Fault-diagnosis and Decision Making Algorithm for Determining Faulty Nodes in Malicious and Dormant Wireless Sensor Networks

Shu-Ching Wang[1], Mao-Lun Chiang[1], Kuo-Qin Yan[1], Yao-Te Tsai[2]
[1] Department of Information Management, Chaoyang University of Technology, Taiwan
[2] Department of International Business, Feng Chia University, Taiwan
{scwang; mlchiang; kqyan}@cyut.edu.tw, yaottsai@fcu.edu.tw

Abstract

Ubiquitous sensing enabled by *Wireless Sensor Network* (WSN) technologies cuts across many areas of modern day living. This offers the ability to measure, infer and understand environmental indicators, from delicate ecologies and natural resources to urban environments. WSNs are expected to be integrated into the *Internet of Things* (IoT), where sensor nodes join the Internet dynamically, and used to collaborate and accomplish their tasks. However, when an ultra large disaster occurs, communication networks would be severely disconnected by the damages of network nodes. The most important issues of fault-tolerance is the *Fault Diagnosis Agreement* (FDA) whose purpose is to make each fault-free node detect/locate a common set of faulty nodes. In this study, the FDA problem is solved by a fault diagnosis protocol using dormant and malicious failure characteristics on nodes in a WSN by collecting the accumulated messages. The proposed protocol cannot only reach an agreement from fault-free nodes but also detect and locate the faulty components in a fallible WSN. Moreover, the proposed protocol can also tolerate, detect and locate the maximum number of faulty components to make a WSN never die by minimum number of rounds of message exchanges.

Keywords: Fault tolerance, Fault detection, Fault diagnosis, Wireless sensor network

1 Introduction

The *Internet of Things* (IoT) is the massive deployment of trillions wireless Internet Protocol (IP)-based sensor nodes to identify and monitor every object around us [1]. Conventionally, WSN is often used to construct IoT [2].

Usually, since WSNs of IoT are deployed in open areas without protection from disaster, they are vulnerable to various types of attacks. Sensor nodes of WSN are essential for detecting various kinds of data of the serious disasters in residential areas [1]. Each sensor node communicates with other sensor nodes by using broadcast in WSN, but also leads to a broadcast storms problem [2]. Therefore, the researchers proposed *Cluster-based WSN* (CWSN) to ameliorate the broadcast storm [3]. Figure 1 is a topology example of CWSN. Moreover, the network configuration underlying the emergency situation must be considered in setting up a new network. Therefore, the stability and reliability of CWSNs are important issues to keep environment good for data transmission [2]. In other words, to propose a mechanism to allow all well-perform nodes reach an agreement is necessary to ensure CWSN stable and reliable.

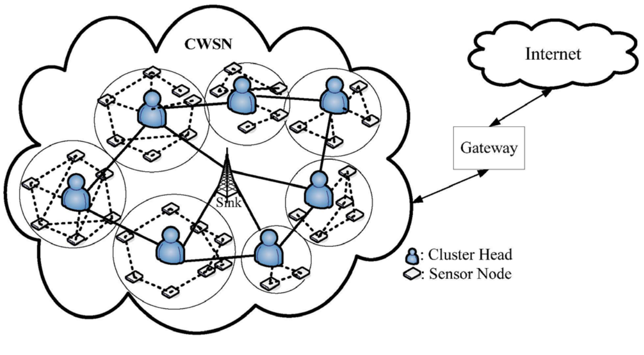


Figure 1. The topology example of CWSN

To improve the reliability of CWSN, a mechanism to allow a set of nodes to agree on a common value is required. The *Byzantine Agreement* (BA) problem is one of the most fundamental problems in which a common value is reached in a distributed system [4]. According to the definition of the BA problem by Lamport [4]: (1) the nodes communicate with each other; (2) a node is chosen as the source node to start with an initial value and communicate to each other by exchanging messages; (3) after message exchanges, all fault-free nodes should reach a common agreement, if and only if the number of faulty nodes f_n is less than one-third of the total number of nodes in the network

*Corresponding Author: Yao-Te Tsai; E-mail: yaottsai@fcu.edu.tw

DOI: 10.3966/160792642018121907016

($f_n \leq \lfloor (n\text{-}1)/3 \rfloor$).

Based on these assumptions, the BA requirement can be satisfied when the following constraints are met:

Agreement: All fault-free nodes agree on a common decision value.

Validity: If the source node is fault-free, then all fault-free nodes agree on the initial value sent by the source node.

A related FDA problem is to make each fault-free node be able to detect/locate the faulty components in the distributed system [5]. Protocols designed to solve the FDA problem should meet the following requirements [6]:

Agreement: All fault-free nodes should be able to identify the common set of faulty nodes.

Fairness: No fault-free node is falsely detected as faulty by any fault-free node.

In this study, a new protocol FDWSN (Fault Diagnosis based WSN) is proposed for solving the FDA problem in a CWSN. FDWSN collects the messages accumulated in a BA protocol and then detects/locates the common set of faulty components by examining the collected evidence.

The rest of this paper is organized as follows. The relevant knowledge is explained in Section 2. In Section 3, the BA protocol TAP is introduced. Our new protocol FDWSN is illustrated in Section 4. The correctness and complexity are proved in Section 5. Finally, the conclusion is presented in Section 6.

2 Relevant Knowledge

The symptoms of a faulty component can be classified into two categories. They can be either dormant faults or malicious faults [7]. In the synchronous system, each fault-free node can detect the components with dormant faults if the protocol appropriately encodes a message before transmission by using the Manchester code [7]. In case of a malicious fault, the behavior of the faulty component is unpredictable and arbitrary. For example, the behavior of a faulty component with the malicious, may lie, lose, or mangle messages.

The dual failure mode is one where both dormant faults and malicious faults are allowed to happen to the faulty components in the network. In previous researches, the focus seems to always be fixed upon the components with malicious faults only [6]. However, the failure type of a faulty component can be either dormant or malicious [7]. Therefore, concentrating upon malicious faults only will make BA or FDA protocol not able to handle the maximum number of faulty components when faulty components with dormant faults exist in the system. So, if we can solve the BA problem and FDA problem with the dual failure mode, then our new protocol must be more powerful and practical.

The FDA problem is taken care of in a synchronous network, where the bounds of delay for each fault-free component are finite [5]. The assumptions and parameters of our protocol to solve the FDA problem in a CWSN are as follows:

- Let N be the set of all nodes in the network and $|N|= n$.
- Let C be the set of all clusters in the network, and $|C|= c$, where c is the number of clusters in the network.
- If there are at least $\lceil \mu_i /2 \rceil$ malicious faulty nodes in C_i, then C_i will be a malicious faulty cluster. Here, C_i is the i-th cluster, and μ_i is the number of nodes in C_i, $0 \leq i \leq c$.
- If there are at least $\lceil \mu_i /2 \rceil$ dormant faulty nodes in C_i, then C_i will be a dormant faulty cluster.
- Let f_{mc} be the number of malicious faulty clusters allowed.
- Let f_{dc} be the number of dormant faulty clusters allowed.
- Let f_{mn} be the number of malicious faulty nodes.
- Let f_{dn} be the number of dormant faulty nodes.
- Let f_n be the maximum number of faulty nodes, where $f_n = f_{mn} + f_{dn}$.
- The number of rounds of message exchange is f_c +1, where $f_c = \lfloor (c\text{-}1)/3 \rfloor$.

The number of faulty nodes allowed in the network depends on the total number of nodes in the network and the node failure types. In Lamport *et al.* [4], the assumption of node fault type is malicious in a static network. The constraints of Lamport *et al.* is $n > 3f_m$, where $3f_m$ is the number of malicious faulty nodes [4].

