FAILURE DIAGNOSIS OF DECENTRALIZED DISCRETE EVENT SYSTEMS

by

Rami Ismail Debouk

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2000

Doctoral Committee:
Professor Stéphane Lafortune, Co-Chair
Professor Demosthenis Teneketzis, Co-Chair
Dr. George Barrett, Johns Hopkins University
Professor Pramod Khargonekar
Assistant Professor Nandit Soparkar
Assistant Professor Dawn Tilbury

اللهم صل على محمد وآله ولا ترفعني في الناس درجة إلا حططتني عند نفسي مثلها، ولا تحدث لي عزا ظاهرا إلا أحدثت لي ذلة باطنة عند نفسي بقدرها

الامام زين العابدين (ع)

إلى الحجة المنتظر

عجل الله تعالى فرجه الشريف

إلى أمي

رحمها الله

إلى أبي

حفظه الله

ACKNOWLEDGEMENTS

I have been fortunate to have Stéphane Lafortune and Demosthenis Teneketzis as my advisors. I am indebted to them, not only for supervising my research, but also for being the good friends they have been over the past few years. I know a simple "thank you" would not suffice, but then what would, as my thankfulness to them cannot be expressed in words.

I would like to thank Professors Pramod Khargonekar, Nandit Soparkar, Dawn Tilbury, and Dr. George Barrett for accepting to sit on my committee. Special thanks for George for the many discussions, technical or philosophical, that we had while he was at The University of Michigan.

I cannot forget the help of Linda Cox, the Systems Division Graduate Program Coordinator, and for that I am grateful. I would also like to thank the staff of the Systems Division, in particular Ann Pace, for their help and kindness.

My friends at The University of Michigan have helped me "survive" the life of a graduate student. Nah-Oak, Chris, Tara, and most recently Tudor (who is planning to take over my desk!), thank you for making 4212 EECS a pleasant and enjoyable office to work in. I apologize from my other friends for not mentioning their names, however each and every one of them has my gratitude for her/his support during the last five years.

On a personal note I would like to thank my friend Hiba for the support and encouragement she gave me during the past year.

Finally, I owe my family a lot since without them I would not have been the person I am today. I would like to acknowledge my sister Rima, her husband Riad, and my brother Ramez for being there for me and helping me overcome the hurdles of life. Also, for Haifa, thank you for believing in me. To my mother Ghazwa and father Ismail, words cannot express my gratitude for all your sacrifices. I know that whatever I do is not sufficient to pay you back, nonetheless I will never stop trying. Mother, I am not questioning God's will but I really wish you were present to witness these moments; in any case you are and always will be present in my heart.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

LIST OF APPENDICES viii

CHAPTERS

1 INTRODUCTION 1
- 1.1 The Problem of Failure Diagnosis: Definition and Challenges 1
- 1.2 Approaches to Failure Diagnosis 2
- 1.3 Contribution of the Thesis 6
- 1.4 Organization of the Thesis 8

2 A DISCRETE EVENT SYSTEMS APPROACH TO FAILURE DIAGNOSIS 9
- 2.1 The System Model 9
- 2.2 Notation 10
- 2.3 Definition of Diagnosability 11
- 2.4 The Diagnoser 12
- 2.5 Necessary and Sufficient Conditions for Diagnosability 14
- 2.6 The Extended Diagnoser 15

3 A COORDINATED DECENTRALIZED ARCHITECTURE 20
- 3.1 The Architecture 20
- 3.2 Addressing Communication Delay Issues in the Coordinated Decentralized Architecture 22
- 3.3 Definition of Diagnosability 23
- 3.4 Objective 24

4 PROTOCOL 1 26
- 4.1 Specification of the Protocol 26
- 4.2 Diagnostic Properties of Protocol 1 31
- 4.3 Necessary and Sufficient Conditions for Diagnosability 34
- 4.4 Discussion 36

4.5 Protocol 1D: Relaxation of Assumption **A5** 38

5 PROTOCOL 2 49
5.1 Objective and Assumptions 49
5.2 Specification of the Protocol 51
5.3 Diagnostic Properties of Protocol 2 53
5.4 Necessary and Sufficient Conditions for Diagnosability 61
5.5 Discussion 65
5.6 Protocol 2D: Relaxation of Assumption **A5** 68

