퍼지 및 지능시스템학회 논문지 2005, Vol. 15, No. 5, pp. 615-620

Application of Consensus Algorithm to Mate' for Identifying Faulty Sensor Node in Sensor Networks

Sung-Ho Kim*, Hyeong-Joo Kim**, Yun-Jong Han*, Diaconescu Bogdana*

*School of Electronics and Information Engineering, Kunsan National University

** School of Civil and Environmental Engineering, Kunsan National University

Abstract

Sensor networks are usually composed of tens or thousands of tiny devices with limited resources. Because of their limited resources, there will often be some faulty nodes within the network. As nodes in some certain regions rely on each other to route the information gathered by different sensors to a base station (sink), the network should be able to detect a non-operational node in order to determine new paths for routing the information. Failure detection, which identifies the faulty nodes, is rather necessary in sensor networks and a very important research issue. The detection of a non-operational node can be performed using Consensus Algorithm with the purpose of achieving agreement about a node which is supposed to be faulty (non-operational). In this work, we discuss the application of a Consensus Algorithm to sensor node called "mote". Our experimental results show that it works efficiently for identifying faulty nodes in sensor networks.

Key Words : Sensor networks, detection of faulty nodes, Consensus Algorithm.

1. Introduction

Recent development in wireless communication and micro-electro-mechanical systems technology have enabled the deployment of sensor networks consisting of small, low-cost, low-power and multi-functional sensor nodes with sensing, computation and communication capabilities. These features make sensor networks suited for a wide range of applications: environmental and habitat monitoring, seismic detection, military surveillance, etc.

Recently, a tiny sensor node called "mote" was developed for sensor networks by the Computer Science Division of University of California, Berkeley. It has several platforms depending on the types of microprocessors. TinyOS which was developed for sensor network is installed on mote. TinyOS provides high parallelism and efficiency through a somewhat tricky programming interface. This interface is badly suited to non-expert programmers, such as the biologists and civil engineers. A simpler programming model, which allows novice programmers to express their desired behavior without worrying about timing and asynchrony, would greatly improve the usefulness of mote. To achieve this goal, Mate', which is a byte-code interpreter running on TinyOS, was developed. Mate' is a single TinyOS component that sits on top of several system components, including sensors and the network stack.

In the past years, many researchers in the field of sensor network have focused their attention on data aggregation, routing protocols and energy-efficient MAC protocol [1-2]. However, there is room for many other important research issues to be addressed. Generally, sensor nodes are more likely to have problems or die out due to many different reasons: their batteries may be depleted or they may be accidentally destroyed. These faulty nodes can cause many problems such as deterioration of sensor network performance. Therefore, it is critical for the base station (sink) to find out whether a node is operational or not and more precisely, which node is non-operational. Many researchers have proposed different approaches for identifying faulty nodes. One of these approaches advocates the Consensus Algorithm. It requires each node to maintain its own view regarding the status of the other nodes and also monitor the suspicions of all other fault-free nodes[3].

In this work, to improve the functionality of TinyOS-based sensor node called "mote", a simple Consensus Algorithm for identifying faulty nodes is implemented on Mate'. The paper is organized as follows: Section 2 describes a tiny virtual machine for sensor networks Section 3 describes the concept of Consensus Algorithm and Section 4 describes the experimental application of Consensus Algorithm on Mate'. Finally, Section 5 shows results.

접수일자 : 2005년 6월 2일
완료일자 : 2005년 8월 8일
This work was supported by grant No. R01-2005-000-11088-0. from the Basic Research Program of the Korea Science & Engineering Foundation.

2. Virtual Machine for Mote

TinyOS for sensor networks provides high parallelism and efficiency through a somewhat tricky programming interface, badly suited to non-expert programmers. Furthermore, as wireless sensor networks must be re-programmable in response to changing needs (simple parameter adjustments, the upload of complete binary images), a concise way to represent a wide range of programs is needed. For this, Mate' which allows novice programmers to express their desired behavior without worrying about timing and asynchrony was developed to improve the usefulness of mote.

