ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2012, Vol.41, No.2

A New Approach to the Fault Detection Problem for Mobile P2P Network

Mao-Lun Chiang[1], Hui-Ching Hsieh[2*]

[1]*Department of Information and Communication Engineering, Chaoyang University of Technology 168, Jifeng E. Rd., Wufeng District, Taichung County, 41349 Taiwan, R.O.C*
e-mail: mlchiang@cyut.edu.tw

[2]*Department of Information Communication, Hsing Wu Institute of Technology, no. 101, Sec.1, Fenliao Rd., LinKou District, New Taipei City 244, Taiwan, R.O.C.*
e-mail: luckyeva.hsieh@gmail.com

crossref http://dx.doi.org/10.5755/j01.itc.41.2.854

Abstract. To improve the performance of mobile P2P network systems, all the faulty-free peers must be able to function collaboratively. Regrettably, some peers may be untrustworthy and unwilling to cooperate with others. Some peers may even attack the network resulting in the performance degrades. For this reason, it is very important to provide a reliable protocol to detect and remove faulty peers. In the past, there have some traditional BA protocols been proposed for fault detection, in which all peers require to exchange $2 * (\lfloor (n-1)/3 \rfloor + 1)$ rounds of message to collect messages; and the complexity of messages is $O(n^{\lfloor (n-1)/3 \rfloor * \lfloor (n-1)/3 \rfloor})$. However, the previous protocols are inefficient and unsuitable for the mobile P2P network because most of the protocols do not concern the mobility issue, and can cause large number of message results in a large protocol overhead. In this study, we proposed a new fault detection protocol to detect/locate faulty peers by using only three rounds of message exchange. Furthermore, the complexity of protocol we proposed can be reduced to $O(n^2)$ even if some peers move around the network. Since, our proposed protocol is more suitable and efficient for mobile P2P network.

Keywords: Peer-to-Peer; P2P; Fault Diagnosis; Byzantine Agreement; Fault-tolerance.

1. Introduction

Over the past decade, the world has witnessed an explosion in the development and deployment of new network technologies, in which, mobile Peer-to-Peer (P2P) networks have attracted much attention and have been widely deployed on the Internet for various purposes, such as file sharing networks, collaborative computing and distributed data storages etc. Fundamentally, the success of mobile P2P network systems relies on the cooperation among the peers. This means that some mobile P2P systems can serve thousands of peers with acceptable quality of service [21] with a prerequisite that all peers cooperate closely in the mobile P2P network.

Unfortunately, some peers may be unwilling to share resources with other peers in the mobile P2P network. Besides, some peers may alter messages, send corrupt data, and disseminate viruses to attack the mobile P2P network. As a result, the availability of resources will be decreased in the mobile P2P network. For improving these problems, fault diagnosis of mobile P2P network for assuring that all peers can collaborate with each other becomes an important issue. In previous studies, most researches are focused on peer selection for requesting files [11], scheduling files for distributed system [22] and clustering peers in P2P networks [2], [13] based on the assumption that all peers are truthfully cooperative. However, it is not reasonable to assume each peer can cooperate with others well.

According to the reason above, some approaches have been proposed to find out the uncooperative peer. Jun, Ahamad and Xu [19] designed a protocol to compute a trust level for each peer according to their behavior. Subsequently, the multicast tree can be constructed by these trustful peers. Besides, Aberer and Despotovic [7] provided a method to manage peer reputation by using a DHT-like distributed information access system. However, these protocols do not consider the influence of faulty peer. Therefore, Kamvar et al. [18] proposed a protocol to make each peer to evaluate other peers' reputations repetitively according to reports from third-party peers. However, the evaluation procedure needs to take a long time to converge the final values of reputation.

Similarly, Jin et al. [20] also proposed a history-based method for detecting faulty peers. In their research, the server can decide which peers are faulty by continuously analyzing the monitoring reports

* Corresponding author

received from all peers. Essentially, this method will increase the overhead of server and it is inefficient to decide which peers are faulty according to their history reports. Besides, peers may have different behaviors at different time. In other words, peers make work incorrectly at this second, and work correctly at next moment. If these peers are determined to be faulty and then be removed, the available mobile P2P resource will be removed at the same time. This will decrease the available resources and the performance for the network. Since, these protocols still have rooms for improvement.

Besides, there also have some fault detection protocols [5], [6], [9], [17], [23] based on the Byzantine Agreement (BA) protocol [3], [8], [10], [12], [14], [15] been proposed in a distributed system (Basically, the mobile P2P network is one kind of a distributed system. Peers in the P2P network can be seen as processors in a distributed system.). Basically, these protocols can detect the faulty processors based on the comparison among the messages which are received from other processors. Through this scheme, the detection results are more objective and correct.

Basically, the BA problem was first studied by Lamport et al. [10], and was defined as follows:

1. There are n processors in the network system, of which $\lfloor (n-1)/3 \rfloor$ processors could fail.
2. The processor communicates with each other through message exchange and the network model is a fully connected network.
3. The messages sender is always identifiable by the receiver.
4. One of the processors is chosen as the launch processor and its initial message is broadcasted to others and to itself.

Based on these assumptions, all fault-free processors can agree on a common message. In addition, the protocol for solving the BA problem must meet the following requirements [3], [4], [8], [10], [12], [14], [15], [16]:

(Agreement): All non-faulty processors agree on a common message.

(Validity): If the source processor is fault-free and its initial message is v, then all fault-free processors must agree on the message v.

A closely related and important issue, Fault Diagnosis Agreement (FDA) [5], [17], [23], [24], [25] is also in need of review. In general, FDA can be divided into two categories: test-based approaches [24], [25] and evidence-based approaches [5], [17], [23].

In a test-based approach, a processor P_a can test the condition of a processor P_b unaided. However, this is impracticable, particularly in light of malicious faulty processors. The symptoms of malicious faulty processors are usually unrestrained (have unrestrained behaviors); and can hide their faulty behavior, allowing them to avoid detection. Thus, test-based approaches are not suitable for arbitrary faults.

