Abstract

Reliability is an important research topic of distributed systems. To achieve fault-tolerance in the distributed systems, healthy processors need to reach a common agreement before performing certain special tasks, even if faults exist in many circumstances. This problem is called as the Byzantine Agreement (BA) problem and it must be addressed. In general, the traditional BA problem is solved in well-defined networks. However, the MANETs (Mobile Ad-hoc NETwork) are increasing in popularity and its network topology is dynamic in nature. In this paper, the BA problem is re-examined in MANETs. Our protocol uses the minimum number of message exchanges to reach an agreement within the distributed system while tolerating the maximum number of faulty processors in MANETs.

Keywords—Byzantine Agreement; Distributed System; Fault-tolerant; Consensus; Mobile Ad-hoc Network

1 Introduction

A distributed computing system consists of a set of processors, which can communicate with each other by exchanging messages. In order that a computer system is reliable, a mechanism allowing a set of processors to agree on a common value is needed [7,12].

Some examples of such applications are: a commitment problem in a distributed database system [4,12], a clock synchronization problem [5], and a landing task controlled by a flight path finding system [1]. Such a unanimity problem was first studied by Lamport et al. [7], and called a Byzantine Agreement (BA) [1,4,5,7,9]. This problem requires a number of independent processors to reach an agreement in cases where some of those processors might be faulty. Furthermore, the goal of BA is that the healthy processors achieve a common value.

In BA problem, the symptoms of processor failure can be classified into two categories, the dormant fault and arbitrary fault [5,7]. Dormant processor faults include broken processors (crash faults) and message misses (omission faults), and are easy to detect and solve. However, arbitrary faults are unpredictable and damaging, and thus are a more serious problem than dormant faults.

In addition, a closely related sub-problem, the consensus problem, has been extensively studied
The consensus problem has \( k \) initial values in \( k \)-processors system, and subsequently achieves a common value even if certain processors fail \([5,9,12,18]\). Therefore, the consensus problem is similar to the BA problem in that it executes \( k \) copies BA processes. Subsequently, the result of Fischer et al. \([5]\), showing agreement, is impossible in an asynchronous environment with even one processor failure. In addition, Lamport argues for the consensus problem under the assumption of synchronous behavior BA, showing that \( 3f_p+1 \) processors are allowed \( f_p \) failures where \( f_p \) is the number of faulty processors in the network \([7]\). For clearing of this study, the assumptions of the BA are used to explain the concept of the consensus problem.

Traditionally, the BA problem was defined by Lamport et al. \([7]\), as follows:

1. There are \( k \) (\( k \geq 3 \)) processors, of which at most one-third of the total number of processors could fail without breaking down a workable network;
2. The processors communicate with each other through message exchange in a fully connected network (or well-defined network);
3. The message’s sender is always identifiable by the receiver;
4. An arbitrary faulty processor is chosen as a source, and its initial value is broadcasted to other processors and to itself to execute the protocol.

In general, a healthy source processor sends the same values to all processors and arbitrary processors cannot affect the root values sent from the healthy source processor. However, the source processor, which has arbitrary faults, may transmit different values to different processors. This situation is the worst case of the BA problem and is worth discussing. Therefore, we assume the source processor is an arbitrary processor in (4).

Besides, various protocols for the BA problem have been developed in order to meet the following requirements \([1,4,5,7,9,16,18,19]\):

- (BA1) Agreement: All healthy processors agree on a common value \( v \).
- (BA2) Validity: If the initial value of the source is \( v_s \) and the source is healthy, then all healthy processors shall agree on the value \( v_s \); i.e., \( v = v_s \).

Under these assumptions and requirements, several protocols \([1,4,5,7,9,16,18,19]\) have been proposed for solving such problems. The protocol in Lamport et al. \([7]\) indicates that \( f_p+1 \) (\( f_p \leq \lfloor (k-1)/3 \rfloor \)) rounds (a round denotes the interval of message exchange) of message exchange are required to reach a common agreement in a synchronous fully connected network with \( k \) processors. Further, Fischer and Lynch \([5]\) point out that \( f_p+1 \) rounds are the minimum number of rounds needed for sufficient messages to achieve BA.

As the network technology continues to grow at a high rate of speed, traditional network topology is improved with wireless topology such as Mobile Ad-hoc NETwork (MANETs). MANETs, consisting of wireless processors that communicate with each other in the absence of a fixed infrastructure, is different
from the traditional network structures [1,7].

MANETs can be used flexibly and quickly in automated battlefields, disaster relief, and rescue. In general, each mobile processor can communicate with another within its own wireless transmission range. The mobile processor sends the message to the destination processor located outside its wireless transmission range by forwarding it via another mobile processor. Its topology, as shown in Figure 1, can be modeled as a unit-disk graph [2,6] according to the strength of transmission power.

In general, a MANET is built randomly when mobile processors want to communicate with each other within a specific range. Therefore, there exist several challenges to a MANET due to its dynamic nature, such as low battery power, limited bandwidth, and restricted mobility. As mentioned, the traditional routing protocols focusing on aspect include hierarchical routing [10], linking state [8], and distance vector [2,3]. Unfortunately, processors may immigrate into or emigrate away from the network at any time, thus the previous routing path will be destroyed. Therefore, many researches use the concept of a virtual backbone for routing MANETs [2,8,13,14,15,17].

Regardless of how the processors reach an agreement in the MANETs, the presence of faulty processors needs to be addressed. The symptom of a faulty processor is usually unrestrained, and is commonly called an arbitrary fault [1,7]. In such a fault, a processor can withhold messages or collude with other faulty processors to send irregular message to others. However, some arbitrary-resilient BA protocols [4,7] treat all faults as arbitrary faults, even though some faults may be subjected to dormant faults, such as crash or fail-stop faults. This treatment ignores that the faulty behaviors of dormant faults are be served as those of arbitrary faults. When a dormant fault exhibits its faulty behavior, it can be detected and ignored by all healthy processors.

Thus, arbitrary-resilient BA protocols cannot tolerate the maximum number of faults if the dormant faults exist. These observations motivate the study of the BA problem under a dual fault model (arbitrary and dormant faults exist simultaneously) [9,5,12,16]. The goal of such study is to maximize the number of allowable faulty processors in dual fault mode. Therefore, in this paper, the BA problem is re-examined by investigating the dual fault and exploring how healthy processors reach agreement in MANETs.

The rest of this paper is organized as follows: Section 2 illustrates the basic assumption of MANETs. Section 3 shows the basic concept and approaches in this study. The detail of the protocol GCAP we propose is shown in Section 4. Section 5 illustrates examples of GCAP in detail. Subsequently, correctness and complexity are illustrated in Section 6. Finally, the conclusion is presented in Section 7.

