Article

Reaching Reliable and Trustworthy Agreement in a Multiple Damage Vehicular Ad Hoc Network

Shu-Ching Wang [1], Ya-Jung Lin [1],Yao-Te Tsai [2,*] and Kuo-Qin Yan [1,*]

[1] Chaoyang University of Technology, Taiwan; scwang@cyut.edu.tw
[2] Feng Chia University, Taiwan; yttsaifc@gmail.com
* Correspondence: yttsaifc@gmail.com; kqyan@cyut.edu.tw; Tel.: +886-4-23323000

Abstract: The era of Internet of Things (IoT) has begun to evolve and with this the devices around us are getting more and more connected. Vehicular Ad-hoc NETworks (VANETs) is one of the applications of IoT. VANET allow vehicles within these networks to communicate effectively with each another. VANETs can provide an extensive range of applications that support and enhance passenger safety and comfort. It is important that VANETs are applied within a safe and reliable network topology; however, the challenging nature of reaching reliable and trustworthy agreement in such distributed systems is one of the most important issues in designing a fault-tolerant system. Therefore, protocols are required so that systems can still be correctly executed, reaching agreement on the same values in a distributed system, even if certain components in the system fail. In this study, the agreement problem is revisited in a VANET with multiple damages. The proposed protocol allows all fault-free nodes (vehicles) to reach agreement with minimal rounds of message exchanges, and tolerates the maximal number of allowable faulty components in the VANET.

Keywords: Internet of Things; Agreement; Intelligent Transportation Systems

1. Introduction

Recently, VANET becomes increasingly popular in many countries. It is an important element of the Intelligent Transportation Systems (ITSs) [1]. The basic application of a VANET is to allow arbitrary vehicles to broadcast safety messages to other vehicles and nearby stationary roadside units (RSUs) [2]. Other vehicles may adjust their travel routes based on information received, and RSUs may instruct a traffic control center to adjust traffic lights in order to avoid possible traffic congestion.

VANET's consist of vehicles and roadside equipment that are able to communicate between each other by wireless and multi-hop communication. VANET's are prone to interference and propagation issues, as well as different types of attacks and intrusions that can harm ITS services [3]. These networks characteristically have highly mobile nodes (vehicles), rapid and significant network topology changes, wireless links subject to interference and fading due to multipath propagation [4]. To achieve high reliability in a VANET, a mechanism that allows a set of nodes to reach a common agreement, even in the presence of faulty nodes, is needed.

It is crucial that a VANET needs a reliable network topology that provides a good environment for data transmission, and this has become an important research topic. Therefore, the achievement of reliable and trustworthy agreement in the VANET is one of the most important components to be considered in designing a fault-tolerant system. In previous works, it was concluded that the ability to reach a common agreement among fault-free nodes in order to cope with influences from faulty components was a crucial component of any fault-tolerant system.

Such an agreement problem was first introduced by Pease et al. in 1980 [5], and the problem has been since been called the Byzantine Agreement (BA) problem [6]. The classical BA problem is considered for a synchronous fixed network in which the bounds on processing and communication delays of fault-free components are fine [7]. In 1985, Fischer et al. indicated that agreement in an

 doi:10.20944/preprints201801.0203.v1

asynchronous network is impossible if just a single faulty node crashes [7]. However, most VANETs are asynchronous networks. Therefore, the previous results for the BA problem cannot solve the BA problem in asynchronous VANETs, if there is just a single faulty node in the VANET. To cope with asynchrony, Chandra and Toueg proposed a failure detector in 1996 [8]. This failure detector is used to detect the non-responsive nodes in asynchronous VANETs.

2. Related Works

2.1 BA problem

BA problem is a fundamental problem when implementing fault-tolerant distributed services. In many applications, a fault-free node in a distributed system should be able to reach a common agreement even if certain nodes in the distributed system fail. With the agreement, many applications can be achieved, such as a task of locating the whereabouts of a replicated file in a distributed environment [9], a two-phase commitment could be made in a distributed database system [10] and a landing task controlled by the nodes in a flight control system [11]. Such a unanimous problem is called the Byzantine Agreement (BA) problem and is first studied by Lamport et al. [6]. The definitions of the BA problem are:

1) There are n nodes ($n \geq 4$), of which at most one-third of the total number of nodes could fail without breaking down a workable network.
2) The nodes communicate with each other through message exchange in a fully connected network.
3) The message's sender is always identifiable by the receiver.
4) A node is chosen as a source, and its initial value v_s is transmitted to other nodes for executing the protocol.

In a distributed system, the components of network may not always work well. A node is said to be fault-free if it follows protocol specifications during the execution of a protocol; otherwise, the node is said to be faulty. The symptoms of node failure can be classified into dormant fault and malicious fault (also called as the Byzantine fault) [12]. A dormant faulty node always can be identified by the receiver if the transmitted message was encoded appropriately (i.e. by NRZ-code, Manchester code) before transmission [13]. The behavior of a malicious faulty node is unpredictable and arbitrary. The message transmitted by a malicious faulty node is random or arbitrary. It is the most damaging failure type and causes the worst problem. In this study, the agreement problem is revisited to enlarge the fault tolerant capability by allowing hybrid faulty nodes (both dormant fault and malicious fault) exist in the VAMET.

2.2 VANET architecture

VANET are expected to support a large spectrum of mobile distributed applications that range from traffic alert dissemination and dynamic route planning to context-aware advertisement and information sharing. VANET architecture spans across various hardware and software components. Two primary types of devices used in VANETs are on-board units (OBUs) and RSUs. OBUs are mounted on vehicles, and RSUs are deployed along roadsides as infrastructure [14]. The communication between OBUs, or between an OBU and an RSU is achieved through a wireless medium.

There are a large number of nodes that participate in VANETs and they have high mobility. The mobile VANET differs from a fully connected network or broadcast network in that the nodes in a mobile environment have high mobility. These nodes may join or leave the network at any time. How nodes reach agreement in the VANET is critical to network reliability. The network technology continues to grow very rapidly, and applications in mobile VANETs have reached astonishing achievements in the last year. It is thus very important to solve the BA problem in mobile VANETs. Thus, this research will focus on the VANET, and propose a protocol that will allow all the network's fault-free nodes to reach an agreement. In this study, the BA problem is considered in an asynchronous VANET with multiple damage faulty nodes and the definitions and assumptions used in mobile VANETs are listed as follows:

1) Nodes have mobility in a VANET. Thus, each node can join the network or leave the network

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

at any time.

