Reaching Fault Diagnosis Agreement under a Hybrid Fault Model

Hsien-Sheng Hsiao, *Member, IEEE Computer Society*, Yeh-Hao Chin, *Senior Member, IEEE*, and Wei-Pang Yang, *Senior Member, IEEE*

Abstract—The goal of the fault diagnosis agreement (*FDA*) problem is to make each fault-free processor detect/locate a common set of faulty processors. The problem is examined on processors with *mixed fault model* (also referred to as *hybrid fault model*). An *evidence-based* fault diagnosis protocol is proposed to solve the *FDA* problem. The proposed protocol first collects the messages which have accumulated in the Byzantine agreement protocol as the *evidence*. By examining the collected evidence, a fault-free processor can detect/locate which processor is faulty. Then, the network can be reconfigured by removing the detected faulty processors and the links connected to these processors from the network. The proposed protocol can detect/locate the maximum number of faulty processors to solve the *FDA* problem.

Index Terms—Byzantine agreement, fault diagnosis agreement, fault-tolerant distributed system, hybrid fault model, mixed fault model.

1 Introduction

In order to maintain the performance and integrity of a distributed system, fault diagnosis models have been proposed to detect/locate faulty processors. Under a distributed environment, two fault diagnosis models, namely *nonagreement* [1], [23], [25], [32], [36] and *agreement* [33], have been presented to identify faulty processors. With the nonagreement fault diagnosis model, one or more processors can detect the faulty processors, but the detection results of one may not agree with those of the others. Conversely, the detection results of every fault-free processor eventually agree with those of the others when the fault diagnosis agreement model is used. In a highly reliable system, such as flight control system [34], each fault-free processor should have a common agreed upon a set of faulty processors in the system. After the set of faulty processors is detected/located by all the fault-free processors, the system can be reconfigured (eliminate the detected faulty processors) to make the system's state safe. Therefore, an agreement approach is more useful than a nonagreement approach for fault diagnosis in a highly reliable fault-tolerant distributed system. For this reason, the *fault diagnosis agreement* (*FDA*) problem [33] is considered in this paper. The protocol designed for the *FDA* problem should make each fault-free processor detect/locate a common set of faulty processors, and it must also satisfy the following conditions:

Consensus: All the fault-free processors identify the *common* set of faulty processors. In other words, the number of faulty processors and the identifiers of these faulty processors are identical with respect to each fault-free processor; and

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

The *FDA* problem is considered in a network model with the following properties:

1. The underlying network topology does not have to be fully connected.
2. More than one fault type can exist on the processors (also referred to as *mixed fault model* or *hybrid fault model* [6], [13], [18], [31]). According to the symptoms of faults, there are two broad classes of faults, called *dormant* and *arbitrary* faults, classified by Meyer and Pradhan [18]. A dormant fault reflects the case where the fault consists merely of omission of messages or delay in sending or relaying messages while an arbitrary fault can exhibit arbitrarily behavior.
3. The relative processor speeds and communication delays are finite and bounded, i.e., a *synchronous network* [10], [30] is considered.

For such a network model, we propose a protocol, called FDAMIX, for solving the *FDA* problem. FDAMIX is an *evidence-based* fault diagnosis approach [25]. FDAMIX first collects the messages which have accumulated in a Byzantine agreement (*BA*) protocol as *evidence*[1] and then detects/locates the common set of faulty processors by examining the collected evidence. The *BA* problem [8], [9], [12], [20], [17], [19] is one of the most important problems for designing a fault-tolerant distributed system. The goal of the *BA* protocol is to make each fault-free processor reach a common agreement in a general network. Most of the applications of a distributed system, such as clock synchronization and system reconfiguration [2], require that agreement be reached among all the fault-free processors prior to executing the cooperating tasks. Therefore, a *BA* protocol can be treated as the *building block* (primitive component) or *prestep* for distributed applications. To influence the system operations, the symptoms of a faulty processor may eventually appear in the collected messages during execution of the *BA* protocol. Thus, these symptoms can be used to detect/locate faulty processors.

Traditionally, most of the fault diagnosis protocols, either *test-based approaches* [15], [16], [22] or *evidence-based* approaches [4], [23], [25], have been designed with single fault type only. In a test-based approach, when a processor P tests a processor Q, the result of the test can *absolutely* specify the healthy condition of Q (whether fault-free or faulty). However, arbitrary faults can hide their faulty behavior and pass the test by other fault-free processors [25]; thus, test-based approaches are not suitable for arbitrary fault. The main goal of an evidence-based fault diagnosis approach is to handle arbitrary faults. Shin and Ramanathan [25] proposed an evidence-based protocol to detect faulty processors. In their protocol, all faults are treated as arbitrary faults. This treatment ignores the fact that faulty behaviors of dormant faults are more easily detectable than those of arbitrary faults. Thus, based on the discussion of the mixed fault model [13], [18], [27], [28], [31], this protocol is unable to detect the maximum number of faulty processors if dormant faults existed. Also, this protocol is a nonagreement approach.

In an asynchronous distributed system, in which message transmission times and relative processor speeds are both unbounded, Chandra and Toueg [5] proposed failure detectors for crash fault on processors. This is an important result since most of the previous research of fault diagnosis focused on synchronous network. However, the mixed fault model has not be solved in this solution. In this paper, synchronous network is considered.

1. An *evidence* is a diagnosis information received from one processor that is validated by diagnosis information received from other processors.