2. The Basic Assumption of MANETs

MANETs have enjoyed an amazing rise in popularity. MANETs requires no infrastructure due to its dynamic nature. Therefore, previous research [2,13,14,15,17] has proposed the concept of a virtual backbone to organize MANETs. To build a virtual backbone, the processors in MANETs can be classified into
gateway or non-gateway processors. In this gateway/non-gateway model, the gateway processors can organize the entire network and forward messages for non-gateway processors. Namely, the source processor can forward its message to one of its adjacent gateway processors when the source processor is not also a gateway processor. Subsequently, the gateway processor acts as the new source processor to route the message to another gateway processor. Finally, the destination gateway processor forwards this message to the destination processor directly.

The main advantage of the gateway/non-gateway model of virtual backbones is that the routing process is simplified to gateway processors. Namely, only gateway processors need to account for the routing table and the search space of routing path is reduced to itself. Furthermore, non-gateway processors can change their status to sleep mode to conserve their battery power. Therefore, the gateway/non-gateway model is being adapted for use as a virtual backbone in MANETs. We review some of the popular virtual backbone construction algorithms as follows:

The Connected Dominating Set (CDS) algorithm was proposed by Wu and Li [17] to build the virtual backbone of MANET. CDS-based routing is related to the concept of dominating in graph theory [13,17]. A subset of vertices of a graph is a dominating set if every vertex not in the subset is adjacent to at least one vertex in the subset. Therefore, the processors of the dominating set should be connected to build the path of communication.

Their algorithm consists of a marking phase and a re-marking phase. The marking phase is a localized algorithm [17] that marks every vertex in a given connected and simple graph G = (V, E). Let \( m(v) \) be a marker for \( v \in V \), which is either T (marked) or F (unmarked). Let \( N(v) = \{u \mid \{v,u\} \in E\} \) represent the open neighboring set of processor \( v \) (i.e., \( v \notin N(v) \)). The \( N[v] = N(v) \cup \{v\} \) represents the closed neighboring set of vertex \( v \) (i.e., \( v \in N[v] \)). The marking phase assigns a marker \( m(v) \) to \( T \) if there are two unconnected neighbors.

After this marking phase, the set \( V' \) of vertices that are marked T in \( V \) is obtained, i.e., \( V' = \{v \mid \forall v \in V, m(v) = T\} \) and \( G' \) is the subgraph induced by all processors with marker T. According to the marking phase, the initial CDS is constructed as the backbone of MANETs. The initial CDS subset consists of those processors having at least two non-adjacent neighbors. Subsequently, the algorithm removes all locally redundant gateway processors from \( V' \) via connectivity degree of gateway processors and id (identifier) during the re-marking phase.

Das et al. [2], proposed another famous distributed algorithm for building a virtual backbone. Their algorithm first finds an approximation to the minimum dominating set by a greedy algorithm and then constructs a spanning forest \( F \) in the second stage. Subsequently, the third stage expands the spanning forest \( F \) into a spanning tree \( T \) and forms a minimum CDS.

Besides, Stojmenovic et al. [15] demonstrated three synchronized distributed constructions of CDS.
Their methods classify the processors in CDS into clusterhead processors and border processors. Processors are divided into clusters with one of them serving as the clusterhead processor in each cluster that can also connect to any of the processors in its cluster directly; however, the clusterhead processors are not adjacent to each other.

In this architecture, the source processor sends the message to destination processor by forwarding its message to its clusterhead processor. Subsequently, the clusterhead processor forwards this message to the border processor. Similarly, the border processor forwards this message to the destination cluster. Eventually, the communication session is established. The communication between any two adjacent clusters has to rely on their common border processors. These methods based on [13] utilize the lowest-ID clustering algorithm and the highest-connectivity clustering algorithm to determine the clusterhead processor.

Based on the descriptions above, the CDS construction algorithm can be used as a virtual backbone in MANETs. The routing information can be saved and the searching space for a route to the gateway processor in this virtual backbone can be reduced. In addition, the gateway processor can manage its group when processors immigrate into or emigrate away from the network at any time.

However, previous researches into BA problem were discussed with respect to a well-defined network, such as fully connected network or broadcast network [1,7]. The traditional protocols cannot adapt to the wireless environment. It is noteworthy to solve the BA problem in MANETs. Therefore, this paper proposes a protocol in MANETs to make all healthy processors reach an agreement using the CDS virtual backbone. In this study, the definitions and assumptions of MANETs are shown as follows:

1. Processors can immigrate into or emigrate away from MANETs due to its dynamic nature.
2. Each processor has only one identifier (id) to distinguish it from others.
3. If the processors immigrate into or emigrate away from MANETs, the gateway processors can detect these conditions.
4. When a processor immigrates into a new region, it needs to send a probe message to the gateway processor of this region to register its id.
5. Each gateway processor knows total number of gateway processors and the number of non-gateway processors in its group.
6. Each non-gateway (gateway) processor cannot know the total number of processors in MANETs.
7. When a gateway processor of a region needs to change into a non-gateway processor, each processor can elect a new gateway processor by the replication technique (Backup or candidate gateway processors) [6,17].

Due to Fischer et al. [5] found that the BA problem could not achieve consensus in an asynchronous network even if only one processor has failed, and the failure is a crash failure. Therefore, the BA problem is considered in a synchronous network in this paper. We also assume that the bounds on the processing and communication delays of healthy components are finite [4]. The parameters in this synchronous
MANETs are assumed as follows:

1. \( G \): The total number of gateway processors in a MANET.
2. \( n \): The total number of non-gateway processors in a MANET.
3. \( n_i \): The total number of processors in group \( i \).
4. \( S \): The source processor \( i \), and \( S \in G \cup n \).
5. \( d_i \): The decision value of gateway processor \( i \).
6. \( \text{Sign}_{sk}(d_i) \): The gateway processor \( i \) signs its decision value \( d_i \) by using its signing key \( sk \).
7. \( v_k \): The vector in processor \( k \).
8. \( v_s \): The initial value of processor \( s \) broadcasts to all other processors.
9. \( \text{val}(\alpha) \): The collected values at vertex \( \alpha \).
10. \( \text{LMAJ} \): The majority value of processor \( i \) collected from the non-gateway processors.
11. \( \text{MAT} \): the gateway processor \( i \) collects all received vectors from other processors.
12. \( \sigma \): The value of dormant faulty processors.
13. \( c \): The connectivity of MANETs. Based on the Menger theorem [3], at least \( c \) disjoint paths need to exist between any pairs of processor A and B when the connectivity of the network is \( c \).
14. \( G_p_a \): The number of gateway processors with an arbitrary fault.
15. \( G_p_d \): The number of gateway processors with a dormant fault.
16. \( n_p_a \): The number of non-gateway processors with an arbitrary fault.
17. \( n_p_d \): The number of non-gateway processors with a dormant fault.
18. \( \Phi \): The default value, and \( \Phi \in \{0, 1\} \).
19. \( \sigma \): The required rounds of message exchange (\( \sigma \leq \left\lfloor (G-1)/3 \right\rfloor + 1 \)).

The Figure 2 shows an example of the network structure of a five-processor MANET. The basic concept of protocol GCAP is shown in the next section.