Another kind of FDA, the evidence-based approach, primarily collects messages that have accumulated in BA protocols as evidence to detect/locate faulty processors. Since each processor can obtain the evidences about the transmission behaviors of the faulty processors from other processors, these evidences can be used to detect/locate faulty processor. It is more accurate and objective.

Upon achieving FDA, the performance and integrity of a distributed network can be guaranteed. A protocol designed for evidence-based FDA must satisfy the following conditions [5], [17], [23]:

(Agreement): All non-faulty peers identify the common set of faulty peers.

(Fairness): No non-faulty peer is incorrectly detected as faulty by any non-faulty peer.

In the past, there have some evidence-based FDA protocols been proposed [5], [17], [23]. These protocols can make each processor agree on a common set of faulty processors. The amount of the faulty processors that can be detected out is maximal. Unfortunately, all processors need to run $2*(\lfloor (n-1)/3 \rfloor+1)$ rounds (The term "round" presents the interval of message exchange between any pair of peers [10]) of message exchange, and the message complexity of previous protocols [5], [9], [23] is $O(n^{\lfloor (n-1)/3 \rfloor)*\lfloor (n-1)/3 \rfloor})$. This is not suitable for mobile P2P network, because the P2P network may exist millions of peers and will generate a large number of transmission overhead. Furthermore, each peer can move around different mobile P2P networks at any time. The previous protocols [5], [9], [23] can only find out the faulty peers under a static or well-defined network environment, such as fully connected network, broadcast network and so on.

Therefore, the Fault Detection Protocol for mobile P2P networks (FDP2P) is proposed to determine whether peers are faulty. The FDP2P protocol can detect/locate the maximum number of faulty peers by using three rounds of message exchange. It is far superior to previous works [5], [9], [23]. Besides, the message complexity of FDP2P is $O(n^2)$. It is less than previous works [5], [9], [23] (The message complexity of previous works is $O(n^{\lfloor (n-1)/3 \rfloor)*\lfloor (n-1)/3 \rfloor})$). Furthermore, we also consider the mobility issue into FDP2P. To sum up, FDP2P can find out the faulty peers with constant rounds of message exchange even when there have some peers moving around the network. Thus the FDP2P protocol has higher performance than previous works [5], [9], [17], [23].

The remainder of this study is organized as follows. Section 2 introduces the assumptions and concepts of the proposed protocol. Section 3 presents the detail of the proposed protocol. An example is given in Section 4. Section 5 presents the correctness and complexity. Finally, Section 6 provides the conclusion.

2. The assumptions and concepts of the proposed protocol: FDP2P

For improving the performance of network, peers which have attack actions or abnormal behaviors must be detected and removed in our protocol FDP2P. The assumptions and parameters of protocol FDP2P are listed as follows:

1. Mobile P2P network is one kind of distributed systems where peers can be served as processors in a distribution system.
2. Each peer in the mobile P2P network is unique.
3. Let N be the set of all peers in the mobile P2P network and | N | = n.
4. Whether there have peers immigrating into or emigrating from the network, the protocol is valid only when the total number of faulty peers in the mobile P2P network is less than or equal to $\lfloor (n-1)/3 \rfloor$.
5. Each peer knows the total number of peers in a mobile network at any time.
6. If the peers emigrate from the network and then immigrate into the same network later, FDP2P will treat these peers as new participators.
7. Peers can move around the mobile P2P network arbitrarily. Here, we assume that peers can only immigrate into or emigrate from the network during the period of message exchange, but not in the period of deciding whether peers are fault-free or not. It is because that all peers will not exchange messages anymore. If peers can move around the network during the period of deciding which peers are faulty, peers will not have enough messages to find out the faulty peers accurately. Hence, this assumption is helpful for FDP2P to find out the faulty peers accurately.

For clarity of the proposed protocol FDP2P, the following relevant information must be defined first.

2.1. The ms-tree

During the execution of FDP2P, the received messages are stored in a convenient tree structure called the message storage tree (ms-tree), which is similar to what Bar-Noy et al. proposed in [1]. Each faulty-free peer maintains and stores the received messages in its ms-tree. An example of a five-peer network is shown in Fig. 1. In the first round of message exchange, the source peer *s* broadcasts its initial value v_s to the others and itself. Due to the message sender can be identified, each faulty-free peer can denote the received message as $v(s)$ from the source and store in the root of its ms-tree. However, each peer cannot identify whether the source is a faulty-free peer, thus each faulty-free peer requires more rounds to remove the faulty influence generated by faulty source peers.

In the second round of message exchange, each peer broadcasts the root's value of its ms-tree to all peers. Subsequently, each peer receives the values from other peers and stores the received values into the second level of its ms-tree. Furthermore, the process of the third rounds is the same as the second round, and the received values will be stored into the third level of its ms-tree.

Here, the vertices with repeated names of peers (*ss*, *saa*, *sbb*, *scc*, and *sdd*) must be removed from the ms-tree to remove the cyclical influences from the faulty peers. To put it plainly, when the faulty-free peers store the messages which are received from the faulty peers in the ms-tree repeatedly, the faulty-free peers may get an incorrect common value after taking a simple majority. As a result, all these repeated names of peers must be deleted.

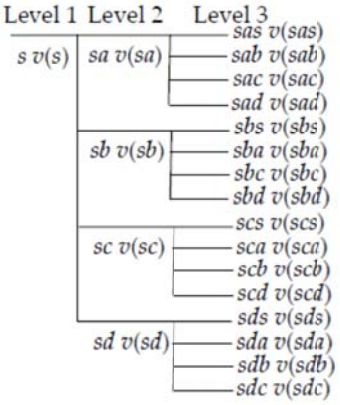

Figure 1. A peer's ms-tree

2.2. Fault detection processes

In this study, we propose the new fault detection processes by comparing the messages of the ms-tree. The peers can be checked with the following two conditions to determine whether the peers are faulty peers.

1. The majority value of the vertices in the third level of the ms-tree (denoted as maj3(ancestorax), $1 \le x \le n$) is equal to their ancestor v(ax) in the second level of the ms-tree. Take Fig. 1 for example, v(sas), v(sab), v(sac) and v(sad) are siblings, and their ancestor is v(sa). Here, FDP2P must check whether the majority of v(sas), v(sab), v(sac) and v(sad) are equal to their ancestor v(sa) or not.
2. There are at least $(n-\lfloor (n-1)/3 \rfloor)$ values derived from the condition 1 that are the same as the majority value maj3(ancestorax).

