A Fault-Tolerant Schema for Clock Synchronization and data aggregation in WSN

Na Wang
School of software, East China Normal University,
NO. 3663 north ZhongShan Road
Shanghai, 200062, China

School of Computer and information, Shanghai Second Polytechnic University,
NO. 2360 Jinhai Road, Shanghai, 201209, China
E-mail: wnoffice@126.com

Haihui He
School of Computer and information, Shanghai Second Polytechnic University,
NO. 2360 Jinhai Road, Shanghai, 201209, China
E-mail: hhhe@it.sspu.cn

DongQian Liu
School of software, East China Normal University,
NO. 3663 north ZhongShan Road,
Shanghai, 200062, China
E-mail: maggie.liudon@gmail.com

Received 12 March 2012

Accepted 11 July 2012

Event monitoring and determination is a popular application in WSN. Considering the special circumstance that some nodes of the wireless sensor network are faulty, a fault tolerant schema for data aggregation based on event clustering was proposed. Also, an improved HRTS algorithm named T-HRTS which based on Hierarchy Referencing Clock Synchronization resolving the byzantine general problem will be introduced.

Keywords: wireless sensor network, Clock Synchronization, data aggregation, fault-tolerant, event clustering.

1. Introduction

Over the past few years distributed wireless sensor networks (WSN) have been the focus of considerable research for both civil and military applications. Wireless sensor network consisting of large number of micro sensor nodes can complement the collaborative awareness in coverage area. Sensor network is the bridge connecting the objective physical world and the virtual digital world accurately monitoring remote environment intelligently by combing the data from individual nodes. [1]

When event was detected, data from individual sensor must be aggregated to determine abnormality level. Control algorithm is based on the level of the abnormality, so how to determine a relatively exact level is crucial. When measuring the abnormality, we should consider either the transmission route throughout the network or the same time spot sensors gathering the data.

Data acquisition, processing and transmission in sensor network have the nature of timing sequence which usually requires nodes in the network have the same clock, thus the clock synchronization technology is one of the important supporting technology for sensor networks.

Node in the distributed system has its own local clock, and it's difficult to achieve long-term time synchronization between nodes due to some internal factors such as crystal oscillator frequency deviation and a number of external influences such as temperature changes, electromagnetic interference, malicious attacks including external and internal attacks and plant ageing. External attacks are those in which an attacker manipulates the communication between pairs of trusted nodes and causes the nodes to desynchronize, or to remain un-synchronized even after a successful run of the synchronization protocol. Pulse delay attack is an example of external attack. Internal attacks are those in which internal attackers (group members) report false clock references to their neighboring nodes. For such a process to be fault-tolerant, the clock synchronization algorithm must work despite faulty behavior by some processes and clocks.[14]

Several clock synchronization protocols have been proposed for sensor networks to achieve either pair-wise clock synchronization or global clock synchronization.[9,16,17] Pair-wise clock synchronization aims to obtain high precision clock synchronization between pairs of neighbor nodes, while global clock synchronization aims to provide network-wide clock synchronization in a sensor network. Existing pair-wise clock synchronization protocols use either receiver-receiver synchronization,[16] in which a reference node broadcasts a reference packet to help pairs of receivers identify the clock differences, or sender-receiver synchronization,[9] where a sender communicates with a receiver to estimate the clock difference. Most of the global clock synchronization protocols establish multi-hop paths in a sensor network so that all the nodes can synchronize their clocks to a given source based on these paths and the pair-wise clock differences between adjacent nodes in these paths. [18]

It is natural to consider fault-tolerant clock synchronization techniques, which have been studied extensively in the context of distributed systems. [8,14,15,19,20] However, traditional fault-tolerant clock synchronization techniques are not directly applicable to sensor networks. These techniques were developed for distributed systems that do not have the same resource constraints as sensor networks. All of these techniques involve heavy communication among the nodes, and sometimes heavy computation as well. This is because these techniques either use digital signatures or multiple copies of messages to prevent a malicious node from modifying or destroying clock information sent by non-faulty nodes without being detected. Digital signature is usually not practical in resource constrained sensor networks. Even when digital signature is used, each node still needs to send a message to every other node in each synchronization round, resulting in at least O (n^2) communication complexity, where n is the number of nodes. Some schemes require that all nodes that receive certain messages process and forward these messages to all the other nodes immediately, resulting in a high probability of message collisions if used in sensor networks. [20,21]

In this paper, we develop a fault-tolerant clock synchronization scheme for clusters of sensor nodes, where the nodes in each cluster can communicate with cluster-head directly. In each round of clock synchronization within a cluster, every node broadcast received data from other nodes in last round.

