IN-NETWORK OUTLIER DETECTION IN WIRELESS SENSOR NETWORKS

PROJECT REPORT

Submitted by

P.ALLEN FOSTER	**71206205003**
P.ARUN MANI	**71206205006**
N.R.NIRMAL	**71206205030**

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY:: CHENNAI 600025

APRIL 2010

ANNA UNIVERSITY :CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report " IN-NETWORK OUTLIER DETECTION IN WIRELESS SENSOR NETWORKS " is the bonafide work of "P.ALLEN FOSTER , P.ARUN MANI and N.R.NIRMAL", who carried out the project work under my supervision.

SIGNATURE
Ms.N. CHITRA DEVI,M.E.
ASSISTANT PROFESSOR
Dept of Information Technology
Kumaraguru College of Technology
Coimbatore – 641 006.

SIGNATURE
Dr.L.S JAYASHREE,M.E.,Ph.D
HEAD OF THE DEPARTMENT
Dept of Information Technology
Kumaraguru College of Technology
Coimbatore – 641 006.

The candidates with the university register number **71206205003**, **71206205006** and **71206205030** were examined by us in the project viva-voce examination held on .19.04.2010...

INTERNAL EXAMINER 19/4/10

EXTERNAL EXAMINER 19.4.10

DECLARATION

We,

P.ALLEN FOSTER (71206205003)
P.ARUN MANI (71206205006)
N.R.NIRMAL (71206205030)

Hereby declare that the project entitled " **IN-NETWORK OUTLIER DETECTION IN WIRELESS SENSOR NETWORKS** "submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru College of Technology, Coimbatore

Place: Coimbatore
Date: 19-04-2010

[P. Allen Foster] [P. Arunmani] [N.R. Nirmal]

Project Guided by

[Ms.N. Chitra Devi]

ACKNOWLEDGEMENT

We express our sincere thanks to our Chairman **Padmabhushan Arutselvar Dr.N.Mahalingam.B.Sc.,F.I.E.,** Co-Chairman **Dr.B.K.Krishnaraj Vanavarayar,** Correspondent **Shri.M.Balasubramaniam** and Director **Dr.J.Shanmugam,Ph.D,** for all their support and ray of strengthening hope extended.

We are immensely grateful to our Principal **Dr.S.Ramachandran,Ph.D.,** for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy,B.E(Hons).,Ph.D** Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project co-ordinator, **Dr.L.S.Jayashree.M.E.,Ph.D,** Head of the department, Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide and extend our gratitude towards **Ms.N.Chitradevi ,M.E,** Assistant.Prof, Department of Information Technology for her helpful guidance and valuable support given to us throughout this project.

We sincerely thank our class advisor, Department of Information Technology, who gave the initial idea and guidance to carry out every further step for his everlasting counseling and untiring help throughout our project.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

ABSTRACT

ABSTRACT

Sensors are prone to failure in harsh and unreliable environments. Faulty sensors are likely to report arbitrary readings which do not reflect the true state of environmental phenomenon or events under monitoring. Meanwhile, sensors may sometimes report noisy readings resulted from interferences. 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, 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 Sensor Rank, is developed. In light of Sensor Rank, an efficient in-network voting algorithm, called TrustVoting, is proposed to determine faulty sensor readings.

TABLE OF CONTENTS

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**vi**
	LIST OF FIGURES	**x**
1.	**INTRODUCTION**	
	1.1 PROBLEM DEFINITION	2
	1.2 OBJECTIVE OF THE PROJECT	3
	1.3 EXISTING SYSTEM	4
2.	**SYSTEM REQUIREMENT**	
	2.1 HARDWARE REQUIREMENTS	6
	2.2 SOFTWARE SPECIFICATION	
	2.2.1 MATLAB	6
	2.2.2 KEY FEATURES	7
3.	**SYSTEM STUDY**	
	3.1 WIRELESS SENSOR NETWORK	
	3.1.1 AN OVERVIEW	9
	3.1.2 ISSUES	11
	3.1.3 APPLICATIONS OF WSN	11
	3.2 LITERATURE REVIEW	16

3.3 DETAILS OF METHODOLOGY EMPLOYED

3.3.1 CORRELATION NETWORK FORMATION 19

3.3.2 SENSOR RANK CALCULATION 20

3.3.3 TRUST VOTING ALGORITHM 22

3.3.3.1 SELF DIAGNOSIS PHASE..... 23

3.3.3.2 NEIGHBOR DIAGNOSIS PHASE..... 24

4. **CONCLUSION**..... 28

5. **APPENDICES**

5.1 SOURCE CODE..... 30

5.2 SCREEN SHOTS..... 35

6. **REFERENCES**..... 41

LIST OF FIGURES

FIG NO	TITLE	PAGE NO
1	The MATLAB development environment	7
2	Sensor Network model	10
3	Example of Sensor rank evaluation	21
4	An Example for trust voting.....	25

INTRODUCTION

1. INTRODUCTION

1.1 PROBLEM DEFINITION

A wireless sensor network (WSN) is a wireless network which consists of equally distributed autonomous devices using sensors capable of monitoring the physical or environmental conditions such as temperature, sound, vibration, pressure, motion or pollutants, at various locations. Sensor nodes are small, lightweight and portable. Every sensor node is equipped with a transducer, microcomputer, transceiver and power source.

These Sensors are prone to failure in harsh and unreliable environments. Faulty sensors are likely to report arbitrary readings which do not reflect the true state of environmental phenomenon or events under monitoring. Meanwhile, sensors may sometimes report noisy readings resulted from interferences. Both arbitrary and noisy readings are viewed as *faulty readings*. 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.

Sensors can easily deviate from their result due to technical fault in the sensor component. The reading would be a faulty one even when the sensors are closely oriented in distance. It becomes really hard to detect which of the sensors is showing a faulty reading as all of them are closely oriented and analyzing a particular situation. In such case the usefulness of the sensors are hindered as the faulty readings are shown and it becomes a tedious process to judge the result based on the readings of the sensor.

1.2 OBJECTIVE OF THE PROJECT

Based on the above observation, 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. 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. By using Sensor Rank, our voting scheme takes the trustworthiness of each sensor into account. A vote with small Sensor Rank has only a small impact on the final voting result.

1.3 EXISTING SYSTEM

We target at the problem of determining faulty readings in sensor networks. Our design consists of two parts:

1) An algorithm that calculates Sensor Rank for each sensor node.

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 SensorRank. In light of SensorRank, the TrustVoing 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 TrustVoting algorithm is able to effectively identify faulty readings and outperforms majority voting and distance weighted voting, two state-of-the-art voting schemes for in-network faulty reading detection for sensor networks. 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 between sensor nodes as weights.

SYSTEM REQUIREMENT

2. SYSTEM REQUIREMENT

2.1 HARDWARE REQUIREMENTS

Processor	:	Pentium IV
Speed	:	Above 500 MHz
RAM capacity	:	2 GB
Floppy disk drive	:	1.44 MB
Hard disk drive	:	200 GB
Key Board	:	Samsung 108 keys
Mouse	:	Logitech Optical Mouse
CD Writer	:	52x LG
Printer	:	DeskJet HP
Motherboard	:	Intel
Cabinet	:	ATX
Monitor	:	17" Samsung

2.2 SOFTWARE SPECIFICATION

2.2.1 MATLAB

MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. Using the MATLAB product, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

You can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement,

financial modeling and analysis, and computational biology. Add-on toolboxes (collections of special-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas.

2.2.2 KEY FEATURES

- High-level language for technical computing
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration
- 2-D and 3-D graphics functions for visualizing data
- Tools for building custom graphical user interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM.