If the conditions above are met, then the peer *x* can be added into the Non-Faulty-Like Peer set (The set is belong to $v(ax)$ (denoted as NFLP(ancestor_{ax}))). Furthermore, the peer *y* is also added into NFLP (ancestor_{ax}) when $v(axy)$ $(1 \le y \le n) = v(ax)$. Subsequently, all peers must count the frequency of each peer that appears in the NFLP. If the frequency of the peer is less than $(n-\lfloor (n-1)/3 \rfloor)$, this peer is defined as a faulty peer. In other words, the values, which are

sent by the faulty peers, will not be agreed by (n-⌊(n-1)/3⌋) number of peers. Deservedly, the frequency of these peers in NFLP will be less than (n-⌊(n-1)/3⌋). Hence, these peers will be defined as faulty peers.

2.3. Whether there have peers immigrate into or emigrate from the network, the protocol is valid only when the total number of faulty peers in the mobile P2P network is less than or equal to ⌊(n-1)/3⌋

According to the previous fault detection protocols [5], [9], [17], [23], the tolerable number of faulty processors is ⌊(n-1)/3⌋. When the total number of faulty processors exceeds the limit, the faulty-free processors cannot decide which processors are faulty. This means that when the total number of faulty peers exceeds this bound, then there may have some fault-free peers been defined as faulty peers. Furthermore, there may also have the condition that the peers cannot be decided as faulty or fault-free peers. In other words, FDP2P is valid only when the total number of faulty peers is less than or equal to ⌊(n-1)/3⌋, even when there have peers moving around the mobile P2P network.

3. The proposed protocol: FDP2P

In this section, the FDP2P protocol is invoked to detect/locate common set of faulty peers in an n-peers mobile P2P network within three rounds of message exchange. There are three phases in FDP2P, the *message exchange phase*, the *fault detection phase* and the *reorganize phase*. Each each faulty-free peer executes the same protocol FDP2P simultaneously.

Basically, the procedure of FDP2P is stated in below. Moreover, the details of protocol FDP2P are described as Fig. 2.

Protocol FDP2P

Notation:

- $maj_3(\text{ancestor}_{ax})$: The majority value of the vertices (Which are descended from the ancestor $v(ax)$) in the third level of the ms-tree.
- $\#\ maj_3(\text{ancestor}_{ax})$: The total number of the values (within the vertices which are descended from the ancestor $v(ax)$) that are equal to $maj_3(\text{ancestor}_{ax})$.
- NFLP(ancestor_{ax}): The Non Faulty-Link Peer set for the sub-tree which is descended from $v(ax)$.
- $\#p_z$: The total number of times that the peer z appears in all NFLPs.
- N: The set of peers in the mobile P2P network.
- S_{faulty}: The set of faulty peers in the mobile P2P network.

Message exchange phase:

r = 1 do:

- The source broadcasts its initial value v_s to other peers and itself.
- Each peer stores v_s in the root of its ms-tree;

For r = 2 to 3 do:

- If a new peer immigrates into the network, then {Execute the function processing-immigrate};
- If a peer emigrates from the network, then {Execute the function processing-emigrate};
- Each peer broadcasts the value at the (r-1)th level of its ms-tree to other peers and itself.
- Each peer stores the received values at the rth level of its ms-tree.

Fault Detection phase:

For each sub-tree of vertex $v(ax)$ in the second level of ms-tree. {

If $v(ax) = maj_3(\text{ancestor}_{ax})$and $\# maj_3(\text{ancestor}_{ax}) \geq (n\text{-}\lfloor(n\text{-}1)/3\rfloor\text{-}1)$ then {

Add peer x into NFLP(ancestor_{ax}).

For each vertex whose parent is vertex $v(ax)$ {

If $v(axy) = v(ax)$ then{

Add peer y to NFLP(ancestor_{ax}) }}}}

For each peer z (denoted as p_z) {

Count $\#p_z$ from all NFLPs

If $\#p_z < (n\text{-}\lfloor(n\text{-}1)/3\rfloor)$ then {

Peer z is a faulty peer and will be added into the faulty peers set S_{faulty} }}

Reorganization phase

Set $N = N\text{-}\ S_{faulty}$

Function processing-immigrate:

r = 2 do:

- The source broadcasts its initial value v_s to the new peers.
- The new peers store the received value to the root of its ms-tree.

r = 3 do:

- The source peer sends its initial value to the new peers.
- The new peers store the received value to the root of its ms-tree.
- Each peer in the original network sends its value received in the second round to the new peers.
- The new peer takes the majority values on the values received from other peers
- The new peer stores the majority values at the r-1 level of its ms-tree.
- The new peers send the majority values to other peers.

Function processing-emigrate:

- Delete the values received from the left peer and reconstruct the ms-tree.

Figure 2. The protocol FDP2P

Message Exchange Phase:

In this phase, three rounds of message exchange are collected and stored into each peer's ms-tree. Furthermore, all peers can move around the network during *message exchange phase*. In order to construct a correct ms-tree, there have two cases (peer immigrate

into the network and peer emigrate from the network) must be considered.

Case 1: New peer immigrates into the network:

If a new peer immigrates into the network before the second round of the message exchange, the source peer must send its initial value to this new peer. The new peer must store the received value into the root of their ms-tree.

If a new peer immigrates into the network before the third round of message exchange, the source peers must send its initial value to the new peers. Besides, the other peers who are in the network originally must send their values in the second level of their ms-tree to this new peer. After that, this new peer must take the majority values on the received values, and then store the majority values into the root and second level of their ms-tree. The new peer must also send the values in the root and the second level of the ms-tree to others.

Case 2: Peer emigrates from the network:

If a peer emigrates from the network, the remaining peers only need to delete the message received from the left peers.

Fault Detection Phase:

In the *fault detection phase*, each peer must decide which peers are faulty by comparing the messages sent from all peers by using the conditions described in Section 2.

