A Feature-Oriented Fault Diagnosis Agreement Protocol in Distributed Systems

Hui-Ching Hsieh[1], Mao-Lun Chiang[2], Wen-Chung Tsai[2], Yen-Chiu Chen[3]

[1] Department of Information Communication, Hsing Wu University, Taiwan
[2] Department of Information and Communication Engineering, Chaoyang University of Technology, Taiwan
[3] Department of Information Management, Chung Hua University, Taiwan
luckyeva.hsieh@gmail.com, {mlchiang, azongtsai }@cyut.edu.tw, yhchenedu@gmail.com

Abstract

Understanding the fault-tolerance of distributed systems is crucial for achieving reliability. One of the most important issues surrounding fault-tolerance is the Fault Diagnosis Agreement (FDA) problem. The purpose of FDA is to help each fault-free processor detecting/locating a common set of faulty processors. In general, FDA protocols need $\lfloor(n\text{-}1)/3\rfloor+2$ rounds of message exchange to detect or locate faulty components, regardless of the presence or absence of faulty processors. However, the number of messages produced results in a large protocol overhead. To solve the FDA problems efficiently, a novel concept of *feature value* is proposed in our algorithm to reach an agreement using the minimum constant number of rounds characterized by the presence of dual failure characteristics of processors. In addition, the proposed protocol can detect/locate the maximum possible number of faulty processors in a network.

Keywords: Byzantine agreement, Fault diagnosis agreement, Fault-tolerance, Rule based diagnosis

1 Introduction

In a distributed computing system, processors allocated to different places, or units are connected to create greater power and ability. To achieve reliability in distributed computing systems, the Byzantine Agreement (BA) [1, 3-7, 9-10, 12-14, 30] must be considered. The BA problem, first studied by Lamport in 1982 [15], was solved to make a distributed system run and agree on a common value even if a given number of processors in the system fail. The general model of the problem describes a system containing n communicating processors of which at most f_m [15] are corrupted (f_m represents the number of faulty processors and $f_m \le \lfloor(n\text{-}1)/3\rfloor$). Each processor will agree on a common value if the number of faulty components is less than the fault tolerant boundary f_m. The goal of BA is achieved if a number of independent processors reach agreement in cases where some of those processors might be faulty. The faulty symptoms (*fau_sym*) of processor failure can be classified into two categories: dormant faults and malicious faults [2, 6-7, 11, 29]. The dormant faults of processors is easy to detect and to solve. Furthermore, a dormant fault is detectable by other processors if a transmitted message is encoded appropriately by either the Non-Return-to-Zero code or the Manchester code [16] before transmitting. However, malicious faults are unpredictable and damaged. The malicious faulty processor can withhold messages to be sent and can send irregular messages, or it can collude with other malicious faulty processors to send false messages. Thus, malicious faults are more serious than dormant faults. The following applications are examples of applications that need healthy processors to achieve a common value: the commitment problem in a distributed database system [6, 17], the clock synchronization problem [7], and a landing task controlled by a flight-path finding system [1].

Basically, an agreement is reached if all healthy processors agree upon a common value. Thus, various protocols for the BA problem should meet the following requirements [1, 7, 10, 12-14, 18-20, 30]:

(BA1) Agreement: All healthy processors should agree on a common value v.

(BA2) Validity: If the initial value of the source is v_s, and the source is fault-free, then all healthy processors must agree on the value v_s; i.e., $v = v_s$.

A closely related and important sub-issue, the Fault Diagnosis Agreement (FDA) problem [18, 21-23], is also in need of review. In general, the FDA problem can be divided into two models: *test-based* approaches [15, 21, 24-26] and *evidence-based* approaches [4, 8, 18, 22-23, 28]. In a *test-based* model, a processor P_a can test the condition of a processor P_b unaided. However, this is impracticable particularly if malicious faulty processors exist. The malicious processors can

*Corresponding Author: Mao-Lun Chiang; E-mail: mlchiang@cyut.edu.tw
DOI: 10.3966/160792642019092005008

hide their faulty behavior and then avoid detection. Therefore, *test-based* approaches are not suitable for systems with malicious faults.

Alternatively, the *evidence-based* protocol proposed by Shin and Ramanathan [22] primarily collect messages that have accumulated in BA protocols as evidence to detect or locate faulty processors. To detect/locate the maximum number of faulty processors, the proposed protocol must consider dual failure modes (including malicious and dormant faults) of processors and instruct all healthy processors to detect/locate all faulty components in the network. Upon achieving FDA, the performance and integrity of a distributed network can be guaranteed. A protocol designed for FDA must satisfy the following requirements [18, 23]:

(FDA1) Agreement: All healthy processors identify the common set of faulty processors.

(FDA2) Fairness: No fault-free component is falsely identified as faulty by any healthy processor.

To achieve the requirements above, Hsiao et al. [18], proposed an *evidence-based* protocol FDAMIX with dual failure modes to solve the FDA problem. The FDAMIX protocol first collects received messages according to the BA protocol GPBA [27] which is designed to determine a common value in a distributed system under dual failure mode for use as evidence to detect/locate faulty processors. In the FDAMIX protocol, all healthy processors can identify the maximum number of faulty processors under dual failure modes using f_m+2 rounds of message exchange because the GPBA requires f_m+1 rounds of message exchange to achieve agreement. However, message passing is a time-consuming process. It will cause a large protocol overhead under a large number of messages. Thus, previous protocols [16, 18, 20, 25] cannot reach the goal of FDA efficiently and quickly.

Therefore, this paper proposes a brand-new protocol FFDA (Feature-oriented Fault Diagnosis Agreement) to reach agreement only using three rounds of message exchange while simultaneously solving the FDA problem. Besides, the protocol FFDA can still tolerate and detect/locate a maximal number of faulty processors in a minimal number of message exchanges under dual failure mode.

The rest of this paper is organized as follows. Section 2 illustrates the basic assumptions and concept underlying our protocol. Section 3 shows the details of our proposed protocol and examples. The correctness and complexity of our proposed protocol are illustrated in Section 4; and finally, the conclusion is presented in Section 5.

2 The Underlying Assumptions and Concept

Before describing our protocol, the basic assumptions must be defined. According to Fischer *et al*. [6], the BA problem cannot achieve agreement in an asynchronous network if even only one processor has failed and that failure is a crash failure. Therefore, only a synchronous network in which bounds on processing and the communication delays of healthy components are finite is considered [6]. The parameters of this synchronous network are assumed to be the following:

(1) s: The source processor

(2) n: The total number of processors in a distributed network.

(3) $v(s)$: The initial value of processor s broadcasting to all other processors.

(4) $v(sa)$: The value $v(s)$ is sent from the processor P_a.

(5) f_m: The number of processors with malicious faults.

(6) f_d: The number of processors with dormant faults.

(7) λ: When a processor is detected as a dormant processor, the value sent from it will be replaced by λ.

(8) c: The connectivity of a distributed network. Based on Menger's theorem [5, 15], at least c disjoint paths must exist between any pairs of processors P_x and P_y when the connectivity of a distributed network is c.

(9) *mSet*: The set of malicious faulty processors.

(10) *dSet*: The set of dormant faulty processors.

(11) T_i: An information collecting tree (*ic-tree*) of processor P_i.

(12) *FP*(): The function to determine the *feature processor*.

(13) MAT_i: The processor i collects all received vectors (V_j) from other processors.

(14) $MAJ_i(sa)$: The majority value of the level i of the *ic-tree*.

(15) $num\_MAJ_i(sa)$: The number of $MAJ_i(sa)$

(16) PFP_i: The possible *feature processor* set of level i.

(17) Feq_P_z: The frequency that the processor z appears in all PFP.

(18) FP_i: The *feature processor* set of level i.

(19) $MAJ_3\_FP_x(sa)$: The majority value of sub-tree x of $v(sa)$ of the FP in Level 3.