- *H.S. Hsiao is with the Department of Industrial Technology Education, National Taiwan Normal University, Taipei, Taiwan 106, ROC. E-mail: hssiu@ite.ntnu.edu.tw.*
- *Y.-H. Chin is with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 30043, ROC.*
- *W.-P. Yang is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan 30050, ROC.*

Manuscript received 27 June 1996; revised 27 Apr. 2000; accepted 20 June 2000.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 102045.

TABLE 1
The Different Assumptions among the Previous Work and the Proposal Protocol on the Fault Diagnosis Problem

Assumptions	Agreement		Approaches		Fault Types		
Previous Work	Non	Agreement	Test Based	Evidence Based	Dormant	Arbitrary	Mixed
Stahl *et al.* [29]	✓			✓		✓	
Mallela and Masson [15]	✓		✓		✓		
Mallela and Masson [16]	✓		✓		✓		
PCM model [22]	✓		✓		✓		
Buskens and Bianchini [4]	✓			✓	✓	✓	
Ramarao and Adams[23]	✓			✓		✓	
Shin and Ramanathan [25]	✓			✓		✓	
Chandra and Toueg [5]*	✓			✓	✓		
Wang et al. [33]		✓	✓			✓	
FDAMIX		✓		✓			✓

**: asynchronous network is considered.*

As for the agreement approach for fault diagnosis, Wang et al. [33] proposed a test-based protocol to detect the arbitrary faults. Similarly to of [13], [18], [27], [28], [31], this protocol is unable to detect the maximum number of faulty processors if the mixed fault model is considered.

Table 1 lists the differences among previous protocols and the proposed protocol. To summarize the above discussion, no existing fault diagnosis protocol can solve the *FDA* problem with hybrid fault model such as the one shown in Fig. 1. In this paper, we provide such a solution.

2 The System Model

The *FDA* problem is considered for a *synchronous* network in which the bounds on processing and the communication delays of fault-free components are finite [10], [30]. The notations are summarized below:

- N: the set of all processors, the processor's identifier is unique, and $|N| = n$.
- T: the Transmitter of a *BA* protocol.
- V: the set of all possible values of a *BA* protocol.
- v_t: the initial value of T to be broadcast to all other processors, and $v_t \in V$.
- P_a: the number of processors subjected to an arbitrary fault.
- P_d: the number of processors subjected to a dormant fault.
- c: the *connectivity* of the underlying network. Following the Menger theorem [7], at least c disjoint paths exist between any pairs of processors S and R if the connectivity of the network is c. For any two paths, the only common components are S and R.

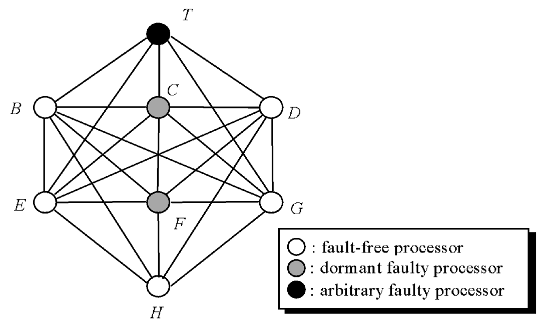

Fig. 1. A network with eight processors and five connectivity.

Since the detected results among the fault-free processors should be common, the constraints on failures for the *FDA* problem should follow those of the *BA* problem. Based on the constraints on failures for the *BA* problem stated in [27], [28], the *FDA* problem can be solved if:

(***CF1***) $n > 3P_a + P_d$ and

(***CF2***) $c > 2P_a + P_d$.

The first constraint specifies the total number of processors required. After the influence of the dormant faults is removed the fault detection agreement can be reached if $n - P_d > 3P_a$, namely $n > 3P_a + P_d$ [27]. On the other hand, the second constraint specifies the required connectivity. In order to decide whether a processor has sent out its message, the total number of arbitrary faults must be less than half of $c - P_d$ (after removing the influence of dormant faults), namely $c > 2P_a + P_d$. Based on these two constraints, namely *CF1* and *CF2*, the maximum number of detectable/locatable faulty processors by FDAMIX is $P_a + P_d$.

3 Concept and Approaches

To solve the fault diagnosis agreement (*FDA*) problem, the proposed protocol, called FDAMIX, first collects the received messages in the *BA* protocol GPBA [27], designed for mixed fault model, as evidence, and examines this evidence to detect/locate faulty processors. Hence, the GPBA protocol will be described first and then the approaches used by FDAMIX will be presented in the following subsection.

3.1 The GPBA Protocol

The GPBA protocol can solve the *BA* problem when the constraints on failures, namely $n > 3P_a + P_d$ and $c > 2P_a + P_d$, hold. In the *BA* problem, let processor T be the *transmitter* and v_t be the initial value of T to be broadcast to all other processors. v_t belongs to a finite set, V, of possible values. After execution of GPBA, the common value of the fault-free processors shall be the value defined in the following conditions:

Agreement: All fault-free processors agree on the same common value v.

Validity: If the transmitter is fault-free, then the common value v should be the initial value v_t of the source, i.e., $v = v_t$.