2) If a node joins the network or leaves the same network later, the protocol will treat this node as a new participator.
3) Each node knows the total number of nodes in the VANET at any time.

The VANET environment contains numerous challenges for communication, many of which can be addressed by a clustered network [15]. Due to VANETs demand a high frequency of broadcast messages to keep the surrounding vehicles updated on position and safety information. These broadcasts lead to the ''broadcast storm problem'' [16], which describes the resulting congestion in the network. Both [16,17] recommend a clustered topology to effectively alleviate this congestion. VANETs have a highly-mobile environment with a rapidly changing network topology. Clustering the vehicles into groups of similar mobility will reduce the relative mobility between communicating neighbor nodes, and simplify routing. Currently, the cluster VANET is a more practical kind of VANET. Multiple nodes in a cluster of the VANET cooperate to achieve some objectives [18]. A cluster-based VANET consists of a set of loosely or tightly connected nodes that work together so that, in many respects, they can be viewed as a single system. For example, the nodes in a cluster at the same traffic intersection can detect the status of traffic is smooth, with lots of traffic or traffic congestion. The cluster-based VANET is shown in Figure 1.

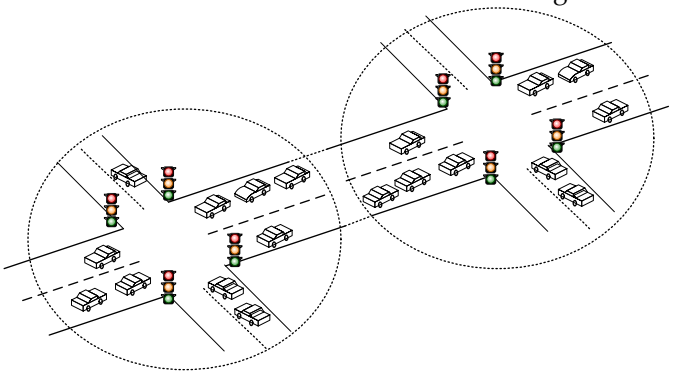


Figure 1. Example of VANET

3. Basic concept of RTAP

In this study, a distributed system whose nodes are reliable during the agreement execution in a VANET is considered; the nodes (vehicles) may be faulty due to interference from some noise or a hijacker and result in the exchanged message exhibiting arbitrary behavior. In the VANET, numerous nodes are interconnected. Achieving agreement on a same value in a distributed system; even if certain components in distributed system fail, the protocols are required so that systems can still operate correctly.

The proposed protocol, Reliable and Trustworthy Agreement Protocol (RTAP), allows all fault-free nodes to reach agreement with minimal rounds of message exchanges, and tolerates the maximal number of allowable faulty components in the VANET. In RTAP, there are two phases: the *message gathering phase*, and the *agreement making phase*. In order for all fault-free nodes to reach agreement, each node must collect enough exchanged messages from all other nodes if they are fault-free. As a result, exchanging the received values helps fault-free nodes to collect enough exchanged messages.

Fischer et al. proved that t+1 are the necessary and sufficient rounds of message exchange to solve a BA problem where $t = \lfloor (n\text{-}1)/3 \rfloor$ and n is the number of nodes in the underlying network [7]. Based on the works of Fischer and Lynch [7] and Bar-Noy et al. [19], t+1 rounds of message exchanges are the lower bound for solving the BA problem. However, the total number of nodes in the mobile VANET may change at any time, so the number of required rounds is not inherent with (t+1) in the beginning. Nevertheless, the required rounds are expectable in the VANET, and it will follow the result of the protocol proposed by Bar-Noy et al. [19]. For instance, there are six nodes in the original VANET, and RTAP must execute RR = t+1 = ($\lfloor (n\text{-}1)/3 \rfloor$)+1 = ($\lfloor (6\text{-}1)/3 \rfloor$)+1 = 2 rounds of message exchanges. If a node joins the network and the total number of the nodes in the VANET

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

changes to seven after the first round, RTAP will need RR = t+1 = $\lfloor(n\text{-}1)/3\rfloor$)+1 = ($\lfloor(7\text{-}1)/3\rfloor$)+1 = 3 rounds of message exchanges. Therefore, RTAP needs one extra round to exchange messages.

The received messages are stored in a tree structure called the message collect tree (mc-tree), which is similar to that proposed by Bar-Noy et al. [19]. Each fault-free node maintains such a mc-tree during the execution of RTAP. In the first round, source node s transmits its initial value to other nodes. We assume that each receiving node can always identify the sender of a message. When a fault-free node receives a message sent from a source node, it stores the received value, denoted as $val(s)$, at the root of its mc-tree. In the second round, each node transmits the root value of its mc-tree to all other nodes. If node a sends message val(s) to node b, then node b stores the received message from node a, denoted as $val(sa)$, in vertex sa of its mc-tree. Similarly, if node b sends message $val(sa)$ to node c then the received value is named $val(sab)$ and stored in vertex sab of the node c's mc-tree in the third round. Generally, message $val(sa...g)$ stored in the vertex $sa….g$ of a mc-tree implies that the message just received was sent through the source node, the node a,…, the node g; and the node g is the latest nodes to pass the message. In summary, the root of a mc-tree is always named s to denote that the stored message is sent from the source node in the first round; and the vertex of a mc-tree is labeled by a list of node names. The node name list contains the names of the nodes through which the stored message was transferred. Figure 2 shows an example of mc-tree.

