Detection of Faulty Reading in Wireless Sensor Networks

Neelam
Department of computer science and engineering
Kurukshetra University, Kurukshetra, India

Abstract—*In this paper, the problem of determining faulty readings in A wireless sensor network without compromising detection of important events is studied. By exploring correlations between readings of sensors, a correlation network is built based on similarity between readings of two sensors. By exploring Markov Chain in the network, a mechanism for rating sensors in terms of the correlation, called SensorRank, is developed. In light of SensorRank, an efficient in-network voting algorithm, called TrustVoting, is proposed to determine faulty sensor readings. Performance studies are conducted via simulation. Experimental results show that the proposed algorithm outperforms majority voting and distance weighted voting, two state-of-the-art approaches for in-network faulty reading detection. Categories and Subject Descriptors: H.3.4 [Systems and Software]: Distributed Systems*

Keywords: Algorithms, Design, Reliability, faulty readings, wireless sensor networks

I. INTRODUCTION

Sensors are prone to failure in harsh and unreliable enviourments. Faulty sensors are likely to report arbitrary readings which do not reject the true state of environmental phenomenon or events under monitoring. Meanwhile, sensors may sometimes report noisy readings resulted from interferences [3]. Both arbitrary and noisy readings are viewed as *faulty readings* in this paper. The presence of faulty readings may cause inaccurate query results and hinder their usefulness. Thus, it is critical to identify and filter out faulty readings so as to improve the query accuracy. In this paper, we target at the problem of determining faulty readings in sensor networks. Obviously, a naive approach to this problem is to collect all readings to a sink, where statistical analysis is performed to determine what readings are outliers. However, this centralized approach may not be practical due to limited energy budget in sensor nodes. If readings are sent to the sink all the time, sensor Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.nodes may soon exhaust their energy. Nevertheless, simply
Fltering out unusual readings at individual sensor nodes may compromise monitoring accuracy of some important events. The goal of this study is to design an energy eficient in-network algorithm for determining faulty readings without compromising detection of important events.

The fact that data readings of nearby sensors are similarcan be captured by *spatial correlation* [6]. Thus, an idea for determining faulty readings is to exploit this spatial correlation. In other words, if a sensor obtains an unusual reading, the sensor could inquire its nearby sensors (referred to as the *witness set*) by sending the suspected reading to them in order to determine whether the reading is faulty or not.

Based on the classical *majority voting*, each sensor (e.g., sensor *si*) in the witness set makes a judgment by comparing its own reading with the unusual reading sent by the suspected sensor (e.g., sensor *sj*). If the difference between these two readings exceeds a predefined threshold, *si* considers the reading sent by *sj* as faulty and gives a negative vote to *sj*. Otherwise, *si* claims that *sj* is normal and returns a positive vote to *sj*. After collecting votes from the nearby sensors, *sj* then can conclude whether the reading is faulty or not. If the number of negative votes is smaller than that of positive votes, the unusual reading reported by *sj* is identified as a faulty reading. Otherwise, it is viewed as an observed event. However, this simple majority voting approach does not work well when the number of faulty sensors increases. To address the problem, two *weighted voting* methods has been proposed in the literature [5, 9]. Motivated by an assumption that the closer sensors have more resemble readings, the weighted voting algorithms give more weights to closer neighbors in voting (i.e., the weights are assigned inverse to the distances from a sensor node to its neighbors).

In this paper, however, we argue that the distance be-tween two sensors does not fully represent the correlationbetween readings of those two sensors. Furthermore, if thenearest sensor is faulty, the voting result may be seriouslycontaminated by this faulty sensor. We refer to this problem as a *domination problem* in the paper. Figure 1 illustrates a sensor network where the neighboring sensor nodes are linked. Each link is labeled by a weight (determined based on heuristics adopted by different voting methods) that will be used in voting. Assume that the weights of sensors *s*2, *s*3 and *s*4 are 0:3*;* 0:4 and 0:9, respectively, and sensor *s*4 is a faulty sensor. Obviously, the reading of sensor *s*1 is identified as a faulty reading when the weighted voting method is