This paper is an extension of paper named "Time Synchronization for Failure Tolerance in Wireless Sensor Network" in SNPD2012.

The rest of the paper is organized as follows. In Section II, we introduce the fault tolerant schema for data aggregation. In Section III, we describe HRTS. In Section IV, we put forward the approach of Clock Synchronization with Failure Tolerance. Section V provides the conclusion.

2. A Fault Tolerant Schema for Data Aggregation

2.1. *Abnormality determination*

When detect abnormality, sensors must be clustered into numbers of local sensor networks according to the region they are located. Besides, each region of sensors has their own autonomy. In other words, all sensors, which are in the same region, can execute the proposed protocol without the sink and other unconcerned sensors. This can reduce the time for collecting data and designing the final result.

The clustering process is as follows:

A reference node will be regarded as level 1 named root. Generally, node receiving the event with the most power is selected as reference. The reference node broadcasts the rating information containing its level and number. Each node receiving the rating information sets its levels as reference node's level plus 1 and adds the node number and level to its neighbor list.

Then, second-level nodes broadcast rating information. The nodes receiving the information set their level as sender's level plus 1 if they have still no level, at the same time adding the node number and level to their neighbor list. And so on, until all nodes have their own levels, and each node knows the number and level of all their neighbor nodes. So the nodes can be divided into three categories as parent node, neighbor node at the same level and child node.

After clustering, the proposed protocol can let each sensor reach an agreement and do the corresponding action with the following assumptions.

➢ Let N be the set of all sensors in the local autonomous WSN and| N |=n.

➢ The total number of faulty sensors and transmission media in the local WSN is ⌊(n-1)/3⌋.

➢ Each sensor needs to collect messages through ⌊(n−1)/3⌋+1 rounds of message exchange.

➢Each sensor has its own initial value Vi.

In classic WSN, if the detected value meets some condition, then the initial value must be set to 1, otherwise, 0. Take the fire control system for instance, if the sensor detects the temperature is higher than 50°C, then its initial value is 1, otherwise, 0 as default. [23]

While considering the level of abnormality in WSN, take the water quality control system for instance, if the sensor detects the PH is higher than 8 or lower than 6, then its initial value is 1, if the PH is higher than 9 or lower than 5, then its initial value is 2, if the PH is higher than 10 or lower than 4, then its initial value is 3, otherwise, 0 as default.

When sensors detect an event, they must decide their own initial value for running the consensus problem algorithm. After that, each sensor continues to execute the protocol. The proposed protocol includes message exchange phase and decision phase. Message in this paper consists of node ID, node state, PH value, sampling frequency and local clock.

In the message exchange phase, each sensor collects and exchanges messages from other sensors with ⌊(n−1)/3⌋+1 rounds of message exchange. As shown in Fig. 1, all the received messages are used to construct a tree called M-tree.[24] In Figure 1, the message received from each sensor in the first round will be saved in the first level of the tree and we use Vi to represent it. During the second to the (⌊(n−1)/3⌋+1)th rounds of message exchange phase, each sensor exchanges the received messages, which come from previous rounds, to other sensors. Then it stores the received messages to the second level and we use Vj:i (where j<>i) to represent it, continuing to the (⌊(n−1)/3⌋+1)th level of the M-Tree. Each level of the M-Tree contains a round of received messages and each vertex is labeled with a non-repeating sequence of sensor identifiers to avoid the repeating effect from the sender.

