Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Autonomous fault-diagnosis and decision-making algorithm for determining faulty nodes in distributed wireless networks

Adel KHOSRAVI†, Yousef SEIFI KAVIAN

(*Electrical Engineering Department, Faculty of Engineering, Shahid Chamran University of Ahvaz, Ahvaz 61357-43337, Iran*)

†E-mail: a-khosravi@phdstu.scu.ac.ir

Received June 2, 2015; Revision accepted July 26, 2015; Crosschecked Aug. 16, 2016

Abstract: In this paper, we address fault-diagnosis agreement (FDA) problems in distributed wireless networks (DWNs) with arbitrary fallible nodes and healthy access points. We propose a new algorithm to reach an agreement among fault-free members about the faulty ones. The algorithm is designed for fully connected DWN and can also be easily adapted to partially connected networks. Our contribution is to reduce the bit complexity of the Byzantine agreement process by detecting the same list of faulty units in all fault-free members. Therefore, the malicious units can be removed from other consensus processes. Also, each healthy unit detects a local list of malicious units, which results in lower packet transmissions in the network. Our proposed algorithm solves FDA problems in $2t$+1 rounds of packet transmissions, and the bit complexity in each wireless node is $O(n^{t+1})$.

Key words: Fault diagnosis, Decision making, Byzantine agreement, Distributed wireless networks, Consensus
http://dx.doi.org/10.1631/FITEE.1500176 **CLC number:** TP393

1 Introduction

Distributed wireless networks (DWNs), such as mobile ad hoc networks (MANETs), vehicular ad hoc networks (VANETs), and wireless sensor networks (WSNs), may experience several types of failures that can result in faulty operation of the network and, in some cases, complete shutdown of network devices. Such faults can range from communication-based faults to software and hardware failures and malfunctions, which may introduce inaccurate information into the network. On the other hand, faulty behavior of a single node in distributed networks can interfere with the task of other units. Many applications, such as distributed clock synchronization and cooperative estimation and motion coordination (Jiang and He, 2005; Colon Osorio, 2007; Hsieh and Chiang, 2013; Wu and Rabbat, 2013; Silvestre *et al.*, 2014), need a consensus-based approach to help the members reach a common value. Researchers in previous studies have considered the consensus problem in two different approaches. From the first point of view, all network members are considered as fault-free units (Alekeish and Ezhilchelvan, 2012; Maggs *et al.*, 2012). Therefore, there is no adverse condition affecting the consensus process. In fact, this approach considers all network members as fault-free members, and the main goal is to find a common value among the members. Maggs *et al.* (2012) proposed average consensus algorithms for the clock synchronization problem in the WSNs. The algorithms compensate clock jitters in each unit. From the second point of view, the distributed network subjects to various kinds of failure. The Byzantine agreement (BA) (Lamport *et al.*, 1982; Wang SC *et al.*, 1995; Wang SS *et al.*, 2010) problem is one of the well-known problems in the area of distributed consensus. Since the introduction of BA by Lamport *et al.* (1982), various algorithms have been proposed for different network models (Okun and Barak, 2008; Wang *et al.*, 2010; Pasqualetti *et al.*, 2012). While BA

ORCID: Adel KHOSRAVI, http://orcid.org/0000-0002-8493-4462

has been studied extensively over the years, the original model has mostly continued to be used for different topologies.

A DWN is a system of distributed wireless nodes communicating using access points. The access points allow a distributed network to have a complete connectivity graph and also result in a larger network topology by connecting to other access points. The main goal of different studies is to develop an optimum algorithm to minimize the number of message exchange rounds and communication bits, and also to increase the number of allowable faulty units in the system. Many algorithms have been designed to solve the BA problem in different network structures, such as generalized connected and ad hoc networks (Wang *et al.*, 1995; Chiang *et al.*, 2009). Pease *et al.* (1980) presented an algorithm to solve the BA problem for fully connected networks in t+1 rounds, where t is the number of arbitrary faulty nodes in network N ($|N|=n$, $n>3t$, and $|\cdot|$ gives the number of nodes in network N). In general, all the algorithms designed to solve the BA problem must satisfy the following conditions:

1. All fault-free processors should agree on a common value.

2. If the source processor is fault-free, then all fault-free processors should consider the initial value of the source as the agreement value.

In this paper, the term 'processor' is used for faulty or fault-free wireless nodes and a router is named an access point.

Fischer and Lynch (1982) showed that t+1 rounds are necessary for solving any BA problem. In fact, local fault diagnosis will not help solve the BA problem in fewer rounds. When a faulty unit is detected by the network, there is no agreement between fault-free units about the detection of this fault. In other words, a fault-free unit does not know whether a specific faulty unit is detected by other fault-free units or not.

Many fault-diagnosis algorithms have been proposed to achieve a reliable distributed system. The main aim of fault diagnosis is to isolate faulty processors from the network for a better management of the network resources. There are two well-known types of fault-diagnosis approaches, namely test-based approaches (Wang *et al.*, 1990; Elhadef *et al.*, 2007) and evidence-based approaches (Buskens and Bianchini, 1993; Hsiao *et al.*, 2000; Ayeb and Farhat, 2004). Test-based approaches consist of a test phase in which a tester processor uses some test methods on the target units to check whether they are faulty or not. However, this type of approach is not well applicable in malicious systems in which the processors may have arbitrary faults. In contrast, evidence-based approaches use the information collected from the network, usually in consensus processes like BA, to detect faulty processors.

Fault-diagnosis models can also be divided into non-agreement (Ayeb and Farhat, 2004; Khosravi *et al.*, 2011) and agreement-based (Hsiao *et al.*, 2000) models. In non-agreement models, the list of detected faulty units may be different from one unit to another. In contrast, in agreement-based models, all fault-free processors find the same set of faulty units and malicious behavior of faulty units that does not affect the process. By solving the fault-diagnosis agreement (FDA) problem, faulty units could be removed from message transmissions, and this will help solve the BA problem in fewer rounds. All FDA algorithms must satisfy the following conditions:

1. Consensus: all the fault-free processors should detect the same set of faulty processors.

2. Fairness: no fault-free processor is falsely detected as a faulty unit by any fault-free processor.

Ayeb and Farhat (2004) proposed some efficient strategies for masking and locating faulty units. Their algorithm is an early stopping algorithm for solving the BA problem, which results in fewer rounds. However, their fault-detecting algorithm is an NP-hard problem in a general case and will intensify the computational complexity when the network size increases. Furthermore, they proposed a local fault-diagnosis model for situations in which the healthy members did not agree on the list of faults.

