An Adaptive Fault Identification Protocol for an Emergency/Rescue-Based Wireless and Mobile Ad-Hoc Network*

Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki
School of Information Technology and Engineering
University of Ottawa, Ottawa, Canada

Abstract

In this paper, we consider the fault diagnosis problem in MANETs, i.e. the problem of identifying faulty hosts by fault-free ones. The diagnosis scheme that we consider is that based on the comparison approach, where hosts transmit test tasks to their neighbors and the outcomes are compared. By comparing the received outcomes fault-free hosts are able to diagnose the fault status of the network. We propose an adaptive distributed diagnosis algorithm that uses an adaptable spanning tree to disseminate the local diagnosis views throughout the ad-hoc network. The protocol allows all fault-free hosts to correctly identify all faulty ones, and it constitutes a viable addition to existing self-diagnosis protocols.

1. Introduction

Mobile ad-hoc networks (MANETs) are self-organizing, rapidly-deployable, and require no fixed infrastructure. Their main components are wireless mobile hosts, which can be deployed anywhere, that cooperate in order to dynamically establish communications. Mobile hosts in a MANET may be highly mobile, or stationary, and may vary widely in terms of their characteristics, uses, and capabilities. That is, they may differ in terms of their communication transmission ranges, processing, storage, and power capabilities, exhibiting hence varying degree of reliability. Consequently, the design of dependable MANETs is gaining much more importance in the research community [3, 10, 13].

An important problem in designing dependable MANETs that are subject to the failure of mobile hosts is the distributed self-diagnosis problem. In distributed self-diagnosis each working (fault-free) mobile host must maintain correct information about the status (working or failed) of each mobile host in its neighborhood. Moreover, for some MANETs' applications each mobile host should be able to diagnose the state of all mobiles in the network. In fact, a faulty mobile host that is not in the neighborhood of a mobile host, say u, might sooner become one of u's neighbors, since the mobile hosts are moving with varying speeds. In this paper, we consider the problem of achieving diagnosis despite the occurrence of soft faults. That is, under the assumption that faulty mobiles continue to operate and to communicate with corrupted behaviors.

The distributed diagnosis problem has been extensively studied by the distributed systems' community [2, 4, 5, 11, 15, 16, 17]. The elegant diagnosis algorithms that have been devised consider mainly wired networks. In addition, various diagnosis approaches have been used based either on invalidation models, such as the PMC model, or comparison models, such as the broadcast comparison model [5]. The most promising practical diagnosis approach is the comparison approach. In this approach, a set of jobs is assigned to the system's nodes, and their outputs are compared. The agreements and disagreements among the nodes are used to diagnose their status. For wireless and ad-hoc networks few works using the comparison approach have been done. The seminal work of Chessa and Santi [6] adapted the comparison approach to the wireless environment. They presented a distributed diagnosis algorithm that allows any fault-free mobile to diagnose the fault status of all mobiles in its neighborhood and in the network. Recently, in [7, 8] two self-diagnosis protocols have been introduced.

In this paper, we present an adaptive distributed self-diagnosis protocol to solve the diagnosis problem in MANETs. The organization of this paper is as follows: basic concepts are described in Section 2. Section 3 details the comparison-based diagnostic model. The distributed self-diagnosis protocol is presented in Section 4. Next, we prove that our protocol is correct and complete, i.e., it allows fault-free mobiles to identify all faulty hosts, and provide a com-

*This work was partially supported by the NSERC Strategic Grant, Canada Research Chair Program, the Canada Foundation for Innovation Funds, and the OIT/Distinguished Researcher Award, and the Ontario Early Career Award.

parison with related work in Section 6. Section 7 concludes the paper.

2. Preliminaries

The system we consider is composed of n mobile hosts, henceforth called *mobiles*, with unique identifiers, that communicate via packet radio network. Packet radio network, also called mobile ad-hoc network (MANET), can be described by directed graph $G_t = (V, L_t)$, called the *communication graph*, where V is the set of mobiles and L_t is the set of *logical links* at time t. A logical link, $l_{u \to v} \in L_t$ between nodes $u, v \in V$, means that the transmitter of u can reach the receiver of node v. Without loss of generality, we will assume from now on that the communication graph G_t is undirected, i.e., for any $l_{u \to v} \in L_t, l_{v \to u} \in L_t$. We indicate as one-hop *neighbors* two hosts that are connected by a logical link. If two hosts are neighbors, then they can receive from one another.

We assume that a link level protocol is responsible for providing a *1-hop reliable broadcast* primitive [14], called $\_rb(\cdot)$. The set of mobiles reachable from a node $u \in V$ at time t is called the neighborhood set and is denoted by $N_t(u)$. In order to simplify our notation, the subscript t will be dropped when it is clear from the context. k_G is the connectivity of G, i.e., minimum number of nodes whose removal results in a disconnected graph.

Faults can be classified either as *hard* or *soft*. A hard-faulted mobile is unable to communicate with the rest of the system. Whereas, a soft-faulted mobile continues to operate and to communicate, with the other mobiles, with altered behavior. If faults are allowed to occur during the execution of the diagnosis algorithm, then the faults are assumed to be *dynamic*. Whereas, *static* faults are not assumed to occur during the diagnosis phase. In this paper, we consider only static permanent faults that can be of either types: hard or soft.