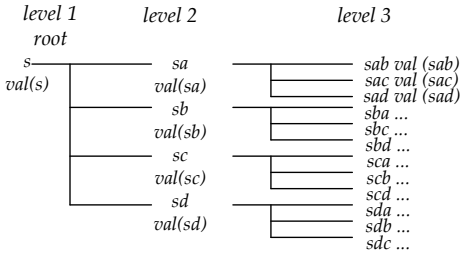


Figure 2. An example of mc-tree

Basically, all fault-free nodes in each cluster of VANET execute RTAP in order to make all fault-free nodes in the same cluster reach an agreement. In the message gathering phase, each fault-free node communicates with other nodes and itself. Furthermore, each node has high mobility, and the nodes may join the network or leave the network at any time. If a new node joins a specific cluster through the message gathering phase, the *node-join* function of RTAP will be executed to obtain the values from other nodes in the same cluster. Furthermore, if a node leaves the cluster, then the *node-leave* function of RTAP will be executed to reconstruct the mc-tree. Because of the mobility of nodes, the required rounds (a round denotes the interval of a message exchange) of message exchanges will not be an inherent value in the beginning, but it is expectable and required that at most (t+1) rounds of message exchanges must be performed to reach a common value, as proposed by Fischer and Lynch., at any time [7]. Each node in the same cluster must take care of the total number of nodes in the same cluster to decide the required number of rounds of message exchanges. Thus, the protocol will use Required Rounds (RR) to represent the required rounds of message exchanges. After RR, the collected messages are stored in the mc-tree. In the agreement making phase, each fault-free node in the same cluster computes a common value by applying the majority voting function to the messages, collected by the message gathering phase and stored on a node's mc-tree to reach an agreement.

Due to the number of faulty nodes allowed in the network depends on the total number of nodes in the network and the failure types of nodes. In Lamport et al. [6], the assumption of node fault type is malicious in a static network. The constraints of Lamport et al. [6] is $n>3f_m$, where $3f_m$ is the number of malicious faulty nodes. RTAP can tolerate f_m malicious faulty nodes, f_d dormant faulty nodes and f_a absent nodes at any time, where $n>3f_m+f_d+f_a$. When the total number of faulty nodes exceeds the limit, then the fault-free nodes cannot reach agreement. This is because each fault-free node can reach a common agreement value if $n>3f_m+f_d+f_a$. Therefore, at least $n\text{-}\lfloor(n\text{-}1\text{-}f_d\text{-}f_a)/3\rfloor\text{-}f_d\text{-}f_a$ nodes are fault-free and have the same agreement value. That is, in the worst case, a return node can receive $n\text{-}\lfloor(n\text{-}1\text{-}f_d\text{-}f_a)/3\rfloor\text{-}f_d\text{-}f_a$ copies of the same values larger than $\lfloor(n\text{-}1\text{-}f_d\text{-}f_a)/3\rfloor$, so a return node can determine the agreement value by the VOTE function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

In this study, the BA problem is considered in an asynchronous cluster-based VANET with dormant and malicious faulty nodes. And, the nodes in a cluster at the same traffic intersection can detect congestion or normal traffic status at the same intersection. Therefore, the agreement of each cluster is reached separately. The assumptions and parameters of the proposed RTAP are listed as follows:

- The underlying VANET is asynchronous.
- Each node in the VANET can be identified uniquely.
- Let N_i be the set of nodes in cluster i of VANET and $|N_i| = n_i$, where n_i is the number of nodes in the underlying cluster i of VANET, and $n_i \geq 4$.
- The nodes of the VANET are assumed to be fallible.
- There is only one source node of each cluster that transmits the message at the first round in the BA problem.
- Let f_{im} be the number of malicious faulty nodes in cluster i.
- Let f_{id} be the number of dormant faulty nodes in cluster i.
- Let f_{ia} be the maximum number of absent nodes in cluster i.
- Let f_{in} be the maximum number of faulty nodes in cluster i, where $f_{in} = f_{im}+f_{id}+f_{ia}$.

4. The proposed RTAP protocol

This section introduces the proposed RTAP to solve the BA problem in a cluster-based VANET. Generally, each fault-free node of the same cluster executes the same RTAP simultaneously to reach agreement among fault-free nodes in the same cluster. There are two phases, the *message gathering phase*, and the *agreement making phase*. Because nodes in the VANET have high mobility, nodes may join the VANET at any time, and leave the VANET at any time after. In the proposed system model, the node that joins the VANET before the agreement making phase is called the "joined node", and the node that leaves the VANET in the message gathering phase is called the "leaved node". Since the leaved node leaves the VANET, it cannot transmit and receive the message from other nodes in the VANET. Thus, the failure detector will also be unable to detect the leaved node by the same concept (no response node).

In executing RTAP, some nodes may join the network or leave the network, but each fault-free node predetermines the expectable number of required rounds in the message gathering phase, and then collects that number of rounds' messages by exchanging each node's received messages. Finally, an agreement can be reached if majority voting is applied to the collected messages in each fault-free node. In short, the BA protocol makes each fault-free node agree on a common value transmitted by the source node. Therefore, there are two phases in RTAP, they are the message gathering phase, and the agreement making phase. In addition, the number of rounds required for executing RTAP in cluster i is t_i +1 ($t_i = \lfloor (n_i-1)/3 \rfloor$). RTAP can tolerate f_{im} malicious faulty nodes, f_{id} dormant faulty nodes and f_{ia} absent nodes, where $n_i > 3f_{im}+f_{id}+f_{ia}$.

The goal of the message gathering phase is to collect the messages. In the cluster of VANET, each node has common knowledge of the entire or partial graphic information of the underlying cluster, and each node can transmit the message(s) to other nodes in the cluster. In the message gathering phase, the number of required rounds (RR) δ must first be computed, where $\delta = t_i + 1$, and $t_i = \lfloor (n_i -1)/3 \rfloor$. In the first round of the message gathering phase, the source node of cluster i transmits its initial value v_s to all other nodes in cluster i, and then each node in cluster i stores the value from the source node in the root s (level 1) of its mc-tree. After the first round of message exchanges (r>1), each node of cluster i transmits the values at level r-1 in its mc-tree to all other nodes in cluster i. However, each node of cluster i stores the values received at level r of its mc-tree, where $1 \leq r \leq \delta$.

In addition, the node that received the message can always detect the message(s) through dormant faulty components if the transmitted message is encoded by Manchester code [13]. Therefore, the message(s) through dormant faulty nodes can be detected and the value λ is replaced as the message received. The value λ is used to represent the absence message.

