Information Technology Journal 3 (3): 275-282, 2004
ISSN 1682-6027

Detecting the Faulty Communication Links under the Loose Coupled Distributed System

[1]K.Q. Yan and [2]S.C. Wang
[1]Department of Business Administration,
[2]Department of Information Management,
Chaoyang University of Technology, No.168, Jifong E. Rd.,
Wufong Township, Taichung County 41349, Taiwan, Peoples Republic of China

Abstract: Our previous study solved the Byzantine Agreement (BA) problem in the presence of mixed faults on processors and links in the general network by proposed protocol GPBA. After that, they also solve the Fault Diagnosis Agreement (FDA) problem with mixed faults on processor by proposed protocol FDAMIX. Nevertheless, the protocol FDAMIX is designed for the FDA problem with faulty processors only. Therefore, the FDA problem with faulty links remains to be solved. That is, when the agreement is achieved by the GPBA, then executes the protocol FDAMIX can let the general network without faulty processors, but the faulty links are still existed in the general network. In this study, the proposed protocol FDAML can detect/locate the faulty links to solve the FDA problem and reconfigure the unstable general network be a stable general network.

Key words: Fault diagnosis agreement, fault tolerance, loose coupled, distributed system, mixed fault model

INTRODUCTION

A loose coupled distributed system is a collection of processors that do not share memory or a clock and each processor can communicate with each other processors through communication links in the un-fully connected network[1]. To cope the influences from faulty components, a distributed system must provide a mechanism to allow all fault-free processors in the distributed system to reach a common agreement[2]. Such an agreement problem was also called as the Byzantine Agreement (BA) problem[3-7]. Siu *et al.*[8] solve the BA problem in the presence of mixed faults on processors and links in the general network by protocol GPBA. The general network is that the network topology may not be fully connected and processors and links can both be subjected to different types of failures (dormant fault and arbitrary fault) simultaneously[9-11]. After reaching a common agreement by the protocol GPBA, each fault-free processor in the network can cope with the influences from faulty components.

In order to provide more stable environment, reaching a common agreement is not enough, so we must visit another related problem which called as the Fault Diagnosis Agreement (FDA) problem[12-14]. The FDA is used to detect/locate the faulty components in the network. Therefore, Siu *et al.*[12] proposed the protocol FDAMIX to solve the FDA problem with mixed faults on processors in the general network. That is, the protocol FDAMIX can detect/locate the dormant and arbitrary faulty processors in the general network.

However, the faulty components in the network can be classified into two catalogues; they are the processors and communication links[6,8]. The FDAMIX can solve the FDA problem with mixed faults on processors. However, the FDA problem with mixed faults on links in the general network remains to be solved. Therefore, this study, focused the FDA problem with mixed faults on links in the general network and the proposed protocol which called fault diagnosis agreement with mixed faults on links (FDAML) can detect/locate the maximum number of dormant faulty links and arbitrary faulty links.

The conditions for FDA problem: The FDA problem is considered in a synchronous network, so the processing time and communication time of each fault-free components are finite[15]. Moreover, the symptoms of a faulty communication link can be divided into two types. They are dormant fault (include crash and stuck-at) and arbitrary fault. In the synchronous system, each fault-free processor can detect the messages from dormant faulty communication link, if the protocol appropriately encodes a transmitted message by Manchester code[16] before transmission. Because, in the Manchester encoding, logic

Corresponding Author: S.C. Wang, Department of Information Management, Chaoyang University of Technology, No.168, Jifong E. Rd., Wufong Township, Taichung County 41349, Taiwan, Peoples Republic of China
Tel: 886-4-2374-2308 Fax: 886-4-2374-2309 E-mail: scwang@cyut.edu.tw

0 is indicated by a 0 to 1 (upward transition at bit centre) transition at the centre of the bit and logic 1 is indicated by a 1 to 0 (downward transition at bit centre) transition at the centre of the bit[16]. Due to a crash fault happens when a component is broken, so the component would not send any signal to the receiver processor. A stuck-at fault happens when the signal from a certain communication link is always constant. Therefore, a fault-free processor can detect the crash fault and stuck-at fault if the protocol encodes a transmitted message by Manchester code before transmission. However, the message from the arbitrary faulty communication link is arbitrary and unrestricted, so it is the most damaging failure types of all and causes the worst problem.

The assumptions and parameters of our protocols

- The underlying network is synchronous.
- Let Þ be the set of all processors in the network and .Þ.= n, where, n is the number of processors in the general network and n.4.
- Each processor in the network can be identified uniquely.
- Each processor in the network are fault-free, this can be achieved by using the FDA protocol FDAMIX[16].
- The communication links in the network are assumed fallible.
- A processor that transmits messages is called a sender processor.
- Each processor in the network has the same initial value; this can be achieved by using the BA protocol GPBA[6].
- The proposed protocol encodes a transmitted message by Manchester code before transmission.
- Let a be the maximum number of arbitrary faulty communication links allowed.
- Let d be the maximum number of dormant faulty communication links allowed.
- Let c be the connectivity of the general network.
- A processor does not know the faulty status of communication links, but the message(s) received through dormant faulty communication links can be detected.