The FDAMIX algorithm (Hsiao *et al.*, 2000) uses the BA framework to solve the FDA problem. FDAMIX examines information collected from the BA solution to locate faulty processors. At the end of the message exchange phase in a network with n processors, the BA process must be performed n times on the stored message sets in each processor to find a common set of faulty units in all the processors. In fact, FDAMIX requires a huge number of messages be exchanged between the processors to solve the FDA problem. The algorithm proposed in this study reduces the number of message exchange rounds,

while increasing the number of detected faulty units. Each unit has two sets of faulty units. The first set stores locally detected faulty units, and the second set stores the detected faulty units that are common between the fault-free processors. Therefore, the number of faults detected using the proposed algorithm is increased compared to previous algorithms.

Our contributions in this paper include solving the FDA problem for a DWN, reducing bit complexity and transmission complexity of the FDA problem, and reducing bit complexity and transmission complexity of the BA problem by finding a local list of faults.

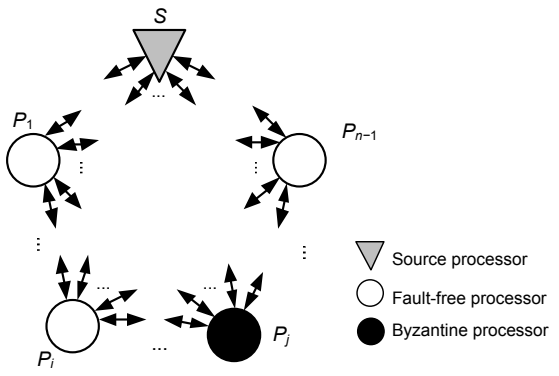

Fig. 1 A fully connected network with n processors and $\lfloor (n-1)/3 \rfloor$ faulty processors

2 The proposed algorithm

In the FDAMIX algorithm (Hsiao *et al.*, 2000), all arbitrary faulty units are detected at the end of the agreement process. However, the arbitrary faulty nodes are detected during the BA process in this study. Therefore, as a faulty unit is detected locally by some fault-free processors, the fault-free processors stop transmitting messages to the detected faulty processor. The network members in the proposed algorithm make an agreement about diagnosis of faulty units at the end of the FDA phase, and the detected faulty nodes are removed from the subsequent consensus processes. We assumed a fully connected DWN with reliable access points and arbitrary fallible processors in a synchronous environment. The definitions and terms used in this study are as follows:

S: the source processor of the BA process;

V: the set of all possible values for the BA problem, $V=\{0, 1, \lambda\}$, where λ is a default value such as NIL (not in list) or zero;

t: the maximum number of arbitrary faulty units;

p_a: the set of locally faulty units detected by a fault-free processor;

p_c: a list of globally faulty units detected in all fault-free processors;

$\mathrm{val}_i^k(P_j)$: the value sent from processor P_j to P_i at level (round) k;

FU(P_j): indicating that P_j is a faulty unit.

2.1 Basic fault identification concepts

Consider a network N with n processors and $\lfloor (n-1)/3 \rfloor$ arbitrary faulty processors (Fig. 1). At first, the source processor S sends its initial value to the rest of the network. Therefore, $\mathrm{val}_i^1(S)$ will be the value received by processor P_i from the source processor at the first level of exchanging message. In the second round, all receiver processors retransmit the received value from the source processor S to other members. All processors store the received values in an evidence-gathering tree (EG-tree) structure. The EG-tree for a fault-free processor contains the messages received from the neighbors. Similarly, in the third round, all non-source processors transmit the stored messages from the second round to other units and the received messages are stored at the third level of the EG-tree. In general, the source processor might also have arbitrary faults and send different values to different processors. In this case, all non-faulty processors must agree on a common value, either the majority of transmitted values or the default value λ.

Assume that the fault-free processor P_i has a (t+1)-level EG-tree as depicted in Fig. 2. In general, processors at level k of the message exchange phase send the stored messages from level $k-1$ to processor P_i. Let $\mathrm{val}_i^{k-1}(P_j)=\mathrm{val}(P_jP_i)$ be the values that P_i receives directly from processor P_j at level $k-1$ and $\mathrm{val}_i^k(\mathrm{val}_v^{k-1}(P_j))=\mathrm{val}(P_jP_vP_i)$ be the same values received by P_i from P_j via P_v at level k. Therefore, the following statement holds:

$$\text{if } \mathrm{val}(P_jP_i)\neq \mathrm{val}(P_jP_vP_i) \text{ then } \mathrm{FU}(P_j)\vee \mathrm{FU}(P_v), \tag{1}$$

where '$\vee$' is the logical OR symbol.

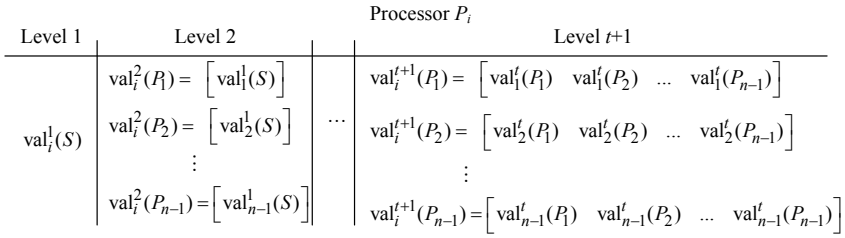

Fig. 2 A (t+1)-level EG-tree of processor P_i in Fig. 1

Since there is a disagreement between the values transmitted by P_v and P_j, and P_i is a fault-free receiver, at least one of the processors P_v and P_j is faulty. This fact is shown by the statement $\mathrm{FU}(P_j)\vee\mathrm{FU}(P_v)$, and we call it a probabilistic statement as is depicted in Eq. (1). By comparing the stored messages at the jth column of the matrices and at the kth level of the EG-tree, P_i generates a set of probabilistic statements as follows:

$$\sum P_i=\{\mathrm{FU}(P_{v1})\vee\mathrm{FU}(P_j),\mathrm{FU}(P_{v2})\vee\mathrm{FU}(P_j),\cdots\},\quad P_v,P_j\in N,\tag{2}$$

where $\sum P_i$ is the set of probabilistic statements in processor P_i related to the jth column of the matrices at the EG-tree and $\{P_{v1}, P_{v2}, \ldots\}$ are the processors that send different values from $\mathrm{val}_i^{k-1}(P_j)=\mathrm{val}(P_jP_i)$ at level $k-1$. $\sum P_i$ always leads to a prime argument as follows:

$$\Delta=\{\mathrm{FU}(P_j)\vee(\mathrm{FU}(P_{v1})\wedge\mathrm{FU}(P_{v2})\wedge\cdots)\},P_v,P_j\in N,\tag{3}$$

where ‘$\wedge$’ is the logical AND symbol.