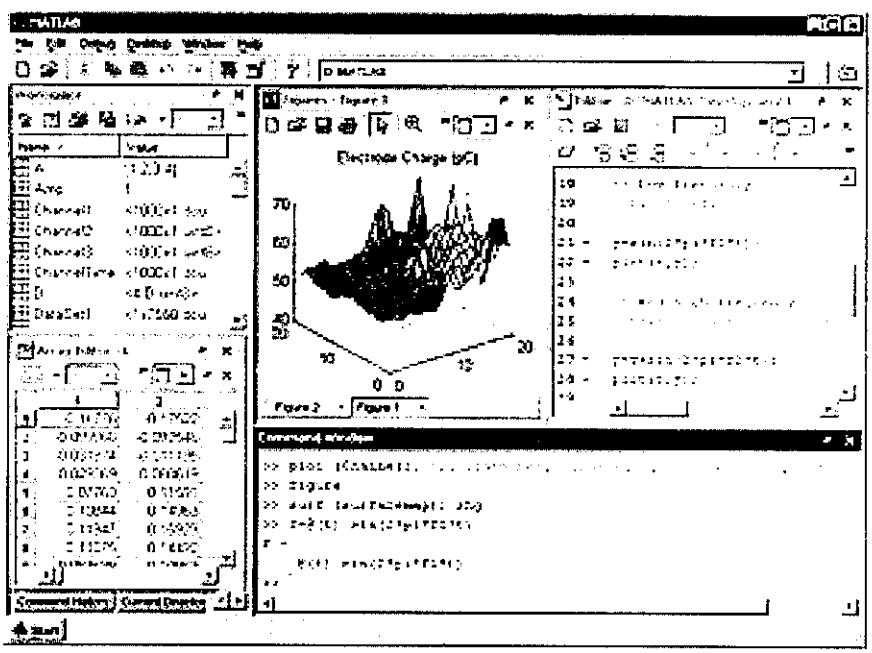


Fig 1 *The MATLAB development environment lets you develop algorithms, interactively analyze data, view data files, and manage projects.*

SYSTEM STUDY

3. SYSTEM STUDY

3.1 WIRELESS SENSOR NETWORKS

3.1.1 AN OVERVIEW

A wireless sensor network (WSN) is a wireless network which consists of equally distributed autonomous devices using sensors capable of monitoring the physical or environmental conditions such as temperature, sound, vibration, pressure, motion or pollutants, at various locations. Sensor nodes are small, lightweight and portable. Every sensor node is equipped with a transducer, microcomputer, transceiver and power source. The transducer generates electrical signals based on sensed physical effects and phenomena. The microcomputer processes and stores the sensor output. The transceiver, which can be hard-wired or wireless, receives commands from a central computer and transmits data to that computer. The power for each sensor node is derived from the electric utility or from a battery.

Wireless Sensor Networks has been an active research area during recent years. Compared with traditional wireless networks, WSN has many unique features. WSN devices usually consume very low power, and they can withstand harsh environmental conditions. In addition, WSN devices are highly mobilized and they are mostly automated so that not many operations are needed. Due to these features WSN has been used in a broad variety of applications. Some typical applications are monitoring nature inhabitants, health system and eco-system, gathering data on temperature, sound and pressure, etc. However, to make WSN applicable for different applications, people need to put a lot of effort because of the non-standard structure of them. The cost and size vary a lot. Accordingly,

energy, memory, computational speed and bandwidth of different sensors are very much alike.

Sensor networks provide a web of interconnectivity: multiple sources of information that will allow decision-making processes to be more accurate and efficient. These processes can be complex and demanding however, and are often constrained in a number of possibly conflicting dimensions such as quality, responsiveness and cost.

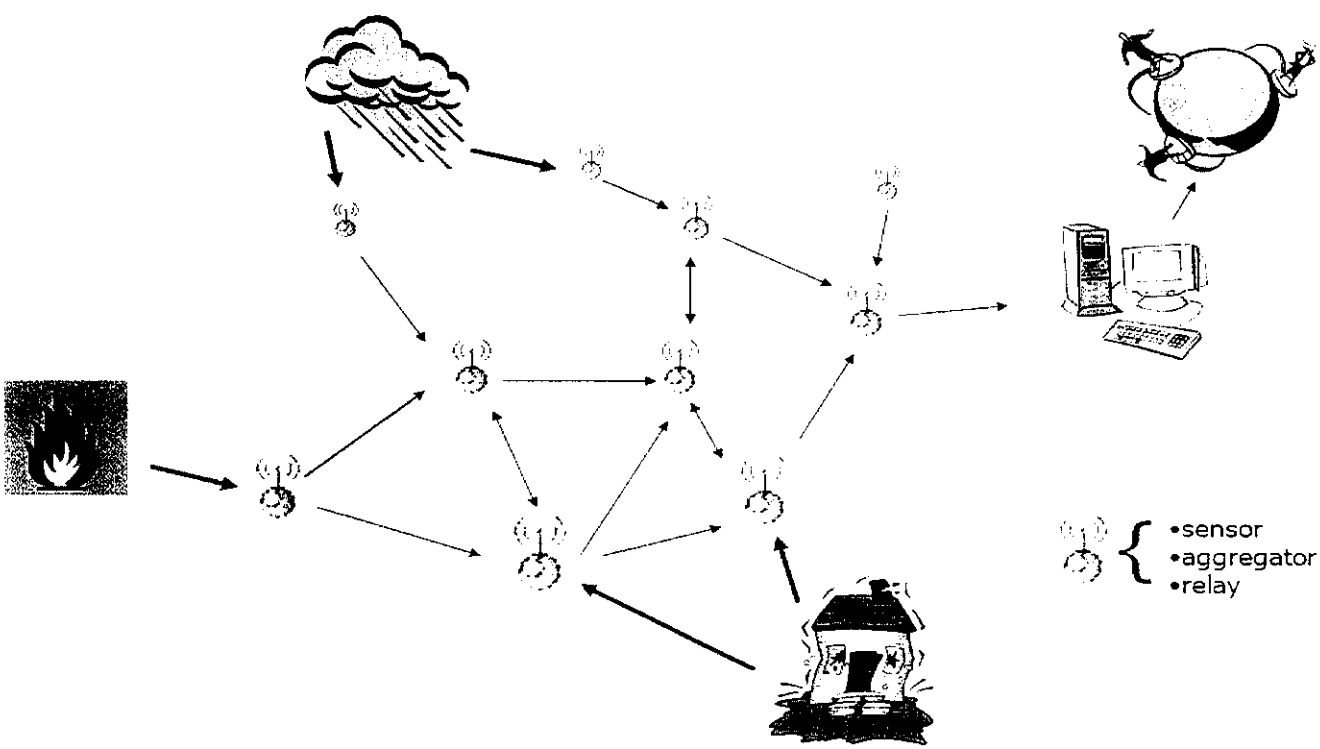


Figure 2 Sensor network model

3.1.2 ISSUES

Considerable advances have been made in recent years in hardware and software for building wireless sensor networks. However, to ensure effective data gathering by sensor networks for monitoring remote outdoor environments, the following problems remain:

Reactivity: the ability of the network to react to its environment, and provide only relevant data to users.

Robustness: the ability of network nodes to function correctly in harsh outdoor environments.

Network lifetime: maximising the length of time the network is able to deliver data before batteries are exhausted.

Communication failures: The capacity of a sensor network to deliver sensed data can vary significantly over time as a result of communication failures.

Fortunately, most environment monitoring applications can tolerate delays of hours or even days to deliver gathered data. Furthermore, the demand from environment sensor nodes to report data is usually a very small percentage of the network's normal capacity.

3.1.3 APPLICATIONS OF WSN

The applications for WSNs are varied, typically involving some kind of monitoring, tracking, or controlling. Specific applications include habitat monitoring, object tracking, nuclear reactor control, fire detection, and traffic monitoring. In a typical application, a WSN is scattered in a region where it is meant to collect data through its sensor nodes.