(20) *fau_sym*: The symptoms of faulty processors, such as a noticeable change in the message.

Furthermore, the proposed protocol can solve the BA problem and the FDA problem if the following constraints [15, 23, 27] are satisfied:

Constraint 1: $n > \lfloor (n-1)/3 \rfloor + 2f_m + f_d$,

Constraint 2: $c > 2f_m + f_d$.

Constraint 3: The number of the *fau_sym* of malicious fault $> \lfloor (n-1-f_d)/3 \rfloor$.

The first constraint represents the number of processors required; it must be greater than $\lfloor (n-1)/3 \rfloor + 2f_m + f_d$, including malicious faulty processors (f_m) and dormant faulty processors (f_d). The number of malicious faulty processors must be less than $(2n+3-3f_d)/6$ when the influence of the dormant faults is clearly known. Namely, the number of $n-(\lfloor (n-$

1)/3⌋+2f_m+f_d) healthy processors can provide the complete evidence that allows our proposed protocol reaching a common value and detecting/locating a common set of faulty processors.

Based on Menger's theorem [5], at least c disjoint paths must exist between any pairs of processors P_x and P_y when the connectivity of a distributed network is c. Therefore, the second constraint specifies connectivity requirements; that is to say that the total number of received messages should be greater than the number of fake messages originating from the faulty processor(s) when the influences of dormant faults are removed.

Shin *et al*. proved that no fault diagnosis protocol for malicious faults could complete detecting/locating faulty processors [22]. The protocol cannot detect unobvious malicious faulty symptoms (*fau_sym*) of processors when the number of faulty behaviors of faulty processors is less than ⌊(n-1-f_d)/3⌋. It is because that the number of faulty processors needs to conform to the BA constrains 1 (n>⌊(n-1)/3⌋+2f_m+f_d) [1, 6, 10, 12-15, 19-20, 24]. To detect/locate faulty processors, we assume the number of faulty symptoms (*fau_sym*) of faulty processors can appear in the collected messages in the last constraint. According to the constraints above, an agreement can be achieved, and the number of detectable/locatable faulty processors is f_m+f_d in our protocol.

Subsequently, a convenient data structure, the *ic-tree* (an information collecting; T_i) [1, 18-20] is invoked in our proposed protocol. This is because that each processor can easy to store and accumulate the received messages from other processors into corresponding vertices round by round. Besides, the structure of *ic-tree* can be used to prove the correctness of the protocol as Section 4. The vertex of an *ic-tree* is labeled with a list of processor names, and the value received from the source processor is denoted as v(s) at the root of the *ic-tree*. The name list of the processor contains the names of the processors through which the stored message has been transferred. For example, the statement v(*sbc*) represents the processor having received the value *sb* from processor P_c which was sent from source processor P_s to processor P_b. Similarly, the value v(*sbd*) of sibling vertex represents the processor having value *sb* from processor P_d. Subsequently, the *ic-tree* is constructed by following reorganization rules:

- The leaves in level f_m + 2 of the *ic-tree* are deleted.
- The vertices with repeated processor's names are deleted.

According to reorganization rules, the *ic-tree* can be constructed to avoid cyclical influences [10, 17] from the faulty processors. The cyclical influences originate from messages sent by faulty processors that may be stored repeatedly in the *ic-tree* and this could result in an incorrect common value being obtained by taking a simple majority. Therefore, the *ic-tree* (T_i) can be used to store the received messages and to eliminate the influence of faulty components, as shown in Figure 1.

Level 1 *root*	*Level 2*	*Level 3*
v (*s*)	v (*sa*)	~~v (*saa*)~~
		v (*sab*)
		v (*sac*)
		v (*sad*)
		v (*sae*)
	v (*sb*)	v (*sba*)
		~~v (*sbb*)~~
		v (*sbc*)
		v (*sbd*)
		v (*sbe*)
	v (*sc*)	v (*sba*)
		v (*scb*)
		~~v (*scc*)~~
		v (*scd*)
		v (*sce*)
	v (*sd*)	v (*sda*)
		v (*sdb*)
		v (*sdc*)
		~~v (*sdd*)~~
		v (*sde*)
	v (*se*)	v (*sea*)
		v (*seb*)
		v (*sec*)
		v (*sed*)
		~~v (*see*)~~

Figure 1. The *ic-tree* (T_i)

After constructing the *ic-trees*, our protocol can remove dormant faulty processors because recipients can always identify faulty messages produced by a dormant component (i.e., crash and omission faults) if the Manchester code [16] is used in encoding before transmitting. This is because Manchester encoding is a synchronous clock encoding technique; thus, the recipient can easily distinguish between the dormant faulty components. The values sent by dormant faulty components are replaced by λ in our protocol. Furthermore, each processor in our protocol can obtain an *ic-tree*; subsequently, the majority function MAJ(α), shown as Figure 2, is invoked in Level 3 of the *ic-tree* to compute a common value. In Figure 3, the majority value v(*sa*) can be obtained by taking the MAJ(α) on {v(*sab*), v(*sac*), v(*sad*), v(*sae*)} in Level 3. Otherwise, the majority value is replaced by the complement of v(α) when the majority value does not exist.

In addition, the prior works of literature [7, 9, 11-12, 15] argue for the BA problem under the assumption of synchronous behavior BA, showing that 3f_m+1 processors are allowed f_m failures where f_m is the number of faulty processors in a distributed network. Besides, the f_m+1 rounds of message exchange are required under the dual failure mode. In the case of the FDA problems [1, 15, 21-23], the f_m+2 rounds of message exchange are required to collect enough messages as evidence to detect/locate faulty processors

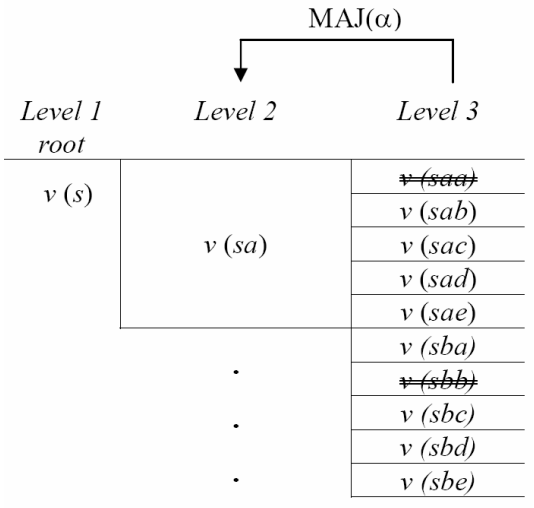


Figure 2. The function MAJ

The function MAJ(α)	
MAJ(α)=	1. The majority value in the set of $\{v(\alpha j)\|1\le j\le n\}$, if such a majority value exists. 2. The complement value of $v(\alpha)$, denoted as $\neg v(\alpha)$, is chosen, otherwise.

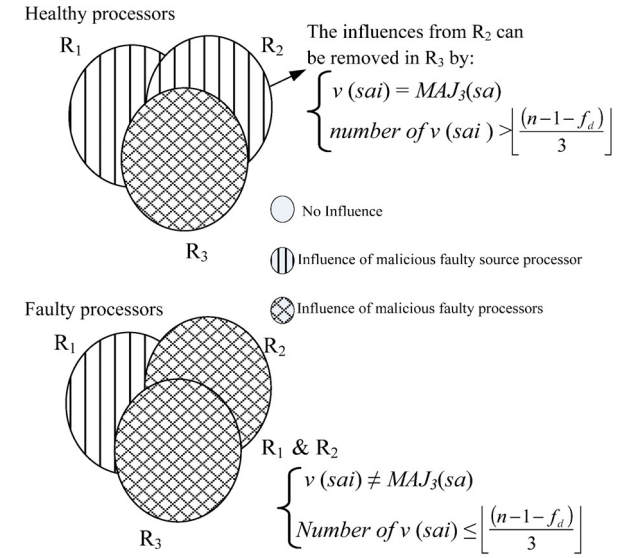
Figure 3. Take the MAJ functions in Level 3

under dual failure modes due to the agreement protocol, such as GPBA [27].