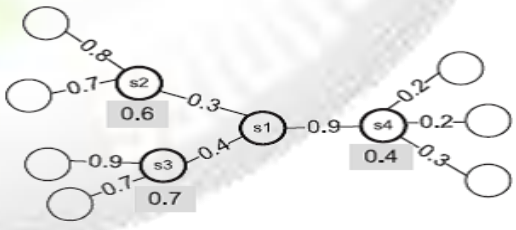

Fig. 1: An illustrative topology of a wireless sensor network.

performed (i.e., 0.3*1+0.4*1+0.9*(-1)=-0.2) where positive and negative votes are represented by 1 and -1, respectively).As shown above, the distance-based weighted voting method has two primary deficiencies: 1) while the distance between sensor nodes may be a factor in generating similar readings of nearby sensor nodes, it does not precisely capture what we really care about as the *correlation* between sensor readings; 2) while it is a good

idea to inquire opinions of neighbors, the *trustworthiness* of neighbors is not considered.

Based on the above observation, in this paper, we propose an innovative in-network voting scheme for determining faulty readings by taking into account the correlation

of readings between sensor nodes and the trustworthiness of a node. To obtain the pair-wise correlations of sensor readings, we construct a logical *correlation network* on top of the sensor network. The correlation network can be depicted by a set of vertices and edges, where each vertex represents a sensor node and an edge between two sensor nodes denotes their correlation (i.e., the similarity between their readings).If two nearby sensor nodes do not have any similarity in their readings, these two sensor nodes do not have an edge connected. Therefore, only sensor nodes that are connected by correlation edges are participated in voting. The weighted voting method actually uses the correlation (modeled as a function of distance) between sensor nodes as weights. However, using the correlation alone may not correctly identify faulty readings due to the domination problem discussed above. Thus, in the proposed algorithm, each sensor node is associated with a trustworthiness value (called *Sensor-Rank*) that will be used in voting. Sensor Rank of a sensor node implicitly represents the number of 'references' (i.e., similar sensor nodes nearby) it has to support its opinions.

A sensor node will obtain a high Sensor Rank if this sensor has many references. The number under each sensornode in Figure 1 is its Sensor Rank. In the figure, *s*4 has asmall Sensor Rank because the readings in *s*4 are not verysimilar to that of its neighbors. By using Sensor Rank, ourvoting scheme takes the trustworthiness of each sensor intoaccount. A vote with small Sensor Rank has only a smallimpact on the final voting result. For example, in Figure 1,when s1 inquires opinions from its neighbors (i.e., *s*2, *s*3 and *s*4), the vote from *s*4 has a small impact due to its lower Sensor Rank.Our design consists of two parts: 1) an algorithm thatcalculates Sensor Rank for each sensor node; and 2) an algorithm that use Sensor Rank to determine faulty readings.Specifically, we first obtain correlations among sensor readings and then model the sensor network as a Markov chain to determine Sensor Rank. In light of Sensor Rank, the *Trust Voting* algorithm we developed will be invoked as needed in operation to effectively determine faulty readings. A preliminary performance evaluation is conducted via simulation.Experimental result shows that the proposed Trust Votingalgorithm is able to effectively identify faulty readings andoutperforms *majority voting* and *distance weighted voting*,two state-of-the-art voting schemes for in-network faulty reading detection for sensor networks. A significant amount of research effort has been elaborated upon issues of identifying faulty sensor readings [2,5, 6, 9]. In [6], the authors explored spatial correlationamong sensors and proposed a distributed Bayesian algorithm for detecting faulty sensors. By assuming that faultymeasurements are either much larger or much smaller thannormal measurements, the authors in [2] use a statisticalmethod to detect outlier measurements. Some variations ofthe weighted voting technique for detecting faulty sensors are proposed in [5] and [9]. In [5],

II. Background And History

This Chapter describes relevant background knowledge and related work for readers to easily understand the proposed protocol and the methodology and analysis of our experiments.

A. Wireless Sensor Network

Wireless sensor network consists of large number of wireless sensor nodes located over a geographic area. The "wireless sensor node" term is for devices that use low power and are equipped with one or more sensors, a radio unit, power supply, processor and an optional actuator. The sensor node can have sensors for the detection and measurement of thermal, mechanical, optical, magnetic, chemical or biological signals. In a basic WSN, the integrated radio unit in a sensor node sends the data collected to the base station. The base station is
normally located far from the sensor nodes and acts as a gateway between the network and subsequent communication centers.