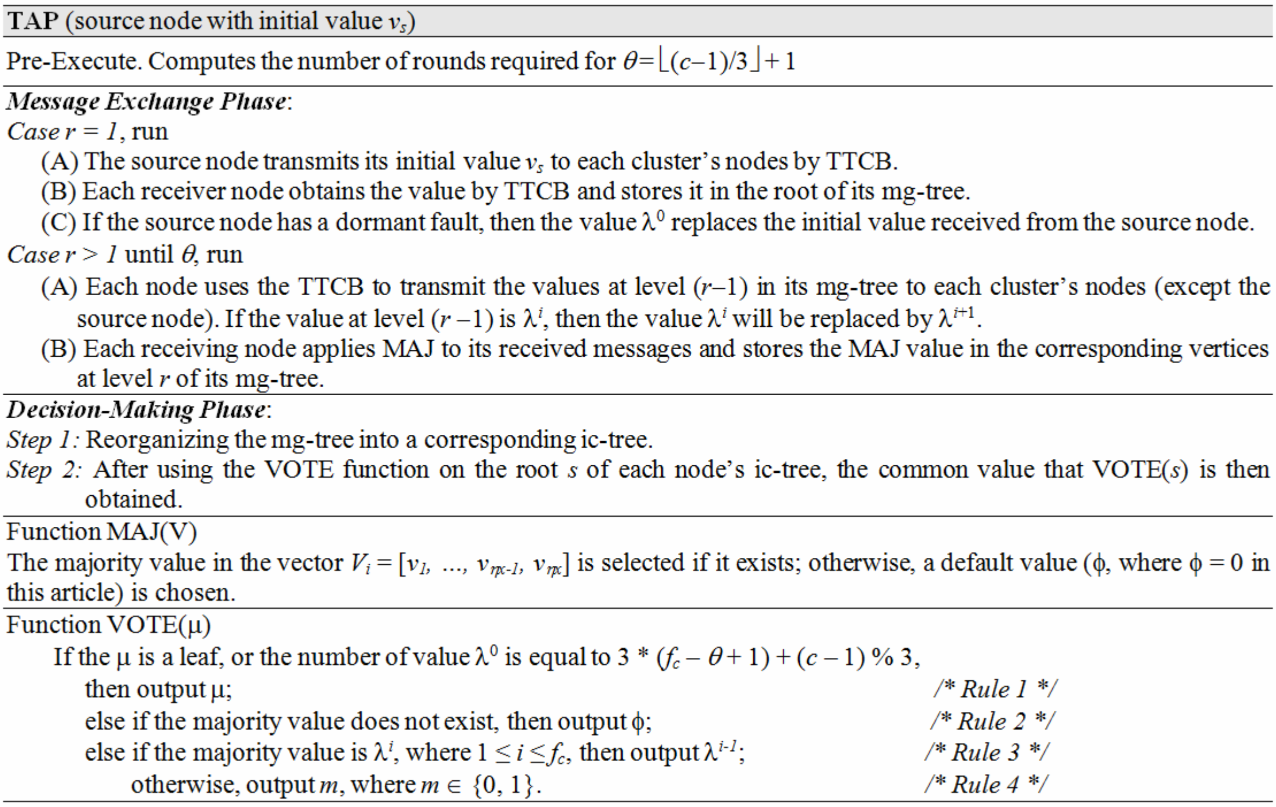
The BA problem in a CWSN with dual fallible nodes was solved by Wang et al. [8]. The constraint of Wang et al. is $c > \lfloor (c\text{-}1)/3 \rfloor + 2f_{mc} + f_{dc}$, where c is the number of clusters, f_{mc} is the number of allowable malicious faulty clusters, and f_{dc} is the number of allowable dormant faulty clusters in a CWSN [8].

Since the FDA protocol is used to detect/locate the faulty components by considering the evidence dug out from the BA problem, the constraints of the FDA protocol falls within the BA problem. Hence, the proposed FDWSN protocol can solve the FDA problem in a CWSN by the evidence in the BA problem if the contrant is satisfied.

3 The BA Protocol TAP

FDWSN is used to solve the FDA problem in a CWSN by collecting the received messages in the BA protocol TAP (Trust Agreement Protocol) [8]. Hence, the BA protocol TAP must be observed first.

To ensure a network reliable and synchronous, the *Trusted Timely Computing Base* (TTCB) is used in the BA protocol TAP when messages are transmitted [9]. There are two phases in TAP, the *message exchange* *phase* and the *decision making phase*. In the *message exchange phase*, each node gets enough information and stores the received messages in the corresponding vertices at level r of its mg-tree [8], which needs $\lfloor(c-1)/3\rfloor+1$ rounds of message exchanges, where c is the number of clusters in the CWSN. In the *decision making phase*, each fault-free node turns its mg-tree into a corresponding ic-tree [8]. The definition of TAP is shown in Figure 2.



```
TAP (source node with initial value $v_s$)
Pre-Execute. Computes the number of rounds required for $\theta=\lfloor(c-1)/3\rfloor+1$
Message Exchange Phase:
Case $r = 1$, run
    (A) The source node transmits its initial value $v_s$ to each cluster's nodes by TTCB.
    (B) Each receiver node obtains the value by TTCB and stores it in the root of its mg-tree.
    (C) If the source node has a dormant fault, then the value $\lambda^0$ replaces the initial value received from the source node.
Case $r > 1$ until $\theta$, run
    (A) Each node uses the TTCB to transmit the values at level $(r-1)$ in its mg-tree to each cluster's nodes (except the
        source node). If the value at level $(r-1)$ is $\lambda^i$, then the value $\lambda^i$ will be replaced by $\lambda^{i+1}$.
    (B) Each receiving node applies MAJ to its received messages and stores the MAJ value in the corresponding vertices
        at level $r$ of its mg-tree.
Decision-Making Phase:
Step 1: Reorganizing the mg-tree into a corresponding ic-tree.
Step 2: After using the VOTE function on the root $s$ of each node's ic-tree, the common value that VOTE($s$) is then
        obtained.
Function MAJ(V)
The majority value in the vector $V_i = [v_1, ..., v_{\pi x-1}, v_{\pi x}]$ is selected if it exists; otherwise, a default value ($\phi$, where $\phi = 0$ in
this article) is chosen.
Function VOTE($\mu$)
    If the $\mu$ is a leaf, or the number of value $\lambda^0$ is equal to 3 * $(f_c - \theta + 1) + (c - 1)$ % 3,
        then output $\mu$;                                                        /* Rule 1 */
        else if the majority value does not exist, then output $\phi$;            /* Rule 2 */
        else if the majority value is $\lambda^i$, where $1 \le i \le f_c$, then output $\lambda^{i-1}$;   /* Rule 3 */
            otherwise, output $m$, where $m \in \{0, 1\}$.                        /* Rule 4 */
```

Figure 2. The TAP protocol [8]

In this section, an example of executing TAP is given. A CWSN is shown in Figure 3(a). There are twelve nodes falling into eight clusters. n_s and n_1 belong to C_1; n_2 belongs to C_2, n_3 belongs to C_3, n_4 and n_5 belong to C_4, n_6 belongs to n_5, n_7 belongs to C_6, n_8, n_9 and n_{10} belong to C_7, and n_{11} belongs to C_8. The malicious faulty nodes are n_s, n_9, and n_{10}, and the dormant faulty node is n_6.

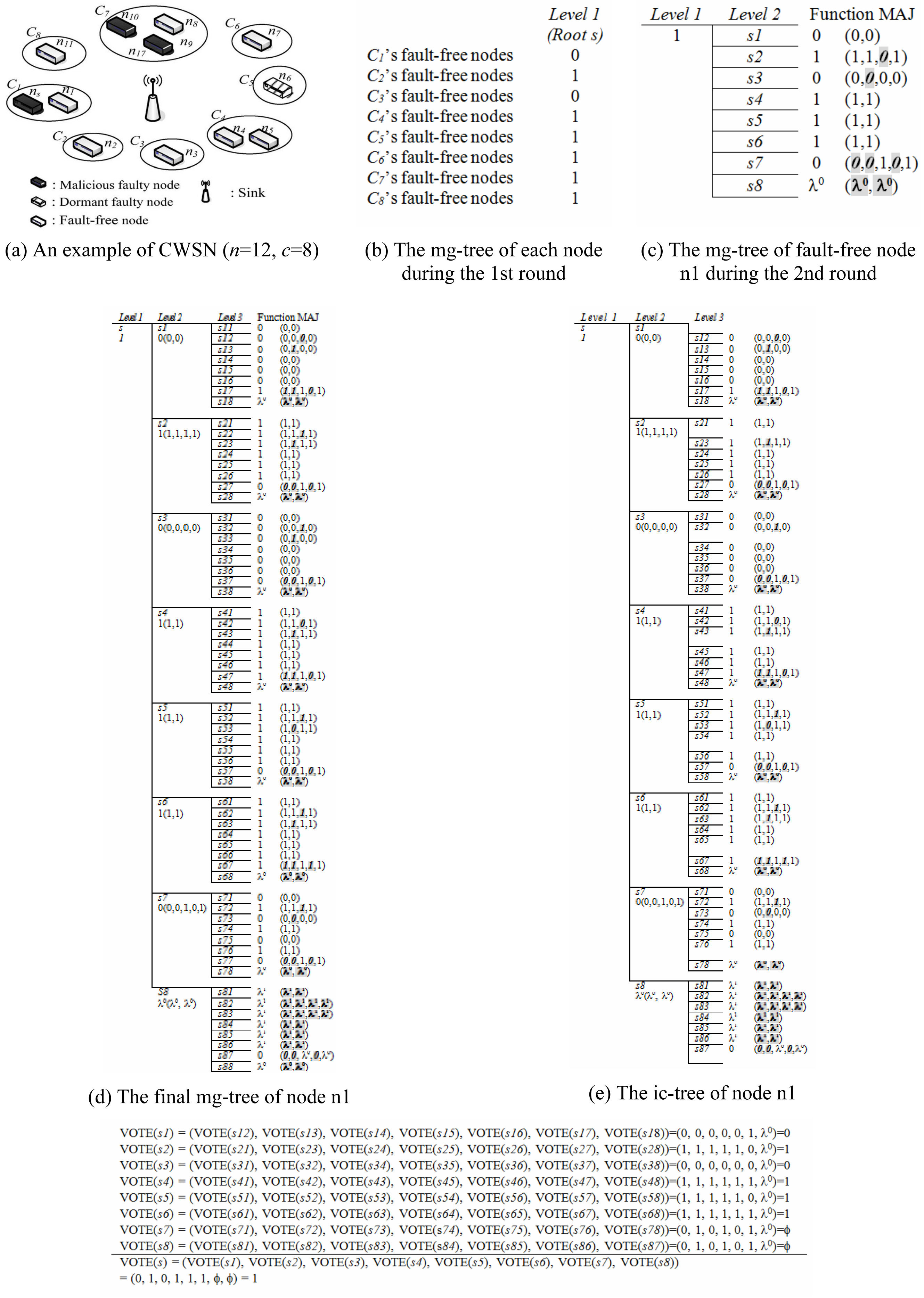
The worst case of the BA problem is that the source node commits malicious faults. For example, suppose n_s is the source node, which means n_s may transmit different values to different clusters. In order to reach a common value among fault-free nodes, TAP needs 3 ($\lfloor(c$ -1)/3$\rfloor$+1) rounds of message exchanges, where c is the total number of clusters in the CWSN.

