uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Hisham Idris A. Elkadiki
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Fault Detection in Mobile Ad-Hoc Networks

TITRE DE LA THÈSE / TITLE OF THESIS

A. Boukerche
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

El Saddik

G. Wainer

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

# Fault Detection In Mobile Ad-Hoc Networks

by

Hisham Idris A. Elkadiki

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For a M.Sc. degree in
Electrical Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

©Hisham Idris El-Kadiki, Ottawa, Canada, 2008.

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

*Your file Votre référence*
*ISBN: 978-0-494-41659-4*
*Our file Notre référence*
*ISBN: 978-0-494-41659-4*

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

# Abstract

In this thesis, we consider the problem of self-diagnosis of mobile ad-hoc networks (MANETs) using the comparison approach. In this approach, a MANET consists of a collection of $n$ independent heterogeneous mobile hosts interconnected via wireless links, and it is assumed that at most $\sigma$ of these mobile hosts are faulty. In order to diagnose the state of the MANET, tasks are assigned to pairs of mobiles and the outcomes are compared. The agreements and disagreements among mobiles are the basis for identifying the faulty ones.

The comparison approach is viewed as one of the most practical diagnosis approaches. We have developed two distributed self-diagnosing protocols (DSDPs) for MANETs the first (Dynamic-DSDP) is more energy efficient, since its communication complexity is lower, while the second (Adaptive-DSDP) has a lower diagnosis latency and is more appropriate for MANETs that have more dynamic topologies. Correctness and complexity proofs are provided. Using the ns-2 simulator, we implemented the two protocols, which provided us with further insight into their effectiveness and illustrated the effect the number of faults had on their efficiency.

# Acknowledgements

To my family and especially to my parents who motivated me and supported me throughout my period of study and without whom I would have never been able to complete my work.

I would like to extend my thanks and gratitude to my supervisor, Prof. Azzedine Boukerche of the School of Information Technology and Engineering at the University of Ottawa for his advice, enduring patience, constant support and encouragement.Thanks are also given to Prof. Mourad Elhadef for his help and support and fruitful discussions. I would also like to thank my colleagues at the Paradise Lab at the University of Ottawa for their fruitful discussions and challenging debates.

Last but not least I would like to express my gratitude to both the School of Information Technology and Engineering at the University of Ottawa and all their staff and professors for providing me with the means with which I was able to complete my degree and to the Libyan Ministry of Higher Education for providing me with a scholarship to complete my degree.

# Contents

# Glossary

**DSDP** Distributed Self-Diagnosis Protocol

**MANET** Mobile Ad Hoc Network

**MM model** Meang/Malek comparison model

**MTBF** Mean Time Between Failure

**MTTF** Mean Time To Failure

**ST** Spanning Tree

# List of Figures

# List of Tables

# Chapter 1

# Introduction

As of lately, Mobile Ad-hoc Networks (MANETs) have gained much attention especially in the field of emergency preparedness. This attention can be attributed to the need of rescue/emergency personnel to be able to communicate *reliably* during times of emergency and disasters; as normal communication networks (wired/wireless) will almost always be dysfunctional [28]. As usual, the rise of a new technology gives rise to new problems and potential fields of research. Given the importance of providing a *reliable* substitute for regular communication methods during emergencies, fault-tolerance in MANETs has become an important research field that should be given extra attention so as to enhance the reliability/dependability of MANETs during times of emergency. In our work, I address the issue of fault detection in MANETs, proposing two algorithms that can efficiently diagnose a MANET and identify its faulty nodes within a bounded time period and with bounded communications overhead.

Before presenting my work and the resulting performance metrics obtained from the simulation of both algorithms, I will be providing a brief description of the main features of MANETs and their characteristics so we can identify the computational and physical limits within which we must operate. I will then go on to introduce fault-tolerance (dependability) in general and follow that with a detailed description of fault detection in MANET's before introducing my algorithms.

## 1.1 Mobile Ad-Hoc Networks

Wireless networks can operate in one of two modes, either in infrastructure mode or ad-hoc mode. In the infrastructure mode there is a central node responsible for routing and for the management of the networks main functions, i.e., this configuration is centralized. On the other hand in the ad-hoc mode all the network functions are distributed as there is no central controller, a network of nodes connected using this mode is referred to as a MANET. MANET's are composed of an autonomous collection of mobile nodes that are connected by wireless communication links. Due to the mobility of the MANET's nodes its network topology may change over time with varying speeds depending on the speed of each node [28]. In retrospect given the distributed and mobile nature of MANET's, they can be characterized as follows :

- *Limited energy source:* This is a direct result of the mobility of the nodes and the need to sometimes deploy them in remote inaccessible areas.
- *Relatively high communication error rate:* This is because of the nature of the wireless medium used during communication, where these errors are usually attributed to propagation path loss, fading and shared medium collisions.
- *Dynamic network topology:* The movement of the nodes in the network almost always result in a change in the topology of the network and this characteristic adds another level of complexity to the design of protocols for MANETs.
- *Distributed functionality:* Given the dynamic topology of MANETs and their limited energy, the distribution of functionality becomes a necessity for both energy preservation and mobility support.

## 1.2 Fault Tolerance

Fault tolerance is an established field, with the first work in it being presented by von Neumann in the mid 1950s [30]. The applications of fault tolerance were mainly in the fields of distributed computing systems and VLSIs; but the growth of the internet in the early 1980s attracted renewed attention to the field. The advent of Mobile ad hoc networks and

its applications in emergency preparedness has made the incorporation of fault tolerance into MANETs a necessity in order to fulfill the requirements of reliability that are required for such applications.

Fault tolerance is defined as the ability of a system to behave in a well defined manner once a fault occurs [11], i.e. to still be operational,if even with a lower efficiency. Where a fault is a defect in the lowest level of abstraction of the system, which could place the system in an erroneous state. Should the system deviate from its correctness specification as a result of the error we would have a system failure.

Faults can be classified based on their duration, on their underlying cause, or on how a failed component behaves once it has failed. Based on duration, faults can be classified as either *permanent*, *intermittent*, or *transient*. A transient fault will eventually disappear without any apparent intervention, whereas a permanent one will remain unless it is repaired and/or removed by some external administrator. A particularly problematic type of transient faults is the intermittent fault that recurs, often unpredictably. While it may seem that permanent faults are more severe, from an engineering perspective, they are much easier to diagnose and handle.

Based on how a failed component behaves once it has failed, we could simply classify faults as either *hard* or *soft*. A hard-faulted mobile is unable to communicate with the rest of the system, whereas, a soft-mobile continues to operate and communicate with the other mobiles in the MANET, with altered behaviors. If faults are allowed to occur during the diagnosis session these faults are considered dynamic faults, otherwise they are considered static faults.

There are three aspects to the field of fault tolerance [10], they are respectively, fault models, fault-detection and diagnosis and resiliency techniques. Fault detection is considered the first step to fault-correction and is a necessity for resiliency. It is also the aspect of fault-tolerance to which we have dedicated our work, as a first step towards fault-tolerance in MANETs.

## 1.3 Motivation

The work presented here was inspired by the work of Chessa and Santi in [8], in which they provide a model for fault diagnosis in Mobile Ad-Hoc Networks. The main motivation behind our work in this field is to provide a protocol for diagnosing faults in MANETs within a finite amount of time and with a low order of communication complexity such that its overall efficiency is an improvement over previous work in this field.

## 1.4 Contribution

In this work we present two fault-identification protocols. In the first protocol we improve on the communication complexity of previously known protocols by reducing the number of test requests a mobile node has to reply to. We also use a spanning tree for the dissemination of the diagnosis results, which also helps in reducing the overall communication complexity. In the second protocol, we use an adaptive spanning tree which is constructed at the beginning the MANETs deployment and adapts dynamically to conform to the MANETs mobility. This reduces the diagnosis latency of the second protocol, making it faster than our first proposed protocol and making it more appropriate for MANETs have high mobility.

The following publications are also relevant to this thesis :

**In Journals:**

1. M. Elhadef, A. Boukerche, H. Elkadiki, *"A distributed fault identification protocol for wireless and mobile ad hoc networks"*, Journal of Parallel and Distributed Computing, Vol. 68 No. 3, pp. 321-335, 2008.

**In Conferences:**

1. M. Elhadef, A. Boukerche, H. Elkadiki, *"Self-Diagnosing Wireless Mesh and Ad-Hoc Networks using an Adaptable Comparison-Based Approach"*, Proceedings of the The Second International Conference on Availability, Reliability and Security ( ARES 2007 ), pp. 983-990, 2007.

2. M. Elhadef, A. Boukerche, H. Elkadiki, *"An Adaptive Fault Identification Protocol for an Emergency/Rescue-Based Wireless and Mobile Ad-Hoc Network"*, 21th International Parallel and Distributed Processing Symposium ( IPDPS 2007 ), pp. 1-8, 2007.

3. M. Elhadef, A. Boukerche, H. Elkadiki, *"A Dynamic Distributed Diagnosis Protocol for Wireless and Mobile Ad-Hoc Networks"*, Proceedings of the Global Telecommunications Conference ( GLOBECOM '06 ), pp. 1-5, 2006.

4. M. Elhadef, A. Boukerche, H. Elkadiki, *"Diagnosing mobile ad-hoc networks: two distributed comparison-based self-diagnosis protocols"*, Proceedings of the Forth ACM International Workshop on Mobility Management and Wireless Access (MOBIWAC 2006), pp. 18-27, 2006.( which was awarded the 'Best Research Paper Award' at this conference )

5. M. Elhadef, A. Boukerche, H. Elkadiki, *"Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks"*, 9th International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2006), pp. 165-172, 2006.

## 1.5 Thesis Outline

The rest of this thesis is organized as follows :

- **Chapter 2 :** Introduces some background knowledge and some related work.
- **Chapter 3 :** Presents our first fault detection protocol ( Dynamic - DSDP )
- **Chapter 4 :** Presents an adaptive fault detection protocol ( Adaptive - DSDP )
- **Chapter 5 :** Concludes this thesis and provides a summary of our work in this field, here we also provide suggestions for future work

# Chapter 2

# Background and Related Work

In this chapter we discuss the development of the comparison approach and its application in solving the system level fault-diagnosis problem. We discuss the use of the comparison approach in solving the fault-diagnosis problem in wired communication systems and the relevant research that led to its use in wireless communication systems and how it was exploited to take advantage of the shared nature of the medium used in wireless communication systems. After that we present a brief description of the distributed self-diagnosing comparison-based protocol ( DSDP ) Chessa and Santi presented in [8] as an example of the previous work done in this field.

## 2.1 Fault Detection in MANETs

MANET Nodes may differ in terms of their communication, processing, storage, and power capabilities. Because of this diversity, *dependable* MANETs raises serious challenges and has been an active area of research over the past years [14], [6]. As a result, several approaches to dependability in MANETs have been proposed and evaluated. A crucial problem in this area, known as the system-level fault diagnosis problem, is to identify the fault status of mobile hosts in a MANET. Fault identification is one of the main building blocks of many dependable protocols. Several approaches have been proposed to cope with this problem [27], [1]. One of the most promising approaches is the comparison approach [8], [23], [22], [12], [26]. The comparison approach assumes that, instead of testing one node

by another, the same task is assigned to a pair of distinct nodes and the results are compared. The outcomes of such a comparison test can be 0 (matching results) or 1 (mismatching results). The agreements and disagreements among the nodes are the basis for identifying the faulty ones. The comparison approach is viewed as one of the most practical diagnosis approaches and is more efficient since it relies on more realistic assumptions about the system under consideration.

The comparison approach was first introduced independently by Malek [23], and by Hakimi and Chwa [12]. Malek's model is known as *asymmetric* comparison since it assumes that two faulty nodes always give mismatching results. Whereas, in the Hakimi and Chwa's model, called *symmetric* comparison, the outcome of comparing a pair of faulty nodes is unreliable and may be 0 or 1. It was assumed under these comparison models (symmetric and asymmetric) that all comparison tests are performed by a central observer that monitors the system, i.e., it is assumed that the fault diagnosis is centralized. In [22], Meang and Malek introduced another comparison model, henceforth known as the *Meang/Malek* (MM) model, in which they assumed that comparisons are performed by the nodes themselves, and that a comparison is performed by each node for each pair of distinct neighbors with which it can communicate directly. Since the comparator node itself might be faulty, a matching comparison outcome implies that if the comparator node is fault-free, then the compared nodes are fault-free. Similarly, a mismatching comparison outcome implies that at least one of the three nodes involved in the comparison must be faulty.

Sengupta and Dahbura generalized the MM model by allowing the comparator node to be one of the two nodes being compared [26]. The Sengupta and Dahbura model is called the *generalized comparison model*, and is known to be the most general model, i.e., it generalizes both comparison models and invalidation models. Recall that invalidation models [25], [5], [20], consider that units test each other directly, i.e., the comparator node is one of the nodes under comparison. In both the MM model and the generalized comparison model, the comparison outcomes are still sent to a central observer for diagnosis, even though the comparison tests are performed by the system's nodes. In [7], Blough and Brown developed the *broadcast comparison model*. Blough and Brown's model relies on the existence of a weak reliable broadcast protocol[1], and assumes that the outputs (or

[1]The weak reliable broadcast protocol ensures that any message sent by a fault-free node be received by any other fault-free node in the system.

checksums of the outputs) of two nodes being compared are broadcasted to all nodes in the system.

A more recent comparison-based diagnostic model for MANETs has been developed by Chessa and Santi [8]. Their model exploits the shared nature of the communication in MANETs in order to distribute the comparison test outcomes among the neighborhood of nodes.