A general structure of WSN is presented in Figure 1. The WSN can be structured or unstructured. A basic wireless sensor network requires very little infrastructure. In one
such network, nodes can be deployed in an ad hoc fashion. The network is not attended after deployment and does monitoring and reporting on its own. However, the sensor network deployed to obtain data from the environment may require a large number of sensor nodes, numbering thousands to tens of thousands depending on the area to be covered. Due to large number of nodes the management of network becomes difficult and complex structure is required. The structured wireless sensor network has planned deployment of sensor nodes, and this means that fewer nodes are required to cover the area compared to an unstructured network. Cost of maintenance and management is reduced.

The wireless sensor network nodes have limitations in terms of limited power available for working, low bandwidth, limited processing capabilities, small range and limited data storage. The network design is based on the environment of operation. Thus, network topologies, theschemes of deployment are decided on a case-to-case basis. Normally, small numbers of nodes are sufficient for indoor coverage whereas outdoor coverage requires large numbers of nodes.[1]

For inaccessible areas only ad hoc deployment is used. Ad hoc deployment is also used when the number of nodes ranges from 100s to1000s

The protocol stack of sensor network is extremely similar to the protocol stack of the traditional ad-hoc networks, with the following layers: Application, Transport, Network, Data Link, and Physical

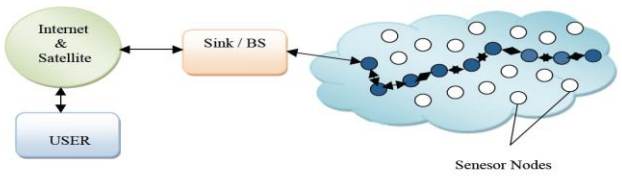

Fig. 1: WSN overview

- Application layer: The application layer is responsible for user interface and data processing.
- Transport layer: This layer specifies the methodology for reliable packet transportation.
- Network layer: The network layer's function is to take care of addressing and forwarding packets.
- Data link layer: The data link layer's function is data streams multiplexing, error control, frame detection and ensuring reliable connections.
- Physical layer: The physical layer functions are to define frequency in use, signal characteristics such as modulation scheme and encryption of a wireless sensor device (node) are illustrated in Figure 2. A wireless sensor device is generally composed of four basic components: a sensing unit, a processing unit, a transceiver unit and a power unit usually in the form a battery

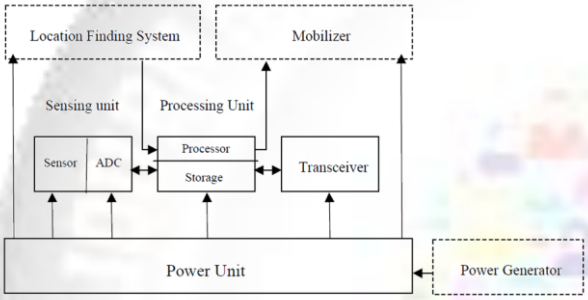

Fig.2 general hardware architecture of a sensor node

Sensing unit Processing Unit

Figure .2: General hardware architecture of a sensor node

Each sensing unit comprises of sensor(s) for sensing environment and analog-to-digital converter (ADC) . Nodes transmit their sensed data if certain pre-defined conditions are met. The environmental signal is received in the form of an analog signal by the sensor and then is converted into a digital signal by the ADC. The Processing unit consists of a microcontroller or in some applications a microprocessor and is responsible for analyzing the attributes of the sensed data by using digital signals. The Transceiver is for connecting the nodes and the BS through a radio transmitter. Lastly, the power unit is usually a battery. Based on different applications, there might be extra components such as localization unit, energy producer, position changer, etc. These components are shown in Figure .2 by the dashed boxes.

B. WSN routing protocols

Many new energy saving protocols distinctively designed for sensor networks, are results of the recent advancements in WSN. Wireless communication is considered the primary component of energy consumption in WSN So particular attention was given to the routing protocols, which can vary contingent on the application and network architecture. The routing protocols in WSNs are broken down into three categories. First, direct communication (DC), which is the simplest protocol, where sensor nodes send data directly to the BS. The second category involve Minimum Transmission Energy (MTE) protocols, where nodes route data to the base station through intermediate nodes, each node acting as a router for the other nodes. The third and perhaps most interesting category are made up of clustering protocols. Hierarchical or cluster-based routing, originally presented in wire-line networks, are recognized techniques with particular advantages related to scalability and efficient communication. Cluster-based routing has been shown to be more effective than DC and MTE and is hence focused on in this work.