Reorganization Phase:

After finding out the faulty peers, all faulty-free peers can remove the faulty peers and reorganize the network topology. Since there have no faulty peers in the network, all the fault-free peers can get correct resources with better performance. In other words, the performance of network and correctness can be enhanced without the influence caused by the faulty peers at the same time.

To sum up, previous works [5], [9], [17], [23] require $2*(\lfloor(n\text{-}1)/3\rfloor+1)$ rounds of message exchange, however, in FDP2P, all peers only need to run three rounds of message exchange. The message complexity of FDP2P is $O(n^2)$ and is more efficient even if the number of peers is getting larger. Furthermore, the transmission time can be reduced when the rounds of message exchange are decreased. For example, there is a 1000-peers mobile P2P network, and each peer has 2Mb download rate per second. The time complexity for transmitting the messages in previous protocols is $O((1000^{333*333})/2Mb)$ seconds. In opposition to the previous protocols, the time complexity of the FDP2P is $O(1000^2/2Mb)$ seconds. FDP2P also considers the mobility issue of peers. Namely, FDP2P can detect/locate the faulty peers even when there have some peers immigrating or emigrating from the network. Hence, FDP2P is more efficient than previous protocols. The comparison results are shown in Table 1.

Table 1. Comparison between FDP2P and previous protocols

	Needed rounds	Message complexity	Transmission time complexity (2Mb/s)	Allowable faulty peers
Previous protocol [5], [9], [17], [23]	$2*(\lfloor(n\text{-}1)/3\rfloor+1)$	$O(n^{\lfloor(n\text{-}1)/3\rfloor*\lfloor(n\text{-}1)/3\rfloor})$	$O((n^{\lfloor(n\text{-}1)/3\rfloor*\lfloor(n\text{-}1)/3\rfloor})/2Mb)$	$\lfloor(n\text{-}1)/3\rfloor$
FDP2P	3	$O(n^2)$	$O(n^2/2Mb)$	$\lfloor(n\text{-}1)/3\rfloor$

4. An example of implementing FDP2P

In this section, an example including ten peers is shown in Fig. 3 to illustrate how to detect and locate the faulty peers by three rounds of message exchange with FDP2P. In this example, we assume the peer *a* as the faulty source peer, which sends different values to all peers. Besides, peers *b* and *c* are also assumed to be faulty peers in Fig. 3(a). For checking the validity of FDP2P, a worst case scenario (the number of 0 and 1 are almost the same) is designed and the transmission behavior of the faulty peers is shown in Fig. 3(b).

At the beginning of the protocol, the source peer *a* broadcasts its initial value to all pees in the first round of the message exchange phase. Unfortunately, the source peer *a* is a faulty peer; it sends different values 0, 1, 0, 1, 0, 1, 0 to peers *d*, *e*, *f*, *g*, *h*, *i*, and *j*. Here, each faulty-free peer stores the received value in the root of its ms-tree in the first round, and the results are shown in Fig. 3(c). However, the results of faulty peers do not need to be discussed because the goal of the protocol is to make all faulty-free peers to obtain a common set of faulty peers. Hence, this example only shows the results of faulty-free peers.

Before the start of the second round of message exchange phase, peer *k* wants to immigrate into the P2P network. The source peer *a* must send its initial value to peer *k* first. After receiving the value 1 from peer *a*, peer *k* must store the value into the root of its ms-tree, and then continue to execute the protocol. The result is shown in Fig. 3(d).

In the second round of the message exchange phase, each peer exchanges the received value from first round of message exchange phase with all peers. Similarly, the received messages are stored in second level of their ms-tree. The results are shown in Fig. 3(e).

Now, we assume that peer *f* wants to emigrate from the mobile P2P network. Here, the ms-tree of peer *f* will be deleted automatically. Similarly, all peers need to delete the value v(*af*) received from peer *f*, and the results are shown in Fig. 3(f).

In third round of the message exchange phase, each peer exchanges the received values from the second round of the message exchange phase with all peers and stores the received values in the third level of their ms-tree. In this example, the result of peer d's ms-trees is shown in Fig. 3(g).

After the message exchange phase, each peer can determine which peers are faulty in the fault detection phase. For example, the peers *a*, *c*, *d*, *e*, *g*, *h*, *i*, *j*, and *k* can be added to the NFLP(ancestor$_{ad}$) when the following conditions are satisfied:

1. $v(ad) = maj_3(\text{ancestor}_{ad}) = 0$,
2. $\#\ maj_3(\text{ancestor}_{ad}) = 8 \geq (n-\lfloor(n-1)/3\rfloor - 1) = 6$, and
3. $v(adx) = maj_3(\text{ancestor}_{ad})$ {such as, $v(ada)$, $v(adc)$, $v(ade)$, $v(adg)$, $v(adh)$, $v(adi)$, $v(adj)$, and $v(adj) = maj_3\ (\text{ancestor}_{ad}) = 0$}.

Then, the protocol will count the frequency ($\#p_z$) of each peer that appears in all of NFLPs and then compute whether $\#p_z$ is less than ($n-\lfloor(n-1)/3\rfloor$). In this example, peers *a*, *b* and *c* are faulty peers (the appearance frequency of peers *a*, *b*, and *c* is 4 times, 3 times, and 4 times, respectively. The appearance frequency is less than $n-\lfloor(n-1)/3\rfloor = 7$). The procedure for peer *d* is shown in Fig. 3(h).

After detecting the faulty peers, all the faulty-free peers can remove the faulty peers in the Reorganization phase. At the same time, the data will not be altered, and the performance will not be degraded by faulty peers. All faulty-free peers then can do some activities more efficiently and correctly. In the original network environment shown in Fig. 3(a), peers *d*, *e*, *g*, *h*, *i*, *j*, and *k* are faulty-free peers, and peers *a*, *b* and *c* are faulty peers. After running FDP2P, all the faulty-free peers can remove all the faulty peers. The new network structure is shown in Fig. 3(i).