3 Basic Concepts and Approaches

Initially, we assume the virtual backbone of MANETs is constructed by a CDS construction algorithm and the elected gateway processors of the virtual backbone have higher capability than non-gateway processors in MANETs. Subsequently, the proposed protocol Group Consensus Agreement Protocol (GCAP) is introduced to solve the BA/consensus problem in MANETs. There are four parts of the GCAP: group agreement process, consensus agreement process, broadcasting agreement process, and maintenance function.

The main work of the group agreement process is collecting initial values from non-gateway processors to decide an initial value of a gateway processor. Subsequently, each gateway processor forwards the majority value of received initial values from the non-gateway processors of its region to other gateway
processors in a **consensus agreement process**.

In a **consensus agreement process**, there are the **message exchange phase** and the **decision making phase**. The **message exchange phase** needs to collect enough messages from gateway processors for all healthy gateway processors to reach an agreement. After taking a majority, the gateway processors can send the messages to other gateway processors. In the second phase of a **consensus agreement process**, the **decision making phase**, each healthy gateway processor computes a common value by applying the majority voting function to messages, collected by the **message exchange phase**, to reach an agreement.

However, the gateway processors may have faults, such as arbitrary and dormant faults. Therefore, each gateway processor in MANETs needs to broadcast its package \( \text{Sign}_{id}(d_i) \) to all gateway processors in a **broadcasting agreement process**. This package contains its \( id \) and its decision value obtained from the **decision making phase** of the **consensus agreement process**.

During the **broadcasting agreement process**, each gateway processor has a set of decision values and forwards these to its members in the same region. Subsequently, each healthy non-gateway processor can obtain a common value by taking a majority of these.

The last part, the **maintenance function**, is used to handle the mobility of processors. Conceptually, each processor has unrestricted mobility; it may immigrate into the network or emigrate away from the network at any time. Therefore, the movements of processors must be heeded. Thus, processor movement is discussed in the next section.

Due to the mobility of processors, the number of required rounds of message exchange may not be an inherent value from the outset, but it is expected and required that at most \( f_p + 1 \) \((f_p \leq \lfloor (k-1)/3 \rfloor) \)) rounds of message exchange occur to reach a common value as proposed by Fischer et al. [5], at any time. Due to each processor does not know the movement of other processors in MANETs. For obtaining the expected number of rounds, each gateway processor acts in an important role to manage the movement of non-gateway processors in MANETs. Based on our virtual backbone, only gateway processors need to heed the total number of gateway processors in MANETs to determine the required rounds of message exchange. The main advantage of our virtual backbone is that the movement of non-gateway processors cannot influence on the number of rounds of required message exchange. Before an example is illustrated, the following relevant information is given.

### 3.1 The Number of Required Rounds for GCAP is Predictable in MANETs

In general, the BA/consensus problem needs \( f_p + 1 \) \((f_p \leq \lfloor (k-1)/3 \rfloor) \)) rounds of message exchange to be solved. Fischer et al. [5], and Bar-Noy et al. [1], proved that \( f_p + 1 \) rounds of message exchange are the necessary and sufficient lower bound for agreement problems. However, while processors may migrate at any time, they lack for messages sent from the previous rounds. Based on the CDS virtual backbone, the gateway processors can regulate the movement of non-gateway processors and retransmit messages on
demand. Therefore, the number of required rounds can be predicted in MANETs and follows the result of the protocol proposed by Bar-Noy et al. [1]. For example, assume there are 25 processors in MANETs; 5 gateway processors, and 20 non-gateway processors. In that case the GCAP must execute \( \sigma \left( \lceil (G-1)/3 \rceil + 1 \right) = \lceil (5-1)/3 \rceil + 1 = 2 \) rounds of message exchange. If the total number of the gateway processors in MANETs is changed to 7, the GCAP will need \( \sigma \leq 3 \left( \lceil (7-1)/3 \rceil + 1 = 3 \right) \) rounds of message exchange. Thus, the rounds of message exchange of GCAP conform to the previous results [1].

In general, each processor in MANETs cannot connect to one another directly due to the limitations of transmission power. The non-gateway processors must transmit the messages by forwarding them via gateway processors. Therefore, the gateway processors of our protocol GCAP must collect the messages from their non-gateway processors by means of group agreement process. Consequently, this group agreement process is necessary to suit actual conditions.

Furthermore, another process is needed, broadcasting agreement, to exchange decision values. This is because the gateway processors may contain faults, such as dormant and arbitrary faults. Therefore, each gateway processor in a broadcasting agreement process must send the package \( \text{Sign}_{i}(d_{i}) \) to ensure the healthy processors can reach a common value.

3.2 GCAP can Tolerate \( \left( \lceil (G-1-Gp_{d})/3 \rceil \right) + \left( \lceil (\sum \lceil (n_{i}-n_{p_{d}})/2 \rceil - (Gp_{a}+Gp_{d}) \right) \) Faulty Processors

If the total number of faulty processors does not exceed the limit \( \left( \lceil (G-1-Gp_{d})/3 \rceil \right) + \left(\lceil (\sum \lceil (n_{i}-n_{p_{d}})/2 \rceil - (Gp_{a}+Gp_{d}) \right) \), the healthy processors can reach consensus. An example is shown in Figure 3 in which there are 15 processors, 5 gateway processors and 10 non-gateway processors, in the MANETs. The processors can be divided into the two-layer hierarchy structure, the gateway layer and non-gateway layer are shown in Figure 4. The processors of the gateway layer are elected gateway processors (dominator) by the CDS algorithm, the other processors are non-gateway processors (dominatee). In this hierarchical structure, each non-gateway processor only belongs to one gateway processor. When a non-gateway processor can belong to exceeds one neighboring gateway processor, it can choose the better one, base on factors such as less distance, higher batter power, better load balance, and so on. We assume these criteria can be obtained during the building of a virtual backbone. Based on the reasoning above, the processor \( f \) elects the processor \( b \) as its gateway processor in Figure 4. A region consists of one gateway processor and several non-gateway processors, as seen in region D in Figure 4. Subsequently, GCAP computes the capability of fault tolerance as follows: \( Gp_{a} = \left( \lceil (G-1-Gp_{d})/3 \rceil \right) =1, Gp_{d} = 1, np_{a} = \left(\lceil (\sum \lceil (n_{i}-n_{p_{d}})/2 \rceil - (Gp_{a}+Gp_{d}) \right) =3, \) and \( np_{d} = 0. \)

The GCAP protocol can tolerate four arbitrary faulty processors and one dormant faulty processor. However, our protocol is better suited to the MANETs environment then those of previous research [1,5,7,11,12]. This is because each processor in a wireless network cannot be connected to each other directly. In addition, the number of rounds of message exchange can be reduced according to the number of gateway processors.
3.3 The LMAJ Function, MAT Function, ic-tree(T_i), and VOTE Function as Used by GCAP

At first, the source processor broadcasts its initial value to others and to itself. Each gateway processor must forward this message to its members. Subsequently, each gateway processor must collect the message from non-gateway processors in a group agreement process. Each gateway processor takes a local majority LMAJ(α) on collected values of vertex α and stores it as its new initial value as shown in Figure 5(a). The majority value is replaced by complement of val(α) when the majority value does not exist. Subsequently, the consensus agreement process is to be invoked.