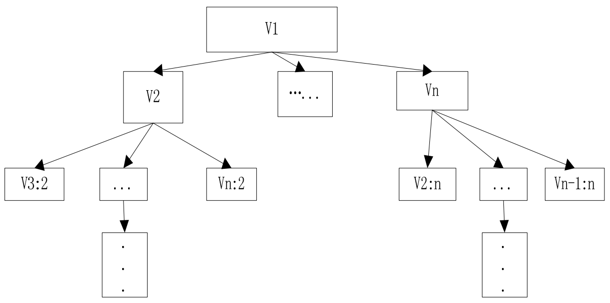

Fig.1. M-TREE for V1

2.2. *Proposed protocol*

After finishing ⌊(n−1)/3⌋+1 rounds of message exchange, each sensor must execute the agreement phase. The proposed protocol is shown in Fig. 2.

```
Protocol
Definitions:
n: The number of sensors in one cluster
Vi: The initial message sent by each sensor
V j:i: The message received by sensor i from sensor j
f: ⌊(n−1)/3⌋
Major(v): majority value of each vertex
Resolve(v): A function of taking the common value
Message exchange phase:
Round 1:
1.  Initiate each sensor's value in level 0
2.  store value Vi in level 1
Round 2:
1.  send level 1 of your tree to all
2.  store value V j:i in level 2
Continue for f + 1 rounds
Calculating the Decision:
In round f + 1, each sensor uses the values in its tree to
compute its decision.Recursively compute the "decision"
value for the root of the tree.
Function major()

Definitions:
Vi: The initial message sent by each sensor
Major(x,y): get majority when value is in {x,y}, return the
number of the majority
initiate:
      if sensor's value is 0 then vi=0;
      else vi=1;
major(0,1):
      if the number of 0 is more than one half
      major()=0;
      else if major()=1
        {
          delete vertex having value 0;
         Get vertex's true value vi;
      //vi must value in {1,2,3} in this paper
        major(1,major(2,3));
        }
Function decision()
Vi: The initial message sent by each sensor
decision(v):
    if v a leaf
        decision(v)= vi
    else
        decision(v)=major( decision (v') : v' is a child of V)
```

Fig.2. The proposed protocol and function

3. Hierarchy Referencing Clock Synchronization Protocol

3.1. *HRTS in single-hop network*

HRTS is based on the sender-receiver clock synchronization mechanism. HRTS achieve clock synchronization between the sender and the receiver mainly through three data communications. [11] In the first synchronous communication process, the reference node (sender) broadcasts a synchronization request command frame named f1 and record the time t1. The synchronization request command frame contains a response node which is randomly selected in the reference node's neighbor table to complete the communication process. All nodes within the broadcast range of the reference node record the f1's arriving time, but only the response node will reply to the command frame. The reception moment of the neighbor node i is denoted by t2i and that of answer node is denoted by t2.

In the second synchronous communication process, response node reply to the reference node with a synchronization response command frame named f2 which contains the moment t2 when response node received the synchronization request command frame F1 and the moment t3 when response node sends frame f2. Reference node recorded its reception moment of f2 as t4.

The above-mentioned time is local time in each node. It's assumed that the message transmission time between any two nodes is the same and is denoted by d. The time offset between the sender and receiver is fixed during the period of time t1 to t4 which denoted by Δ. The local time of reference node is denoted by Tr, response node by Tp and other nodes i by Ti. When the reference node receives f2, we can get the following relationship:

$$\Delta = (t2 - t1 + t3 - t4)/2$$

With the above relationship, the reference node will broadcast a synchronization command frame again named f3 which is filled with t2 and the value of Δ. Neighbor nodes can calibrate their own local time after receiving f3 according to the information contained in the command frame. The specific relationship is as follows:

$$Tr = Ti + t2 - t2i - \Delta$$

The equation above can calibrate the local time of node i and t2 - t2i –Δ is called compensation time of node i. After the synchronization process, all neighbor nodes can keep pace with reference node.

It can be seen that it is bidirectional synchronization between reference node and response node, while unidirectional broadcast synchronization between the reference node and other neighbors.

3.2. *Hierarchical cluster tree in HRTS*

After clustering, child nodes having the same parent compose a cluster. The node having the largest degrees will be selected as the head of itself and its neighbors at the same level in the cluster. Each node and its child nodes constitute a connected single-hop area, and the network is divided into a lot of single-hop areas that nodes can communicate with each other directly.

When applying HRTS into multi-hop network, with a hierarchical cluster tree, HRTS can be applied in single-hop areas directly.

If the nodes are deployed as in Fig. 3, where node 1 is most powerful, the tree will be created as Fig. 4.

4. Clock Synchronization for Failure Tolerance

4.1. *Node Faults*

Nodes have several hardware and software components that can produce malfunctions. For example, the enclosure can suffer impacts and expose the hardware of the sensor node to the extreme conditions of the environment.