The constraints: Since the FDAML is based on the BA protocol GPBA and FDA protocol FDAMIX. Therefore, the constraints of the FDAML are also based on the GPBA and FDA. The proposed FDAML can solve the FDA problem in the general network if $c > 2a+d$.

The proposed protocol: In this section, the proposed FDAML is used to detect/locate the dormant faulty communication link and arbitrary link in the general network. There are three phases in the FDAML; there are the message exchange phase, fault diagnosis phase and reconfiguration phase. The message exchange phase is used to collect the messages received from all processors in the network. The fault diagnosis phase is used to diagnosis the messages received in the message exchange phase to find out the faulty communication links. The reconfiguration phase is used to reconfiguration the network by eliminating the faulty communication links logically. The definition of FDAML is shown in Fig. 1.

The message exchange phase: In the message exchange phase, the FDAML transmits the message(s) by using the concept of virtual channel; the virtual channel can let an un-fully connected network work just like a fully connected network. Because, each processor in the network has the common knowledge of the graphic information of the entire general network and there are at least c (c>2a+ d) node-disjoint paths to each processor. An example of virtual channels between the processor P_1 and processor P_3 is in Fig. 2. Figure 2a shows a general network. The connection state of the sender processor P_1 and connection state of the destination processor P_3 is shown in Fig. 2b. The node-disjoint paths from sender processor P_1 to destination processor P_3 is shown in Fig. 2c.

In the first round of message exchange phase, each destination processor P_j collects other processors' initial value to construct the MAT_j. According to the MAT_j, processor P_j can find out the dormant faulty paths by examining the value λ in the MAT_j and sets the $d\_Path_j = d\_Path_j$.{dormant faulty path}. The processor P_j also can find out the arbitrary faulty path if the message in the MAT_j is the non-agreement value and non-λ.value and sets the $a\_Path_j = a\_Path_j$. {arbitrary faulty path}, where, $1 \le j \le n$.

In the second round of message exchange phase, each processor P_i (for $1 \le i \le n$) transmits its d_Path_i and a_Path_i to all processors through c node-disjoint paths. Then each destination processor P_j can take the majority message from c paths to get the d_Path_i and m_Path_i. Then each destination processor P_j constructs the same $D\_Path = d\_Path_1 . d\_Path_2 . \ldots . d\_Path_{n-1} . d\_Path_n$ and $A\_Path = a\_Path_1 . a\_Path_2 . \ldots . a\_Path_{n-1} . a\_Path_n$.

The fault diagnosis phase: In the fault diagnosis phase, each processor can detect/locate the dormant faulty links by examining the D_Paths to find out the paths which only contain one link and set the D_Links = D_Links . {dormant faulty links}.

Each processor also can detect/locate the arbitrary link by examining the A_Paths to find out the links which

Protocol FDAML (For all fault-free processor P_i with the same initial value, $1 \le i \le n$.)

Definition:

- Each processor has the common knowledge of the graphic information of the entire general network G=(E,Þ), where, Þ is the set of processors in the network and E is a set of processor pairs (P_i,P_j) indicating a physical link between processor P_i and processor P_j, where, $1 \le i, j \le n$.
- There are at least c (c>2a+ d) paths to each processor.
- The c disjoint paths between the sender processor to destination processor can be predefined (Deo, 1974 and Meyer *et al*., 1991).
- These c paths from sender processor to destination processor are node-disjoint paths (Deo, 1974).
- Each intermediate node on these c paths should not be passed through more than once.

Message exchange phase:

Round 1

Step 1: Each processor P_i ($1 \le i \le n$) transmits its agreement value v_i from GPBA to all processors through c node-disjoint paths.

Step 2: The destination processor P_j can receive the vector $V_i = [v_{path\,1}, v_{path\,2}, \ldots, v_{path\,c-1}, \ldots, v_{path\,c}]$ form processor P_i for c>2a +d. (If the node-disjoint path from sender processor to destination processor passes through any dormant faulty communication links, then replace λ.)

Step 3: After receiving the vectors from all processors, the destination processor P_j can construct the matrix

$$MAT_j = \begin{bmatrix} vector: v_1 \\ vector: v_2 \\ \ldots\ldots\ldots \\ vector: v_{n-1} \\ vector: v_n \end{bmatrix}$$

Step 4: Find out the paths, which stored the value λ in the MAT_j for each processor P_j ($1 \le j \le n$).

- Set the dormant faulty path d_Path_j =Null;
- $d\_Path_j = d\_Path_j$.{dormant faulty paths}.

Find out the non-agreement value and non-λ value in the MAT_j for each processor P_j.