In the consensus agreement process, each gateway processor broadcasts its value to all gateway processors. When a healthy gateway processor receives these values from other gateway processors, it will store them in its vertex V_i (1<i<G). Then, each gateway processor broadcasts its vertex again and constructs MAT_i (1<i<G). The details of MAT_i are shown in Figure 5(b). Subsequently, each gateway processor exchanges its messages repeatedly until round σ. Eventually, a convenient tree structure, ic-tree (a information collection tree structure, T_i) will be constructed by taking the local majority of each row k of MAT_i, for each round as shown in Figure 5(c). The vertices with repeated names of gateway processors will be eliminated from the ic-tree for eliminating the cyclical influences from the faulty processors. The cyclical influences are due to the messages of faulty processors and may be stored repeatedly in the MAT_i, resulting in an incorrect common value caused by taking a simple majority.

In last phase of the consensus agreement process, the decision making phase, the VOTE function is applied to the root of an ic-tree to eliminate the influence of faulty processors as shown in Figure 5(d).

Subsequently, the detail of GCAP will be introduced in the next section.

4 Protocol GCAP

In this section, the protocol GCAP is introduced to solve the BA/consensus problem in MANETs. GCAP can tolerate \( \left\lfloor \frac{(G-1-Gpd)}{3} \right\rfloor + \left\lfloor \frac{\sum (n_i - npd)}{2} \right\rfloor - (Gp_a + Gp_d) \) faulty processors and requires (σ) rounds of message exchange to reach an agreement. Namely, all healthy processors can reach an agreement under a MANET environment where \( \left\lfloor \frac{(G-1-Gpd)}{3} \right\rfloor + \left\lfloor \frac{\sum (n_i - npd)}{2} \right\rfloor - (Gp_a + Gp_d) \) faulty processors exist.

However, processors in MANETs have another serious challenge, low battery power. To save the battery power, the non-gateway processors can enter idle mode. Therefore, only the gateway processors of our protocol need to execute the same protocol simultaneously to reach consensus among all gateway processors.

GCAP has four parts: group agreement process, consensus agreement process, broadcasting agreement process, and maintenance function. The main work of the group agreement process, consensus agreement process, and broadcasting agreement process ensures the healthy processors can achieve a
common value in the MANETs. However, some processors may immigrate into and emigrate away their region due to the dynamic nature of MANETs. Therefore, we need the extra function, maintenance function, to control the mobility of processors. The maintenance function is invoked at any time when the movement of processors occurs. In general, the processors can be elected as gateway processors due to certain critical factors [2,15,15,17], such as low mobility, high battery power, position, and connectivity. Therefore, the gateway processors cannot easily move out of their region and convert into non-gateway processors. When a gateway processor needs to change into a non-gateway processor or move away from its region, all non-gateway processors of this region must re-elect a new gateway processor and redo its work continuously by means of candidate and backup processors. Based on the gateway/non-gateway model, we discuss the mobility of non-gateway processors and assume the hand-off gateway processors are atomic work. Moreover, the GCAP protocol is described as Figure 6.

✧ The group agreement process: The main advantage of virtual backbone mode is that the routing process is simplified among gateway processors. The gateway processors can forward messages to other regions for non-gateway processors and reduce the complexity of route searching in MAENT[2,8,14]. Therefore, the gateway processors in the group agreement process need to gather the messages of group and decide the initial value of gateway processors.

✧ The consensus agreement process: The consensus agreement process consists of two phases, the message exchange and decision-making phase; the descriptions of which are shown as follows:
  ➢ Message exchange phase: The message exchange phase of the proposed GCAP needs to collect σ rounds exchanged messages and store the received messages in the processors’ matrices. The main objective of the message exchange phase is collecting enough messages to eliminate the influence of faulty processors.
  ➢ Decision making phase: Upon completion of the message exchange phase, each gateway processor has already collected enough messages to eliminate the influence of faulty processors. Then, the function VOTE is used to take a decision value VOTE(s) of the root s of each healthy gateway processor’s ic-tree.

✧ The broadcasting agreement process: Each gateway processor in broadcasting agreement process needs to broadcast its signature $Sign_{id}(d_i)$ of the decision value $d_i$ to all non-gateway processors. After receiving $Sign_{id}(d_i)$ and $d_i$, each non-gateway processor verifies the signature by using the corresponding verification key. Eventually, each healthy non-gateway processor can decide a common value by using the function LMAJ.

✧ Maintenance function: In general, each processor in MANETs may immigrate into or emigrate away its original region at any time. Therefore, each gateway processor acts an important role in MANETs to manage the movement of non-gateway processors due to the gateway/non-gateway model. The mobility of processors needs only to be discussed with respect to the consensus agreement process. This is because
the mobility of processors cannot influence in any way the group agreement process due to the fact that messages are not being sent to other gateway processor during this process. In addition, the mobility of processors cannot influence the decision value in the broadcasting agreement process. Therefore, each processor only needs to re-collect or delete the messages in its group when the movement of processors occurs during group agreement process and broadcasting agreement process. Based on the reasons above, the mobility of processors is the focus of the consensus agreement process and is divided into three states, before the message exchange state, between the message exchange and decision making state, and after the decision making state. The details of this procedure in GCAP are showing as Figure 7.

The most serious effect of these states relates to the between message exchange and decision making state. The main of work of this state is exchanging messages and preparing to make a decision; thus the fate of new message may be changed. When the new processor immigrates into a new region, the new processor must broadcast the request for initial value $v_i$ via other processors. The gateway processor must calculate a majority value from this new initial value and previous values. Subsequently, the gateway processor must decide whether it needs to resend the messages to other gateway processors or to do nothing. When a processor emigrates away its region, the gateway processor just needs to delete the processor from its membership and ignore its message.

5 Example of Execution

In this section, two examples are shown to illustrate the GCAP protocol. The first example illustrates how GCAP facilitates the healthy processors achieving agreement when a new processor immigrates into a new region. The processor moving away from its region will be shown as the second example.

In Figure 8, there are 15 processors in the original MANETs. In this paper, the CDS protocol [13,14,5,17] is used as virtual backbone to elect 5 gateway processors and 10 non-gateway processors in the MANETs. Basically, each gateway processor can know the total number of gateway processors in MANETs, but the non-gateway processors cannot know this. The outer communication must be forwarded by gateway processors. Therefore, the protocol GCAP requires $\sigma = \lceil (5-1)/3 \rceil + 1 = 2$ rounds to exchange messages. Each healthy gateway processor reaches a common value after executing GCAP, if the total number of faulty gateway processors do not exceed $\lceil (G-1-G indis) / 3 \rceil$ at any time.

