Mobile Information Systems 6 (2010) 229–247
DOI 10.3233/MIS-2010-0101
IOS Press

Fault reconnaissance agent for sensor networks

Elhadi M. Shakshuki[a,*], Xinyu Xing[b] and Tarek R. Sheltami[c]
[a]*Jodrey School of Computer Science, Acadia University Wolfville, Nova Scotia, B4P 2R6 Canada*
[b]*Department of Computer Science, University of Colorado at Boulder, CO, USA*
[c]*Computer Engineering Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia*

Abstract. One of the key prerequisite for a scalable, effective and efficient sensor network is the utilization of low-cost, low-overhead and high-resilient fault-inference techniques. To this end, we propose an intelligent agent system with a problem solving capability to address the issue of fault inference in sensor network environments. The intelligent agent system is designed and implemented at base-station side. The core of the agent system – problem solver – implements a fault-detection inference engine which harnesses Expectation Maximization (EM) algorithm to estimate fault probabilities of sensor nodes. To validate the correctness and effectiveness of the intelligent agent system, a set of experiments in a wireless sensor testbed are conducted. The experimental results show that our intelligent agent system is able to precisely estimate the fault probability of sensor nodes.

Keywords: Management, intelligent agents, expectation maximization algorithm, wireless sensor networks

1. Introduction

An embedded sensor network is a system of nodes, each of which is equipped with a certain amount of sensing, actuating, computation, communication and storage components. Two major components of sensor nodes are sensing unit and wireless transceiver. They directly interact with nodes in wireless sensor networks (WSNs) that are easily prone to failure due to hardware failure, communication link errors, energy depletion, malicious attacks, etc. Even if the sensor node hardware is in excellent condition, still the communication between sensor nodes is dependent upon many factors such as signal strength, obstacles and interferences. Degradation in these factors results in low reliability of sensor nodes. One of the key prerequisite for a scalable, effective and efficient sensor network is the utilization of low-cost, low-overhead and high-resilient fault-inference techniques. To this end, we propose an intelligent agent system with a problem solving capability to address the issue of fault inference in sensor network environments. The core of the intelligent agent system is the problem solver, which implements a fault-detection inference engine that harnesses Expectation Maximization (EM) algorithm to estimate fault probabilities of sensor nodes.

Due to the characteristics (e.g. energy awareness, constraint bandwidth and so on) of wireless sensor networks, it is infeasible for each sensor to announce its working state to a centralized node (base station).

*Corresponding author: Jodrey School of Computer Science, Acadia University, Nova Scotia, Canada B4P 2R6, E-mail: elhadi.shakshuki@acadiau.ca.

Therefore, we propose an intelligent agent system which utilizes EM algorithm to infer the sensor-node fault probabilities from a limited dataset. The benefits of our intelligent agent system are summarized as follows. (1) The intelligent agent system does not involve extra message transmission, since it can infer sensor-node fault probabilities directly from aggregated dataset that sensor nodes collect. (2) The intelligent agent system can provide highly accurate estimation.

In WSNs, it is natural that sensor nodes experience some faults at high frequencies due to the following two factors [2]. Firstly, WSNs put significant constraints on the resource expenditure. Nodes operate under strict energy constraints, which limits the amount of energy devoted to testing and fault tolerance. Secondly, some applications are equally complex with the involved technology and architecture. Sensor networks often operate without human intervention [31]. Furthermore, security and privacy concerns prevent extensive testing procedures. It should be noted that not only testing and fault tolerance are adversely impacted, but related tasks such as debugging where reproduction of specific conditions under which fault has occurred are extremely difficult. Similarly, sensor nodes are often deployed in uncontrolled and sometimes even hostile environment for surveillance and detection [35,36]. Therefore, knowing sensor-node working states is indispensable.

Due to the energy awareness characteristics in WSNs, there is a trade-off between prolonging the network lifetime via conserving the energy of individual nodes, and maintaining the high quality of network services by implementing complex fault management schemes in a sensor network. To the best of our knowledge, contemporary fault management mechanisms have to involve extra traffic overhead and energy expenditure. Therefore, most researchers regularly believed that the expenditure for fault management is fairly expensive. Based on our previous study and the following two motivations; we however, introduce an intelligent agent named ATLAS (An inTelligent agent for fauLt reconnAisSance in sensor networks).

1.1. Motivation-1: Energy awareness based on data aggregation

Over the past few years, several researches have attempted to propose various mechanisms to prolong the network lifetime [32]. An agent-based paradigm is a trend [3–9]. This paradigm adopts energy-efficient data aggregation to eliminate the redundant transmission and thus to minimize energy consumption. Under the agent-based paradigm, communication links and partial nodes in a sensor network are usually perceived as a reverse multicast tree. Generally, a sink node dispatches mobile agents with processing code to destination nodes through the reverse multicast tree. After collecting the sensing data from the destination nodes, mobile agent migrates to the sink carrying aggregated data. To the best of our knowledge, contemporary researches on the agent-based paradigm fail to take the fault management into consideration.

1.2. Motivation-2: Infeasible individual fault reconnaissance

Fault management in sensor networks can be classified into three phases: fault reconnaissance, fault diagnosis and recovery. Fault reconnaissance is the most fundamental phase for the simple reason that only if faults are identified, previously established mechanisms and algorithms [16–19,37] can be further applied for fault diagnosis and its recovery. Obviously, the trustworthiness and effectiveness of fault reconnaissance, to a large extent, rely on utilization of communication and hardware resources (i.e. bandwidth and energy). Consider the following scenario. Each of sensor nodes attempts to report (either directly or indirectly) its own fault probability to a command center (sink) in a scalable sensor network. This incurs tremendous energy consumption and wastes limited bandwidth resources. According to our literature survey on fault management, even though some researchers presented some efficient optimization methodologies for fault reconnaissance (as mentioned in Section 2), these methodologies still result in extra traffic overhead and energy consumption.

To achieve effective, efficient fault reconnaissance with “zero” extra traffic overhead as well as energy expenditure, we introduce ATLAS – an intelligent agent for fault reconnaissance in sensor networks. The core of ATLAS, inference engine, is abstracted as a mathematical model based on Expectation Maximization (EM) algorithm [20]. The mathematical model of our proposed agent’s problem solver is based on two assumptions. (1) Data gathering mechanism utilize reverse multicast tree. (2) The reverse multicast tree is known and is relatively stationary. One of the main contributions of this work is to utilize an agent-based centralized approach [33] to infer and identify failures among sensor nodes. At sink, an agent is loaded with fault inference engine that profits from an expectation-maximization algorithm to identify faults at the cost of reduced energy expenditure and bandwidth saturation. Experimental evaluation of our ATLAS prototype in a wireless sensor testbed, developed at University of Colorado, Boulder, shows that ATLAS is effective at inferring faults.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 provides the basic mathematical model. Section 4 introduces an intelligent agent – ATLAS. Section 5 describes and discusses experimental results. Finally, Section 6 provides conclusions and future research.

2. Related work

Fault reconnaissance is the first phase of fault management, where an unexpected failure should be properly identified within a network. The fault reconnaissance approaches in WSNs are generally classified into two types: centralized and distributed approach.

The centralized approach is the most common approach to identify the cause of failures or suspicious nodes in WSNs. Usually, a centralized sensor node equipped with abundant power supply (base station manager [22,23], central controller [24] or sink [25]) that monitors [failed or misbehaved nodes in a network. Centralized node banked with abundant power supply is able to execute a wide range of fault management maintenance.

In distributed approach, a node is allowed to make certain level of decisions before communicating with a central node. This is due to the fact that the more decisions a sensor node can make, the less information needs to be delivered to the central node. This reduces the unnecessary bandwidth utilization and energy expenditure. This means that the central node should not be informed unless there is really a fault occurred in the network. Examples of such approach are: node fault self-reconnaissance on its hardware (including physical malfunction, i.e. sensor, battery, RF transceiver) [26], failure detection via neighbour co-ordination [19], and utilization of Watchdog to detect misbehaving neighbour [27].