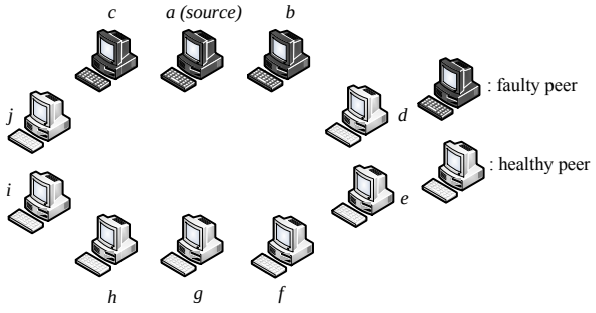

Figure 3. (a) The environment of the original P2P network.

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

Figure 3. (b) The transmission behavior of faulty peers.

d	*v(a)=0*
e	*v(a)=1*
f	*v(a)=0*
g	*v(a)=1*
h	*v(a)=0*
i	*v(a)=1*
j	*v(a)=0*

Figure 3. (c) The results of the first round of message exchange phase.

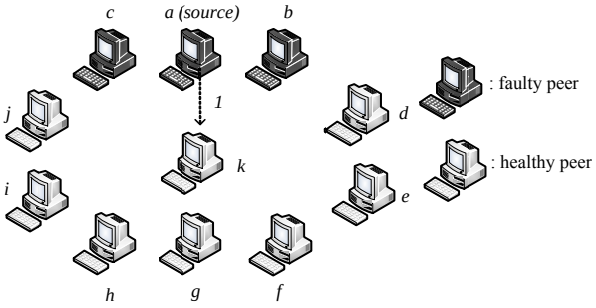

Figure 3. (d) Peer *k* stores the received value received from peer *a* into the root of its ms-tree at the start of the second round of message exchange.

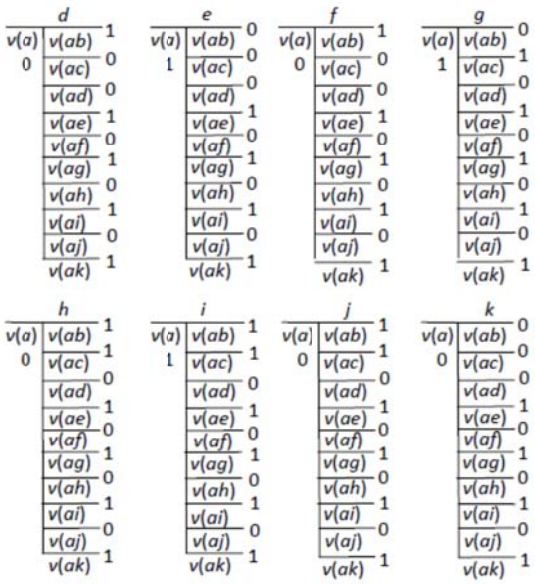

Figure 3. (e) The results of the second round of message exchange phase.

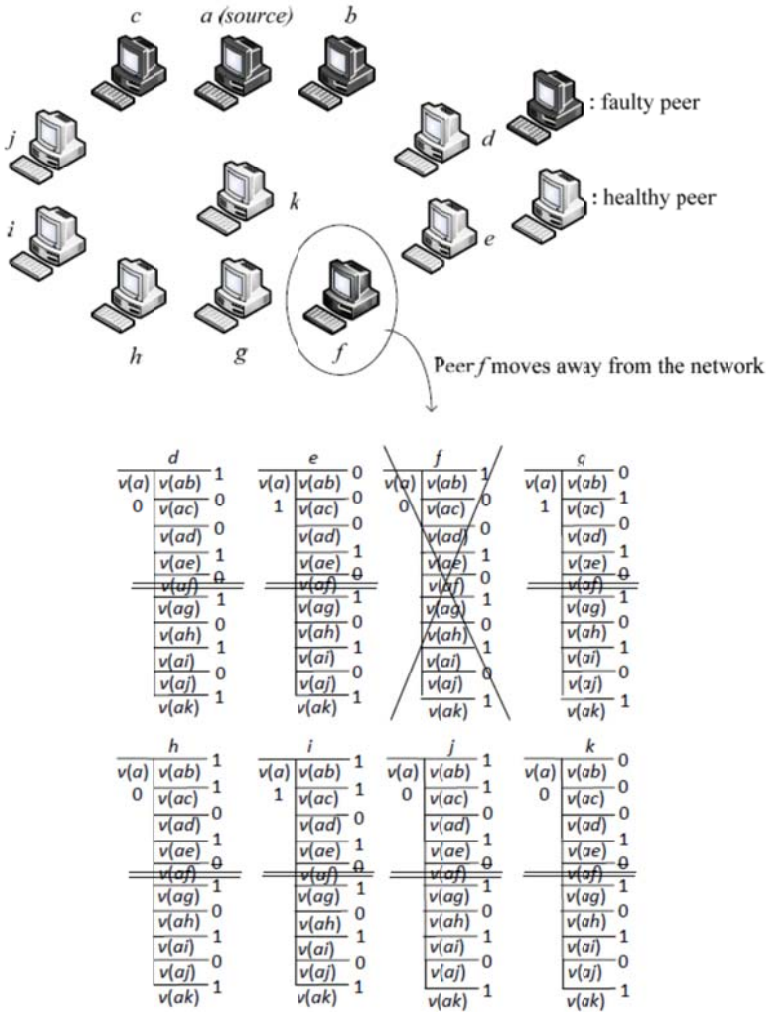

Figure 3. (f) Each peer deletes the value received from peer *f*

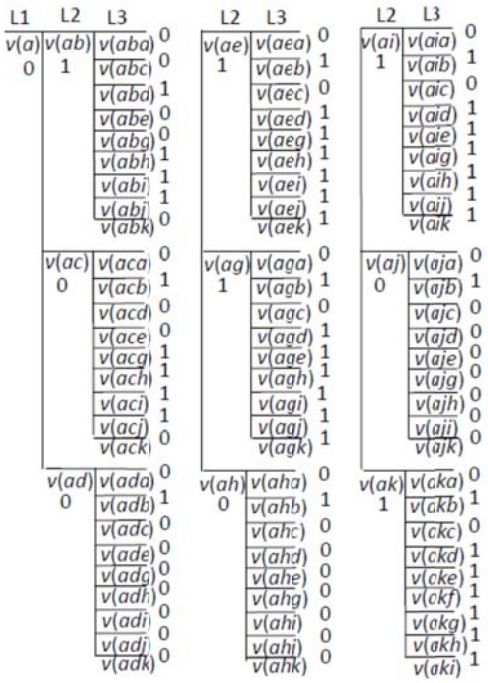

Figure 3. (g) The results of the third round of message exchange phase for peer *d*.