Figure 8 shows the complete steps needed to execute GCAP with a healthy processor $a$ when other processors have immigrated into the MANETs. The steps required for other healthy processors are the same as those for processor $a$. However, we do not discuss at which value the faulty processors agree on.

Based on the CDS protocol [13,14,15,17], the processors in Figure 8 can be divided into gateway processors and non-gateway processors. Each gateway processor can manage non-gateway processors when the non-gateway processors are located in its region. According to the ability of gateway processors,
a two-layer hierarchy structure can be built, such as in Figure 4. Figure 8(a), shows the processors $h$, $b$, and $k$ are arbitrary faulty processors and processor $d$ and $n$ are dormant faulty processors, respectively. In the beginning of the protocol, we assume $h$ as the source processor $h$; it broadcasts an initial value to all gateway processors. Unfortunately, the source processor $h$ is an arbitrary processor; it may broadcast different values to all gateway processors. Figure 8(b) assumes the source processor $h$ sends values to regions A, B, C, D, and E as 1, 0, 0, 1, and 0, respectively. Subsequently, each gateway processor uses the LMAJ function to obtain a majority value and store it to its vector. The vector of each gateway processor is shown as Figure 8(c). After the group agreement process, each gateway processor has an initial value to broadcast to other gateway processors by means of the consensus agreement process.

In consensus agreement process, each gateway processor needs to exchange its value with other gateway processors. The result of healthy gateway processor $a$ executing the first round of consensus agreement process are shown as Figure 8(d). Each healthy gateway processor can detect whether processors are dormant faulty processors or not. Therefore, all gateway processors can identify gateway processor $d$ as a dormant faulty processor. The details of all results in first round are shown as Figure 8(e). During the consensus agreement process, a processor $p$ immigrates into region A as shown in Figure 8(f). As it does not participate into the group agreement process, it does not obtain any value. In Figure 8(g), each processor in region A must send the value that is received from last round to the new processor. Subsequently, the new processor $p$ takes the majority value of the received values and sends it to gateway processor $a$. If the new majority value is equal to previous majority values, then gateway processor $a$ does nothing. Otherwise, gateway processor $a$ changes its value in the $(\sigma-1)$th of its vector and resends to others.

Continuously, each gateway processor broadcasts its value to other gateway processors in the second round. Due to the protocol GCAP only requiring $\sigma = \lceil(5-1)/3\rceil+1 = 2$ rounds of message exchange, thus the message exchange is stopped. Figure 8(h) shows the results of gateway processor $a$ in the message exchange phase of consensus agreement process. The common result of gateway processors is shown as Figure 8(i), and it is equal to that of other healthy gateway processors. Subsequently, we take the local majority of each MAT$_i$ of each round to construct a corresponding ic-tree $T_i$ and use the function VOTE to the root of each processor’s ic-tree $T_i$ in Figure 8(j) and 8(k) to make a decision value. Eventually, the common value (“0”) of gateway processors can be reached if the number of faulty gateway processors is less than or equal to $\lceil(G-1-Gpd)/3\rceil$.

During next process, each gateway processor broadcasts its signature $\text{Sign}_{sk}(d_i)$ on the decision value $d_i$ to all non-gateway processors, as shown in Figure 8(l). Then, each non-gateway processor can use the corresponding verification key to verify the signature and make a common decision by using the LMAJ function if the number of faulty non-gateway processors in a group is less than or equal to $\lceil(n_r-1-np_d)/2\rceil$. Therefore, the healthy processors in protocol GCAP can reach consensus in a MANET en-
vironment where \( \left\lfloor \frac{(G-1-Gp_d)}{3} \right\rfloor + \left\lceil \left( \sum (n_i-1-np_d)/2 \right) - (Gp_a+Gp_d) \right\rceil \) faulty processors exist.

Furthermore, the processors may emigrate out of MANETs. In our virtual backbone structure, mobility of processors can be solved easily. This is because gateway processors can control and manage the activity of non-gateway processors. Figure 9 shows how GCAP can make each healthy processor reach an agreement in MANETs when processors emigrate out of MANETs. There are 18 processors containing 5 gateway processors and 13 non-gateway processors in MANETs; such an environment requires \( \sigma = \left\lceil \frac{(5-1)}{3} \right\rceil + 1 = 2 \) rounds during the message exchange phase. In Figure 9, we show all necessary steps to execute GCAP on the healthy gateway processor \( e \) when processors emigrate out of the MANETs.

We assume the source processor \( h \) as an arbitrarily faulty processor, and the original environment is shown as Figure 9(a). The source processor \( h \) first broadcasts its initial value to all processors. Due to the source processor being an arbitrary faulty processor, each processor may obtain a different initial value. In Figure 9(b), the source processor \( h \) sends to all gateway processors \( a, b, c, d, \) and \( e \) being 1, 0, 0, 1, and 0, respectively. Then, each gateway processor forwards this value to its members. Finally, each gateway processor can obtain its initial value during group agreement process, as shown in Figure 9(b).

During consensus agreement process, each gateway processor broadcasts its initial value to other gateway processors. Figure 9(c) only presents vector \( V_e \) of healthy gateway processor \( e \), the other healthy gateway processors execute the same procedures in our protocol. However, the processor \( q \) leaves region \( E \) in Figure 9(d). Gateway processor \( e \) must eliminate its initial value from its vector and re-compute its majority value. In general, the influence cannot exceed \( \left\lfloor \frac{(n-1)}{2} \right\rfloor \), thus the majority value is still the same. In Figure 9(e), the decision value of the healthy gateway processor \( e \) is 0 and the decision value of other healthy gateway processors are the same. The subsequent processes are the same as in the example above.

6 The Correctness and Complexity of GCAP

The following proofs for the agreement and validity are given in this section that the BA problem needs to meet. The lemmas and theorems are used to prove the correctness and complexity of GCAP.

6.1 Correctness of GCAP

In this section, we prove the correctness of GCAP if the number of faulty processors does not exceed \( \left\lfloor \frac{(G-1-Gp_d)}{3} \right\rfloor + \left\lceil \left( \sum (n_i-1-np_d)/2 \right) - (Gp_a+Gp_d) \right\rceil \) at any time. To achieve agreement, each healthy processor should be insulated from the influence of a faulty processor. If the influence of a faulty processor can be removed from the decision making phase, then the messages collected by each healthy processor are free from the influence of that processor, and agreement is reached.

To prove the correctness of our protocol, a tree structure, ic-tree\((T_i)\) is used to illustrate it. The main
feature of ic-tree, to store the received messages and to eliminate the cyclical influence of faulty processors by eliminating the repeated names, is shown as Figure 5(c). The function VOTE also must obtain a common value from the ic-tree in the consensus agreement process. Therefore, the ic-tree structure in this paper proves the correctness of our protocol.