Based on Eqs. (1) and (3), at least one of the statements $\mathrm{FU}(P_j)$ and $(\mathrm{FU}(P_{v1})\wedge\mathrm{FU}(P_{v2})\wedge\cdots)$ must hold. The statement $(\mathrm{FU}(P_{v1})\wedge\mathrm{FU}(P_{v2})\wedge\cdots)$ implies that all the processors $v=\{P_{v1}, P_{v2}, \ldots\}$ must be faulty. On the other hand, there are at most t faulty units in the network:

$$\text{if}\ \ |v|>t\ \ \text{then}\ \ \mathrm{FU}(P_j).\tag{4}$$

Eq. (4) implies that if the number of processors in set $\{P_{v1}, P_{v2}, \ldots\}$ is greater than t, then the statement $(\mathrm{FU}(P_{v1})\wedge\mathrm{FU}(P_{v2})\wedge\cdots)$ cannot be a diagnosis of faults; therefore, $\mathrm{FU}(P_j)$ holds.

Let $p_a=\{P_{j1}, P_{j2}, \ldots\}$ ($|p_a|=\tau$) be the set of faulty processors detected by the fault-free processor P_i. All probabilistic statements containing at least one of the units of p_a must be removed from the set of probabilistic statements, since they have no more information about the undetected faulty units. Also, removing the unacceptable statements from a prime argument as used in Eq. (3) would be based on $t-\tau$ faulty processors.

Let the following be the jth matrix at the kth level of the EG-tree of processor P_i and also suppose P_i to be a fault-free unit:

$$\mathrm{val}_i^k(P_j)=[\mathrm{val}_j^{k-1}(P_1)\ \ \mathrm{val}_j^{k-1}(P_2)\ \cdots\ \mathrm{val}_j^{k-1}(P_{n-1})].$$

Using a similar argument, P_i compares each column of $\mathrm{val}_i^k(P_j)$ (i=1, 2, …, $n-1$) with the corresponding directly received value from $P_1, P_2, \ldots, P_{n-1}$ at level $k-1$ and prepares a new set of probabilistic statements. For example, if $\mathrm{val}_j^{k-1}(P_{v1})$ is different from $\mathrm{val}_i^{k-1}(P_{v1})$, then a new probabilistic statement, $\mathrm{FU}(P_{v1})\vee\mathrm{FU}(P_j)$, is obtained.

Let P_i be a fault-free receiver processor and the matrices depicted in Fig. 3 be the messages stored at the kth level of the EG-tree. Also, suppose that P_j is a processor whose faulty status P_i wants to verify. Processor P_i identifies a set of different processors that have accused processor P_j. For this purpose, processor P_i compares the shaded section of the EG-tree in Fig. 3 with the values received directly from the corresponding processors at level $k-1$, and if there is a mismatch between the received messages, then the accuser processor is added to the list of processors that accuse P_j of being faulty, called the

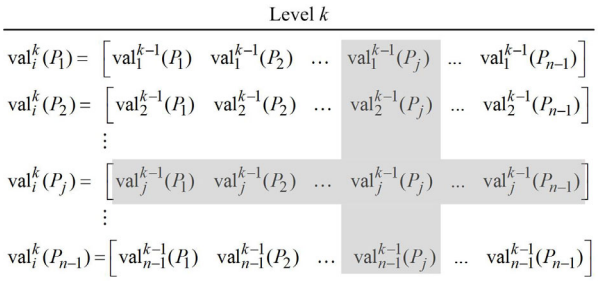

Fig. 3 Level k of the EG-tree stored in processor P_i (The shaded area is used to check the faulty status of P_j)

accuser list, denoted as AL_j. Let t and $|p_\mathrm{a}|=\tau$ be the maximum number of faulty processors in the network and the number of faulty processors detected by P_i, respectively. When a new processor is going to be added to AL_j, it has to be checked that the processor does not belong to p_a. Therefore, Rule 1 is used to identify faulty units in processor P_i:

Rule 1 If $|\mathrm{AL}_j|>t-\tau$ then $\mathrm{FU}(P_j)$.

If the fault of P_j is not detected using Rule 1, continue adding new accuser processors to AL_j in the next rounds. As soon as P_j is detected as a faulty processor, it is added to p_a, and the related matrices and parameters of P_j for detection of other faulty units will be removed from the list of accusers.

2.2 Byzantine agreement phase

This subsection illustrates the BA phase for the proposed algorithm. The BA phase is the first stage of the consensus process. This phase consists of two parallel subphases: message exchange and local fault diagnosis. The message exchange phase starts when the source processor transmits its initial value to all other processors, and since then the message exchange process is used to build a (t+1)-level EG-tree in each fault-free processor. The proposed algorithm benefits from the stored messages, which help identify faulty processors.

Using Rule 1, processors detect faulty units in each round. The list of detected faulty processors, p_a, for each fault-free processor is a set of locally detected faulty units. Therefore, at the end of the message exchange phase, all processors detect $|p_\mathrm{a}|$ faulty units locally. Fig. 4 shows the procedure for finding the agreement value in fault-free processor P_i. Now, all processors are ready to move to the FDA phase.

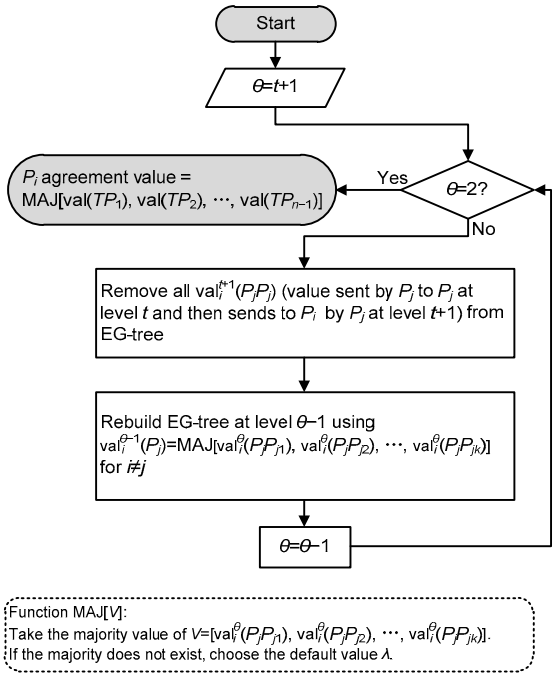

Fig. 4 Finding the Byzantine agreement value in fault-free processor P_i (T=source processor and MAJ is the majority function)

2.3 Fault-diagnosis agreement phase

To obtain a same set of faulty units for all fault-free processors, the processors are moved to the FDA phase. Using this phase, a general decision could be made on the detection of faulty processors. In this phase, there is no initial source processor as we know from the original BA problem. However, processors use their list of detected faulty processors, p_a, as the initial value received from a 'virtual source'. This phase consists of two subphases, namely, fault list exchange and global detection of faulty units.