Several attempts have been made by many researchers to find effective centralized approaches that can achieve the fault management in WSNs with the minimal exploitation of limited bandwidth and energy. Specifically, Sympathy [25] used a message-flooding approach to pool event data and current states (metrics) from sensor nodes. In order to minimize the number of communication messages that nodes must send as well as to conserve node energy. A Sympathy node can selectively transmit important events to the Sympathy sink node. As a complementary of Sympathy, Staddon et al. [23], to minimize extra traffic overhead further, seek the solution of appending network topology information (i.e. node neighbour list) into node’s routing update messages rather than in a separate approach. Based on this approach, the base station can construct the entire network topology by integrating each portion of the network topology information embedded in route update messages and identify the communication faults within sensor nodes. In addition, some common routing protocols (e.g., SPINs [22]) can also detect failed or misbehaving nodes through routing discovery and update. However, all of these approaches typically require the nodes to send additional messages, which is relatively expensive.

Recently, agent-based approaches have been proposed for efficient data dissemination in WSNs. Furthermore, most of researchers attempted to harness a reverse multicast tree paradigm to achieve their agent-based approaches [3,5,7]. However, none of them take into account the efficient fault reconnaissance due to the resource constraints in WSNs. Based on the reverse multicast tree paradigm, some other researchers proposed a couple of efficient inference schemes [12,13], which to some extent, provided the foundation of fault reconnaissance. For example, the work proposed by Caceres et al. [13], a maximum-likelihood estimator for fault inference is developed. This mechanism is based on link losses observed by multicast receivers and exploits the inherent correlation between such observations to infer the performance of paths. Many researchers attempted to consider fault-reconnaissance inference using unicast measurements [10,11]. The end-to-end measurement scheme proposed by Coates et al. [10] is a case in point. It provides a statistical model and computation framework for network loss inference. However, both types of inference mechanisms (unicast and multicast inference schemes) are based on the traditional IP-based networks. In order to overcome the limitation of the traditional inference schemes, Hartel et al. [14] attempted to address such a drawback in WSNs. Theoretically their approach may lead into converge to local maxima; furthermore, it also requires sufficient observation data (over 300 observation data) to make approximately accurate fault inference. Therefore, while faults occur in a sensor node arbitrarily, the approach proposed in [14] may not be able to accurately infer the fault probability of a sensor node. All of the aforementioned mechanisms focused on the link measurements rather than on the sensor nodes. To address the aforementioned limitations, we propose an intelligent agent – ATLAS to perform fault detection inference in WSNs.

3. Basic mathematical algorithm

Expectation-maximization algorithm for finding maximum likelihood estimates of parameters is proposed by Dempster et al. [20]. This algorithm is applied for computing maximum likelihood estimates from incomplete data. The term “incomplete data” in its general form implies the existence of two sample spaces Φ and Ψ, and many-to-one mapping from Φ to Ψ, as shown in Fig. 1. The observed data y is a realization from Ψ, while the corresponding x in Φ is observed indirectly through y only. More specifically, we assume there is a mapping x $\rightarrow$ y(x) from Φ to Ψ, and x is known only to lie in Φ(y), the subset of Φ determined by the equation y = y(x). Where, y is the observed data and x is referred to it as complete data.

Consider first a family of sampling densities $f\,(x|\alpha)$ depending on parameter α (natural parameters) and derive its corresponding family of sampling densities $g\,(y|\alpha)$. The complete data specification $f\,(...|...)$ is related to the incomplete data specification $g\,(\ldots|\ldots)$, which can be described as:

$$g\,(y|\alpha) = \int_{\Phi(y)} f\,(x|\alpha)\,dx. \tag{1}$$

The EM algorithm is directed at finding a value of α which maximizes $g\,(y|\alpha)$ given an observed y, but it does so making essential use of the associated family $f\,(x|\alpha)$. The EM algorithm is briefly described in the following section, using an application case.

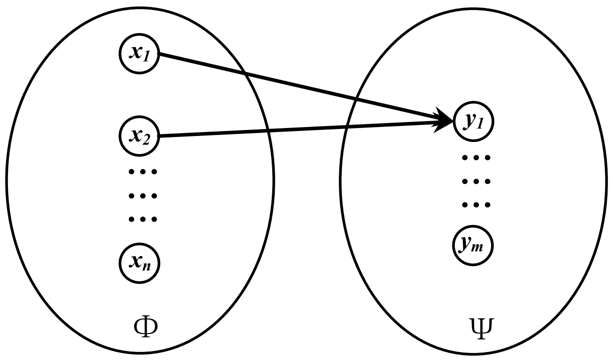

Fig. 1. Many-to-one mapping from the set Φ to set Ψ.

3.1. Application case in glance

In order to give a distinct explanation concerning the EM algorithm, we borrowed a classic application case from [34]. Suppose that in an image pattern-recognition problem, there are two general classes to be distinguished: a class of dark objects and a class of light objects. The class of dark objects may be further subdivided into two shapes: round and square. Using a pattern recognizer, it is desired to determine the probability of a dark object. In this case, let's assume that the objects are known to be trinomially distributed. Let the random variable $\boldsymbol{X}_1$ represent the number of round dark objects, $\boldsymbol{X}_2$ represent the number of square dark objects, and $\boldsymbol{X}_3$ represent the number of light objects. Also assume that we have enough information about the probabilities of the different classes. Therefore, the probability may be written as:

$$\begin{aligned} &P\left(X_1 = x_1, X_2 = x_2, X_3 = x_3 | p\right) \\ &= \left(\frac{n!}{x_1! x_2! x_3!}\right) \left(\frac{1}{4}\right)^{x_1} \left(\frac{1}{4} + \frac{p}{4}\right)^{x_2} \left(\frac{1}{2} - \frac{p}{4}\right)^{x_3} \\ &= f\left(x_1, x_2, x_3 | p\right) \ . \end{aligned} \tag{2}$$

Where, p is the unknown parameter of the distribution and $n = x_1 + x_2 + x_3$. The notation $f(x_1, x_2, x_3$ |p) is used to indicate the probability function which may be either a probability density function (pdf) or a probability mass function (pmf).

The pattern recognizer can distinguish between different classes of objects whether they are dark or light, but cannot distinguish between the shapes. Let $[y_1, y_2]^T = \boldsymbol{y}$ be the number of dark objects and number of light objects detected, respectively. Thus, $y_1 = x_1 +$ x_2 and $y_2 = x_3$. Let's also assume that Y_1, and Y_2 are the corresponding random variables. There is a many-to-one mapping between $\{x_1, x_2\}$ and $\{y_l\}$. For example, if $y_1 =$ 3, there is no way to tell from the measurements whether $x_1 =$ 1 and $x_2 =$ 2 or $x_1 =$ 2 and $x_2 =$ 1. Therefore, the EM algorithm is designed for addressing such problem with many-to-one mappings using the following probability function.

$$\begin{aligned} P\left(Y_1 = y_1 | p\right) &= \binom{n}{y_1} \left(\frac{1}{2} + \frac{p}{4}\right)^{y_1} \left(\frac{1}{2} - \frac{p}{4}\right)^{n-y_1} \\ &= g\left(y_1 | p\right) \ . \end{aligned} \tag{3}$$

The symbol g is used to indicate the probability function for the observed data. From the observation of y_1 and y_2, we can compute the maximum likelihood estimate of p, using the following equation:

$$p_{ML} = \arg \max_p g\left(Y_1 = y_1 | p\right). \tag{4}$$

Where "arg max" means the value that maximizes the function. In this example, it would be a simple matter to determine a maximum likelihood estimate of p. In more interesting problems, however, such straightforward estimation is not possible. In the interest of introducing the EM algorithm, we will not take the direct approach to the maximum likelihood estimate. Taking the logarithm of the likelihood often simplifies the maximization and yields equivalent results since log is an increasing function, so Eq. (4) can be rewritten as:

$$p_{ML} = \arg \max_p \log \binom{n}{y_1} \left(\frac{1}{2} + \frac{p}{4}\right)^{y_1} \left(\frac{1}{2} - \frac{p}{4}\right)^{n-y_1} . \tag{5}$$

Based on EM algorithm, even though we do not know x_1 and x_2, the knowledge of the underlying distribution f (x_1, x_2, $x_3|$ p) can be used to determine an estimate for p. This is done by first estimating the underlying data, then using these data to update current estimate of the parameter. This is repeated until convergence. Let $p^{[k]}$ indicate the estimate of p after the k^{th} iteration (k = 1, 2, 3, ...). An initial parameter value $p^{[0]}$ is assumed. The algorithm consists of two major steps, namely: expectation step and maximization step.

