Networks," *Performance Evaluation*, vol. 66, no. 3-5, pp. 191–208, 2009.

[17] A. De Paola, S. Gaglio, G. Lo Re, and M. Ortolani, "Multi-Sensor Fusion through Adaptive Bayesian Networks," in *AI*IA 2011: Artificial Intelligence Around Man and Beyond*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6934, pp. 360–371.

[18] G. Lo Re, F. Milazzo, and M. Ortolani, "A Distributed Bayesian Approach to Fault Detection in Sensor Networks," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 634–639.

[19] H. Zhang, "The optimality of Naive Bayes," in *Proceedings of the 17th International Florida Artificial Intelligence Research Society Conference*, 2004, pp. 562–567.

[20] Y. Weiss and W. Freeman, "On the Optimality of Solutions of the Max-Product Belief-Propagation Algorithm in Arbitrary Graphs," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 736–744, 2001.