The comparison approach assumes that instead of testing one node by another, the same job (or task) is assigned to a pair of distinct nodes and the results are compared. The outcome of such a comparison test can be 0 (matching results) or 1 (mismatching results). It is assumed that two fault-free nodes give matching results, while a faulty and a fault-free node provide mismatching outcomes. Based on the outcomes of comparison tests involving two faulty nodes, two diagnostic models have been studied: *symmetric* and *asymmetric* [2]. In the symmetric model, both comparison outcomes are possible in this case (0 or 1). Whereas, in the asymmetric model (see Table 1) two faulty nodes always give mismatching outputs, and hence, the comparison outcome is 1. Several generalizations of the comparison approach have been studied by considering that the comparison tests are carried by the nodes themselves [5, 12, 15].

Table 1. Asymmetric comparison outcomes

	Faulty mobiles under comparison		
Comparator	None	One	Both
Fault-free	0	1	1
Faulty	0 or 1	0 or 1	0 or 1

Recall that a MANET is called *σ-self-diagnosable* if every fault-free mobile participates and correctly diagnoses the state of all mobiles in the MANET, provided the number of faulty mobiles does not exceed σ. Chessa and Santi provided in [6] an upper bound for σ. They proved that the maximum number of faulty mobiles that can be tolerated in a diagnosis session is $k_G - 1$, i.e., $\sigma \leq k_G - 1$. The rational behind this is that when more than $k_G - 1$ mobiles fail simultaneously, the MANET might be disconnected.

An Example of an Emergency Preparedness and Response Class of Applications

Due to their ease of deployment, MANETs are an attractive choice for scenarios where the fixed network infrastructure is non-existent or unusable. Examples of applications include search and rescue, crisis management services applications, such as in disaster recovery, digital battlefield, and convert military operations. Figure 1 shows an example of a search and rescue application that comprises six rescuers and four trucks, all equipped with radio transmitters/receivers. The MANET's topology at the time of this snapshot is shown in dark lines. Note that $k_{G_t} = 3$, and hence, the MANET is 2-self-diagnosable.

3. Diagnostic Model for Ad-Hoc Networks

The comparison-based diagnostic model that is used for wireless and ad-hoc networks has been developed by Chessa and Santi [6]. Their model uses the comparison approach as a basis in order to help mobile hosts self-diagnose each other in a distributed fashion. The Chessa and Santi's model is rather an extension to the generalized comparison model [15] devised originally for wired networks. It exploits the shared nature of the communication in wireless networks by allowing each fault-free mobile u to test its neighbors by sending them a test request. Based on the responses provided by u's neighbors, mobile nodes can be diagnosed as faulty or fault-free using the comparison-based invalidation rules described in Table 1.

In this work, we assume that the network topology does not change during the testing phase. This means that if node u transmits its test request at time t, and given T_{out}

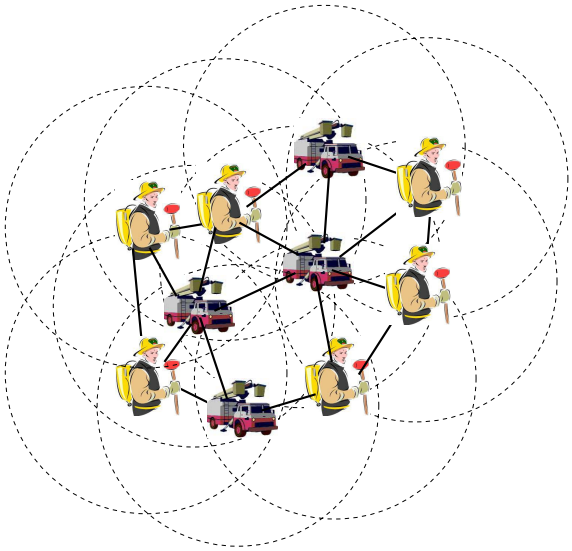

Figure 1. A search and rescue application

as the timeout time, then $N_t(u) = N_{t'}(u)$ for any $t \leq t^{'} \leq t + T_{out}$. Note that this assumption does not mean that the hosts are static, rather than its topology does not change during diagnosis. Mobiles are allowed to move without quitting their neighbors' transmission ranges. Although rather stringent, this assumption is realistic in many applications, especially for wireless mesh networks.

The main rules to which the mobile hosts should conform during a self-diagnosis session, and that constitute the *fixed topology comparison protocol*, are as follows:

R1) Test generation: In order to test its neighbors each mobile generates a message of type TEST. The test message is identified by a sequence number i and carries a test task T_i. Mobile u performs also the test task by itself, and generates the result R_i^u it expects from its neighbors, i.e., $N(u)$. Hence, node u sends the message $m_u =< \text{TEST}, i, T_i >$ at time t, and initiates a timer to T_{out}. Mobile u expects to receive all its neighbors' responses within this bound. The rational behind the use of a timeout is that neighbors that suffer from hard faults are unable to respond within the bounded time. The timeout T_{out} must be chosen such that all u's fault-free neighbors, which are able to communicate with u since it still in their transmission ranges, are guaranteed to respond to the test request within at most that time.