The fault list exchange subphase is similar to the message exchange phase mentioned in Section 2.2. In this phase, each processor sends the list of detected faulty units, p_a, to other units. To provide a better vision to the exchange of lists of faulty units, each list is shown in a binary format. For example, if there are seven processors in network N and $p_\mathrm{a}=\{S, P_5\}$ for processor P_2, then p_a is given as follows:

$$p_\mathrm{a}=[1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0]. \tag{5}$$

Values 1 and 0 in Eq. (5) indicate faulty and fault-free statuses of a unit, respectively.

A tree structure similar to the EG-tree, called

FG-tree, is used here to store the detected faulty sets received from processors. Fig. 5 shows the FG-tree of P_i for two rounds. The FDA phase needs t+1 message exchange rounds for the agreement. If we consider p_a as the set of faulty units detected by P_i, then $\mathrm{val}_i^1(P_i)=p_a$ is considered as the value sent from the virtual source, as is presented in Fig. 5. Each processor sends its list of detected faulty units in a second round to other processors. In next rounds, the processors send the latest stored messages in their FG-tree to the rest of the network. However, vertices with repeated index are excluded from message transmission. For example, if val(P_jP_i) is the value received by P_i from P_j in the current round, then P_i would not send it to P_j in the next round to save energy. When processors have prepared a (t+1)-level FG-tree, they are ready to move to the subphase of global detection of faulty units and determine which of the detected faulty units is detected by all fault-free processors.

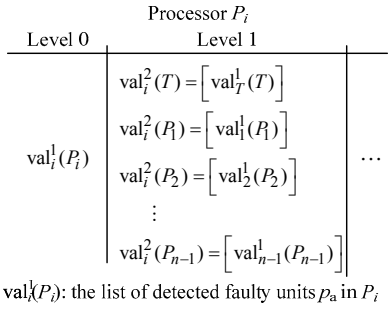

Fig. 5 The FG-tree stored in processor P_i for two levels (T=source processor)

In the subphase of global detection of faulty units, processors make a common decision about the faulty status of each processor in the network and put the commonly detected faulty units in list p_c. For this purpose, a decision-making strategy is implemented on each column of received lists of faulty units. Fig. 6 shows the flowchart of the subphase of global detection of faulty units. Fault-free processor P_i uses the messages stored at level θ=k to prepare the voted messages for level θ=k−1. Therefore, when P_i prepares the final matrix of voted messages at level θ=2, this matrix can be used to make an agreement about the detection of faulty units. Let [val(P_1), val(P_2), …, val(P_{n-1})] be the prepared matrix of voted messages at the second level of the FG-tree. If the number of elements with value 1 in [val(P_1), val(P_2), …, val(P_{n-1})] corresponding to processor P_{jk} ($P_{jk}\notin p_c$ and $P_{jk}\neq P_j$) is more than $t-|p_c|$, then processor P_i adds P_j to p_c.

The procedure in Fig. 6 is repeated for all columns of p_a to determine the statuses of all processors. An example is provided in the following section to illustrate different aspects of the proposed algorithm.

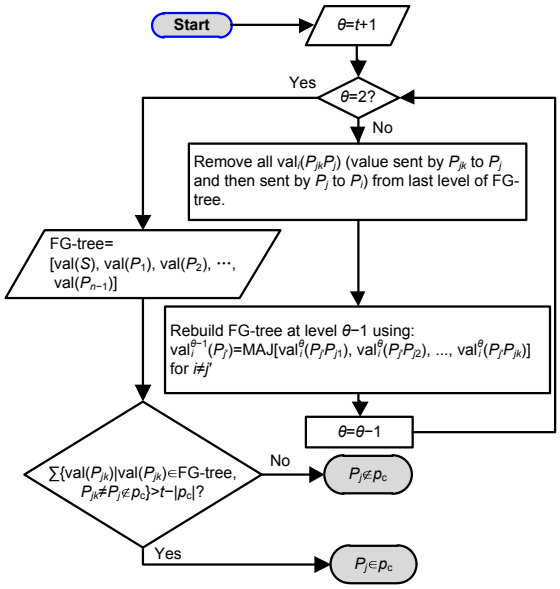

Fig. 6 Flowchart of the subphase of global detection of faulty units in processor P_i to check the status of P_j

3 An example of execution of the proposed algorithm

This section shows how the proposed algorithm acts using an example. A network N={S, P_1, P_2, …, P_6} with seven processors and two faulty processors {S, P_6} are considered. The worst case in the BA problem is when the source processor is faulty. Therefore, this situation is considered for the example.

The process starts by the source processor transmitting its initial value to other processors. Since the source processor is supposed to be faulty, it might transmit arbitrary values to the rest of the processors. According to the number of faulty units in the network, the BA problem needs $\lfloor(7-1)/3\rfloor+1=3$ rounds of message exchange. Therefore, the message exchange continues for three rounds to prepare a three-level EG-tree in each processor. Consider that the faulty source sends {1, 1, 1, 1, 0, 0} to {P_1, P_2, P_3, P_4, P_5, P_6} in the first round of message exchange,

respectively. At the next round, the processors send the received value from the source processor to each other. Fig. 7 shows the EG-tree saved in fault-free processor P_1 for three rounds. Based on the EG-tree and Rule 1, processor P_1 locates faulty units in the network. From data stored at the second level of the EG-tree, P_1 adds $\{P_5, P_6\}$ to AL_S. Also, it can be seen that the source processor accuses both P_5 and P_6. Therefore, S is added to AL_{P5} and AL_{P6}. From the received messages at the third level, P_6 is added to AL_{P4} and AL_{P5} and $\{S, P_4, P_5\}$ is added to AL_{P6}. Fig. 8 shows the detailed fault detection procedure from the data stored at the EG-tree in Fig. 7. Processor P_1 has $p_a=\{\}$, $p_c=\{\}$, and $t=2$ so far. Since $|AL_{P6}|=3>t-|p_a|$, P_6 is detected as a faulty unit and is added to p_a by P_1. Now, a new faulty unit is detected, and P_1 must reconsider the evidence received from other processors. Therefore, P_6 is removed from AL_S, AL_{P4}, and AL_{P5}, and also the maximum number of undetected faulty units decreases to $t-|p_a|=1$. However, $AL_S=\{P_5\}$ and $AL_{P4}=\{\}$, and therefore, no new faulty processor is detected using Rule 1.

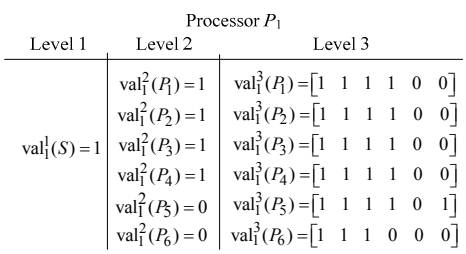

Fig. 7 A three-level EG-tree for processor P_1 in the example

At the end of the message exchange phase, the processors move to the decision-making phase to find the agreement value. To find the agreement value, the last level of the EG-tree should be modified by removing redundant values. The majority function (MAJ) is performed on each column of the EG-tree to reconstruct the EG-tree in the previous round. Fig. 9 shows the procedure of finding the agreement value from the modified EG-tree. Since P_1 is a fault-free receiver, the majority function is used only to find the values of other processors, and the value received directly from the source by P_1 helps find the final agreement at the second level as depicted in Fig. 9.