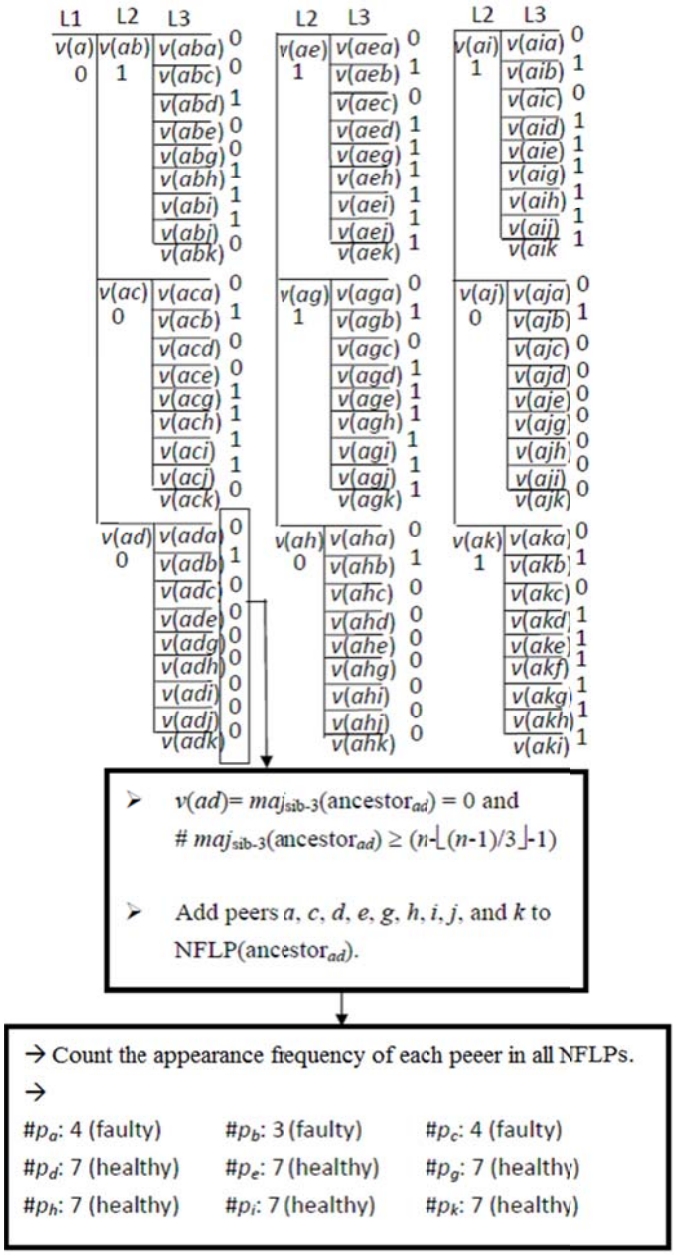

Figure 3. (h) The procedure of finding out the faulty peers.

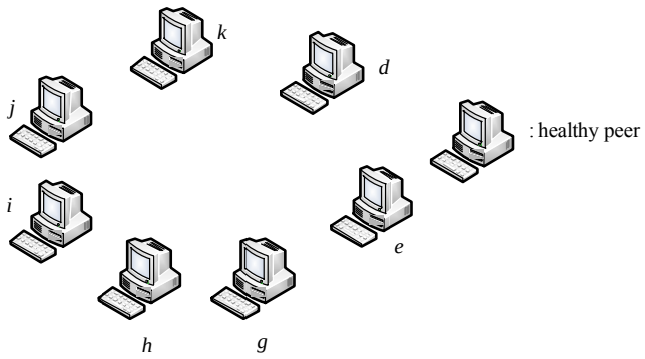

Figure 3. (i) New network structure.

Figure 3. An example of executing FDP2P

5. The correctness and complexity

In the first subsection, we will prove the correctness of the proposed protocol FDP2P. The complexity is proven in Section 5.2.

5.1. The correctness of FDP2P

Here, the correctness of FDP2P can be proved by considering the following requirements:

(Agreement'): Root s is common.

(Fairness'): No faulty-free peer is incorrectly detected as faulty by any non-faulty peer.

Before analyzing the protocol, several terms must be defined.

A vertex α is called common [1] if each faulty-free peer computes the same value for α. In other words, the values stored in vertex α of each faulty-free peer's ms-tree are identical to all peers, if the values are sent from the faulty-free peers.

To prove that the vertex is *common*, the term *common frontier* [1] is defined as: when every root-to-leaf path of the ms-tree contains a common vertex, then the collection of the common vertices forms a common frontier. In other words, every faulty-free peer has the same messages collected in the common frontier if a common frontier exists in a faulty-free peer's ms-tree. The above concepts can be used to prove the correctness of the proposed protocol FDP2P.

The term *correct vertex* is defined as follows: vertex αi of a tree is a correct vertex if peer i is faulty-free. For instance, vertices $v(ad)$, …, $v(ak)$ in Fig. 3(e) are correct because peers d, …, and k are faulty-free. The following lemmas, corollary and theorem are used to prove the correctness of FDP2P.

Lemma 1: All correct vertices of the ms-tree are common.

▼**Proof.** In the *Fault Detection phase*, there are no repeatable vertices in the ms-tree obtained by deleting the repeating vertices. At the second and third levels, the correct vertex α has at least n-1 children in which at least n-$\lfloor(n\text{-}1)/3\rfloor$ children are correct. The values, which are stored within these n-$\lfloor(n\text{-}1)/3\rfloor$ correct vertices, will be the same. The correct vertex α is common in the ms-tree, if the level of α is less than or equal to three. Thus, all correct vertices of the ms-tree are common. ▲

Lemma 2: A common frontier does exist in the ms-tree.