The goal of the agreement making phase is to compute a common agreement value for the BA problem. After the message gathering phase, each node in cluster i has its own mc-tree; and in the agreement making phase, the repeatable vertices in the mc-tree are deleted to avoid duplication of interference by the faulty nodes. Then, the VOTE function is used for each node's mc-tree in cluster i

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

from the t_i+1 level to root s of the mc-tree, and the agreement value VOTE(s) is obtained. Finally, the agreement value VOTE(s) of cluster i is transmitted to the joined nodes. RTAP tolerates f_{im} malicious faulty nodes, f_{id} dormant faulty nodes and f_{ia} absent nodes at any time, and requires RRs of message exchanges to reach an agreement in cluster i. The RTAP protocol is shown in Figure 3.

Reliable and Trustworthy Agreement Protocol (RTAP)

Each node of the same cluster executes the function *preprocessing* to get the required rounds (RR) δ.

***Message gathering phase*:**

For r = 1 do:
- The source of each cluster broadcasts its initial value v_s to other nodes in the same cluster and itself.
- Each node stores v_s in the root of its mc-tree; if the source node has a dormant fault, then the value λ replaces the initial value received from the source node.

For r = 2 to δ do:
- If a new node joins the cluster then function *node-join* is executed.
- If a node leaves the network then function *node-leave* is executed.
- The function *preprocessing* is executed to check the required rounds.
- Each node broadcasts the value at level (r-1)th of its mc-tree to other nodes in the same cluster and itself.
- Each node stores the received values at level r of its mc-tree, if the sending node has a dormant fault, then the value λ replaces the value received from the sending node.

***Agreement making phase*:**

If a new node joins the network then function *node-join* is executed.
If a node has left the network then function *node-leave* is executed.
Delete the repeatable vertices in the mc-tree.
A common agreement value of each node is determined by using VOTE function.

Function *preprocessing*:
To determine the total number of required rounds (RR) δ ($\delta = \lfloor (n_i-1)/3 \rfloor + 1$).

Function *node-join*:
1) Each node in the original cluster i sends its value received in the (r-1)th round to the new node.
2) The new node obtains the majority value from the values received from other nodes in cluster i.
3) The new node stores the majority value at level r-1 of its mc-tree.

Function *node-leave*:
Each node deletes the values received from the absent node and reconstructs its mc-tree.

Function *VOTE*(α)
If the α is a leaf, or the number of value λ is $3*(f_n-\delta+1)+(n-1)\%3$, then output α; /*Rule 1*/
else if the majority value does not exist, then output ϕ; /*Rule 2*/
else if the majority value is λ, then output λ; /*Rule 3*/
otherwise, output m, where $m \in \{0, 1\}$. /*Rule 4*/

Figure 3. The RTAP protocol

5. Example of executing RTAP

This section describes two examples to illustrate the proposed protocol. Figure 4 is a 2-cluster VANET used to illustrate the mobility of node in VANET. The first example shows how RTAP enables fault-free nodes to reach agreement when a new node joins Cluster A of VANET. The second example will show how the protocol operates when a node leaves Cluster B of VANET.

Figure 5 shows how RTAP enables each fault-free node to reach an agreement in the Cluster A of a mobile VANET when some nodes have joined the network. There are five nodes in Cluster A originally. In the BA problem, the worst case is that the source is no longer honest [12]. As an example of such a case, suppose the source node b (the source) is maliciously faulty and node c be in dormant fault which means that node b may send arbitrarily different values to different nodes in Cluster A. Therefore, in order to solve the BA problem among the fault-free node in Cluster A of the example, RTAP requires RR= t+1=$\lfloor (5-1)/3 \rfloor$+1=2 rounds in the message gathering phase.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

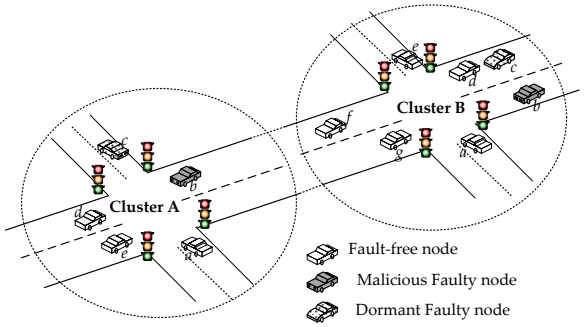


Figure 4. An example of 2-cluster VANET

After the message gathering phase is executed, a two-level mc-tree is constructed. During the agreement making phase, the VOTE function is applied to each fault-free node's mc-tree to compute a common VOTE(*s*) value. Figure 5 shows the complete steps to execute RTAP on fault-free node *a* in Cluster A when some nodes have joined the network. The steps for other fault-free nodes in Cluster A are the same as those for node *a*. The value that a faulty node agrees on is irrelevant.

When the message gathering phase begins, the source (node *b*) of Cluster A will broadcast its initial value "1" to all nodes of Cluster A in the first round if it is fault-free. If the source is malicious, it may broadcast messages to all nodes maliciously. Figure 5(a) assumes that the received values of nodes *a*, *c*, *d* and *e* are 1, 0, 1 and 0, respectively. Each node of Cluster A stores the received value into the root of its mc-tree. Figure 5(a) only presents the nodes' mc-tree. In the example, each fault-free node executes the same procedures in order to reach an agreement.

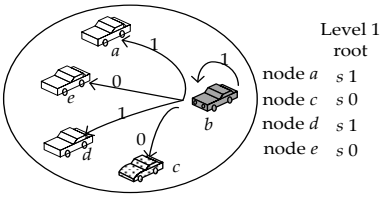
Figure 5(b) shows that a new node *f* has joined cluster A. Because it does not participate in the message gathering phase in the first round, it does not receive a value. Now, each node in the original Cluster A must send the value that it received in the first round to the new node, and the new node *f* must take the majority value of the received values as the value received in the first round. The procedure is shown in Figure 5(c).