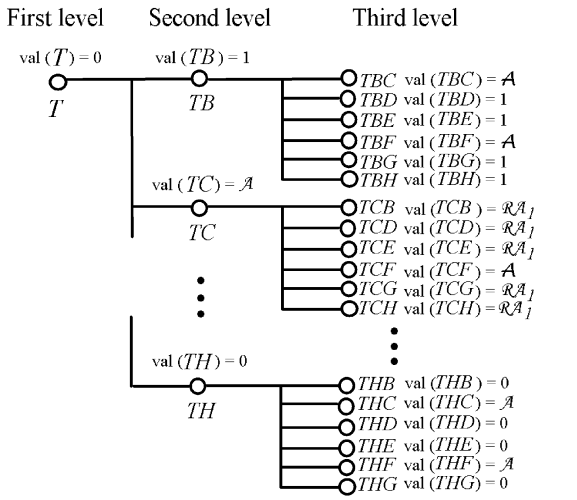

Fig. 2. The messages which have accumulated in GPBA for processor H in Fig. 1.

GPBA uses $\lfloor (n-1)/3 \rfloor + 1$ message exchange rounds to collect the messages for computing the common value. Fig. 2 illustrates the messages which have accumulated in GPBA, stored in an *Information Gathering tree* (IG-tree) [3], [27], of level $\lfloor (n-1)/3 \rfloor + 1$, of a fault-free processor, say H, in the network shown in Fig. 1. The messages which have accumulated in the other fault-free processors are similar. The IG-tree shown in Fig. 2 is constructed and labeled as follows: After the first message exchange round, processor H stores the message "0" received from transmitter T, denoted as $\mathrm{val}(T) = 0$, at the root T of its IG-tree. In the second round, each processor broadcasts the root's value of its IG-tree to all the processors. If processor B sends a message $\mathrm{val}(T)$ ($\mathrm{val}(T) = 1$) to processor H, then H will store the message received from B, denoted as $\mathrm{val}(TB) = 1$, at vertex TB of its IG-tree. Vertex TB is said to correspond to processor B. The processes of the third round are similar to the second round. Note that each level of an IG-tree contains a round of received messages and each vertex is labeled by a nonrepeating sequence of processor identifiers. Because the label of an IG-tree is nonrepeating, the root (labeled by the transmitter) has $n-1$ children and a vertex at the tth level has $n-t$ leaves, as shown in Fig. 2.

In Fig. 2, suppose that the dormant faulty processors C and F do not send any messages during the entire execution of GPBA. To remove the influence of processors C and F, the absent rule uses the values $\mathcal{A}$ and $\mathcal{RA}_1$ to replace the messages received, directly or indirectly, from C and F as shown in Fig. 2. With the messages which have accumulated in GPBA, FDAMIX can be derived and is proposed in the next subsection.

3.2 Concept and Approaches of FDAMIX

FDAMIX examines the messages which have accumulated from GPBA for detecting/locating faulty processors. In each round of GPBA, a processor should broadcast its received messages. Therefore, a processor can be detected/located as faulty by using the following evidence: 1) It did not send out its messages; 2) it sent an illegal message to the other processors; or 3) it sent different messages to different processors. Using this evidence, we propose two rules, namely *the local fault detection/location rule* ***LR*** and *the global fault detection/location rule* ***GR***, to identify the faulty processors. We distinguish between *local detection/location*, which describes the detection of a single processor by examining the *locally* collected messages, and *global detection/location*, in which all the fault-free processors have detected/located a common set of faulty processors. With these two rules, FDAMIX consists of

1. the Message-collection,
2. Agreed-on,
3. Fault-diagnose, and
4. Reconfiguration steps,

used to detect/locate the faulty processors. The main functions of these steps are shown in Fig. 3.

FDAMIX is illustrated by means of an example, executed on processor H in the network shown in Fig. 1, that uses the messages which have accumulated in GPBA, as shown in Fig. 2. The same procedure is executed by each processor. When FDAMIX is finished, all the fault-free processors agree on a common set of faulty processors $\{T, C, F\}$ and do not falsely detect any fault-free processors as faulty, i.e., the conditions of *Consensus* and *Fairness* of the *FDA* problem are both satisfied. Hence, FDAMIX does solve the *FDA* problem.

3.2.1 Step 1: The Message-Collection Step

- Collect the messages which have accumulated in GPBA as evidence

  The goal of the Message-collection step is to collect the messages which have accumulated in GPBA and to use the collected messages as evidence for detecting/locating faulty processors. Following the example of GPBA, processor H collects the IG-tree, the messages which have accumulated in GPBA, shown in Fig. 2, as evidence.
- Reach the set of local detected/located faults LDF_H.

  To influence the execution of GPBA, a faulty processor may exhibit faulty behavior by sending unhealthy messages to other processors during the message exchange phase of GPBA.[2] Therefore, a fault-free processor can detect/locate faulty processors by examining the messages collected from GPBA (the evidence). Using the following *local fault detection/location rule* ***LR***, the fault-free processor H can detect whether processor Q is faulty.

The Local Fault Detection/Location Rule LR. Let H be a fault-free processor. H can identify the faulty processor Q if:

***LR1*:** H receives the value A from Q (no message received from Q); or

***LR2*:** H receives a message m sent by Q and the number of received copies of m (by FTVC) is not greater than $(c - \mathrm{NULL})/2$, where NULL is the number of copies of null message received by H; or

***LR3*:** H receives a message m sent by Q and $m \notin V\{\mathcal{RA}_i\}$, where $1 \le i < \lfloor (n-1)/3 \rfloor$.