Along with all these comparison models, different centralized or distributed diagnosis algorithms have been proposed [9], [4]. Previously developed distributed diagnosis protocols [3], [19], [15], [18], [17], [7], [29], were designed for wired multicomputer networks, and hence they do not profit from the shared nature of communication in MANETs. Moreover, the use of such protocols for MANETs would either reduce significantly the available bandwidth, or affect the diagnosis latency. Furthermore, the broadcast-comparison-based diagnosis protocol proposed by Blough and Brown [7] assumes that a weak reliable broadcast protocol is used. This assumption in unrealistic in wireless networks.

## 2.2 Chessa and Santi's DSDP

Chessa and Santi presented a DSDP based on the *fixed topology comparison protocol* [8], where they assume that the topology of the MANET is fixed during the diagnosis phase. We will refer to their DSDP from here on as *Static-DSDP*, since their dissemination phase is fixed, i.e, it does not change from one diagnosis session to anther.

Their DSDP is compromised of a testing, gathering and dissemination phase. To diagnose the MANET nodes they use an extension of the generalized comparison model [9], in which a fault free mobile say $u$ sends a test to its neighbors and based on the response provided by $u$'s neighbors, the nodes can be diagnosed as either faulty or fault-free using the comparison-based invalidation rules and the asymmetric assumptions. Their focus was to reduce the diagnosis latency of their protocol so as to reduce the restrictions imposed on the movement of the nodes by the fixed topology comparison protocol.

The testing phase of Static-DSDP starts when a node, either periodically or as a result of an abnormal behavior by one of the nodes, sends a test message to its neighboring nodes. Other nodes in the MANET initiate their testing phases either after receiving a test

request message or a test response message from one of its neighbors. After sending a test message or receiving a test response message, depending on which occurs first, a node enters the gathering phase ( note both the test and gathering phases can overlap ). During the gathering phase the nodes continues to accumulate the test response messages sent to them by their neighbors and store the diagnosis of the fault state of their neighboring nodes. At the end of the gathering phase, all nodes initiate the dissemination phase.

The purpose of the dissemination phase is for every fault-free node to share its local diagnosis with all of the other fault-free nodes in the MANET so that all the nodes have a global view of the fault-state of the MANET. To accomplish this every node broadcasts its local diagnosis to its neighbors who pass it on to theirs and so on until the local diagnosis reaches all the nodes. When a node receives a local diagnosis from another node it registers that it has already received the local diagnosis from that given node in its local registry so as to reduce the amount of redundant data broadcasted. Another mechanism applied is to not broadcast a local diagnosis, from a node, until that node is diagnosed so no faulty data is shared.

Even though the comparison model used in Static-DSDP reduces the diagnosis latency of the protocol, the use of flooding the dissemination of the local diagnosis adds to its communication complexity. In our work we will try to address both these issues.

# Chapter 3

# A Dynamic Distributed Comparison-Based Self-Diagnosis Protocol for MANETS

In this chapter, we present a distributed self-diagnosis protocol (DSDP) to solve the diagnosis problem in MANETs. The DSDP is based on the comparison approach and comprises four main phases: a testing phase, a gathering phase, a building phase and a dissemination phase. During the testing phase, each mobile tests its neighbors by sending them a test task, while during the gathering phase, each mobile collects its neighbors outputs and diagnoses them either as faulty or fault-free by comparing these results. A spanning tree is constructed during the building phase, as a means to disseminate the diagnosis messages throughout the network. Finally, during the last phase, i.e. the dissemination phase, all diagnostic information throughout the network is disseminated in order to ensure that each mobile will have a global view of the network status, i.e. each fault-free mobile correctly diagnoses the state of all the mobiles in the system.

The organization of this chapter is as follows: basic concepts and the fault model are described in Section 3.1. Section 3.2 details the diagnostic model, based on the comparison approach, for MANETs. The DSDP is presented in Section 3.3. In Section 3.4, we analyze the time and communication complexities of our diagnosis protocol and provide a proof of correctness. The results of our simulation of our protocol is presented in section 3.5.

## 3.1 System and Fault Model

The mobile ad-hoc network (MANET) we consider is composed of $n$ mobiles hosts, with unique identifiers, that communicate via a packet radio network. We assume that all the mobiles hosts have similar computing and storage resources. At a given point of time $t$, the topology of the multi-hop packet radio network, also called a mobile ad hoc network (MANET), can be described by directed graph $G_t = (V, L_t)$, known as the *communication graph*, where $V$ is the set of nodes and $L_t$ is the set of *logical links* at time $t$. A logical link, $l_{u \to v} \in L_t$ between nodes[1] $u, v \in V$ 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 1-hop $neighbors$ two hosts that are connected by a logical link. If two hosts are neighbors, then they can receive from one another. A link layer protocol is responsible for providing the following properties, which are found in the link layers of most network equipment :

- Mobiles have unique IDs, and each one knows the IDs of its neighbors.
- The ID of any message sender can be correctly identified by fault-free mobiles.
- A *1-hop reliable broadcast* primitive, called $\_rb(\cdot)$, is provided by the communication protocol used by the MANET. An example of how to implement this primitive can be found in [24].
- Messages transmitted by fault-free mobiles are correctly received by their fault-free neighbors in bounded time.

The first property ensures that fault-free mobiles will be able to correctly identify the sender of a message, whereas, the third property guarantees that any message broadcasted from a fault free mobile is correctly received on all other fault-free mobiles in its neighborhood in bounded time.

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. The connectivity of $G$, i.e. the minimum number of

[1] We will use interchangeably the words unit, host, node and mobile.

nodes whose removal results in a disconnected or trivial graph [13], is denoted by $k_G$. The diameter of graph $G$ is denoted by $\triangle_G$.

### 3.1.1 Types of faults

Various type of faults are studied in the literature. In our work, only permanent faults are considered. A permanent faulty mobile remains faulty until it is repaired and/or replaced. If the faulty mobile is unable to communicate with its neighbors, then the fault is said to be *hard*. However, if the faulty mobile continues to communicate and to function with corrupted behaviors, then it is a *soft* fault. Also, we will assume that faults are *static*, that is, they cannot occur during a diagnosis session, or more specifically during the testing phase when comparison tests are conducted. Note that if faults are allowed to occur during the testing phase then it would be hard to diagnose the state of a mobile given that it may fail just after it has been diagnosed as fault-free by its neighbors. Although rather stringent, this assumption is realistic since the diagnosis latency, i.e., the duration of a diagnosis session, is too short compared to the MTTF (Mean-Time-To-Failure) and the MTBF (Mean-Time-Between-Failure), and given that mobiles are more likely[2] to be fault-free rather than faulty.

### 3.1.2 The comparison-based diagnosis approach

In the seminal work [8], Chessa and Santi adapted the generalized comparison-based diagnostic model, originally developed for wired networks [26], to wireless and ad-hoc networks. In their diagnostic model, mobiles are required to diagnose the fault status of their neighbors by sending them test tasks, and by comparing their outcomes. Agreements and Disagreements between mobiles are used to identify their fault status. Since the comparator mobile may be faulty, then various scenarios should be considered. If the comparator mobile is fault-free, then the comparison outcome is 0 if the compared mobiles are fault-free; and it is 1 if at least one of them is faulty. Now, if the comparator mobile is faulty, then the comparison result should not be trusted, i.e., it is unreliable, and hence, it may be 0 or 1. In Table 3.1 all these scenarios are summarized.

[2]Probabilistic diagnosis models assume in general that the a priori probability of failure of a node is smaller than 0.5 [21].

| State of mobiles under comparison | Comparator's state | |
|---|---|---|
| | Fault-free | Faulty |
| None is faulty | 0 | x |
| One is faulty | 1 | x |
| Both are faulty | 1 | x |

Table 3.1: Possible comparison outcomes of the generalized comparison-based diagnostic model.

### 3.1.3 Diagnosis terminology

**Definition 1.** *A MANET is called* $\sigma$-diagnosable *if all faulty mobiles can be unambiguously identified provided the number of faulty mobiles does not exceed* $\sigma$.

Chessa and Santi provided in [8] an upper bound for $\sigma$. 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 rationale is that when more than $k_G - 1$ nodes fail simultaneously, the MANET is disconnected, and thus no mobile will be able to diagnose the status of all mobiles in the MANET.

**Definition 2.** *A MANET is set to be* self-diagnosable *if each fault-free mobile participates in each diagnosis session and correctly diagnoses the state of all mobiles in the MANET.*

A distributed self-diagnosis session is initiated either periodically or if disruptions of the specific behavior of the mobiles occurred.

**Definition 3.** *The distributed self-diagnosis is said to be correct if there are no fault-free mobiles mistakenly diagnosed as faulty; otherwise, it is an incorrect diagnosis. The distributed self-diagnosis is said to be complete if all faulty mobiles are correctly identified; otherwise, the diagnosis is incomplete.*

A distributed self-diagnosis is a correct and complete diagnosis if it identifies the exact set of faulty mobiles. In this work, we consider only the deterministic diagnosis of MANETs. The output of such a diagnosis is the set of mobiles that the self-diagnosis algorithm identifies as faulty.

### 3.1.4 An example

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, rescue, and crisis management services applications, such as the one used in disaster recovery, digital battlefields, and covert military operations.

Figure 3.1 shows an example of a military MANET that comprises ten tanks equipped with radio transmitters/receivers. The MANET'S topology at the time of this snapshot is shown in dotted lines. The connectivity of this communication graph is $k_G = 3$, and hence, the maximum number of faults that can be tolerated is $\sigma = 2$ according to Chessa and Santi's work [8]. It follows that the considered MANET is 2-diagnosable.

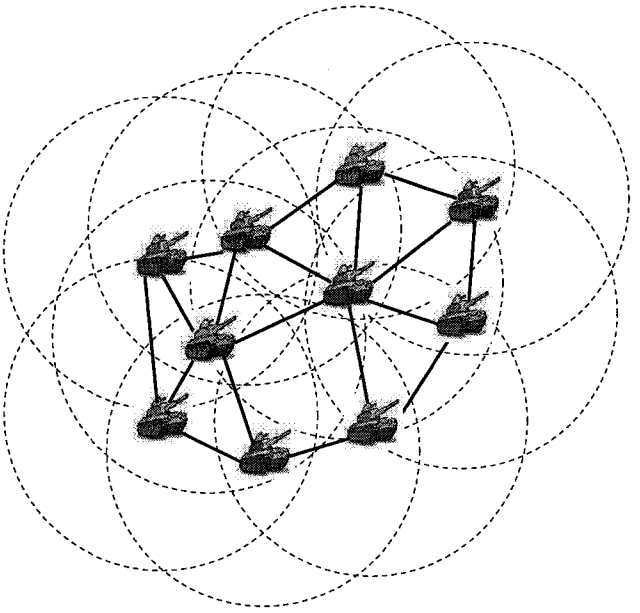

Figure 3.1: A military MANET composed of 10 tanks and its communication graph .

## 3.2 The diagnostic model for MANETs

The comparison-based diagnostic model that is used for MANETs has been developed by Chessa and Santi [8]. Their model uses the comparison model as a basis in order to help mobile hosts to self-diagnose each other in a distributed fashion. Chessa and Santi's model

is rather an extension to the generalized comparison model [26] devised originally for wired multicomputer and multiprocessor systems. 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 3.1. Chessa and Santi's diagnostic model relies on the asymmetric assumption. The rationale is that the asymmetric model is much more appropriate for MANETs.

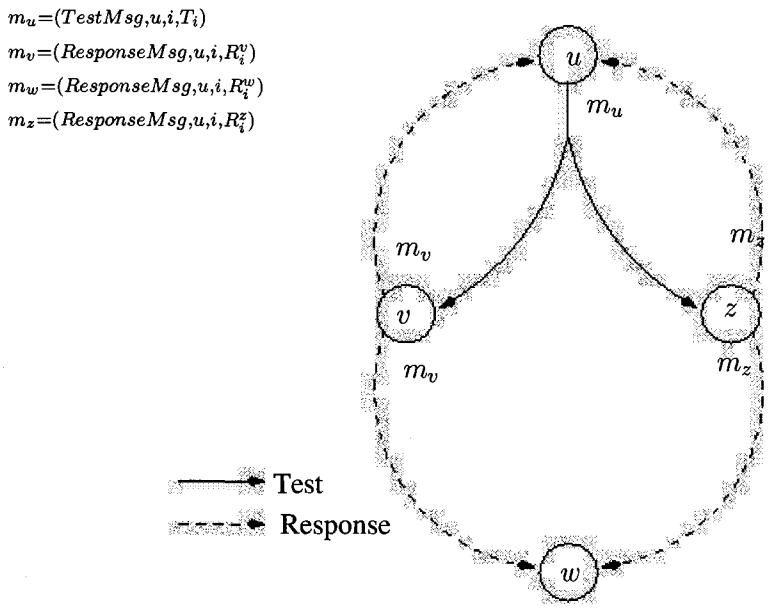


Figure 3.2: Mobile $w$ receives $v$ and $z$'s response messages corresponding to $u$'s test task.

In fact, suppose that mobile $u$ tests both its neighbors $v$ and $z$ by asking them to perform the same task. Assume now that the results output by mobiles $v$ and $z$ reach node $w$ that is not a neighbor of $u$, given the shared nature of communication in MANETs (see Fig. 3.2). Node $w$ is able to diagnose the state of both mobiles $v$ and $z$ by simply comparing their outcomes for the same task. Under asymmetric assumptions node $w$ can deduce that $v$ and $z$ are fault-free if they both provide the same result. However, mobile $w$ will not be able to deduce that if symmetric assumptions are considered. In fact, two faulty nodes might provide identical incorrect outputs, and could thus lead to an incorrect diagnosis.