Expectation Step (E-step). In this step, the expected value of x data is computed using the current estimate of the parameter and the observed data. In the current example, the expected value of x_l, given the measurement y_1 and based upon the current estimate of the parameter, may be computed as follows:

$$x_1^{[k+1]} = \boldsymbol{E}[x_1| \quad y_1, p^{[k]}].$$

$$x_1^{[k+1]} = y_1 \frac{\frac{1}{4}}{\frac{1}{2} + \frac{p^{[k]}}{2}} .$$

Similarly for x_2 and x_3,

$$\begin{aligned} x_2^{[k+1]} &= E\left[x_1 | y_1, p^{[k]}\right] = y_1 \frac{\frac{1}{4} + \frac{p^{[k]}}{2}}{\frac{1}{2} + \frac{p^{[k]}}{2}} . \\ x_3 &= y_3. \end{aligned} \tag{6}$$

Maximization Step (M-step). In this step, a maximum likelihood estimate of the parameter is determined using the data from the expectation step as if it were actually measured data.

In this example, with $x_l^{[k+l]}$, $x_2^{[k+1]}$ and $\boldsymbol{x}_3$ available, the maximum likelihood estimate of the parameter is obtained by taking the derivative of *log* f ($\boldsymbol{x}_l^{[k+l]}$, $\boldsymbol{x}_2^{[k+1]}$, $\boldsymbol{x}_3|p$) with respect to p, set it to zero, and solving for p as follows:

$$\begin{aligned} 0 &= \frac{d}{dp} \log f\left(x_1^{[k+1]}, x_2^{[k+1]}, x_3 | p\right) \\ \Rightarrow p^{[k+1]} &= \frac{2x_2^{[k+1]} - x_3}{x_2^{[k+1]} + x_3} . \end{aligned} \tag{7}$$

Table 1
Results of the Em Algorithm for pattern recognition problem

k	$x_1^{[k]}$	$x_2^{[k]}$	$p^{[k]}$
1	31.500000	31.500000	0.379562
2	26.475460	36.524540	0.490300
3	25.298157	37.701843	0.514093
4	25.058740	37.941260	0.518840
5	25.011514	37.941260	0.519773
6	25.002255	37.997745	0.519956
7	25.000441	37.999559	0.519991
8	25.000086	37.999914	0.519998
9	25.000017	37.999983	0.520000
10	25.000003	37.999997	0.520000

Now, we need to compute the estimate of $\boldsymbol{x}_1^{[k+1]}$. The estimate $\boldsymbol{x}_1^{[k+1]}$ is not used in Eq. (7). Thus, there is no need to compute it, for this example. The EM algorithm consists of E-step and M-step until convergence. Intermediate computation and storage may be eliminated by combining E-step equation and M-step one to obtain a one-step update.

Using a numerical example, suppose that the true parameter is $p = 0.5$, $n = 100$ samples, and $y_1 = 100$. The true values respectively of x_1 and x_2 are 25 and 38, but this is unknown to the algorithm. Table 1 illustrates the results of the computed algorithm, starting from $p^{[0]} = 0$ which is similar to [20].

4. ATLAS

This section first provides an intelligent agent-ATLAS- with failure Inference capacity. Furthermore, we provide terminology and a fundamental mathematical model used in the inference engine. Finally, we propose the core of ATLAS – a failure inference engine which is formulated as a maximum-likelihood estimate problem.

4.1. An Intelligent agent with fault inference

Most of agent-based distributed systems consist of two types of agents based on their functionalities, i.e. stationary and mobile agents. In this paper, ATLAS is a stationary agent with an embedded fault inference engine. The architecture of ATLAS (shown in Fig. 1) with fault-reconnaissance functionality is based on the agent model described in [1]. Generally, ATLAS resides on a central node equipped with surplus power supply (Base station).

As demonstrated in Fig. 2, the architecture of ATLAS consists of knowledge base and executable components. The knowledge base contains the information about the WSNs environment such as packet-store table and topology structure. The packet-store table is composed of the received packets from target sensor nodes; ATLAS maintains the topology structure of reverse multicast trees as well. For some applications in WSNs, an application may generate several reverse multicast trees to achieve a better quality of service. The learning component provides ATLAS with the capability of monitoring observation data stored in its knowledge base. Based on the observations, ATLAS collaborates with various routing protocols and thus dynamically selects a routing protocol that provides the best performance benefits for WSNs. The scheduler component provides ATLAS with a time agenda to start or stop certain activities such as monitoring and aggregating data. In this architecture, the problem solver

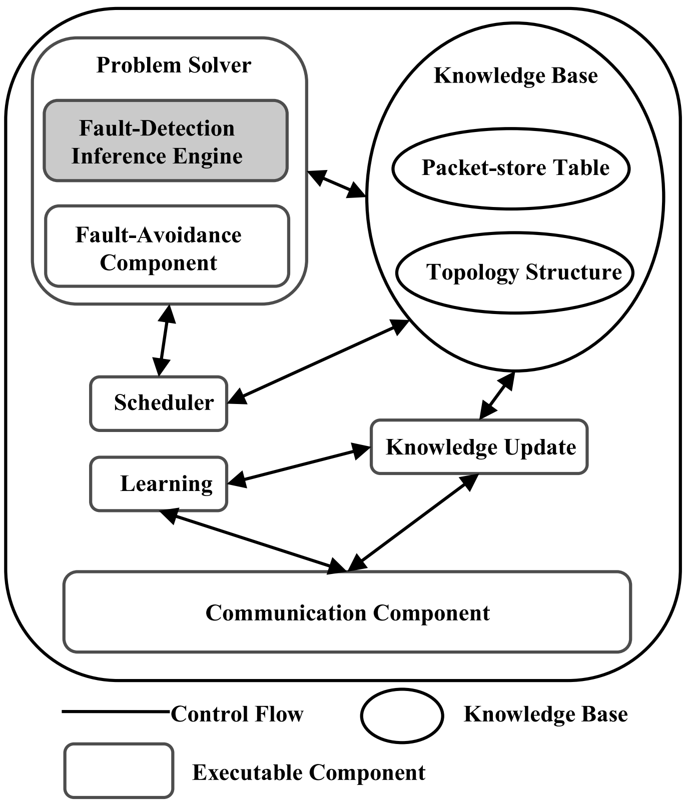

Fig. 2. The architecture of ATLAS.

is an intelligent component of ATLAS. It includes failure-reconnaissance inference engine (which is the core of ATLAS) and fault-avoidance component. Fault-reconnaissance inference engine harnesses the information in knowledge base to infer the working states of sensor nodes in a reverse multicast tree. And then, the inferred consequences assist fault-avoidance component in choosing and switching to the appropriate routing strategy among various routing protocols.

4.2. Terminology and mathematical model

Failure inference engine adopts a reverse-multicast-tree mechanism. This scheme is based on the fundamental of data aggregation. In the inference engine, we formulate a reverse multicast tree as a mathematical model $T := (V, E)$, where T is the reverse multicast tree that is subject to the following conditions:

V is a set, whose elements are called wireless sensor nodes; E is a set of pairs of distinct sensor nodes called communication link. Obviously, the node set $V = \{n_1, n_2, n_3, \ldots, S\}$ also contains sink S. And a communication link from node i to node j is represented by $e_{i,j} \in V \times V$.

For our modeling purposes in this paper, we make a simplifying assumption – the communication link between node i and j is always symmetrically bidirectional link such that $e_{i,j} \equiv e_{j,i}$. Further, let $c(n_i)$ and $d(n_i)$ denote the child set and the descendent set of node i, respectively. In reverse multicast tree paradigm, each node except sink has a unique parent node denoted by $p(n_i)$ where $n_i \in V \wedge n_i \notin \{S\}$. Finally, denote the set of leaf nodes in the tree by $l(T)$. That is $l\,(T) = \{n_k \in V \wedge c\,(n_k) = \phi\}$.