The conditions *LR1*, *LR2*, and *LR3* identify that the faulty processor Q has not sent out its message to H, sent inconsistent messages from different paths to H, or sent an illegal message to H, respectively. By *LR1*, processor H can detect/locate the faulty processors C and F because the value $\mathcal{A}$ is stored in the vertices TC and TF in the second level of the IG-tree of Fig. 2 (i.e., H does not receive any messages from C and F). The processor identifiers of C and F are added to the set of *local detected/located faults* LDF_H (the subscript is omitted when no confusion will arise), i.e., $LFD_H = LFD_H \cup \{C, F\}$. The messages received from the processors included in LDF_H will be ignored in the subsequent steps of FDAMIX. This means that the faulty processors C and F are treated as absentees (no further influence on processor H). At the end of the Message-collection step, an IG-tree, shown in Fig. 2, and LFD_H are passed to the Agreed-on step.

2. In GPBA, a fault-free processor uses FTVC to send c copies of its messages through c disjoint paths to other processors [27].

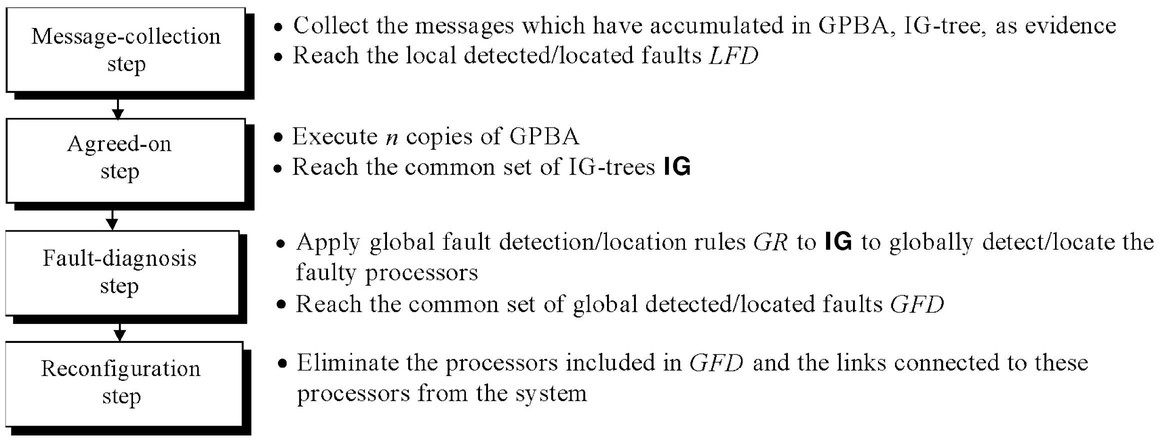
Message-collection step
• Collect the messages which have accumulated in GPBA, IG-tree, as evidence
• Reach the local detected/located faults LFD
Agreed-on step
• Execute n copies of GPBA
• Reach the common set of IG-trees IG
Fault-diagnosis step
• Apply global fault detection/location rules GR to IG to globally detect/locate the faulty processors
• Reach the common set of global detected/located faults GFD
Reconfiguration step
• Eliminate the processors included in GFD and the links connected to these processors from the system

Fig. 3. The procedure of FDAMIX.

3.2.2 Step 2: The Agreed-On Step

- Execute n copies of GPBA.

  In order to make each fault-free processor obtain a common set of detected faulty processors, each processor should exchange the messages collected in the Messages-collection step with every other processor. Thus, at the Agreed-on step, each processor executes GPBA with its IG-tree as the initial value (n copies of GPBA are executed totally). To remove contamination due to detected faulty processors (included in LFD_H), the received messages from processors C and F are ignored during execution of the Agreed-on step. This means that H uses value $\mathcal{A}$ to replace the messages received from C and F as shown in the IG-tree$_C$ and IG-tree$_F$ of Fig. 4.
- Reach the common set of IG-trees **IG**.

  After these GPBA protocols are executed, the set of IG-trees $\mathbf{IG} = [\textit{IG-tree}_T, \textit{IG-tree}_B, \ldots, \textit{IG-tree}_H]$, as shown in Fig. 4, can be obtained by the fault-free processor H, where $\textit{IG-tree}_i$ corresponds to the IG-tree broadcast by processor I. By the agreement and validity conditions of the *BA* problem stated in Section 2, the set of IG-trees **IG** obtained by the fault-free processors are common (each fault-free processor has the same **IG** as shown in Fig. 4). Then, **IG** and LFD_H are passed to the Fault-diagnosis step.

3.2.3 Step 3: The Fault-Diagnosis Step

- Apply the global fault detection/location rules ***GR*** to **IG**.

  The goal of this step is to make each fault-free processor obtain a common set of faulty processors. The fault-free processor H applies the *global fault detection/location rule* ***GR*** to **IG** to globally detect/locate faulty processors. By means of a *top-down and level-by-level* sequence, ***GR*** examines the values stored in the same labeled vertices σ ($= \alpha Q$, the vertices correspond to processor Q) of **IG** to detect whether Q is faulty. Formally, *GR* can be defined as follows:

The Global Fault Detection/Location Rule GR. Let H be a fault-free processor. H can detect processor Q as faulty if:

GR1: the most common value stored in all σ of **IG** is $\mathcal{A}$; or

GR2: the number of the most common value stored in all σ $(= \alpha Q)$ of **IG** is not greater than $n - (n_A + \lfloor (n - n_A - 1)/3 \rfloor)$, where n_A is the number of value $\mathcal{A}$ stored in all σ.