Nevertheless, asymmetric assumptions are more restrictive than symmetric ones, since they assume that two faulty mobiles will not provide identical outputs for the same task. 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 $T_{out}$ 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, but rather that its topology does not change during diagnosis. Mobiles are allowed to move without exiting their neighbors transmission ranges. Although rather stringent, this assumption is realistic in many applications, where either the diagnosis latency[3] is too small or where mobiles are moving with very low speeds.

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

**(R1) Test message generation :** In order to test its neighbors, each mobile generates a message of type TESTMSG. The test message is identified by a sequence number $i$ and carries a test task $T_i$ . Mobile $u$ also performs 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 $m_u = (\text{TESTMSG}, i, T_i)$ at time $t$, and initiates a timer set to $T_{out}$. Mobile $u$ expects to receive all its neighbors responses within this bound. The rationale 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 of $u$'s fault-free neighbors, which are able to communicate with $u$ since it is still within their transmission ranges, are guaranteed to respond to the test request within at most that time, i.e. $T_{out}$.

**(R2) Test message reception :** Let $v \in N(u)$. Upon receiving the test request $m_u = (\text{TESTMSG}, i, T_i)$, mobile $v$ proceeds as follows. First, it executes the test task $T_i$ and generates the result $R_i^v$ . Mobile $v$ next sends a response message, i.e. a message of type RESPONSEMSG, to all its neighbors $N(v)$. That is, it transmits $m_v = (\text{RESPONSEMSG}, u, i, R_i^v)$ 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 and is known as the *header* of the message $m_v$.

[3]The diagnosis latency is defined as the elapsed time between the inception and the end of the diagnosis session

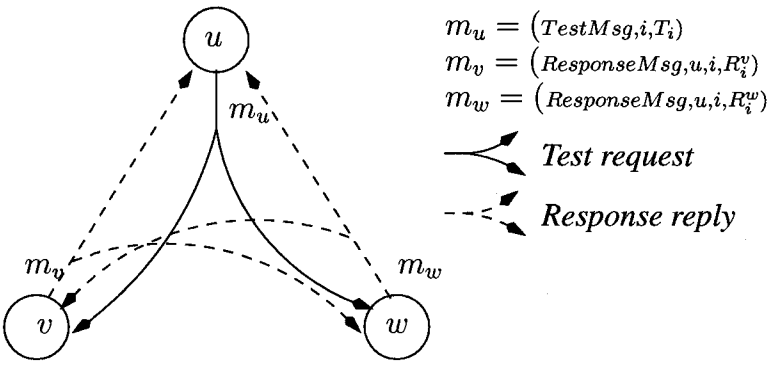


Figure 3.3: Mobile $w$ received $u$'s test task and $v$'s corresponding response reply.

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

**Case 1:** $w = u$

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 fault-free; otherwise, it is considered faulty.

**Case 2:** $w \neq u$

This holds once the mobile $w$ is not the testing unit. 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:

- $w \in N(u)$ :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 $R_i^v$, i.e. $v$'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. Fig. 3.3 depicts this scenario.
- $w \notin N(u)$ :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 only one neighbor in common does not help with deducing $v$'s status. In this case, the message $m_v = (\text{RESPONSEMSG}, u, i, R_i^v)$ is 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 one. 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 both 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 $w$ already knows 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. Fig. 3.2 depicts this scenario.

**(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 TIMEOUTMSG, 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 of $u$'s fault-free neighbors, and the set $F_u$ including all of $u$'s neighbors diagnosed as faulty, i.e. $F_u = F_u \cup \{N(u) - (F_u \cup FF_u)\}$.

Furthermore, since we are dealing with soft faults, a faulty node can alter the message header and this could result in an incorrect diagnosis as shown in Fig. 3.4. In this scenario, four mobiles are considered, say $u$, $v$, $w$, and $z$, such that $u$ and $w$ are neighbors of $v$, $v$ is a neighbor of $u$, and $z$ is a neighbor of $w$. Let $v$ be a faulty mobile. Two test messages have been generated: $m_u = (\text{TESTMSG}, i, T_i)$ and $m_z = (\text{TESTMSG}, j, T_j)$. Mobile $v$ replies to $u$'s test by transmitting a corrupted response message. Suppose that the faulty node $v$ altered the header of its response message. That is, $v$ has sent $m_v = (\text{RESPONSEMSG}, z, j, R_i^v)$. Note that $v$ was expected to send a response message with the header $(u, i)$ not $(z, j)$. Moreover, note that the corrupted header $(z, j)$ corresponds exactly to that of the test message issued by mobile $z$. Since mobile $w$ is a common neighbor to both nodes $v$ and $z$, it follows that $w$ has already computed its output $R_j^w$ for the test request sent by node $z$. Also, $w$ received $v$'s altered response message, i.e. $R_i^v$ . Since both outcomes correspond to the same header $(z, j)$, mobile $w$ will erroneously compare $R_j^w$ and $R_i^v$ , and thus lead to an incorrect diagnosis, i.e. node $w$ can incorrectly diagnose

mobile $v$ as fault-free. In fact, two different tests can provide the same result, especially when computed by faulty nodes. In [8], Chessa and Santi proposed the use of a consistency check on the message header before processing it in order to reduce the occurrence of such a problem. Unfortunately, the suggested check can only reduce the likelihood of the occurrence of this incorrect diagnosis, and hence there might be situations in which the consistency check fails.

We propose the use of *signed* headers in order to completely overcome this problem and thus ensure that a correct diagnosis will always be achieved. The tester mobile signs its test message, i.e. adds a digital signature to its header, before transmitting it. The signature is chosen such that if any node changes the content of the header of a message it has received and then it forwards the altered message, any fault-free receiver can detect this change. If the message header was altered by its sender, one can easily conclude that the sender mobile is faulty. With signed headers, the previously described problem will not occur.

In Chessa and Santi's diagnostic model, it was assumed that mobiles respond to each test request they receive. Although this approach works correctly, it increases the communication complexity of the diagnosis protocol. In fact, this approach is only needed if each mobile is required to diagnose the fault status of its neighbors. In this case, we have no other choice than to adopt this approach. However, if the main objective of the self-diagnosis protocol is to identify the fault status of all the mobiles in the MANET, then a better solution could be achieved by reducing the number of response messages, i.e. by decreasing the number of redundant information. Indeed, given the fact that at most $\sigma$ mobiles can fail simultaneously since we are dealing with $\sigma$-diagnosable ad hoc networks, we can let mobiles respond to only $\sigma + 1$ test requests. That is, we allow each fault-free mobile to inform at least one of its fault-free neighbors about its status. Note that by doing this, once a timeout occurs a mobile will no longer be able to conclude that all of its neighbors that did not respond within the time bound are faulty. In fact, some of the fault-free neighbors might have reported their status to other mobiles. It follows that a node should classify these neighbors as suspected until it gets a confirmation from other mobiles that they are either fault-free or faulty as will be described in Section 3.3. Hence, a node u now maintains three sets: the set of fault-free neighbors $FF_u$, the set of faulty ones $F_u$, and the set of suspected neighbors $\text{SUSPECTS}_u$. In the following, we propose a DSDP that allows any fault-free mobile to diagnose the fault status of all of the mobiles in the MANET. The

algorithm is based on the diagnostic model for wireless and ad hoc networks described above.

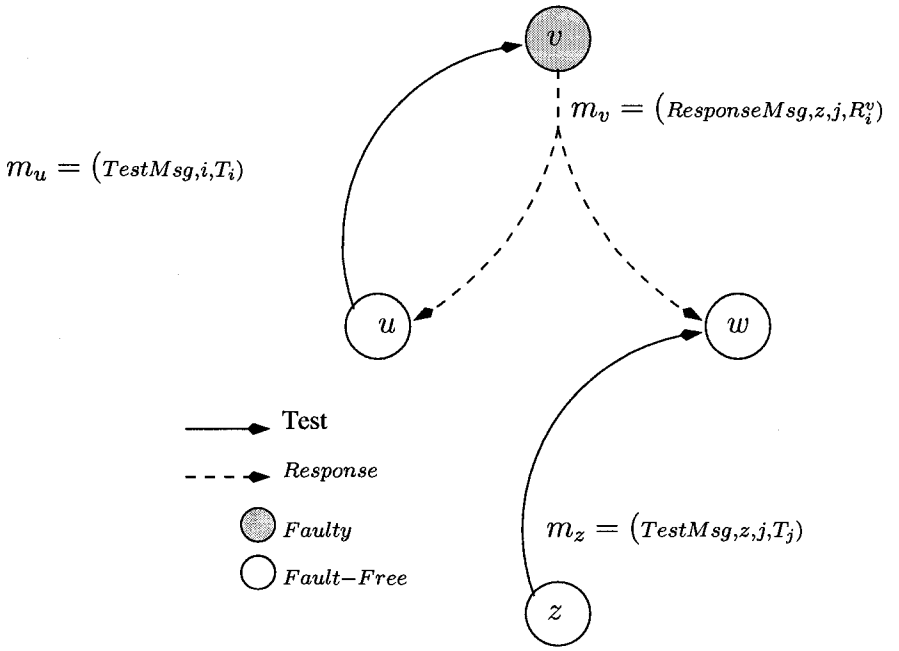


Figure 3.4: Mobile $w$ received $u$'s test task and $v$'s corresponding response reply.

## 3.3 A dynamic distributed self-diagnosis protocol

The dynamic distributed self-diagnosis protocol (Dynamic-DSDP) is thus named because it uses a dynamically constructed spanning tree (ST) in order to disseminate the local diagnosis views throughout the network. The protocol proceeds in four phases: a *testing* phase, a *gathering* phase, a *building* phase, and a *disseminating* phase. Note that the first two phases could overlap with each other; that is, some nodes may start the gathering phase while their testing phase has not yet been started. However, the building phase is started only once the first two phases have been terminated. During the testing phase, mobiles test each other by sending test requests to their neighbors. This phase is started periodically or once an abnormal behavior of the mobiles is detected. The generation of the test requests follows the rule R1 of the fixed topology comparison protocol described in Section 3.2. Once a test request is generated or received, a mobile enters the gathering phase, during which it collects the

response messages broadcasted by its neighbors and behaves as described by rules R2 and R3. The gathering phase ends once a timeout occurs (rule R4), i.e. the time bound during which non-faulty neighbors are expected to respond has expired. At this stage, any fault-free mobile knows about the fault status of some of its neighbors. The end of the gathering phase triggers the building phase that constructs a tree spanning all fault-free mobiles. The ST is then used during the disseminating phase. During this phase, all fault-free mobiles exchange their local diagnosis views, thus generating a complete diagnosis of the MANET. The self-diagnosis session ends once a global view of the fault status of the whole system is reached.

### 3.3.1 The testing phase

The triggering of the testing phase by mobile $u$ is done by sending a message of type TESTMSG to its neighbors, i.e. it sends $m_u = (\text{TESTMSG}, i, T_i)$ to each neighbor $v \in N(u)$.The generation of this test request is described in the procedure TESTNEIGHBORS illustrated below. Any other mobile $v$ performs the following steps upon receiving a TESTMSG or RESPONSEMSG from one of its neighbors for the first time. First, it generates its own test request $m_v = (\text{TESTMSG}, j, T_j)$, if not done yet, and transmits it to its own neighbors, i.e. $N(v)$. Then it initiates a timer to $T_{out}$. Second, if the received message is a test request, $T_i$ , it replies by transmitting a response message $m_v = (\text{RESPONSEMSG}, u, i, R_i^v)$ to all $v$'s neighbors. Finally, it enters the gathering phase.

**Procedure** TESTNEIGHBORS () {
  $i = GenerateSequenceNumber()$;
  $T_i = GenerateTestTask()$;
  $R_i^u = ComputeResult(T_i)$;
  $\_rb(m_u = (\text{TESTMSG}, u, i, T_i))$;
  $SetTimer(T_{out})$;
  $TestGenerated = \text{TRUE}$;
}

### 3.3.2 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 TIMEOUTMSG, a mobile terminates its gathering phase and diagnoses any neighbors that did not reply to the test request within the bounded limit as suspected. The pseudocode for the gathering phase is illustrated below.

```
Procedure GATHERINGPHASE() {
    do{
1.    Receive(msg); // msg has been sent by mobile v ∈ N(u)
2.    if (CheckHeader(msg, v) == FALSE )
3.      F_u = F_u ∪ {v};
      else {
4.      case msg.Type of {
5.        TESTMSG: // i.e.  msg = (TESTMSG, w, j, T_j)
6.        if (NbResponses ≤ σ + 1) {
7.            R_j^u = ComputeResult(T_j);
8.            _rb(m_u = (RESPONSEMSG, v, j, R_j^u));
9.            Log = Log ∪ {< v, j, R_j^u >};
10.            NbResponses + +; }
11.       RESPONSEMSG: // i.e.  msg = (RESPONSEMSG, w, j, R_j^v)
12.        if (w ∉ F_u ∪ FF_u) { // w has not been diagnosed yet
13.            if ((w == u) OR (w ∈ N(u))
14.                if (R_j^v == R_j^u) FF_u = FF_u ∪ {v};
15.                else                F_u = F_u ∪ {v};
16.            else // w ∉ N(u)
17.                if (∃ < w, j, R_j^z > ∈ Log : z ∈ N(u))
18.                    if (R_j^z == R_j^v) FF_u = FF_u ∪ {v, z};
19.                    elsif (z ∈ FF_u) F_u = F_u ∪ {v}; }
20.                        Log = Log ∪ {< w, j, R_j^v >};
21.       TIMEOUTMSG: // i.e. the delay T_out has expired
22.                    SUSPECTS_u = N(u) − (F_u ∪ FF_u); }
23      if (TestGenerated == FALSE) TESTNEIGHBORS();
      }
24. }while(F_u ∪ FF_u ∪ SUSPECTS_u == N(u)); }
```

At the end of the gathering phase, each mobile knows about the fault status of a subset of its neighbors. Thus, we need the disseminating phase during which the mobiles exchange their local diagnosis views, allowing each mobile to know the states of the suspected neighbors.