3.1.3.1 AREA MONITORING

Area monitoring is a common application of WSNs. In area monitoring, the WSN is deployed over a region where some phenomenon is to be monitored. For example, a large quantity of sensor nodes could be deployed over a battlefield to detect enemy intrusion instead of using landmines When the sensors detect the event being monitored (heat, pressure, sound, light, electro-magnetic field, vibration, etc), the event needs to be reported to one of the base stations, which can take appropriate action (e.g., send a message on the internet or to a satellite). Depending on the exact application, different objective functions will require different data-propagation strategies, depending on things such as need for real-time response, redundancy of the data (which can be tackled via data aggregation and information fusion techniques), need for security, etc.

3.1.3.2 MACHINE HEALTH MONITORING OR CONDITION BASED MAINTENANCE

Wireless sensor networks have been developed for machinery condition-based maintenance (CBM) as they offer significant cost savings and enable new functionalities. Previously inaccessible locations, rotating machinery, hazardous or restricted areas, and mobile assets can now be reached with wireless sensors. Often, companies use manual techniques to calibrate, measure, and maintain equipment. This labor-intensive method not only increases the cost of maintenance but also makes the system prone to human errors. Especially in US Navy shipboard systems, reduced manning levels make it imperative to install automated maintenance monitoring systems. Wireless sensor networks play an important role in providing this capability.

3.1.3.3 LANDFILL GROUND WELL LEVEL MONITORING AND PUMP COUNTER

Wireless sensor networks can be used to measure and monitor the water levels within all ground wells in the landfill site and monitor leachate accumulation and removal. A wireless device and submersible pressure transmitter monitors the leachate level. The sensor information is wirelessly transmitted to a central data logging system to store the level data, perform calculations, or notify personnel when a service vehicle is needed at a specific well.

It is typical for leachate removal pumps to be installed with a totalizing counter mounted at the top of the well to monitor the pump cycles and to calculate the total volume of leachate removed from the well. For most current installations, this counter is read manually. Instead of manually collecting the pump count data, wireless devices can send data from the pumps back to a central control location to save time and eliminate errors. The control system uses this count information to determine when the pump is in operation, to calculate leachate extraction volume, and to schedule maintenance on the pump.

3.1.3.4 FLARE STACK MONITORING

Landfill managers need to accurately monitor methane gas production, removal, venting, and burning. Knowledge of both methane flow and temperature at the flare stack can define when methane is released into the environment instead of combusted. To accurately determine methane production levels and flow, a pressure transducer can detect both pressure and vacuum present within the methane production system.

Thermocouples connected to wireless I/O devices create the wireless sensor network that detects the heat of an active flame, verifying that methane is burning. Logically, if the meter is indicating a methane flow and the temperature at the flare

stack is high, then the methane is burning correctly. If the meter indicates methane flow and the temperature is low, methane is releasing into the environment.

3.1.3.5 GREENHOUSE MONITORING

Wireless sensor networks are also used to control the temperature and humidity levels inside commercial greenhouses. When the temperature and humidity drops below specific levels, the greenhouse manager must be notified via e-mail or cell phone text message, or host systems can trigger misting systems, open vents, turn on fans, or control a wide variety of system responses. Because some wireless sensor networks are easy to install, they are also easy to move as the needs of the application change

3.1.3.6 STRUCTURAL HEALTH MONITORING – SMART STRUCTURES

Sensors embedded into machines and structures enable condition-based maintenance of these assets. Typically, structures or machines are inspected at regular time intervals, and components may be repaired or replaced based on their hours in service, rather than on their working conditions. This method is expensive if the components are in good working order, and in some cases, scheduled maintenance will not protect the asset if it was damaged in between the inspection intervals. Wireless sensing will allow assets to be inspected when the sensors indicate that there may be a problem, reducing the cost of maintenance and preventing catastrophic failure in the event that damage is detected. Additionally, the use of wireless reduces the initial deployment costs, as the cost of installing long cable runs is often prohibitive.

In some cases, wireless sensing applications demand the elimination of not only lead wires, but the elimination of batteries as well, due to the inherent nature of the machine, structure, or materials under test. These applications include

sensors mounted on continuously rotating parts, within concrete and composite materials, and within medical implants.

3.1.3.7 INDUSTRIAL AUTOMATION

In addition to being expensive, leadwires can be constraining, especially when moving parts are involved. The use of wireless sensors allows for rapid installation of sensing equipment and allows access to locations that would not be practical if cables were attached. An example of such an application on a production line is shown in Figure. In this application, typically ten or more sensors are used to measure gaps where rubber seals are to be placed. Previously, the use of wired sensors was too cumbersome to be implemented in a production line environment. The use of wireless sensors in this application is enabling, allowing a measurement to be made that was not previously practical.

The various applications are follows:

- 1.1 Area monitoring
- 1.2 Environmental monitoring
- 1.3 Industrial Monitoring
  - 1.3.1 Water/Wastewater Monitoring
    - 1.3.1.1 Landfill Ground Well Level Monitoring and Pump Counter
    - 1.3.1.2 Flare Stack Monitoring
    - 1.3.1.3 Water Tower Level Monitoring
  - 1.3.2 Vehicle Detection
  - 1.3.3 Agriculture
    - 1.3.3.1 Windrow Composting
    - 1.3.3.2 Green house monitoring

3.2 LITERATURE REVIEW

Sensors are prone to failure in harsh and unreliable environments. Faulty sensors are likely to report arbitrary readings which do not reflect the true state of environmental phenomenon or events under monitoring. Meanwhile, sensors may sometimes report noisy readings resulted from interferences. 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.

T. Clouqueur[1] proposed Fault tolerance in collaborative sensor networks for target detection. Collaboration in sensor networks must be fault-tolerant due to the harsh environmental conditions in which such networks can be deployed. This paper focuses on finding algorithms for collaborative target detection that are efficient in terms of communication cost, precision, accuracy, and number of faulty sensors tolerable in the network. Two algorithms, namely, value fusion and decision fusion, are identified first. When comparing their performance and communication overhead, decision fusion is found to become superior to value fusion as the ratio of faulty sensors to fault free sensors increases. The impact of hierarchical agreement on communication cost and system failure probability is evaluated and a method for determining the number of tolerable faults is identified.

Eiman Elnahrawy[3] proposed Online Data Cleaning in Wireless Sensor Networks[3] .We present our ongoing work on data quality problems in sensor networks. Specifically, we deal with the problems of outliers, missing information, and noise. We propose an approach for modeling and online learning of spatio-

temporal correlations in sensor networks. We utilize the learned correlations to discover outliers and recover missing information. We also propose a Bayesian approach for reducing the effect of noise on sensor data online.