6 PROTOCOL 3 78
6.1 Objective and Assumptions 78
6.2 Specification of the Protocol 79
6.3 Diagnostic Properties of Protocol 3 80
6.4 Necessary and Sufficient Conditions for Diagnosability 81
6.5 Discussion 83

7 REFLECTIONS ON THE PROTOCOLS AND THE ARCHITECTURE . 85
7.1 "Performance vs. Complexity" trade-off 85
7.2 Relaxation of Assumption A5 87
7.3 Extension to m Sites 88
7.4 General Thoughts on the Approach 89

8 EXAMPLE: DIAGNOSING A WIRELESS LOCAL AREA NETWORK . 91
8.1 Description of the System 91
8.2 Applying the Diagnostic Methodology to the Wireless LAN 94

9 AN OPTIMIZATION PROBLEM IN SENSOR SELECTION 98
9.1 Introduction 98
9.2 Problem Formulation 100
9.3 Solution 101
9.4 Concluding Remarks 123

10 CONCLUSION 124
10.1 Summary of Contributions 124
10.2 Related Work 125
10.3 Future Research Directions 126

APPENDICES 129

BIBLIOGRAPHY 140

LIST OF TABLES

Table

4.1 Information update rule at the coordinator site (Protocol 1) 30
4.2 Illustration of the application of Protocol 1 36
4.3 Sorting out possible orders at the coordinator site 40
5.1 Information update rule at the coordinator site (Protocol 2) 54
7.1 Comparison of the three protocols 86
7.2 Comparison of the three protocols (continued) 86
8.1 Definition of events for the LAN model 93
8.2 Application of Protocol 2: system executes $ltf, t, t, t, t, \ldots$ 96

LIST OF FIGURES

Figure

1.1 Coordinated decentralized architecture 6
2.1 The system G, left, and the diagnoser G_{d1} for Example 2.4.1 13
2.2 The extended diagnoser G_{d1}^{e} for Example 2.6.1 16
3.1 Coordinated decentralized architecture 21
3.2 The protocols realizing the coordinated decentralized architecture 25
4.1 The extended diagnoser G_{d2}^{e} for Example 4.1.1 28
4.2 The diagnosers for Example 4.3.1 35
4.3 The local diagnosers for Example 4.4.1 37
5.1 The system (left) and centralized diagnoser for Example 5.3.1 59
5.2 The diagnosers G_{d1} (left) and G_{d2} for Example 5.3.1 59
5.3 The system (left) and centralized diagnoser for Example 5.3.2 60
5.4 The diagnosers G_{d1} (left) and G_{d2} for Example 5.3.2 60
5.5 G_{test2} for Example 5.4.1 64
5.6 G_{test2} for Example 5.4.2 65
5.7 G_{test2} for Example 5.5.1 66
5.8 G_{test2} with the new set of projections for Example 5.5.1 67
6.1 G_{test3} for Example 6.5.1 84
7.1 Comparison of Protocols 87
8.1 Partial LAN model for a three vehicle platoon 92
8.2 An $F1$-indeterminate cycle in G_{test3} 95
9.1 The digraph in the case where there are 4 sensors. The notation 12, for instance, is used to denote the set of sensors {1,2} 102
9.2 Policy comparisons for Example 9.3.2 113

LIST OF APPENDICES

APPENDIX

A 129

B 139

CHAPTER 1

INTRODUCTION

1.1 The Problem of Failure Diagnosis: Definition and Challenges

Failure diagnosis is the process of detecting any abnormality in the behavior of a system and isolating the cause or the source of this abnormality. Abnormalities[1] are defined to be deviations from the normal or intended behavior of the system. As technology advances, more performance requirements are imposed on systems and consequently they become more and more complex. Unfortunately, the more complex systems become the more they are subject to errors. Hence, reliability becomes of great concern in such systems and that necessitates the development of systematic approaches to failure diagnosis. Consequently, failure diagnosis modules have become an important entity in the automatic control of large complex systems. Not only diagnosing these systems improves their performance and productivity, but it also protects life and property. For these reasons, approaches to failure diagnosis have been extensively studied in the literature. A brief discussion of these approaches is presented in Section 1.2.