C. SENSORRANK

Sensor Rank is to represent the trustworthiness of sensornodes. By our design, two requirements need to be met in
deriving Sensor Rank for each sensor.Requirement 1: If a sensor has a large number of neighbors with correlated readings, the opinion of this sensor istrustworthy and thus its vote deserves more weight.Requirement 2: A sensor node with a lot of trustworthyNeighbors is also trustworthy.These two requirements ensure that 1) a sensor node whichhas a large number of similar neighbors to have a high rank; and 2) a sensor node which has a large number of 'good references' to have a high rank. Given a correlation network$G = (V; E)$ derived previously, we determine SensorRank for each sensor to meet the above two requirements.We model the correlation network as a Markov chain *M*, where each sensor *si* is viewed as the state *i*, and the *transition probability* from state *i* (i.e., sensor *si*) to state *j*(i.e., sensor *si*) is denoted as *pi;j* and formulated as *pi;j*=P *corri;j k2nei*(*i*) *corri;k*
. For example, in Figure 2, *p2;*3 = 0*:*1
0*:*4+0*:*1+0*:*7 =
0*:*083. Based on the above setting, we can formulate Sen-Sor Rank of *si*, denoted as rank *i*, as follows:
ranki=X *sj2nei*(*i*)
rankj ¢ pj;i
where*nei*(*i*) is the witness set of node *i*.
The computation of Sensor Rank can be viewed as a random walk over the correlation network. Several iterationsis required to perform random walks until a steady stateis achieved (i.e., Sensor Ranks become stable).Specifically, *rank*(*k*) *i* is the value of SensorRank at the *k*-th iterations.

At the beginning, the initial *rank*(0) *i* is set to 1. Note that*rank*(0) *i* can be set to any constant *c*, and the results will be *c* times the value generated when the initial SensorRank is set to 1. In the first round, each sensor node *si* updates its Sensor Rank as *rank*(1) *i* using the initial SensorRanks of its neighbors. Now each sensor node has considered the first level neighbors to calculate its SensorRank. In the second round, each sensor node can indirectly obtain some information from the second level neighbors through its first level neighbors since its first level neighbors have explored their first level neighbors as well. Therefore, after the *k*th round, sensor node *si*has explored the *k*th level neighbors and up-dated SensorRank as *rank*(*k*) *i*.Consider an example in Figure 2. In the first round, *s*3 has some similarity information from its first level neigh-

Algorithm 1 *SensorRank*

Input: a sensor s_i, and a threshold δ.
Output: $rank_i$ for s_i.

1: $rank_i^{(0)} = 1$
2: for $k = 1$ to δ do
3: for all $s_j \in nei(s_i)$ do
4: $p_{i,j} = \frac{corr_{i,j}}{\sum_{s_k \in nei(i)} corr_{i,k}}$
5: send $rank_i^{(k-1)} \cdot p_{i,j}$ to s_j
6: receive all $rank_j^{(k-1)} \cdot p_{j,i}$ from every $s_j \in nei(i)$
7: $rank_i^{(k)} = \sum_{s_j \in nei(i)} rank_j^{(k-1)} \cdot p_{j,i}$

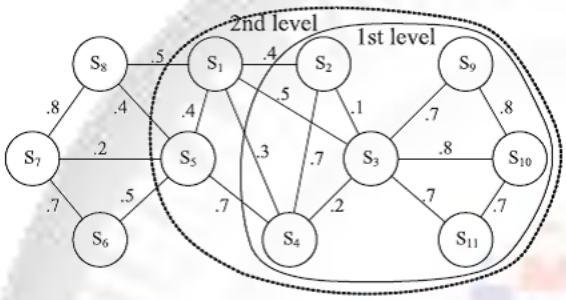

Fig. 2: An example of Sensor Rank.

bors $\{s_2, s_4, s_9, s_{10}, s_{11}\}$. Similarly, both s_2 and s_4 could exchange some information with their neighbors. In the second round, s_3 can obtain similarity information from the second level neighbors $\{s_1, s_5\}$ since its first level neighbors s_2 and s_4 have explored s_1 and s_5 during the first round. If k is larger, SensorRanks will be more accurate since every sensor can explore more neighbors. In sensor networks, the computation cost will be larger when the number of iterations is larger. Therefore, we can limit k to a preset bound δ.