Moreover, at the end of the disseminating phase, every mobile diagnoses the state of the whole MANET. In Dynamic-DSDP, the dissemination phase is performed based on a tree spanning all fault-free mobiles. This tree is constructed during the building phase and once the gathering phase ends. Three new types of messages are needed to complete the disseminating phase, and are described as follows:

- STMSG:This type of message is used during the building phase to construct the ST.
- LOCALDIAGNOSTICMSG:During the disseminating phase any fault-free mobile should inform its parent in the ST about what it knows regarding the fault status of its neighbors. Hence, a mobile $u$ transmits the message $m_u = (\text{LOCALDIAGNOSTICMSG}, F_u, FF_u)$ to its parent. These messages are aggregated with each other to form a global diagnosis view.
- GLOBALDIAGNOSTICMSG:This message is used by fault-free mobiles to disseminate the global diagnosis view, once computed, throughout the MANET.

In the following we describe how the ST is constructed and how it is used to disseminate the local diagnosis views.

### 3.3.3 The building phase

The construction of a tree spanning only fault-free mobiles can easily be done by assuming that this step is initiated by a unique fault-free node, known as the *initiator*, in response to an explicit request of the external system administrator. The *initiator* starts by transmitting to its neighbors the message $m_{initiator} = (\text{STMSG}, initiator, initiator)$, hence triggering at this stage the construction of the ST. Since each mobile knows at least one fault-free neighbor at this stage, then the STMSG messages will be propagated throughout the MANET. The STMSG messages sent by faulty mobiles are simply ignored by their neighbors.

When a mobile, say $u$, receives an STMSG message for the first time, it verifies whether the sender is fault-free by using its local diagnosis. If it is fault-free, then it declares the sender, say $v$, as its parent in the ST. Mobile $u$ next transmits $m_u = (\text{STMSG}, u, v)$, i.e. its own STMSG message, to its neighbors specifying that the fault-free mobile $v$ is

now its parent in the ST. Upon receiving this message, mobile $v$ knows that $u$ is one of its children. Consequently, a tree spanning all fault-free nodes is built during propagation. Furthermore, node $v$ knows the identities of its parent and children in the ST in at most $T_{out}$ time, since a timer is initiated to this bound once an STMSG message is sent by a mobile. The pseudocode for the construction of the ST is provided below.

```
Procedure BUILDINGPHASE() {
// Children_u : set of mobiles that are children of node u in the ST.
    do{
1.    Receive(msg); // received from mobile v ∈ N(u)
2.    case msg.Type of {
3.      STMSG : // i.e.  m_v = (STMSG, v, w)
4.        if (w ∈ FF_u)
5.          if (u == w)
6.            Children_u = Children_u ∪ {v}
7.          elsif (ParentSelected == FALSE) {
8.            Parent_u = v
9.            ParentSelected = TRUE
10.           Broadcast(m_u = (STMSG, u, v));
11.           SetTimer(T_out); }
12.     TIMEOUTMSG : // i.e. the delay T_out has expired
13.       _rb(m_u = (LOCALDIAGNOSTICMSG, F_u, FF_u));}
14. }while((Children_u ≠ FF_u)AND
          (msg.Type ≠ TIMEOUTMSG)); }
```

### 3.3.4 The dissemination phase

Once the ST has been constructed, the disseminating phase starts. All leaves of the ST, i.e. nodes without children, send their local diagnosis views to their parents, using a message of type LOCALDIAGNOSTICMSG. 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 $m_{initiator} = (\text{GLOBALDIAGNOSTICMSG}, F_{initiator}, FF_{initiator})$ 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. A formal description of the disseminating phase is given below.

**Procedure** DISSEMINATINGPHASE () {
**do**{
1. *Receive*(*msg*); // received from mobile $v \in N(u)$
2. **case** $msg.Type$ **of** {
3. LOCALDIAGNOSTICMSG: $//m_v$ = (LOCALDIAGNOSTICMSG, $F_v, FF_v$)
4. **if** ($v \in Children_u$) {
5. $FF_u = FF_u \cup FF_v$;
6. $F_u = F_u \cup F_v$;
7. $\_children = \_children \cup \{v\}$;
8. **if** ($Children_u == \_children$)
// mobile $u$ waits for all its children's diagnosis views
9. $\_rb$((LOCALDIAGNOSTICMSG, $F_u, FF_u$)); }
10. **if** ($u == initiator$ {
11. $F_u = F_u \cup (V - (F_u \cup FF_u))$;
12. $\_rb$((GLOBALDIAGNOSTICMSG, $F_u, FF_u$));
13. $SystemDiagnosed$ = TRUE; }
14. GLOBALDIAGNOSTICMSG: $//m_v$ = (GLOBALDIAGNOSTICMSG, $F, FF$)
15. **if** ($v == Parent$) {
16. $FF_u = FF$;
17. $F_u = F$;
18. $Broadcast$((GLOBALDIAGNOSTICMSG, $F, FF$));
19. $SystemDiagnosed$ = TRUE; }
20. }**while**($SystemDiagnosed$ == FALSE); }

## 3.4 Protocol analysis

In this section, we will first prove that Dynamic-DSDP provides 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 $\sigma$-diagnosable MANET. Then we will analyze its communication and time complexities.

### 3.4.1 Proof of correctness of the Dynamic-DSDP protocol

To establish the correctness of our protocol, we define the concepts of *correct partial local diagnosis* and *correct dissemination*. A fault-free mobile achieves a correct partial local

diagnosis if at the end of the gathering phase its state is correctly diagnosed by at least one fault-free neighbor. A correct dissemination is achieved if the disseminating phase terminates in a finite time and every fault-free mobile correctly diagnoses the state of all mobiles. That is, all mobiles correctly receive the global diagnosis view disseminated by the *initiator*. To prove the correctness of Dynamic-DSDP we need to show that it satisfies the correct partial local diagnosis and the correct dissemination properties. Lemma 1 shows the correct partial local diagnosis that each mobile should satisfy.

**Lemma 1.** ***(Correct partial local diagnosis)*** *Given a $\sigma$-diagnosable MANET represented by the connected graph $G = (V, L)$, 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 of Dynamic-DSDP its fault-free status is correctly diagnosed by at least one of its fault-free neighbors.*

*Proof:* This lemma can be proved by assuming that $u$ has $\sigma$ neighbors, i.e. $|N(u)| \leq \sigma$, and that all of them are faulty. Consequently, the removal of $u$'s neighbors will result in a disconnected graph, and hence $|N(u)| \geq k_G$. It follows that $\sigma \geq k_G$. Furthermore, the maximum number of faults $\sigma$, that can be tolerated in a diagnosable MANET, should not exceed $k_G$, i.e. $\sigma \leq k_G - 1$, which is a contradiction. Thus, $|N(u)| \geq \sigma$. That is, mobile $u$ has at least one fault-free neighbor, say $v$, that will generate a test request at a given point of time $t$ and $u$ will respond with a correct test output within at most $T_{out}$ time. As a result, mobile $v$ will be able to correctly identify that $u$ is fault-free. ■

To prove the correct dissemination property, we need first to show that the tree built at the end of the gathering phase spans all fault-free mobiles. This property is henceforth known as the *fault-free ST* and is given in the lemma below followed by the proof of the correct dissemination property.

**Lemma 2.** ***(Fault-Free ST)*** *Let $G = (V, L)$ be the communication graph representing the $\sigma$-diagnosable MANET once the diagnosis session is initiated and let $\widetilde{F}$ denote the set of faulty mobiles such that $|\widetilde{F}| \leq \sigma$. The tree constructed during the building phase of Dynamic-DSDP spans all the fault free mobiles of the MANET.*

*Proof:* First, note that there is no need to prove that a tree is constructed at the end of the gathering phase since the construction phase of the ST is very simple, and hence the proof is trivial. However, we need to prove that the tree spans all fault-free mobiles. Given

that the graph $G$ is connected and since the number of faulty mobiles $|\widetilde{F}| \leq \sigma < k_G$, it follows that the graph $\widetilde{G}$ induced on $G$ by node set $V - \widetilde{F}$ is connected. Since $|\widetilde{F}| < k_G$ then every fault-free node must be adjacent to at least one other fault-free neighbor, and hence it is correctly diagnosed by at least one fault-free mobile (see Lemma 1). Moreover, since STMSG messages, used to build the tree, that originated from faulty mobiles are discarded by fault-free receivers (line 4 of procedure BUILDINGPHASE in section 3.3.3), it follows that the tree constructed at the end of the gathering phase spans all $\widetilde{G}$'s nodes, that is, spans all fault-free mobiles. ■

**Lemma 3.** ***(Correct dissemination)*** *Let $G = (V, L)$ be the connected communication graph representing the $\sigma$-diagnosable MANET, and $\widetilde{F}$ the set of faulty mobiles such that $|\widetilde{F}| \leq \sigma$. If ST is the ST constructed at the end of the building phase, and if ST is used to disseminate the local diagnostic views, then the disseminating phase terminates in a finite time.*

*Proof:* The proof of this lemma follows trivially from that of Lemmas 1 and 2. In fact, by Lemma 2 we know that the tree created before the starting of the disseminating phase spans all fault-free nodes. Also, using Lemma 1 and given that $|\widetilde{F}| < k_G$, it follows that each fault-free mobile is adjacent to at least one fault-free neighbor, and that its state is correctly diagnosed by at least one fault-free mobile. Given that mobiles are able to verify whether they are leaves of the ST, it follows that all leaf mobiles should start the dissemination phase at some point of time. Let $u$ denote a leaf mobile. Mobile $u$ triggers the dissemination 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* broadcast the complete diagnosis down the tree, and it is received in finite time by any fault-free mobile. ■

The following theorem states the correctness of our Dynamic-DSDP protocol and its proof follows trivially from Lemmas 1-3. In fact, by Lemma 1 we showed that the fault-free status of any mobile is known to at least one fault-free neighbor, that is, each fault-free mobile knows a subset of the fault-free set. In Lemma 2, we proved that all these fault-free mobiles are connected to each other via the ST and that no faulty mobile is part of this tree. Finally, Lemma 3 showed that the mobiles can transmit their local diagnostic views up the

tree to the initiator, which aggregates the global view and sends it back down the tree in a finite time. Hence, at the end of the diagnosis session, all fault-free mobiles have a global view of the fault status of the MANET.

**Theorem 1.** *Given a $\sigma$-diagnosable MANET with communication graph $G = (V, L)$, Dynamic-DSDP is correct and complete, i.e., the output of Dynamic-DSDP at each fault-free mobile $u$ satisfies $F_u = \widetilde{F}$ and $FF_u = V - \widetilde{F}$, where $\widetilde{F}$ is the set of all faulty mobiles in the MANET, and $F_u$ and $FF_u$ are, respectively, the sets of faulty and fault-free units maintained by any fault-free mobile $u$ at the end of the diagnosis session.*

### 3.4.2 Communication complexity of the Dynamic-DSDP protocol

Let $n$ and $d_{max}$ denote, respectively, the total number of mobiles in the $\sigma$-diagnosable MANET and the maximum of the node degrees.

**Theorem 2.** *The communication complexity of Dynamic-DSDP is $O(n\sigma)$.*

*Proof:* The number of 1-hop reliable broadcast messages that are exchanged during a Dynamic-DSDP-based self-diagnosis session are summarized in the following Table ( Table 3.2). The total number of exchanged messages is $n(\sigma + 4) - 1 \leq n(k_G + 3) - 1$. That is, the communication complexity is $O(nk_G) \simeq O(n\sigma)$. ■

### 3.4.3 Time complexity of the Dynamic-DSDP protocol

Let $G = (V, L)$ be the communication graph of the MANET. Let $\triangle_G$ and $d_{ST}$ denote, respectively, the diameter of $G$ and the depth of the spanning tree used by the disseminating phase. The time complexity will be expressed in terms of the following two bounds :

- $T_{gen}$: an upper bound to the elapsed time between the reception of the first diagnostic message and the generation of the test request.
- $T_f$: an upper bound to the time needed to propagate a message.

| **Message types** | **♯Messages** | **Explanation** |
|---|---|---|
| TESTMSG | $n$ | All mobiles will generate at most one test message |
| RESPONSEMSG | $n(\sigma+1) \leq nk_G$ | Each mobile responds to exactly $\sigma+1$ test messages and $\sigma \leq k_G - 1$ |
| STMSG | $n$ | The worst-case scenario occurs when all mobiles are fault-free. The initiator sends one STMSG message and any other node sends one |
| LOCALDIAGNOSTICMSG | $n-1$ | Each mobile, excluding the *initiator*, sends one message of this type |
| GLOBALDIAGNOSTICMSG | $n-1$ | In the worst case scenario the depth of the spanning tree is $n-1$. hence, $n-1$ GLOBALDIAGNOSTICMSG messages are needed to broadcast the complete diagnosis down the tree |

Table 3.2: Communication complexity of messages in Dynamic-DSDP

The time required to complete a self-diagnosis session is composed of the time needed to complete the testing and gathering phases, plus that required by the building phase, and finally, plus the time needed to terminate the disseminating phase.