- Set the arbitrary faulty path a_Path_j =Null;

  $a\_Path_j = a\_Path_j$.{arbitrary faulty paths}.

Round 2

Step 1: Each processor P_i ($1 \le i \le n$) transmits its d_Path_i and a_Path_i to all processors through c node-disjoint paths.

Step 2: The destination processor P_j ($1 \le j \le n$) takes the majority messages from c node-disjoint paths to get the d_Path_i and a_Path_i.

Step 3: The destination processor P_j can construct the D_Paths by d_Path_i and A_Paths by a_Path_i from all processors P_i (for $1 \le i \le n$).

- Set the dormant faulty path D_Path =Null;

  $D\_Paths = d\_Path_1 . d\_Path_2 . \ldots . d\_Path_{n-1} . d\_Path_n$
- Set the malicious faulty path D_Path =Null;

  $A\_Paths = a\_Path_1 . a\_Path_2 . \ldots . a\_Path_{n-1} . a\_Path_n$

Fault diagnosis phase:

Step 1: Detect the dormant faulty link

Examine the D_Paths to find out the paths, which only contain one link.

- Set the dormant faulty link D_Links =Null,

  D_Links = D_Links . {dormant faulty links}.

Step 2: Detect the arbitrary faulty link

There are at most $\lfloor$(c-.D_Links.-1)/2$\rfloor$ arbitrary faulty links in the network if c>2a +d. So that, examines the A_Paths to find out the links which appear times is the top $\lfloor$(c-.D_Links.-1)/2$\rfloor$ of the A_Paths.

- Set the arbitrary faulty link A_Links =Null,

  A_Links = A_Links . {arbitrary faulty links}.

Reconfiguration phase:

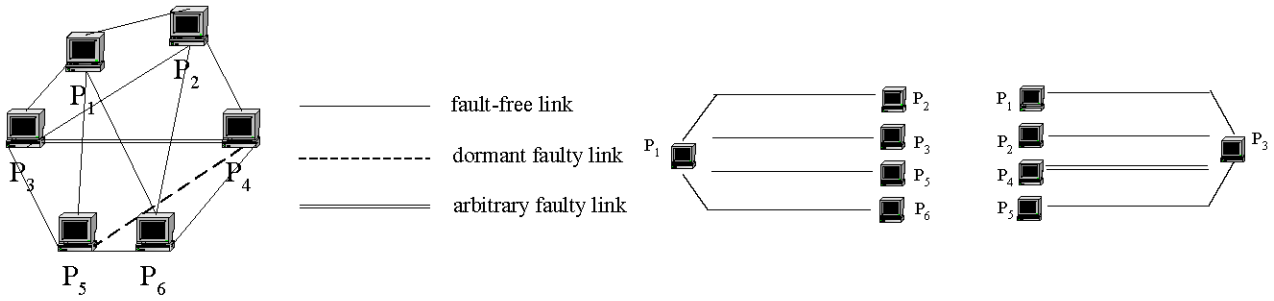
Eliminate the link in the D_Links and A_Links to reconfigure the network logically.

Fig. 1: The proposed protocol FDAML

appear times is the top $\lfloor$(c-.D_Links.-1)/2$\rfloor$ of the A_Paths and set the A_Links = A_Links . {arbitrary faulty links}. Because, there are at most $\lfloor$(c-.D_Links.-1)/2$\rfloor$ arbitrary faulty links in the network if c>2a +d. This treatment can ensure that all the arbitrary faulty communication links are detected.

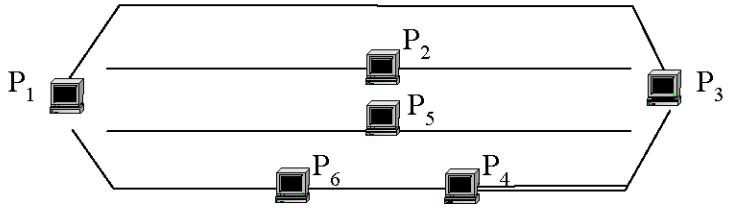
The reconfiguration phase: Each processor uses the results of D_Links and A_Links from the fault diagnosis phase to reconfigure the network by isolating the faulty communication links logically. After the reconfiguration, the distributed system can be more stable and the performance and integrity of the network can be guaranteed.

An example of executing FDAML: In this section, we give an example of executing FDAML. In Fig. 3a, there is a network with six processors and the connectivity c of the network is four. There is a dormant faulty link between processor P_3 and processor P_4 ($Link_{34}$) and an arbitrary faulty link between Processor P_1 and P_3 ($Link_{13}$). The initial value of each processor is assumed to be the same



(a) The general network with 4-connectivity

(b) The connection state of the sender processor P_1 and that of the destionation processor P_3



(c) The node-disjoint paths from sender processor P_1 to destination processor P_3

Fig. 2: An example of virtual channel between processor P_1 and processor P_3

(This can be achieved by the protocol GPBA). In this scenario, we assume that the initial value of each processor is "1" (Fig. 3b).