In the first round of the *message exchange phase*, the source node n_s transmits its initial value v_s to each cluster's nodes. The message stored by each cluster's fault-free nodes in the first round of the message exchange phase is shown in Figure 3(b). In the r-th (where r>1) round of message exchanges, each node transmits the values at level $r-1$ in its mg-tree to the others and itself. Then, each receiving node applies MAJ to its received messages and stores the MAJ value in the corresponding vertices at level r of its mg-tree. The mg-tree of fault-free node n_1 at the 2nd and 3rd round in the message exchange phase are shown in Figure 3(c) and Figure 3(d).

In the *decision making phase*, each fault-free node turns its mg-tree into a corresponding ic-tree by deleting the vertices with duplicated cluster names. An example ic-tree is illustrated in Figure 3(e). Finally, apply the function VOTE to value s in the root for each node's ic-tree, VOTE(s)=ϕ, acommon value ϕ can be obtained. That is, after executing the BA protocol TAP, all the fault-free nodes can agree on a common value ϕ.

4 The FDA Protocol FDWSN

The proposed protocol FDWSN is used to solve the FDA problem by using the evidence gathered from the BA protocol TAP [8] in a CWSN. There are three phases in the FDWSN: the *message collection phase*, the *fault diagnosis phase*, and the *reconfiguration phase*. The *message collection phase* is used to collect all the nodes' ic-trees. The *fault diagnosis phase* is used to detect/locate the dormant and malicious faulty components. The *reconfiguration phase* is used to reconfigure the network. The definition of FDWSN is shown in Figure 4.



(a) An example of CWSN (n=12, c=8)

	Level 1 (Root s)
C_1's fault-free nodes	0
C_2's fault-free nodes	1
C_3's fault-free nodes	0
C_4's fault-free nodes	1
C_5's fault-free nodes	1
C_6's fault-free nodes	1
C_7's fault-free nodes	1
C_8's fault-free nodes	1

(b) The mg-tree of each node during the 1st round

Level 1	Level 2	Function MAJ	
1	s1	0	(0,0)
	s2	1	(1,1,*0*,1)
	s3	0	(0,*0*,0,0)
	s4	1	(1,1)
	s5	1	(1,1)
	s6	1	(1,1)
	s7	0	(*0*,*0*,1,*0*,1)
	s8	λ^0	(λ^0, λ^0)

(c) The mg-tree of fault-free node n1 during the 2nd round

Level 1	Level 2	Level 3	Function MAJ	
s 1	s1 0(0,0)	s11	0	(0,0)
		s12	0	(0,0,*0*,0)
		s13	0	(0,*1*,0,0)
		s14	0	(0,0)
		s15	0	(0,0)
		s16	0	(0,0)
		s17	1	(*1*,*1*,1,*0*,1)
		s18	λ^0	(λ^0,λ^0)
	s2 1(1,1,1,1)	s21	1	(1,1)
		s22	1	(1,1,*1*,1)
		s23	1	(1,*1*,1,1)
		s24	1	(1,1)
		s25	1	(1,1)
		s26	1	(1,1)
		s27	0	(*0*,*0*,1,*0*,1)
		s28	λ^0	(λ^0,λ^0)
	s3 0(0,0,0,0)	s31	0	(0,0)
		s32	0	(0,0,*1*,0)
		s33	0	(0,*1*,0,0)
		s34	0	(0,0)
		s35	0	(0,0)
		s36	0	(0,0)
		s37	0	(*0*,*0*,1,*0*,1)
		s38	λ^0	(λ^0,λ^0)
	s4 1(1,1)	s41	1	(1,1)
		s42	1	(1,1,*0*,1)
		s43	1	(1,*1*,1,1)
		s44	1	(1,1)
		s45	1	(1,1)
		s46	1	(1,1)
		s47	1	(*1*,*1*,1,*0*,1)
		s48	λ^0	(λ^0,λ^0)
	s5 1(1,1)	s51	1	(1,1)
		s52	1	(1,1,*1*,1)
		s53	1	(1,*0*,1,1)
		s54	1	(1,1)
		s55	1	(1,1)
		s56	1	(1,1)
		s57	0	(*0*,*0*,1,*0*,1)
		s58	λ^0	(λ^0,λ^0)
	s6 1(1,1)	s61	1	(1,1)
		s62	1	(1,1,*1*,1)
		s63	1	(1,*1*,1,1)
		s64	1	(1,1)
		s65	1	(1,1)
		s66	1	(1,1)
		s67	1	(*1*,*1*,1,*1*,1)
		s68	λ^0	(λ^0,λ^0)
	s7 0(0,0,1,0,1)	s71	0	(0,0)
		s72	1	(1,1,*1*,1)
		s73	0	(0,*0*,0,0)
		s74	1	(1,1)
		s75	0	(0,0)
		s76	1	(1,1)
		s77	0	(*0*,*0*,1,*0*,1)
		s78	λ^0	(λ^0, λ^0)
	s8 λ^0(λ^0, λ^0)	s81	λ^1	(λ^1,λ^1)
		s82	λ^1	(λ^1,λ^1,λ^1,λ^1)
		s83	λ^1	(λ^1,λ^1,λ^1,λ^1)
		s84	λ^1	(λ^1,λ^1)
		s85	λ^1	(λ^1,λ^1)
		s86	λ^1	(λ^1,λ^1)
		s87	0	(*0*,*0*, λ^0,*0*,λ^0)
		s88	λ^0	(λ^0,λ^0)

(d) The final mg-tree of node n1

Level 1	Level 2	Level 3		
s 1	s1 0(0,0)	s12	0	(0,0,*0*,0)
		s13	0	(0,*1*,0,0)
		s14	0	(0,0)
		s15	0	(0,0)
		s16	0	(0,0)
		s17	1	(*1*,*1*,1,*0*,1)
		s18	λ^0	(λ^0,λ^0)
	s2 1(1,1,1,1)	s21	1	(1,1)
		s23	1	(1,*1*,1,1)
		s24	1	(1,1)
		s25	1	(1,1)
		s26	1	(1,1)
		s27	0	(*0*,*0*,1,*0*,1)
		s28	λ^0	(λ^0,λ^0)
	s3 0(0,0,0,0)	s31	0	(0,0)
		s32	0	(0,0,*1*,0)
		s34	0	(0,0)
		s35	0	(0,0)
		s36	0	(0,0)
		s37	0	(*0*,*0*,1,*0*,1)
		s38	λ^0	(λ^0,λ^0)
	s4 1(1,1)	s41	1	(1,1)
		s42	1	(1,1,*0*,1)
		s43	1	(1,*1*,1,1)
		s45	1	(1,1)
		s46	1	(1,1)
		s47	1	(*1*,*1*,1,*0*,1)
		s48	λ^0	(λ^0,λ^0)
	s5 1(1,1)	s51	1	(1,1)
		s52	1	(1,1,*1*,1)
		s53	1	(1,*0*,1,1)
		s54	1	(1,1)
		s56	1	(1,1)
		s57	0	(*0*,*0*,1,*0*,1)
		s58	λ^0	(λ^0,λ^0)
	s6 1(1,1)	s61	1	(1,1)
		s62	1	(1,1,*1*,1)
		s63	1	(1,*1*,1,1)
		s64	1	(1,1)
		s65	1	(1,1)
		s67	1	(*1*,*1*,1,*1*,1)
		s68	λ^0	(λ^0,λ^0)
	s7 0(0,0,1,0,1)	s71	0	(0,0)
		s72	1	(1,1,*1*,1)
		s73	0	(0,*0*,0,0)
		s74	1	(1,1)
		s75	0	(0,0)
		s76	1	(1,1)
		s78	λ^0	(λ^0, λ^0)
	s8 λ^0(λ^0, λ^0)	s81	λ^1	(λ^1,λ^1)
		s82	λ^1	(λ^1,λ^1,λ^1,λ^1)
		s83	λ^1	(λ^1,λ^1,λ^1,λ^1)
		s84	λ^1	(λ^1,λ^1)
		s85	λ^1	(λ^1,λ^1)
		s86	λ^1	(λ^1,λ^1)
		s87	0	(*0*,*0*, λ^0,*0*,λ^0)