At first, this paper defined a vertex α as common [1] if each healthy processor computes the same value for α. In other words, the value stored in vertex α of each healthy processor’s ic-tree is common to all. Once each healthy processor has a common initial value from the source processor in the root of its ic-tree, an agreement is reached since the root is common to all. Thus, the agreement (BA₁) and (BA₂), can be rewritten as:

(BA₁'): Root $s$ is common, and
(BA₂'): $\text{VOTE}(s) = v_s$ for each healthy processor, if the source processor is healthy.

The term common frontier [1] is defined as follows: “If every root-to-leaf path of the ic-tree contains a common vertex, the collection of the common vertices forms a common frontier.” In other words, every healthy processor collects the same messages within a common frontier if a common frontier exists in that healthy processor’s ic-tree. Subsequently, using the same voting function VOTE to compute the root value of the ic-tree, every healthy processor can obtain the same root value because they utilize the same input and the same computing function. Due to the above concepts can be used to prove the correctness of consensus/BA problem, thus the GCAP will follow this way to prove the correctness.

Before proving the correctness of GCAP, the term correct vertex is defined as:

- Correct vertex: Vertex $\alpha_i$ of a tree is a correct vertex if processor $i$ is healthy. In other words, a correct vertex is a place to store values received from healthy processors.
- True value: For a correct vertex $\alpha_i$ in the tree of a healthy processor $i$, $\text{val}(\alpha_i)$ is the true value of vertex $\alpha_i$. Namely, the stored value is called the true value.

By the definition of a correct vertex, the stored value in the ic-tree is received from healthy processor, and a healthy processor always transmits the same value to other processors. Furthermore, repeated vertices in ic-trees are deleted; thus, correct vertices in ic-trees are common. Based on the definition of correct vertex, a common frontier does exist in the ic-tree. Namely, the root can be proven to be a common vertex (BA₁') due to the existence of a common frontier, regardless of the correctness of a source processor. Based on reasoning above, an agreement among the root values is reached.

Subsequently, we will check the condition of (BA₂'). Based on (BA₂'), we know that when the source processor fails, the (BA₂') is true. This is because the propositional logic $P \rightarrow Q$ indicates (NOT(P) OR Q), then (NOT(P) OR Q) or (P $\rightarrow$ Q) is true when P is false; where P implies “the source processor is healthy” and (P $\rightarrow$ Q) implies BA₂'. Conversely, root $s$ is a correct vertex by the definition of a correct vertex if the source processor is healthy. If all correct vertices’ true values can be computed by GCAP, then the true value of the root on ic-tree can be computed because the root is a correct vertex. By defini-
tion, the true value of the root is the initial value of the source processor if the source processor is healthy. Namely, each healthy processor’s root value is the initial value of the source processor; if the source processor is healthy, then BA2’ is true when the source processor is healthy. In short, the BA1’ and BA2’ are both true whether the source processor is healthy or fails, the consensus/BA problem is solved.

**Lemma 1.** Messages sent through dormant faulty components can be detected by healthy destination processors.

*Proof.* The healthy destination processor can detect the message(s) from dormant faulty components if the protocol appropriately encodes a transmitted message by using either the Non-Return-to-Zero code or the Manchester code [6] before transmission. □

**Theorem 1.** The destination processor can receive messages free of influence from faulty components existing between the sending processor via virtual backbone in each round if \( c > G_{pa} + G_{pd} \).

*Proof.* By Lemma 1, we can remove the influence of dormant faulty components between any pair of processors in each round of message exchange; and we can rule out the influence of the arbitrarily faulty components between any pairs of processors in each round of message exchange if \( c > G_{pa} + G_{pd} \). This is because the healthy sending processor sends \( c \) copies of a message to healthy destination processors. In the worst case, a healthy destination processor can receive \( c - G_{pd} \) messages transmitted by the healthy sending processor because dormant faulty components can be detected [6] and \( c - G_{pd} > G_{pa} \). Therefore, a healthy destination processor can decide which the correct messages are by taking into account the majority value. □

**Lemma 2.** A healthy destination processor can detect a dormant faulty sending processor by means of the forwarding technique uses in the gateway/non-gateway model.

*Proof.* If the number of value \( \lambda \) is greater than or equal to \( c - \lfloor (G-1)/3 \rfloor \), then the sending processor contains a dormant fault. This is because there are at most \( \lfloor (G-1)/3 \rfloor \) arbitrarily faulty components in the network, hence there are at most \( \lfloor (G-1)/3 \rfloor \) non-\( \lambda \) value in the vector \( V_i \). □

**Theorem 2.** A healthy processor can detect the dormant faulty processor in the network.

*Proof.* In the protocol GCAP, there are \( \sigma \) rounds of message exchange in consensus agreement, where \( G_{pa} \leq \lfloor (G-1)/3 \rfloor \) and \( G \geq 3 \), so there are at least two rounds of message exchange in the message exchange phase. Each healthy processor can receive a message from the source processor in the first round of message exchange and receives other processors’ messages in the second round of message exchange. Each processor can receive all other processors’ messages in the network after two rounds of message exchange. According to the Lemma 2, each healthy processor can detect the dormant faulty processor in the network. □

**Lemma 3.** All correct vertices of an ic-tree are common.

*Proof.* After reorganization, no repeated vertices remain in an ic-tree. At the level of \( G_{pa} + 1 \) or above, the
correct vertex \(\alpha\) have at least \(2Gpa+1\) children, out of which at least \(Gpa+1\) children are correct. The true value of these \(Gpa+1\) correct vertices is common, and the majority value of vertex \(\alpha\) is common. The correct vertex \(\alpha\) is common in the ic-tree if the level of \(\alpha\) is less than \(Gpa+1\). As a result, all correct vertices of the ic-tree are common. \(\blacksquare\)

**Lemma 4.** A common frontier exists in the ic-tree.

**Proof.** There are \(Gpa+1\) vertices along each root-to-leaf path of an ic-tree in which the root is labeled by the source name, and the other vertices are labeled by a sequence of group names. Since at most \(Gpa \leq \left\lfloor \frac{(G-1)}{3} \right\rfloor\) processors can be faulty, at least one vertex is correct along each root-to-leaf path of the ic-tree. By Lemma 3, the correct vertex is common, and a common frontier exists in each healthy processor’s ic-tree. \(\blacksquare\)

**Lemma 5.** Let \(\alpha\) be a vertex, where \(\alpha\) is common if there is a common frontier in the subtree rooted at \(\alpha\).

**Proof.** If the height of \(\alpha\) is 0 and a common frontier (\(\alpha\) itself) exists, and then \(\alpha\) is common. If the height of \(\alpha\) is \(\sigma\), the children of \(\alpha\) are all in common by using the induction hypothesis with the height of the children at \(\sigma-1\), then the vertex \(\alpha\) is common.