The problem of failure diagnosis encompasses many challenging aspects. Given a dynamic system, the issue of determining the abnormalities that may occur while the system is operating is of a paramount importance. The failures of interest, that is the abnormalities that may deviate the system from its normal behavior, need to be determined. Also of importance is the system model to be used by the diagnosis scheme. The level of abstraction at which the system is modeled is mainly determined by the failures to be diagnosed. The two issues just discussed are modeling issues, nonetheless they are critical in defining the diagnosis problem. Another challenging aspect is the selection of the type of measurements that may be helpful in inferring the occurrences of failures and identify the sensors required

[1]The term failure is often used to denote a complete operational breakdown, however the abnormalities we are considering do not cause such type of breakdown. Such abnormalities are often called faults; we will use the terms fault and failures synonymously in this thesis.

to record these measurements. The physical nature of the system along with some other constraints, e.g., economical and technological, play an important role in the process of choosing the sensors that may be used in the diagnosis task. It is not always the case that all these sensors are indeed needed to achieve the diagnosis task. This raises the issue of sensor selection for failure diagnosis. Another aspect to be considered is that the physical system determines whether the diagnostic information can be generated in a centralized fashion, i.e., all measurements are accessible by one centralized working station which is responsible for generating the diagnostic decision. If that is not the case, the (decentralized) architecture generating the diagnostic information and the diagnostic decisions has to be specified. The type of the architecture is mainly dictated by the physical layout of the system components as well as other organizational constraints of the system. This motivates the need to develop methodologies to account for decentralized information. Finally, an approach for diagnosis has to be determined and that is mainly dictated by the type of failures one is interested in diagnosing along with the type of available sensors.

1.2 Approaches to Failure Diagnosis

The design and implementation of automated diagnostic systems has received considerable attention both in academia and industry. Many approaches, differing in their theoretical framework, have been proposed. Most of these approaches can be divided, from a conceptual point of view, into the following five classes: (i) fault-tree based methods, (ii) analytical redundancy methods; (iii) expert systems and other knowledge-based methods; (iv) model-based reasoning methods; and (iv) discrete event systems (DES) based methods. From an implementation point of view, these diagnostic systems can be classified as off-line or on-line. Off-line diagnosis assumes that the system is in a test-bed and is to be tested for possible prior failures, while on-line diagnosis monitors the operation of the system and attempts simultaneously to detect and isolate failures. The following provides a brief overview of the abovementioned approaches for failure diagnosis. The reader is referred to [31] for a detailed discussion of these and some other existing approaches to diagnosis such as statistical decision theory, control charts, and vibration and noise analysis methods.

Fault-tree methods [25, 26, 47, 49, 50] provide a graphical representation of cause-effect relationships of faults in a system. Starting from an alarm that indicates a system failure event, a fault tree is built by reasoning backwards from the system failure to basic failures that represent the main cause of the failure. This technique requires a huge effort in constructing the trees and cannot handle feedback systems adequately [29, 42].

Most of the approaches for failure diagnosis proposed in the control systems literature [4,

19, 24, 48, 53] are based on analytical redundancy. The analytical redundancy method involves (i) generation of residual signals by comparing predicted values of system variables (from the available mathematical models of the system) with the actual observed values, and (ii) decision and fault isolation by examining the residuals for the likelihood of faults using for instance likelihood ratio functions. Such an approach has the advantage of detecting both abrupt faults and slowly developing faults, however it is very sensitive to modeling errors and measurement noise, and requires an involved computational expenditure for the detailed on-line modeling of the process.

Expert systems methods [42] are suited for systems that are difficult to model. Heuristic knowledge of experts is captured in a set of rules that attempts to associate symptoms of abnormalities to the faults that produce them. This technique is very domain dependent and it may be the case that a considerable amount of time may elapse before enough knowledge is accumulated to develop the necessary set of rules. Moreover, it is difficult to validate an expert system.

Model-based reasoning methods rely, as is the case with the analytical redundancy methods, on the observation and prediction paradigm [11]. They employ a general purpose model of the structure and behavior of the system which are constructed using standard artificial intelligence techniques. The algorithm they employ for diagnosis are also based on some standard artificial intelligence techniques. Applicability of such techniques to dynamic systems has not yet been fully demonstrated, and is still an area of active research.