Figure 5(d) shows the results of executing the second round of RTAP. In the second round, each fault-free node of Cluster A broadcasts the first level messages to others in Cluster A and itself and stores the values to the second level of its mc-tree. For instance, each fault-free node of Cluster A broadcasts its val(*s*) to others in Cluster A and itself. Because, node *c* in Cluster A has a dormant fault, then the value λ replaces the value received from node *c*. If node *a* receives 1 from node *a*, 0 from node *b*, λ from node *c*, 1 from node *d*, 0 from node *e* and 1 from node *f*, then, node *a* stores (1,0,λ,1,0,1) to vertices (*sa*, *sb*, *sc*, *sd*, *se*, *sf*) respectively. Since the vertex of an mc-tree is un-repeatable, val(*sb*) is omitted. Figure 5(d) shows that node *a* stores (1,λ,1,0,1) into the vertices (*sa*, *sc*, *sd*, *se*, *sf*) of its mc-tree. RTAP requires RR = t+1 = $\lfloor(n\text{-}1)/3\rfloor$+1 = $\lfloor(6\text{-}1)/3\rfloor$+1 = 2 rounds, thus, the RTAP message gathering phase is stopped at the end of the 2nd round.

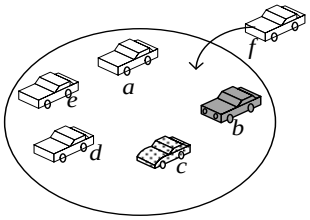
Subsequently, each fault-free node of Cluster A executes the agreement making phase to the messages on its mc-tree in order to compute a common value. The messages are collected during the message gathering phase, and stored on each fault-free node's mc-tree. Figure 5(d) shows the mc-tree of node *a* of Cluster A. In the agreement making phase, the VOTE function is applied to root *s* of a mc-tree. Figure 5(e) shows the common result (VOTE(*s*)=1) of each fault-free node in Cluster A. The agreement is reached. Since all fault-free nodes execute the same procedures, the agreement can be reached if the number of faulty nodes in Cluster A is less than or equal to $\lfloor(6\text{-}1)/3\rfloor = 1$.

When some nodes leave the network, they will affect the agreement of other fault-free nodes in the same cluster. How the nodes reach a common value in this condition is also important. Figure 6 shows how RTAP enables each fault-free node to reach an agreement in Cluster B of Figure 4 when some nodes leave Cluster B. There are seven nodes in the original Cluster B, and suppose the source node *b* is maliciously faulty and node *c* is dormant faulty. In this case, the required rounds RR = t+1 = $\lfloor(7\text{-}1)/3\rfloor$+1 = 3 in the message gathering phase.

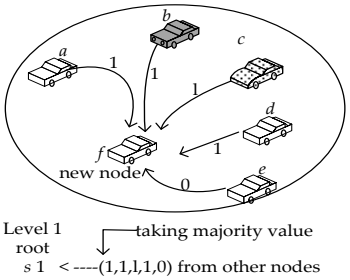
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1



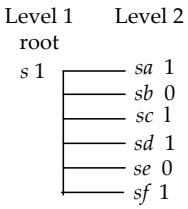
(a) The mc-tree of each fault-free node in Cluster A at the 1st round in the message gathering phase



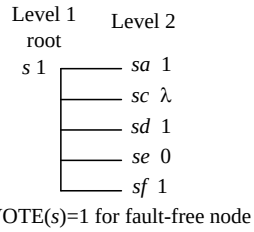
(b) The node *f* has joined Cluster A



(c) Node *f* takes the majority value of the received values



(d) The result of node *a* in Cluster A to execute the 2nd round of message gathering phase



(e) The agreement making phase

Figure 5. An example of five nodes of Cluster A to execute RTAP

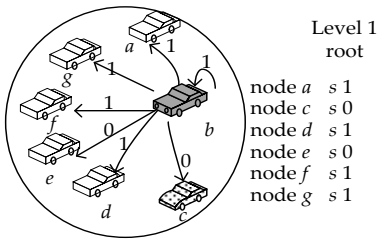
The original Cluster B is shown in Figure 4. In this case, RTAP requires 3 (RR = t+1 = $\lfloor$(7-1)/3$\rfloor$+1 = 3) rounds of message exchanges. Thus, each node of Cluster B will construct a three-level mc-tree at the message gathering phase. During the agreement making phase, the VOTE function is applied to each fault-free node's mc-tree to compute a common value VOTE(s). Figure 6 shows the complete steps to execute RTAP on fault-free node *a* of Cluster B when some nodes have left Cluster B.

First, when the message gathering phase is started, the source (node *b*) of Cluster B broadcasts its initial value "1" to all nodes of Cluster B in the first round if it is fault-free. If the source is malicious, it may broadcast messages to all nodes of Cluster B maliciously. Figure 6(a) assumes that the received values of nodes *a*, *c*, *d*, *e*, *f* and *g* of Cluster B are 1, 0, 1, 0, 1 and 1, respectively. Each node stores the received value into the root of its mc-tree. Figure 6(b) only presents the fault-free nodes' mc-tree. In the example, each fault-free node of Cluster B executes the same procedures in order to reach an agreement.

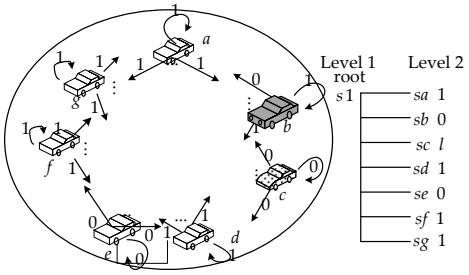
Starting the second round, as shown in Figure 6(b), every node of Cluster B broadcasts its root's value to others and itself. If some nodes fail (such as nodes *b* and *c*), they may send out faulty messages to others; otherwise, nodes will broadcast a common message to others and themselves. Each fault-free node stores the values received from other nodes into the corresponding vertices at the second level of mc-tree. Figure 6(c) shows that node *g* of Cluster B leaves Cluster B. Other nodes of Cluster B will not receive any message from node *g*. The nodes that are still in Cluster B will delete the messages received from node *g*, and the node is shown in Figure 6(d). The nodes of Cluster B will continue to execute the RTAP. Because node *g* leaves Cluster B, the total number of nodes in the Cluster B will be six. At this time, the required rounds of message exchange will change to t+1 = $\lfloor$(6-1)/3$\rfloor$+1 = 2, so the message gathering phase is stopped at the end of the 2nd round. In order to avoid the effect of a faulty node repeating, the mc-tree is un-repeatable, thus val(*sb*) is omitted. Figure 6(d) shows that node *a* of Cluster B stores (1,λ,1,0,1) into vertices (*sa*, *sc*, *sd*, *se*, *sf*) of its mc-tree.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