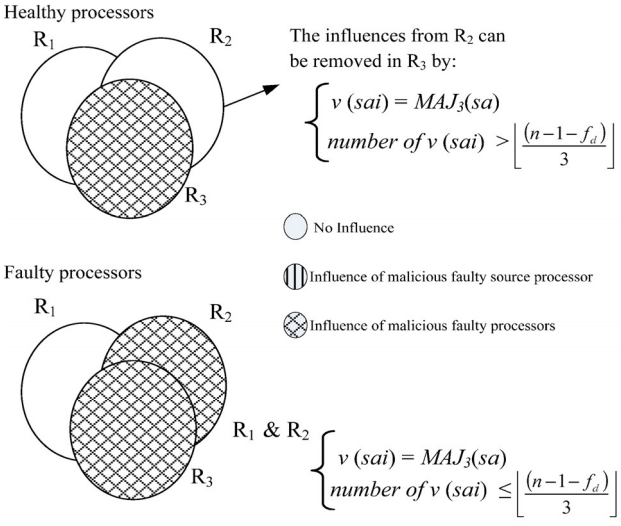
However, the message passing is a time-consuming process and the number of messages results in a large protocol overhead. This is unreasonable and inefficient for a distributed system containing a large number of processors. For reducing the number of message exchange, a novel algorithm, the Feature-based Fault Diagnosis Agreement (FFDA) is proposed in this paper and the number of rounds is less than all of the previous BA algorithms [1, 6-7, 12-15, 17, 19-20 24, 27]. This is because a brand-new concept of *feature value* is proposed to reduce the number of messages and the number of rounds rapidly. Besides, the FFDA algorithm is still better than IBA [28] due to the IBA needs to remove faulty processors first, and requires extra rounds of message exchange to achieve agreement.

With regard to the FFDA, the *feature value* of fault-free processors can be derived from a large number of messages and defined as the following paragraphs, subsequent the processors with the *feature value* can be recognized as the *feature processors*. Based on the *feature processors*, the faulty processors can easy to filter out and the agreement can be reached simultaneously regardless of the number of processors in a distributed network. The detail is described as follows.

In general, a source processor may be a malicious faulty processor; thus, two situations must be considered, malicious faulty source processors and healthy source processors, as shown in Figure 4. In Figure 4(a), the values in round 1 (R_1), under the influence of the malicious faulty source processor may send different values to each processor. Similarly, the values in R_2 demonstrate influences from the malicious faulty processors and malicious faulty source processor. Namely, the stored values of R_2 can be influenced by R_1 and R_2.



(a) Malicious faulty source processor



(b) Healthy source processor

Figure 4. The influence of each round

However, the influences from R_2 can be removed from R_3 by taking the majority on messages in Level 3 of its *ic-tree*, such as MAJ(α){v(*sab*), v(*sac*), v(*sad*), v(*sae*)} = MAJ$_3$(*sa*). This is because there are more than $\lfloor (n\text{-}1\text{-}f_d)/3 \rfloor$ healthy processors in a distributed network even if the source is a malicious faulty processor. Hence, MAJ$_3$(*sa*) of Level 2 in R_3 can be obtained without the faulty influence of R_2.

Based on the reason above, the messages in Level 2 can also be removed in R_3 when the source is a healthy processor, as shown in Figure 4(b). Regardless whether the source is healthy, the stored values of Level 2 in R_3 are real values sent from the predecessor and uninfluenced by the malicious faulty processors in R_2 and are called feature values. Therefore, each vertex i ($1 \le i \le n$) in Level 3 of an *ic-tree* is a possible feature processor P_i when it has this feature value ($v(sai) = MAJ_3(sa)$). For each sub-tree of the ic-tree, we can identify processor P_i as the feature processor if the number of P_i in (PFP) exceeds (n- f_m). This is because there are more than $\lfloor (n\text{-}1\text{-}f_d)/3 \rfloor$ healthy processors able to transfer the feature value received from the predecessor to other processors. Based on this novel concept, the goals of FDA and BA can be achieved. Besides, the number of messages can be reduced even if a large number of processor exists. Subsequently, the details of our proposed protocol FFBA are introduced in the next section.

3 The FFDA Protocol

The Feature-based Fault Diagnosis Agreement (FFDA) protocol we proposed includes three phases: *message exchange phase*, *fault diagnosis phase*, and *decision-making phase*. The FFDA protocol is shown in Figure 5. The details of the protocol are shown as follows.

3.1 The Message Collection Phase

This phase is used to collect three rounds of message exchange and store the received messages in the processors' *ic-tree*. The collected three rounds of messages are necessary to be used as evidence to eliminate the influence of faulty processors in the next phase.

3.2 The Fault Diagnosis Phase

In general, healthy destination processors can detect message(s) from dormant faulty components if the protocol appropriately encodes a transmitting message by using either the Non-Return-to-Zero code or the Manchester code [5, 7] before transmission. Therefore, the messages sent from dormant faulty components can be replaced by λ and can be removed during each round of message exchange by *dormant diagnosis rule*. Furthermore, we use the *dSet* (dormant faulty processor set) to record dormant faulty processors during this phase.

However, the *malicious diagnosis rule* is used to search the *feature processors*. Subsequently, the *mSet* (malicious faulty processor set) is used to record malicious faulty processors. In first *for loop* of function *FP*, the $MAJ_3(sa)$ is taken by MAJ function in level 3 of the *ic-tree*. The processors can be stored into the PFP (*possible feature processor*) set when the $v(sa)$ $=MAJ_3(sa)$ and the $num\_MAJ_3(sa) \ge f_m$. After visiting all sub-trees of vertex $v(sa)$ in the *ic-tree*, the processor P_z of PFP can be identified as a *feature processor* when the $Feq\_P_z \ge (n\text{-}f_m)$ is satisfied. Subsequently, the values of non-*feature processors* are set as $MAJ_3\_FP_i(sa)$ in second *for loop*. Finally, the dormant and malicious faulty processors can be obtained.

Message Exchange Phase
Collect r (r = 3) rounds of messages which have accumulated in the GPBA [27] as evidences.
r = 1, do:
for i ≤ n *{*
*source processor (*s*) broadcasts initial value* v*(*s*) to* P_i*}*
r = 2, do:
for *each* P_i *(*i ≤ n*) {*
broadcast v*(*s*) && receive others broadcasted values*
store v*(*sa*) into the corresponding* ic-tree T_i*}*
r = 3, do:
for *each* P_i *(*i ≤ n*) {*
broadcast T_i *&& receive others broadcasted* T_i
construct the MAT_i*}*
for *each* MAT_i *(*i ≤ n*) {*
for *each row* k *(*k ≤ n*) {*
If $v_{ki} = \lambda$, **then** *{* v_{ki} *is ignored}*
majority *(row* k *of* MAT_i*) && construct a corresponding* ic-tree T_i*}*
return *(*T_i*)*

Fault Diagnosis Phase
***dormant diagnosis rule*: Let processor i ($1 \le i \le n$) be a healthy processor. P_i can detect processor P_k ($1 \le k \le n$) as faulty processor if:**
P_i receives the λ from P_k (or no message is received from P_k) and the number of copies of λ is greater than $\lfloor (n\text{-}1\text{-}f_d)/3 \rfloor$, then $dSet = dSet \cup \{P_k\}$;
malicious diagnosis rule:
Run *FP*( )
for each sub-tree of vertex $v(sa)$ in level 2 of *ic-tree* T_i ($i \le n$){
if $v(sa) = MAJ_3(sa)$ && $num\_MAJ_3(sa) \ge f_m$ ***then***{
$P_a \in PFP_i$.
for each sub-vertex of the sub-tree of vertex $v(sa)$**{**
if $v(sab) = v(sa)$ ***then*** $PFP_i \cup \{P_b\}\}\}\}$
for each PFP_i ($i \le n$){
Count (Feq_P_z)
if $Feq\_P_z \ge (n\text{-}f_m)$ ***then***{
FP $\cup$ $\{P_z\}$
else $mSet \cup \{P_z\}$ && $v(sab) = MAJ_3\_FP_i(sa)\}$
return ($mSet$ && $dSet$)

Decision Making Phase:
Step 1: Applying function VOTE to the root of each *ic-tree* T_i and common value, VOTE(s) is obtained.