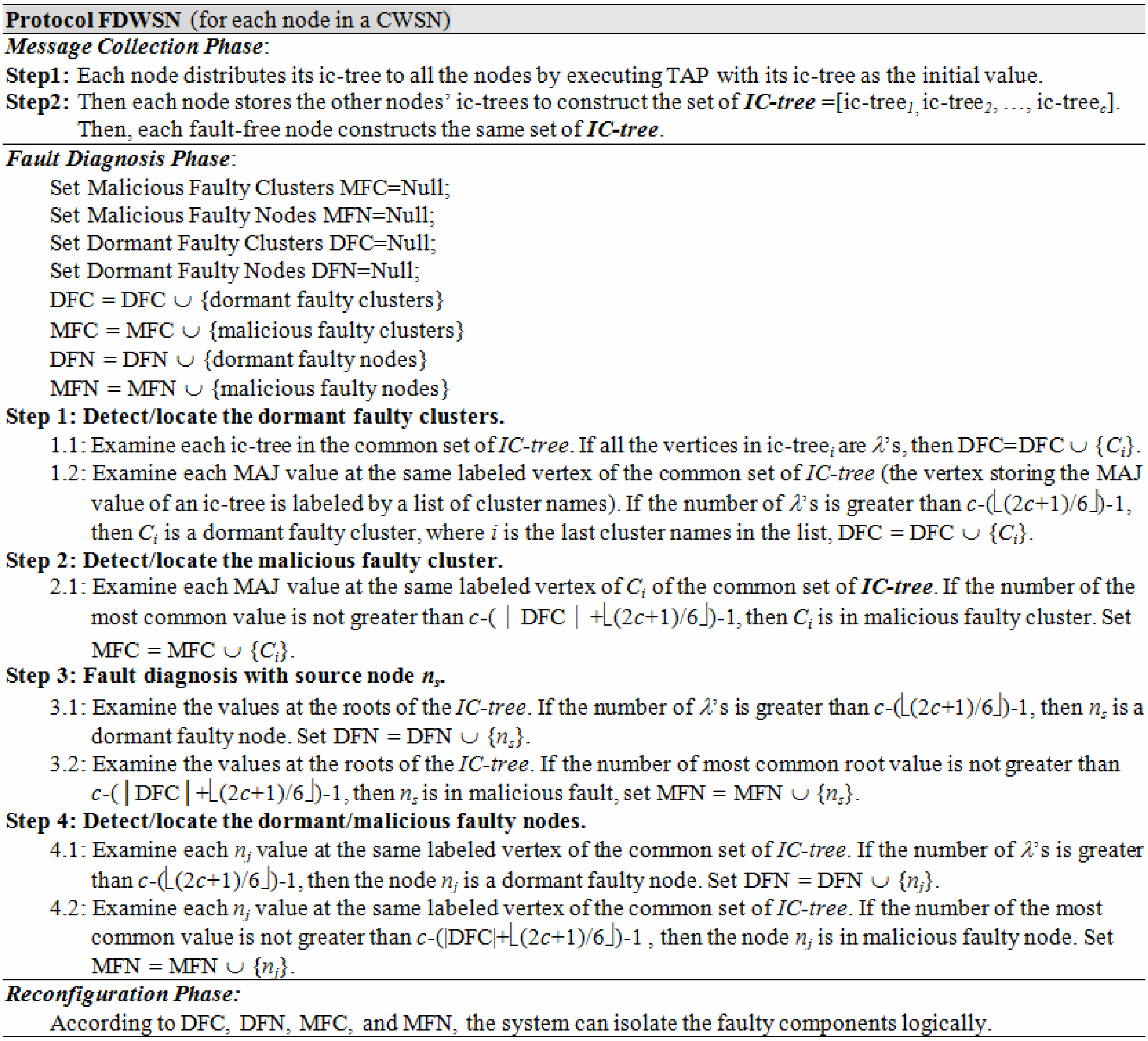
(e) The ic-tree of node n1

VOTE($s1$) = (VOTE($s12$), VOTE($s13$), VOTE($s14$), VOTE($s15$), VOTE($s16$), VOTE($s17$), VOTE($s18$))=(0, 0, 0, 0, 0, 1, λ^0)=0
VOTE($s2$) = (VOTE($s21$), VOTE($s23$), VOTE($s24$), VOTE($s25$), VOTE($s26$), VOTE($s27$), VOTE($s28$))=(1, 1, 1, 1, 1, 0, λ^0)=1
VOTE($s3$) = (VOTE($s31$), VOTE($s32$), VOTE($s34$), VOTE($s35$), VOTE($s36$), VOTE($s37$), VOTE($s38$))=(0, 0, 0, 0, 0, 0, λ^0)=0
VOTE($s4$) = (VOTE($s41$), VOTE($s42$), VOTE($s43$), VOTE($s45$), VOTE($s46$), VOTE($s47$), VOTE($s48$))=(1, 1, 1, 1, 1, 1, λ^0)=1
VOTE($s5$) = (VOTE($s51$), VOTE($s52$), VOTE($s53$), VOTE($s54$), VOTE($s56$), VOTE($s57$), VOTE($s58$))=(1, 1, 1, 1, 1, 0, λ^0)=1
VOTE($s6$) = (VOTE($s61$), VOTE($s62$), VOTE($s63$), VOTE($s64$), VOTE($s65$), VOTE($s67$), VOTE($s68$))=(1, 1, 1, 1, 1, 1, λ^0)=1
VOTE($s7$) = (VOTE($s71$), VOTE($s72$), VOTE($s73$), VOTE($s74$), VOTE($s75$), VOTE($s76$), VOTE($s78$))=(0, 1, 0, 1, 0, 1, λ^0)=ϕ
VOTE($s8$) = (VOTE($s81$), VOTE($s82$), VOTE($s83$), VOTE($s84$), VOTE($s85$), VOTE($s86$), VOTE($s87$))=(0, 1, 0, 1, 0, 1, λ^0)=ϕ

VOTE(s) = (VOTE($s1$), VOTE($s2$), VOTE($s3$), VOTE($s4$), VOTE($s5$), VOTE($s6$), VOTE($s7$), VOTE($s8$))
= (0, 1, 0, 1, 1, 1, ϕ, ϕ) = 1

(f) The common value VOTE(s) of node n_1

Figure 3. An example of TAP (continue)



```
Protocol FDWSN (for each node in a CWSN)
Message Collection Phase:
Step1: Each node distributes its ic-tree to all the nodes by executing TAP with its ic-tree as the initial value.
Step2: Then each node stores the other nodes' ic-trees to construct the set of IC-tree =[ic-tree$_1$, ic-tree$_2$, ..., ic-tree$_c$].
       Then, each fault-free node constructs the same set of IC-tree.
Fault Diagnosis Phase:
    Set Malicious Faulty Clusters MFC=Null;
    Set Malicious Faulty Nodes MFN=Null;
    Set Dormant Faulty Clusters DFC=Null;
    Set Dormant Faulty Nodes DFN=Null;
    DFC = DFC ∪ {dormant faulty clusters}
    MFC = MFC ∪ {malicious faulty clusters}
    DFN = DFN ∪ {dormant faulty nodes}
    MFN = MFN ∪ {malicious faulty nodes}
Step 1: Detect/locate the dormant faulty clusters.
    1.1: Examine each ic-tree in the common set of IC-tree. If all the vertices in ic-tree$_i$ are $\lambda$'s, then DFC=DFC ∪ {$C_i$}.
    1.2: Examine each MAJ value at the same labeled vertex of the common set of IC-tree (the vertex storing the MAJ
         value of an ic-tree is labeled by a list of cluster names). If the number of $\lambda$'s is greater than $c-(\lfloor(2c+1)/6\rfloor)-1$,
         then $C_i$ is a dormant faulty cluster, where $i$ is the last cluster names in the list, DFC = DFC ∪ {$C_i$}.
Step 2: Detect/locate the malicious faulty cluster.
    2.1: Examine each MAJ value at the same labeled vertex of $C_i$ of the common set of IC-tree. If the number of the
         most common value is not greater than $c-(|DFC|+\lfloor(2c+1)/6\rfloor)-1$, then $C_i$ is in malicious faulty cluster. Set
         MFC = MFC ∪ {$C_i$}.
Step 3: Fault diagnosis with source node $n_s$.
    3.1: Examine the values at the roots of the IC-tree. If the number of $\lambda$'s is greater than $c-(\lfloor(2c+1)/6\rfloor)-1$, then $n_s$ is a
         dormant faulty node. Set DFN = DFN ∪ {$n_s$}.
    3.2: Examine the values at the roots of the IC-tree. If the number of most common root value is not greater than
         $c-(|DFC|+\lfloor(2c+1)/6\rfloor)-1$, then $n_s$ is in malicious fault, set MFN = MFN ∪ {$n_s$}.
Step 4: Detect/locate the dormant/malicious faulty nodes.
    4.1: Examine each $n_j$ value at the same labeled vertex of the common set of IC-tree. If the number of $\lambda$'s is greater
         than $c-(\lfloor(2c+1)/6\rfloor)-1$, then the node $n_j$ is a dormant faulty node. Set DFN = DFN ∪ {$n_j$}.
    4.2: Examine each $n_j$ value at the same labeled vertex of the common set of IC-tree. If the number of the most
         common value is not greater than $c-(|DFC|+\lfloor(2c+1)/6\rfloor)-1$ , then the node $n_j$ is in malicious faulty node. Set
         MFN = MFN ∪ {$n_j$}.
Reconfiguration Phase:
    According to DFC, DFN, MFC, and MFN, the system can isolate the faulty components logically.
```