Semantically, condition *GR1* identifies that processor Q has not sent out its message to major number of fault-free processors, while *GR2* identifies that Q has sent sufficiently different messages to different processors. According to the constraint on processors, namely $n > 3P_a + P_d$, more than $n - (n_A + \lfloor (n - n_A - 1)/3 \rfloor)$ fault-free processors should receive the identical message sent by processor Q if Q is fault-free (n_A processors do not send out their messages); otherwise, Q can be globally detected/located as faulty by *GR2* (Q does send sufficiently different messages to different processors).

Examining this in a top-down and level-by-level sequence, ***GR*** first examines the values stored in vertices T of **IG** as shown in Fig. 4. The value stored in vertices T of IG-tree$_T$ is 1, that of IG-tree$_B$ is 1, that of IG-tree$_C$ is $\mathcal{A}$, ..., and that of IG-tree$_H$ is 0, as shown in the first level of the IG-trees of **IG** in Fig. 4. In other words, the global values stored in vertices T of all the processors' IG-trees examined by *GR* are $(1, 1, \mathcal{A}, 1, 1, \mathcal{A}, 0, 0)$. Using *GR2*, therefore, processor T can be globally detected/located as faulty because the number of the most common value "1" stored in vertices T is 4, which does not exceed 5 $(= n - (n_A + \lfloor (n - n_A - 1)/3 \rfloor) = 8 - (2 + \lfloor (8 - 2 - 1)/3 \rfloor)$, where n_A is 2). Next, ***GR*** examines the values stored in all the vertices labeled TB at the second level of all the IG-trees shown in Fig. 4. The corresponding eight values stored in TB are $(1, 1, \mathcal{A}, 1, 1, \mathcal{A}, 1, 1)$ and the number of the most common value "1" of TB is 6, so neither *GR1* nor *GR2* is satisfied. Hence, the fault-free processor B will not be falsely detected as faulty by processor H. Similarly, ***GR*** examines the values stored in vertices TC, namely $(\mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A}, \mathcal{A})$. By *GR1*, processor H can globally detect/locate that the processor C is in fault.[3] This procedure continually examines the values stored in all the vertices of **IG** until all the vertices at the last level are examined by ***GR***. After this procedure is finished, the faulty processors T, C, and F have been detected/located by processor H.

- Reach the common set of global detected/located fault GFD_H.

  When the faulty processors T, C, and F are globally detected/located by fault-free processor H, the identifiers of these faulty processors are added to the set of *global detected faults* GFD_H (the subscript is omitted when no confusion will arise), i.e., $GFD_H = GFD_H \cup \{T, C, F\}$.

3. The major goal is to detect/locate which processor is in fault and where it is located. The fault type is irrelevant and processor C's fault type can even be treated as dormant.