R2) Test reception: Let $v \in N(u)$. Upon receiving the test request $m_u =<\text{TEST}, i, T_i >$, mobile v proceeds as follows. First, it executes the test task T_i and generates the result R_i^v. Mobile v sends next a response message to all its neighbors $N(v)$. That is, it transmits $m_v =<\text{RESPONSE}, u, i, R_i^v >$ at time $t^{'}$, with $t < t^{'} < t + T_{out}$. Note that the pair (u, i) is used to uniquely identify the test task and its originator.

R3) Response reception: Upon the reception of $m_v =<\text{RESPONSE}, u, i, R_i^v >$, node $w \in N(v)$ proceeds as follows:

▶ ***Case 1:*** $\mathbf{w = u}$. This means that w is the testing unit itself. In this case, node w compares R_i^w and R_i^v, i.e., the expected result and that produced by its neighbor v. If $R_i^w = R_i^v$, then node v is considered as fault-free; Otherwise, it is considered as faulty.

▶ ***Case 2:*** $\mathbf{w \neq u}$. In this situation, the relation between both mobiles w and u can help deduce the fault status of either mobile w or v. In other words, we need to check whether w is one of u's neighbors:

▷ $\mathbf{w \in N(u)}$**:** In this case, we have $w \in N(u) \cap N(v)$, i.e., mobiles w and u (the tester node) share at least one neighbor. Since $w \in N(u)$ this means that w has already generated the result R_i^w for u's test T_i. It follows that w is able to diagnose v since it holds R_i^w and R_i^v (v's outcome for the same test T_i). Node v will be diagnosed as fault-free if $R_i^w = R_i^v$, and as faulty otherwise.

▷ $\mathbf{w \notin N(u)}$**:** In this case, we have $v \in N(u) \cap N(w)$, i.e., mobiles u and w have at least one neighbor in common. Having a single common neighbor does not help deducing v's status. Thus, the message $m_v =<\text{RESPONSE}, u, i, R_i^v >$ will be stored until mobile w receives another outcome for the same test T_i. In fact, we need at least two mobiles in common in order to be able to determine the fault status of each mobile. Consider now the scenario in which mobiles u and w have a second neighbor in common, say z, from which node w received its test outcome R_i^z. It follows that mobile w is able to diagnose the status of units v and z. Hence, if $R_i^z = R_i^v$, then both nodes are diagnosed as fault-free since we are dealing with asymmetric invalidation rules; Otherwise, if mobile w knows already that either v or z is fault-free, then it can deduce that the other mobile is faulty. For example, if $v \in FF_w$, i.e., w knows already that the node v is fault-free, then it can infer that the mobile z is faulty.

R4) Timeout occurrence: After sending its test message, mobile u initiates a timer to T_{out} in order to guarantee that its neighbors will respond within that bound. Once this time bound expires, i.e., the mobile u receives a message of type TIMEOUT, it diagnoses its neighbors that did not respond within the bounded time, i.e., $N(u) - \{F_u \cup FF_u\}$, as faulty. At this stage, mobile u knows about the status of all its neighbors, that is, u maintains two sets: FF_u, the set that contains all u's fault-free neighbors, and the set F_u

including all u's neighbors diagnosed as faulty, i.e., $F_u = F_u \cup \{N(u) - (F_u \cup FF_u)\}$.

In the following, we propose an adaptive distributed self-diagnosis protocol, based on this diagnostic model and on the use of a spanning tree, that allow any fault-free mobile to diagnose the fault status of all the mobiles in the MANET.

4. An Adaptive Self-Diagnosis Protocol

The adaptive distributed self-diagnosis protocol (Adaptive-DSDP) is called so because it uses a spanning tree (ST) that is initially configured with the MANET, and that is adapted then to any faulty situation that might affect any of its internal nodes. The ST is self-maintained connected while mobiles are moving, and it is used to disseminate the local diagnosis views throughout the network. The protocol proceeds in five phases: the *self-maintaining* phase, the *testing* phase, the *gathering* phase, the *self-repairing* phase, and the *disseminating* phase. Initially, a pre-configured ST is initialized with the MANET. In the following, each of these phases is described.

4.1. Self-Maintenance of the Tree

As specified earlier we assume that there exists a tree that spans all mobiles in the MANET. The ST can be constructed as described in [8], or by using much more efficient algorithms.such as [1]. A minimum ST would also be more appropriate in this case. Once configured, the ST should be maintained connected while the mobiles are moving. That is, when a parent migrates out the transmission range of some of its children, then these children should be reconnected to the ST. Or, when a child moves away from its parent, a new parent should be reassigned to this mobile to keep it connected to the ST. In addition, we assume that this tree is rooted at a node called the *initiator*, and that the nodes maintain the set of their children in the ST.