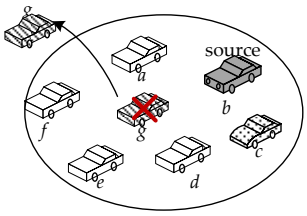
After finishing the message gathering phase, each fault-free node of Cluster B executes the agreement making phase on the message on its mc-tree in order to compute a common value. The messages are collected during the message gathering phase and stored on each fault-free node's mc-tree as shown in Figure 6(d). The VOTE function is applied to root *s* of mc-tree. Figure 6(e) shows the common result (VOTE(*s*)=1) of each fault-free node of Cluster B. Then, the agreement is reached. Since all fault-free nodes of Cluster B execute the same procedures, the agreement can be reached if the number of faulty nodes is less than or equal to $\lfloor (n\text{-}1)/3 \rfloor = \lfloor (6\text{-}1)/3 \rfloor = 1$.



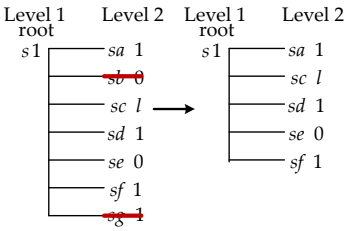
(a) The mc-tree of each node of Cluster B at the 1st round in the message gathering phase



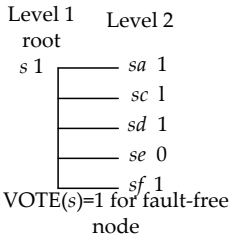
(b) The mc-tree of node *a* of Cluster B at the end of 2nd round



(c) Node *g* leaves Cluster B



(d) The result of node *a* of Cluster B to execute the 2nd round of message gathering phase



(e) The agreement making phase of each node of Cluster B

Figure 6. An example of seven nodes Cluster B to execute RTAP

6. The correctness and complexity of RTAP

6.1 Correctness of RTAP

Underlying the proof of our protocol's correctness, a vertex α is called common [7] if the value stored in vertex α of each fault-free node's mc-tree is identical. If each fault-free node shares a common initial value of the source node in the root of a mc-tree, and if the root *s* of a mc-tree in a fault-free node is common and the initial value received from the source node is stored in the root of the mc-tree, then agreement is reached because the root is common. Thus, the constraints, (Agreement) and (Validity), can be rewritten as:

(Agreement'): Root *s* is common, and

(Validity'): VOTE(*s*) = v_s for each fault-free node, if the source node is fault-free.

To prove that a vertex is common, the term *common frontier* [7] is defined as follows: When every root-to-leaf path of a mc-tree contains a common vertex, the collection of common vertices forms a common frontier. In other words, every fault-free node has the same messages collected within the common frontier if it exists within a fault-free node's mc-tree. Subsequently, using the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

same majority voting function to compute the root value of mc-tree, every fault-free node can compute the same root value because they all use the same input (the same collected messages within the common frontier). The same computing function results in the same output (the root value).

Since RTAP can solve the BA problem, the correctness of RTAP should be examined with respect to the following two terms:

(1) **Correct vertex**: Vertex αi of mc-tree is a correct vertex if node i is fault-free. In other words, a correct vertex is a place to store the value received from a fault-free node.

(2) **True value:** For a correct vertex αi in the mc-tree of a fault-free node in the fault-free node j, val(αi) is the true value of vertex αi. In other words, the stored value is the true value.

By the definition of a correct vertex, its stored value is received from the nodes in a fault-free node, and a fault-free node always transmits the same value to all nodes. Therefore, the correct vertices of such a mc-tree are common. As a result, all the correct vertices of a mc-tree are also common. Again, by the definition of a correct vertex, a common frontier does exist within the mc-tree. Thus, the root can be proven to be a common vertex ((Agreement') is true) due to the existence of a common frontier, regardless of the correctness of a source node. An agreement on the root value can now be reached. Next, we must check the validity of (Validity'). By definition, the true value of the root is the initial value of the source node if the source node is fault-free. In short, each fault-free node's root value is the initial value of the source node if the source node is fault-free; therefore, (Validity') is true when the source node is fault-free. Since (Agreement') and (Validity') are both true regardless whether the source node is fault-free or failed, the BA problem is solved.

Lemma 1. The message(s) sent through dormant faulty nodes can be detected by fault-free receiving nodes.

Proof:

The message(s) from dormant faulty components can be detected if the protocol appropriately encodes a transmitted message with the Manchester code before transmission [13].

Theorem 1. A fault-free receiving node can receive message(s) from sending nodes without influence from any faulty components between the sending node and receiving node in cluster i if $n_i > 3f_{im}+f_{id}+f_{ia}$.

Proof:

By Lemma 1, we can remove the influence of dormant faulty components between any paired sending node and receiving node in each round of message exchange, and we can rule out the influence of malicious faulty components between any pairs of nodes in each round of message exchange if $n_i > 3f_{im}+f_{id}+f_{ia}$. This is because the fault-free sending node sends n_i copies of a message to fault-free receiving nodes. In the worst case, a fault-free receiving node receives $n_i-f_{id}+f_{ia}$ messages transmitted by the fault-free sending node because message(s) from dormant and absent components can be detected; in addition, $n_i-f_{id}+f_{ia} > 3f_{im}$. Therefore, a fault-free receiving node can determine the fault-free messages by taking the majority value.

Lemma 2. A fault-free receiving node can detect the dormant faulty sending node.

Proof:

If the value of λ is greater than or equal to $(n_i-1)-\lfloor(n_i-1)/3\rfloor$ in cluster i then the sending node has a dormant fault. This is because there are at most $\lfloor(n_i-1)/3\rfloor$ malicious faulty components in the network, hence there are at most $\lfloor(n_i-1)/3\rfloor$ non-λ values.

Theorem 2. A fault-free node can detect all dormant faulty nodes in the network.

Proof:

In the protocol RTAP, there are t_i+1 rounds of message exchanges in cluster i, where $t_i \leq \lfloor(n_i-1)/3\rfloor$ and $n_i \geq 4$. Thus, there are at least two rounds of message exchanges during the message gathering phase. Each fault-free node can receive the message from the source node of cluster i during the first round of message exchanges, and receive other nodes' message(s) during the second round of message exchanges. Therefore, each node of cluster i can receive all other nodes' message(s) in same cluster after two rounds of message exchanges. According to Lemma 2, each fault-free node can detect all dormant faulty nodes within the cluster.