**Lemma 4.** *The time complexity of the testing and gathering phases is* $\triangle_G T_{gen} + T_{out}$.

*Proof:* The last mobile to generate its test message will do so at most in $T_{gen}$ since the first diagnostic message was received. It follows that in at least $\triangle_G T_{gen}$ all non-faulty mobiles generate their test requests. Any fault-free mobile diagnoses at least one fault-free neighbor in at most $T_{out}$ time. Hence the testing and gathering phases requires at most $\triangle_G T_{gen} + T_{out}$. ■

**Lemma 5.** *The time complexity of the disseminating phase is* $d_{ST} T_f$ .

*Proof:* The disseminating phase starts at mobiles with no children, which have to transmit their local diagnostic views to their parents. Once a parent collects all its children's local diagnostic views, it forwards the aggregated diagnostic views along with its own local diagnostic view to its parent, and so on up the tree. In at most $d_{ST} T_f$ , the initiator has collected all diagnostic views and disseminates the global diagnostic view that reaches the farthest mobile in at most $d_{ST} T_f$ . Hence, the disseminating phase requires $2 d_{ST} T_f$ time to complete. ■

**Lemma 6.** *The time complexity of the building phase is* $d_{ST} T_f + T_{out}$.

*Proof:* The construction of the ST starts once the initiator transmits the first STMSG message. The reception of an STMSG message by a mobile triggers it to transmit its own STMSG message. Thus, STMSG messages are propagated throughout the MANET. The farthest mobile generates its STMSG message in at most $d_{ST} T_f$ . The last mobiles to generate this type of messages should wait for $T_{out}$ before discovering that they are the leaves of the tree and thus initiate the disseminating phase. It follows that the building phase needs $d_{ST} T_f + T_{out}$ time to complete. ■

**Theorem 3.** *The time complexity of Dynamic-DSDP is* $O(\triangle_G T_{gen} + d_{ST} T_f + T_{out})$.

The proof of this theorem is trivial. In fact, Dynamic-DSDP comprises the testing and gathering phases, the building phase and the disseminating phase. It follows that, by Lemmas 4-6, it requires at most $\triangle_G T_{gen} + 3d_{ST}T_f + 2T_{out}$ to complete a diagnosis session.

## 3.5 Simulation Results

In order to further characterize our protocol and measure its efficiency, we chose to conduct an extensive set of simulations using the $NS-2$ simulator. For simulation purposes we generated a set of communication graphs $G$, where nodes were randomly distributed, with a known minimum common connectivity $k_G$ and the number of nodes ranging from 10 to 100. We then used a graph-connectivity measurement utility that was based on the algorithm proposed in [16]. In both scenarios we used the same set of simulation parameters for $NS-2$; these are summarized in Table 3.4.
The performance of our protocol will be compared to Chessa and Santi's work [8], which we refer to as Static-DSDP.

| **Parameter** | **Value** |
|---|---|
| kG | 4 |
| Number of Nodes in Network | 10 - 100 |
| Simulation Time | 150s |
| Transmission Range | 150m |
| Topology Size | 500 x 500 m |
| Propagation Delay | 25 us |
| Link Layer Queue Length | 10000 |
| Propagation Model | TwoRayGround |
| Antenna Model | OmniAntenna |

Table 3.3: Simulations environment variables for Dynamic-DSDP

### 3.5.1 Efficiency with regards to the number of packets

Here we subjected Dynamic-DSDP to an extensive set of simulations using the previously generated set of graphs $G$, and a large, randomly generated, subset of all possible faulty nodes ( for some MANETs, all possible faulty nodes ), such that the number of faulty nodes did not exceed the bound $\sigma$ for the graph to be diagnosable. The results we obtained are illustrated in Fig. 3.5. We can see that the communication complexity of Dynamic-DSDP is linear.

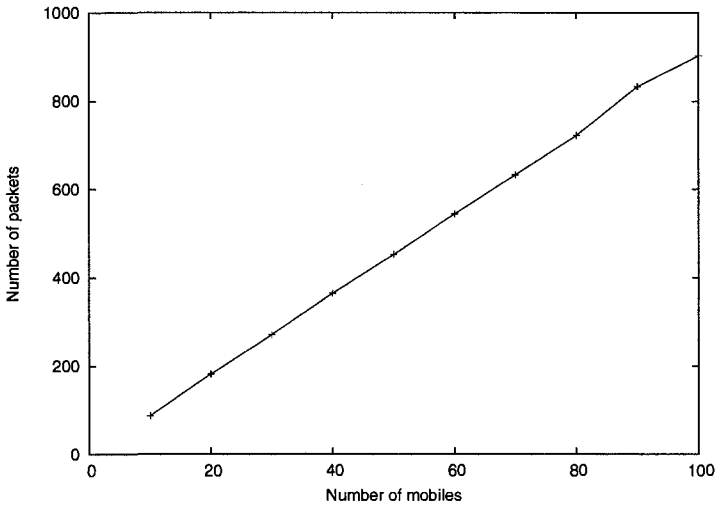


Figure 3.5: Communication Complexity for Dynamic-DSDP

When exposing Static-DSDP to the same set of tests we obtained the results provided in Fig. 3.6. This figure allows us to clearly see that the communication complexity of Static-DSDP is of a higher order than that of Dynamic-DSDP. Communication complexity is of significant importance in MANETs as the energy consumed by a mobile is directly proportional to the amount of traffic it generates or receives. Thus, a reduction in the number of packets required to diagnose a given network would significantly extend its life-span. These results show the energy efficiency of our protocol when compared to that of Chessa and Santi [8].

The stark difference between the results obtained for both protocols can be directly attributed to the upper bound placed on the maximum number of responses a node can generate to the tests it receives in Dynamic-DSDP. In order to illustrate this further, we have

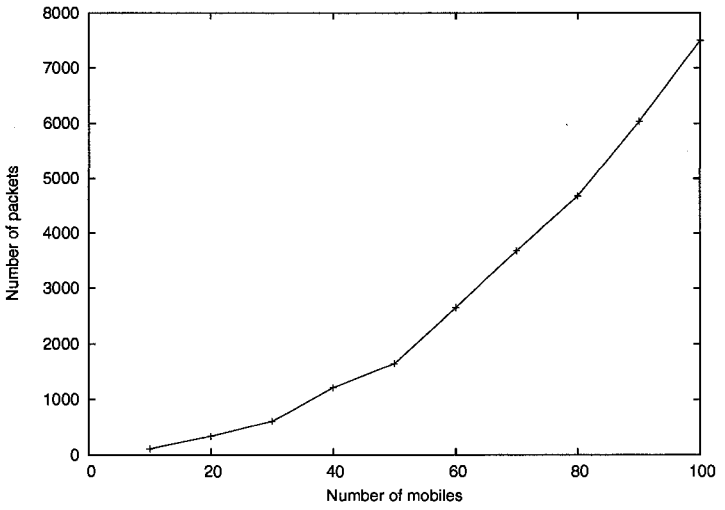


Figure 3.6: Communication Complexity for Static-DSDP

provided Fig. 3.7, which compares the average number of response messages generated by both protocols for all of the tested graphs. The use of a ST protocol by Dynamic-DSDP as a means of data dissemination in a network is also a contributing factor to the reduction in the number of overall packets needed to diagnose a given network. While the use of a flooding protocol for data dissemination in Static-DSDP does reduce the time required to diagnose a given network, communication efficiency should be given precedence over time efficiency for the sake of energy conservation.

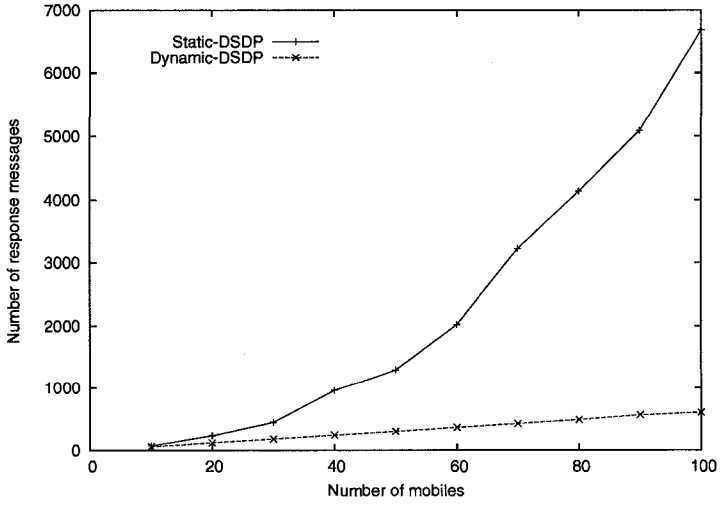


Figure 3.7: Comparison Between Response Messages in Dynamic/Static-DSDP

### 3.5.2 Efficiency with regards to the number of faults

In this simulation scenario, we sought to illustrate the time and communication efficiency of Dynamic-DSDP as a function of the number of faults that occur in a given network. We used one of the previously generated graphs, specifically the graph with 80 mobiles, which has a connectivity of 30. We chose random sets of faulty mobiles in which the number of faulty mobiles ranged from 2 to 28. The results of this configuration are shown in Fig. 3.8 and Fig. 3.10, which illustrate both the time efficiency of Dynamic-DSDP and communication efficiency as a function of the number of faults respectively. As expected, the time required to diagnose the MANET increases on average, even though the order of increase is relatively small (0.5 ms). The change in the average time required to diagnose the network is a result of the change in the depth of the ST constructed to disseminate the global view because the testing phase is not affected by the number of faults. This increase in time complexity is counterbalanced by the decrease in the number of packets needed to diagnose the MANET (Fig. 3.10).

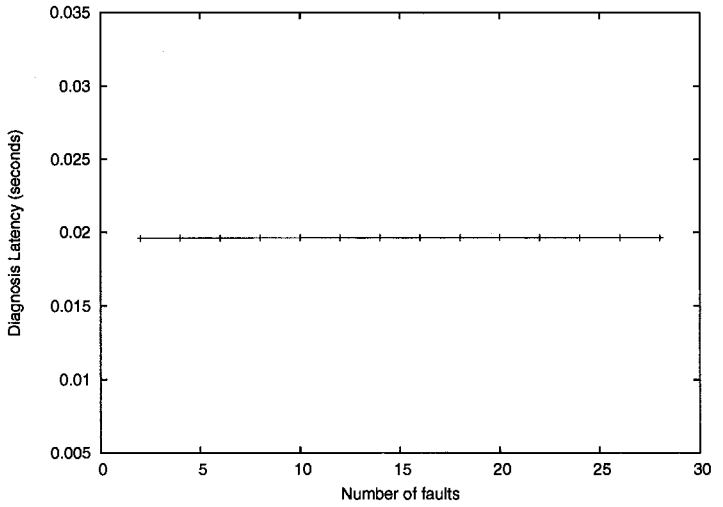


Figure 3.8: Efficiency of Dynamic-DSDP with regards to time

As the tree depth changes as a result of an increase in the number of faulty nodes, so does the number of packets required to disseminate the global view, which decreases as the number of nodes participating in the ST decreases (faulty nodes do not participate in the dissemination phase). This is illustrated in the results shown in Fig. 3.10.

When subjecting Static-DSDP to the same scenario, we obtained the results illustrated

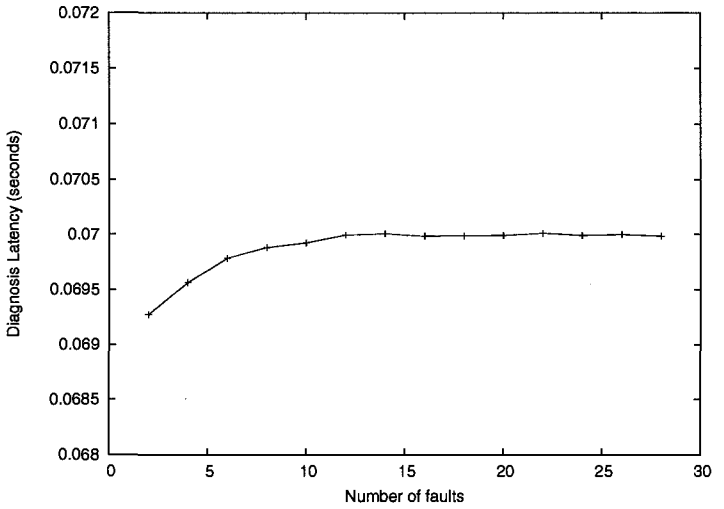


Figure 3.9: Efficiency of Static-DSDP with regards to time

in Figs. 3.9 and 3.11. Fig. 3.9 illustrates time efficiency and Fig. 3.11 illustrates communication efficiency. From Fig. 3.10 we can see that time efficiency is on average approximately the same for both Static/Dynamic-DSDP, but we also notice from Fig. 3.11 that the average number of packets required to diagnose the network increases. This increase is due to the change in the average number of packets, which, as in Dynamic-DSDP, is attributed to the protocol used in disseminating the global view. However, we notice an increase in the number of average packets required to disseminate the global view in Static-DSDP since it is a flooding-based protocol.