In order to maintain a connected ST, mobiles should periodically check whether they are still connected to the ST or not. If a mobile notices that it has lost its parent, caused either by the parent moving away from it or by the mobile migrating outside the transmission range of its parent, then it should ask its neighbors to be reconnected to the ST. That is, it seeks for a new parent connected to the ST. Furthermore, if a mobile notices that the *initiator* is now in its transmission range, i.e., it is one of its neighbors, then it should trigger the self-maintenance procedure to reconnect itself to the initiator. This will reduce the depth of the ST, and hence, the latency of the diagnosis, which can be defined as the elapsed time between the inception and end of the diagnosis session. Another heuristic that could be used also to reduce the tree's depth is to let each node maintain its depth in the tree in a variable. Once a mobile seeks for a new parent, neighbors that are connected to the ST will respond by proposing their depths in the tree. The mobile will hence choose the parent with the lowest depth.

Let $Children_u$ denote the children of mobile u in the ST at a given point of time. Periodically, every mobile, including the *initiator*, updates its knowledge about its children using the formula: $Children_u = Children_u \cap N(u)$. That is, children of mobile u can only be a subset of its neighbors. Those that migrated out of its transmission range are no longer its children.

The self-maintaining phase can be summarized by two main scenarios. The first one describes the behavior of a mobile that discovers that it can reach the *initiator* directly. Whereas, the second scenario occurs when a mobile finds out that its parent is no longer in its transmission range. In this case, the mobile triggers the maintenance phase.

If the first scenario occurs, then the procedure CONNECTTOINITIATOR, provided below, is performed. When the mobile is reconnected to the ST, the variable $ConnectedToST$ is re-initialized to TRUE. For a mobile u to declare the initiator as a new parent, it needs first to inform its actual parent, if it still in its transmission range, about this change by transmitting a message of type REMOVECHILD. Second, it should inform the *initiator* that from now on it is one of its children. This is accomplished by sending an ADDCHILD message to the *initiator*. Note that this message can be piggybacked with the first one.

Procedure CONNECTTOINITIATOR () {
// $ConnectedToST$: initially TRUE. Set to FALSE if
// mobile u discovers that it is not connected to the ST.
1. **if** $((initiator \in N(u))\ AND\ (Parent_u \neq initiator))$ {
2. $\_rb(<\text{REMOVECHILD}, u, Parent_u>)$;
3. $Parent_u = initiator$;
4. $\_rb(<\text{ADDCHILD}, u, initiator>)$;
5. $ConnectedToST = \text{TRUE}$; }}

In the second scenario, the procedure RECONNECTTOST, given below, is executed and triggers the self-maintaining phase. If a mobile discovers that it has lost its parent (line 1), it sends a RECONNECT request if it has not done yet (lines 2-4). The term α_0 has no impact at this stage since $F_u = \emptyset$, but will be useful in the self-repairing phase.

Procedure RECONNECTTOST () {
1. **if** $(Parent_u \notin N(u) \underbrace{-F_u}_{\alpha_0})$
2. **if** $(ConnectedToST == \text{TRUE})$ {
3. $ConnectedToST = \text{FALSE}$;
4. $\_rb(<\text{RECONNECT}, u>)$; }}

During the self-maintaining phase the messages of type REMOVECHILD, ADDCHILD, and RECONNECT are handled. A formal description of the self-maintenance phase is provided below, and it describes the steps performed by the mobile u. Handling REMOVECHILD and ADDCHILD are

simple, and need no more clarifications. However, processing messages of type RECONNECT needs to be clarified. Note that the terms α_1 (line 6), α_2 and α_3 (lines 8 and 14) have no impact since $F_u = \emptyset$, and will be pertinent only for the self-repairing phase (see Section 4.3). Recall that a RECONNECT message is sent once a mobile, say v, discovers that it is no longer connected to the ST. In this case the mobile should seek for a new neighbor, say u, that is connected to the ST. When mobile u receives the RECONNECT message, two scenarios might be happened. In the first scenario, mobile v is not the parent of u. In this case, if u is connected to the ST, then it adopts v as a new child by transmitting a message of type ADDPARENT (line 12). The second scenario occurs when the mobile u is the parent of v, that is, both mobiles u and v have lost connection to the ST. In this case, mobile u needs to find a new parent connected to the ST by forwarding the RECONNECT message to its neighbors (line 11). Handling ADDPARENT messages requires also some explanations. First, note that mobile u might receive many ADDPARENT messages from its neighbors. Only one parent is selected in this case. Note also that nodes can include their depth in the ST with ADDPARENT messages. This will allow the receivers to select the parent with the lowest depth, hence reducing the number of messages that will be exchanged during the diagnosis session. Once a mobile has selected a parent it should inform it by transmitting an ADDCHILD message (line 17). In addition, it should inform its neighbors that it is now connected to the ST by transmitting an ADDPARENT message (line 17). These two messages can be piggybacked.