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

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

**Proof.** By Lemmas 3, 4, 5, and Corollary 1, the theorem is proven.

**Theorem 4.** Protocol GCAP solves the consensus/BA problem in MANETs.

**Proof.** To prove the theorem, one must show that GCAP meets the constraints (BA1’) and (BA2’). (BA1’): Root \(s\) is common. By Theorem 3, (Agreement’) is satisfied. (BA2’): VOTE(\(s\))=\(v\) for all healthy processors, if the initial value of the source is \(v_s\), say \(v=v_s\). Since most of processors are healthy, they transmit the message to all others. As a result, each of the correct vertices of the ic-tree is common (Lemma 3), and its true value is \(v\). By Theorem 3, this root is common. The computed value VOTE(\(s\)) = \(v\) is stored in the root for all healthy processors. (Validity’) is satisfied. \(\blacksquare\)

### 6.2 Complexity of GCAP

The complexity of GCAP is judged in terms of: (1) the minimal number of rounds, (2) the maximum number of allowable faulty components, and (3) the number of exchanged messages.

**Theorem 5.** GCAP requires \(\sigma\) rounds to solve the consensus/BA problem by dual failure mode (containing arbitrary and dormant faults) in MANETs if \(G>\left\lfloor \frac{(G-1)}{3} \right\rfloor + 2\ Gpa + Gp_d\) and \(c>2\ Gpa + Gp_d\) where \(Gpa \leq \left\lfloor \frac{(G-1)}{3} \right\rfloor\) and \(Gpa+1\) are the minimum number of rounds of message exchange.

**Proof.** Due to the fact that message passing is required in the message exchange phase, it is very time-consuming. Fischer and Lynch [5] pointed out that \(f+p+1\) \((f_p \leq \left\lfloor \frac{(k-1)}{3} \right\rfloor)\) rounds are the minimum number of rounds needed to obtain enough messages to achieve consensus/BA. The unit utilized by Fischer and Lynch [5] is the processor, so the number of required rounds of message exchange in MA-
NETs is $\sigma \leq \lfloor (G-1)/3 \rfloor + 1$. Thus, GCAP requires $\sigma$ rounds and this number is the minimum. □

**Theorem 6.** The total number of allowable faulty components by GCAP is $G_p + n_p$ arbitrarily faulty processors and $G_d + n_d$ dormant faulty processors, where $G > [(G-1)/3] + 2G_p + G_d$, $n > 2n_p + n_d$, and $c > 2G_p + G_d$.

*Proof.* According to the constraints of the BA problem for processors which was proposed by Siu et al. [12]. In this study, we use gateway processors to substitute the unit of Siu et al., thus the constraints in our protocol are $G > [(G-1)/3] + 2G_p + G_d$, $c > 2G_p + G_d$. Furthermore, our protocol can tolerate other faulty non-gateway processors in MANETs. The number of allowable faulty non-gateway processors is $(\sum (n_i-1-n_d)/2)$ if $n > 2n_p + n_d$. This is because each healthy non-gateway processor in each group can send $\lfloor (n-n_d)/2 \rfloor$ messages to healthy gateway processors. In worst case, the gateway processor may receive $(\sum (n_i-1-n_d)/2)$ wrong messages from arbitrary non-gateway processors. It is clear that $\lfloor (n-n_d)/2 \rfloor$ is larger than $(\sum (n_i-1-n_d)/2)$ even if dormant processors do exist. So the total number of allowable faulty components by GCAP is $G_p + n_p$ arbitrarily faulty processors and $G_d + n_d$ dormant faulty processors, which reach a maximum if $G > [(G-1)/3] + 2G_p + G_d$, $n > 2n_p + n_d$, and $c > 2G_p + G_d$. □

**Theorem 7.** The number of message exchanges of GCAP is minimal and the computational complexity of each round is $O(n+G)^2$.

*Proof.* In group agreement process of protocol GCAP, each gateway processor must collect messages from its members. We assume the total number of non-gateway processors as $n$. Thus, the group agreement process needs to exchange messages $n$ times. Subsequently, the number of message exchanges in consensus agreement processes is $G(G-1)$ for each round; $G$ is total number of gateway processors. Due to the protocol GCAP needing $\sigma$ rounds of message exchange to collect enough messages, the total number of exchanged messages is $[\sigma G(G-1)]$. In addition, the broadcasting agreement process and maintenance agreement process only need to exchange messages $(n+G)$ times, respectively. Therefore, the total number of exchanged messages of GCAP is $[\sigma G(G-1)] + 2G + 3n$ (having a maintenance policy). However, the total number of processors is $(n+G)$ when the topology does not follow the gateway/non-gateway model. Fischer et al. [5] pointed out that $f_p + 1$ ($f_p \leq \lfloor (k-1)/3 \rfloor$) rounds are the minimum number of rounds needed to get sufficient messages to achieve consensus/BA. Therefore, the minimum number of rounds is $\sigma_1 \leq \lfloor (p-1)/3 \rfloor + 1$ ($p=n+G, \sigma_1>0$) when the topology does not follow the gateway/non-gateway model. The total number of messages is $[\sigma_1 p(p-1)] + p$ (having a maintenance policy). Therefore, the number of rounds and the total number of message exchanges are minimal due to $\sigma_1 > \sigma$ and $p > G$. Similarly, the computational complexity of GCAP is $O(n+G)^2$. This is because the main costs surround message exchange in consensus agreement process. The computational complexity of the other processes is constant. □

17
As a result, the GCAP utilizes the minimum number of rounds and tolerates the maximum number of faulty processors of all systems in order to ensure that all healthy processors reach a common agreement. The superiority of the protocol is thus proven.

7 Conclusion

In this study, the consensus/BA problem in MANETs is revisited with respect to dual failure mode in fallible processors. The processors in MANETs may immigrate into or emigrate away their region due to the dynamism of their nature. Therefore, previous attempts [5,7,12,18] cannot adapt to this feature of MANETs. Furthermore, a virtual backbone to organize the MANETs in the absence of a fixed infrastructure is necessary. Therefore, our GCAP protocol uses the gateway/non-gateway model of CDS as a virtual backbone to organize the MANETs. In the CDS virtual backbone, only the gateway processor needs to maintain the routing table, and the search space is reduced to the scope of the virtual backbone. Each non-gateway processor can save energy in this virtual backbone of MANET.

The rounds of message exchange in our protocol are more efficient than those of previous efforts [1,7]. For example, if there are 100 processors are divided into 20 regions, each region has one gateway processor and four non-gateway processors \((n > 2)\). The protocols proposed by Lamport et al. [7], Meyer et al. [9], Siu et al. [12], and Wang et al. [16] can meet the requirements of consensus/BA, but all of them need \(34 \left\lceil \frac{(100-1)}{3} \right\rceil +1\) rounds of message exchange to reach consensus. However, there are 7 rounds of message exchange in our GCAP protocol \(\left\lceil \frac{(20-1)}{3} \right\rceil +1\). If the capabilities of gateway processors are improved, they can handle more non-gateway processors. In that case, the number of rounds of message exchange can be decreased rapidly. Therefore, our protocol is more efficient than the others [7,12,9,16]. The comparison of various protocols used in different environment is shown in Table 1.