Next, all processors proceed to the next phase to make an agreement about the detection of faulty units.

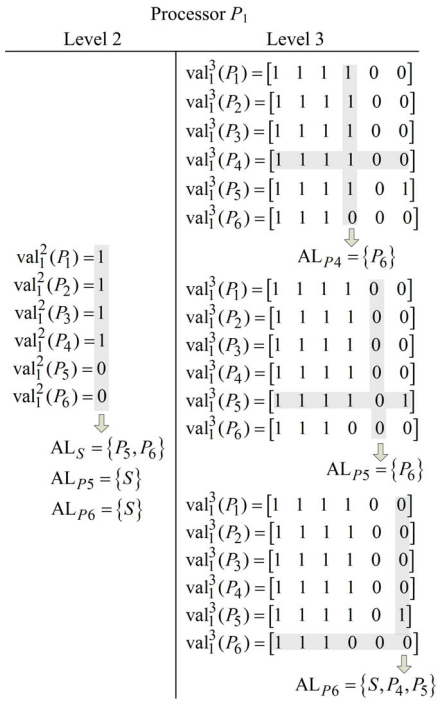

Fig. 8 Local fault diagnosis in processor P_1 during the Byzantine agreement process

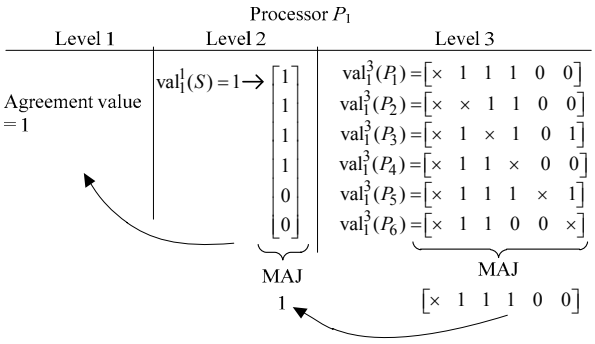

Fig. 9 Decision-making phase to find the Byzantine agreement value in processor P_1

At the beginning of the FDA phase, P_1 has $p_a=\{P_6\}$ and is ready to send it to other processors. However, in the BA phase, processors P_2, P_3, P_4, and P_5 have detected $p_a=\{P_6\}$, $p_a=\{P_6\}$, $p_a=\{P_6\}$, and $p_a=\{S, P_6\}$, respectively, and $p_a=\{S, P_5\}$, a random value, for P_6 is sent to P_1. In the fault list exchange subphase, processors send their detected faulty units to each other. Fig. 10 shows the stored FG-tree in P_1 for three rounds.

When all processors prepare a three-level FG-tree, they are ready to find the commonly detected faulty processors from the FG-tree. For this purpose, two sets of values should be removed from the last

level of FG-trees. First, if P_1 is verifying the status of P_j, then the majority of values indicating P_j's opinion about its own faulty status should be removed from the FG-tree. Second, processor P_1 does not need to identify its own faulty status. Therefore, the majority function is not performed on the related values.

Fig. 11 illustrates the subphase of global detection of faulty units in processor P_1. The shaded area in Fig. 11 indicates the vertices that have performed the majority function on them, and the resultant values at the bottom are the values to be replaced at the second level. However, since P_1 is the receiver processor, it is not necessary to find the majority of values for P_1 at the third level, and $\mathrm{val}_1^1(P_1)$ would be used for this purpose. As soon as the FG-tree is reconstructed at the second level, an agreement is made about the faulty processors. For this purpose, P_1 counts the number of elements with value 1 at the columns of the second level to verify the condition of the corresponding processor. At first, P_1 has p_c={}, and if the number of elements with value 1 in all jth columns is larger than $t-|p_c|$=2, then P_{j-1} is faulty (considering P_0=S). Based on this, P_1 adds P_6 to p_c, and this means that all fault-free processors have also detected p_c={P_6}. Next, P_1 rechecks messages at the second level, this time using $t-|p_c|$=1 bound and noticing that $\mathrm{val}_1^2(P_6)$ must be removed from the messages first. As shown in Fig. 11, no new faulty unit is added to p_c and processor P_1 finishes the algorithm by p_a={} as a set of locally detected faulty units and p_c={P_6} as an FDA list. By detecting a common set of faults, processors can make a decision to remove faulty processors from the network or rearrange the network structure based on this agreement.

The next section provides proofs for correctness of the proposed algorithm. Also, the complexity of the proposed algorithm is compared with those of previous algorithms in terms of the number of message exchange rounds and bit complexity.

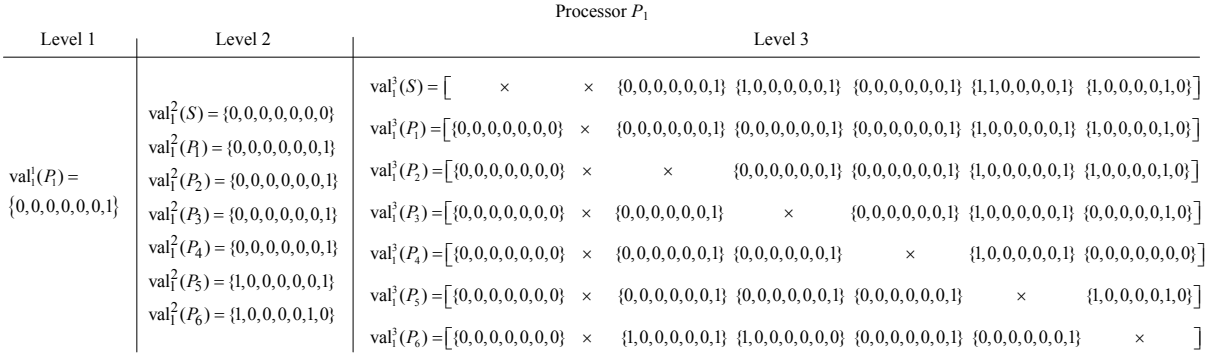

Fig. 10 The FG-tree stored in processor P_1 in FDA's subphase of global detection of faulty units