2.1 Hardware platform for Mate'

Many hardware platforms are currently available for designing of sensor nodes . Motes hardware platforms which use TinyOS as an operating system were developed by the Computer Science Division from University of California, Berkeley. In this work, mote based on ATMega 128 microprocessor was developed for our experiment as in fig.1. ATMega128 is a low-power micro-controller which runs TinyOS from its internal flash memory. Using TinyOS, a single processor board can be configured to run a certain sensor application/processing and the network/radio communication stack simultaneously. Our board supports analog input port, digital I/O and UART interfaces. These interfaces make it easy to connect to a wide variety of external peripherals.

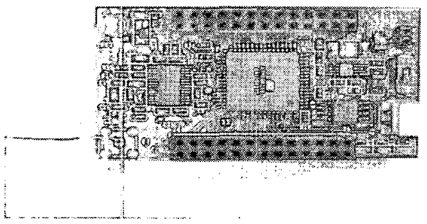


Fig.1. Sensor node based on ATMega 128 microprocessor

The developed sensor node supports 40Kbit communication on its radio. The radio can be put into 10 Kbit mode for backwards compatibility. All mote platforms are Harvard architectures with separate instruction and data memory. Installing new binary code requires a reset to take effect. To change the behaviour of a TinyOS program, one must either hard code (revise) a state transition in a program (upon receiving a type of packet, start reading light instead of temperature), or one must modify source code, recompile a TinyOS image and place the entire new image on a mote.

2.2 Mate'

Mate' which is a tiny communication-centric virtual machine was developed. It is a byte-code interpreter that runs on TinyOS and that can be thought of as a single TinyOS component that sits on top of several system components, including sensors, the network stack and a non-volatile storage. Code is broken in capsules of 24 instructions, each of which is a single byte long (this limit allows a capsule to fit into a single TinyOS packet); larger programs can be composed of multiple capsules. In addition to byte codes, capsules contain identifying and version information. It has a stack-based architecture which allows a concise instruction set; most instructions do not have to specify operands, as they exist on the operand stack. There are three classes of Mate instructions: basic, s-class and x-class. Mate has two stacks: an operand stack and a return address stack as shown in figure 2. The operand stack has a maximum depth of 16 while the call stack has a maximum depth of 8. Most instructions operate solely on the operand stack, but a few instructions control program flow and several have embedded operands. There are three execution contexts that can run concurrently. Each context has its own stack: the operand stack and the return address stack. The operand stack is used for all instructions handling data and the return address stack is used for subroutine calls. There are three operand types: values, sensor readings and messages. Some instructions can only operate on certain types[6]. However, many instructions are polymorphic. For example, some instructions can be used to add any combination of the types, with different results. Mate capsules can forward themselves through a network with a single instruction. A capsule sent in a packet contains a type and a version number. If Mate receive a more recent version capsule, it installs it, otherwise the capsule is discarded. After installing the capsule, this is transmitted to other motes using the forw instruction. Mate provides both a built-in ad-hoc routing algorithm (the send instruction) as well as mechanisms for writing new ones (the send instruction). Mate and all its subcomponents must fit in 1 KB of RAM and 16 KB of instruction memory.

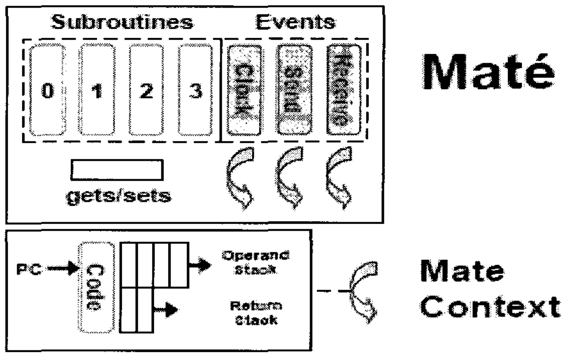


Fig.2. Mate' architecture and execution model: Capsules, Contexts and Stacks

3. Consensus algorithm in sensor network

Sensor nodes are densely deployed in the field in various ways. The nodes are linked by a wireless medium, using a multi-hop routing protocol to communicate with each other. Generally, sensor nodes are prone to failures due to their limited resources or environmental influence. The failure of sensor nodes should not affect the overall task of the sensor network. In order to get consistent failure detection, it is necessary for all nodes to come to an agreement on the status of a faulty node through discovery and notification. Formally stated, the Consensus Algorithm is defined as the process by which agreement is reached among the fault-free sensor nodes in order to maintain the performance and integrity of the system. This section describes an algorithm in which consensus is reached when each fault-free node in the network realizes that all the fault-free nodes have detected a faulty node.