The message exchange phase: In the first round of message exchange phase, each processor P_i (for $1 \le i \le n$) in the network transmits its initial value to all other processors through c node-disjoint paths, then each destination processor P_j can construct the matrix MAT_j by the messages received from other processors as shown in Fig. 3c, e, g, i, k and m. Each processor P_j examines the values in the MAT_j to find out the dormant faulty paths and arbitrary faulty paths. And sets the $d\_Path_j = d\_Path_j$.{dormant faulty paths} and $a\_Path_j = a\_Path_j$.{arbitrary faulty paths}. In Fig. 3d, f, h, j, l and n, shows the d_Path_j and m_Path_j of each destination processor P_j.

In the second round of message exchange phase, each processor P_i (for $1 \le i \le n$) exchanges its d_Path_i and a_Path_i to all processors through c node-disjoint paths. Then, each processor P_j can construct the same D_Paths and A_Paths. Example of D_Paths and A_Paths are shown in Fig. 3o.

The fault diagnosis phase: Each processor examines the D_Paths in Fig. 3o to find out the $Link_{34}$ is in dormant fault then set the D_Links={$Link_{34}$}. There are at most $1(\lfloor(c\text{-}.D\_Links.\text{-}1)/2\rfloor = \lfloor(4\text{-}1\text{-}1)/2\rfloor)$ arbitrary faulty link in the network for c>2a +d. Therefore, each processor examines the A_Paths to find out the link, which appear times is the top 1 at the A_Paths. So set the A_Links={$Link_{13}$}

The reconfiguration phase: According to the D_Links and A_Links, to eliminate the $Link_{34}$ and $Link_{13}$ and then can reconfigure the network logically as shown in Fig. 3p. After reconfiguration, the performance and integrity of the network can be guaranteed.

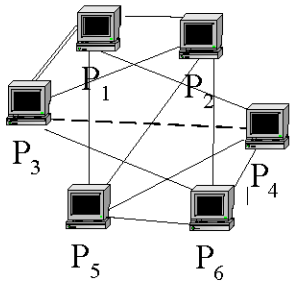
The correctness and complexity of DFAML: The following lemmas and theorems are used to prove the correctness of DFAML.

Lemma 1: Each processor in the general network is fault-free and the initial value of each processor is the same.

Proof: Executing the BA protocol GPBA by Siu *et al.*[8] can let each fault-free processor has the same initial value and after executing the FDA protocol FDAMIX by Siu *et al.*[12] can detect/locate the faulty processors in the general network.

Theorem 1: Each processor can detect/locate the dormant faulty link in the general network.

Proof: Due to the message received from dormant faulty link can be detected if the protocol encodes a transmitted message by Manchester code before transmission. If the



fault-free link

dormant faulty link

arbitrary faulty link

Fig. 3a: Network with six processors and connectivity c is 4

V_1	1
V_2	1
V_3	1
V_4	1
V_5	1
V_6	1

Fig. 3b: Initial value

The node-disjoint paths from P_1 to P_1	$P_1 \rightarrow P_2 \rightarrow P_1$	$P_1 \rightarrow P_3 \rightarrow P_1$	$P_1 \rightarrow P_4 \rightarrow P_1$	$P_1 \rightarrow P_5 \rightarrow P_1$
The message received	1	0	1	1
The node-disjoint paths from P_2 to P_1	$P_2 \rightarrow P_1$	$P_2 \rightarrow P_3 \rightarrow P_1$	$P_2 \rightarrow P_5 \rightarrow P_1$	$P_2 \rightarrow P_6 \rightarrow P_4 \rightarrow P_1$
The message received	1	0	1	1
The node-disjoint paths from P_3 to P_1	$P_3 \rightarrow P_1$	$P_3 \rightarrow P_2 \rightarrow P_1$	$P_3 \rightarrow P_4 \rightarrow P_1$	$P_3 \rightarrow P_6 \rightarrow P_5 \rightarrow P_1$
The message received	0	1	λ	1
The node-disjoint paths from P_4 to P_1	$P_4 \rightarrow P_1$	$P_4 \rightarrow P_3 \rightarrow P_1$	$P_4 \rightarrow P_5 \rightarrow P_1$	$P_4 \rightarrow P_6 \rightarrow P_2 \rightarrow P_1$
The message received	1	1	1	1
The node-disjoint paths from P_5 to P_1	$P_5 \rightarrow P_1$	$P_5 \rightarrow P_2 \rightarrow P_1$	$P_5 \rightarrow P_4 \rightarrow P_1$	$P_5 \rightarrow P_6 \rightarrow P_3 \rightarrow P_1$
The message received	1	1	1	0
The node-disjoint paths from P_6 to P_1	$P_6 \rightarrow P_2 \rightarrow P_1$	$P_6 \rightarrow P_3 \rightarrow P_1$	$P_6 \rightarrow P_4 \rightarrow P_1$	$P_6 \rightarrow P_5 \rightarrow P_1$
The message received	1	0	1	1