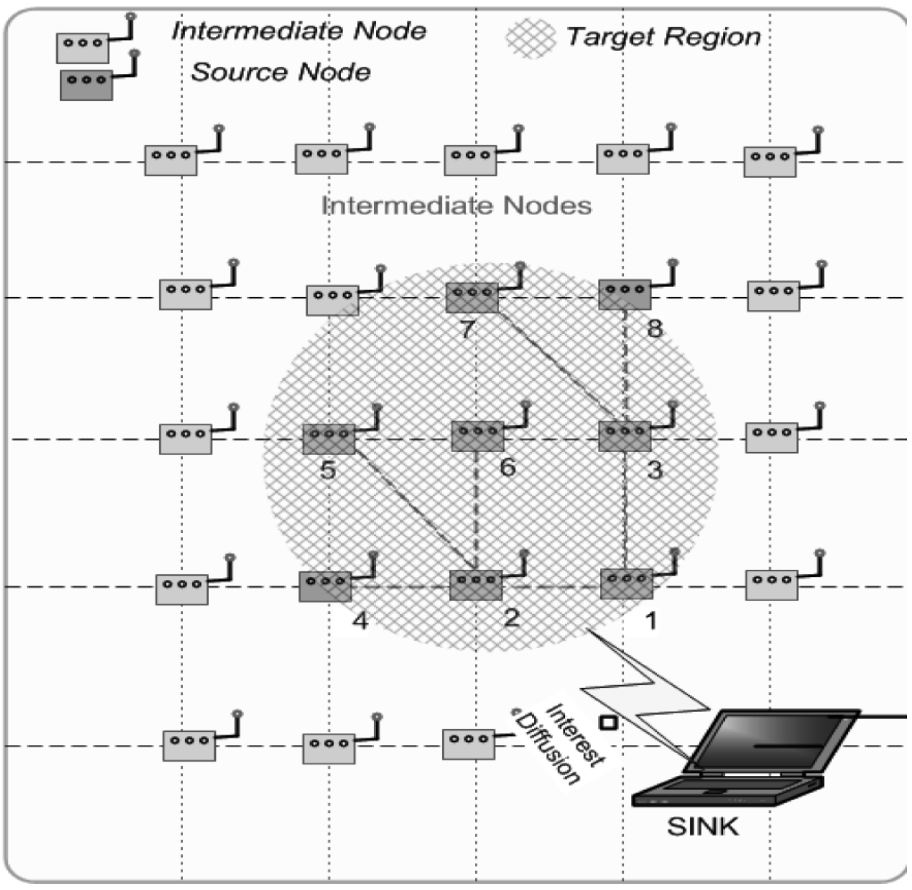

Fig. 3. Eight-node reserve multicast tree in a square grid topology.

For example, in the tree depicted in Fig. 3, for node 2 the proper descendant nodes are 4, 5 and 6 and the parent node is node 1 i.e., $d(n_2) = \{4, 5, 6\}$ and $p(n_2) = \{1\}$ respectively. In this whole tree, the leaf nodes are 4, 5, 6, 7, 8 denoted as $l(T) = \{4, 5, 6, 7, 8\}$. In wireless sensor networks, sensor nodes are prone to failures [15]. Therefore, each sensor node is modeled by an independent Bernoulli process [14].

Associated with every node $n_i \in V$, a success probability α_{n_i} represents the packets sent from node j ($\forall j\ n_j \in c\,(n_i)$) to node i that are received successfully. On the contrary, the fault probability of node i denotes $\beta_{n_i}(\alpha_{n_i} + \beta_{n_i} = 1)$. Finally, let $\alpha = \langle \alpha_{n_1}, \alpha_{n_2}, \alpha_{n_3}, ...\rangle$ denote the set of success probabilities for all sensor nodes. The packets transmitted through a reverse multicast tree, and the packets received by sink construct a sample space $\Omega = \left\langle x^1, x^2, x^3, ..., x^m \right\rangle$.

A possible outcome x^i in data collection round i is composed of a sequence of partial nodes in reverse multicast tree. Based on the reverse-multicast-tree paradigm, each node aggregates its data with that of its descendants. Therefore, if the sink does not receive data from a node, it also does not gather data from all the descendents of that node. For example in Fig. 3, the sink receives an aggregated packet x^i with the sequence of nodes <1, 2, 3, 5, 6, 7>. Furthermore, each round of data collection is considered a random trial. Since the outcome of each random trial denotes a record of which nodes the sink received data from that round, we consider a discrete random variable X. Let $p\,(X = x;\ \alpha) = p\,(x;\ \alpha)$ denotes the probability distribution for a given set of node success probabilities α. Also, assume an m-round trial, the probability of observing x^1, ..., x^m in m data collection rounds is denoted by:

$$p\left(x^1,\, ..., x^m\right) = \prod_{i=1}^{m} p\left(x^i; \alpha\right). \tag{8}$$

4.3. An intelligent agent with fault inference

Failure inference engine adopts a reverse-multicast-tree mechanism. From previous section, $p\,(x;\ \alpha)$ is the probability mass function of a single outcome x. Nevertheless, the success probability α is an unknown parameter and a quantity we expect to estimate. So, our proposed inference engine utilizes maximum likelihood estimation to infer parameter of underlying probability distribution from a collected packet set. Assume that N represents packet collection rounds, the packet sequence x^1, ..., x^N observed at the sink comprises a random sample space (due to malfunction of WSNs, x^i may be an empty node sequence). Based on the sample space of N values, we choose $\widehat{\alpha}$ to maximize $p\left(x^1,\ ...,\ x^N; \widehat{\alpha}\right)$, i.e., $\widehat{\alpha} = \arg\ \max p\left(x^1, ..., x^N; \alpha\right)$ which is maximum likelihood estimation.

In our approach however, we require a sophisticated estimation technique due to the fact that the current knowledge base in inference engine (sample space) is incomplete. Therefore, as an efficient iterative computing approach, the expectation maximization algorithm is incorporated in the fault-detection inference engine. This algorithm makes inference about parameter α using incomplete knowledge base. It should be noted that "incomplete knowledge base" is equivalent to "complete observed data" in expectation maximization algorithm.

It is more convenient to operate the log-likelihood function of the complete data. Let $\{x^1, x^2, \ldots,$ $x^N\}$ be the set of complete data. Then, log-likelihood equation can be described as follows:

$$L\left(x^1, x^2, ..., x^N; \alpha\right) = \log\ p\left(x^1, x^2, ..., x^N; \alpha\right)\ . \tag{9}$$

Where, the joint distribution probability is divided into two parts: leaf and non-leaf nodes. The expression for leaf nodes is:

$$p\left(x^1, x^2, ..., x^N; \alpha\right) = \prod_{i=1}^{N} \alpha_{n_i}^{N_{n_i \to n_i}} \cdot \beta_{n_i}^{N - N_{n_i \to n_i}}\ . \tag{10}$$

Where $N_{n_i \to n_i}$ denotes the number of data collection rounds that node i successfully sent out packets. For the non-leaf nodes, it is described as:

$$p\left(x^1, x^2, ..., x^N; \alpha\right) = \prod_{i=1}^{N} \alpha_{n_i}^{N_{d(n_i) \to n_i} + N_{n_i \to n_i}} \cdot \beta_{n_i}^{N + N_{c(n_i) \to n_i} - N_{d(n_i) \to n_i}}. \tag{11}$$

Where $N_{d(n_i) \to n_i}$ denotes the number of data collection rounds that node i successfully received packets from its descendents, and $N_{c(n_i) \to n_i}$ represents the number of data round that node i successfully received child nodes' packets.