The Consensus Algorithm requires each node to not only maintain its own view regarding the status of the other nodes and also monitors the suspicions of all other fault-free nodes. To achieve these goals, each node in a network of n nodes maintains an n x n suspect matrix S and a fault vector F. The element S[i, j] is set to 1 if node i suspects node j of being faulty. Otherwise it is set to 0. Consensus is reached when a given column of S contains a 1 for all i corresponding to a fault-free node. The fault-free nodes are monitored maintaining a fault vector F at each node. Element j of F is set to 1 only if all the other fault-free nodes suspect it to be faulty. The logical OR operation between S and F is performed and consensus is reached when the result of OR operation produce a bit array in which all the elements are 1. For simplification, the principle of Consensus Algorithm is explained for a small 4-node network as in fig. 3.

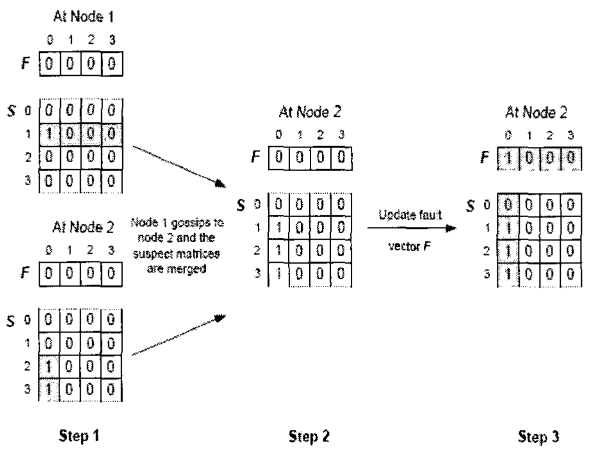


Fig.3. The Principle of Consensus Algorithm for a 4-node network

Step1: the suspect matrix of node 1 shows that it suspects node 0 to be faulty (non-operational). Node 2 also suspects node 0 to be faulty (non-operational) and it received a suspect matrix from node 3 with the same supposition.

Step 2: node 1 send its suspect matrix to node 2 which performs a logical OR operation, merges the two matrices. At this moment node 2 "notice" that all fault-free nodes suspect node 0 to be faulty.

Step 3: node 2 detects that consensus has been reached since all fault-free nodes detect the failure of node 0. The fault vector F is updated; the value corresponding to node 0 is set to 1.

Next step, performed by node 2, is to multicast a message (indicating that consensus has been reached) to nodes 1 and 3 and performs operations required for the network reconfiguration (looking for new paths of routing the information).

The detailed procedure for consensus algorithm is shown in fig. 4.

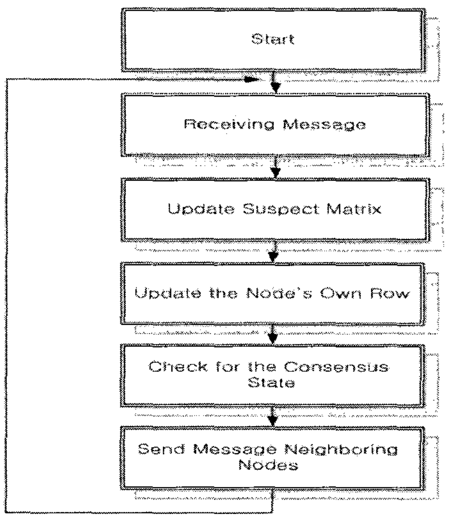

Start
Receiving Message
Update Suspect Matrix
Update the Node's Own Row
Check for the Consensus State
Send Message Neighboring Nodes