▼**Proof.** There are three vertices along each root-to-leaf path of the ms-tree at any time, in which the root is labeled by the source name, and the others are labeled by a sequence of peers' names. Since at most $\lfloor(n\text{-}1)/3\rfloor$ peers can fail, at least one vertex is correct along each root-to-leaf path of the ms-tree. By lemma 1, the correct vertex is common, and a common frontier exists in each faulty-free peer's ms-tree. ▲

Lemma 3: Let α be a vertex. If there is a common frontier in the sub-tree rooted at α, then α is common.

▼**Proof.** By induction hypothesis on the height of α.

If the height of α is 0 and a common frontier (α itself) exists, α is common.

If the height of α is l, the children of α are all in common, based on the induction hypothesis with the height of the children as l-1; therefore, vertex α is common. ▲

Corollary 1: *If a common frontier exists in the ms-trees, the root is common.*

Theorem 1: *The root of a faulty-free peer's ms-tree is common.*

▼**Proof**. By Lemmas 1 to 3, and Corollary 1, the theorem is proved**.** ▲

Theorem 2: *FDP2P can solve the fault detection problem.*

▼**Proof.** To prove this theorem, FDP2P must meet the requirements: Agreement and Validity.

Agreement: Root *s* is common.

By Theorem 1, Agreement is satisfied.

Validity: No faulty-free peer is incorrectly detected as faulty by any non-faulty peer.

If the peer is faulty-free, it will send identical value to other peers. Since there have at least n-$\lfloor(n\text{-}1)/3\rfloor$ faulty-free peers, the majority value of the faulty-free peers will be equal to the value sent by the faulty-free sender. In other words, there will have at least n-$\lfloor(n\text{-}1)/3\rfloor$ faulty-free peers agree that the sender is faulty-free. Hence, no faulty-free peers will be treated as faulty peers. The theorem is proved. ▲

5.2. The complexity of FDP2P

The complexity of the protocol is evaluated in terms of 1) the number of rounds about message exchange, 2) the number of allowable faulty peers and 3) the message complexity of the FDP2P protocol.

Theorem 3: *FDP2P requires only three rounds of message exchanges to detect/locate the faulty peers in the mobile P2P network.*

▼**Proof.** In the second round of the message exchange phase, the values are sent and received correctly by other n-$\lfloor(n\text{-}1)/3\rfloor$ faulty-free peers if the sending peers are faulty-free. All these correct values are sent in the third round of the message exchange phase. Then, these three level ms-trees can be used to determine the faulty peers. Here, they have (n-1) vertices in the second level of the ms-tree. Thus, there will be (n-1) $NFLP_x$ ($1 \le x \le (n\text{-}1)$). If the frequency of the peer x appearing in all NFLPs is less than n-$\lfloor(n\text{-}1)/3\rfloor$, there has less than n-$\lfloor(n\text{-}1)/3\rfloor$ peers believe that peer x is faulty-free. Furthermore, based on Theorem 2, the protocol can solve the fault

detection problem. Hence, three rounds of message exchange can solve the fault detection problem in the mobile P2P network. ▲

Theorem 4: *FDP2P can solve the fault detection problem by using three rounds of message exchanges, which is the minimum*

▼**Proof.** The number of messages is insufficient to find out the faulty peers within one round of message exchange because the source peer may be faulty. The source faulty peer may send 0's and 1's at the same frequency. It is impossible to determine the faulty peers within two rounds of message exchange if the number of peers is large. Furthermore, based on Theorem 3, FDP2P can solve the fault detection problem utilizing the concept of non-faulty like set in three rounds of message exchange without regard to the number of peers. Hence, three rounds of message exchange is the minimum. ▲

Theorem 5: *The number of allowable faulty peers is $\lfloor (n-1)/3 \rfloor$ in FDP2P protocol, which is the maximum.*

▼**Proof.** If the faulty peers exceed $\lceil n/2 \rceil$, then all may send different values to each peer. Faulty-free peers cannot obtain the common vertices or frontier. Thus, the protocol cannot be certain that all faulty-free peers can find out the faulty peers. If the total number of faulty peers is equal to $\lceil n/2 \rceil$, and n is an even number, then the number of 0's and 1's in the second level may be equal after applying the VOTE function. Under such conditions, all faulty-free peers cannot obtain a common value. Furthermore, according to the assumptions and constraints of the BA problem, the allowable component is peer only, and the faulty peers cannot exceed $\lfloor (n-1)/3 \rfloor$. These are identical to our constraints. Thus, the total number of allowable faulty peers is $\lfloor (n-1)/3 \rfloor$ in FDP2P. ▲

Theorem 6: *The message complexity is $O(n^2)$.*

▼Proof. In the first round of the message exchange phase, the source peer will send its initial value. Hence, one message must be generated. In the second round of the message exchange phase, all peers must send the received value during the first round of message exchange, and n messages must be generated. In the third round, $n*n$ messages must be generated. Therefore, the total quantity of messages to be generated during the execution of FDP2P is $(1 + n + n*n)$. The message complexity is $O(n^2)$. ▲

5. 3. Conclusions

In previous studies [9], [11], [20] of mobile P2P networks, most proposed protocols usually assume that peers are cooperative. Unfortunately, some peers may be un-cooperative or perform some transgression to crash and decrease the efficiency of the network systems in practice. As a result, it is important to detect faulty peers for the mobile P2P network, in which peers can immigrate into or emigrate from the network at any time.

There have some protocols [5], [9], [17], [23] been proposed to detect/locate the faulty processors based on BA problem protocol. However, these protocols require $2 * (\lfloor (n-1)/3 \rfloor + 1)$ rounds of message exchange and the message complexity is $O(n^{\lfloor (n-1)/3 \rfloor * \lfloor (n-1)/3 \rfloor})$. These protocols [5], [9], [17], [23] also do not concern the mobility issue. Hence, these protocols are not suitable and not efficient for the mobile P2P network in which there exits millions of peers, and peers can move around the network at any time.