The flooding itself is not the main reason for which the number of packets increases. Rather, the way in which it is used allows the faulty nodes to participate in the disseminating phase while being ignored by the rest of the fault-free mobiles, thereby increasing the number of overall packets.

Finally, we conducted a set of simulations to illustrate the time and communication scalability of both Static-DSDP and Dynamic-DSDP when considering large MANETs. The results of these simulations are tabulated in Table 3.4. We can clearly see the effect that the lower order of communication complexity has on the number of packets generated in large networks and the advantage that Dynamic-DSDP presents over Static-DSDP with regards to communication complexity. Another observation that can be made from these results is the time efficiency of Static-DSDP with regard to large networks, where it

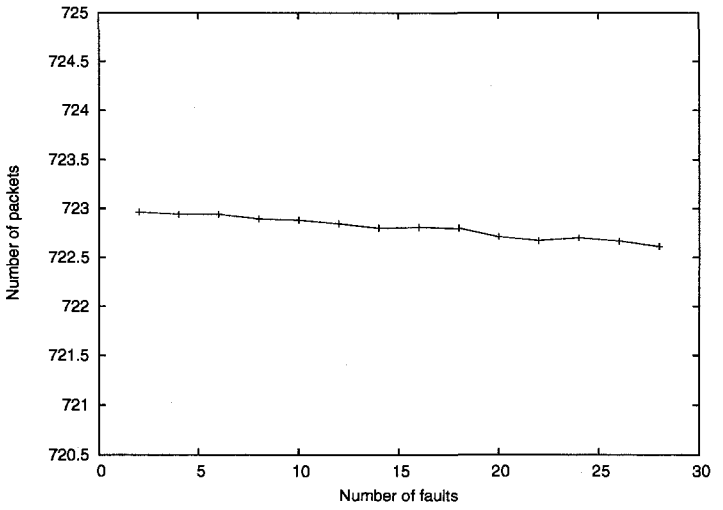


Figure 3.10: Efficiency of Dynamic-DSDP with regards to number of packets

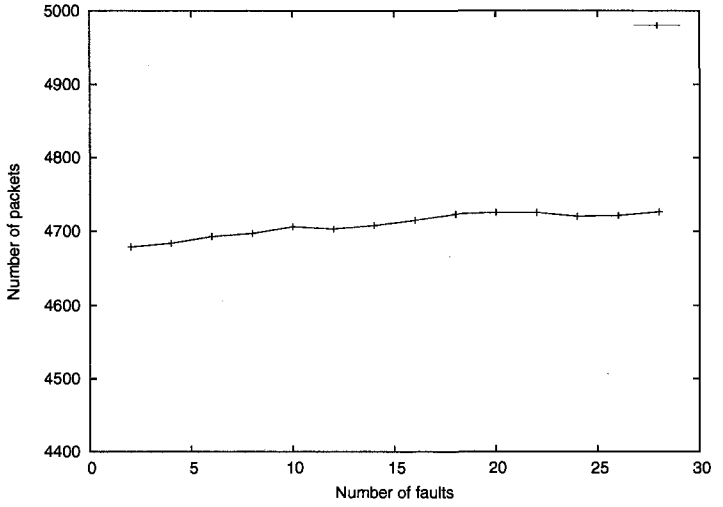


Figure 3.11: Efficiency of Static-DSDP with regards to no. of packets

outperforms Dynamic-DSDP in terms of time efficiency in an order of more than 40 ms. If we put these results into context, we will find that since the proposed protocols will be used in MANETs, which are known to be energy-limited, it would be preferable for a proposed protocol to be communication-efficient. In other words, it would be better for it to be energy-efficient as opposed to time-efficient, which is what Dynamic-DSDP tries to achieve by limiting the number of overall response messages and using the ST protocol in the dissemination phase of the protocol.

### 3.5.3 Discussion and comparison with related work

We believe that the design of an efficient DSDP for MANETs should not exclude from the design goals the reduction of the communication complexity, i.e. the number of exchanged messages. In fact, one drawback of Static-DSDP is the use of flooding, which result in redundant broadcasts. These re-broadcasts can cause serious contention and collision problems, especially in resource-constrained (e.g., power, bandwidth) MANETs [31]. Moreover, a DSDP which exchanges a large number of messages can consume a relatively large amount of energy (and also time) and may not be suitable in energy-constrained MANETs.

Our DSDP uses a ST-based disseminating phase, where the ST is built during the diagnosis session. This constitutes the first main difference between our work and that of Chessa and Santi. As one can easily deduce from Table 3.4, the communication complexity of Dynamic-DSDP largely outperforms that of the self-diagnosis algorithm developed by Chessa and Santi. However, this improvement is counterbalanced by an increase in diagnosis latency. Nevertheless, we believe that Dynamic-DSDP will on the average outperform Chessa and Santi's algorithm.

| **No. mobiles** | **Dynamic-DSDP** | | **Static-DSDP** | |
|---|---|---|---|---|
| | **No. Packets** | **Latency (s)** | **No. Packets** | **Latency(s)** |
| 500 | 6066 | 0.1497 | 41241 | 0.0911 |
| 600 | 7260 | 0.1872 | 45923 | 0.0932 |
| 700 | 8468 | 0.1399 | 63572 | 0.0938 |

Table 3.4: Simulations results for large MANETs.

The second main difference between Dynamic-DSDP and Chessa and Santi's DSDP resides in the gathering phase. In [8], Chessa and Santi assumed that every mobile should reply to any test request it receives. In Dynamic-DSDP, each mobile is required to respond to exactly $\sigma + 1$ test requests. This is feasible since we are dealing with $\sigma$-diagnosable MANETs. In fact, by replying to only $\sigma$+1 neighbors and given that at most $\sigma$ neighbors might be faulty simultaneously, the mobile should be able to inform at least one fault-free neighbor about its status. Furthermore, the assumption concerning the existence of an initiator mobile that will initiate the construction of the ST and on which is based the

disseminating phase can be easily relaxed by asking $\sigma + 1$ mobiles to start this step. Since up to $\sigma$ mobiles can be faulty at the same time, it follows that the fault-free one will be considered as the root of the ST. In the case that there is more than one fault-free node that initiated the construction of the ST, only one node should be elected as the root of the ST. This can be done by assuming that all mobile IDs are ordered and hence the mobile with the smallest ID will be considered as the root of the ST. The design of such an approach is beyond the scope of this paper and is a matter of ongoing research.

# Chapter 4

# An Adaptive Distributed Self-Diagnosis Protocol for MANETS

In this chapter, we present Adaptive-DSDP a distributed self-diagnosis protocol (DSDP) that solves the diagnosis problem in MANETs using an Adaptive spanning tree, which is constructed at the beginning of the deployment of the MANET. Adaptive-DSDP is based on the comparison approach and comprises four main phases: a self-maintenance phase, a testing and gathering phase, a self-repairing of the spanning tree phase, and a dissemination phase.

Upon deployment of the MANET a spanning tree is created, the spanning tree is maintained by continuing to dynamically adapt to the movement of the nodes so that the spanning tree is functional at the time of the diagnosis, this occurs during the self-maintenance phase. During the testing phase, each mobile tests its neighbors by sending them a test task, while during the gathering phase, each mobile collects its neighbors outputs and diagnoses them either as faulty or fault-free by comparing these results. During the self-repairing of the spanning tree phase, every node that finds itself disconnected from the spanning tree as the result of an orphaned or faulty parent reconnects to the spanning tree through another non-faulty node. Finally, during the last phase, i.e. the dissemination phase, all diagnostic information throughout the network is disseminated in order to ensure that each mobile will have a global view of the network status, i.e. each fault-free mobile correctly diagnoses the state of all the mobiles in the system. The system, fault and diagnosis models we use in our design and analysis of this protocol are the same as those presented in the previous chapter

in sections 3.1 and 3.2.

The organization of this chapter is as follows: the Adaptive-DSDP itself is described in Section 4.1. In Section 4.2, we analyze the time and communication complexities of our diagnosis protocol and provide a proof of correctness. The results of our simulation of our protocol is presented in section 4.3.

## 4.1 An Adaptive Distributed 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 then adapted to any faulty situation that might affect any of its internal nodes. The ST is self-maintained connected while the mobiles are moving, and it is used in order to disseminate the local diagnosis views throughout the network.

The protocol proceeds in four phases: the *self-maintaining* phase, the *testing and 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.

### 4.1.1 Self-Maintenance of the Spanning 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 section 3.3.3, or by using a more efficient protocol as described in [2]. A minimum ST would also be more appropriate in this case. Once Configured, the ST should be maintained connected while the mobile are moving. That is, when a parent migrates out of 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 itself 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 as a new parent with the lowest depth.

Let $Children_u$ denote the children of mobile $u$ in the ST at a given point in 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 in the following page 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 is 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 the second message can be piggybacked with the first one.

In the second scenario, the procedure RECONNECTTOST, presented in the next page, 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 (line 2-4). The term $\alpha_0$ has no impact on this stage since $F_u = 0$, but will be useful during the self-repairing phase.

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

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 on the following page, and it describes the steps performed by the mobile $u$. Handling REMOVECHILD and ADDCHILD are trivial, and need no more clarifications. However processing messages of type RECONNECT needs to be clarified. Note that the terms $\alpha_1$(line 6), $\alpha_2$ and $\alpha_3$(line 8 and 14) have no impact since $F_u = 0$, and will be pertinent only for the self repairing phase.

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

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 a new neighbor, say $u$, that is connected to the ST. When mobile $u$ receives the RECONNECT message, two scenarios might happen. 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 13). 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).

```
Procedure SELFMAINTAININGPHASE () {
    do {
1.    Receive(msg); // received from mobile v ∈ 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) AND(v ∉ F_u))   [α1]
            Children_u = Children_u ∪ {v};
7.      RECONNECT : // msg = < RECONNECT, v >
8.        if (ConnectedToST == TRUE AND(v ∉ F_u))   [α2]
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) AND(v ∉ F_u)) {   [α3]
15.         ConnectedToST == FALSE;
16.         Parent_u = v;
17.         _rb(< ADDCHILD, u, v >, < ADDPARENT, u >);
            }
      }
    }
}
```

Handling ADDPARENT messages requires also some explanation. 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 receiver 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 him by transmitting an ADDCHILD message (line 17). Also, 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.

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., $\alpha_0$ (line 1 of procedure RECONNECTTOST), $\alpha_1$, $\alpha_2$, and $\alpha_3$ ( 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 $F = 0$. 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 parents or children are faulty. This step is accomplished during the self-repairing phase that is described in Section 4.1.3.

### 4.1.2 Testing and Gathering Phases

Either periodically or once an altered behavior of the MANET is detected, a self-diagnosis session is initiated by 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 a response message $m_v =< RESPONSE, u, i, R_i^v >$ 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 faulty-free based on their outputs, as described in the rule R3 (3.2). 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 all

its neighbors. Hence, we need the disseminating phase during which the mobiles exchange their local diagnostics views, allowing 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.

### 4.1.3 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, i.e., it maintains two sets: $F_u$ the set of faulty neighbors and $FF_u$ the set of faulty-free ones. 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 unique 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 the mobiles know who is faulty and who is not, it turns out to be a self repairing phase rather a self-maintaining one. note that in our implementation $\alpha_0$ (line 1 of procedure RECONNECTTOST), $\alpha_1$, $\alpha_2$, and $\alpha_3$ ( lines 6, 8, and 14 of procedure SELFMAINTAININGPHASE ) are the key instructions since during the self-repairing phase $F_u$ is nonempty, and hence, mobiles will consider only replies transmitted by fault-free neighbors.

To reduce the diagnosis latency of the protocol we attach the local-diagnosis of the

disconnected nodes to the RECONNECT message, so that all of its neighbors become aware of its local diagnosis. This creates a sub broadcast domain within the spanning tree prior to the repair of the spanning tree, which could allow the dissemination of redundant diagnosis that would add to the communication complexity of the protocol. We resolve the added communication complexity by comparing the local diagnosis received by a node to its own and dropping it, if it is redundant.

### 4.1.4 Disseminating Phase

The disseminating phase starts when all connected leaves of the ST, i.e., nodes without children, send their local diagnosis views to their parents, using a message of type LOCALDIAGNOSTIC. 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 sub-tree rooted at $u$. It then transmits the aggregated diagnostic message to its parent in the ST, and so on. If a node receives a LOCALDIAGNOSTIC or a RECONNECT message, after it has sent its own local diagnosis view to its parent, it will forward it to its parent as long as the local diagnosis data in it isn't redundant and adds to its own.

The *initiator* collects all the local diagnostic, and it disseminates the final message $m_{initiator} =<$ GLOBALDIAGNOSTIC$, F_{initiator}, FF_{initiator} >$ down the tree to all the 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.

## 4.2 Protocol analysis of Adaptive-DSDP

In the following, we prove the correctness of Adaptive-DSDP as a DSDP and then we go on to analyze the communication and time complexities of Adaptive-DSDP. The total number of 1-hop reliable broadcast exchanged during a self-diagnosis session refers to the communication complexity. Whereas, the time between the start and the end of the diagnosis session denotes the time complexity.

### 4.2.1 Proof of correctness of the Adaptive-DSDP protocol

In this section, we will first prove that Adaptive-DSDP provides 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 $\sigma$-diagnosable MANET. Then we will analyze its communication and time complexities. 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 the gathering phase satisfies the correct local diagnosis property. 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.