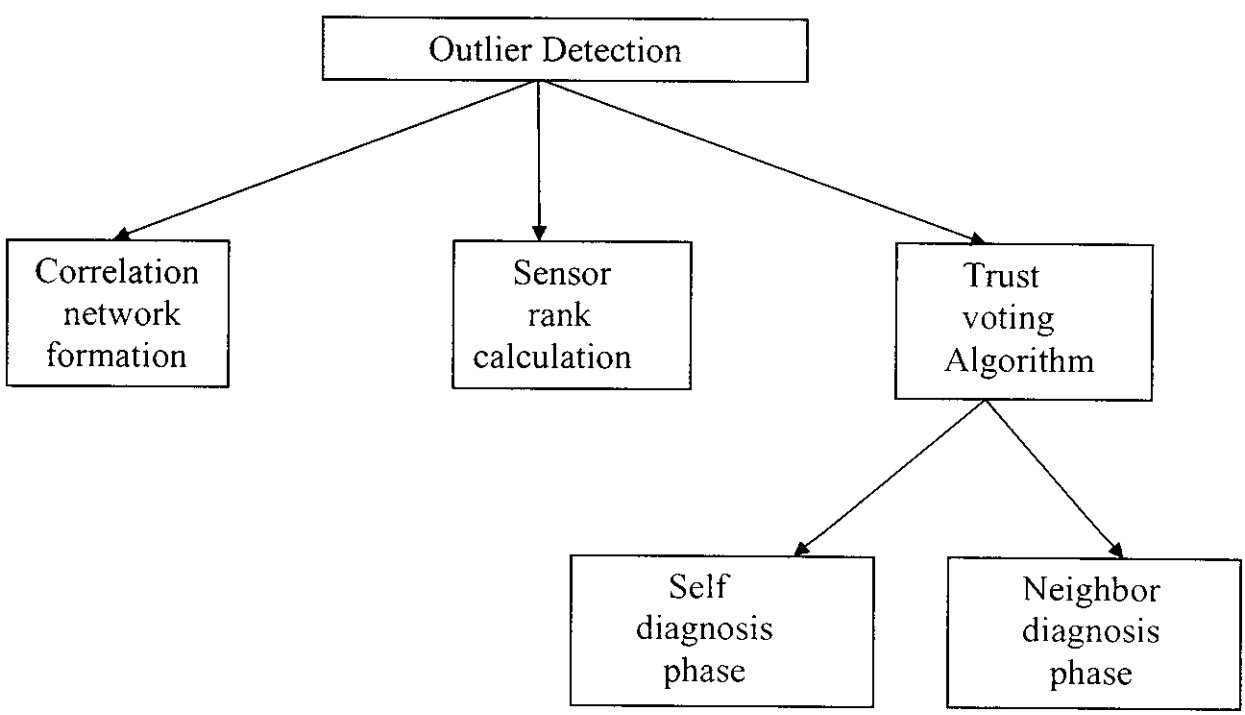
Farinaz Koushanfar [4]proposed Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks. Embedded sensor network is a system of nodes, each equipped with a certain amount of sensing, actuating, computation, communication, and storage resources. One of the key prerequisites for effective and efficient embedded sensor systems is development of low cost, low overhead, high resilient fault-tolerance techniques. Cost sensitivity implies that traditional double and triple redundancies are not adequate solutions for embedded sensor systems due to their high cost and high energy-consumption.

Mark D. Krasniewski[5] proposed Trust Index Based Fault Tolerance for Arbitrary Data Faults in Sensor Networks. Since sensor data gathering is the primary functionality of sensor networks, it is important to provide a fault tolerant method for reasoning about sensed events in the face of arbitrary failures of nodes sending in the event reports. In this paper, we propose a protocol called TIBFIT to diagnose and mask arbitrary node failures in an event-driven wireless sensor network. In our system model, sensor nodes are organized into clusters with rotating cluster heads. The nodes, including the cluster head, can fail in an arbitrary manner generating missed event reports, false reports, or wrong location reports. Correct nodes are also allowed to make occasional natural errors. Each node is assigned a trust index to indicate its track record in reporting past events correctly. The cluster head analyzes the event reports using the trust index and makes event decisions.

3.3 DETAILS OF METHODOLOGY

Sensors are prone to failure in harsh and unreliable environments. Faulty sensors are likely to report arbitrary readings which do not reflect the true state of environmental phenomenon or events under monitoring. Meanwhile, sensors may sometimes report noisy readings resulted from interferences. Both arbitrary and noisy readings are viewed as faulty readings. 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. So the following modules are proposed to identify the outliers in the wireless sensor networks.

The project has been split into four modules. They are,



Module Diagram

3.3.1 CORRELATION NETWORK FORMATION

Prior works only take the distance between sensor nodes into consideration when modeling the correlation of sensor readings. However, it is also possible that the readings of two geographically close sensor nodes to have dramatically different readings. Thus, it's critical to truly capture the correlation of sensor readings rather than their distance.

Reading Vector: Assume that the over- all readings of a sensor s_i consists of a series of readings in a sliding window .Clearly, the readings of a sensor within a sliding window is represented as a *reading vector*. Therefore, we can define the similarity of two sensor nodes in terms of their reading vectors. Since a faulty reading may be very different from other normal readings from the perspectives of trend and magnitudes, we employ the Extended Jaccard similarity as our similarity function. The Extended Jaccard similarity function for calculating the similarity of two sensors s_i and s_j is denoted as $corr_{i,j}$ and defined.

When the readings of two sensors have neither the similar trend nor the similar difference, the value of $corr_{i,j}$ is close to 0. On the other hand, the value will be set to 1 when the reading vectors of two sensors are exactly the same. Assume that the communication range of a sensor node is denoted as R and the geographical distance of two sensor nodes is represented as $dist(s_i, s_j)$.The correlation value is calculated by using the following formula.

$$corr_{i,j} = \frac{b_i(t) \cdot b_j(t)}{\|b_i(t)\|_2^2 + \|b_j(t)\|_2^2 - b_i(t) \cdot b_j(t)},$$

$$\text{where } \|b_i(t)\|_2^2 = x_i(t-\Delta t+1)^2 + \cdots + |x_i(t)|^2.$$

Correlation network: The correlation network is modeled as a graph $G = (V,E)$, where V represents the sensor nodes in the deployment region and $E = f(s_i, s_j)/s_i, s_j \in V$, $dist(s_i, s_j) < R$ and $corr_{i,j} > 0$. The weight of an edge (s_i, s_j) is assigned to be $corr_{i,j}$. Once the correlation network of sensors is constructed (and maintained), one can easily deduce the correlations among sensor nodes. Based on the correlation network, we shall further develop an algorithm to compute Sensor Rank for each sensor node, in terms of the correlation with its neighbors in network.

3.3.2 SENSOR RANK CALCULATION

Sensor Rank is to represent the trustworthiness of sensor nodes. If a sensor has a large number of neighbors with correlated readings, the opinion of this sensor is trustworthy and thus its vote deserves more weight. A sensor node with a lot of trustworthy neighbors is also trustworthy.

These ensure that

1) A sensor node which has a large number of similar neighbors to have a high rank .

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 Sensor Rank for each sensor to meet the above two requirements. We model the correlation network as a Markov chain M, where each sensor s_i is viewed as the state i, and the *transition probability* from state i (i.e., sensor s_i) to state j (i.e., sensor s_j) is denoted as $p_{i,j}$ and formulated as

$$\frac{corr_{i,j}}{\sum_{k \in nei(i)} corr_{i,k}}$$

Based on the above setting, we can formulate SensorRank of s_i, denoted as rank_i, as follows:

$$rank_i = \sum_{s_j \in nei(i)} rank_j \cdot p_{j,i}$$

where *nei*(*i*) is the witness set of node *i*.

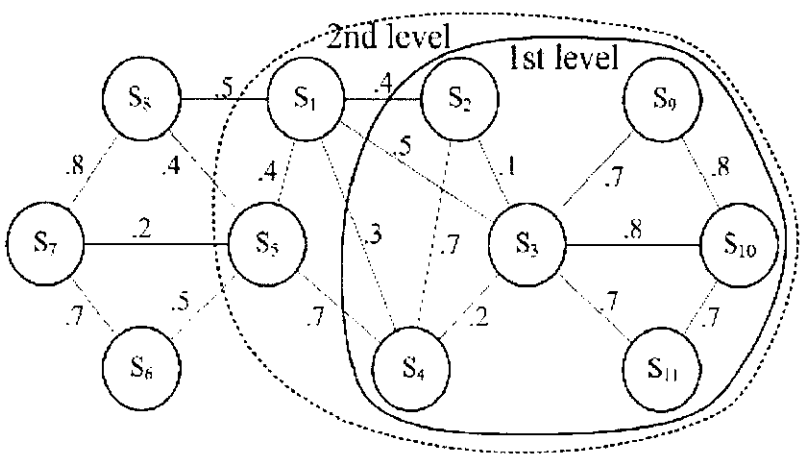


Fig 3 *Example of Sensor rank*

In the first round, each sensor node s_i updates its SensorRank 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 s_i has explored the *k*th level neighbors and up dated SensorRank as *rank*(*k*).