Procedure SELFMAINTAININGPHASE () {
do{
1. $Receive(msg)$; //received from mobile $v \in N(u)$
2. **case** $msg.Type$ **of** {
3. REMOVECHILD: //msg =<REMOVECHILD, v, $Parent_v$ >
4. **if** $(u == Parent_v)$
$Children_u = Children_u - \{v\}$;
5. ADDCHILD://msg =<ADDCHILD, v, w >
6. **if** $((u == w)$ $\underbrace{AND\ (v \notin F_u)}_{\alpha_1})$
$Children_u = Children_u \cup \{v\}$;
7. RECONNECT: //msg =<RECONNECT, v >
8. **if** $(ConnectedToST ==$ TRUE $\underbrace{AND\ (v \notin F_u)}_{\alpha_2})$
9. **if** $(Parent_u == v)$ {
10. $ConnectedToST =$ FALSE;
11. $rb(<$ RECONNECT$, u>)$;
} **else**
12. $rb(<$ ADDPARENT$, u>)$;
13. ADDPARENT: //i.e. msg =<ADDPARENT, v >
14. **if** $((ConnectedToST ==$ FALSE$)$
$\underbrace{AND\ (v \notin F_u)}_{\alpha_3})$
{
15. $ConnectedToST =$ TRUE;
16. $Parent_u = v$;
17. $rb(<$ ADDCHILD$, u, v>, <$ ADDPARENT$, u>)$;
}}}}

Note that the execution of the self-maintaining phase terminates when a diagnosis session has been initiated, i.e., no mobile will ask to be reconnected to the ST if it finds out that a diagnosis session has been initiated. In addition, some instructions, i.e., α_0 (line 1 of procedure RECONNECTTOST), α_1, α_2, and α_3 (lines 6, 8, and 14 of procedure SELFMAINTAININGPHASE) have been added to this phase so that it can be transformed easily into a self-repairing phase during a diagnosis session. During the self-maintaining phase these instructions have no impact since $F_u = \emptyset$. However, once the testing and gathering phases are completed, these instructions will guide the self-repairing phase by selecting only those mobiles that are fault-free.

From now on, we will assume that using the self-maintenance procedure all mobiles are maintained connected to the ST. That is, once a diagnosis session is initiated each mobile knows its parent and its children in the ST. However, Since mobiles forming the ST might be faulty, it follows that the ST should be repaired once nodes discover during the diagnosis phase that their parent or children are faulty. This step is accomplished during the self-repairing phase that is described in Section 4.3.

4.2. Testing and Gathering Phases

Either periodically or once an altered behavior of the MANET is detected, a self-diagnosis session is initiated by starting the testing phase. The triggering of the testing phase by mobile u is done by sending a message of type TEST to its neighbors, i.e., it sends $m_u =<$TEST$, i, T_i >$ to each neighbor $v \in N(u)$. Any other mobile v upon receiving, for the first time, a TEST or RESPONSE message from one of its neighbors, it performs the following steps. First, it generates its own test request $m_v =<$TEST$, j, T_j >$ if not done yet, and transmits it to its own neighbors, i.e., $N(v)$. Then, it initiates a timer to T_{out}. If the received message is a test request T_i it replies by transmitting $m_v =<$RESPONSE$, u, i, R_i^v >$ to all v's neighbors. Second, it enters the gathering phase.

While in the gathering phase, a mobile collects response messages and diagnoses the status of the senders as faulty or fault-free based on their outputs, as described in the rule R3. Upon receiving a message of type TIMEOUT, a mobile terminates its gathering phase, and diagnoses its neighbors, that did not reply to the test request within the bounded limit, as faulty. At the end of the gathering phase, each mobile knows about the fault status of its neighbors. Hence, we need the disseminating phase during which the mobiles exchange their local diagnostic views, allowing hence each mobile to diagnose the state of the whole MANET. In our Adaptive-DSDP, the dissemination phase is performed based on the tree that spans all the fault-free mobiles. It follows that the ST maintained during the self-maintaining

phase should be repaired in order to exclude faulty mobiles.

4.3. Self-Repair of the Spanning Tree

Once the diagnosis session has been initiated and after performing the testing and gathering phases, mobiles are able to check whether they are still connected to the ST via a fault-free parent. In fact, at the end of the gathering phase each mobile u knows which neighbors are fault-free and which are faulty. In Adaptive-DSDP, we assume that each mobile should respond to all the test requests it receives. Hence, we are making sure that each mobile knows about the fault status of its neighbors and especially its parent at the end of the gathering phase. Thus, if a mobile finds out that its parent is faulty, it needs to initiate the self-repairing phase during which a new fault-free parent is found and it replaces the faulty one.

The self-repairing phase is triggered at the end of the gathering phase by running the RECONNECTTOST procedure, and it is quite identical to the self-maintaining phase. In fact, in the self-maintaining phase we connect mobiles that have lost their parents to new ones. However, in the self-repairing phase we reconnect mobiles that have faulty parents to new fault-free ones. The difference is that in the self-repairing phase only fault-free mobiles should intervene. It follows that once a diagnosis session is initiated, mobiles stop maintaining the ST and start the testing and gathering phases. At the end of the gathering phase, the self-maintaining phase is resumed, but since at this stage mobiles know who is faulty and who is not it turns out to be a self-repairing phase rather than a self-maintaining one. Note that in our implementation, α_0 (line 1 of RECONNECTTOST), α_1, α_2, and α_3 (lines 6, 8, and 14 of SELF-MAINTAININGPHASE) are the key instructions since during the self-repairing phase F_u is nonempty. Hence, only replies transmitted by fault-free ones will be considered.