Lemma 3. All correct vertices of a mc-tree are common.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

Proof:

There are no repeatable vertices remain in a mc-tree. At the level t_i+1 or above, the correct vertex α has at least 2t_i+1 children of which at least t_i+1 children are correct. The true value of these t_i+1 correct vertices is common, and the majority value of vertex α is common. The correct vertex α is common in the mc-tree if the level of α is less then t_i+1. As a result, all correct vertices of the mc-tree are common.

Lemma 4. A common frontier exists in the mc-tree.

Proof:

There are t_i+1 vertices along each root-to-leaf path of a mc-tree in which the root is labeled by the source name, and the others are labeled by a sequence of node names. Since at most t_i nodes in cluster i can fail, there is at least one vertex that is correct along each root-to-leaf path of the mc-tree. Using Lemma 3, the correct vertex is common, and the common frontier exists in each fault-free node's mc-tree.

Lemma 5. Let α be a vertex; α is common if there is a common frontier in the subtree rooted at α.

Proof:

If the height of α is 0, and the common frontier (α itself) exists, then α is common. If the height of α is θ, the children of α are all consistent using the induction hypothesis with the height of the children at θ-1, the vertex α is then common.

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

Theorem 3. The root of a fault-free node's mc-tree is common.

Proof: By Lemma 3, 4, 5, and Corollary 1, the theorem is proven.

Theorem 4. Protocol RTAP solves the BA problem in a VANET.

Proof:

To prove the theorem, it must be shown that the RTAP algorithm meets (Agreement') and (Validity').

(Agreement'): Root s is common. By Theorem 3, (Agreement') is satisfied.

(Validity'): VOTE(s) = v_s for each fault-free node, if the source node is fault-free.

If the source is fault-free, then it broadcasts the same initial value v_s to all nodes. The value of correct vertices for all fault-free nodes' mc-tree is v_s. Thus, each correct vertex of the mc-tree is common (Lemma 1), and its value is v_s. Since the source is fault-free, the root of the mc-tree is also a correct vertex by Lemma 5. By Theorem 3, this root is common. The computed value VOTE(s) = v_s is stored in the root for all fault-free nodes. Thus, (Validity') is satisfied.

6.2 Complexity of RTAP

The complexity of RTAP is evaluated in terms of 1) the number of rounds of message exchanges, and 2) the number of allowable faulty nodes. Theorems 5, 6 7, and 8 below will show that the optimal solution is reached. Theorem 5 describes the number of required rounds of message exchanges and the fault tolerance capability of RTAP. Theorems 6 and 7 show that RTAP solves the BA problem by using the expectable number of message exchanges and the maximum number of allowable faulty nodes, respectively. The fault tolerant capability of RTAP is proved in Theorem 8. Thus, the optimality of RTAP is proved.

Theorem 5: RTAP requires RR (t_i+1) message exchanges in cluster i, and can tolerate f_{im} malicious faulty nodes, f_{id} dormant faulty nodes and f_{ia} absent nodes, where $t_i \le \lfloor (n_i-1)/3 \rfloor$.

Proof: 1) Some nodes may join cluster i or leave cluster i, and the total number of nodes may change at any time. The required rounds must be re-estimated at any time. Although the number of required rounds may be different at any time, it is always expectable. RR (expectable rounds) is used to represent the required rounds of RTAP. Note that RR always follows the result of Fischer et al. at any time [7], and RR = t_i+1 = $\lfloor (n_i-1)/3 \rfloor$+1.

2) By Theorem 1, RTAP can tolerate f_{im} malicious faulty nodes, f_{id} dormant faulty nodes and f_{ia} absent nodes at any time, where $n_i > 3f_{im}+f_{id}+f_{ia}$. When the total number of faulty nodes in cluster i exceeds the limit, then the fault-free nodes cannot reach agreement. Hence, the theorem is proved.

Theorem 6: RTAP solves the BA problem in cluster i by using the expectable number of message exchanges and it is the minimum.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 January 2018 doi:10.20944/preprints201801.0203.v1

Proof: Fischer et al. noted that t+1 rounds is the minimum number of rounds required to obtain enough messages to achieve BA [7]. The unit of Fischer et al. is a node, and it is the same with RTAP. Thus, the number of required rounds of message exchanges in RTAP is RR = t_i+1 rounds at any time in cluster i and this number is the minimum.

Theorem 7: The number of allowable f_{im} malicious faulty nodes, f_{id} dormant faulty nodes and f_{ia} absent nodes, where $n_i>3f_{im}+f_{id}+f_{ia}$ in RTAP of cluster i is the maximum.

Proof: If the total number of faulty nodes exceeds the limit, then there will not be sufficient messages for the fault-free nodes to remove the influence caused by the faulty nodes. This is because each fault-free node can reach a common agreement value if $n_i>3f_{im}+f_{id}+f_{ia}$. Thus, at least $n_i-\lfloor(n_i-1-f_{id}-f_{ia})/3\rfloor-f_{id}-f_{ia}$ nodes in cluster i are fault-free and have the same agreement value. That is, in the worst case, a return node of cluster i can receive $n_i-\lfloor(n_i-1-f_{id}-f_{ia})/3\rfloor-f_{id}-f_{ia}$ copies of the same values, which is larger than $\lfloor(n_i-1-f_{id}-f_{ia})/3\rfloor$, so a return node can determine the agreement value by the VOTE function. At this point, the messages will still be influenced by the faulty nodes in the agreement making phase.

Theorem 8: Using RTAP, the total number of allowable faulty nodes of VANET is optimal, and the total number of message exchanges is minimal.