Figure 4. The proposed protocol FDWSN

4.1 The Messages Collected Phase

In the *message collection phase*, each fault-free node collects all the nodes' ic-trees in the TAP as evidence. In order to make sure the fault diagnosis result of each fault-free node is the same, each fault-free node should collect the same evidence (the common set of *IC-tree*). Hence, in the FDWSN, each node distributes its ic-tree to all the nodes by executing TAP with its ic-tree as the initial value.

4.2 The Fault Diagnosis Phase

In this phase, the collected IC-trees are examined to detect/locate the dormant and malicious faulty components. The sets of MFC, DFC, MFN and DFN are used to record the malicious faulty clusters, dormant faulty clusters, malicious faulty nodes and dormant faulty nodes, and the examination sequence by each fault-free node is top-down and level by level.

Step 1: Detect/Locate the dormant faulty clusters. First, each fault-free node detects/locates the dormant faulty clusters by examining each ic-tree in the common set of *IC-tree*. If all the vertices in ic-tree$_i$ are λ, then C_i is a dormant faulty cluster, and then DFC=DFC ∪ {C_i}.

Second, each fault-free node examines each MAJ value at the same labeled vertex of the common set of *IC-tree* (the vertex storing the MAJ value of an ic-tree is labeled by a list of cluster names). If the number of λ's is greater than $c-(\lfloor(2c+1)/6\rfloor)-1$, then C_i is a dormant faulty cluster, where i is the last cluster name in the list, and DFC = DFC ∪ {C_i}.

Step 2: Detect/Locate the malicious faulty clusters. The protocol examines each MAJ value at the same labeled vertex of C_i of the common set of *IC-tree*. If the most common value does not appear more than $c-(|DFC|+\lfloor(2c+1)/6\rfloor)-1$ times, then C_i is a malicious faulty cluster, and then MFC = MFC∪{C_i}.

Step 3: Fault diagnosis with source node n_s. The protocol examines all the values at the roots of the *IC-*

tree. If the number of λ's is greater than c-($\lfloor(2c+1)/6\rfloor$)-1, then n_s is a dormant faulty node, and the system sets DFN = DFN$\cup\{n_s\}$. If the most common root value does not show up more than c-(|DFC|+$\lfloor(2c+1)/6\rfloor$)-1 times, then n_s is a malicious faulty node, and then MFN= MFN $\cup$ $\{n_s\}$.

Step 4.1: Detect/Locate the dormant faulty nodes. The protocol examines each n_j value at the same labeled vertex of the common set of *IC-trees*, if the number of λ's is greater than c-($\lfloor(2c+1)/6\rfloor$)-1, then the node n_j is a dormant faulty node, and then DFN = DFN$\cup\{n_j\}$.

Step 4.2: Detect/Locate the malicious faulty nodes. FDWSN examines each n_j value at the same labeled vertex of the common set of *IC-trees*. If the most common value appears more than c-(|DFC|+ $\lfloor(2c+1)/6\rfloor$)-1 times, then the node n_j is a malicious faulty node, and then MFN = MFN $\cup$ $\{n_j\}$.

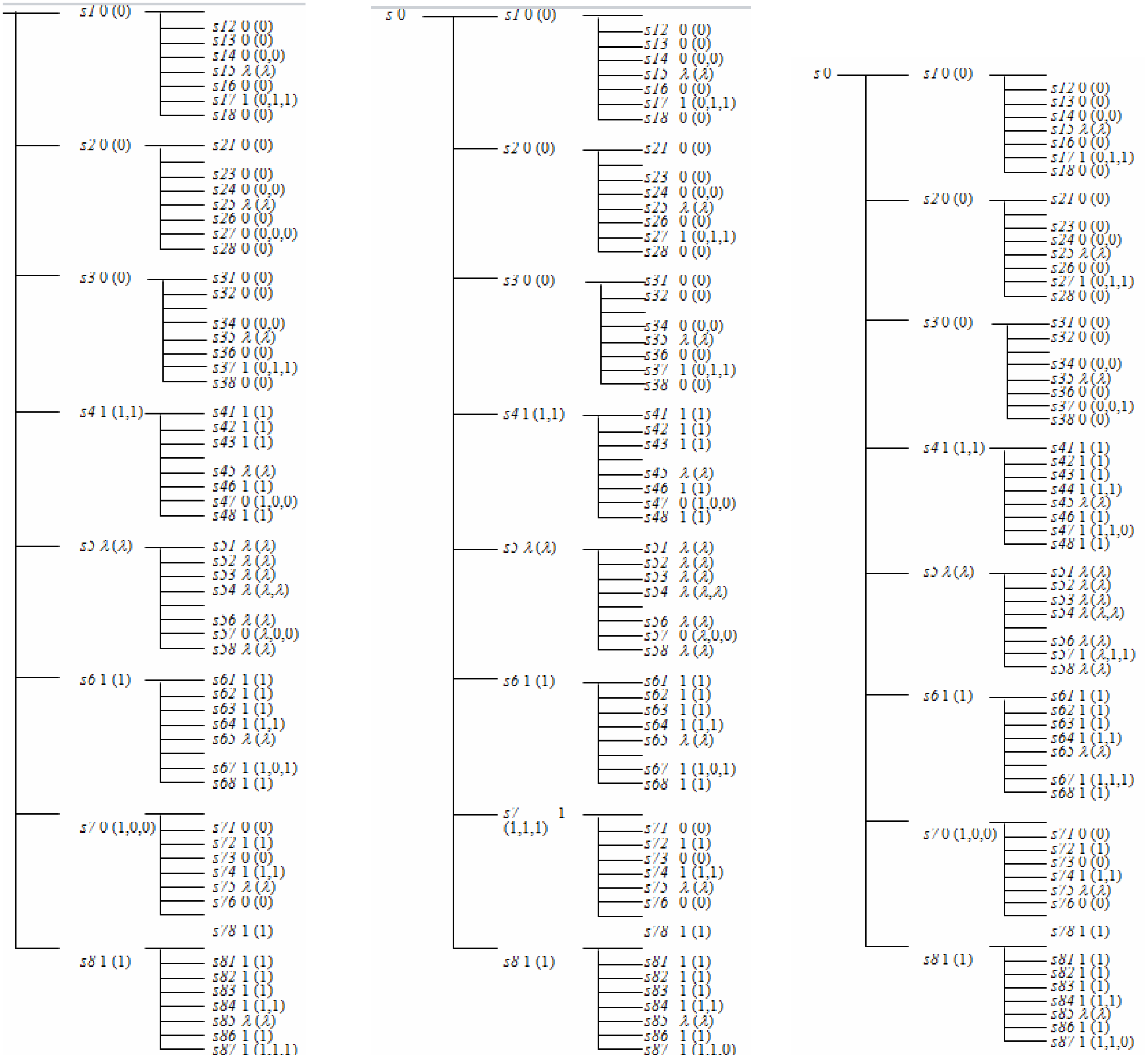
4.3 The Reconfiguration Phase

Tthe results of MFC, DFC, MFN and DFN from the *fault diagnosis phase* are used to reconfigure the network by isolating the faulty components logically. After the reconfiguration, the performance and integrity of the network can be guaranteed.

An example of FDWSN executed is given here. First, FDWSN collects all the nodes' ic-trees as evidence in the *message collection phase*.