VOTE Function

The function VOTE(α)=	1. $v(\alpha)$, if α is a leaf. 2. The majority value in the set of {VOTE(αi)| $1 \le i \le n$, and vertex αi is a child of vertex α}, if such a majority value exists. 3. A default value ϕ is chosen, otherwise

Figure 5. FFDA Protocol

3.3 The Decision-making Phase

In this phase, the function VOTE is applied to root *s* of each healthy processor's *ic-tree*. Finally, each processor can obtain a decision value VOTE(*s*) during this phase.

Table 1 shows the important comparisons between FFDA and previous BA protocols. Basically, the overall performance is better than other protocols.

Furthermore, another important contribution of our protocol is to reduce the complexity of the diagnostic procedure; the FDA problem and BA problem [1, 15-16, 18, 20-23] are solved early by our proposed protocol FFDA that uses the minimum number of rounds characterized by dual failure of processors. The comparisons between FFDA and previous diagnosis protocols are assumed to solve the BA and FDA problems simultaneously are shown in Table 2. As a result, the FFDA we proposed is more efficient and reasonable than previous protocols, such as the number of message exchanges, the message complexity, and the number of required rounds.

Table 1. The comparisons between FFDA and previous BA protocols

	Required rounds	Message complexity	Allowable faulty processors
Previous protocols [8, 16-17, 20]	$f_m + 1$	$O(cn^{\sigma})$; $(\sigma \leqq \lfloor (n-1)-f_d)/3 \rfloor)$	f_m
FFDA	3	$O(cn^2)$	f_m

Table 2. The comparisons between FFDA and previous diagnosis protocols

	Required rounds	**Message complexity**	**Fault types**	**The number of processor required**
Hsiao et al. [8, 19]	$\lfloor (n-1))/3 \rfloor+3$	$O(cn^{\sigma}+ cn^{\sigma+1}+ n)$ $\sigma \leqq \lfloor (n-1)-f_d)/3 \rfloor+1$	Dual faults	$n>3f_m+f_d$
IBA [3]	4	$O(cn^3+ n)$;	Dual faults	$n>3f_m+f_d$
FFDA	3	$O(cn^3)$;	Dual faults	$n>3f_m+f_d$

3.4 Example of Executing FFDA

In this section, an example is provided to show the overall procedure of FFDA. The initial environment is shown in Figure 6(a). Besides, to prove the validity of FFDA, a worst case is designed (the numbers of 0's and 1's are approximately the same) and the transmission behavior of the faulty processors is shown in Figure 6(b). Based on [4, 17, 19, 20, 22-23], BA and FDA requirements, the result of an agreement of faulty processors need not be discussed. This is because the goal of the BA/FDA protocol is to allow all healthy processors to reach a common value. Besides, the faulty processors cannot influence the agreement results of healthy processors by BA and FDA requirements. Therefore, this example only shows the results of healthy processors.

At the beginning of the protocol, the faulty source processor P_a broadcasts its initial value to all processors during the first round of *message exchange phase*. Unfortunately, the source processor P_a is a malicious faulty processor; it sends different values, 0, 1, 0, 1, 0, 1, 0, 1, and 0, respectively, to processors P_d, P_e, P_f, P_g, P_h, P_i, P_j, P_k, P_l, and P_m. Here, each healthy processor stores the received value into the root of its *ic-tree* in the first round, as shown in Figure 6(c).

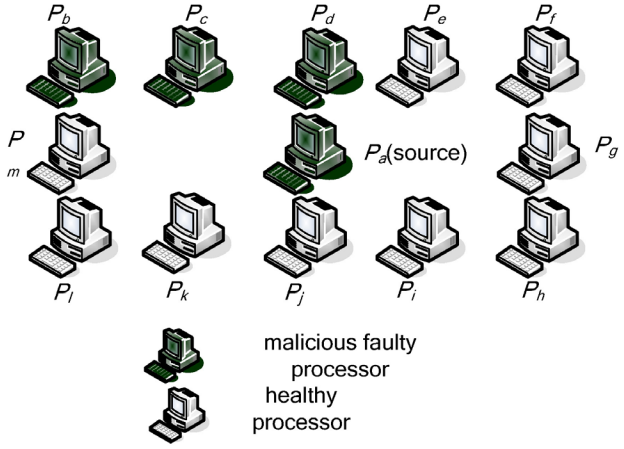


Figure 6(a). The 13-processor environment

	e	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*
a	0	1	0	1	0	1	0	1	0
b	1	0	1	1	1	0	1	0	1
c	1	1	1	1	0	0	0	0	1
d	0	0	1	1	0	0	1	1	0

Figure 6(b). The transmission behavior of faulty processors

e	v(a)=0
f	v(a)=1
g	v(a)=0
h	v(a)=1
i	v(a)=0
j	v(a)=1
k	v(a)=0
l	v(a)=1
m	v(a)=0

Figure 6(c). The vectors received from source processor P_a in first round of message exchange phase

Subsequently, each processor exchanges the received value from the first round of *message exchange phase* with all processors during the second round of *message exchange phase*. Similarly, the received messages are stored in the second level of its *ic-tree*, as shown in Figure 6(d). In the third round of *message exchange phase*, each processor exchanges the received values from the second round of *message exchange phase* with all processors and stores the received values into the third level of their *ic-trees*. For this example, the result of processor P_e is shown in Figure 6(e).

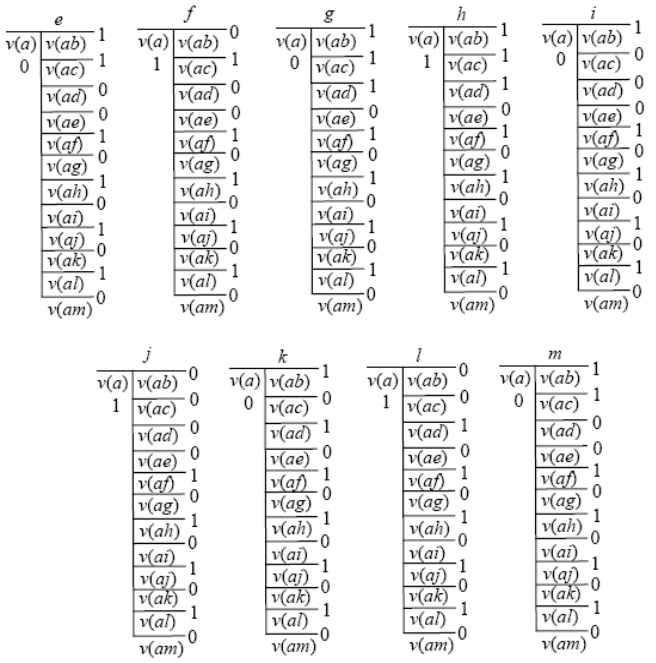


Figure 6(d). The ic-trees of processors P_e, P_f, P_g, P_h, P_i, P_j, P_k, P_l, and P_m in the second round of message exchange phase

In the next phase, the *fault diagnosis phase*, each processor must first determine which processors are *possible feature processors*. For example, the procedure for processor P_e is shown in Figure 6(f). Processors P_a, P_d, P_e, P_f, P_g, P_h, P_i, P_j, P_k, P_l and P_m can be recognized as *possible feature processors* and stored into the PFP set in Figure 6(f) when the following conditions are satisfied.