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: $rank_i^{(k)} = \sum_{s_j \in nei(i)} rank_j^{(k-1)} \cdot p_{j,i}$

Algorithm for sensor rank

3.3.3 TRUSTVOTING ALGORITHM

Here we describe our design of the Trust Voting algorithm, which consists of two phases:

1. Self diagnosis phase.
2. Neighbors diagnosis phase.

In the self-diagnosis phase, each sensor verifies whether the current reading of a sensor is unusual or not. Once the reading of a sensor goes through the self diagnosis phase, this sensor can directly report the reading. Otherwise, the sensor node consults with its neighbors to further validate whether the current reading is faulty or not. The sensor nodes generating faulty readings will not participate in voting since these sensors are likely to contaminate the voting result. Note that Trust Voting is an in-network algorithm which is executed in a distributed manner.

Input: a sensor s_i, Sensor Rank $rank_i$ 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 $rank_i$ to the neighbors

3: receive $\{rank_j | s_j \in nei(i)\}$ from the neighbors

/*set *timer* by the priority sorted by Sensor rank*/

4: sort Sensor Rank values received

5: x=$rank_i$'s order in the sorted Sensor rank values

6: n= neighbors of sensor s_i,

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 diagnosis*

12: return *faulty*

Algorithm for trust voting

3.3.3.1 SELF DIAGNOSIS PHASE

When a set of sensor nodes is queried, each sensor in the queried set performs a self-diagnosis procedure to verify whether its current reading vector is faulty or not. Once the reading vector of a sensor node is determined as normal, the sensor node does not need to enter the neighbor-diagnosis phase. To execute a self-diagnosis, each sensor s_i only maintains two reading vectors: i) the current reading vector at the current time t (denoted as b_i (t)); and ii) the last correct reading vector at a previous time tp (expressed by b_i (tp)). b_i (tp) records a series of readings occurred in the previous time and is used for checking whether the current reading

behavior is faulty or not. If these two reading vectors are not similar, b_i (t) is viewed as an unusual reading vector. Once a sensor node is detected an unusual reading vector, this sensor node will enter the neighbor-diagnosis phase to further decide whether the unusual reading behavior is faulty or not. Note that when b_i (t) is identified as a normal vector through the neighbor-diagnosis, bi (tp) is updated so as to reflect the current monitoring state.

3.3.3.2 NEIGHBOR-DIAGNOSIS PHASE

If a sensor node s_i sends b_i (t) to a neighbor s_j,s_j will compare b_i (t) with its own current reading vector b_j (t) and then give its vote with respect to b_i (t). From the votes from neighbors, s_i has to determine whether b_i (t) is faulty or not. Notice that some votes are from sensors with high SensorRank. A sensor node with high SensorRank has more similar neighbors to consult with and thus is more trustworthy. Therefore, the votes from the neighbors with high SensorRank are more authoritative, whereas the votes from the neighbors with low SensorRank should cast less weights. When sensor s_i sends b_i (t) to all its neighbors for the neighbor-diagnosis, each neighbor should return its vote after determining whether b_i (t) is faulty or not. If a neighbor s_j considers b_i (t) is not faulty by comparing the similarity of the two reading vectors (i.e., $corr_{i,j}$), s_j will send a positive vote, denoted $vote_j$ (i), to s_i. Otherwise, the vote will be negative. In addition, the vote from s_j will be weighted by its SensorRank.

$$vote_j\ (i) = \begin{cases} rank_j, & corr_{i,j} \geq \sigma \\ rank_j, & \text{otherwise.} \end{cases}$$

Otherwise, After collecting all the votes from the neighbors, s_i has two classes of votes: one is positive class (b_i (t) is normal) and the other is negative class (b_i (t) is faulty). If the weight of the former is larger than the weight of the later, the most neighbors will view b_i (t) as normal. Note that the weight of a vote represents how authoritative a vote is. It is possible that a neighbor s_j of s_i with a large SensorRank has a small correlation with s_i. In this case, these two sensor nodes may not provide good judgments for each other. Therefore, each vote (i.e., $vote_j$ (i)) has to be multiplied by the corresponding correlation, $corr_{i,j}$.

Thus, we use the following formula to determine whether the reading is faulty or not.

$$dec_i = \sum_{s_j \in nei(i)} corr_{i,j} \cdot vote_j(i)$$

If the weight of the positive votes is more than the weight of the negative votes, dec_i will be positive which means that s_i's reading is normal and the current reading can be reported. Otherwise, dec_i is negative, implying that the current reading of s_i is faulty.

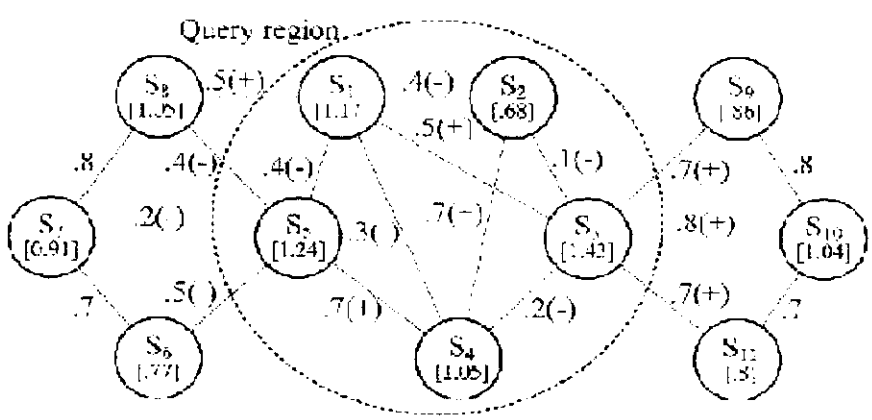


Fig 4 *An Example for trust voting*

For example in this figure, a region of sensors is queried (s_1, s_2, s_3, s_4 and s_5) and four faulty sensors (gray nodes) exist. SensorRanks of sensors are shown in square brackets in nodes and the correlation between sensors is shown on edges. To facilitate presentation of this example, the plus sign (minus sign) shows that two sensor nodes have similar (dissimilar) current readings and they are going to give the positive (negative) votes to each other when executing the neighbors' diagnosis. Consider sensor node s_5 as an example, where s_5 will receive the votes from its neighbors (i.e., s_1, s_4, s_6, s_7 and s_8). It can be obtained that dec_5 = (-1.17) . 0.4 + 1:05 – 0.7 + (-0.77) . 0.5 + (-0.91) . 0.2 + (-1.05). 0.4 = -0.72. Therefore, the reading reported by sensor node s_5 is a faulty reading.

Input: a sensor s_i, its current reading behavior $b_i(t)$, and a threshold σ.
Output: the variable *faulty*.

1: set dec_i=0
2: broadcast $b_i(t)$ to the neighbors
3: **for all** $s_j \in nei(i)$ **do**
4: **if** $sim(b_i(t),b_j(t)) \geq \sigma$ **then**
5: $vote_j(i)=rank_j$
6: **else**
7: $vote_j(i)= - rank_j$
8: $dec_i=dec_i+tr_{ij}.vote_j(i)$
9: **if** $dec_i \geq 0$ **then**
10: return *false*
11: **else**
12: return *true*

Algorithm for neighbor diagnosis phase

CONCLUSION

4. 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 outperforms majority voting and distance weighted voting, two state-of-the-art approaches for in-network faulty reading detection.