Fig. 3c: The message received by P_1 in the first round of message exchange phase (The matrix MAT_1)

$d\_Path_1 = \{(P_3 \rightarrow P_4 \rightarrow P_1)\}$
$a\_Path_1 = \{(P_1 \rightarrow P_3 \rightarrow P_1), (P_2 \rightarrow P_3 \rightarrow P_1), (P_3 \rightarrow P_1), (P_5 \rightarrow P_6 \rightarrow P_3 \rightarrow P_1), (P_6 \rightarrow P_3 \rightarrow P_1)\}$

Fig. 3d: The d_Path_1 and a_Path_1 of processor P_1

The node-disjoint paths from P_1 to P_2	$P_1 \rightarrow P_2$	$P_1 \rightarrow P_3 \rightarrow P_2$	$P_1 \rightarrow P_4 \rightarrow P_6 \rightarrow P_2$	$P_1 \rightarrow P_5 \rightarrow P_2$
The message received	1	1	1	1
The node-disjoint paths from P_2 to P_2	$P_2 \rightarrow P_1 \rightarrow P_2$	$P_2 \rightarrow P_3 \rightarrow P_2$	$P_2 \rightarrow P_5 \rightarrow P_2$	$P_2 \rightarrow P_6 \rightarrow P_2$
The message received	1	1	1	1
The node-disjoint paths from P_3 to P_2	$P_3 \rightarrow P_2$	$P_3 \rightarrow P_1 \rightarrow P_2$	$P_3 \rightarrow P_4 \rightarrow P_5 \rightarrow P_2$	$P_3 \rightarrow P_6 \rightarrow P_2$
The message received	1	0	λ	1
The node-disjoint paths from P_4 to P_2	$P_4 \rightarrow P_1 \rightarrow P_2$	$P_4 \rightarrow P_3 \rightarrow P_2$	$P_4 \rightarrow P_5 \rightarrow P_2$	$P_4 \rightarrow P_6 \rightarrow P_2$
The message received	1	λ	1	1
The node-disjoint paths from P_5 to P_2	$P_5 \rightarrow P_2$	$P_5 \rightarrow P_1 \rightarrow P_2$	$P_5 \rightarrow P_4 \rightarrow P_3 \rightarrow P_2$	$P_5 \rightarrow P_6 \rightarrow P_2$
The message received	1	1	λ	1
The node-disjoint paths from P_6 to P_2	$P_6 \rightarrow P_2$	$P_6 \rightarrow P_3 \rightarrow P_2$	$P_6 \rightarrow P_4 \rightarrow P_1 \rightarrow P_2$	$P_6 \rightarrow P_5 \rightarrow P_2$
The message received	1	1	1	1

Fig. 3e: The message received by P_2 in the first round of message exchange phase (The matrix MAT_2)

$d\_Path_2 = \{(P_3 \rightarrow P_4 \rightarrow P_5 \rightarrow P_2), (P_4 \rightarrow P_3 \rightarrow P_2), (P_5 \rightarrow P_4 \rightarrow P_3 \rightarrow P_2)\}$
$a\_Path_2 = \{(P_3 \rightarrow P_1 \rightarrow P_2)\}$

Fig. 3f: The d_Path_2 and a_Path_2 of processor P_2

The node-disjoint paths from P_1 to P_3	$P_1 \rightarrow P_3$	$P_1 \rightarrow P_2 \rightarrow P_3$	$P_1 \rightarrow P_4 \rightarrow P_3$	$P_1 \rightarrow P_5 \rightarrow P_6 \rightarrow P_3$
The message received	0	1	λ	1
The node-disjoint paths from P_2 to P_3	$P_2 \rightarrow P_3$	$P_2 \rightarrow P_1 \rightarrow P_3$	$P_2 \rightarrow P_5 \rightarrow P_4 \rightarrow P_3$	$P_2 \rightarrow P_6 \rightarrow P_3$
The message received	1	0	λ	1
The node-disjoint paths from P_3 to P_3	$P_3 \rightarrow P_1 \rightarrow P_3$	$P_3 \rightarrow P_2 \rightarrow P_3$	$P_3 \rightarrow P_4 \rightarrow P_3$	$P_3 \rightarrow P_6 \rightarrow P_3$
The message received	m	1	λ	1
The node-disjoint paths from P_4 to P_3	$P_4 \rightarrow P_3$	$P_4 \rightarrow P_1 \rightarrow P_3$	$P_4 \rightarrow P_5 \rightarrow P_2 \rightarrow P_3$	$P_4 \rightarrow P_6 \rightarrow P_3$
The message received	λ	1	1	1
The node-disjoint paths from P_5 to P_3	$P_5 \rightarrow P_1 \rightarrow P_3$	$P_5 \rightarrow P_2 \rightarrow P_3$	$P_5 \rightarrow P_4 \rightarrow P_3$	$P_5 \rightarrow P_6 \rightarrow P_3$
The message received	1	1	λ	1
The node-disjoint paths from P_6 to P_3	$P_6 \rightarrow P_3$	$P_6 \rightarrow P_2 \rightarrow P_3$	$P_6 \rightarrow P_4 \rightarrow P_3$	$P_6 \rightarrow P_5 \rightarrow P_1 \rightarrow P_3$
The message received	1	1	λ	1