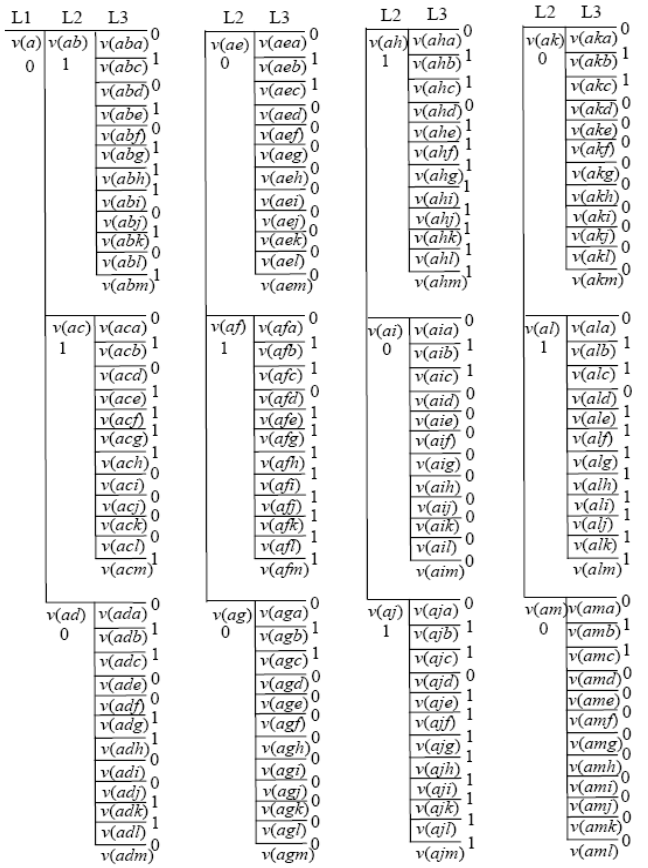


Figure 6(e). The *ic-tree* of processors P_e in the third round of *message exchange phase*

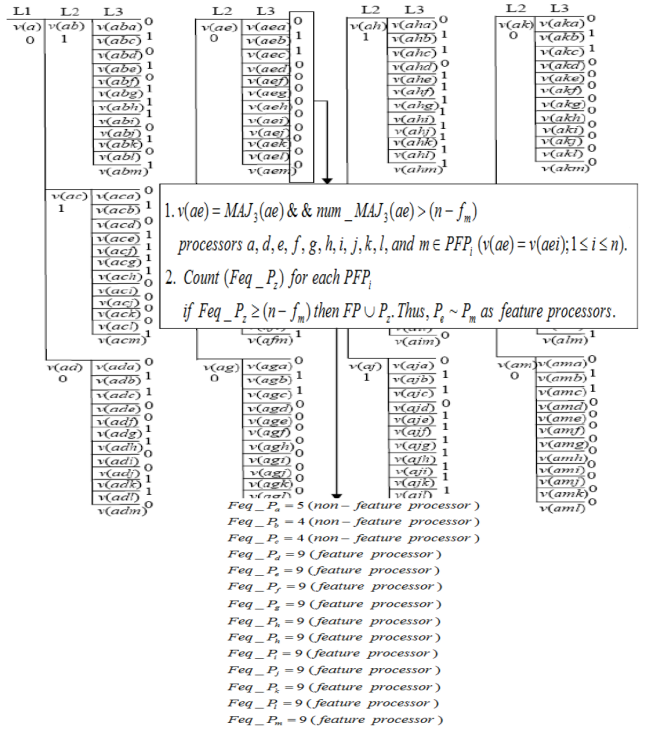


Figure 6(f). The procedure of *FP*( ) to determine the *feature processors* in *fault diagnosis phase*

- $v(ae) = \mathrm{MAJ}_3(ae) = 0$,
- $[num\_\mathrm{MAJ}_3(ae) = 10] \geq [(n\text{-}f_m) = 9]$, and

- $v(aei)$ = $MAJ_3(ae)$\{such as, $v(aea)$, $v(aed)$, $v(aef)$, $v(aeg)$, $v(aeh)$, $v(aei)$, $v(aej)$, $v(aek)$, $v(ael)$ and $v(aem) = MAJ_3(ae) = 0$\}

After that, the FFDA protocol will count the frequency (Feq_P_z) that each processor appears in all PFP and computes whether the Feq_P_z is greater than (n-f_m) or not. In this example, processors P_e, P_f, P_g, P_h, P_i, P_j, P_k, P_l, and P_m are *feature processors*, as shown in Figure 6(f). Subsequently, the values of non-*feature processors* can be replaced by $MAJ_3\_FP_i(sa)$ using *Rep* function. For example, the $v(aba)$ and $v(abd)$ are not equal to the $MAJ_3\_FP_3(ab)$ in Figure 6(g), thus the values $v(aba)$ and $v(abd)$ in the third level of the *ic-tree* will be changed to $MAJ_3\_FP_i(sa)$ =1. Similarly, all sub-trees in the third level of the *ic-tree* can be executed the same procedures; the result of processor P_e is shown in Figure 6(g). Finally, the function VOTE is applied to root $v(a)$ of an *ic-tree* during the *decision- making phase*. The common result (VOTE(a) =ϕ) of the processor P_e are shown in Figure 6(h). Since all healthy processors will execute the same procedures, an agreement is reached while the f_m faulty processors exist. Furthermore, the P_a, P_b, P_c, and P_d can be detected as a malicious faulty processor by *feature processors*. Therefore, only three rounds of message exchange are needed in our protocol FFDA, but the f_m+2 rounds of message exchange are needed to reach an agreement and detect/locate faulty processors in traditional protocols.

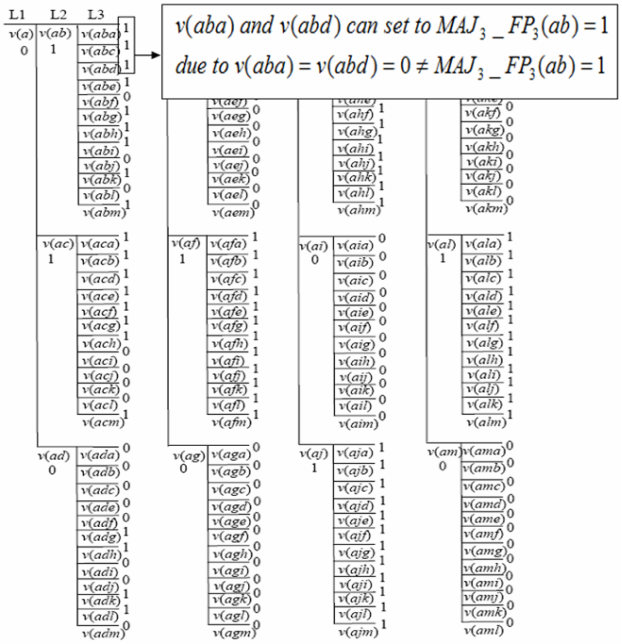


Figure 6(g). The values of non-*feature processors* can be set to $MAJ_3\_FP_i(sa)$ during *fault diagnosis phase*