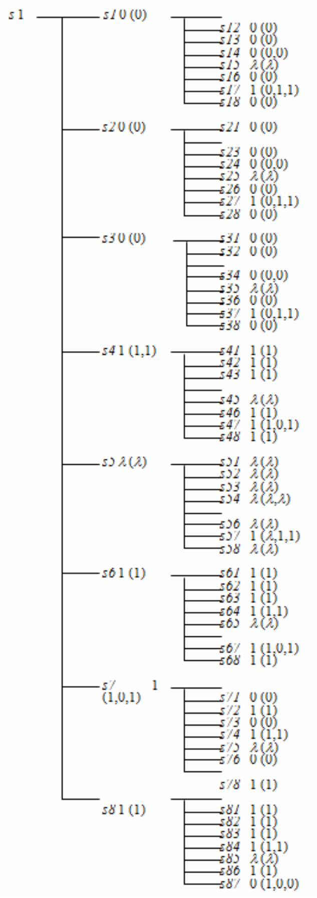
4.4 The Messages Collected Phase

Each node distributes its ic-tree form the example back in Section 3 to all the nodes in the *message collection phase*. Then each fault-free node constructs the common set of *IC-tree* as [ic-tree$_1$, ic-tree$_2$, …, ic-tree$_8$], as shown in Figure 5.

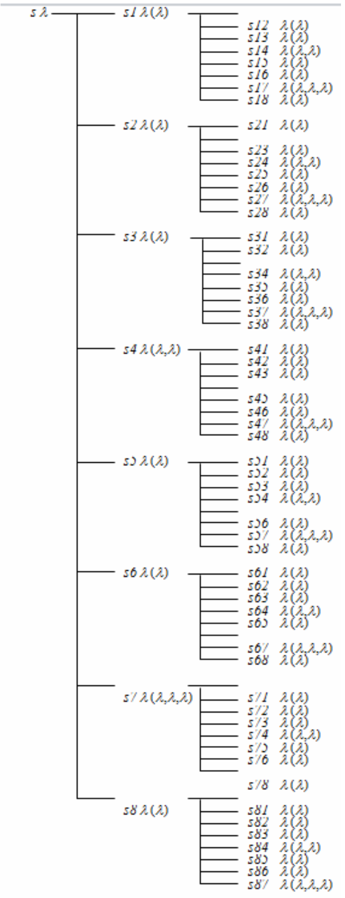


The ic-tree$_1$ from C_1's node The ic-tree$_2$ from C_2's node The ic-tree$_3$ from C_3's node

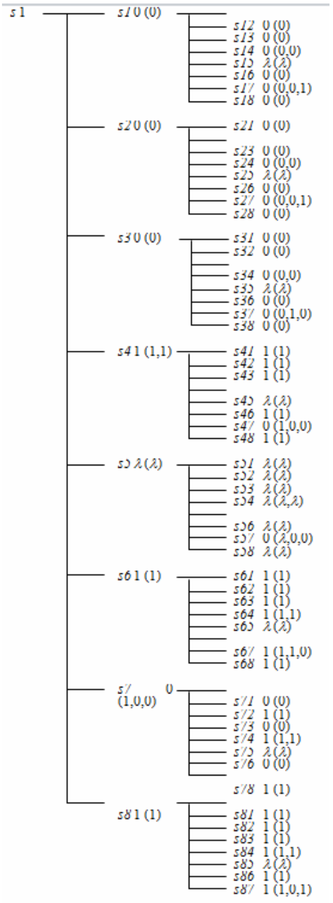
Figure 5. The common set of the IC-tree by each fault-free node



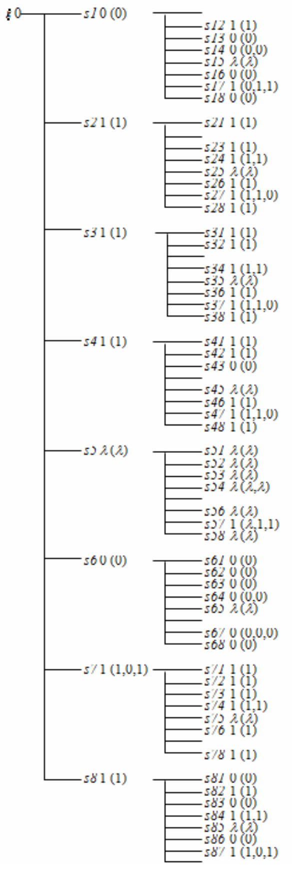
The ic-tree$_4$ from C_4's nodes



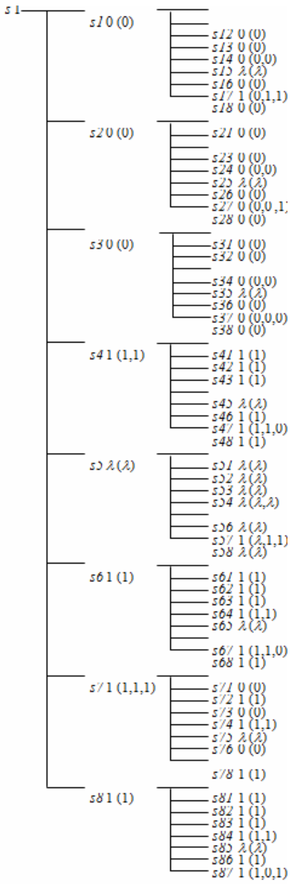
The ic-tree$_5$ from C_5's node



The ic-tree$_6$ from C_6's node



The ic-tree$_7$ from C_7's nodes



The ic-tree$_8$ from C_8's node

Figure 5. (continue)

4.5 The Fault Diagnosis Phase

Each fault-free node can detect/locate the common set of faulty components.

Step 1: An example of detecting/locating dormant faulty clusters. By Steps 1.1 and 1.2, each fault-free node can detect/locate the dormant faulty clusters. By Step 1.1, each fault-free node can detect/locate that C_5 is a dormant faulty cluster because all the vertices in ic-tree$_5$ are λ's. Then, DFC is set to be DFC $\cup$ $\{C_5\}$. By Step 1.2, each node can also detect/locate the dormant faulty cluster C_5. For example, the MAJ values at the vertex *s5* are (λ,λ,λ,λ,λ,λ,λ,λ). The number of λ's is greater than c-($\lfloor(2c+1)/6\rfloor$)-1 =8-2-1=5.

Step 2: An example of detecting/locating malicious faulty clusters. By Step 2.1, each fault-free node can detect/locate the malicious faulty clusters. For example, the MAJ values at the vertex *s7* are (0,1,0,1,λ,0,1,1). The most common value does not appear more than c-(|DFC|+$\lfloor(2c+1)/6\rfloor$)-1 (8-(1+2)-1=4) times. Therefore, C_7 is a malicious faulty cluster. The system sets MFC=MFC $\cup$ $\{C_7\}$.

Step 3: An example of fault diagnosis with source node n_s. By Step 3.1, the root values of the *IC-tree* are (0,0,0,1,λ,1,0,1). The number of most common root value is not greater than c-(|DFC|+$\lfloor(2c+1)/6\rfloor$)-1 =8-(1+2)-1=4. Therefore, n_s is a malicious faulty node. The system sets MFN = MFN $\cup$ $\{n_s\}$.

Step 4.1: An example of detecting/locating dormant faulty nodes. By Step 4.1, the values of n_6 at the vertex *s5* are (λ,λ,λ,λ,λ,λ,λ,λ). The number of λ's is greater than c-($\lfloor(2c+1)/6\rfloor$)-1=8-2-1=5, and so n_6 is a dormant faulty node. Then, DFN= DFN $\cup$ $\{n_6\}$.

Step 4.2: An example of detecting/locating malicious faulty nodes. By Step 4.2, the values of n_{10} at the vertex *s7* are (0,1,0,1, λ,0,1,1). The most common value does not appear more than c-(|DFC|+$\lfloor(2c+1)/6\rfloor$)-1=8-(1+2)-1 =4 times. Therefore, the node n_{10} is a malicious faulty node, and MFN = MFN $\cup$ $\{n_{10}\}$.

Following all the steps in the *fault diagnosis phase*, each fault-free node can detect/locate the dormant faulty cluster *C5*, the malicious faulty clusters C_1 and C_7, the dormant faulty node n_6, and the malicious faulty nodes n_s, n_9 and n_{10}.

4.6 The Reconfiguration Phase

Finally, each fault-free node isolates n_s, n_s, n_9 and n_{10} logically to reconfigure the network as shown in Figure 6.

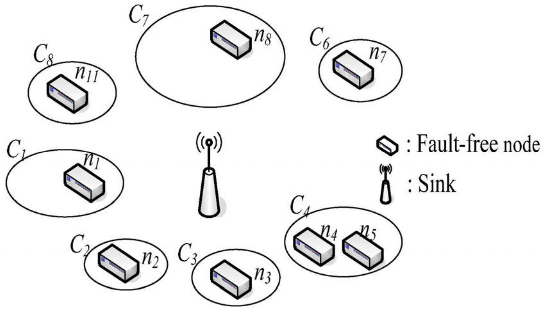


Figure 6. A CWSN after reconfigured (n=8, c=7)

5 Correctness and Complexity of FDWSN

The following lemmas and theorems are used to prove the correctness and complexity of FDWSN.