In the future, we will consider the faults found in transmission links in MANETs. This is because that each processor in MANETs can communicate with one another directly when they exist within their transmission range. A processor sends its message to a destination processor located outside its wireless transmission range by forwarding it via another mobile processor. Therefore, the failure of the transmission link may occur at any time.

References


Mao-Lun Chiang received his M.S. from the Department of Information Management at Chaoyang University of Technology, Taiwan.
Currently, he is pursuing his Ph.D. in Department of Computer Science at National Chung-Hsing University, Taiwan. His current research interests include distributed data processing, and fault tolerant computing.

Shu-Ching Wang received her B.S. in Computer Science from Feng-Chia University, her M.S. in Electrical Engineering from National Chen-Kung University, and her Ph.D. in Information Engineering from National Chiao-Tung University, Taiwan.

Currently, she is a Professor at the Graduate Institute of Informatics, Chaoyang University of Technology, Taichung County, Taiwan. Her current research interests include distributed data processing, parallel processing, mobile computing, algorithm analysis and design, and fault tolerant computing.

Lin-Yu Tseng received his B.S. in Mathematics from the National Taiwan University, his M.S. in Computer Science from National Chiao-Tung University, and his Ph.D. in Computer Science from National Tsing-Hua University, Taiwan.

Currently, he is a Professor with the Department of Computer Science at the National Chung-Hsing University, Taichung County, Taiwan. His current research interests include design and analysis of algorithms, pattern recognition, and genetic algorithms.
Figure 1. The topology of MANETs by unit-disk graphs.

Figure 2. A region of five-processor MANET environment.

Figure 3. A 15-processor MANET.

Figure 4. The two-layer hierarchy structure of Figure 3
The function **LMAJ**($\alpha$)

<table>
<thead>
<tr>
<th>LMAJ($\alpha$)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The majority value in the set of ${\text{val}(\alpha_j)\mid 1 \leq j \leq (G,n)}$, if such a majority value exists.</td>
<td></td>
</tr>
<tr>
<td>2. The complement value of $\text{val}(\alpha)$, denoted as $\neg\text{val}(\alpha)$, is chosen, otherwise.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5(a). The function LMAJ.

The function **MAT**$_{i}$ (for gateway processor $i$ with initial value $v_{i}$, $1 \leq i \leq N$)

1. Receive the initial value $v_{j}$ from processor $j$, for $1 \leq j \leq N$ and $j \neq i$.
2. Construct the vector $V_{i} = \{v_{1}, v_{2}, ..., v_{N} \}$, $1 \leq i \leq N$ and $j \neq i$. If a dormant processor $k$ ($1 \leq k \leq N$) was found, then $v_{k} = \lambda$.
3. Broadcast $V_{i}$ to all processors, and receive column vector $V_{j}$ from processor $j$, $1 \leq i \leq N$.
4. Construct a MAT$_{i}$ (Setting the vector $v_{j}$ in column $j$, for $1 \leq j \leq N$). If a dormant processor $k$ ($1 \leq k \leq N$) was found, then $V_{k} = \lambda, \lambda, ..., \lambda$.

Figure 5(b). The function MAT$_{i}$ on processor $i$.

**Figure 5(c). The ic-tree ($T_i$)**
\[
\text{VOTE}(\alpha) =
\begin{align*}
1. & \text{ val}(\alpha), \text{ if } \alpha \text{ is a leaf.} \\
2. & \text{ The majority value in the set of } \{\text{VOTE}(\alpha_i) \mid 1 \leq i \leq G, \text{ and vertex } \alpha_i \text{ is a child of vertex } \alpha \}, \text{ if such majority values exists.} \\
3. & \text{ A default value } \phi \text{ is chosen, otherwise.}
\end{align*}
\]

Figure 5(d). The function VOTE.

Figure 5. The functions of protocol GCAP.

Figure 7. The procedure of GCAP.

Figure 8(a). The original MANETs environment.
Figure 8(b). The first round in the group agreement process.

Figure 8(c). The results of the group agreement process.

Figure 8(d). The result of processor \( a \) executing the first round of message exchange.

<table>
<thead>
<tr>
<th>( V_a )</th>
<th>( V_b )</th>
<th>( V_c )</th>
<th>( V_d )</th>
<th>( V_e )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>( \lambda )</td>
<td>( \lambda )</td>
<td>( \lambda )</td>
<td>( \lambda )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 8(e). The vector received from gateway processors in the first round of consensus agreement process.
Figure 8(f). Processor $p$ immigrates into region A between the message exchange and decision marking states of the consensus agreement process.

Figure 8(g). Processor $p$ takes the majority value in maintenance process.

Figure 8(h). The result of processor $a$ executing the second round of the message exchange phase in consensus agreement process.
Figure 8(i). The result of all gateway processors executing the second round of the message exchange phase in consensus agreement process.

<table>
<thead>
<tr>
<th>$V_a$</th>
<th>$V_b$</th>
<th>$V_c$</th>
<th>$V_d$</th>
<th>$V_e$</th>
<th>$V_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 8(j). The ic-tree of gateway processor $a$.

<table>
<thead>
<tr>
<th>Source</th>
<th>$val(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$sa$</td>
<td>10111</td>
</tr>
<tr>
<td>$sb$</td>
<td>01000</td>
</tr>
<tr>
<td>$sc$</td>
<td>01000</td>
</tr>
<tr>
<td>$sd$</td>
<td>10111</td>
</tr>
<tr>
<td>$se$</td>
<td>01111</td>
</tr>
</tbody>
</table>

Figure 8(k). The result of gateway processor $a$ in the decision making phase of consensus agreement process.

- $10111 \rightarrow 1$
- $01001 \rightarrow 0$
- $VOTE_a: 01000 \rightarrow 0 \rightarrow 0$
- $20311 \rightarrow 3$
- $01000 \rightarrow 0$

Figure 8(l). Each gateway broadcasts its $Sign_a(d_i)$ and $d_i$ to all processors during broadcasting agreement process.

Figure 8. The GCAP procedure when processors immigrate into the MANETs.
Figure 9(a). The original MANETs environment.

Figure 9(b). The vector of each gateway processor during the first round of the group agreement process.

Figure 9(c). The result of processor e executing the first round of message exchange phase in consensus agreement process.
Figure 9(d). The processor $q$ leaves region E between the message exchange and decision marking states.

$$\begin{align*}
10111 & \rightarrow 1 \\
01000 & \rightarrow 0 \\
\text{VOTE}_e: & \quad 01000 \rightarrow 0 \rightarrow 0
\end{align*}$$

Figure 9(e). The decision value of gateway processor $e$ in the decision making phase of consensus agreement process.

Figure 9. The procedures of GCAP when processors leave the MANETs.

Table 1. The comparison of various protocols

<table>
<thead>
<tr>
<th></th>
<th>Network topology</th>
<th>Processor fault</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PCN</td>
<td>BCN</td>
</tr>
<tr>
<td>Dolev et al [4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lamport et al [7]</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Meyer and Pradhan [9]</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wang et al. [16]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wang and Yan [18]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCAP</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

28