4.4. Disseminating Phase

Once the ST has been repaired, the disseminating phase starts. All leaves of the ST send their local diagnosis views to their parents, using a message of type LOCAL-DIAGNOSTIC. Each parent u has to wait until it collects all its children's diagnostics. Once collected the parent combines all of them with its own local diagnostic into a unique diagnostic message containing all the identities of all the faulty nodes adjacent to at least one fault-free mobile in the subtree rooted at u. It then transmits the aggregated diagnostic message to its parent in the ST, and so on. The *initiator* collects all the local diagnostics, and it disseminates the final message $m_{initiator} =<$ GLOBALDIAGNOSTIC, $F_{initiator}, FF_{initiator} >$ down the tree to all fault-free mobiles. At this stage, the distributed diagnosis session terminates, and each fault-free node correctly diagnoses not only the state of all its neighbors, but also those of all mobiles in the MANET.

5. Correctness Proof of Adaptive-DSDP

In this section, we prove that Adaptive-DSDP provide correct and complete diagnosis, i.e., each fault-free mobile correctly diagnoses, at the end of the diagnosis session, the state of all mobiles in the σ-self-diagnosable MANET. The correctness proof of Adaptive-DSDP can be done in four main steps. First, we need to prove that the self-maintenance phase maintains a tree spanning all the mobiles. That is, before a diagnosis session starts the ST is already constructed. Second, we need to show that at the end of the gathering phase each fault-free mobile correctly identifies the states of all its neighbors, and its fault-free status is known to all its neighbors. Third, we have to show that the self-repairing phase terminates in a finite time and that it repairs the ST. Finally, we have to prove that the disseminating phase terminates in a finite time. Let $G = (V, L)$ denote the communication graph of a σ-self-diagnosable MANET.

Lemma 1 ***(Self-Maintenance Correctness)*** *If G is connected, then the self-maintaining phase maintains a tree that spans all mobiles in the MANET.*

Proof: We can prove by induction that any disconnected node, say u, will be reconnected to the ST at a given point of time before the diagnosis phase starts. Let $N_d(u)$ denote the set of mobiles given by: $N_1(u) = N(u)$ and $N_{d+1}(u) = \Big(\bigcup_{v \in N_d(u)} N_d(v) \Big) - N_d(u)$. Note that d is bounded for any graph. Let $\overline{d}$ denote this bound on d. It follows that for any mobile u, $\bigcup_{d=1 \cdots \overline{d}} N_d(u) = V$. Since the graph G is connected, it follows that for any neighbor v of u, i.e., $v \in N(u)$, with $u \in N_d(initiator)$, either $v \in N_{d'}(initiator), d' \leq d$, or $v \in N_{d+1}(u)$. In addition, if $u \in N_{\overline{d}}(initiator)$ then $N(u) \subseteq \bigcup_{d' \leq \overline{d}} N_{d'}(u)$.
If a mobile u is in the neighborhood of the *initiator*, i.e., $u \in N_1(initiator)$ and given that mobiles check periodically whether they are still connected to the ST, then u will run the procedure CONNECTTOINITIATOR, and hence, will be reconnected to the ST in a finite time.
Now assume that mobiles in $N_d(initiator)$ reconnect to the ST in a bounded time and let us show that any mobile in $N_{d+1}(initiator)$ will be reconnected to the ST in a finite time. Any mobile $u \in N_{d+1}(initiator)$ that discovers that its initial parent is no longer in its transmission range runs the procedure RECONNECTTOST. That is, it transmits a message of type RECONNECT to every mobile $v \in N(u)$. If $v \in N(u) \cap N_d(initiator)$, then mobile v will reply to u's

request with an ADDPARENT message, and hence reconnects u to the ST. If however, $v \in N(u) - N_d(initiator)$, i.e., $v \in N_{d+1}(initiator)$, then v forwards the RECONNECT message to its neighbors, and so on until the message reaches a mobile w that is connected to the ST, i.e., $w \in N_{d' \leq d}(initiator)$. Note that since we are assuming that the MANET is σ-self-diagnosable, then in the worst case every mobile has at least one fault-free neighbor from which it will be reconnected to the ST. ■

The lemma 2 shows the correct local diagnosis that each mobile should satisfy.

Lemma 2 ***(Correct Local Diagnosis)*** *Let $u \in V$ and $N(u)$ denote u's neighbors during the diagnosis session. If u is fault-free, then at the end of the gathering phase mobile u will correctly diagnose the state of all its neighbors $N(u)$.*

The proof of Lemma 2 follows logically from rules R1-R4 of the fixed topology comparison protocol (see Section 3). In fact, given that a fault-free mobile will generate a test request at a given point of time t, then all its neighbors that replied using a response message will be diagnosed either as faulty or fault-free depending on their test outputs, and those neighbors that did not respond within the bounded time T_{out} will be diagnosed as faulty.