Given a correlation network in Figure 2, we now demonstrate how to calculate SensorRank. Initially, sensor s_i sets its sensorRank $rank_i^{(0)}$ to 1. For sensor s_i, s_i calculates the trust relations $p_{i,j}$ to the corresponding neighbor s_j and sends $rank_i^{(0)} \cdot p_{i,j}$ to s_j. For example, s_3 sends $rank_3^{(0)} \cdot p_{3,1} = 1 \cdot \frac{0.5}{3.0} = 0.167$ to s_1, 0.033 to s_2, $\frac{0.2}{3} = 0.067$ to s_4, and etc. At the same time, s_3 receives SensorRanks from its neighbors. For example, s_3 receives $rank_2^{(0)} \cdot p_{2,3} = 1 \cdot \frac{0.1}{0.4+0.1+0.7} = 0.083$ from s_2. Upon receiving all the proportion of SensorRank from the neighbors, s_3 can update its SensorRank to $rank_3^{(1)}$.

$$\begin{aligned} rank_3^{(1)} &= \sum_{i \in \{1,2,4,9,10,11\}} rank_i^{(0)} \cdot p_{j,i} \\ &= 1 \cdot p_{1,3} + 1 \cdot p_{2,3} + 1 \cdot p_{4,3} + 1 \cdot p_{9,3} \\ &\quad + 1 \cdot p_{10,3} + 1 \cdot p_{11,3} \\ &= \frac{0.5}{2.1} + \frac{0.1}{1.2} + \frac{0.2}{1.9} + \frac{0.7}{1.5} + \frac{0.8}{2.3} + \frac{0.7}{1.4} \\ &= 1.74 \end{aligned}$$

After the first round, $\left\{rank_i^{(1)} | i = 1, 2, 3, 4\right\}$ ={1.13, 0.59, 1.11, 1.33}. In the second round, sensors calculate the values of SensorRank with the updated values of SensorRank in the first round. For example, s_1 now sends $rank_1^{(1)} \cdot p_{1,3} = 1.13 \cdot$

III. Literature Review

A. *Trustvoting Algorithm*

Here we describe our design of the *TrustVoting* algorithm, which consists of two phases: a) self-diagnosis; and b)neighbors diagnosis phase. In the self-diagnosis phase,each sensor verifies whether the current reading of a sensoris unusual or not. Once the reading of a sensor goesthrough the self-diagnosis phase, this sensor can directly report the reading.

Otherwise, the sensor node consults with its neighbors tofurther validate whether the current reading is faulty or not.If a reading is determined as faulty, it will be filtered out.The sensor nodes generating faulty readings will not participate in voting since these sensors are likely to contaminatethe voting result. Note that TrustVoting is an in-network algorithm which is executed in a distributed manner. Theexecution order of algorithm TrustVoting has an impact onfaulty reading detection. We will discuss this issue later.

Algorithm 2 *TrustVoting*
Input: a sensor *si*, SensorRank*rank i* and time interval *t*
Output: justify whether the reading is faulty or not (i.e., *faulty= true* or not)
1: set *faulty = false*
2: broadcast *ranki*to the neighbors
3: receive *frankjjsj 2 nei*(*i*)*g*from the neighbors
/* set *timer* by the priority sorted by SensorRank */
4: sort SensorRank values received
5: *x* = *ranki*'s order in the sorted SensorRank values
6: *n* = neighbors of sensor *si*
7: *timer* = *x* ¤ (*t*
n+1) /**t* is the time interval given */
8: while *time* == *timer* do
9: *faulty* = Procedure *Self-Diagnosis*
10: if *faulty* == *true* then
11: *faulty* = Procedure *Neighbor-Diagnos*

IV. Related Works

Primary function of wireless sensor networks is to gather sensor data from the monitored area. Due to faults or malicious nodes, however, the sensor data collected or reported might be wrong. Hence it is important to detect events in the presence of wrong sensor readings and misleading reports. In this paper, we present a neighbor-based malicious node detection scheme for wireless sensor networks. Malicious nodes are modeled as faulty nodes behaving intelligently to lead to an incorrect decision or energy depletion without being easily detected. Each sensor node makes a decision on the fault status of itself and its neighboring nodes based on the sensor readings. Most erroneous readings due to transient faults are corrected by filtering, while nodes with permanent faults are removed using confidence-level evaluation, to improve malicious node detection rate and event detection accuracy. Each node maintains confidence levels of itself and its neighbors, indicating the track records in reporting past events correctly. Computer simulation shows that most of the malicious nodes reporting against their own readings are correctly detected unless they behave similar to the normal nodes. As a result, high event detection accuracy is also maintained while achieving a low false alarm rate.