Fig. 3g: The message received by P_3 in the first round of message exchange phase (The matrix MAT_3)

d_Path_3={(P_1→P_4→ P_3), (P_2→P_5→P_4→P_3), (P_3→P_4→ P_3), (P_4→ P_3), (P_5→P_4→ P_3), P_6→(P_4→ P_3)}
a_Path_3={(P_1→ P_3), (P_2→P_1→P_3)}

Fig. 3h: The d_Path_3 and a_Path_3 of processor P_3

The node-disjoint paths from P_1 to P_4	P_1→ P_4	P_1→P_3→ P_4	P_1→P_2→P_6→ P_4	P_1→P_5→ P_4
The message received	1	λ	1	1
The node-disjoint paths from P_2 to P_4	P_2→P_1→P_4	P_2→P_3→P_4	P_2→P_5→ P_4	P_2→P_6→P_4
The message received	1	λ	1	1
The node-disjoint paths from P_3 to P_4	P_3→ P_4	P_3→P_1→ P_4	P_3→P_2→P_5→ P_4	P_3→P_6→ P_4
The message received	λ	0	1	1
The node-disjoint paths from P_4 to P_4	P_4→P_1→ P_4	P_4→P_3→ P_4	P_4→P_5→ P_4	P_4→P_6→ P_4
The message received	1	λ	1	1
The node-disjoint paths from P_5 to P_4	P_5→ P_4	P_5→P_1→ P_4	P_5→P_2→P_3→ P_4	P_5→P_6→ P_4
The message received	1	1	λ	1
The node-disjoint paths from P_6 to P_4	P_6→ P_4	P_6→P_2→P_1→ P_4	P_6→P_3→ P_4	P_6→P_5→ P_4
The message received	1	1	λ	1

Fig. 3i: The message received by P_4 in first round of the message exchange phase (The matrix MAT_4)

d_Path_4={(P_1→P_3→ P_4), (P_2→P_3→P_4), (P_3→ P_4),(P_4→P_3→ P_4), (P_5→P_2→P_3→ P_4), (P_6→P_3→ P_4)}
a_Path_4={(P_3→P_1→ P_4)}

Fig. 3j: The d_Path_4 and a_Path_4 of processor P_4

The node-disjoint paths from P_1 to P_5	P_1→ P_5	P_1→P_2→ P_5	P_1→P_3→P_6→ P_5	P_1→P_4→ P_5
The message received	1	1	0	1
The node-disjoint paths from P_2 to P_5	P_2→P_5	P_2→P_1→P_5	P_2→P_3→P_4→P_5	P_2→P_6→P_5
The message received	1	1	λ	1
The node-disjoint paths from P_3 to P_5	P_3→P_1→ P_5	P_3→P_2→ P_5	P_3→P_4→ P_5	P_3→P_6→ P_5
The message received	1	1	λ	1
The node-disjoint paths from P_4 to P_5	P_4→ P_5	P_4→P_1→ P_5	P_4→P_3→P_2→ P_5	P_4→P_6→ P_5
The message received	1	1	λ	1
The node-disjoint paths from P_5 to P_5	P_5→P_1→ P_5	P_5→P_2→ P_5	P_5→P_4→ P_5	P_5→P_6→ P_5
The message received	1	1	1	1
The node-disjoint paths from P_6 to P_5	P_6→P_5	P_6→P_2→ P_5	P_6→P_3→P_1→ P_5	P_6→P_4→ P_5
The message received	1	1	1	1

Fig. 3k: The message received by P_5 in the first round of message exchange phase (The matrix MAT_5)

d_Path_5={(P_2→P_3→P_4→P_5), (P_3→P_4→ P_5), (P_4→P_3→P_2→ P_5)}
a_Path_5={(P_1→P_3→P_6→ P_5)}