Lemma 1. Each fault-free node receives the same common set of *IC-tree* as evidence in the message collection phase if c> $\lfloor(c$-1)/3$\rfloor$+2f_{mc}+f_{dc} and c-1>2f_{mc}+f_{dc}.

Proof: The BA protocol can make each fault-free node agree on a single common value no matter whether the source node is fault-free or not. Hence, each node can reliably distribute its ic-tree to all the other nodes by executing TAP with its ic-tree as the initial value (many copies of TAP can be executed in parallel). Hence, each fault-free node can receive the same common set of *IC-tree*.

Lemma 2. Each fault-free node can detect/locate the same faulty components.

Proof: Each fault-free node receives the same evidence by Lemma 1 and uses the same FDA protocol FDWSN, so each fault-free node will surely detect/locate the same faulty components.

Theorem 1. Protocol FDWSN satisfies the agreement of FDA.

Proof: By Lemma 1 and 2, all the fault-free nodes identify the common set of faulty nodes.

Lemma 3. All the dormant faulty nodes/clusters can be detected/located.

Proof: Each fault-free receiver node can detect the message(s) through dormant faulty nodes if the protocol appropriately encodes transmitted messages by using the Manchester code [7] before transmission. Since message(s) through dormant faulty components can be detected, if the all the vertices in ic-tree$_i$ are λ, then the C_i is a dormant faulty cluster. The dormant faulty cluster also can be detected by examine each MAJ value at the same labeled vertex of the common set of *IC-tree*. If the number of value λ is greater than c-($\lfloor(2c+1)/6\rfloor$)-1, then C_i is a dormant faulty cluster. The reason is that there are at most $\lfloor(2c+1)/6\rfloor$ malicious faulty clusters in the network by the constraints of c>$\lfloor(c$-1)/3$\rfloor$+2f_{mc}+f_{dc} and c-1>2f_{mc}+f_{dc}.

Lemma 4. The malicious faulty nodes/clusters can be detected/located if c>$\lfloor(c$-1)/3$\rfloor$+2f_{mc}+f_{dc} and c-1>2f_{mc} +f_{dc}.

Proof: Due to the constraint $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$ and $c\text{-}1>2f_{mc}+f_{dc}$, there are at most f_{dc} dormant faulty clusters. By Lemma 3, all the dormant faulty clusters can be detect/located by each fault-free node, so f_{dc} =|DFC|. By the same constraint, there are at most f_{mc} malicious faulty clusters, so there are at most f_{mc} values (except λ) at the same labeled vertex in the *IC-tree* different from the most common value, that is $c\text{-}\lfloor(c\text{-}1)/3\rfloor\text{-}|\mathrm{DFC}|>2f_{mc}$, $f_{mc}<\lfloor(2c+1)/6\rfloor\text{-}|\mathrm{DFC}|$. So, if the most common value does not appear at the same labeled vertex in the *IC-tree* more than $c\text{-}(|\mathrm{DFC}|+\lfloor(2c+1)/6\rfloor)\text{-}1(c\text{-}(f_{dc}+f_{mc})\ \text{-}1)$ times, then the component is in malicious fault.

Theorem 2. Protocol FDWSN satisfies the fairness requirement of FDA.

Proof: By Lemma 3 and Lemma 4, no fault-free node is falsely detected as faulty by any fault-free nodes if $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$ and $c\text{-}1>2f_{mc}+f_{dc}$.

Theorem 3. Protocol FDWSN solves the FDA problem in a CWSN if $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$ and $c\text{-}1>2f_{mc}+f_{dc}$.

Proof: By theorem 1 and theorem 2, this theorem is proved.

Theorem 4. The maximum number of detectable/ locatable faulty components by FDWSN is f_{mc} malicious faulty clusters and f_{dc} dormant faulty clusters, where $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$ and $c\text{-}1>2f_{mc}+f_{dc}$.

Proof: In Siu et al. [10] indicates the constraints of BA problem for node faults only is $n>\lfloor(n\text{-}1)/3\rfloor+2f_{mn}+f_{dn}$, $c>2f_{mn}+f_{dn}$ and the unit of Siu et al. is node. But, the unit of CWSN is cluster, so we can suppose a node in Siu *et al.* as a cluster in CWSN. Therefore, $n>\lfloor(n\text{-}1)/3\rfloor+2f_{mn}+f_{dn}$ and $c\text{-}1>2f_{mn}+f_{dn}$ in Siu *et al.* [10] imply $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$ *and* $c\text{-}1>2f_{mc}+f_{dc}$ in CWSN. So the total number of detectable/locatable faulty components by FDWSN is f_{mc} malicious faulty clusters and f_{dc} dormant faulty clusters, which is maximum if $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$ and $c\text{-}1>2f_{mc}+f_{dc}$.

Theorem 5. The number of detectable/locatable faulty nodes f_n ($f_n=f_{mn}+f_{dn}$) is the maximum.

Proof: For a cluster, each fault-free node agrees on a value, which is dominated by most of nodes. If the number of faulty nodes exceeds 1/2 in the cluster, it is a faulty cluster; otherwise, it is a fault-free cluster. Two condition of the fault detectable/locatable capability will be discussed that include the best case and the worst case. The best case means that there is the maximum number of faulty nodes in a CWSN and no more faulty node can be increased, the worst case means that if a faulty node is increased in any non-faulty cluster will let the non-faulty cluster be a faulty cluster.

In the best case, the fault detectable/locatable capability of malicious fault and fault detectable/locatable capability of dormant fault are discussed.

[Case B1]: First, the fault detectable/locatable capability of malicious fault in the best case is discussed. Let $\mu_{\max\text{-}j}$ be the number of nodes in the j-th maximum cluster and $\mu_{\min\text{-}j}$ be the number of nodes in the j-th minimum cluster. If there are no dormant faulty nodes, then the number of malicious faulty nodes will be the maximum. That is, the number of malicious faulty nodes in malicious faulty clusters is $\sum_{i=1}^{f_{mc}}\mu_{\max-i}$, because in the best case the number of malicious faulty nodes is that all the nodes in the malicious faulty cluster are all failed. An additional number of malicious faulty nodes $\sum_{j=f_{mc}+1}^{c}\lfloor(\mu_{\max-j}-1)/2\rfloor$ have no influence to the system, and no malicious faulty node can be increased. Because if a malicious faulty node is increased, then a malicious faulty cluster will be increased, that violates the assumption $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$. As a result, the number of detectable/locatable malicious faulty nodes, say $f_{mn}=\sum_{i=1}^{f_{mc}}\mu_{\max-i}+\sum_{j=f_{mc}+1}^{c}\lfloor(\mu_{\max-j}-1)/2\rfloor$ is the maximum number of detectable/locatable malicious faulty nodes in the best case.

[Case B2]: Next, the fault detectable/locatable capability of dormant fault in the best case is discussed.

If there are no malicious faulty nodes, then the number of dormant faulty nodes will be the maximum. That is, the number of dormant faulty nodes in dormant faulty clusters is $\sum_{i=1}^{f_{dc}}\mu_{\max-i}$, because in the best case the number of dormant faulty nodes is that all the nodes in the dormant faulty cluster are all failed. An additional number of dormant faulty nodes $\sum_{j=f_{dc}+1}^{c}\lfloor(\mu_{\max-j}-1)/2\rfloor$ have no influence to the system, and no dormant faulty node can be increased. Because if a dormant faulty node is increased, then a dormant faulty cluster will be increased, that violates the assumption that $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$. As a result, the number of detectable/ locatable dormant faulty nodes, say $f_{dn}=\sum_{i=1}^{f_{dc}}\mu_{\max-i}+\sum_{j=f_{dc}+1}^{c}\lfloor(\mu_{\max-j}-1)/2\rfloor$ is the maximum number of detectable/locatable dormant faulty nodes in the best case.

In the worst case, the fault detectable/locatable capability of malicious fault and fault detectable/locatable capability of dormant fault are also discussed.

[Case W1]: First, the fault detectable/locatable capability of malicious fault in the worst case is discussed.

If there are no dormant faulty nodes, then the number of malicious faulty nodes will be the maximum. The number of malicious faulty nodes in malicious faulty clusters is $\sum_{i=1}^{f_{mc}}\lceil(\mu_{\min-i})/2\rceil$, because if the number of malicious faulty nodes in the cluster exceeds $\lceil(\mu_{\min-i})/2\rceil$, then the cluster will be the malicious faulty cluster, so f_{mc} malicious faulty clusters will be $\sum_{i=1}^{f_{mc}}\lceil(\mu_{\min-i})/2\rceil$ in the worst case. An additional number of malicious faulty nodes $\lfloor(\mu_{\min-f_{mc}+1}-1)/2\rfloor$ have no influence to the system. If a malicious faulty node is increased in the $C_{\min\text{-}n}$, then a malicious faulty cluster will be increased and it will violate the assumption that $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$. As a result, the number of detectable/locatable malicious faulty nodes, say $f_{mn}=\sum_{i=1}^{f_{mc}}\lceil(\mu_{\min-i})/2\rceil+\lfloor(\mu_{\min-f_{mc}+1}-1)/2\rfloor$ is the maximum number of detectable/locatable malicious faulty nodes in the worst case.