APPENDICES

5. APPENDICES

5.1 SOURCE CODE

```
clc
close all
clear all

no=input('Enter the number of nodes::');
%no=10;

  node= unifrnd(1,19.1,no,30);

  time=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25];

sl=input('Enter the time:::');
t1=[sl sl+1 sl+2  sl+3 sl+4];
for t=1:length(t1)

del_t=1;

%% Calculating b(t)

for i=1:no

   C=node(i,:);
    b(i,:)=[C(t-del_t+1),C(t-del_t+2),C(t-del_t+3),C(t-del_t+4),C(t-del_t+5),
    C(t- del_t+6),C(t-del_t+7),C(t-del_t+8),C(t-del_t+9),C(t) ];

end

%% Calculating b(t)^2

for i=1:no

  C=node(i,:);
  b_1(i,:)=[C(t-del_t+1)^2,C(t-del_t+2)^2,C(t-del_t+3)^2,C(t-del_t+4)^2,
 C(t- del_t+5)^2,C(t-del_t+6)^2,C(t-del_t+7)^2,C(t-del_t+8)^2,
  C(t-   del_t+9)^2,C(t)^2 ];
```

```
end

%% Correlation
for i=1:no

 for j=1:no
    num=b(i,:)*b(j,:)';
    den=sum((b_1(i,:)+b_1(j,:)))-(num);
    corr(i,j)=num/den;
    corr(i,j)=(corr(i,j));

  end
 end
% figure
%
%     plot(corr,'rd')
%     axis([1 8 0 1.5])
disp('Correlation Value Is')
disp(corr)

%% Corr Network..
Corr=round(corr);
disp('Correlation Network Is')

disp(Corr)
disp('_____')

%%
level=0;
for i=1:no
   for j=1:no

      if Corr(i,j)<=0
         level=level+1;
         disp(['Sensor in Second Level Is ' num2str(i)])

      end

   end
```

```
end
if level<1
        disp('All The sensors are in First Level')
end
disp('_____')

%% Sensor Ranking

the=3;
for i=1:no
     ran=1;
      for K=1:the
         for j=1:no
              for k=1:no
             de(k)=corr(i,k);
             end
        nume=corr(i,j);
        deno=sum(de)-nume;
         P(:,j)=nume/deno;
          end
    rank_sen(i,K)=sum(P);
     end
disp([ ' Rank Of sensor ' num2str(i) ' is '])
disp(rank_sen(i))
end
disp('_____')
end

%% Trust voting
K=0;
for i=1:no
   for t=1:length(t1);
      fault=0;
         new_rank(:,i)=rank_sen(i);
      for  j=1:no


      new_rank(:,j)=rank_sen(j);    %% step 3

      end
```

```
%    disp('Sorted Rank')
   sort_rank=sort(new_rank,'ascend');      %%step 4
%    disp(sort_rank)

   for order=1:length(sort_rank)           %% step 5
     or=sort_rank(order);

     if or==new_rank(:,i)
        X=order;
%          disp('The value Of X is')
%          disp(X)
     end

   end

   n=no-1;
    rank_timer=X*(t/(n+1));
   %while t==rank_timer

     %% Self diagnosis
        if t==1
           bp=b(i,t);

            dec=0;
        else


           if K>no
              K=no;
           end
           if b(i,t)==b(i,K)

           else
              %% Neighbour
              dec(i)=0;

              %while nei_j<=5~=i
              for nei_j=1:no

                 sim=b(i,t)-b(nei_j,t);
```

```
                if sim>=1

                   vote(i)=rank_sen(nei_j);
                else
                    vote(i)=-(rank_sen(nei_j));

                end

                if nei_j==i
                   vote(i)=0;
                end
                dec(i)=dec(i)+(vote(i));        %corr(i,nei_j)

                j=j+1;
              end
            end
          end


          if dec>=0
             disp(['Sensor  ', num2str(i) ,' shows the usual value at time
',num2str(t1(t))])
             K=K+1;
          else
             disp(['Sensor  ', num2str(i) ,' shows faulty reading at time
',num2str(t1(t))])
          end
     end
    disp('_____________________________________________')
end
```