Fig. 4. Flow chart of Consensus algorithm

4. Application of Consensus Algorithm to Mate'

This section wishes to describe the implementation of Consensus Algorithm to Mate' in order to identify faulty nodes in sensor networks. The top-level TinyOS packet abstraction is an Active Message (AM). The characteristics of AM are very important because they define the capabilities of systems built on top of it. AM can be seen as an asynchronous communication mechanism intended to expose the full hardware flexibility and performance of modern interconnection networks. AM model was originally designed for a large-scale multi-processors, but the limited power, memory and computational capacities of the sensor mote make it appropriate for TinyOS. The TinyOS networking stack handles media access control and single hop communication between motes. Higher layer protocols (e.g. network or transport) are built on top of the AM interface. AM packets can be sent to a specific mote (addressed with a 16 bit ID) or to a broadcast address (0xffff). TinyOS provides a name space for up to 256 types of Active Messages, each of which can be associated with a different software handler. The underlying idea is simple: each message contains the address of a user-level handler as header. The role of the handler is to get the message out of the network and enter the computation running on the processing node [7]. AM types allow multiple network or data protocols to operate concurrently without conflict. The AM layer also provides the abstraction of an 8-bit AM group; this allows logically separate sensor networks to be physically co-present but mutually invisible, even if they run the same application. AM, as asynchronous communication mechanism, exposes the hardware flexibility. In TinyOS, the original AM packet is TOS_Msg and all the other message types are encapsulated in it.

The structure of the TOS_Msg is shown in fig. 5.

```
typedef struct TOS_Msg
   {
        uint16_t addr;
        uint8_t type;
        uint8_t group;
        uint8_t length
        int8_t data[TOSH_DATA_LENGTH]
        uint16_t crc;
        uint16_t strength;
        uint8_t ack
        uint16_t time
        } TOS_Msg
```

Fig. 5. Structure of TOS message

The message is "active" because it contains destination address, group ID and type.

The **addr** field specifies the destination address (a mote ID or the broadcast address).

The **group** field specifies a channel for motes on a network (if a mote receives a packet with a different group ID, the packet is discarded).

The **type** field specifies which handler to be called at the AM level when the packet is received.

The **length** field specifies the length of the data portion of the TOS_Msg. Packets have a maximum payload of 29 bytes.

The **data** field consists of an array of 29 bytes (as specified by TOSH_DATA_LENGTH). The last 2 bytes are assigned to CRC field. When sending a packet, the check-sum is incrementally calculated as each byte of the packet is transmitted. The last three fields of TOS_Msg are the unsigned 2-byte **strength**, the single unsigned byte **ack** field and unsigned 2-byte **time** fields. These three fields are meta-data about the packet and are not transmitted. The **ack** is sent by receiver and set by sender. This is the mechanism that can provide reliability in the stack. The **strength** field is currently unused, and the **time** field stores an atomic capture of a 16-bit 4MHz counter. The maximum length of a transmitted TOS_Msg is 36 bytes[8].

In this work, implementation of Consensus Algorithm on Mate' was executed by utilizing the AM packet. To make possible the dissemination of the suspect matrix and fault vector to every nodes, they are inserted in the data field in AM. As shown in fig. 6, the data field is filled with the suspect message packet (message type, ID number of the sender node and the suspect matrix, S) and the fault message packet (message type, ID number of the sender node and the fault vector, F). For a 4-node sensor network, the suspect matrix S is 16-byte long and the fault vector F is 4-byte long.