When the battery of a node reaches a certain stage, sensor readings may become incorrect. Hardware failures will generally lead to software failure. A Data Acquisition application will not perform properly if the underlying sensors are providing incorrect readings. Nevertheless, some hardware failures do not affect all the services in a sensor node. In the example discussed, although the node cannot be used to provide correct sensor readings it still can be used to route packages in the sensor network.[15]

Organizing a network in clusters is an approach used in many applications, for example to extend the lifetime of the network. A small number of nodes are selected to become cluster heads. They are responsible for coordinating the nodes in their clusters, for instance by collecting data from them and forwarding it to the base station.

In case that a cluster head fails, no messages of its cluster will be forwarded to the base station any longer. The cluster head can also intentionally or due to software bugs forward incorrect information. Depending on the application case, the impact of such a failure can vary from quality degradation of measurements to alarm messages not being delivered to a back-end system.

While forwarding messages, nodes can aggregate data from multiple other nodes in order to reduce the amount of data sent to the base station. One common simple approach is to calculate the average of correlated measured values such as temperature, humidity and pressure, sending only one message to the back-end.

If a node generates incorrect data, the data aggregation results can suffer deviations from the real value. Also, if a node responsible for generating the aggregated data is subject to a value failure, the base station will receive incorrect information of an entire region of the network.

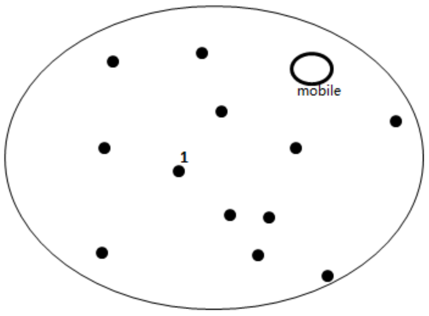

Fig.3. Deployment of nodes

In case of synchronization, a cluster head can suffer power failure and stop responding to requests of synchronization, or it can start sending arbitrary clock either intentionally or due to a malfunction.

4.2. *Major clock in T-HRTS*

In practical applications, it can be considered the same clock when clock error between two sensor nodes is relatively smaller compared with the synchronization precision.

Definition 1: In a cluster, if there are multiple neighbor nodes that have the same clock with the head node, the clock of the head node is called the major clock.

Definition 2: In a cluster, let θ=t2 - t2i, when θ is small enough to be omitted, it is regarded that the clock of cluster head and node i is the same.

4.3. *Fault Detection and recovery*

Generally, if a node is non-faulty, the clock may not shift much in a cycle. There is special case that the intended reference node is attacker and sends error in the third process in HRTS. In this case, the synchronization will fail. Thus, the root of the sub-tree must be changed to deal with the attacker.

After initializing, when synchronization is required, the root node will broadcast its own clock Tr and Δ to its child. The cluster head which is response node decides the major clock. If the root's clock is not in the majority, it may be fault.

Applying the assumption in multi-hop network, when the root of sub-tree is fault, it sends faulty clock to the children. It may be that the clock in the third process is earlier than that in the first process. So the synchronization result will be an unreasonable clock.

In order to detect all the faulty, detection procedure must execute from top to bottom all through the tree. If the root is fault, mobile node can be used instead of it. If the cluster head if fault, we use Byzantine algorithm for each sub-tree.

Exponential Tree Algorithm for Byzantine [2] is as follows:

Each tree node is labeled with a sequence of unique processor indices. Root's level is 0 and root has n children, labeled 0 through n – 1 where n is the number of nodes in the sub-tree.

Child node labeled i has n - 1 children, labeled i : 0

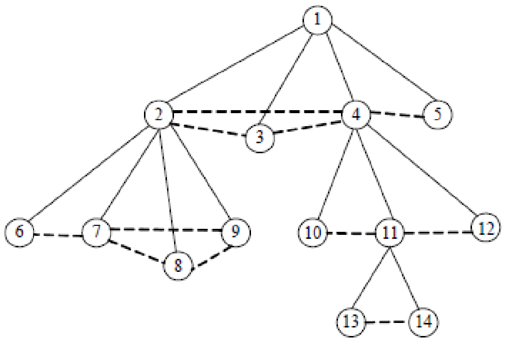

Fig.4. Hierarchical cluster tree

through i : n-1 (skipping i : i).