In this study, we proposed a novel protocol called FDP2P to detect the faulty peers using three rounds of message exchange only, and the complexity of message can be reduced to $O(n^2)$. With less rounds of message exchange, fewer amounts of messages will be generated during executing FDP2P. This can help for saving the storage. In other words, the protocol has less overhead than previous works. Furthermore, the mobility issue is also been considered in FDP2P. Hence, FDP2P is superior to previous studies [5], [9], [17], [23].

References

[1] **A. Bar-Noy, D. Dolev, C. Dwork, R. Strong.** Shifting Gears: Changing Algorithms on the Fly to Expedite Byzantine Agreement. In: *Information and Computation*, vol. 97, pp.205-233, 1992. http://dx .doi.org/10.1016/0890-5401(92)90035-E.

[2] **B. Peter, W. Tim, D. Bart, D. Piet.** *A comparison of peer-to-peer architectures.*Ghent, Belgium: Broadband Communication Networks Group (IBCN), Department of Information Technology (INTEC), Ghent University.

[3] **H. S. Siu,, Y. H. Chin, W. P. Yang.** A Note on Consensus on Dual Failure Modes. In: *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, no. 3, pp. 225-230, 1996. http://dx.doi.org/10. 1109/71.491575.

[4] **H. S. Siu, Y. H. Chin, W. P. Yang.** Byzantine Agreement in the Presense of Mixed Faults on Processors and Links. In: *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 4, pp. 335-345, 1998. http://dx.doi.org/10.1109/71.667895.

[5] **H. S. Hsiao, Y. H. Chin, W. P. Yang.** Reaching Fault Diagnosis Agreement under a Hybrid Fault Model. In: *IEEE Transactions on Computers*, vol. 49, no. 9, pp. 980-986, 2000.

[6] **H. Yoshino, N. Hayashibara, T. Enokido, M. Takizawa.** Hierarchical Protocol for Byzantine Agreement in a Peer-to-Peer Overlay Network. In: *Sixteenth International Workshop on Database and*

Expert Systems Applications, pp. 5-9, 2005.

[7] **K. Aberer, Z. Despotovic.** Managing Trust in a Peer-2-Peer Information System. In: *ACM Fourteenth Conference on Information and Knowledge Management*, pp. 310-317, 2001.

[8] **K. Q. Yan, Y. H. Chin.** Achieving Byzantine Agreement in a Processor and Link Fallible Network. In *Proceedings of International Phoenix Conference on Computers and Communications*, pp. 407-412, 1989. http://dx.doi.org/10.1109/PCCC.1989.37423.

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

[10] **L. Lamport, R. Shostak, M. Pease.** The Byzantine Generals Problem. In: *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3 pp. 382-401, 1982. http://dx.doi.org/10.1145/357172.357176.

[11] **M. Adler, R. Kumar, K. Ross, D. Rubenstein, T. Suel, D. D. Yao.** Optimal peer selection for P2P downloading and streaming. In: *24th Annual Join Conference of the IEEE Computer and Communications Societies (INFOCOM 2005)*, pp. 1538- 1549, 2005.

[12] **M. Pease, R. Shostak, L. Lamport.** Reaching Agreement in the Presence of Faults. In: *Journal of ACM*, vol. 27, no. 2, pp. 228-234, 1980. http://dx.doi.org/10.1145/322186.322188.

[13] **Oztoprak, Kasim, and Akar, Gozde Bozdagi.** Two-Way/Hybrid Clustering Architecture for Peer to Peer Systems. In: *Second International Conference on Internet and Web Applications and Services*, pp. 11-11, 2007. http://dx.doi.org/10.1109/ICIW.2007.66.

[14] **S. C. Wang, K. Q. Yan, G.Y. Zheng.** Dual Agreement Virtual Subnet Protocol for Mobile Ad-hoc Networks. In: *The 22nd Annual ACM Symposium on Applied Computing*, pp. 953-954, 2007.

[15] **S. C. Wang, K.Q. Yan, H.C. Hsieh.** The New Territory of Mobile Agreement. In: *Computer Standards & Interfaces*, vol. 26, no. 55, pp. 435-447, 2004. http://dx.doi.org/10.1016/j.csi.2003.12.002.

[16] **S. C. Wang, K. Q. Yan, M. L. Chiang.** Optimal Agreement in a Scale-Free Network Environment. In: *Informatica International Journal*, vol. 17, no. 1, pp. 137-150, 2006.

[17] **S. C. Wang, Y. H. Chin, K. Q. Yan.** Reaching a Fault Detection Agreement. In: *The proceding of International Conference Parallel Processing*, pp. 251-258, 2007.

[18] **S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina.** The EigenTrust Algorithm for Reputation Management in P2P Networks. In: *Proc. WWW'03*, pp. 640-651, 2003.

[19] **S. Jun, M. Ahamad, J. Xu.** Robust Information Dissemination in Uncooperative Environments. In: *International Conference on Distributed Computing Systems*, pp. 293-302, 2005.

[20] **X. Jin, S. H. Gary Chan, W. P. Ken Yiu, Y. Xiong, Q. Zhang.** Detecting Malicious Hosts in the Presence of Lying Hosts in Peer-to-Peer Streaming. In: I*EEE international Conference on Multimedia and Expo*, pp. 1537-1540, 2006.

[21] **X. Zhang, J. Liu, B. Li, T. -S. P. Yum.** CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming. In: *IEEE Conference on Computer Communications*, March 2005.

[22] **Y. Cai, A. Natarajan, J. Wong.** On Scheduling of Peer-to-Peer Video Services. In: *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 1, pp. 140-145, 2007. http://dx.doi.org/10.1109/JSAC.2007.070114.

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

[24] **F. Preparata, G. Metze, R. Chien.** On the Connection Assignment Problem of Diagnosable Systems. *IEEE Transactions on Electronic Computing*, vol. 16, pp. 848-858, 1967. http://dx.doi.org/10.1109/PGEC.1967.264748.

[25] **Mallela S., Masson G. M.** Diagnosis without repair for hybrid fault situations. In: *IEEE Transactions on Computers*, vol. 29, no. 6, pp. 461-471, 1980. http://dx.doi.org/10.1109/TC.1980.1675605.

Received April 2011.