Fig. 3l: The d_Path_5 and a_Path_5 of processor P_5

The node-disjoint paths from P_1 to P_6	P_1→P_2→ P_6	P_1→P_3→ P_6	P_1→P_4→ P_6	P_1→P_5→ P_6
The message received	1	0	1	1
The node-disjoint paths from P_2 to P_6	P_2→P_6	P_2→P_1→P_4→P_6	P_2→P_3→P_6	P_2→P_5→P_6
The message received	1	1	1	1
The node-disjoint paths from P_3 to P_6	P_3→ P_6	P_3→ P_1→ P_5→ P_6	P_3→P_2→ P_6	P_3→P_4→ P_6
The message received	1	1	1	λ
The node-disjoint paths from P_4 to P_6	P_4→ P_6	P_4→P_1→P_2→ P_6	P_4→P_3→ P_6	P_4→P_5→ P_6
The message received	1	1	λ	1
The node-disjoint paths from P_5 to P_6	P_5→ P_6	P_5→P_1→P_3 → P_6	P_5→P_2→ P_6	P_5→P_4→ P_6
The message received	1	1	1	1
The node-disjoint paths from P_6 to P_6	P_6→P_2→ P_6	P_6→P_3→ P_6	P_6→P_4→ P_6	P_6→P_5→ P_6
The message received	1	1	1	1

Fig. 3m: The message received by P_6 in the first round of message exchange phase (The matrix MAT_6)

d_Path_6={(P_3→P_4→ P_6), (P_4→P_3→ P_6)}
a_Path_6={(P_1→P_3→ P_6)}

Fig. 3n: The d_Path_6 and a_Path_6 of processor P_5

D_Paths = {($P_3 \rightarrow P_4 \rightarrow P_1$),($P_3 \rightarrow P_4 \rightarrow P_5 \rightarrow P_1$), ($P_4 \rightarrow P_5 \rightarrow P_1$), ($P_5 \rightarrow P_4 \rightarrow P_5 \rightarrow P_1$),($P_1 \rightarrow P_4 \rightarrow P_1$), ($P_2 \rightarrow P_5 \rightarrow P_4 \rightarrow P_1$), ($P_3 \rightarrow P_4 \rightarrow P_1$), ($P_4 \rightarrow P_1$), ($P_5 \rightarrow P_4 \rightarrow P_1$), ($P_6 \rightarrow P_4 \rightarrow P_1$), ($P_1 \rightarrow P_3 \rightarrow P_4$), ($P_2 \rightarrow P_3 \rightarrow P_4$), ($P_3 \rightarrow P_4$), ($P_4 \rightarrow P_3 \rightarrow P_4$), ($P_5 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$), ($P_6 \rightarrow P_3 \rightarrow P_4$), ($P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5$), ($P_3 \rightarrow P_4 \rightarrow P_5$), ($P_4 \rightarrow P_3 \rightarrow P_2 \rightarrow P_5$), ($P_3 \rightarrow P_4 \rightarrow P_6$), ($P_4 \rightarrow P_3 \rightarrow P_6$)}

A_Paths = {($P_1 \rightarrow P_3 \rightarrow P_1$), ($P_2 \rightarrow P_3 \rightarrow P_1$), ($P_3 \rightarrow P_1$), ($P_5 \rightarrow P_6 \rightarrow P_3 \rightarrow P_1$), ($P_6 \rightarrow P_3 \rightarrow P_1$), ($P_3 \rightarrow P_1 \rightarrow P_2$), ($P_1 \rightarrow P_3$), ($P_2 \rightarrow P_1 \rightarrow P_3$), ($P_3 \rightarrow P_1 \rightarrow P_4$), ($P_1 \rightarrow P_3 \rightarrow P_6 \rightarrow P_5$), ($P_1 \rightarrow P_3 \rightarrow P_6$)}

Fig. 3o: The D_Paths and A_Paths of each processor

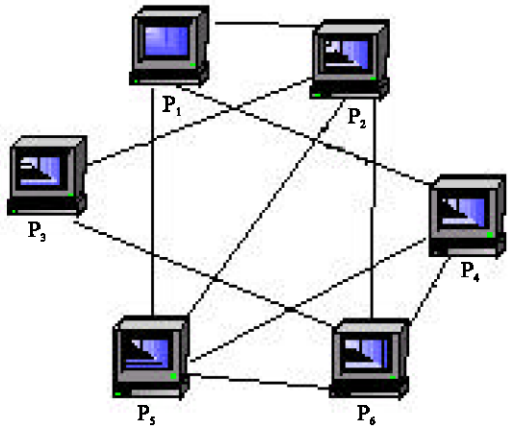


Fig. 3p: An network after reconfiguration

Fig. 3: An example of executing FDAML

message received from the path is λ, then we can ensure that there is at least one dormant faulty link in the path, we call such path as the dormant faulty path. In DFAML, D_Paths is used to store the dormant faulty path. Therefore, we can examine the D_Paths to find out the paths, which only contain one link. In case of only one link in the path, then the link is a dormant faulty link.

Lemma 2: If the path between the sender processor P_i and destination processor P_j is fault-free, then the received messages should be the same as the initial value of each processor by GPBA.

Proof: If the path between sender processor P_i and destination processor P_j is fault-free, then the message sent though the path is transmitted correctly. That is, if the sender processor P_i sends its initial value v_i to the destination processor P_j, then the destination processor should receive the value v_i correctly. According to the initial value of each processor is the same, so the received value v_i should be the same as the processor P_j's initial value.

Lemma 3: Each processor can detect the arbitrary faulty path in the general network.