To maximize the Eqs (10) and (11), we take the logarithm function to simplify maximization likelihood estimation. The results are shown in Eqs (12) and (13).

$$L\left(x^1, x^2, ..., x^N; \alpha\right) = \sum_{n_i \in l(T)} \left(N_{n_i \to n_i} \log \alpha_{n_i} + N \log \beta_{n_i} - N_{n_i \to n_i} \log \beta_{n_i}\right). \tag{12}$$

$$L\left(x^1,x^2,...,x^N;\alpha\right)=\sum_{n_i\notin l(T)}((N_{d(n_i)\to n_i+N_{n_i\to n_i}})\log\alpha_{n_i}+(N+N_{c(n_i)\to n_i}-N_{d(n_i)\to n_i}) \log\beta_{n_i}) . \tag{13}$$

For maximizing Eqs (12) and (13), we utilize Calculus Theory that leads to Eq. (14).

$$\frac{d}{d\alpha}L\left(x^1,x^2,...,x^N;\alpha\right)=0. \tag{14}$$

By combining Eqs (12), (13) and (14), we solve the equations and produce the parameter Eqs (15) and (16) as we expected.

$$\alpha_i=(N_{n_i\to n_i}/N)\cdot 100\% \quad n_i\in l\left(T\right) . \tag{15}$$

$$\alpha_i=\frac{\sum\left(N_{d(n_i)\to n_i}+N_{n_i\to n_i}\right)}{\sum\left(N+N_{c(n_i)\to n_i}\right)}\cdot 100\% \quad n_i\notin l\left(T\right) . \tag{16}$$

For example, in the square grid topology depicted in Fig. 4, the success probability of node 3 is formulated as follows:

$$\alpha_3=\frac{(N_{n3\to n3}+N_{n5\to n3}+N_{n6\to n3}+N_{n4\to n3})}{(N+N_{n6\to n6}+N_{n5\to n5}+N_{n4\to n4})}$$

However, a complete knowledge base requires each sensor node to record and transmit packets that it received from descendents in each round. Unfortunately, this solution not only wastes constrained bandwidth but also consumes energy resources. Therefore, an efficient inference engine, which learns from the collected packets at sink, is utilized to complete knowledge base.

To complete the knowledge base, we adopt expectation maximization algorithm, using Eqs (8) and (9). This algorithm, in this case, consists of five steps. In step 1, initializing fault-detection parameter vector $\widehat{\alpha}^{[0]}$ and $\widehat{\beta}^{[0]}$. In step 2, the conditional expectation is used under the parameter vector $\widehat{\alpha}^{[k]}$ and $\widehat{\beta}^{[k]}$ to estimate incomplete data set $N^{[k+1]}_{d(n_i)\to n_i}$, $N^{[k+1]}_{c(n_i)\to n_i}$ and $N^{[k+1]}_{n_i\to n_i}$. In step 3, maximizing the expected log-likelihood and update the parameter vectors $\widehat{\alpha}^{[k+1]}$ and $\widehat{\beta}^{[k+1]}$. In step 4, repeating steps 2 and 3 until the algorithm converges to local maxima. In step 5, restarting with new initial parameter vector $\widehat{\alpha}^{[0]}$ and $\widehat{\beta}^{[0]}$ since the likelihood surface may not guarantee convex. If the likelihood is not convex, then the probability may converge to local maxima. The entire internal inference process described as a Non-deterministic Finite Accepter (NFA) as shown in Error! Reference source not found..

The iterative steps 2 and 3 require computing $N^{[k+1]}_{d(n_i)\to n_i}$, $N^{[k+1]}_{c(n_i)\to n_i}$ and $N^{[k+1]}_{n_i\to n_i}$from the knowledge base, i.e., completely observed data set $\{x^1,x^2,...,\mathrm{x}^N\}$. The computation process is based on Chapman-Kolmogorov functional equation. The corresponding Markov chain model is shown in 5.

Therefore, the Chapman-Kolmogorov equation based on the matrix of transition probabilities can be represented by Eqs (17) and (18).

$$\begin{bmatrix}\alpha_{n_i}\,\beta_{n_i}\\ \beta_{n_i}\,\alpha_{n_i}\end{bmatrix}\cdot\begin{bmatrix}\alpha_{n_j}\,\beta_{n_j}\\ \beta_{n_j}\,\alpha_{n_j}\end{bmatrix}\cdot\ldots\cdot\begin{bmatrix}\alpha_{n_m}\,\beta_{n_m}\\ \beta_{n_m}\,\alpha_{n_m}\end{bmatrix}=\begin{bmatrix}\alpha\,\beta\\ \beta\,\alpha\end{bmatrix} \tag{17}$$

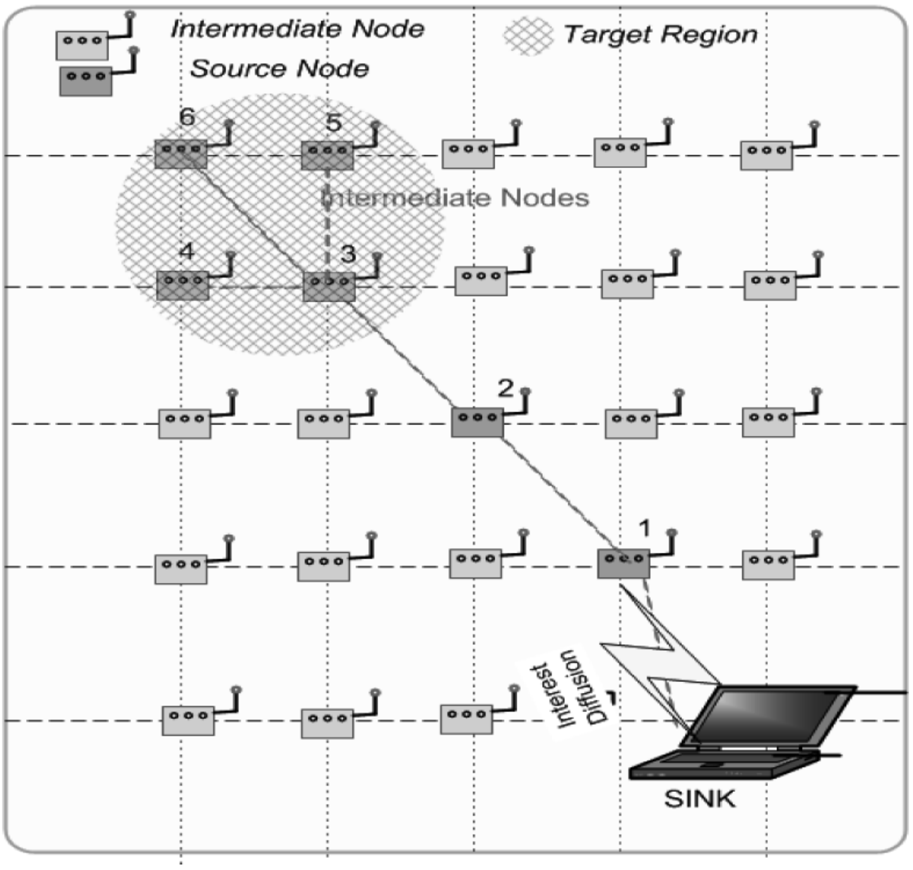

Fig. 4. Six-node reserve multicast tree in a square grid topology.

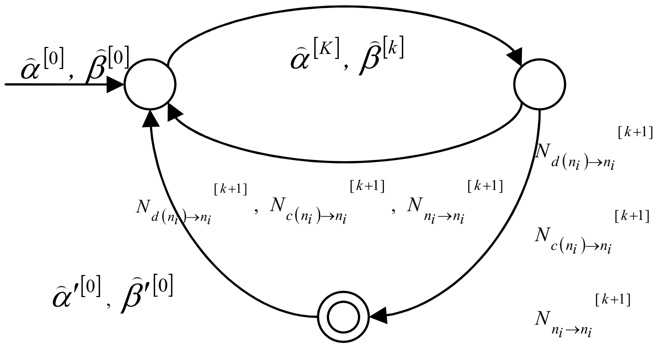
$\hat{\alpha}^{[0]}, \hat{\beta}^{[0]}$
$\hat{\alpha}^{[K]}, \hat{\beta}^{[k]}$
$N_{d(n_i)\to n_i}^{[k+1]}$, $N_{c(n_i)\to n_i}^{[k+1]}$, $N_{n_i\to n_i}^{[k+1]}$
$\hat{\alpha}'^{[0]}, \hat{\beta}'^{[0]}$
$N_{d(n_i)\to n_i}^{[k+1]}$
$N_{c(n_i)\to n_i}^{[k+1]}$
$N_{n_i\to n_i}^{[k+1]}$

Fig. 5. Fault-detection inference engine described in NFA.

For leaf nodes,

$$N_{n_j\to n_i}^{[k+1]} = N_{n_j} + \left(N - N_{n_j}\right)\cdot \beta_{n_i}^{[k]}\,, \quad n_j \equiv n_i. \tag{18}$$

Where, N_{n_j} denotes the number of packet collection rounds with node identification j observed by sink.