Node at level d labeled v has n - d children, labeled v : 0 through v : n-1 (skipping any index appearing in v). Nodes at level f + 1 are leaves and f is the number of fault nodes that must meet with the inequality as 3f+1<=n. The process continues for f+1 rounds.

For example, in Fig. 4, the sub-tree (4, 10, 11, 12) may build a tree like Fig. 5.

In the exponential tree, node 4 is proposed as a common node. At the third level, there are three nodes indicating that node 4 receives local clock from node 10, 11 and 12. With the limitation that 3f+1<=n, the majority clock of the three nodes must be the non-fault clock. Even though the root of sub-tree is fault, the common clock could not be impacted by using the major clock.

4.4. *Synchronization process in T-HRTS*

After recreating the hierarchical cluster tree, the child synchronizes with the root in every sub-tree according to T-HRTS algorithm. As a result, the clock throughout the network will be the same at a moment.

If the major clock has been decided as the cluster head's clock, the clock will be sent to its root and then to the base. If this is not the case, the sub-tree will be synchronized with the root.

In our model, when the root is fault, it may send fake clock to its sub-tree. With using HRTS, the fake clock will spread to the cluster head, then to all the tree nodes. Therefore, fault-tolerant can't be implemented which is a crucial requirement in distributed system. In our algorithm, although root is fault, its children may have the approximate clock that can serve as the major clock to ensure the right clock throughout the network.

We assume four faulty nodes and test the model as the outcome bellow.

Table 1 shows the worst condition that even if all the roots of sub-trees are fault, the clock in the network can be synchronized correctly.

Table1. The faulty nodes and Testing

level	Faulty nodes	Major clock	Most Clock update	fault-tolerant
1	1	mobile	NULL	YES
2	2	7	{6、8、9}	
2	4	11	{10、12}	
3	NULL	NULL	NULL	
2	3	1	{2、4、5}	YES
3	7	2	{6、8、9}	
3	11	4	{10、12}	

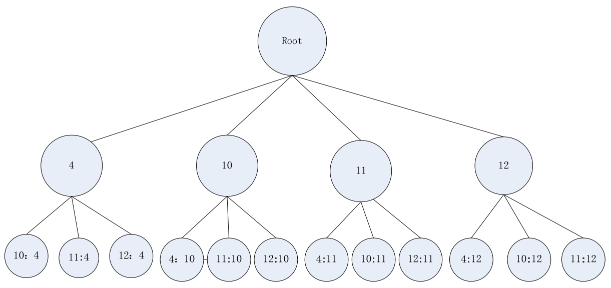

Fig.5. Exponential Tree

4.5. *Performance Analysis*

This algorithm is based on the HRTS algorithm with adding the concept of major clock. In HRTS, the optimization of selecting synchronous reference node is not considered although it would be fault. In T-HRTS, reference node is not necessarily the parent node, but probably the cluster head node having major clock. It can tolerant some failures with executing byzantine algorithm. The sub-tree at all levels need to decide which clock is the common right one, then update local clock for at most n-m times where n is the total number of the sub-tree and m is the number of the nodes having major clock. Although the overhead of the packet is not reduced, the frequency of clock updating is dramatically reduced especially when the scale of the network is large and the required clock synchronization precision is low. In addition, there is a crucial parameter θ that need to be researched especially.

5. Conclusion

When detecting an event, data form sensors in a network must be aggregated to determine a common value which will be used as a crucial parameter of certain execution. It is crucial to get a correct decision with considering transmission route and the same time spot sensors gathering the data. Practically, some sensors may get faulty and the transmission media between sensors may get disturbed by the environment noise. These facts may conduct the fault detected result and then cause an erroneous reaction. In the past, many solutions are proposed to detect faulty sensors. However, we must take the level of abnormality for granted. In this study, we propose a control algorithm solution to raise the correctness of detected result even when some sensors are faulty and some of the transmission media between sensors are disturbed. Under our scheme, sensors can take certain action with informing other nodes to change synchronously so that the local WSN application can take the corresponding actions more accurately and save storage space and electrical power.

Acknowledgements

This work is is supported by the National Basic Research Program of China (Grant No. 2011CB302802), and Shanghai Municipal Natural Science Foundation (No. 11ZR1413700). And thanks for the discussing and instruction of professor Yixiang Chen.