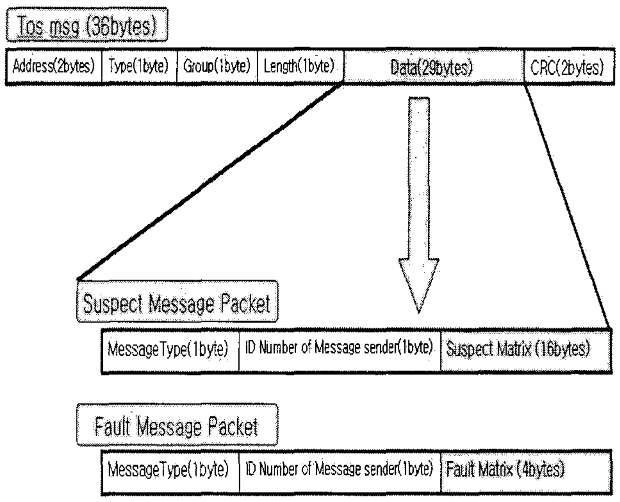


Fig. 6. The structure of AM packet for a 4-node sensor network

```
<SEARCH PATH="../opcodes">
<SEARCH PATH="../contexts">
<SEARCH PATH="../languages">
<SEARCH PATH="../../../../contrib/xbow/tos/lib/ReliableRoute">
<SEARCH PATH="../extensions">
<LOAD FILE="../sensorboards/micasb.vmsf">
<LANGUAGE NAME="tscript">
<FUNCTION NAME="led">
<FUNCTION NAME="send">
<FUNCTION NAME="light">
<FUNCTION NAME="temp">
<FUNCTION NAME="bclear">
<FUNCTION NAME="bufsorta">
<FUNCTION NAME="bufsortd">
<FUNCTION NAME="int">
<FUNCTION NAME="id">
<FUNCTION NAME="son">
<FUNCTION NAME="soff">
<FUNCTION NAME="uart">
<FUNCTION NAME="rand">
<CONTEXT NAME="Once">
<CONTEXT NAME="SendCounter">
<CONTEXT NAME="Timer1">
```

Fig. 7. The source code of a newly added context

In order to verify the feasibility of the Consensus Algorithm on Mate', some experiments for a 4-node sensor network are executed. A Java Virtual machine with the SendCounter context (handler) is designed for monitoring the Consensus Algorithm[9-10]. Fig. 7 shows the source code of a newly added context and fig. 8 shows a newly designed Java Applet to embody a new event handler.

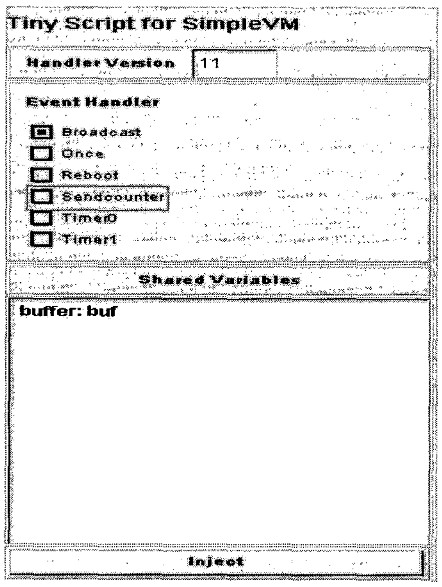


Fig. 8. Newly designed Java applet for new event handler

Fig. 9 shows the test system which is composed of 4 sensor nodes working without any faulty sensor node.

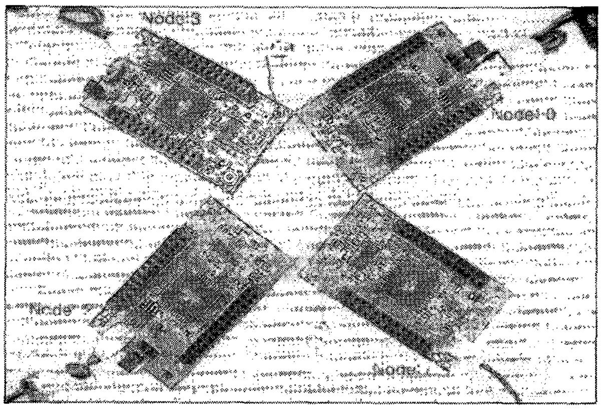
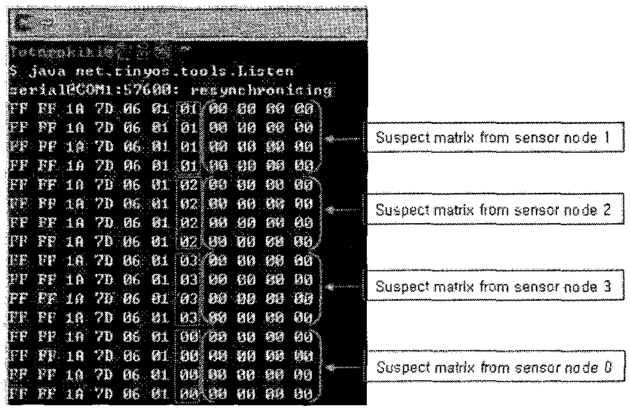