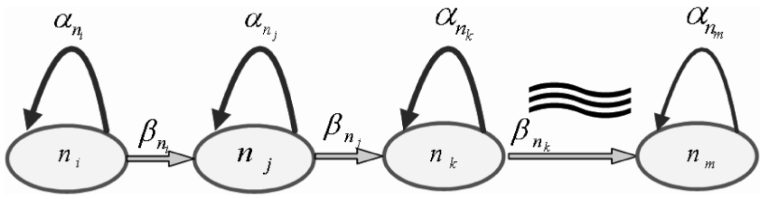

Fig. 6. Morkov chain model.

Fig. 7. A snapshot of Testbed at University of Colorado.

Similarly, this can be used to represent non-leaf nodes, using Eq. (19).

$$N_{n_j \to n_i}^{[k+1]} \\ = N_{n_j} + \left(N - N_{n_j}\right) \cdot \beta \quad n_j \in c\left(n_i\right) \vee n_j \in d\left(n_i\right). \tag{19}$$

Combining E-Step Eqs (15) and (16) with M-Step Eqs (18) and (19), we iteratively compute a set of success probability values. Next section will show a 6-nodes reverse cast tree experiment in detail and analysis of our experiment results.

5. Experimental results

This section discusses our experimental results and analyzes the properties of inference engine in ATLAS. In order to validate the correctness and effectiveness of ATLAS, we implement the prototype of ATLAS on top of the Mantis Operating System (MOS). MOS is an open source, multithreaded operating system developed at University of Colorado at Boulder for use on wireless sensor networking platforms.

For our experiments, we deployed an indoor testbed of TelosB motes as shown in Fig. 7. The TelosB platform was developed at University of California at Berkeley and is marketed and sold by Moteiv and Crossbow. The radio used by the TelosB is the chipcon CC2420, which is an 802.15.4 compliant device, has a data rate of 250Kbps, and operates in the 2.4 GHz ISM band. The mote uses an 8 MHz TI MSP430 processor and has 1 MB of external flash.

The entire experimental scenario we adopted consists of 30 sensor nodes. In the 30-node topology, 7 nodes generated a reverse multicast tree. During each collection round, the sink cached received packets

Table 2
Results of the iterative computation based on equal fault probability

k	$n_1^{[k]}$	$n_2^{[k]}$	$n_3^{[k]}$
1	98.1507%	96.8647%	98.1432%
2	98.8739%	97.6735%	96.3148%
3	98.8795%	97.5608%	96.4541%
4	98.8774%	97.5733%	96.4631%
5	98.8776%	97.5728%	96.4570%
6	98.8776%	97.5727%	96.4587%
7	98.8776%	97.5727%	96.4583%
8	98.8776%	97.5727%	96.4584%
k	$n_4^{[k]}$	$n_5^{[k]}$	$n_6^{[k]}$
1	92.5000%	91.6250%	92.8750%
2	86.1250%	84.6528%	86.7653%
3	87.0812%	85.8206%	87.6359%
4	86.9378%	85.6250%	87.5118%
5	86.9593%	85.6578%	87.5295%
6	86.9561%	85.6523%	87.5270%
7	86.9565%	85.6532%	87.5273%
8	86.9565%	85.6530%	87.5273%

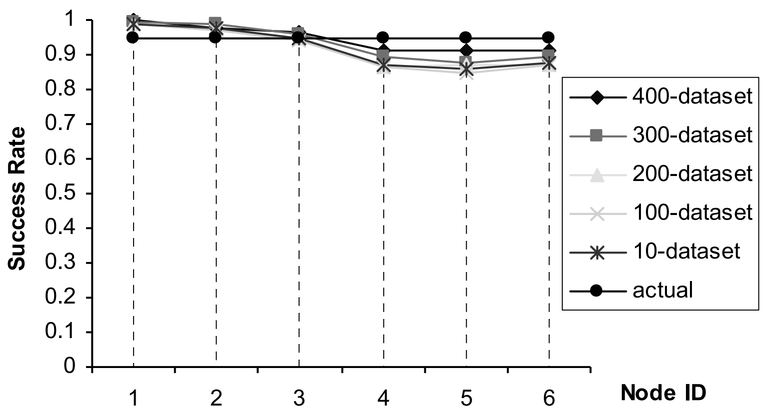

Fig. 8. Inference engine under scenario 1.

from its descendents. After 400 data collection round, the fault-inference engine was triggered at sink. For comparability, the inference engine analyzed observed dataset based on various sub-datasets.

The first experiment is conducted on an equal fault probability (Scenario 1). In this scenario, we assumed the normal success probability of sensor node to be 95%. The 5% of failure probability could be due to signal loss, interference, radio initialization, etc. Thus, the initial parameter $\widehat{\beta}^{[0]}$ is chosen as a set of different values in the interval [0,1] to avoid halting on local maxima. Table 2 illustrates the detailed results based on the iterative computation for equal fault probability scenario.

Furthermore, the failure probabilities as well as the absolute error for each node are reflected in Figs 8 and 9 respectively.

In Fig. 8, it was observed that inferred fault probabilities of the nodes are increasingly accurate along with the increase in the volume of dataset.

Furthermore, it is obvious from the plot of standard deviation shown in Fig. 10 that the inference

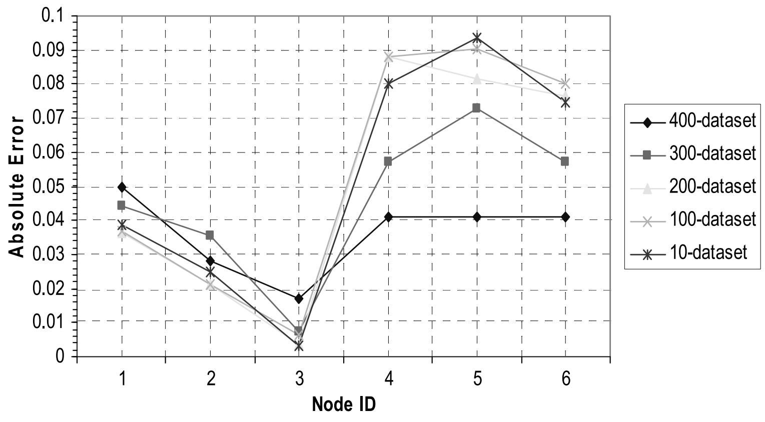

Fig. 9. Absolute error under scenario 1.

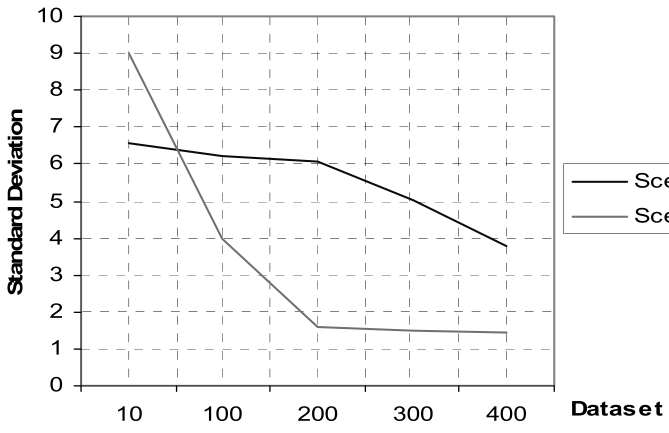

Fig. 10. Standard deviation under disparate scenario.

process is able to accurately reflect the real-time states of nodes.

The second experiment was conducted on a high-frequent-fault node (scenario 2). Faults kept occurring frequently at node 2.

We assume fault probability is set to 40%, while the other nodes ran normally with the same fault probability as that of nodes in first scenario. Based on the experimental results shown in Figs 11 and 12, we may make a reasonable inference that node 2 may experience unexpected high-frequency faults while the rest of nodes 1, 3, 4, 5 and 6 kept running normally.

The results of the reasoning using Eqs (15), (16), (18) and (19) are listed in Table 3.

The volume of dataset in this scenario has great impact on the property of inference engine. More importantly, it can be seen that the standard deviation of the inference engine shown in Fig. 10 drastically decreases along with an increase in dataset that was utilized to infer fault probabilities of nodes. Thus, the proposed inference mechanism is able to successfully target fault-experience nodes.