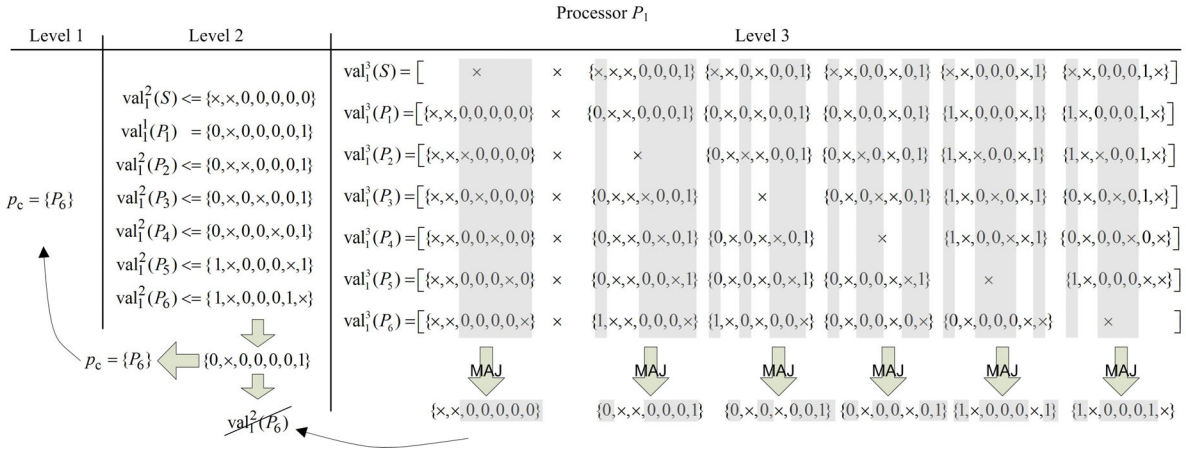

Fig. 11 The subphase of global detection of faulty units in processor P_1

4 Correctness of the proposed algorithm

This section provides the proofs for the correctness of the proposed algorithm. At first, we show in Theorem 1 that no fault-free processor is detected as a faulty unit by the rest of the fault-free processors in the FDA phase. In other words, the FDA phase makes agreement only about detecting faulty units and satisfies the fairness condition.

Theorem 1 Let N ($|N|=n$) be a network of wireless agents, P_0, P_1, …, P_{n-1} its members, and P_{j1}, P_{j2}, ..., P_{jt} the faulty units with $t=\lfloor (n-1)/3 \rfloor$ the maximum number of faulty units in N. If P_i is a fault-free processor in N, then there will be no agreement about detection of P_i as a faulty unit among fault-free processors using the proposed algorithm.

Proof Since there are at most t faulty units in the network and therefore $|\mathrm{AL}_{Pi}|\leq t$ for all fault-free units, no fault-free processor sends $p_a=\{P_i\}$ in the FDA message exchange phase. Based on this fact, there are at most $t=\lfloor (n-1)/3 \rfloor$ faulty units that may send $p_a=\{P_i\}$ to fault-free processors in this phase. This situation can be rephrased to a situation in which a 'virtual' fault-free source sends $p_a=\{0\}$ to network N and only the existing t faulty processors may change this value. However, by considering the virtual source as a member of N, there will be $n+1$ processors in the network and the virtual source acts as a fault-free processor. By removing the values of P_i from the message exchange process in the FDA phase, there will be n processors in the network and $n>3t+1$. Therefore, by implementing the decision-making strategy on the FG-tree and preparing the majority of values at the second level, there will be at most $t=\lfloor (n-1)/3 \rfloor$ elements from P_{j1}, P_{j2}, ..., P_{jt} with $p_a=\{1\}$, which indicates P_i is a faulty unit. When $p_c=\{\}$ in all processors, then $\lfloor (n-1)/3 \rfloor \leq t-|p_c|$ and no processor adds P_i to p_c. On the other hand, if $p_c\neq\{\}$, then there will be an agreement about diagnosis of some faulty units among fault-free processors, and the values related to the detected faulty unit are removed from the FG-trees. Based on this, in a general case, the number of faulty units that have sent $p_a=\{P_i\}$ to the network and whose values are not removed from FG-trees is not larger than $\lfloor (n-1)/3 \rfloor-|p_c|$, and the term $\lfloor (n-1)/3 \rfloor-|p_c|\leq t-|p_c|$ is true in all fault-free processors.

Based on Theorem 2, whenever the status of a faulty processor is checked in the FDA phase, we can suppose that a virtual faulty source has sent different values to different processors. Also, it shows that this situation does not affect the FDA phase.

Theorem 2 Let N ($|N|=n$) be a network, P_0, P_1, …, P_{n-1} its members, and P_{j1}, P_{j2}, ..., P_{jt} the faulty units with $t=\lfloor (n-1)/3 \rfloor$ the maximum number of faulty units in network N. If P_j is a faulty unit, then the reconstructed FG-tree at the second level in the subphase of global detection of faulty units is common in all fault-free processors.

Proof Network N with t faulty processors needs $t+1$ rounds of message exchange to reach the agreement among its processors. In the case of the faulty source in the BA problem, the values reconstructed in the decision-making phase at the second level of all tree structures are common among all fault-free processors. To illustrate the case, suppose that the source processor has sent value 0 to n' processors and value 1 to n' processors in a network with totally $n=2n'+1$ processors. Since BA results in a common value in all fault-free processors and also the resultant majority of values at the second level relating to fault-free processors are the same as their received values from the source processor, the values related to faulty processors must be common at the reconstructed second level of all tree structures to have a common agreement. In the FDA phase and the subphase of global detection of faulty units, when processors want to verify the faulty status of processor P_j, the exchanged values of P_j are removed from the FG-trees. Therefore, the number of remaining faulty units decreases to $t-1$. On the other hand, P_j is detected as a faulty unit by some of the processors, i.e., $P_j\in p_a$ for some processors and $P_j\notin p_a$ for others. This situation is similar to the case of the virtual faulty source sending $p_a=1$ and $p_a=0$ to different processors. However, by removing the values of faulty processor P_j and considering the effects of the virtual faulty source, the number of allowable faulty units does not exceed t bound. Therefore, the numbers of elements with values '1' and '0' are the same at the second level of all FG-trees after the subphase of global detection of faulty units.

It has been shown in Theorem 2 that if the processors check the faulty status of a faulty processor in

the FDA phase, then the reconstructed matrices at the second level of all FG-trees are the same. Theorem 3 shows how to find the faulty status of a unit from the stored data at the second level.

Theorem 3 Let N ($|N|=n$) be a network, P_0, P_1, …, P_{n-1} its members, and P_{j1}, P_{j2}, ..., P_{jt} the faulty units with $t=\lfloor (n-1)/3 \rfloor$ the maximum number of faulty units in network N. Let $P_{j'1}$, $P_{j'2}$, ..., $P_{j'k}$ be the members of p_c and $\mathrm{val}^2(P_{j'1})$, $\mathrm{val}^2(P_{j'2})$, …, $\mathrm{val}^2(P_{j'k})$ be removed from all FG-trees in the subphase of global detection of faulty units. If in the FDA phase, to check the status of P_j, there are more than $t-|p_c|$ elements with value '1' at the second level of the reconstructed FG-tree of fault-free processor P_i, then P_j is a faulty unit.