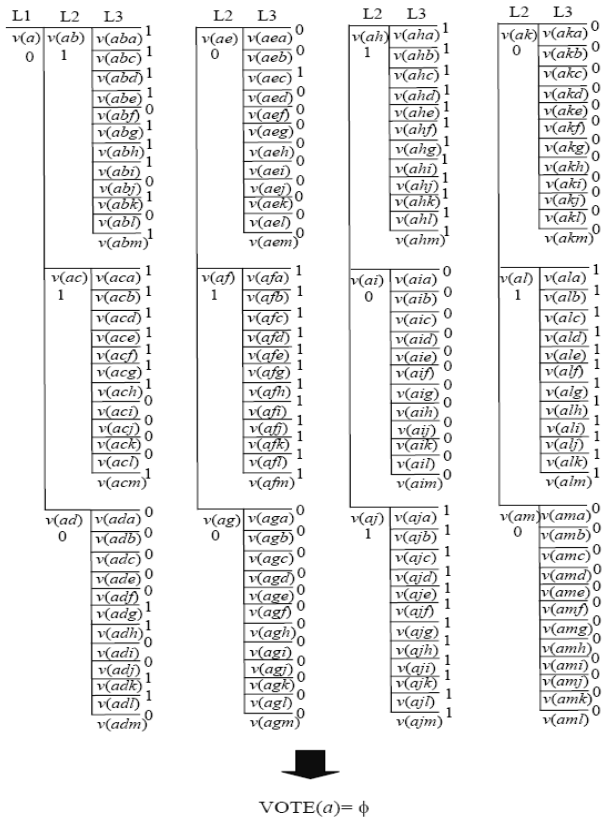


Figure 6(h). An agreement value of the *decision-making phase*

4 The Correctness and Complexity of the FFDA

The following required proofs for agreement and validity of our solution to the BA are given in this section. The lemmas and theorems are used to prove the correctness and complexity of FFDA.

4.1 Correctness of FFDA

To prove the correctness of our protocol, a tree structure, *ic-tree* is used to explain our procedures. The *ic-tree* collected sufficient complete messages to eliminate the influences of faulty components and solve the cyclical influences of the faulty processors by eliminating the repeated names. The function VOTE must also be used to obtain a common value from the *ic-tree* in the *decision-making phase*. Therefore, this paper proves the correctness of our protocol by *ic-tree* structure.

This paper defined a vertex α as common [17] if each healthy processor computes the same value for α. In other words, the value stored in vertex α of each healthy processor's *ic-tree* is common to all. Once each healthy processor has a common initial value from the source processor in the root of its *ic-tree*, an agreement is reached since that root is common to all. Thus, the Agreement and Validity can be rewritten as:

(Agreement'): Root s is common, and

(Validity'): VOTE(s) = v_s for each healthy processor, if the source processor is healthy.

To prove a vertex is common, the term common frontier [1] is defined as follows: When every root-to-leaf path of the *ic-tree* contains a common vertex, the collection of the common vertices forms a common frontier. In other words, every healthy processor has the same messages collected in the common frontier if a common frontier does exist in a healthy processor's *ic-tree*. Subsequently, using the same voting function VOTE to compute the root value of the *ic-tree*, every healthy processor can obtain the same root value as a result of the same input and the same computing function.

Before proving the correctness, the term *correct vertex* is defined as (1) *correct vertex*-vertex αi of a tree is a correct vertex if processor P_i is healthy. In other words, a correct vertex is a place to store the value received from a healthy processor. (2) *true value*: for a *correct vertex* αi in the tree of a healthy processor P_i, $v(\alpha i)$ is the *true value* of vertex αi. Namely, the stored value is called the *true value* and can be recognized as the *feature value*.

By the definition of a correct vertex, the stored value is received from the healthy processors and a healthy processor always transmits the same value to other processors. The repeated vertices of the *ic-tree* are deleted, thus, the correct vertices of such an *ic-tree* are common. Based on the definition of the *correct vertex*, a common frontier does exist in the *ic-tree*. Namely, the root can be proven to be a *common vertex* (Agreement') due to the existence of a common frontier, regardless of the correctness of a source processor. Therefore, an agreement of the root value is reached.

Subsequently, we will check the condition of (Validity'). Based on (Validity'), we know that when the source processor has failed, the (Validity') is true. This is because the propositional logic P→Q means (NOT(P) OR Q), then (NOT(P) OR Q) or (P→Q) is true when P is false, where P implies "the source processor is healthy" and (P→Q) implies BA_2'. Conversely, root s is a correct vertex by the definition of a correct vertex if the source processor is healthy. If all correct vertices' true values can be computed by FFDA, then the true value of the root can be computed because the root is a correct vertex. By definition, the true value of the root is the initial value of the source processor if the source processor is healthy. Namely, each healthy processor's root value is the initial value of the source processor. If the source processor is healthy, then Validity' is true when the source processor is healthy. In short, the Agreement' and Validity' are both true no matter whether the source processor is healthy or failed, and the BA problem is solved.

Lemma 1. *The messages through dormant faulty components can be detected by a healthy destination processor.*

Proof. A healthy destination processor can detect the message(s) from dormant faulty components if the protocol appropriately encodes a transmitted message using either the Non-Return-to-Zero code or the Manchester code [5, 7] before transmitting.

Lemma 2. *The healthy processors can receive messages from healthy processors if* $c > 2f_m + f_d$.

Proof. A healthy sender processor broadcasts a message to others and itself. In general, each healthy processor can receive at least c copies of messages in c-connectivity distributed network based on Menger's theorem [5]. However, a healthy processor can receive $c - f_d$ messages transmitted during each round of message exchange in the worst case (the dormant faulty components can be detected by Return-to-Zero code or the Manchester code). If $c - f_d > 2f_m$, a healthy processor can determine messages from sender processors by taking the majority value from the values received during each message exchange round.

Theorem 1. *A healthy processor can remove the faulty influences from dormant faulty processors if* $c > 2f_m + f_d$.

Proof. By Lemmas 1 and 2, the theorem is proved.

Lemma 3. *The healthy destination processor can detect a dormant faulty sender processor using a forwarding technique in a distributed network.*

Proof. If the number of value λ is greater than or equal to $c - \lfloor (n-1)/3 \rfloor$, then the sender processor is in dormant fault. The reason for this is that there are at most $\lfloor (n-1)/3 \rfloor$ malicious faulty components in a distributed network, hence there are at most $\lfloor (n-1)/3 \rfloor$ non-λ values in the vector V_i.

Theorem 2. *Healthy processors can detect dormant faulty processors in an n-processor system.*

Proof. In the protocol FFDA, there are three rounds of message exchange during the *message exchange phase*, where $f_m \le \lfloor (n-1)/3 \rfloor$ and $n > 3$. Each healthy processor can receive the message from the source processor during the first round of message exchange and receives other processors' messages during the second round of message exchange. Each processor can receive all other processors' messages in a distributed network after two rounds of message exchange. According to the Lemma 3, each healthy processor can detect dormant faulty processors in *an n-processor system*.

Lemma 4. *All healthy correct vertices of an ic-tree are common.*

Proof. Since the vertices with repeated processor names of *ic-trees* are removed, no repeatable vertices are in an *ic-tree*. In the level f_m or above, the correct vertex α has at least $2f_m+1$ children ($n - f_m \ge 2f_m+1$) in which at least f_m+1 children are correct. The *true value* of these f_m+1 correct vertices is common, and the majority value of vertex α is common. The correct vertex α is common in the *ic-tree* if the level of α is

less than f_m+1. Thus, all correct vertices of the *ic-tree* are common.

Lemma 5. *The common frontier exists in an ic-tree.*

Proof. By Lemmas 4, the *true values* of these f_m+1 correct vertex are common. Since at most f_m ($\leq \lfloor (n-1)/3 \rfloor$) processors can be failed, at least one vertex is correct along each root-to-leaf path of the *ic-tree*. Therefore, the correct vertex is common, and the common frontier exists in each healthy processor's *ic-tree*.

Lemma 6. *Let α be a vertex, where α is common if there is a common frontier in the subtree rooted at α.*

Proof. If the height of α is 0 and the common frontier (α itself) exists, α is common. If the height of α is σ, the children of α are all in common using an induction hypothesis with the height of the children at σ-1, then the vertex α is common.