5.2 SCREEN SHOTS

5.2.1 NODE VALUE

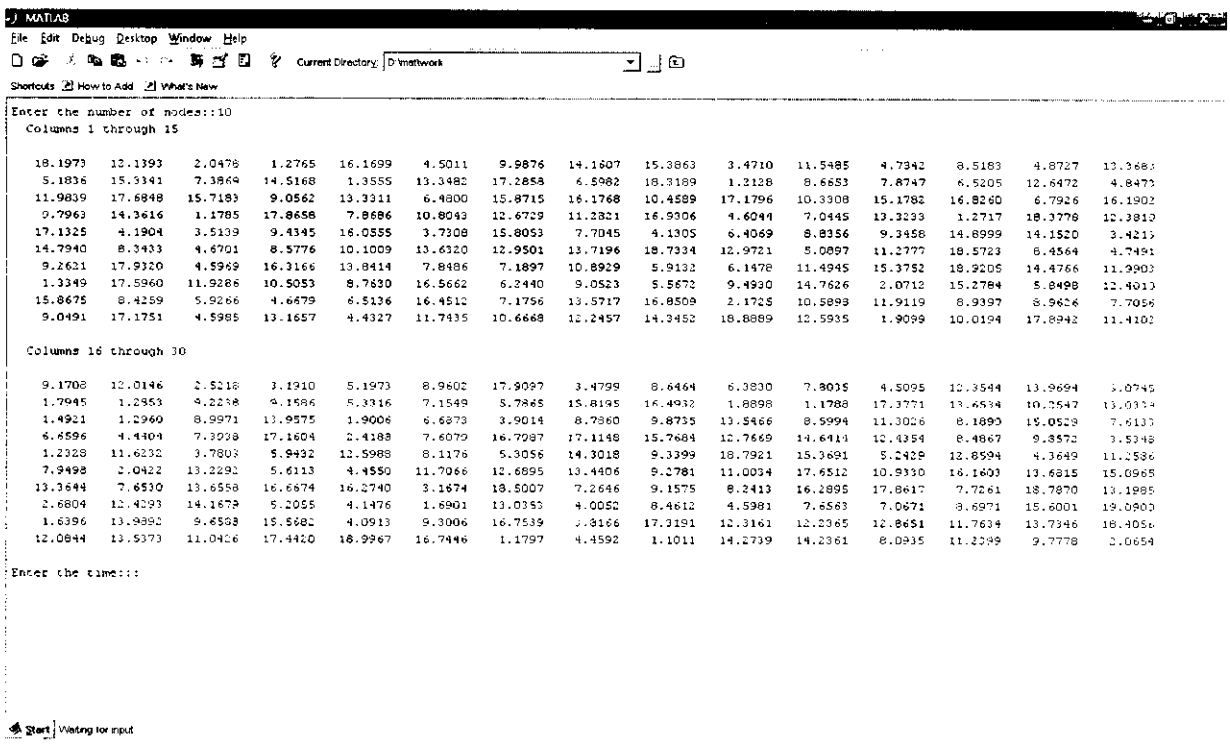


Fig 5 *Randomly generated sensor values*

5.2.2 CORRELATION VALUE

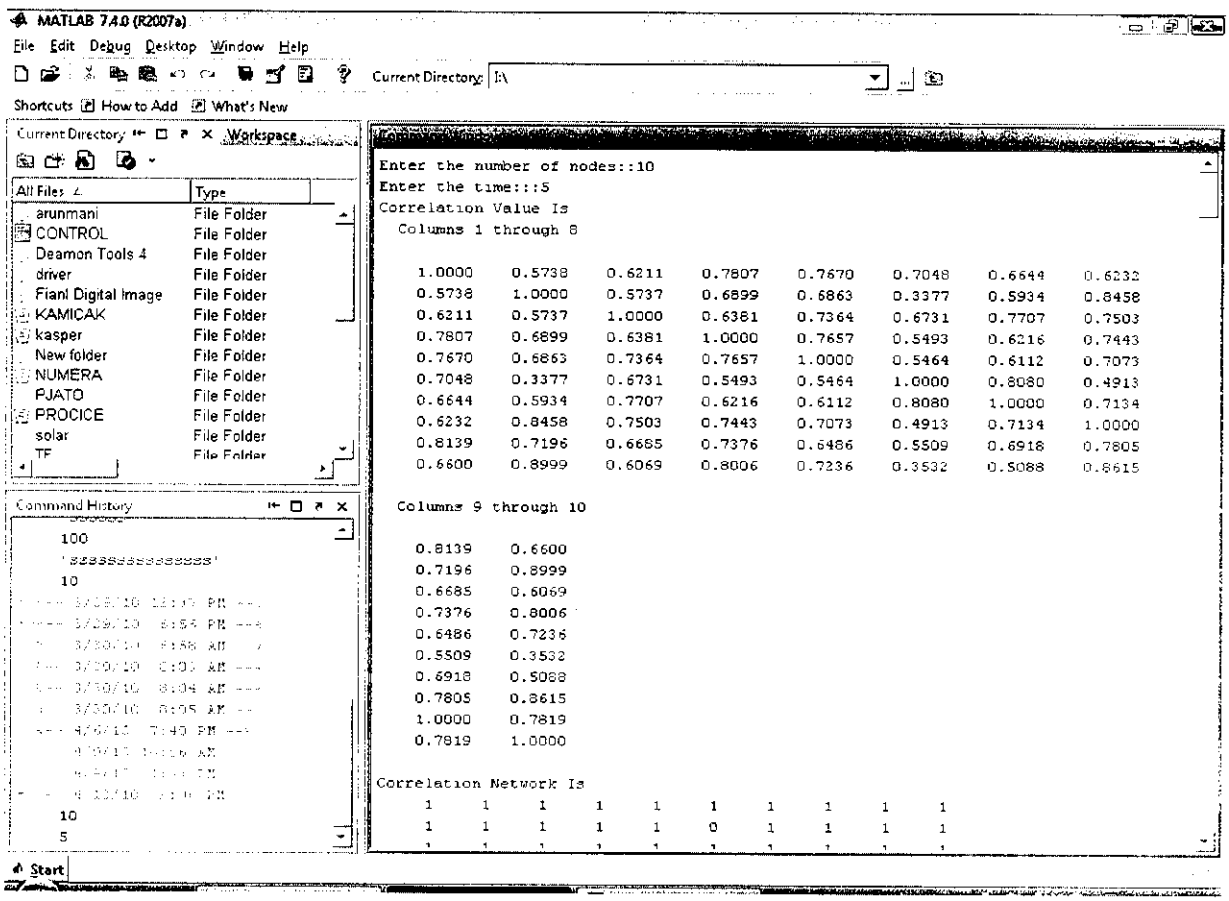


Fig 6 *Correlation values for the sensor nodes*

5.2.3 CORRELATION MATRIX

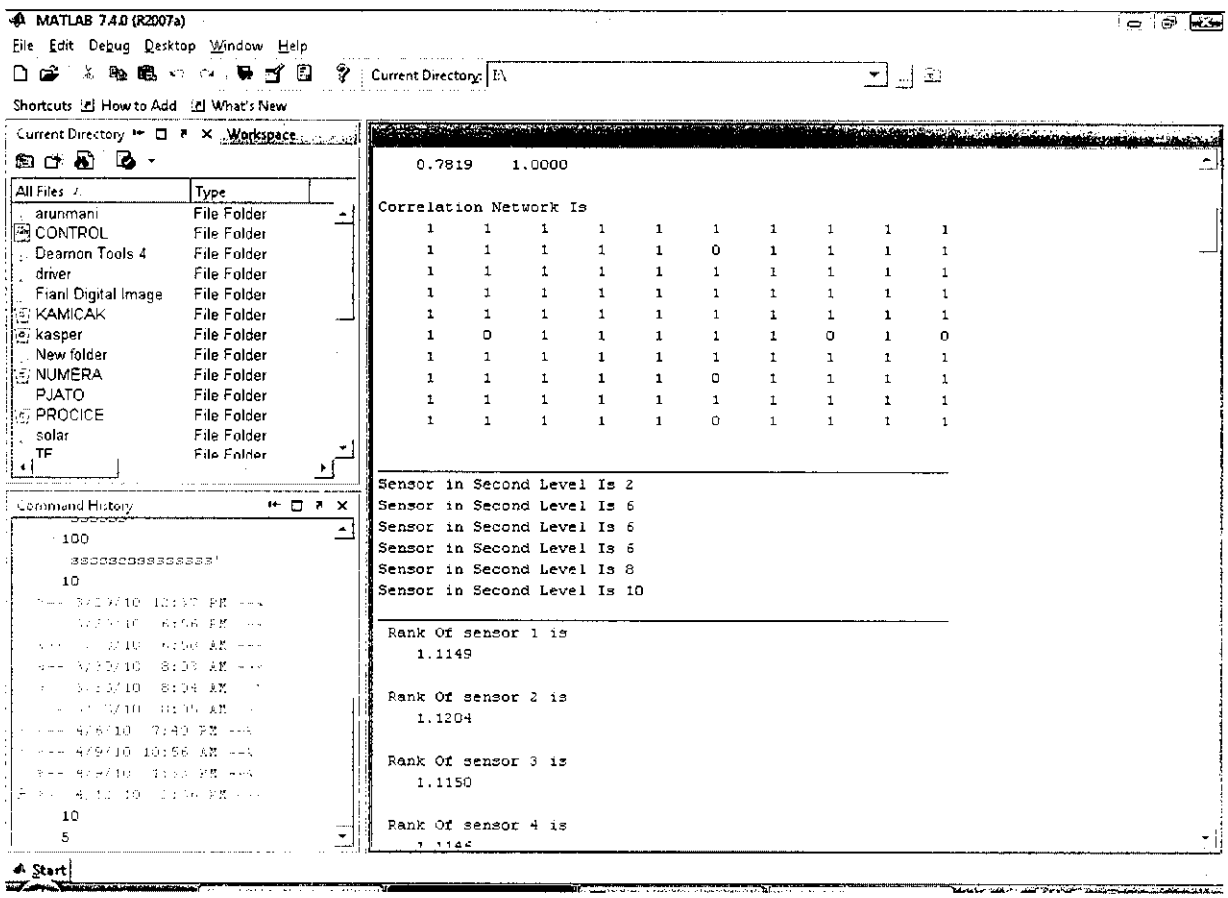


Fig 7 *Correlation network for the sensor nodes*

5.2.4 SENSOR RANK

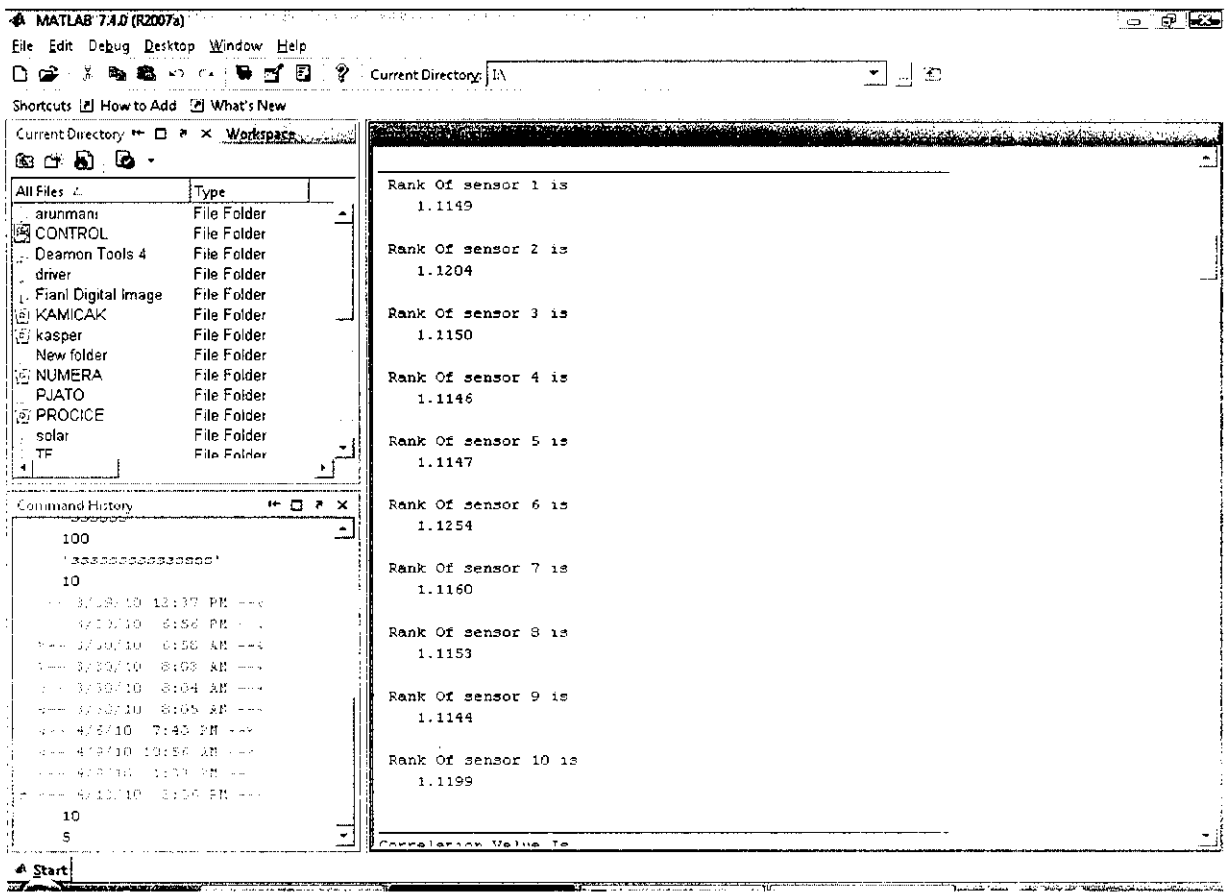


Fig 8 *Sensor rank for the nodes*

5.2.5 TRUSTVOTING ALGORITHM

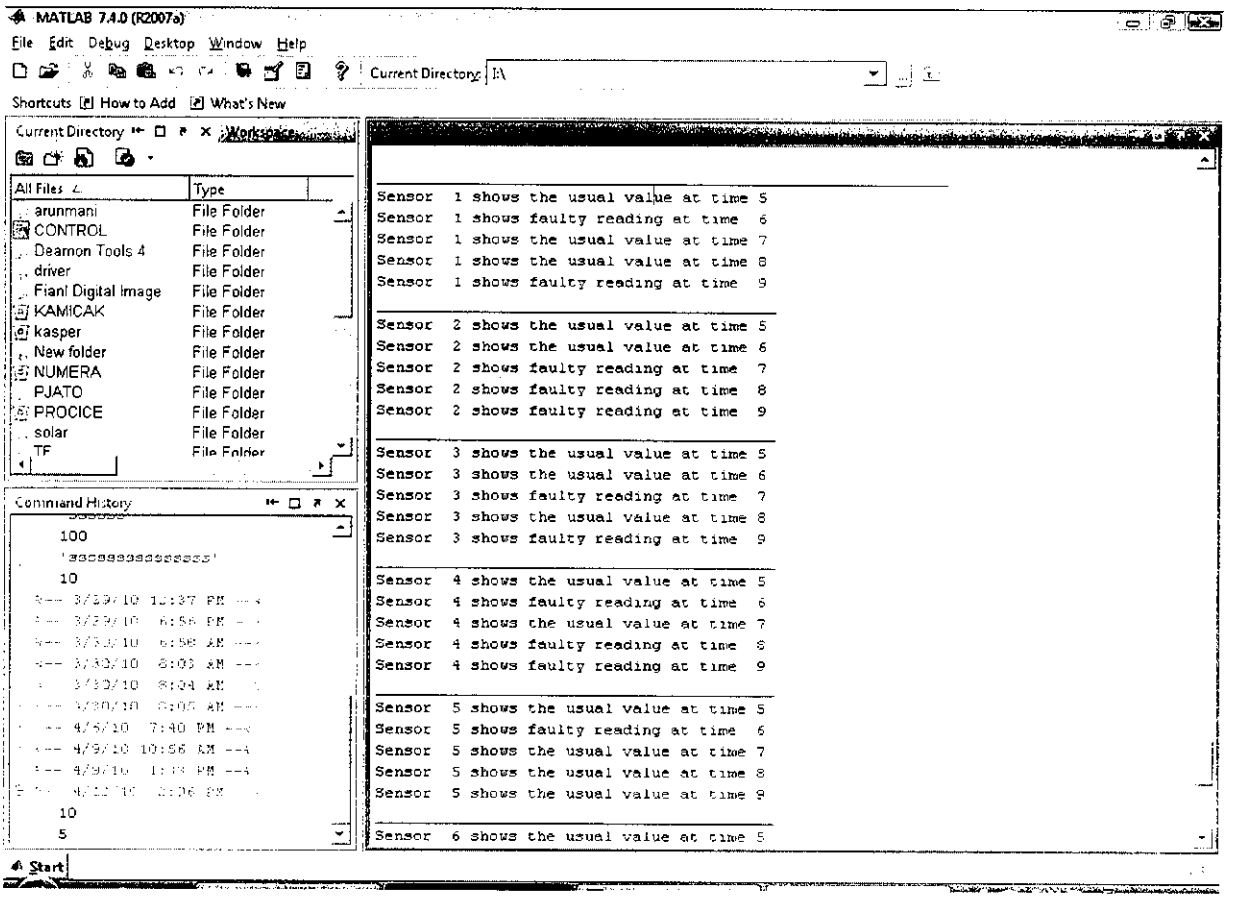


Fig 9 *Detecting faulty reading using trust voting algorithm*

REFERENCES

6. REFERENCES

[1] T. Clouqueur, K. K. Saluja, and P. Ramanathan. Fault tolerance in collaborative sensor networks for target detection. IEEE Transactions on Computers, 53(3).320{333, 2004.

[2] Min Ding, Dechang Chen, Kai Xian, and Xiuzhen Cheng. Localized fault-tolerant event boundary detection in sensor networks. In Proc. of IEEE INFOCOM, March 2005.

[3] Eiman Elnahrawy and Badri Nath. Online data cleaning in wireless sensor networks. In Proc. Of International Conference on Embedded Networked Sensor Systems(SenSys), pages 294{295, 2003.

[4] Farinaz Koushanfar, Miodrag Potkonjak, and Alberto Sangiovanni-Vincentelli. Fault tolerance in wireless ad-hoc sensor networks. In Proc. of IEEE International Conference on Sensors, June 2002.

[5] Mark D. Krasniewski, Padma Varadharajan, Bryan Rabeler, Saurabh Bagchi, and Y. Charlie Hu. Tibfit, Trust index based fault tolerance for arbitrary data faults in sensor networks. In Proc. of International Conference on Dependable Systems and Networks,pages 672{681, 2005.

WEBSITES

www.wikipedia.org