In a wireless sensor network, operating in a harsh and unattended environment, sensor nodes may generate incorrect sensor readings and wrong reports to their neighbors, causing incorrect decisions or energy depletion. The potential sources of incorrect readings and reports include noise, faults, and malicious nodes in the network. Unlike noise and faults, malicious nodes can arbitrarily modify the sensed data and intentionally generate wrong reports. To ensure a reliable event detection in the presence of such wrong data and reports, it is necessary to detect and isolate malicious nodes, greatly reducing their impact on decision-making.

Several fault detection schemes for wireless sensor networks have been proposed in the literature [1-5]. They use centralized, distributed, or hierarchical models. Due to the communication overhead most schemes employ a distributed model, using either neighbor coordination or clustering. As the fault or error models for detection, noise and a few types of faults, such as transient and permanent faults, are typically used. Malicious nodes, however, can generate arbitrary sensor readings which do not conform to the typically used fault models. In that case, the resulting malicious node detection rate becomes much poorer than the estimated one.

Rajasegarar et al. presented an overview of existing outlier detection schemes for wireless sensor networks [6]. Sensor readings that appear to be inconsistent with the remainder of the data set are the main target of the detection. Curiac et al. [7] proposed a detection scheme using auto regression technique. Signal strength is used to detect malicious nodes in [8], where a message transmission is considered suspicious if the strength is incompatible with the originator's geographical position. Xiao et al. developed a mechanism for rating sensors in terms of correlation by exploring Markov Chain [9]. A network voting algorithm is proposed to determine faulty sensor readings.

Atakli et al. [10] presented a malicious node detection scheme using weighted trust evaluation for a three-layer hierarchical network architecture. Trust values are employed to identify malicious nodes behaving opposite to the sensor readings. They are updated depending on the distribution of neighboring nodes. An improved intrusion detection scheme based on weighted trust evaluation was proposed in [12]. The mistaken ratio of each individual sensor node is used in updating the trust values. Trust management schemes have been proposed in routing and communications [13]. Some efforts are also being made to combine communication and data trusts [14]. However, malicious node detection in the presence of various types of misleading sensor readings due to the compromised nodes have not been deeply investigated. In addition, the resulting event detection performance has not sufficiently been taken into account in malicious node detection.

In this paper, we present a neighbor-based malicious node detection scheme for wireless sensor networks. Malicious nodes are modeled as faulty nodes that may intentionally report false data with some intelligence not to be easily detected. The scheme identifies malicious nodes unless they behave similar to normal nodes. Confidence levels and weighted majority voting are employed to detect and isolate malicious nodes without sacrificing normal nodes and degrading event detection accuracy.

V. Proposed Work

The primary function of wireless sensor networks is to gather sensor data from the monitored area. Due to faults or malicious nodes, however, the sensor data collected or reported might be wrong. Hence it is important to detect events in the presence of wrong sensor readings and misleading reports. In this paper, we present a neighbor-based malicious node detection scheme for wireless sensor networks. Malicious nodes are modeled as faulty nodes behaving intelligently to lead to an incorrect decision or energy depletion without being easily detected. Each sensor node makes a decision on the fault status of itself and its neighboring nodes based on the sensor readings. Most erroneous readings due to transient faults are corrected by filtering, while nodes with permanent faults are removed using confidence-level evaluation, to improve malicious node detection rate and event detection accuracy. Each node maintains confidence levels of itself and its neighbors, indicating the track records in reporting past events correctly. Computer simulation shows that most of the malicious nodes reporting against their own readings are correctly detected unless they behave similar to the normal nodes. As a result, high event detection accuracy is also maintained while achieving low false alarm rate.