Lemma 7. The values replaced by the majority value of *feature processors* are common.

Proof. By Lemmas 4, 5, and 6, all correct vertices of the *ic-tree* are common, and each healthy processor's *ic-tree* also has the same common frontier. Further, there are at least n-f_m processors that are healthy and having the *feature value*. Hence, all these healthy processors with the *feature value* must be *feature processors*. Thus, the majority value of these *feature processors* for each sub-tree in the third level must be common. The values replaced by the majority value of *feature processors* are also common.

Lemma 8. The values sent by the healthy processors are the same as the majority value after applying the VOTE function.

Proof. There are at least n-f_m healthy processors in a distributed network. All the healthy processors will transmit their values to others correctly. In each round of message exchange, there will be n-f_m healthy processors that can receive these values and resend them. Then, the majority values which are applied using the VOTE function for the $(i + 1; 1 \leq i \leq f_m)$*th* level of the *ic-tree* must be equal to the values in the *i*th level of the *ic-tree*. It is unnecessary to send the values for (f_m+1) rounds if the sender is a healthy processor.

Corollary 1. *The root is common if the common frontier exists in the ic-tree.*

Theorem 3. *The root of a healthy processor's ic-tree is common.*

Proof. By Lemmas 4, 5, 6, 7, 8 and Corollary 1, the theorem is proved.

Theorem 4. *Protocol FFDA can solve the BA in a distributed network.*

Proof. To prove the theorem, it must show that FFDA meets the constraints (Agreement') and (Validity').

(Agreement'): Root s is common. By Theorem 3, (Agreement') is satisfied.

(Validity'): VOTE(s) = v for all healthy processors, if the initial value of the source is v_s, say $v = v_s$.

Since most of the processors are healthy, they transmit the message to all others. As a result, each correct vertex of the *ic-tree* is common (Lemma 4), and its *true value* (*feature value*) is v. By Theorem 3, this root is common. The computed value VOTE(s) = v is stored in the root for all healthy processors. (Validity') is satisfied.

Lemma 9. *All correct vertices of an ic-tree are identical.*

Proof. After the finishing the *message exchange phase*, no repeatable vertices are located in an *ic-tree*. In level 3, the correct vertex (the vertex uninfluenced by the faulty processors) α has at least 2 $\lfloor (n-1-f_d)/3 \rfloor+1$ children, of which at least $\lfloor (n-1-f_d)/3 \rfloor+1$ children are correct. Therefore, the *feature value* of the $\lfloor (n-1-f_d)/3 \rfloor-1$ round from the correct processor can be obtained. However, the source processor may be a malicious processor, thus we discuss two cases as follows.

CASE 1. In this case, the source processor is a malicious faulty processor. The malicious faulty source processor sends the different initial value $v(s)$ to each processor during the first round of message exchange. Subsequently, each processor broadcasts the received value from the faulty source processor during the second round of message exchange. The value $v(v_{si})$ ($1 \leq i \leq n$) may be also altered to a different value by malicious processors in this round. In the final round, each processor broadcasts the received value from the second round of message exchange to the others. However, the influence of a faulty value from healthy processors produced in the previous round can be removed by the majority function MAJ. This is because there are more than $\lfloor (n-1-f_d)/3 \rfloor$ healthy processors that can receive the *feature value* $v(\alpha)$ from the previous round regardless whether the $v(\alpha)$ is correct or not.

CASE 2. In this case, the source processor is a healthy processor in a distributed network. The healthy source processor sends the same initial value $v(s)$ (the correct value) to each processor during the first round of message exchange. Subsequently, each processor P_i ($1 \leq i \leq n$) broadcasts the received value from the healthy source processor to the others in the second round of message exchange. The value $v(v_{si})$ may be also altered by malicious processors during this round. Finally, each processor can exchange received values during the third round of message exchange. As in CASE 1, the number of *feature value* $v(\alpha)$ received from the healthy processors exceeds n-f_m. Therefore, the faulty processors can easily be detected.

Theorem 5. *The FFDA can solve the FDA problem if* $n > \lfloor (n-1)/3 \rfloor + 2f_m + f_d$ *and* $c > 2f_m + f_d$.

Proof. By Theorem 1, Theorem 2, Theorem 3, Theorem 4, and Lemmas 9, the theorem is proven.

4.2 Complexity of FFDA

Theorem 6. *FFDA requires three rounds to solve the BA and FDA problems with a distributed network if n>*

$\lfloor(n-1)/3\rfloor+2f_m+f_d$ and $c>2f_m+f_d$ are satisfied. Furthermore, the three rounds are the minimum number of rounds of message exchange.

Proof. In general, the traditional protocol [22] needs $\lfloor(n-f_d)/3\rfloor+1$ rounds of message exchange to reach an agreement. Based on the concept of Figure 4, the *feature value* of the previous round can be obtained. However, we take the majority on level 3 of each *ic-tree* and determine the processors which agree. Each vertex can be identified as *possible feature processor* if it is equal to the *feature value*. Subsequently, our protocol accumulates the frequency of processor P_z of *possible feature processor* sets and compares them with $\lfloor(n-f_d)/3\rfloor$. The processor P_z can be identified as a *feature processor* since there are $n-(\lfloor(n-f_d)/3\rfloor)$ healthy processors assuming it is correct. Similarly, there are more than $\lfloor(n-f_d)/3\rfloor+1$ vertices in all *ic-trees* having *feature values* by Lemma 9, thus the faulty processors can be easily detected/located. The three rounds of message exchange are necessary because the influences from R_2 can be removed in R_3 by taking the majority on messages in Level 3 of its *ic-tree*. As a result, our proposed protocol FFDA requires only three rounds to solve the BA and FDA problems; three rounds being the minimum number of rounds of message exchange required for this purpose.

Theorem 7. *The total number of allowable faulty components by the FFDA is f_m+f_d, where $n>\lfloor(n-1)/3\rfloor+2f_m+f_d$ and $c>2f_m+f_d$.*

Proof. According to the constraints of the BA problem for a processor which was proposed by Siu *et al*. [22, 27], the constraints over our protocol are $n>\lfloor(n-1)/3\rfloor+2f_m+f_d$ and $c>2f_m+f_d$. In the worst-case scenario, the malicious faulty processors and dormant faulty processor exist simultaneously, thus the total number of allowable faulty components by the FFDA is f_m malicious faulty processors and f_d dormant faulty processors.

Theorem 8. *The total number of messages in the FFDA is cn^3.*

Proof. The protocol FFDA must use the *ic-tree* as evidence to detect/locate faulty processors. The *ic-tree* is constructed from the *message exchange phase* of each round. Hence, we have $O(cn^3)$ (c represents a constant) messages in each *ic-tree*. Furthermore, previous protocols [22, 27] required $O(cn^{\sigma}+cn^{\sigma+1})$ ($\sigma\leq\lfloor(n-f_d)/3\rfloor+1$) messages to reach an agreement and detect/locate the faulty processors. However, we know the *message exchange phase* is a time-consuming phase. The number of messages burdens the protocol with a large overhead. As a result, the FFDA can reach an agreement and detect/locate the maximum number of faulty processors using a minimal number of rounds efficiently and quickly.