Proof Suppose that the fault-free processor P_i has sent $\mathrm{val}_i^1=1$ to the rest of the processors in the message exchange phase of BA. Since there are not more than $\lfloor (n-1)/3 \rfloor$ faulty processors affecting the MAJ function in the decision-making phase and reconstructing the tree structures in processors, $\mathrm{val}_i^1=1$ is stored at the second level of all tree structures after the reconstruction phase. First, suppose $p_c=\{\}$. If the number of elements with value 1 at the second level of the reconstructed FG-trees equals τ and $\tau>t$, since there are at most t faulty processors in the network, the receiver processor is sure that at least one of the accuser transmitter processors is fault-free, and based on Theorem 1 and Ayeb and Farhat (2004), P_j is detected as a faulty unit. When $p_c\neq\{\}$, data from processors that are members of p_c are removed from the second level of the FG-trees. Therefore, the maximum number of remaining faulty units in the network is $t-|p_c|$, and if $\tau>t-|p_c|$, then it results in the detection of faulty P_j.

Corollary 1 From Theorems 1–3, it is observed that the proposed algorithm satisfies the FDA conditions, and p_c is the common list of faults detected in fault-free processors and also p_a is the local list of detected faults.

Corollary 2 The proposed algorithm needs $2t+1$ rounds of message exchange to solve BA and FDA problems simultaneously.

Corollary 3 The bit complexity for the BA problem in the proposed algorithm is $O(n^t)$ and for the FDA problem is $O(n^{t+1})$.

5 Performance analysis

The algorithm proposed in this study aims to solve the FDA problem in a distributed environment. The algorithm has three properties: (1) It can detect locally a high percentage of malicious faulty units during the BA process; (2) It implements a new agreement framework to make an agreement among fault-free processors about diagnosis of faulty units; (3) The faulty units detected globally are removed from the next consensus processes to decrease the number of message exchange rounds.

The proposed algorithm needs $2t+1$ message exchange rounds to solve the BA and FDA problems simultaneously, in comparison to the $(n+1)(t+1)$ rounds required by the algorithm proposed by Hsiao *et al.* (2000). Also, the bit complexity is improved from $O(n^{2t+1})$ to $O(n^{t+1})$ in the proposed algorithm to solve the FDA problem. Table 1 compares the proposed algorithm with some known fault-diagnosis algorithms. The proposed algorithm is able to detect high percentages of faulty units with minimum message overheads.

Table 1 Comparison of FDA algorithms

Algorithm	Local fault diagnosis	Bit complexity	Number of rounds	Consensus	
				BA	FDA
Hsiao *et al.* (2000)	–	$O(n^{2t+1})$	$(n+1)(t+1)$	√	√
Ayeb and Farhat (2004)	√	$O(n^{t-\rho})$	$t+1-\rho$	√	–
Chiang *et al.* (2009)	√	$O(n)$	3	–	–
Our method	√	$O(n^{t+1})$	$2t+1$	√	√

The algorithm in Ayeb and Farhat (2004) can stop message transmission based on the number of detected faulty units

Here, we investigate the fault-diagnosis capability of the proposed algorithm in the presence of Byzantine faults. For this reason, we simulate a communication network N ($|N|=n$) with a complete network graph. The network has $\lfloor (n-1)/3 \rfloor$ Byzantine faulty units. Each processor has two sets of values at the end of the algorithm, including globally detected faulty units and locally detected faulty processors. The numbers of globally and locally detected faulty processors per 100 runs are presented in Fig. 12, for different fault rates of Byzantine units. The fault rate of Byzantine units is modeled as the rate of

randomness of their transmitted value or the probability of altering the value received from the neighbors. One can predict that the percentage of revealed faults increases with a higher fault rate. For fault rates higher than 10%, the chance of detecting a faulty unit increases dramatically (Fig. 12b). This is because increasing the number of transmitted messages (evidence) in the network helps a healthy processor find a faulty unit.

The proposed algorithm is able to identify faulty processors locally, which is not possible in previous studies. As presented in Fig. 12a, each healthy processor has two sets of detected faults. The globally detected fault set is the only set that was identified by Hsiao *et al*. (2000). However, one can notice from Fig. 12a that for lower fault rates, the number of faults detected globally (FDA) is considerably lower than that of faults detected locally.

Fig. 13 shows the effect of network size on the performance of the proposed algorithm. For a smaller network, a small chance exists for healthy members to identify the faulty processors. Generally, for networks with less than 7 nodes (where there is only one Byzantine unit), regardless of the fault rate, there is a small chance of reaching the FDA. In contrast, for larger networks, the chance of reaching FDA improves dramatically. As presented in Fig. 13b, the fault rate of Byzantine units has a small effect on the number of faulty units detected for larger networks.

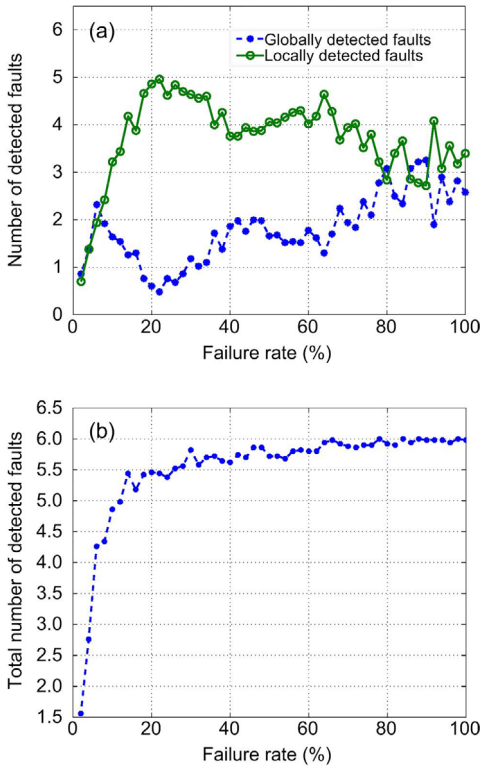

Fig. 12 Number of faults detected in processor P_1: (a) faults detected globally and locally; (b) total number of faults detected