**Lemma 7.** ***(Self-Maintenance Correctness)*** *Let $G = (V, L)$ denote the communication graph of a diagnosable MANET. 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 diagnosing phase starts. Let $N_d(u)$ denote the set of mobiles given by: $N_1(u) = N(u)$ and $N_{d+1}(u) = \left( \bigcup_{v \in N_{d(u)}} N_d(v) \right) - N_d(u)$. Note that $d$ is bounded for any graph. Let $\bar{d}$ denote this bound on $d$. It follows that for any mobile $u$, $\bigcup_{d=1...\bar{d}} N_d(u) = V$. 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 $u \in N_{\bar{d}}(initiator)$ then $N(u) \subseteq \bigcup_{d' \leq \bar{d}} N_{d'}(u)$.

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 lets show that any mobile in $N_{d+1}(initiator)$ will be reconnected to the 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 it 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 $\sigma$-self-diagnosable, then in the worst case every mobile has at least one fault-free neighbors from which it will be reconnected to the ST. ■

The lemma 8 shows the correct local diagnosis that each mobile should satisfy.

**Lemma 8.** ***(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 8 follows logically from rules R1-R4 of the fixed topology comparison protocol ( see section 3.2 ). 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 timeout $T_{out}$ 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 9.** ***(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 7. Once repaired, the ST is used to dissemi-

[1]See $\alpha_0$, in line 1 of the procedure RECONNECTTOST, and $\alpha_1$, $\alpha_2$ and $\alpha_3$ in lines 6, 8, and 14 of the procedure SELFMAINTAININGPHASE in Section 4.1.1

nate the local diagnostic views as described in Section 4.1.4. The correctness proof of the disseminating phase is given by Lemma 10.

**Lemma 10.** ***(Correct Dissemination)*** *Let $\widetilde{F}$ be the set of faulty mobiles such that $|\widetilde{F}| \leq \sigma$. 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 dissemination phase terminates in a finite time.*

*Proof:* The proof follows logically from that of lemmas 7, 8, and 9. In fact, by lemmas 7 and 9 we know that the tree initialized with the MANET spanning all fault-free nodes. Given that the mobiles are able to verify whether they 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's local diagnostics, and transmits a unique diagnostic message to the upper layer in the ST in a finite time. The *initiator* receives the local diagnosis 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 4 states the correctness of our Adaptive-DSDP and its proof follows from lemmas 7, 8, 9, and 10. In fact, by Lemma 8 we showed that the fault-free status of any mobile is known to all its fault-free neighbors. In lemmas 7 and 9, we proved that all fault-free mobiles are connected to each other via the ST, and that no faulty mobile is part of this tree. Finally, Lemma 10 shows that the mobiles can transmit their local diagnostic views upward the tree to the *initiator*, which aggregates it to 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 of the fault status of the MANET.

**Theorem 4.** ***(Adaptive-DSDP Correctness)*** *Adaptive-DSDP is correct and complete, i.e., the output of Adaptive-DSDP at each fault-free mobile $u$ satisfies $F_u = \widetilde{F}$, where $\widetilde{F}$ and $F_u$ denote, respectively, the actual set of faulty mobiles in the MANET, and the set of faulty mobiles output by Adaptive-DSDP.*

### 4.2.2 Communication complexity of the Adaptive-DSDP protocol

Let $n$ and $d_{max}$ denote, respectively, the total number of mobiles and the maximum of the node degrees.

**Theorem 5.** *The communication complexity of Adaptive-DSDP is* $O(nd_{max})$.

*Proof:* Number of TEST messages = $n$, since all mobiles will generate at most one test message. Number of RESPONSE messages = $nd_{max}$, given that each mobile responds to all its neighbors test messages. In the worst-case mobiles have $d_{max}$ neighbors. Number of RECONNECT messages= $n-2$. Note that the worst-case [2] scenario occurs when $n-2$ nodes discover that they have lost their parent. In this case, all of them will send this message. Number of ADDPARENT and ADDCHILD messages = $n-2$. First, note that these two messages can be merged into a single message. The maximum number of such messages occurs in the same worst-case scenario. Hence, the $n-2$ mobiles disconnected from their faulty parent will choose a new parent. Number of LOCALDIAGNOSTIC messages = $n-1$. Each mobile, excluding the initiator, sends one message of this type. Finally, number of GLOBALDIAGNOSTIC messages = $n-1$. In the worst-case scenario the depth of the tree is $n-1$. Hence, $n-1$ messages are needed to broadcast the complete diagnosis down the tree. It follows that the total number of messages exchanged is $n(d_{max}+5)-6$, hence, the communication complexity is $O(nd_{max})$. ■

### 4.2.3 Time complexity of the Adaptive-DSDP protocol

Let $G = (V, Lt)$ be the communication graph, and let $\triangle_G$ and $d_ST$ denote, respectively, the diameter of $G$ and the depth of the spanning tree used by the disseminating phase. The time complexity will be expressed in term of the following two bounds: $T_{gen}$ is an upper bound to the elapsed time between the reception of the first diagnostic message and the generation of the test request, and $T_f$ is an upper bound to the time needed to propagate a message.

[2] The worst-case scenario can be described as follows. Let $u$ be a child of the *initiator*, and assume that all the remaining mobiles, i.e., $n-2$, are children of $u$. Now, suppose that $u$ is faulty. All $u$'s neighbors will transmit a RECONNECT message for their neighbors asking to be reconnected to the spanning tree. That is, $n-2$ RECONNECT messages are sent in this case.

**Theorem 6.** *The time complexity of Adaptive-DSDP is* $O(\triangle_G T_{gen} + (2d_{ST} + n - 1)T_f + T_{out})$.

*Proof:* The last mobile to generate its test message will do so at most in $T_{gen}$ since the first diagnostic message was received. Thus, in at least $\triangle_G T_{gen}$ all non-faulty mobiles generate their test requests. Any fault-free mobile diagnoses at least one fault-free neighbor in at most $T_{out}$ time. Hence, the testing and gathering phases requires at most $\triangle_G T_{gen} + T_{out}$. The disseminating phase starts at mobiles with no children, which have to transmit their local diagnostic views to their parents. Once a parent collects all its children's local diagnostic views, it forwards the aggregated diagnostic views, along with its own local diagnostic view, to its parent, and so on up the tree. In at most $d_{ST}T_f$, the *initiator* has collected all diagnostic views and disseminates the global diagnostic view that reaches the farthest mobile in at most $d_{ST}T_f$ . Hence, the disseminating phase requires $2d_{ST}T_f$ time to complete. The time taken by the self-repairing phase is as follows. The worst-case scenario occurs when the *initiator* has only one child, say $u$, and all the $n-2$ remaining mobiles are children of $u$. Also, assume that $u$ is faulty and that the new spanning tree constructed by the self-repairing phase is a tree of depth $n-2$. The mobile, say $v$, the closest to the *initiator* will discover that its parent $u$ is faulty, and will generate a RECONNECT message. This message reaches the *initiator* after $T_f$ time and triggers the transmission of an ADDPARENT message. The message is then propagated to all the disconnected mobiles, and will reach the farthest descendant in at most $(n-2)T_f$. Thus, the self-repairing phase requires at most $T_f + (n-2)T_f$ time to reconnect all fault-free mobiles to the ST. The thesis follows by observing that Adaptive-DSDP comprises the testing and gathering phases, a self-repairing phase, and a disseminating phase. ■

## 4.3 Simulation Results

As in the case with Dynamic-DSDP, we chose to conduct simulations of our proposed protocol to further characterize it and measure its efficiency. We conducted an extensive set of simulations using the $NS-2$ simulator. For simulation purposes we used the same set of communication graphs we used in our simulation of Dynamic-DSDP, so the results could be compared. The communication graphs $G$, were made up of a number of randomly dis-

| Parameter | Value |
|---|---|
| kG | 4 |
| Number of Nodes in Network | 10 - 100 |
| Simulation Time | 150s |
| Transmission Range | 150m |
| Topology Size | 500 x 500 m |
| Propagation Delay | 25 us |
| Link Layer Queue Length | 10000 |
| Propagation Model | TwoRayGround |
| Antenna Model | OmniAntenna |

Table 4.1: Simulations environment variables for Adaptive-DSDP

tributed nodes, with a known minimum common connectivity $k_G$ and the number of nodes ranging from 10 to 100. In both scenarios we used the same set of simulation parameters for $NS-2$; these are summarized in Table 4.1.
The performance of Adaptive-DSDP will be compared to that of both Static-DSDP, and the Dynamic-DSDP protocol we proposed in the previous chapter.

### 4.3.1 Efficiency with regards to the number of packets

Here we subjected Adaptive-DSDP to an extensive set of simulations using the previously generated set of graphs $G$, and a large, randomly generated, subset of all possible faulty nodes ( for some MANETs, all possible faulty nodes ), such that the number of faulty nodes did not exceed the bound $\sigma$ for the graph to be diagnosable. The results we obtained are illustrated in Fig. 4.1.

If we compare the results in 4.1 to the previous results obtained from simulating both Static-DSDP and Dynamic-DSDSP, as shown in 4.2, we notice that the communication efficiency of Adaptive-DSDP is better than that of Static-DSDP; but not as good as that of Dynamic-DSDP. This is because, as in Static-DSDP, Adaptive-DSDP requires that all the nodes reply to all tests sent to them during the diagnosis stage; but unlike Static-DSDP Adaptive-DSDP uses a pre-constructed spanning tree for the dissemination phase of the protocol which gives it an edge over Static-DSDP.

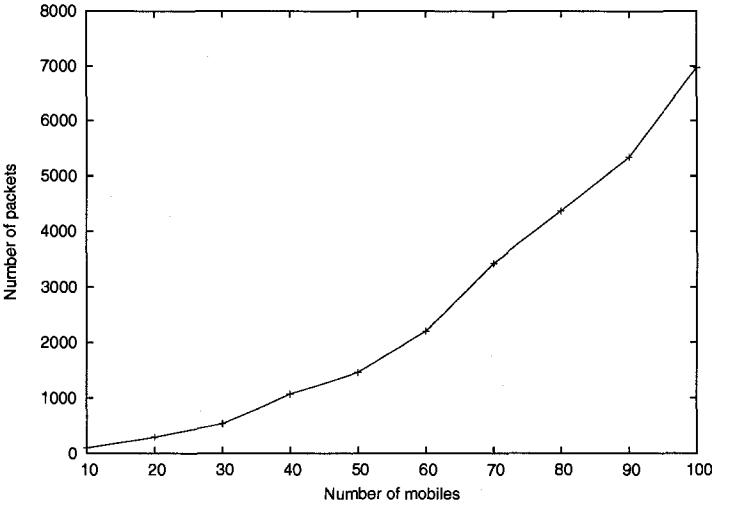


Figure 4.1: Communication Complexity for Adaptive-DSDP

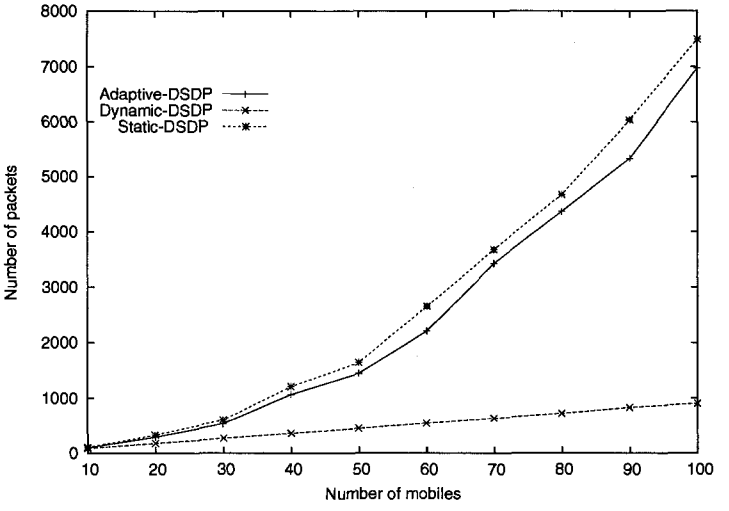


Figure 4.2: Comparison of Communication Complexity between Dynamic/Adaptive and Static DSDP

Given the relationship between the communication complexity of a protocol and the amount the energy consumed by a mobile, we can note that Adaptive-DSDP isn't as energy efficient as Dynamic-DSDP even though it is a more efficient alternative to Static-DSDP. The placement of Adaptive-DSDP's communication efficiency in between its two predecessors is a direct result of its use of a hybrid local-diagnosis dissemination protocol, which is neither a spanning tree ( if their are faults ) nor a simple flooding paradigm.

### 4.3.2 Efficiency with regards to the number of faults

In this simulation scenario, we sought to illustrate the time and communication efficiency of Adaptive-DSDP as a function of the number of faults that occur in a given network. We used one of the previously generated graphs, specifically the graph with 80 mobiles, which has a connectivity of 30. We chose random sets of faulty mobiles in which the number of faulty mobiles ranged from 2 to 28. The results of this configuration are shown in Fig. 4.3 and Fig. 4.5, which illustrate both the time efficiency of Adaptive-DSDP and communication efficiency as a function of the number of faults respectively.