Proof: In a C-clusters based VANET, the nodes in each cluster execute RTAP parallel, where C is the total number of clusters in the VANET. By Theorem 7, the number of allowable faulty nodes in cluster i is $f_{im}+f_{id}+f_{ia}$. Therefore, in this C-clusters based VANET, the total number of allowable faulty nodes of VANET is $\sum_{i=1}^{C}(f_{im}+f_{id}+f_{ia})$, and it is the maximum. By Theorem 6, the total number of message exchanges in cluster i is t_i+1. Therefore, the total number of message exchanges in a C-clusters based VANET is the largest (t_i+1) for all cluster i ($\mathrm{MAX}_{i=1}^{C}(t_i+1)$), and it is necessary.

7. Conclusion

Due to the mobility of the VANET, these nodes may join or leave the network at any time. Furthermore, some of the nodes in the network may be fallible, so the network may not be stable. The network topology developed in recent years demonstrates mobility [4]. However, the previous protocols [6,7] cannot adapt to solve the BA problem in VANET, and none of the BA protocols are designed for the VANET. Therefore, the BA problem in the VANET with hybrid failure mode of a fallible node is revisited, and the proposed protocol can tolerate the most damaging failure type that affects fallible nodes. In this paper, the proposed RTAP ensures that all fault-free nodes in the VANET can reach a common value to cope with the influence of hybrid faulty nodes by using a minimal number of message exchanges and tolerating a maximal number of faulty nodes at any time. That is, RTAP has the following features:

- RTAP can solve the BA problem in a cluster-based VANET.
- RTAP allows return nodes to reach the same agreement value.
- RTAP can solve the BA problem by the minimum number of rounds of message exchanges.
- RTAP increases the fault tolerance capability by allowing for faulty nodes (malicious faulty nodes, dormant faulty nodes and absent nodes).

In addition, a simulation of the given protocol and comparing the results and finding out any practical difficulties especially the issue of synchronizing the network will be done in near future. Furthermore, only considering node faults in the BA problem is insufficient for the highly reliable distributed system of the VANET. In the real world, not only might nodes crash, omission or malicious, but also might transmission medium crash, omission or malicious. On the other hand, our protocol will be extended to solve when dormant or malicious transmission media or nodes exist simultaneously in the underlying VANET in future work.

Acknowledgments: This work was supported in part by the Ministry of Science and Technology MOST 1.6-2221-E-324-009.

Author Contributions: Shu-Ching Wang and Yao-Te Tsai conceived the proposed method; Yao-Te, Tsai, Kuo-Qin Yan, and Ya-Jung Lin performed the experiments; Shu-Ching Wang, Kuo-Qin Yan and Ya-Jung Lin analyzed the data; Yao-Te Tsai and Ya-Jung Lin wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the

 doi:10.20944/preprints201801.0203.v1

design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Merzouki, R.; Samantaray, A.K.; Pathak, P.M.; Bouamama, B.O. Intelligent transportation systems. *Intell Mechatronic Sys* **2013**, 769-867.
2. Hsieh, Y.L.; Wang, K. Dynamic overlay multicast for live multimedia streaming in urban VANETs. *Comput Netw* **2012**, 56(16), 3609-3628.
3. Rawat, P.; Singh, K.D.; Chaouchi, H.; Bonnin, J.M. Wireless sensor networks: a survey on recent developments and potential synergies. *J Supercomput* **2014**, 68(1), 1-48.
4. Dixit, A.; Singh, S.; Gupta, K. Comparative Study of P-AODC and Improved AODV in VANET. *Int J Adv Res Comput Sci Manage Stud* **2015**, 3(1), 270-275.
5. Pease, M.; Shostak, R.; Lamport, L. Reaching agreement in the presence of faults. *J ACM* **1980**, 27(2), 228-234.
6. Lamport, L.; Shostak, R.; Pease, M.; 1982. The Byzantine generals problem. *ACM Trans Program Lang Syst* **1982**, 4(3), 382-401.
7. Fischer, M.J. The consensus problem in unreliable distributed systems (a brief survey). In *International Conference on Fundamentals of Computation Theory*, Springer, Berlin, Heidelberg, 1983, 127-140.
8. Chandra, T.D.; Toueg, S. Unreliable failure detectors for reliable distributed systems. *J ACM* **1996**, 43(2), 225-267.
9. Mansouri, N.; Dastghaibyfard, G.H.; Mansouri, E. Combination of data replication and scheduling algorithm for improving data availability in Data Grids. *J Netw Comput Appl* **2013**, 36(2), 711-722.
10. Hale, D.; Ployhart, R.E.; Shepherd, W. A two-phase longitudinal model of a turnover event: Disruption, recovery rates, and moderators of collective performance. *Acad Manage J* **2016**, 59(3), 906-929.
11. Scaramuzza, D.; Achtelik, M.C.; Doitsidis, L.; Friedrich, F.; Kosmatopoulos, E.; Martinelli, A.; Gurdan, D. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robot Autom Mag* **2014**, 21(3), 26-40.
12. Dolev, D.; Reischuk, R. Bounds on information exchange for Byzantine agreement. *J ACM* **1985**, 32(1), 191-204.
13. Căilean, A.M.; Cagneau, B.; Chassagne, L.; Dimian, M.; Popa, V. Novel receiver sensor for visible light communications in automotive applications. *IEEE Sens J* **2015**, 15(8), 4632-4639.
14. Mejri, M.N.; Ben-Othman, J.; Hamdi, M. Survey on VANET security challenges and possible cryptographic solutions. *Veh Commun* **2014**, 1(2), 53-66.
15. Hassanabadi, B.; Shea, C.; Zhang, L.; Valaee, S. Clustering in vehicular ad hoc networks using affinity propagation. *Ad Hoc Netw* **2014**, 13, 535-548.
16. Tseng, Y.C.; Ni, S.Y.; Chen, Y.S.; Sheu, J.P. The broadcast storm problem in a mobile ad hoc network. *Wirel Netw* **2002**, 8(2/3), 153-167.
17. Chen, W.; Cai, S. Ad hoc peer-to-peer network architecture for vehicle safety communications. *IEEE Commun Mag* **2005**, 43(4), 100-107.
18. Ramakrishnan, B.; Rajesh, R.S., Shaji, R.S. Analysis of routing protocols for highway model without using roadside unit and cluster. *Int J Sci Engi Res* **2011**, 2(1), 1-9.
19. Bar-Noy, A.; Dolev, D.; Dwork, C.; Strong, H.R. Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement. *Inf Comput* **1992**, 97(2), 205-233.