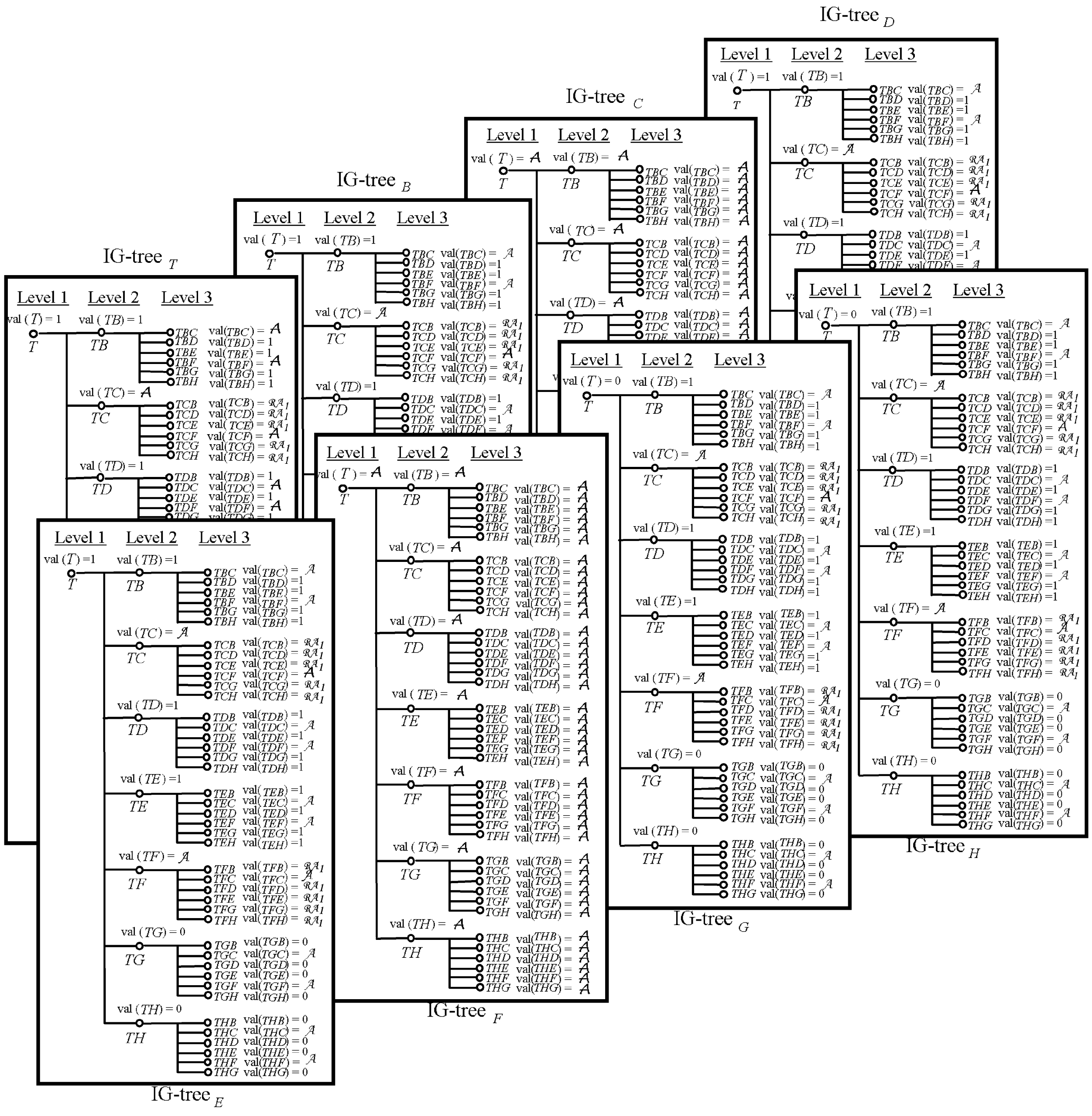

Fig. 4. The set of IG-trees IG obtained for processor H.

3.2.4 Step 4: The Reconfigured Step

After the processor H obtains the set of faulty processors $\{T, C, F\}$, the configuration of the network can be reconstructed by eliminating the processors T, C, and F, and the links connected to these faulty processors can also be eliminated, as shown in Fig. 5. Finally, set $LFD_H = LFD_H - GFD_H$ and $GFD_H = \{\}$. Since Transmitter T is globally detected as faulty, a *leader election protocol* [11], [26] should be executed by the fault-free processors to elect a new Transmitter.

4 Analysis and Evaluation

FDAMIX is an evidence-based fault diagnosis protocol which not only detects faulty processors that can be subjected to mixed faults, but also makes each fault-free processor obtain a common set of faulty processors. The following primitives are used to construct FDAMIX.

- COLLECT_MESSAGE(m, Q): Collect message m sent by processor Q.
- CREATE(Q, v): Create the vertex Q and set $\text{val}(Q) = v$.
- UNFOLD(m, r): According to the structure of the rth level of IG-tree, unfold the message m.
- LR(σ): Apply the local fault detection rules *LR* to the vertex σ of IC-tree.
- GR(σ): Apply the global fault detection rules *GR* to the vertex σ of the set of IC-trees.
- GPBA(T, vt): The transmitter T starts the GPBA protocol with the initial value v_t.
- ELIMINATE(Q): Eliminate processor Q and the links connected to Q from the network.

Using the above primitives, the formal presentation of FDAMIX can be stated as follows:

Protocol FDAMIX (for each processor P)
begin

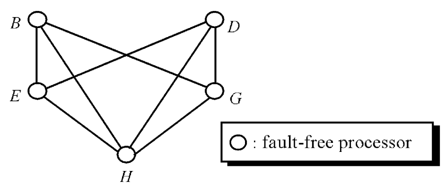

Fig. 5. After the Reconfiguration step.

```
/* Step 1: Message-Collection Step */
    COLLECT_MESSAGE($m, T$);
    CREATE($T, m$);
    LR($T$);
    for $r = 2$ to $\lfloor (n-1)/3 \rfloor + 1$ do
    begin
        for each $Q \in N$ do
        begin
            COLLECT_MESSAGE($m, Q$);
            UNFOLD($m, r$);
            for each $\alpha \in m$ do
            begin
                $v = \mathrm{val}(\alpha)$;
                CREATE($\alpha Q, v$);
                LR($\alpha Q$);
            end
        end;

/* Step 2: Agreed-On Step */
    /* $n$ copies of GPBA are executed in parallel */
    for each $Q \in N$ do PARALLEL
        GPBA($Q, IG\text{-}tree_Q$);
    /* The common set of IG-trees is obtained */
    set $\mathbf{IG} = [IG\text{-}tree_1, \ldots, IG\text{-}tree_Q, \ldots, IG\text{-}tree_n]$;

/* Step 3: Fault-Diagnose Step */
    for each vertex $\alpha \in$ IG-tree do
        GR($\alpha$);

/* Step 4: Reconfiguration Step */
    for each $Q \in GFD_p$ do
        ELIMINATE($Q$);
    set $N = N - GFD_p$;
    set $LFD_p = LFD_p - GFD_p$;
    set $GFD_p = \{\}$;
end.
```

4.1 Correctness

The goal of FDAMIX is to solve the *FDA* problem; hence, FDAMIX can be proven from the fact that the set of global detected/located faulty processors satisfies the conditions of *consensus* and *fairness* stated in Section 1. The basic concept for proving the correctness of FDAMIX is as follows: Following GPBA, Lemma 1 shows that the common set of IG-trees IG can be reached among each fault-free processor. Using ***LR*** and ***GR***, a fault-free processor does not treat other fault-free processors as faulty; thus, FDAMIX satisfies the fairness condition as stated in Lemma 2. Lemma 3 shows that each fault-free processor can *globally* detect/locate a faulty processor R by using FDAMIX if R has not sent out its messages, sent an illegal message to all processors, and/or sent different messages to different processors, i.e., the consensus condition is proven to be satisfied. Both consensus and fairness conditions are satisfied; thus, FDAMIX does solve the *FDA* problem as stated in Theorem 1. For space considerations, we omit all detailed proofs.