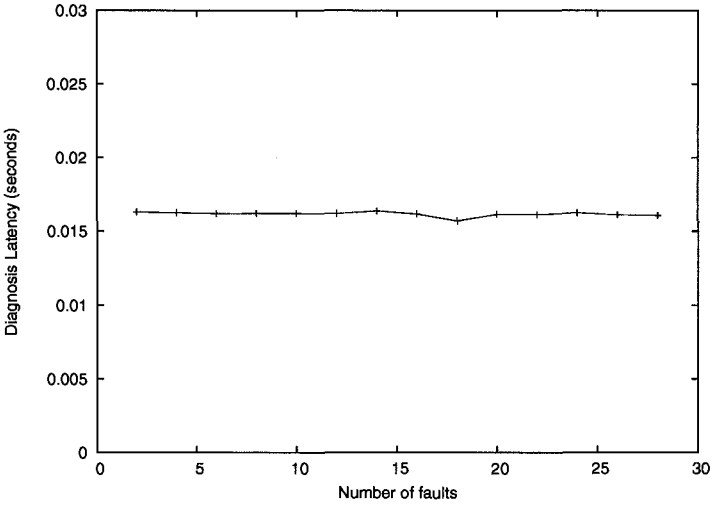


Figure 4.3: Efficiency of Adaptive-DSDP with regards to time

As expected, the time required to diagnose the MANET doesn't really change much as it really is not dependent on the number of faults; but rather on the placement of the faults in the spanning tree. Since the tree is constructed prior to the local diagnosis, any change in the diagnosis time is affected not by the number of faults but more by the number of orphaned nodes that result from a parent node becoming faulty. The delay resulting from nodes becoming disconnected and orphaned is limited in itself by piggy backing the local diagnosis of those orphaned nodes onto the RECONNNECTMSG they send to reconnect to the spanning tree.

When comparing the results obtained from this scenario, in Fig. 4.3, to those obtained from subjecting Dynamic-DSDP to the same scenario, which are shown in Fig 3.8 we notice that on average Adaptive-DSDP takes less time to diagnosis a given network.

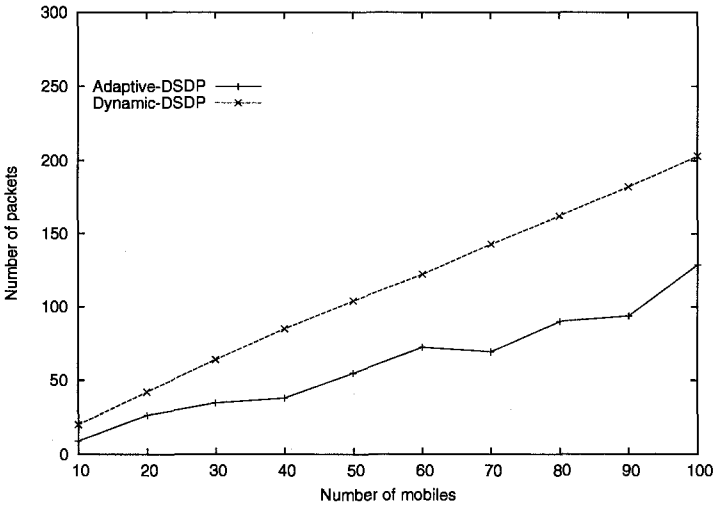


Figure 4.4: Comparison between the number of packets used in the Dissemination phase of both Dynamic/Adpative - DSDP

This is because the spanning tree which is the basis for diagnosis dissemination in both protocols is already constructed prior to the initiation of the diagnosis, when using Adaptive-DSDP, this results in less packets and time being consumed in the disseminating phase of the diagnosis, this comparison is illustrated in Fig. 4.4

As with the time efficiency of Adaptive-DSDP, its communication efficiency as a function of the number of faults isn't dependant on the number of fault but rather on their placement. This, again, is because the number of packets exchanged during the testing phase of the diagnose of a network is fixed ( test and response messages ). The only difference is in the number of packets required to diagnose a network, with different sets of faults, would result from a difference in the number of packets required to reconnect the orphaned nodes back to the spanning tree after they end their own local diagnosis. This gives us the results illustrated in Fig. 4.5.

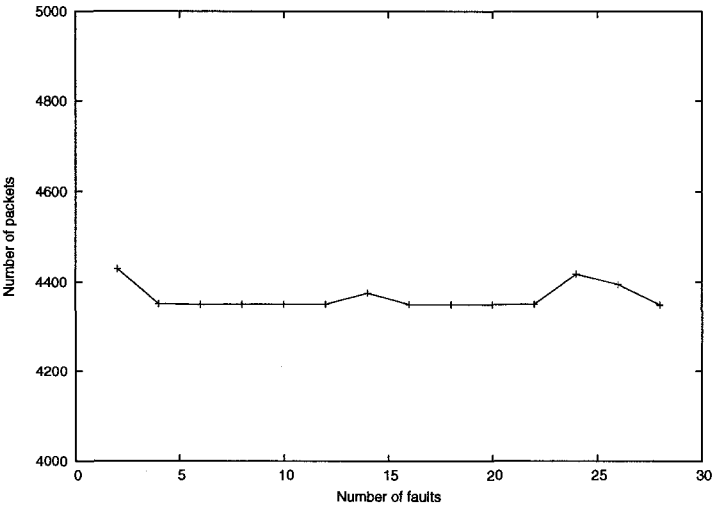


Figure 4.5: Efficiency of Adaptive-DSDP with regards to number of packets

### 4.3.3 Discussion and comparison with related work

In this chapter, we presented a second DSDP, called Adaptive-DSDP, which uses a spanning tree that is pre-configured with the MANET and then repaired during the diagnosis session. The communication complexities of our Adaptive-DSDP and that of Dynamic-DSDP largely outperform that of static-DSDP ( see Table 4.2 ).

| **Algorithm** | **Messages** | **Diagnosis latency** |
|---|---|---|
| Static-DSDP | $nd_{max} + n(n+1)$ | $\triangle_G T_{gen} + \triangle_G T_f + T_{out}$ |
| Dynamic-DSDP | $nk_G + 3n - 1$ | $\triangle_G T_{gen} + 3d_{ST} T_f + 2T_{out}$ |
| Adaptive-DSDP | $nd_{max} + 5n - 6$ | $\triangle_G T_{gen} + (2d_{ST} + n - 1)T_f + T_{out}$ |

Table 4.2: Comparison of time and communication complexities of relevant DSDPs

However, this improvement is counterbalanced by an increase in the order of the diagnosis latency. Nevertheless, we believe that both Dynamic-DSDP and Adaptive-DSDP will on the average outperform Static-DSDP. Moreover, our Adaptive-DSDP will perform better than both. Note that in the worst-case the depth of the spanning tree is $n - 1$, i.e.,

$d_{ST} = n - 1$, and hence, both Dynamic-DSDP and Adaptive-DSDP provide the same time complexity in this case. However, in other cases Adaptive-DSDP is expected to perform better since the repairing of the spanning tree will take less time than building a new one.

# Chapter 5

# Conclusion and Future Work

In our work we present two distributed self-diagnosis protocols ( DSDPs ). Our focus was on the improvement of the communication efficiency and time-efficiency of the diagnosis ( *fault detection* ) protocols, which inevitably improve the protocols power efficiency and compliance with both the diagnosis and Ad-hoc system model.

In this chapter, we will summarize our contributions to the field of fault detection in MANETs and outline the possible directions for future research.

## 5.1 Summary of our Contributions

The purpose of DSDPs, is to allow all fault-free mobiles to identify the fault status of all the mobiles in its MANET within a bounded time. In this thesis we presented two DSDPs, both of which use the comparison approach in the diagnosis of their neighbors and depend on a form of spanning tree in the dissemination of their local diagnosis, below is a summary of the features of both protocols.

(a) **Dynamic-DSDP :**

In Dynamic-DSDP the mobiles use the comparison approach in diagnosing their neighbors, while only replying to $\sigma + 1$ test requests, this significantly reduces the communication complexity of the protocol. After all the mobiles complete their local diagnosis the initiator proceeds to create the spanning tree, which is subsequently

used to collect the local diagnosis from the mobiles and then disseminate the global diagnosis back to them. The use of the spanning tree does help in the improvement of the communication complexity of the Dynamic-DSDP; but its construction during the diagnosis phase increases the latency of the diagnosis, which is a drawback we addressed in Adaptive-DSDP at the expense of the communication complexity.

(b) **Adaptive-DSDP :**

As in Dynamic-DSDP , Adaptive-DSDP uses the comparison approach in diagnosing the mobiles; but in Adaptive-DSDP the spanning tree is created at the beginning of the deployment of the MANET and the tree structure adapts dynamically to accommodate the movements of the mobiles. After the mobiles end their local diagnosis they check if they have become disconnected, as a result of a faulty or orphaned parent, before proceeding to send their local diagnosis to the initiator.

If a mobile node realizes that it is disconnected from the spanning tree after its local diagnosis it needs to know the fault-status of all its neighboring nodes in order to be able to reconnect to the spanning tree through a non-faulty mobile. This means that all mobiles have to reply to all the tests sent to them, which increases the communication complexity of Adaptive-DSDP, but still guarantees an improved diagnosis latency.

## 5.2 Further work

In our future research, we would be interested in investigating dynamic fault identification solutions that will be able to tolerate the occurrence of faults during the diagnosis session. We are also interested in investigating a self-diagnosis protocol that would be more appropriate for wireless sensor networks, which can only be achieved by further reducing the communication overhead of the DSDP's. Another area of interest would the use of our protocols in non-deterministic diagnosis of MANET models, which could allow us to improve their efficiency.

# References


[1] A. Adya, P. Bahl, R. Chandra, and L. Qiu. Architecture and Techniques for Diagnosing Faults in IEEE 802.11 Infrastructure Networks. In *Proc. of the 10th Int. Conf. on Mobile Computing and Networking (MobiCom '04)*, pages 30–44, 2004.

[2] 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. In *J. Parallel Distrib. Comput.*, 2003.

[3] A. Bagchi and S. L. Hakimi. An Optimal Algorithm for Distributed System Level Diagnosis. In *FTCS*, pages 214–221, 1991.

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

[5] F. Barsi, F. Grandoni, and P. Maestrini. A Theory of Diagnosability Without Repairs. *IEEE Trans. on Computers*, C-25:585–593, June 1976.

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

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

[8] 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.

[9] A.T. Dahbura. System-level diagnosis: A perspective for the third decade. Technical report, AT&T Bell Laboratory Report, Concurrent Computations: Algorithms, Architecture, and Technology, S. Tewksbury, B. Dickinson, S. Schwartz, Eds. New York: Plenum, 1988.

[10] Alberto Sangiovanni-Vincentelli Farinaz Koushanfar, Miodrag Potkonjak. Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks. *Sensors, 2002. Proceedings of IEEE*, 2, 2002.

[11] Felix C. Gartner. Fundamentals of Fault Tolerant Distributed Computing in Asynchronous Environments. *ACM Computing Surveys*, 31(1):1–26, March 1999.

[12] S.L. Hakimi and K.Y. Chwa. Schemes for Fault Tolerant Computing: A Comparison of Modularly Redundant and $t$-Diagnosable Systems. *Inform. Contr.*, 49:212–238, June 1981.

[13] F. Harary. *Graph Theory.* Reading, MA, Addison-Wesley, 1972.

[14] 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 Conference*, pages 495–502, Rennes, France. 31 Aug.- 3 Sept., 2004.

[15] S. H. Hosseini, J. G. Kuhl, and S. M. Reddy. A Diagnosis Algorithm for Distributed Computing Systems with Dynamic Failure and Repair. *IEEE Trans. Computers*, 33(3):223–233, 1984.

[16] J. Hao, J.B. Orlin. A faster algorithm for finding the minimum cut in a graph. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 165–174, Orlando, Florida, 1992.

[17] E. Duarte Jr., A. Brawerman, and L. C. P. Albini. An Algorithm for Distributed Hierarchical Diagnosis of Dynamic Fault and Repair Events. In *Proc. of the 7th Int. Conf. on Parallel and Distributed Systems (ICPADS'00), Iwate, Japan*, pages 299–306, 2000.

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

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

[20] S. Kreutzer and S.L. Hakimi. Adaptive Fault Identification in Two New Diagnostic Models. In *Proc. 21st Allerton Conf. on Commun., Cont. and Comput.*, pages 353–362, Univ. of Illinois, Urbana, Ill, 1983.

[21] S. Lee and K.G. Shin. Probabilistic Diagnosis of Multiprocessor Systems. *ACM Computing Surveys*, 26(1):121–139, March 1994.

[22] J. Maeng and M. Malek. A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems. In *Proc. 11th Int. Symp. on Fault-Tolerant Comput.*, pages 173–175, 1981.

[23] M. Malek. A Comparison Connection Assignment for Diagnosis of Multiprocessor Systems. In *Proc. 7th Int. Symp. on Comput. Architecture, New York*, pages 31–35. Association for Computing Machinery Publ., 1980.

[24] E. Pagani and G. P. Rossi. Reliable Broadcast in Mobile Multihop Packet Networks. In *Proc. of the 3rd ACM/IEEE Int. Conf. on Mobile computing and networking*, pages 34–42, New York, NY, USA, 1997.

[25] F.P. Preparata, G. Metze, and R.T. Chien. On the Connection Assignment of Diagnosable Systems. *IEEE Trans. on Electron. Comput.*, 16(6), Dec. 1967.

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

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

[28] Madhavi W. Subbarao. Mobile Ad Hoc Networks for Emerging Preparedness Telecommunications - Dynamic Power-Conscious Routing Concepts. Technical report, National Institute of Standards and Technologies, 2000.

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

[30] T. Anderson and P.A. Lee. *Fault Tolerance Principles and Practice.* Prentice/Hall international, 1981.

[31] K. Viswanath and K. Obraczka. Modeling the Performance of Flooding in MultiHop Ad Hoc Networks. In *Proc. of the Symp. on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'04)*, San Jose, California. July 25-29, 2004.