References

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Computer Networks 38 (2002) 393-422
2. Ryszard Klempous, Jan Nikodem, Byzantine Algorithms in Wireless Sensors Network, ICIA 2006
3. Leslie Lamport, P. M. Melliar-Smith, Byzantine Clock Synchronization, 1984 ACM
4. DOLEV, D. The Byzantine generals strike again. J. Algorithms 3, 1,Jan. 1982.
5. Arun Kumar Tripathi and Ajay Agarwal, An Approach towards Time Synchronization Based Secure Protocol for Wireless Sensor Network, NDT 2010, Part II, CCIS 88, pp. 321-332, 2010.
6. Kun Sun, Peng Ning, Cliff Wang, An Liu, Yuzheng Zhou, TinySeRSync: Secure and Resilient Time Synchronization in Wireless Sensor Networks, CCS'06, October 30-November 3, 2006.
7. LESLIE LAMPORT, ROBERT SHOSTAK, MARSHALL PEASE, The Byzantine Generals Problem, ACM 0164-0925/82/0700-0382, 1982
8. L. Lamport. Using Time Instead of Timeout for Fault-tolerant Distributed Systems. ACM Trans. onProg. Lang. and Sys. 6, 2 (April 1984), 254-280.
9. Ganeriwal S, Kumar R, Srivastava M B. Timing-sync Protocol for Sensor Networks[C]//Proc. of the 1st ACM Conf. on Embedded Network Sensor Systems. Los Angeles, CA, USA: [s. n.], 2003.
10. Jeremy E, Lewis G, Deborah E. Fine-grained Network Time Synch- ronization Using Reference Broadcasts[C]//Proc. of OSDI'02. Boston, MA, USA: [s. n.], 2002.
11. Dai Hui, Han R. TSync: A Lightweight Bi-directional Time Synchronization Service for Wireless Sensor Networks[J]. ACM Mobile Computing and Communications Review, 2004, 18(1): 125-139.
12. Xu Chaonong, Zhao Lei, Xu Yongjun. Broadcast Time Synchroni- zation Algorithm for Wireless Sensor Networks[C]//Proc. of the 11th International Conference on Sensing, Computing and Automation. Chongqing, China: [s. n.], 2006.
13. Capkunl, S., Ganeriwal, S., Han, S., Srivastava, M.: Securing Timing Synchronization inSensor Networks. In: Proceedings of, pp. 369-390. Springer, New York (2006).
14. Song, H., Zhu, G.C.S.: Attack-resilient time synchronization for wireless sensor networks.In: IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, p.772 (2005)
15. Hai Liu, Amiya Nayak, Ivan Stojmenovi?.:Fault Tolerant Algorithms/Protocols in Wireless. Computer Communications and Networks, 2009, 261-291, DOI: 10.1007/978-1-84882-218-4_10
16. J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization Using Reference Broadcasts," ACM SIGOPS Operating Systems Rev., vol. 36, pp. 147-163, 2002.
17. Q. Li and D. Rus, "Global Clock Synchronization in Sensor Networks," Proc. IEEE INFOCOM 2004 Conf., Mar. 2004.
18. Kun Sun, Peng Ning, and Cliff Wang, "Fault-Tolerant Cluster-Wise Clock Synchronization for Wireless Sensor Networks", IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 2, NO. 3, JULY-SEPTEMBER 2005
19. P. Ramanathan, K.G. Shin, and R.W. Butler, "Fault-Tolerant Clock Synchronization in Distributed Systems," IEEE Computer, vol. 23, no. 10, pp. 33-42, 1990.
20. J. Lundelius-Welch and N. Lynch, "A New Fault-Tolerant Algorithm for Clock Synchronization," Information and Computation, vol. 77, no. 1, pp. 1-36, 1988.
21. D. Dolev, J.Y. Halpern, B. Simons, and R. Strong, "Dynamic Fault-Tolerant Clock Synchronization" , J. ACM, vol. 42, no. 1, pp. 143-185, 1995.
22. T.K. Srikanth and S. Toueg, "Optimal Clock Synchronization," J. ACM, vol. 34, no. 3, pp. 626-645, 1987.
23. Hui-Ching Hsieh, Jenq-Shiou Leu, Wei-Kuan Shih, "A fault-tolerant scheme for an autonomous local wireless sensor network", Computer Standards & Interfaces 32 (2010) 215-221.
24. M. Pease, R. Shostak, L. Lamport, Reaching agreement in presence of faults, Journal of ACM 27 (2) (1980) 228-234.