Lemma 1. *After the Agreed-on step, each fault-free processor obtains a common set of IG-trees* **IG** *if* $n > 3P_a + P_d$ *and* $c > 2P_a + P_d$.

Lemma 2. *FDAMIX satisfies the fairness condition if* $n > 3P_a + P_d$ *and* $c > 2P_a + P_d$.

Lemma 3. *FDAMIX satisfies the consensus condition if* $n > 3P_a + P_d$ *and* $c > 2P_a + P_d$.

Theorem 1. *FDAMIX does solve the* FDA *problem if* $n > 3P_a + P_d$ *and* $c > 2P_a + P_d$.

The complexity of FDAMIX is defined in terms of 1) the number of messages required and 2) the number of detectable faulty processors. The following lemma and theorems state that FDAMIX uses $O(cn^2 + tcn^3)$ messages to solve the FDA problem and that FDAMIX can detect/locate the maximum number of faulty processors as stated in Theorem 2.

Lemma 4. *FDAMIX solves the* FDA *problem by using* $O(cn^2 + tcn^3)$ *messages.*

Theorem 2. *The total number of detectable/locatable faulty processors by FDAMIX, namely* $P_a + P_d$, *is maximum if* $n > 3P_a + P_d$ *and* $c > 2P_a + P_d$.

5 Conclusions and Discussion

An evidence-based fault diagnosis protocol FDAMIX has been proposed to solve the *FDA* problem with mixed fault model on processors. FDAMIX collects the messages which have accumulated in a *BA* protocol (GPBA) as evidence for detecting/locating faulty processors. FDAMIX can make each fault-free processor obtain a common set of faulty processors. It can detect/locate the maximum number of tolerable faulty processors as stated in Theorem 3.

Shin and Ramanathan [25] proved that no fault diagnosis protocol for arbitrary faults is *complete*—all arbitrary faults can be detected. Following this property, FDAMIX is not complete. For example, an arbitrary faulty processor Q always sends different messages to different processors, but $\lfloor (n - P_d - 1)/3 \rfloor$ copies of these messages are identical. By ***LR*** and ***GR***, Q cannot be globally detected as faulty because *insufficient evidence* can be obtained to accuse Q. However, FDAMIX still satisfies the consensus and fairness conditions as stated in Theorem 1. Since the time of fault occurrence is unpredictable, a faulty processor Q can exhibit faulty behavior during the message exchanging of the Agreed-on step. However, the faulty behaviors of Q in the Agreed-on step do not effect the FDAMIX to solve the *FDA* problem. When Q is a dormant fault, the messages received from Q are replaced by value $\mathcal{A}$ and value $\mathcal{A}$ is ignored in the Fault-diagnosis step. Thus, the contamination of Q can be removed and the common set of faulty processors can be detected/located in the Fault-diagnosis step. If Q has an arbitrary fault, it may send different IG-tree$_Q$ to different processors in the Agreed-on step. However, Q cannot influence the IG-trees sent by faulty-free processors and a fault-free processor sends its IG-tree to all the processors. By the definition of the Agreement condition of the *BA* problem, each fault-free processor can obtain a common set of IG-trees IG after the Agreed-on step is executed as stated in Lemma 1. Using ***GR***, the common set of faulty processors can be reached by each fault-free processor in the Fault-diagnosis step as shown in Fig. 4.

Since FDAMIX is designed for processor failures only, link failures [21], [24], [26], [34], [35] are treated as processor faults. Because the link diagnosis is important in a distributed system

[14], [21], [26], [34], [35], our future work will extend FDAMIX to handle the case where both processors and links are subjected to mixed faults.

References

[1] J.C. Adams and K.V.S. Ramarao, "Distributed Diagnosis of Byzantine Processors and Links," *Proc. Symp. Distributed Computing Systems,* pp. 562-569, 1989.

[2] M. Barborak, M. Malek, and A. Dahbura, "The Consensus Problem in Fault-Tolerant Computing," *ACM Computing Surveys,* vol. 25, no. 2, pp. 171-220, June 1993.

[3] A. Bar-Noy, D. Dolev, C. Dwork, and R. Strong, "Shifting Gears: Changing Algorithms on the Fly to Expedite Byzantine Agreement," *Information and Computation,* vol. 97, pp. 205-233, 1992.

[4] R.W. Buskens and R.P. Bianchini, "Distributed On-Line Diagnosis in the Presence of Arbitrary Faults," *Proc. Symp. Fault-Tolerant Computing,* pp. 470-479, 1993.

[5] T. Chandra and S. Toueg, "Unreliable Failure Detectors for Asynchronous Systems," *Proc. 10th ACM Symp. Principles of Distributed Computing,* pp. 325-340, 1991.

[6] F. Cristian, "Understanding Fault-Tolerant Distributed Systems," *Comm. ACM,* vol. 34, no. 2, pp. 57-78, Feb. 1991.

[7] N. Deo, *GRAPH THEORY with Applications to Engineering and Computer Science.* Englewood Cliffs, N.J.: Prentice Hall, 1974.

[8] D. Dolev, "The Byzantine Generals Strike Again," *J. Algorithms,* vol. 3, no. 1, pp. 14-30, 1982.

[9] M. Fischer and N. Lynch, "A Lower Bound for the Assure Interactive Consistency," *Information Processing Letters,* vol. 14, no. 4, pp. 183-186, June 1982.