VI. Conclusion

With the presence of faulty readings, the accuracy of query results in wireless sensor networks may be greatly affected. In this paper, we first formulated the correlation among readings of sensors nodes. Given correlations among sensor nodes, a correlation network is built to facilitate derivation of SensorRank for sensor nodes in the network. In light of SensorRank, an in-network algorithm TrustVoting is developed to determine faulty readings. Performance evaluation shows that by exploiting SensorRank, algorithm TrustVoting is able to efficiently identify faulty readings and out performs majority voting and distance weighted voting two state-of-the-art approaches for in-network faulty reading detection.

References

[1] M. Yu, H. Mokhtar and M. Merabti, "Fault Management in Wireless Sensor Networks," IEEE Wireless Communications, Vol. 14, No. 6, 2007, pp. 13-19. doi:10.1109/MWC.2007.4407222

[2] H. S. Hu and G. H. Qin, "Fault Management Frameworks in Wireless Sensor Networks," 4th International Conference Intelligent Computation Technology and Automation, Shenzhen, 28-29 March 2011, pp. 1093-1096. doi:10.1109/ICICTA.2011.559

[3] C.-R. Li and C.-K. Liang, "A Fault-Folerant Event Boundary Detection Algorithm in Sensor Networks," Information Networking: Towards Ubiquitous Networking and Services, Vol. 5200, 2008, pp. 406-414. doi:10.1007/978-3-540-89524-4_41

[4] X. H. Xu, B. Zhou and J. Wan, "Tree Topology Based Fault Diagnosis in Wireless Sensor Networks," International Conference on Wireless Networks and

Information Systems, Shanghai, 28-29 December 2009, pp. 65-69.

[5] M. H. Lee and Y.-H. Choi, "Fault Detection of Wireless Sensor Networks," Computer Communications, Vol. 31, No. 14, 2008, pp. 3469-3475. doi:10.1016/j.comcom.2008.06.014

[6] S. Rajasegarar, C. Leckie and M. Palaniswami, "Anomaly Detection in Wireless Sensor Networks," IEEE Wireless Communications, Vol. 15, No. 4, 2008, pp. 34-40. doi:10.1109/MWC.2008.4599219

[7] D. I. Curiac, O. Banias, F. Dragan, C. Volosencu and O. Dranga, "Malicious Node Detection in Wireless Sensor Networks Using an Autoregression Technique," 3rd International Conference on Networking and Services, Athens, 19-25 June 2007, p. 83.

[8] W. Junior, T. Figueiredo, H. Wong and A. Loureiro, "Malicious Node Detection in Wireless Sensor Networks," 18th International Parallel and Distributed Processing Symposium, 26-30 April 2004, New Mexico, p. 24.

[9] X.-Y. Xiao, W.-C. Peng, C.-C. Hung and W.-C. Lee, "Using Sensor Ranks for In-Network Detection of Faulty Readings in Wireless Sensor Networks," International Workshop Data Engineering for Wireless and Mobile Access, Beijing, 10 June 2007, pp. 1-8. doi:10.1145/1254850.1254852

[10] I. M. Atakli, H. Hu, Y. Chen, W.-S. Ku and Z. Su, "Malicious Node Detection in Wireless Sensor Networks Using Weighted Trust Evaluation," Proceedings of Spring Simulation Multi-Conference, Ottawa, 14-17 April 2008, pp. 836-843.

[11] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor Network Security: A Survey," IEEE Communication Surveys & Tutorials, Vol. 11, No. 2, 2009, pp. 52-73. doi:10.1109/SURV.2009.090205

[12] L. Ju, H. Li, Y. Liu, W. Xue, K. Li and Z. Chi, "An Improved Detection Scheme Based on Weighted Trust Evaluation for Wireless Sensor Networks," Proceedings of the 5th International Conference on Ubiquitous Information Technology and Applications, Sanya, 16-18 December 2010, pp. 1-6.

[13] M. Momani and S. Challa, "Survey of Trust Models in Different Network Domain," International Journal Ad Hoc, Sensor & Ubiquitous Computing, Vol. 1, No. 3, 2010, pp. 1-19.

[14] M. Momani, S. Challa and R. Alhmouz, "Can We Trust Trusted Nodes in Wireless Sensor Networks?" International Conference Computer and Communication Engineering, Kuala Lumpur, 13-15 May 2008, pp. 1227-1232.