Furthermore, we study the reliability of maximum-likelihood estimator. Based on the same dataset generated by the scenario 2, the maximum-likelihood estimator performed poor inference result, as shown in Fig. 13.

With the comparison between our proposed mechanism and maximum-likelihood estimator, we observe

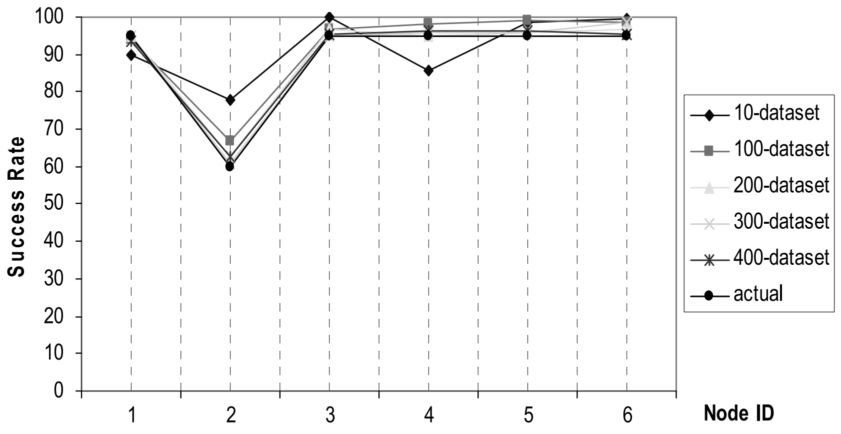

Fig. 11. Inference Engine under scenario 2.

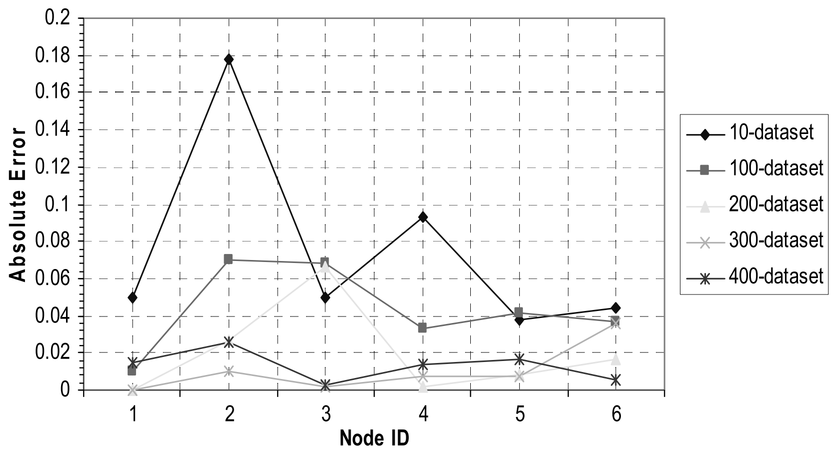

Fig. 12. Absolute Error under scenario 2.

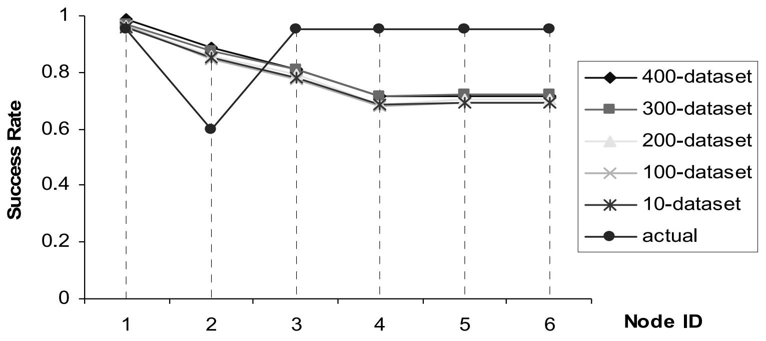

Fig. 13. Maximum-likelihood estimator under scenario 2.

Table 3
Results of the iterative computation based on high-frequent-fault node.

k	$n_1^{[k]}$	$n_2^{[k]}$	$n_3^{[k]}$
1	95.3451%	67.6250%	86.4805%
2	96.8658%	62.7234%	94.8478%
3	96.6865%	62.2154%	94.5067%
4	96.5479%	62.4503%	96.9278%
5	96.6532%	62.3417%	94.2011%
6	96.5946%	62.3919%	94.5636%
7	96.6240%	62.3687%	94.3865%
8	96.6095%	62.3794%	94.4729%
k	$n_4^{[k]}$	$n_5^{[k]}$	$n_6^{[k]}$
1	89.1566%	86.0250%	99.0250%
2	96.9956%	95.7234%	95.4564%
3	96.6219%	96.2154%	95.2655%
4	96.9259%	96.4503%	95.3500%
5	96.2501%	96.3417%	95.3126%
6	96.6086%	96.3919%	95.3291%
7	96.9208%	96.3687%	95.3218%
8	96.7695%	96.3794%	95.3250%

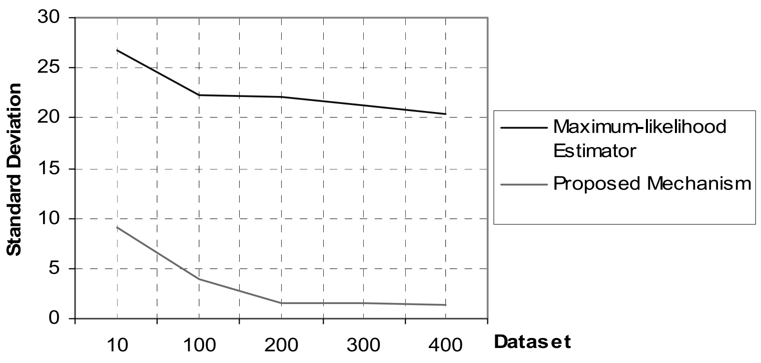

Fig. 14. Standard deviation under disparate inference mechanisms.

from Fig. 14 that our proposed mechanism with the lower standard deviation would satisfy the objective of inference detection. In contrast, maximum-likelihood estimator would not be utilized in WSNs environment.

6. Conclusions and future work

This paper identified the principal faults experienced by WSNs and summarized the existed fault inference mechanisms. This paper presented an intelligent agent – ATLAS with a fault inference problem solving engine as a response to the drawbacks of current inference mechanisms and the features of WSNs. The inference engine is modeled by maximization-likelihood estimation via unobservable data set. In accordance with theoretical computation, the proposed inference engine embedded in ATLAS is abstracted as a nondeterministic finite accepter. This paper provided experimental results that showed the inference engine of ATLAS achieves substantial energy gain by containing the impact of malfunctioning sensor nodes.

In the future, we plan to study and establish the relationship between ATLAS and energy consumption of WSNs. Moreover, we plan to augment our effort to expand our approach to a higher level of fault management, i.e. involving fault diagnosis with minimum exploitation of WSNs resources.

References

[1] E. Shakshuki, H. Ghenniwa and M. Kamel, Agent-based system architecture for dynamic and open environments, *International Journal of Information Technology and Decision Making* **2**(1) (2002), 105–133.

[2] I. Stojmenovic, Localized network layer protocols in sensor networks based on optimizing cost over progress ratio, *IEEE Network* **20**(1) (2006), 21–27.

[3] M. Chen, T. Kwon, Y. Yuan, Y. Choi and V.C.M. Leung, Mobile Agent-Based Directed Diffusion in wireless Sensor Networks, EURASIP Journal on Advances in Signal Processing, Article ID 36871, 2007.

[4] E. Shakshuki, H. Malik and T. Sheltami, Multi-agent Based Clustering Approach to Wireless Sensor Networks, *International Journal of Wireless and Mobile Computing* (*IJWMC*), *Inderscience Publisher* **3**(3) (2009), 165–176.

[5] E. Shakshuki, X. Xing and H. Malik, Mobile Agent for Efficient Routing Among Source Nodes in Wireless Sensor Networks, The 3rd International Conference on Autonomic and Autonomous Systems (ICAS), IEEE Computer Society, June 19–25, Athens, Greece, 2007.