Fig. 9. The test system composed of 4 sensor nodes in a fault-free state

Fig. 10 shows the corresponding suspect matrix from each nodes monitored by sink node. The value of 0 in matrix represents the fault-free state of the network.



```
[illegible] ~
$ java net.tinyos.tools.Listen
serial@COM1:57600: resynchronising
FF FF 1A 7D 06 01 01 00 00 00 00
FF FF 1A 7D 06 01 01 00 00 00 00
FF FF 1A 7D 06 01 01 00 00 00 00
FF FF 1A 7D 06 01 01 00 00 00 00
FF FF 1A 7D 06 01 02 00 00 00 00
FF FF 1A 7D 06 01 02 00 00 00 00
FF FF 1A 7D 06 01 02 00 00 00 00
FF FF 1A 7D 06 01 02 00 00 00 00
FF FF 1A 7D 06 01 03 00 00 00 00
FF FF 1A 7D 06 01 03 00 00 00 00
FF FF 1A 7D 06 01 03 00 00 00 00
FF FF 1A 7D 06 01 03 00 00 00 00
FF FF 1A 7D 06 01 00 00 00 00 00
FF FF 1A 7D 06 01 00 00 00 00 00
FF FF 1A 7D 06 01 00 00 00 00 00
FF FF 1A 7D 06 01 00 00 00 00 00
```

Suspect matrix from sensor node 1
Suspect matrix from sensor node 2
Suspect matrix from sensor node 3
Suspect matrix from sensor node 0

Fig. 10. Monitored suspect message packet in case of fault-free state

To verify the feasibility of proposed scheme, node 1 gets faulty on purpose, as one can notice in fig. 11.

In this case, the received suspect message packets can be monitored as in fig. 12. The underlined elements of message packet show that the node 1 is faulty.

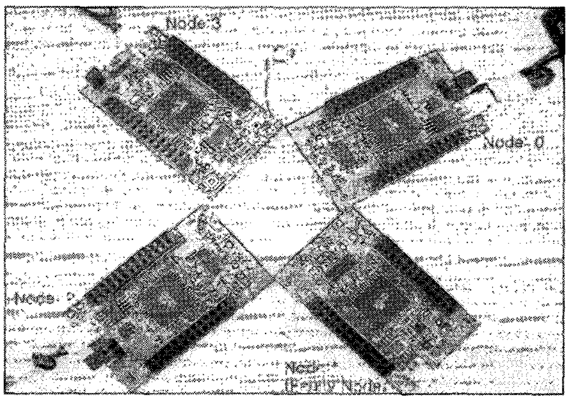
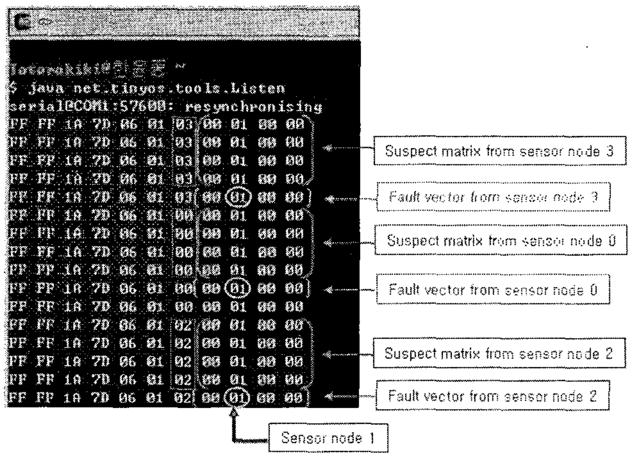


Fig. 11. The test system composed of 4 sensor nodes in a faulty state (node 1 is faulty)



```
[illegible] ~
$ java net.tinyos.tools.Listen
serial@COM1:57600: resynchronising
FF FF 1A 7D 06 01 03 00 01 00 00
FF FF 1A 7D 06 01 03 00 01 00 00
FF FF 1A 7D 06 01 03 00 01 00 00
FF FF 1A 7D 06 01 03 00 01 00 00
FF FF 1A 7D 06 01 03 00 01 00 00
FF FF 1A 7D 06 01 00 00 01 00 00
FF FF 1A 7D 06 01 00 00 01 00 00
FF FF 1A 7D 06 01 00 00 01 00 00
FF FF 1A 7D 06 01 00 00 01 00 00
FF FF 1A 7D 06 01 00 00 01 00 00
FF FF 1A 7D 06 01 00 00 01 00 00
FF FF 1A 7D 06 01 02 00 01 00 00
FF FF 1A 7D 06 01 02 00 01 00 00
FF FF 1A 7D 06 01 02 00 01 00 00
FF FF 1A 7D 06 01 02 00 01 00 00
FF FF 1A 7D 06 01 02 00 01 00 00
```

Suspect matrix from sensor node 3
Fault vector from sensor node 3
Suspect matrix from sensor node 0
Fault vector from sensor node 0
Suspect matrix from sensor node 2
Fault vector from sensor node 2
Sensor node 1

Fig. 12. Monitored suspect message packet in case of faulty case.