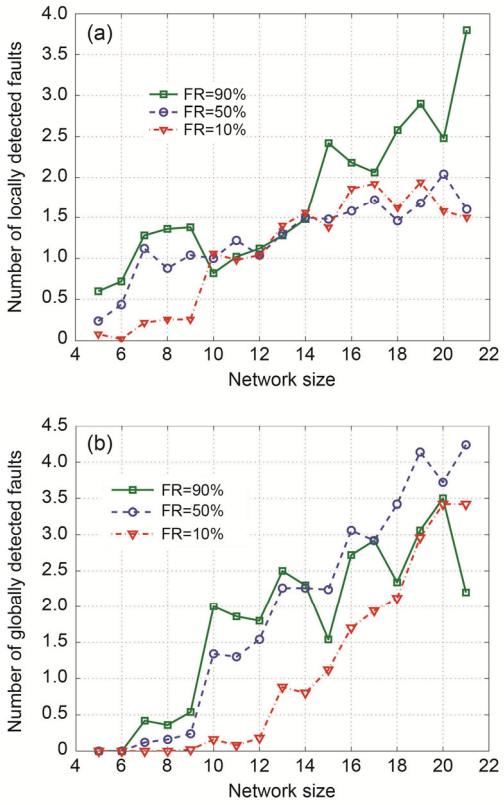

Fig. 13 Number of faults detected locally (a) and globally (b) in processor P_1 for different network sizes (FR: failure rate)

6 Conclusions and future work

In this paper, we provide a solid solution for the fault-diagnosis problem using the framework of the Byzantine general problem. The proposed solution is compared with prominent Byzantine agreement (BA) algorithms in terms of the number of required message transmissions and packet size complexity. The proposed solution simplifies the solution of fault-diagnosis agreement (FDA) and BA problems in complete graph networks. We provide a new evidence-based fault-diagnosis algorithm that

benefits from contradictory transmissions of Byzantine members to reveal their faulty behavior.

BA is a challenging problem in distributed systems and may lead to impossibility in some network topologies where the network graph is incomplete. Also, the problem suffers from synchronization and requires varying rounds to find the agreement. As an alternative to a definite solution for the BA problem, one can consider almost sure solutions. Computing the probability of a value being the agreement value and improving this probability will be an open problem for future work.

References

Alekeish, K., Ezhilchelvan, P., 2012. Consensus in sparse, mobile ad hoc networks. *IEEE Trans. Parall. Distrib. Syst.*, **23**(3):467-474. http://dx.doi.org/10.1109/TPDS.2011.182

Ayeb, B., Farhat, A., 2004. A flexible formal framework for masking/demasking faults. *Inform. Sci.*, **159**(1-2):29-52. http://dx.doi.org/10.1016/j.ins.2003.03.004

Buskens, R.W., Bianchini, R.P., 1993. Distributed on-line diagnosis in the presence of arbitrary faults. 23rd Int. Symp. on Fault-Tolerant Computing, p.470-479. http://dx.doi.org/10.1109/FTCS.1993.627350

Chiang, M.L., Wang, S.C., Tseng, L.Y., 2009. An early fault diagnosis agreement under hybrid fault model. *Expert Syst. Appl.*, **36**(3):5039-5050. http://dx.doi.org/10.1016/j.eswa.2008.06.009

Colon Osorio, F.C., 2007. Using Byzantine agreement in the design of IPS systems. Int. Performance, Computing, and Communications Conf., p.528-537. http://dx.doi.org/10.1109/PCCC.2007.358936

Elhadef, M., Boukerche, A., Elkadiki, H., 2007. An adaptive fault identification protocol for an emergency/rescue-based wireless and mobile ad-hoc network. IEEE Int. Parallel and Distributed Processing Symp., p.1-8. http://dx.doi.org/10.1109/IPDPS.2007.370589

Fischer, M.J., Lynch, N.A., 1982. A lower bound for the assure interactive consistency. *Inform. Process. Lett.*, **14**(4):183-186. http://dx.doi.org/10.1016/0020-0190(82)90033-3

Hsiao, H.S., Chin, Y.H., Yang, W.P., 2000. Reaching fault diagnosis agreement under a hybrid fault model. *IEEE Trans. Comput.*, **49**(9):980-986. http://dx.doi.org/10.1109/12.869331

Hsieh, H.C., Chiang, M.L., 2013. Robustness improvement for mobile P2P network by the Byzantine Agreement problem. 10th Annual Conf. on Wireless On-demand Network Systems and Services, p.104-106. http://dx.doi.org/10.1109/WONS.2013.6578329

Jiang, J., He, C., 2005. A novel mutual authentication and key agreement protocol based on NTRU cryptography for wireless communications. *J. Zhejiang Univ.-Sci.*, **6A**(5):399-404. http://dx.doi.org/10.1631/jzus.2005.A0399

Khosravi, A., Mohammadi, K., Shiroie, M., 2011. Locating malicious links in fully-connected networks using a formal framework. Proc. Int. Conf. on Systems Engineering, p.247-250. http://dx.doi.org/10.1109/ICSEng.2011.51

Lamport, L., Shostak, R., Pease, M., 1982. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, **4**(3):382-401. http://dx.doi.org/10.1145/357172.357176

Maggs, M.K., O'Keefe, S.G., Thiel, D.V., 2012. Consensus clock synchronization for wireless sensor networks. *IEEE Sensors J.*, **12**(6):2269-2277. http://dx.doi.org/10.1109/JSEN.2011.2182045

Okun, M., Barak, A., 2008. Efficient algorithms for anonymous Byzantine agreement. *Theory Comput. Syst.*, **42**(2):222-238. http://dx.doi.org/10.1007/s00224-007-9006-9

Pasqualetti, F., Bicchi, A., Bullo, F., 2012. Consensus computation in unreliable networks: a system theoretic approach. *IEEE Trans. Autom. Contr.*, **57**(1):90-104. http://dx.doi.org/10.1109/TAC.2011.2158130

Pease, M., Shostak, R., Lamport, L., 1980. Reaching agreement in the presence of faults. *J. ACM*, **27**(2):228-234. http://dx.doi.org/10.1145/322186.322188

Silvestre, D., Rosa, P., Hespanha, J.P., *et al.*, 2014. Finite-time average consensus in a Byzantine environment using set-valued observers. American Control Conf., p.3023-3028. http://dx.doi.org/10.1109/ACC.2014.6859426

Wang, S.C., Chin, Y.H., Yan, K.Q., 1990. Reaching a fault detection agreement. Proc. Int. Conf. on Parallel Processing, p.251-258.

Wang, S.C., Chin, Y.H., Yan, K.Q., 1995. Byzantine Agreement in a generalized connected network. *IEEE Trans. Parall. Distrib. Syst.*, **6**(4):420-427. http://dx.doi.org/10.1109/71.372796

Wang, S.S., Yan, K.Q., Wang, S.C., 2010. An optimal solution for Byzantine agreement under a hierarchical cluster-oriented mobile ad hoc network. *Comput. Electr. Eng.*, **36**(1):100-113. http://dx.doi.org/10.1016/j.compeleceng.2009.06.002

Wu, S., Rabbat, M.G., 2013. Broadcast gossip algorithms for consensus on strongly connected digraphs. *IEEE Trans. Signal Process.*, **61**(16):3959-3971. http://dx.doi.org/10.1109/TSP.2013.2264056