[10] M. Fischer, M. Paterson, and N. Lynch, "Impossibility of Distributed Consensus with One Faulty Process," *J. ACM,* vol. 32, no. 4, pp. 374-382, Apr. 1985.

[11] H. Garcia-Molina, "Election in a Distributed Computing System," *IEEE Trans. Computers,* vol. 31, no. 1, pp. 48-59, Jan. 1982.

[12] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Programming Languages and Systems,* vol. 4, no. 3, pp. 382-401, July 1982.

[13] P. Lincoln and J. Rushby, "A Formally Verified Algorithm for Interactive Consistency under a Hybrid Fault Model," *Proc. Symp. Fault-Tolerant Computing,* pp. 402-411, 1993.

[14] J. Martin, *Telecommunications and the Computer,* third ed. Englewood Cliffs, N.J.: Prentice Hall, 1990.

[15] S. Mallela and G.M. Masson, "Diagnosable Systems for Intermittent Faults," *IEEE Trans. Computers,* vol. 27, no. 6, pp. 560-566, June 1978.

[16] S. Mallela and G.M. Masson, "Diagnosis without Repair for Hybrid Fault Situations," *IEEE Trans. Computers,* vol 29, no. 6, pp. 461-470, June 1980.

[17] B.M. McMillin et al., "Byzantine Fault-Tolerance through Application Oriented Specification," *Proc. Computer Software and Application Conf.,* pp. 347-353, 1987.

[18] F.J. Meyer and D.K. Pradhan, "Consensus with Dual Failure Modes," *IEEE Trans. Parallel and Distributed Systems,* vol. 2, no. 2, pp. 214-222, 1991.

[19] H.G. Molina, F. Pittelli, and S. Davidson, "Applications of Byzantine Agreement in Database Systems," *ACM Trans. on Data Systems,* vol. 11, no. 1, pp. 27-47, Mar. 1986.

[20] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in Presence of Faults," *J. ACM,* vol. 27, no. 2, pp. 228-234, Apr. 1980.

[21] A. Pelc, "Reliable Communication in Networks with Byzantine Link Failures," *NETWORKS,* vol. 22, no. 5, pp. 441-459, Aug. 1992.

[22] F. Preparata, G. Metze, and R. Chien, "On the Connection Assignment Problem of Diagnosable Systems," *IEEE Trans. Computers,* vol. 16, no. 6, pp. 848-854, 1967.

[23] K.V.S. Ramarao and J.C. Adams, "On the Diagnosis of Byzantine Faults," *Proc. Symp. Reliable Distributed Systems,* pp. 144-153, 1988.

[24] V. Ramaswami and J.L. Wang, "Analysis of the Link Error Monitoring Protocols in the Common Channel Signaling Network," *IEEE/ACM Trans. Networking,* vol. 1, no. 1, pp. 31-47, Feb. 1993.

[25] K. Shin and P. Ramanathan, "Diagnosis of Processors with Byzantine Faults in a Distributed Computing Systems," *Proc. Symp. Fault-Tolerant Computing,* pp. 55-60, 1987.

[26] G. Singh, "Leader Election in the Presence of Link Failures," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 3, pp. 231-236, Mar. 1996.

[27] H.S. Siu, Y.H. Chin, and W.P. Yang, "Byzantine Agreement in the Presense of Mixed Faults on Processors and Links," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 4, pp. 335-345, Apr. 1998.

[28] H.S. Siu, Y.H. Chin, and W.P. Yang, "A Note on Consensus on Dual Failure Modes," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 3, pp. 225-230, Mar. 1996.

[29] M. Stahl, R. Buskens, and R. Bianchini, "On-Line Diagnosis in General Topology Networks," *Proc. 1992 IEEE Workshop Fault-Tolerant Parallel and Distributed Systems,* pp. 114-121, 1992.

[30] N. Suri, M.M. Hugue, and C.J. Walter, "Synchronization Issues in Real-Time Systems," *Proc. IEEE,* vol. 82, no. 1, pp. 41-53, Jan. 1994.

[31] P. Thambidurai and Y.-K. Park, "Interactive Consistency with Multiple Failure Modes," *Proc. Symp. Reliable Distributed Systems,* pp. 93-100, Oct. 1988.

[32] N.H. Vaidya and D.K. Pradhan, "Safe System Level Diagnosis," *IEEE Trans. Computers,* vol. 43, no. 3, pp. 367-370, Mar. 1994.

[33] S.C. Wang, Y.H. Chin, and K.Q. Yan, "Reaching a Fault Detection Agreement," *Proc. Int'l Conf. Parallel Processing,* pp. 251-258, 1990.

[34] K.Q. Yan, Y.H. Chin, and S.C. Wang, "Optimal Agreement Protocol in Byzantine Faulty Processors and Faulty Links," *IEEE Trans. Knowledge and Data Eng.,* vol. 4, no. 3, pp. 266-280, June 1992.

[35] C.L. Yang and G.M. Masson, "Hybrid Fault Diagnosability with Unreliable Communication Link," *IEEE Trans. Computers,* vol. 37, no. 2, pp. 175-181, Feb. 1988.

[36] C.L. Yang and G.M. Masson, "A Distributed Algorithm for Fault Diagnosis in Systems with Soft Failures," *IEEE Trans. Computers,* vol. 37, no. 11, pp. 1,476-1,480, Nov. 1988.