[6] H. Qi, Y. Xu and X. Wang, Mobile-agent-based Collaborative Signal and Information Processing in Sensor Networks, *Proceeding of the IEEE* **91**(8) (2003), 1172–1183.

[7] Q. Wu, N.S.V. Rao, J. Barhen et al., On computing mobile agent routes for data fusion in distributed sensor networks, *IEEE Transactions on Knowledge and Data Engineering* **16**(6) (2004), 740–753.

[8] C.-L. Fok, G.-C. Roman and C. Lu. Mobile Agent Middleware for Sensor Networks: An Application Case Study, In Proceedings of the 4th International Conference on Information Processing in Sensor Networks (IPSN'05), Los Angeles, California, April 25–27, pp. 382–387, 2005.

[9] D. Massaguer, C.-L. Fok, N. Venkatasubramanian, G.-C. Roman and C. Lu, Exploring sensor networks using mobile agents, AAMAS, 2006, 323–325.

[10] M. Coates and R. Nowak, Network Loss Inference Using Unicast End-to-End Measurement, in ITC Seminar on IP Traffic, Measurement and Modelling, September 2000.

[11] N. Duffield, F.L. Presti, V. Paxson and D. Towsley, Inferring Link Loss Using Striped Unicast Probes, in Proceedings of IEEE INFOCOM 2001.

[12] S. Ratnasamy and S. McCanne, Inference of Multicast Routing Trees and Bottleneck Bandwidths using End-to-End Measurements, in Proceedings IEEE INFOCOM 1999.

[13] R. Caceres, N.G. Duffield, J. Horowitz and D. Towsley, Multicast-based inference of network-internal loss characteristics, *IEEE Trans Inform Theory* **45** (1999), 2462–2480.

[14] G. Hartl and B. Li, Loss Inference in Wireless Sensor Networks based on Data Aggregation, in the Proceedings of the Third IEEE/ACM International Symposium on Information Processing in Sensor Networks (IPSN 2004), pp. 396–404, Berkeley, California, April 26–27, 2004.

[15] F. Koushanfar, M. Potkonjak and A. Sangiovanni-Vincentelli, Fault-Tolerance in Sensor Networks, Book chapter, in: *Handbook of Sensor Networks*, I. Mahgoub and M. Ilyas, eds, CRC press, Section VIII, no. 36, 2004.

[16] C. Hsin and M. Liu, A Distributed Monitoring Mechanism for Wireless Sensor Networks, in the 3rd workshop on Wireless Security, ACM Press, 2002.

[17] C. Hsin and M. Liu, Self-monitoring of Wireless Sensor Networks, *Computer Communications* **29** (2005), 462–478.

[18] M. Ding, D. Chen, K. Xing and X. Cheng, Localized Fault-Tolerant Event Boundary Detection in Sensor Networks, in Processing of INFOCOM 2005.

[19] J. Chen, S. Kher and A. Somani, *Distributed Fault Detection of Wireless Sensor Networks*, in Processing of DIWANS'06, Los Angeles, USA, ACM Press, 2006.

[20] A.P. Dempster, N.M. Laird and D.B. Rubin, Maximum-likelihood from incomplete data via the EM algorithm, *J Roy Stat Soc* **39** (1977), 1–38.

[21] M. Yu, H. Mokhtar and M. Merabti, *A Survey of Network Management Architecture in Wireless Sensor Network*, in Proceedings of the Sixth Annual Post-Graduate Symposium on The Convergence of Telecommunications, Networking and Broadcasting, 2006.

[22] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J.D. Tygar, *SPINS: Security Protocols for Sensor Networks*, in Processing of ACM MobiCom'01 Rome, Italy ACM Press, 2001.

[23] J. Staddon, D. Balfanz and G. Durfee, *Efficient Tracing of Failed Nodes in Sensor Networks*, in Processing of First ACM International Workshop on Wireless Sensor Networks and Applications, Altanta, GA, USA, ACM, 2002.

[24] W.-L. Lee, A. Datta and R. Cardell-Oliver, *WinMS: Wireless Sensor Network-Management System, An Adaptive Policy-Based Management for Wireless Sensor Networks*, UWA, aUSTRALIA, 2006.
[25] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler and D. Estrin, *Sympathy for the Sensor Network Debugger*, In Processing of the 3rd Embedded networked sensor systems, San Diego, USA, ACM Press, 2005.
[26] F. koushanfar, M. Potkonjak and A. Sangiovanni- Vincentelli, Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks, 2000.
[27] S. Marti, T.J. Giuli, K. Lai and M. Baker, *Mitigating Routing Misbehavior in Mobile Ad Hoc Networks*, in Processing of the 6th International Conference on Mobile Computing and Networking, Boston, Massachusetts, USA, ACM, 2000.
[28] RF Monolithics Inc. TR1001 868.35 MHz Hybrid Transceiver Data Sheet, Aug. 2001.
[29] J.-H. Huang, S. Amjad and S. Mishra, *CenWits: A Sensor-Based Loosely Coupled Search and Rescue System using Witnesses*, in Proceedings of the 3rd international conference on Embedded networked sensor systems (Sensys'05), San Diego, USA, ACM, 2005.
[30] C. Hartung, C. Seielstad, S. Holbrook and R. Han, *FireWxNet: A Multi- Tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments*, Fourth International Conference on Mobile Systems, Applications, and Services (MobiSys), 2006, pp. 28–41.
[31] F. Erik, Y. Golen and S. Nirmala, An Evolutionary Approach to Underwater Sensor Deployment, *IJCIS* **2–3** (2009), 184–201.
[32] Gowrishankar and P.S. Satyanarayana, Neural Network Based Traffic Prediction for Wireless Data Networks, *IJCIS* **1–4** (2008), 379–389.
[33] E. Shakshuki, X. Xing and T. Sheltami, *An Intelligent Agent for Fault Reconnaissance in Sensor Networks*, the 11th International Conference on Information Integration and Web-based Applications & Services, pp. 137–144, December 14–16, Kuala Lumpur, Malaysia, 2009.
[34] T.K. Moon, The expectation-maximization algorithm, Signal Processing Magazine, *IEEE* **13**(6) (November 1996), 47–60,
[35] E. Shurkova, R. Ishak, S. Olariu and S. Salleh, Emergent Behavior in Massively-deployed Sensor Networks, *Mobile Information Systems* **4**(4) (2008), 313–331.
[36] S. Mahfoudh and P. Minet, Maximization of Energy Efficiency in Wireless Ad hoc and Sensor Networks with SERENA, *Mobile Information Systems* **5**(1) (2009), 33–52.
[37] S. Kumar and S. Park, Probability Model for Data Redundancy Detection in Sensor Networks, *Mobile Information Systems* **5**(2) (2009), 195–204.

Elhadi M. Shakshuki is a professor at the Jodrey School of Computer Science at Acadia University, Canada. He is the founder and the head of the Cooperative Intelligent Distributed Systems Group at the Computer Science Department, Acadia University. He received the BSc degree in computer engineering in 1984 from El-Fateh University, and the MSc and PhD degrees in systems design engineering respectively in 1994 and 2000, from the University of Waterloo, Canada. Dr. Shakshuki is an Adjunct Professor at Dalhousie University, Canada. He manages several research projects in his research expertise in the area of intelligent agent technology and its applications. He is a member of IEEE, ACM, AAAI and APENS.

Xinyu Xing is a PhD student at the Department of Computer Science, University of Colorado at Boulderat, USA. He received his MSc degree from Jodrey School of Computer Science, Acadia University, Canada in 2008. His research interests are in the areas of multi-agent systems and wireless sensor network.

Tarek R. Sheltami is currently associate professor at the Computer Engineering Department at King Fahd University of Petroleum and Minerals (KFUPM) Dhahran, Kingdom of Saudi Arabia. He joined the department on September, 2004. Before joining the KFUPM, Dr. Sheltami was a research associate professor at the School of Information Technology and Engineering (SITE), University of Ottawa, Ontario, Canada. He has two years of industrial experience at GamaEng Inc (2002–2004). Dr. Sheltami has been a member of a technical program and organizing committees of several international IEEE/ACM conferences. Dr. Sheltamifls research interests are in the area of wireless communications, wireless ad hoc and sensors networks, mobile infrastructure protocols, network control/mobility management, UMTS, and performance evaluation of wireless communication networks.