At this stage, we have to show now that the self-repairing phase terminates in a finite time and that it repairs the ST.

Lemma 3 ***(Self-Repair Correctness)*** *If G is connected, then the self-repairing phase terminates in a finite time and repairs the ST, i.e., at the end of the self-repairing phase the ST spans only all fault-free mobiles.*

As stated earlier, the self-repairing phase is identical to the self-maintaining phase. The main difference is that in the self-repairing phase only fault-free neighbors are considered. [1] Hence, the correctness proof of the self-repairing phase follows logically from that of the self-maintaining phase, i.e., Lemma 1. Once repaired, the ST is used next to disseminate the local diagnostic views as described in Section 4.4. The correctness proof of the disseminating phase is given by Lemma 4.

Lemma 4 ***(Correct Dissemination)*** *Let $\mathbb{F}$ be the set of faulty mobiles such that $|\mathbb{F}| \leq \sigma$. If ST is the spanning tree constructed at the end of the self-repairing phase, and if ST is used to disseminate the local diagnostic views, then the disseminating phase terminates in a finite time.*

Proof: The proof follows logically from that of lemmas 1, 2, and 3. In fact, by lemmas 1 and 3 we know that the tree initialized with the MANET spans all fault-free nodes. Given that mobiles are able to verify whether their are leaves of the ST, it follows that all leaf mobiles will start the disseminating phase at some point of time. Let u denote a leaf mobile. Mobile u triggers the disseminating phase by sending a local diagnostic message to its parent in the ST. u's parent collects all its children local diagnostics, and transmits a unique diagnostic message to the upper layer in the ST in a finite time. The *initiator* receives the local diagnoses of all the nodes in a finite time. Finally, the *initiator* broadcasts the complete diagnosis down the tree, which is received in finite time by any fault-free mobile. ■

Theorem 1 states the correctness of our Adaptive-DSDP and its proof follows from lemmas 1, 2, 3, and 4. In fact, by Lemma 2 we showed that the fault-free status of any mobile is known to all its fault-free neighbors. In lemmas 1 and 3, we proved that all fault-free mobiles are connected to each others via the ST, and that no faulty mobile is part of this tree. Finally, Lemma 4 showed that the mobiles can transmit their local diagnostic views upward the tree to the *initiator*, which it aggregates the global view and sends it back downward the tree in a finite time. Hence, at the end of the diagnosis session all fault-free mobiles have a global view the fault status of the MANET.

Theorem 1 ***(Adaptive-DSDP Correctness)*** *Adaptive-DSDP is correct and complete, i.e., the output of Adaptive-DSDP at each fault-free mobile u satisfies $F_u = \mathbb{F}$, where $\mathbb{F}$ and F_u denote, respectively, the actual set of faulty mobiles in the MANET, and the set of faulty mobiles output by adaptive-DSDP.*

6. Discussion and Comparison with Related Work

In [6], Chessa and Santi developed a distributed self-diagnosis protocol for ad-hoc networks based on the fixed topology comparison protocol. In addition to the testing and gathering phases, they used a flooding-based disseminating phase since their main design objective was to minimize the diagnosis latency. We will refer the Chessa and Santi's protocol as *static-DSDP*. In a more recent work [8], Elhadef, Boukerche, and Elkadiki developed a more efficient DSDP, known as *Dynamic-DSDP*, that uses a ST-based disseminating phase where the ST is built during the diagnosis session. Whereas, in Adaptive-DSDP the ST is reconfigured with the MANET and then repaired during the diagnosis session. It has been shown in [9] that the communication complexities of our Adaptive-DSDP and that of Dynamic-DSDP largely outperform that of the self-diagnosis algorithm of Chessa and Santi. However, this improvement is counterbalanced by an increase in the diagnosis latency. Nevertheless, we

[1] See α_0, in line 1 of the procedure RECONNECTTOST, and α_1, α_2 and α_3 in lines 6, 8, and 15 of the procedure SELFMAINTAININGPHASE, in Section 4.1.

believe that both Dynamic-DSDP and Adaptive-DSDP will on the average outperform the Chessa and Santi's algorithm. Moreover, our Adaptive-DSDP will perform better than both. However, in other cases Adaptive-DSDP is expected to perform better since the repairing of the ST will take less time than building a new one. To better compare these algorithms extensive simulations are being conducted on various topologies using the network simulator NS-2. Besides, the strategy for building the ST could be improved to take into account balancing the workload between the MANET's mobiles. The definition of alternative approaches to construct the ST and the analytic and simulative evaluation are matter of ongoing studies.

It should be observed that the results stated so far rely on the main assumption that mobiles are static during the testing phase, i.e., the MANET topology does not change during the testing phase. This assumption is realistic for some ad-hoc applications in stable systems and wireless mesh networks. However, in highly dynamic MANETs such an assumption is inconceivable. For these time-varying topology MANETs a new self-diagnosis approach, called Mobile-DSDP, has been devised in [7], where the fixed topology comparison diagnostic model has been updated first to deal with these dynamic systems. Mobile-DSDP assumes that each mobile includes the test task with the response message. Hence, the state of each mobile will be either diagnosed by the mobile that originated the test, or by any other mobile that receives its response message.