5. Conclusion

In this work, we have examined the applicability of Consensus Algorithm to motes. To verify the faulty sensor detection ability of the proposed scheme, several tests were conducted utilizing a simple 4-node network.

The result shows that Consensus Algorithm can be successfully applied to pinpoint which sensor node is faulty just by monitoring the suspect matrix and fault vector. However, in this work we considered the simple 4-node network. Generally, it will take more time to reach consensus in case of larger sensor networks. Therefore, a further study on reducing the time for reaching consensus will be required.

Reference

[1] B.Krishnamachari, D. Estrin and S. Wicker, "Modelling Data-Centric Routing in Wireless Sensor Networks," IEEE INFOCOM'02, June 2002

[2] W.Ye, J. Heidemann and D.Estrin "An Energy-Efficient MAC Protocol for Wireless Sensor Networks" IEEE INFOCOM'02, June 2002

[3] S. Ranganathan, A.D. George, R.W. Todd, Matthew C.Chidester, "Gossip-Style Failure Detection and Distributed Consensus for Scalable Heterogeneous Clusters", HCS Research Laboratory, 2000

[4] M. Barborak, A. Dahbura, M. Malek, "The Consensus problem in Fault-Tolerant Computing, ACM Computing Surveys", Vol 25, No 2, 171-220, 1993.

[5] Ian F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, Erdal Cayirci, "A Survey on Sensor Networks", IEEE Communications Magazine, August 2002, pages 102-114

[6] P. Levis, D. Culler, "Mate' : A Tiny Virtual Machine for Sensor Networks", Computer Science Division and Intel Research, University of California, Berkeley

[7] T.von Eichen, D. Culler, S. C. Goldstein, K. E. Schauser, "Active Messages: a Mechanism for Integrated Communication and Computation", 19th International Symposium on Computer Architecture, 1992

[8] N. Lee, P. Levis, J. Hill, "Mica High Speed Radio Stack", Sept. 11, 2002

[9] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler and Kristofer Pister. System architecture directions for networked sensors[C]. proc. 9th International Conference on Architectural support for Programming Languages and Operating Systems, pages 93-100t, Cambridge, MA, USA, November 2000

[10] Gay David, Welsh Matt, Levis Philip, Brewer Eric,Von Benren, Robert, Culler David. the nesC language: A holistic approach to networked embedded systems[C]. proc. ACM SIGPLAN Conference on Programming Language Design and Implementation.

저 자 소 개

김성호(Sung-Ho Kim)
1984년: 고려대학교 공과대학 학사
1986년: 고려대학교 대학원 석사
1991년: 고려대학교 대학원 박사
1995-1996년: JAPAN HIROSHIMA UNIVERSITY, POST-DOC.
1997-현재: 군산대학교 전자정보공학부 교수

관심분야: 고장진단, 공장 자동화, Sensor Network

김형주(Hyeong-Joo Kim)
1983년: 고려대학교 공과대학 학사
1985년: 고려대학교 대학원 석사
1990년: 일본히로시마대학교 박사
1990-91년:일본 Fuken Eng(주)책임연구원
1991-92년: 동아 Eng(주) 기술개발 실장
1993-현재: 전라북도 전주시, 군산시 주택공사 기술자문위원
1993-현재: 군산대학교 토목환경공학부 교수

관심분야: 지반 및 기초공학, 준설 매립공학, 연약 지반

한윤종(Yoon-Jong Han)
2001년: 군산대학교 제어계측공학과 학사
2001-2003: 대학원전자정보공학부 석사
2003-2005: 대학원전자정보공학부 박사과정

관심분야: 원격 모니터링, 공장 자동화, Sensor Network

Bogdana Diaconescu
1996년: 루마니아 부카레스트 대학교 기계공학부 석사
2004-현재: 군산대학교 전자정보공학부 석사과정

관심분야: 고장진단, Sensor Network