Proof: If the message received from the sender processor P_i is not the initial value of each processor by GPBA and also not the value λ, then the path between the sender processor P_i and destination processor P_j is in arbitrary.

Lemma 4: There are at most $\lfloor (c\text{-}.D\_Links.\text{-}1)/2 \rfloor$ arbitrary faulty links in the general network if $c>2a+d$.

Proof: By Theorem 1, the dormant faulty link can be detected and according to the constraint $c>2a+d$, we can ensure that there are at most $\lfloor (c\text{-}.D\_Links.\text{-}1)/2 \rfloor$ arbitrary faulty links in the network, where, d =.D_Links..

Theorem 2: Each processor can detect/locate the arbitrary faulty links in the general network.

Proof: By Lemma 3, each processor can detect the arbitrary faulty paths in the general network. In DFAML, A_Paths is used to store the arbitrary faulty paths. According to Lemma 4, there are at most $\lfloor (c\text{-}.D\_Links.\text{-}1)/2 \rfloor$ arbitrary faulty links in the network if $c>2a+d$. So that, examines the A_Paths to find out the arbitrary links which appear times is the top $\lfloor (c\text{-}.D\_Links.\text{-}1)/2 \rfloor$ at the A_Paths.

Theorem 3: The FDA protocol DFAML can detect/locate the faulty links in a general network.

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

CONCLUSION

By using the BA protocol GPBA[8] and FDA protocol FDAMIX[16] by Siu *et al.*[8] can solve the BA problem on processors and communication links with mixed faults and solve the FDA problem on processors with mixed faults. Then using our FDA protocol FDAML can solve the FDA problem on communication links with mixed faults. That is, after executing the protocol GPBA, FDAMIX and FDAML can reconfigure the unreliable general network to be a stable general network without faulty components. In addition, the proposed FDAML can detect/locate the maximum number of faulty communication links with mixed faults in a general network.

REFERENCES

1. Silberschatz, A., P. Galvin and G. Gagne, 2000. Applied Operating Sys. Concepts, John Wiley and Sons, Inc., Ch. 14, pp: 469-480.
2. Silberschatz, A., P. Galvin and G. Gagne, 2002. Operating Systems. Concepts, 6th. Edn., John Wiley and Sons, Inc.
3. Dolev, D., 1982. The byzantine generals strike again. J. Algorithms, 1: 14-30.
4. Lamport, L., R. Shostak and M. Pease, 1982. The byzantine generals problem. ACM Transactions on Programming Languages Sys., 4: 382-401.
5. Pease, M., R. Shostak and L. Lamport, 1980. Reaching agreement in the presence of faults. J. ACM., 2: 228-234.
6. Wang, S.C., Y.H. Chin and K.Q. Yan, 1995. Byzantine agreement in a generalized connected network. IEEE Trans. Parallel Distributed Sys., 6: 420-427.
7. Yan, K.Q., Y.H. Chin and S.C. Wang, 1992. Optimal Agreement Protocol in Malicious Faulty Processors and Faulty Links. IEEE Transactions on Data and Knowledge Eng., pp: 266-280.
8. Siu, H.S., Y.H. Chin and W.P. Yang, 1998. Byzantine agreement in the presence of mixed faults on processors and links. IEEE Trans. Parallel Distributed Sys., 9: 980-986.
9. Meyer, F.J. and D.K. Pradhan , 1991. Consensus with Dual Failure Modes, IEEE Trans. Parallel Distributed Sys., 2: 214-222.
10. Siu, H.S., Y.H. Chin and W.P. Yang, 1998. Reaching strong consensus in the presence of mixed failure types. J. Inform. Sci., 108: 157-180.
11. Siu, H.S., Y.H. Chin and W.P. Yang, 1996. A Note on Consensus on Dual Failure Modes, IEEE Trans. Parallel Distributed Sys., 7: 225-230.
12. Hsiao, H.S., Y.H. Chin and W.P. Yang, 2000. Reaching fault diagnosis agreement under a hybrid fault model. IEEE Trans. Computers, 9: 980-986.
13. Ramarao, K.V.S. and J.C. Adams 1988. On the diagnosis of byzantine faults. Proc. Symp. Reliable Distributed Sys., pp: 144-153.
14. Wang, S.C., Y.H. Chin and K.Q. Yan, 1990. Reaching a fault detection agreement. Proc. Intl. Conf. Parallel Processing, pp: 251-258.
15. Fischer, M., M. Paterson and N. Lynch, 1985. Impossibility of distributed consensus with one faulty process. J. ACM., 32: 374-382.
16. Halsall, F., 1995. Data Links, Computer Networks and Open Sys., 4th Edn., Addison-Wesley Publishers Ltd., Ch. 3, pp: 112-125.
17. Deo, N., 1974. Graph Theory with Applications to Engineering and Computer Sci., Englewood Cliffs, N. J. Prentice-Hall.