7. Conclusion

We have presented in this paper a new adaptive fault identification protocol, called Adaptive-DSDP, for fixed-topology MANETs. The diagnosis is based on the comparison approach and accomplishes a correct and complete fault identification. Adaptive-DSDP uses a spanning tree in order to disseminate the local diagnosis views gathered separately by the mobiles. The spanning tree is initially configured with the MANET, and then adapted to any faulty situation that might affect any of its internal nodes.

In future work, we are investigating dynamic fault identification solutions that will be able to tolerate the occurrence of faults during the diagnosis session. We are also investigating a self-diagnosis approach that would be more appropriate for sensor networks. Last but not least, we aim the development of new adaptive failure detector that can be used by MANETs' applications or routing protocols in order collect information on the fault status of the MANETs.

References

[1] H. Baala, O. Flauzac, J. Gaber, M. Bui, and T. El-Ghazawi. A Self-Stabilizing Distributed Algorithm for Spanning Tree Construction in Wireless Ad Hoc Networks. *J. Parallel Distrib. Comput.*, 63(1):97–104, 2003.

[2] M. Barborak, M. Malek, and A. Dahbura. The Consensus Problem in Fault-Tolerant Computing. *ACM Computing Surveys*, 25(2):171–220, June 1993.

[3] C. Basile, M. Killijian, and D. Powell. A Survey of Dependability Issues in Mobile Wireless Networks. Technical Report LAAS CNRS Toulouse, France, 2003.

[4] R. Bianchini and R. Buskens. Implementation of On-Line Distributed System-Level Diagnosis Theory. *IEEE Trans. Computers*, 41(5):616–626, 1992.

[5] D. Blough and H. Brown. The Broadcast Comparison Model for On-Line Fault Diagnosis in Multiprocessor Systems: Theory and Implementation. *IEEE Trans. on Comp.*, 48(5):470–493, May 1999.

[6] S. Chessa and P. Santi. Comparison-Based System-Level Fault Diagnosis in Ad Hoc Networks. In *Proc. of the 20th IEEE Symp. on Reliable Distributed Systems (SRDS-2001)*, pages 257–266, New Orleans, LA, USA, Oct. 2001.

[7] M. Elhadef, A. Boukerche, and H. Elkadiki. Diagnosing Mobile Ad-hoc Networks: Two Distributed Comparison-Based Self-Diagnosis Protocols. In *Proc. of the 4th ACM Int. Workshop on Mobility Management and Wireless Access Protocols*, pages 18–27, Torremolinos, Spain, Oct. 2006.

[8] M. Elhadef, A. Boukerche, and H. Elkadiki. Performance Analysis of a Distributed Comparison-Based Self-Diagnosis Protocol for Wireless Ad-Hoc Networks. In *9th Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems.*, pages 165–172, Malaga, Spain, Oct. 2006.

[9] M. Elhadef, A. Boukerche, and H. Elkadiki. Self-Diagnosing Wireless Mesh and Ad-Hoc Networks using an Adaptable Comparison-Based Approach. In *Proc. of the 1st IEEE Int. Workshop on Foundations of Fault-tolerant Distributed Computing*, Vienna, Austria, Apr. 2007.

[10] M. Hollick, I. Martinovic, T. Krop, and I. Rimac. A Survey on Dependable Routing in Sensor Networks, Ad hoc Networks, and Cellular Networks. In *Proc. of the 30th EUROMICRO Conf.*, pages 495–502, France, Aug., 2004.

[11] E. P. D. Jr. and T. Nanya. A Hierarchical Adaptive Distributed System-Level Diagnosis Algorithm. *IEEE Trans. on Comp.*, (1):34–45, 1998.

[12] J. Maeng and M. Malek. A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems. In *Proc. 11th Int. Symp. on Fault-Tolerant Comput.*, 1981.

[13] M. Malek. Towards Dependable Networks of Mobile Arbitrary Devices - Diagnosis and Scalability. In *Future Directions in Distributed Computing*, Lecture Notes in Computer Science, pages 191–196. Springer, 2003.

[14] E. Pagani and G. P. Rossi. Reliable Broadcast in Mobile Multihop Packet Networks. In *Proc. of the 3rd Int. Conf. on Mobile computing and networking*, pages 34–42, 1997.

[15] A. Sengupta and A. Dahbura. On Self-Diagnosable Multiprocessor Systems: Diagnosis by the Comparison Approach. *IEEE Trans. on Comp.*, 41(11):1386–1395, 1992.

[16] M. Steinder and A. S. Sethi. A Survey of Fault Localization Techniques in Computer Networks. *Science of Computer Programming*, 53(2):165–194, 2004.

[17] A. Subbiah and D. M. Blough. Distributed Diagnosis in Dynamic Fault Environments. *IEEE Trans. on Parallel and Distributed Systems*, 15(5):453–467, May 2004.