[Case W2]: Next, the fault detectable/locatable capability of dormant fault in the worst case is discussed.

If there are no malicious faulty nodes, then the number of dormant faulty nodes will be the maximum. The number of dormant faulty nodes in dormant faulty clusters is $\sum_{i=1}^{f_{dc}}\lceil(\mu_{\min-i})/2\rceil$, because if the number of dormant faulty nodes in the cluster exceeds $\lceil(\mu_{\min-i})/2\rceil$ then the cluster will be the dormant faulty cluster, so f_{dc} dormant cluster will be at least $\sum_{i=1}^{f_{dc}}\lceil(\mu_{\min-i})/2\rceil$ in the worst case. An additional number of dormant faulty nodes $\lfloor(\mu_{\min-f_{dc}+1}-1)/2\rfloor$ have no influence to the system. If a dormant faulty node is increased in the $C_{\min\text{-}n}$, then a dormant faulty cluster will be increased and it will violate the assumption that $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$. As a result, the number of detectable/locatable dormant faulty nodes, say $f_{dn}=\sum_{i=1}^{f_{dc}}\lceil(\mu_{\min-i})/2\rceil+\lfloor(\mu_{\min-f_{dc+1}}-1)/2\rfloor$ is the maximum number of detectable/locatable dormant faulty nodes in the worst case.

According to the cases B1, B2, W1 and W2, the number of detectable/locatable faulty nodes f_n ($f_n=f_{mn}+f_{dn}$) is the maximum.

6 Conclusion and Future Works

When an ultra large disaster occurs, the WSNs of IoT might be damaged. However, it is significant that the sensing information sensor nodes should be transferred to the related applications of IoT.

In this study, the fault-diagnosis based WSN is studied that the proposed FDA protocol FDWSN can detect/locate the maximum number of faulty components with the dual failure mode in a CWSN. The fault detecting/locating capability of FDWSN is shown in Table 1, where the general case is a usual situation. The best case means that there is the maximum number of faulty nodes in a CWSN and no more faulty nodes can be increased. The worst case means that if we increase a faulty node in any non-faulty cluster will let the non-faulty cluster be a faulty cluster.

Table 1. The detectable/locatable faulty nodes by FDWSN ($f_n=f_{mn}+f_{dn}$, where $c>\lfloor(c\text{-}1)/3\rfloor+2f_{mc}+f_{dc}$)

	General Case	Worst Case	Best Case
f_{mn}	$f_{mc}*\lceil\mu_{\min}/2\rceil\le f_{mn}\le$ $f_{mc}*\mu_{\min}+(c-f_{mc}-f_{dc})*\lfloor(\mu_{\min}-1)/2\rfloor$	$f_{mn}=\sum_{i=1}^{f_{mc}}\lceil(\mu_{\min-i})/2\rceil+\lfloor(\mu_{\min-f_{mc}+1}-1)/2\rfloor$ (if $f_{dn}=0$)	$f_{mn}=\sum_{i=1}^{f_{mc}}\mu_{\max-i}+\sum_{j=f_{mc}+1}^{c}\lfloor(\mu_{\max-j}-1)/2\rfloor$ (if $f_{dn}=0$)
f_{dn}	$f_{dc}*\lceil\mu_{\min}/2\rceil\le f_{dn}\le$ $f_{dc}*\mu_{\min}+(c-f_{mc}-f_{dc})*\lfloor(\mu_{\min}-1)/2\rfloor$	$f_{dn}=\sum_{i=1}^{f_{dc}}\lceil(\mu_{\min-i})/2\rceil+\lfloor(\mu_{\min-f_{dc}+1}-1)/2\rfloor$ (if $f_{mn}=0$)	$f_{dn}=\sum_{i=1}^{f_{dc}}\mu_{\max-i}+\sum_{j=f_{dc}+1}^{c}\lfloor(\mu_{\max-j}-1)/2\rfloor$ (if $f_{mn}=0$)

Note. Let $\mu_{\min}$ be the minimal number of nodes in all clusters.

Let $\mu_{\min-j}$ be the number of nodes in the *j*-th minimalclusters.

Let $\mu_{\max-j}$ be the number of nodes in the *j*-th minimalclusters.

The comparision of various FDA protocols underlying different network topologies with different failure type assumptions on fallible components are shown in Table 2. In short, the proposed protocol FDWSN cannot only reach an agreement from fault-free nodes but also detect and locate the faulty components in an unreliable CWSN. Therefore, the proposed protocol can enlarge the fault tolerance capability by allowing malicious faults exist in a network. That is, FDWSN can tolerate, detect and locate the maximum number of faulty components with the malicious and dormant failure mode to solve the fault diagnosis agreement problem in a CWSN by minimum number of rounds of message exchanges.

Since FDWSN is designed to detect/locate faulty nodes underlying a CWSN, the communication media faults are treated as node faults. However, this would decrease the fault detection/location ability. Therefore, our future works will be focused on solving the FDA problem with the dual failure mode on both nodes and communication media in a CWSN.

Table 2. The comparison of various FDA protocols

	Network Topology		Failure Types of Fallible Nodes		
	Wired Network	Wireless Network	Dormant	Malicious	Dual FailureMode
Chiang *et al.* [6]		◆		◆	
Siu *et al.* [10]	◆		◆	◆	◆
FDWSN		◆	◆	◆	◆

References

[1] A. E. Kouche, Towards a Wireless Sensor Network Platform for the Internet of Things: Sprouts WSN Platform, *2012 IEEE International Conference on Communications (ICC)*, Ottawa, ON, Canada, 2012, pp. 632-636.

[2] J. Zhang, V. Varadharajan, Wireless Sensor Network Key Management Survey and Taxonomy, *Journal of Network and Computer Applications*, Vol. 33, No. 2, pp. 63-75, March, 2010.

[3] I. Gupta, D. Riordan, S. Sampalli, Cluster-head Election using Fuzzy Logic for Wireless Sensor Networks, *the 3rd Annual Conference on Communication Networks and Services Research*, Halifax, NS, Canada, 2005, pp. 255-260.

[4] L. Lamport, R. Shostak, M. Pease, The Byzantine Generals Problem, *ACM Transactions on Programming Language and Systems*, Vol. 4, No. 3, pp. 382-401, July, 1982.

[5] A. Khosravi, Y. Kavian, Fault-diagnosis and Decision Making Algorithm for Determining Faulty Nodes in Malicious Networks, in: H. Sharif, Y. S. Kavian (Eds.), *Technological Breakthroughs in Modern Wireless Sensor Applications*, IGI Global, 2015, pp. 207-223,.

[6] M. L. Chiang, H. C. Hsieh, A New Approach to the Fault Detection Problem for Mobile P2P Network, *Information Technology and Control*, Vol. 41, No. 2, pp. 151-161, June, 2012.

[7] N. Olifer, V. Olifer, *Computer Network: Principles, Technologies and Protocols for Network Design*, John Wiley & Sons, 2006.

[8] S. S. Wang, S. C. Wang, K. Q. Yan, Reaching Trusted Byzantine Agreement in a Cluster-based Wireless Sensor Network, *Wireless Personal Communications*, Vol. 78, No. 2, pp. 1079-1094, September, 2014.

[9] P. Verıśsimo, A. Casimiro, The Timely Computing base Model and Architecture, *IEEE Transactions on Computers*, Vol. 51, No. 8, pp. 916-930, August, 2002.

[10] H. S. Siu, Y. H. Chin, W. P. Yang, Reaching Fault Diagnosis Agreement under a Hybrid Fault Model, *IEEE Transactions on Computers*, Vol. 49, No. 9, pp. 980-986, September, 2000.

Biographies

Shu-Ching Wang received her Ph.D. in Information Engineering from National Chiao-Tung University, Taiwan. Currently, she is a Professor of Chaoyang University of Technology, Taiwan. Her current research interests include distributed data processing, grid computing, and cloud computing.

Mao-Lun Chiang received the Ph.D. degree in Department of Computer Science from National Chung-Hsing University, Taiwan. He is an associate professor of the Chaoyang University of Technology, Taiwan. His current research interests include mobile computing, distributed processing, fault tolerant, and cloud computing.

Kuo-Qin Yan received his Ph.D. in Computer Sciences from National Tsing-Hua University, Taiwan. Currently, he is a Professor of Chaoyang University of Technology, Taiwan. His current research interests include distributed fault tolerant computing, mobile computing, and cloud computing.

Yao-Te Tsai is an assistant professor of Feng Chia University. He received his Ph.D. in Industrial and Systems Engineering from Auburn University in 2015. His research interests include internet of things, transportation safety, data analytics, supply chain management, and operations management.