5 Conclusion

In general, the previous protocols [20, 23] require f_m+1 rounds of message exchange to reach agreement in BA problem, and f_m+2 rounds of message exchange are needed to detect/locate the faulty processors in FDA problem. Besides, the complexities of message exchanges in BA and FDA problem are $O(cn^{\sigma})$ and $O(cn^{\sigma}+cn^{\sigma+1}+n)$ [13-14, 18] respectively in previous works. However, the traditional protocols are inefficient and unsuitable for a distributed network topology due to a large number of message exchange imposes a heavy overhead on the protocol. To reduce a large number of message exchange, a concept of *feature value* which is used to recognize the *feature processors* is proposed in FFDA to make each healthy processor solve BA and FDA problems simultaneously with dual failure mode by using three rounds of message exchange regardless of the number of processors in a distributed network. Furthermore, the number of messages can be reduced to $O(cn^3)$. As a result, the FFDA protocol is more efficient and reasonable in a distributed network topology than previous protocols [1, 4, 10, 12-14, 18-20, 23-24, 28] where $n>\lfloor(n-1)/3\rfloor+2f_m+f_d$ and $c>2f_m+f_d$.

References

[1] I. Abraham, D. Dolev, Byzantine Agreement with Optimal Early Stopping, Optimal Resilience and Polynomial Complexity, *Proceeding of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, ACM New York*, NY, USA, 2015, pp. 605-614.

[2] W. Ahmed, Y. W. Wu, A Survey on Reliability in Distributed Systems, *Journal of Computer and System Sciences*, Vol. 79, No. 8, pp. 1243-1255, December, 2013.

[3] A. Bar-Noy, D. Dolev, C. Dwork, H. Raymond Strong, Shifting Gears: Changing Algorithms on the Fly to Expedite Byzantine Agreement, *Information and Computation*, Vol. 97, No. 2, pp. 205-233, April, 1992.

[4] R. W. Buskens, R. P. Bianchini, Distributed on-line Diagnosis in the Presence of Arbitrary Faults, *Proceeding of the Symposium. Fault-tolerant Computing*, 1993, pp. 470-479.

[5] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, Prentice-Hall, 1974.

[6] D. Dolev, R. Reischuk, Bounds on Information Exchange for Byzantine Agreement, *Journal of ACM*, Vol. 32, No. 1, pp. 191-204, January, 1985.

[7] M. Fischer, The Consensus Problem in Unreliable Distributed Systems (A Brief Survey), *Proceeding of the International FCT-Conference on Fundamentals of Computation Theory*, 1983, pp. 127-140.

[8] A. Mostéfaoui, H. Moumen, M. Raynal, Signature-Free Asynchronous Binary Byzantine Consensus with $t < n/3$, $O(n^2)$ Messages, and O(1) Expected Time, *Journal of the ACM* (*JACM*), Vol. 62, No. 4, Article No. 31, 2015.

[9] D. Ramesh, C. Kumar, An Optimal Novel Byzantine Agreement Protocol (ONBAP) for Heterogeneous Distributed Database Processing Systems, *Procedia Technology*, Vol. 6, pp. 57-66, 2012.

[10] S. S. Wang, S. C. Wang, K. Q. Yan, An Optimal Solution for Byzantine Agreement under a Hierarchical Cluster-Oriented Mobile Ad-hoc Network, *Computers and Electrical Engineering*, Vol. 36, No. 1, pp. 100-113, January, 2010.

[11] S. C. Wang, K. Q. Yan, C. C. Ho, S. S. Wang, The Optimal Generalized Byzantine Agreement in Cluster-based Wireless Sensor Networks, *Computer Standards & Interfaces*, Vol. 36, No. 5, pp. 821-830, September, 2014.

[12] S. C. Wang, K. Q. Yan, H. C. Hsieh, The New Territory of Mobile Agreement, *Computer Standards & Interfaces*, Vol. 26, No. 5, pp. 435-447, September, 2004.

[13] K. Q. Yan, S. C. Wang, S. S. Wang, An Optimal Solution of Byzantine Agreement in a Scale Free Network, *Proceeding of IEEE 22nd International Conference on Advanced Information Networking and Applications (AINA 2008)*, Ginowan, Okinawa, Japan, 2008.

[14] K. Q. Yan, S. S. Wang, S. C. Wang, Reaching an Agreement under Wormhole Networks within Dual Failure Component, *International Journal of Innovative Computing, Information and Control*, Vol. 6, No. 3, pp. 1151-1164, March, 2010.

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

[16] F. Halsall, *Data Communications, Computer Networks and Open Systems*, 4th ed., Addison-Wesley, 1995.

[17] H. S. Siu, Y. H. Chin, W. P. Yang, A Note on Consensus on Dual Failure Modes, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, No. 3, pp. 225-229, March, 1996.

[18] H. S. Hsiao, 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.

[19] S. C. Wang, K. Q. Yan, S. S. Wang, G. Y. Zheng, Reaching Agreement among Virtual Subnets in Hybrid Failure Mode, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 19, No. 9, pp. 1-11, June, 2008.

[20] K. Q. Yan, S. C. Wang, Grouping Byzantine Agreement, *Computer Standard & Interfaces*, Vol. 28, No. 1, pp. 75-92, July, 2005.

[21] S. Mallela, G. M. Masson, Diagnosis without Repair for Hybrid Fault Situations, *IEEE Transactions on Computers*, Vol. 29, No. 6, pp. 461-471, June, 1980.

[22] K. Shin, P. Ramanathan, Diagnosis of Processors with Byzantine Faults in a Distributed Computing System. *Proceeding of the symposium Fault-tolerant Computing*, 1987, pp. 55-60.

[23] K. Q. Yan, S. C. Wang, Reaching Fault Diagnosis Agreement on an Unreliable General Network, *Information Sciences*, Vol. 170, No. 2-4, pp. 397-407, February, 2005.

[24] F. J. Meyer, D. K. Pradhan, Consensus with Dual Failure Modes, *IEEE Transactions on Parallel Distribute System*, Vol. 2, No. 2, pp. 214-222, April, 1991.

[25] F. Preparata, G. Metze, R. Chien, On the Connection Assignment Problem of Diagnosable Systems. *IEEE Transactions on Electronic Computers*, Vol. 16, No. 12, pp. 848-854, December, 1967.

[26] D. F. Zhang, G. G. Xie, Y. H. Min, Node Grouping in System-Level Fault Diagnosis, *Journal of Computer Science and Technology*, Vol. 16, No. 5, pp. 474-479, September, 2001.

[27] H. S. Siu, Y. H. Chin, W. P. Yang, Byzantine Agreement in the Presence of Mixed Faults on Processors and Links, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 4, pp. 335-345, April, 1998.

[28] M. L. Chiang, S. C. Wang, L. Y. Tseng, The Incremental Agreement, *Information Processing Letters*, Vol. 107, No. 5, pp. 165-170, August, 2008.

[29] R. Chen, J.-M. J. Park, K. Bia, Robustness against Byzantine Failures in Distributed Spectrum Sensing, *Computer Communications*, Vol. 35, No. 17, pp. 2115-2124, October, 2012.

[30] S. C. Wang, S. S. Wang, K. Q. Yan, L. H. Chang, C. P. Huang, Reaching Fast Agreement in a Generalized Cloud Computing Environment, *Journal of Internet Technology*, Vol. 11, No. 7, pp. 975-984, December, 2010.

Biographies

Hui-Ching Hsieh received her Ph.D. degree in computer science at National Tsing Hua University in Taiwan in 2010. She is an assistant professor in the Department of Information Communication at Hsing Wu University, Taiwan. Her research interests include distributed data processing, fault tolerant computing, cloud computing and Software-Defined network.

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

Wen-Chung Tsai received the Ph.D. degree in Electronics Engineering from National Taiwan University in 2011. During 2011 to 2013, he was a researcher at Industrial Technology Research Institute of Taiwan and focuses on, but not limited to 4G/LTE

(Long Term Evolution) researches. Dr. Tsai is currently a faculty of Department of Information and Communication Engineering, Chaoyang University of Technology.

Yen-Chiu Chen received her Ph.D. degree in Computer Sciences from National Tsing-Hua University, Taiwan in 2010 and B.S. degree in information management from Chung Hua University in 2004. Her research interests are in Mobile Edge Computing, Cloud Computing, Real-time scheduling theory.