Many approaches for failure diagnosis using discrete event models have recently appeared in the literature [1, 3, 5, 9, 21, 27, 32, 36, 40, 49, 55, 56]. In [1] the authors use Petri nets to model concurrent alarm indications in large distributed systems. The diagnosis problem is defined as the computation of the most likely history of the net given a sequence of observed alarms. In [49] the authors also use Petri nets to model the system for the purpose of failure detection, while fault trees are implemented for isolation. In [27] the authors study off-line diagnosability of DES. The diagnostic procedure involves issuing off-line a sequence of test commands, observing the resulting outputs, and drawing inferences on the set of possible states that the system could be in. The latter approach is extended in [5] to determine the optimal set of sensors which would ensure testability (which is equivalent to off-line diagnosability of [27]) of a given system and the infimal partition of the state space with respect to which the system is testable given a fixed set of sensors. In [40] the authors suggest a language based approach to failure diagnosis of logical DES. They model the system to be diagnosed by a language that accounts for both the normal and failed modes of operation of the system. A finite state machine, the diagnoser, is at the core of their diagnostic methodology; it is used off-line to analyze the diagnostic properties of the system,

and it performs diagnostics by observing the system on-line. In [9] the authors extend the approach of [40] to timed DES, and [55], [56] derive an analogous approach to [40] and [9], respectively, using a state-based approach. In [36] the authors extend the approach of [40] to account for inter-system communication, intermittent faults, and temporal constraints. In [32] the authors suggest an integrated approach for control and diagnosis of DES. When the system functionality degrades, the most-likely failures are isolated using causal networks, and then the control mechanism will generate the least-cost actions that attempt to recover from the failure and maintain the control objectives.

Each of the abovementioned approaches possesses certain advantages and disadvantages and is best applicable under certain circumstances and/or for certain systems. As stated previously, the approach for diagnosis that needs to be used is mainly dictated by the type of failures one is interested in diagnosing along with the type of available sensors.

Almost all of the failure diagnosis approaches in the literature have been developed for systems where the information used for fault diagnosis is centralized. Many systems such as the majority of technological complex systems (computer and communication networks, manufacturing systems, process control and power systems, etc.) are informationally decentralized. In decentralized information systems there are several work stations (decision makers, controllers, diagnosers) each having access to its own local information. The stations may communicate and exchange limited information among each other. Since this information is exchanged in real-time and over channels of limited capacity, there are propagation delays, along with faults and transmission errors. Thus, the information available to each station is incomplete, delayed, and possibly erroneous. Most approaches to failure diagnosis suggested in the literature do not apply directly to informationally decentralized systems, hence the need to develop diagnostic methodologies for these systems. This fact is also recognized in [3], [12], [17], [21], [28], [30], [35], and [43].

In [3] the authors present a modular technique, amenable to parallel implementation, for the diagnosis of large-scale, distributed, asynchronous event-driven systems which they refer to as active systems. An active system can be modeled as a network of communicating automata. The main goal of the diagnostic technique is the reconstruction of the behavior of the active system starting from a set of observable events. First, the technique generates many representations of parts of the active system based on available observation. Next, these representations are merged to generate a unified representation by using the history of observable events. Finally, the diagnostic information is generated on the basis of fault events possibly incorporated within the reconstructed behavior.

In [12], the authors implement a centralized real time diagnosis algorithm, TEAM-RT, in a distributed fashion. The system they monitor is decomposed into sub-systems. Each

sub-system is monitored by a local unit running TEAM-RT. Each local unit is cognizant of its immediate neighbors, i.e., units monitoring sub-systems that affect or may be affected by the monitored sub-system. By allowing exchange of information among neighboring local units, the authors propose an algorithm that achieves the global diagnosis, that is, the diagnosis of the system as if one unit, running TEAM-RT, is collectively monitoring all the sub-systems.

In [21], the authors present a distributed fault monitoring method for manufacturing systems called time templates monitoring. A time template, by definition, consists of a trigger event and a set of consequences. There is a set of distributed interconnected processors available for monitoring. The set of templates modeling the system is distributed among the processors. The processors use the sets of timing and sequencing relationships available through their assigned templates to establish when events are expected to occur, and to determine if an event has occurred in the appropriate context of some previous event.

In [28], the authors develop a highly modular fault diagnosis methodology using digraph models of process behavior. The method is developed from graph theory and uses off-line analysis of digraph structure to reduce the on-line computation wherever possible. The on-line work is designed for a distributed implementation in which each sensor and controller currently in an abnormal state examines the states for a small number of adjacent measurements to suggest possible faults, including both measured and unmeasured variables.

In [30][2], the authors are interested in diagnosing a telecommunication network composed of many sub-systems. They build local diagnosers for these sub-systems. Each local diagnoser generates local diagnosis. All available local diagnoses are combined, using the history of observable events and the models of the sub-systems, to generate a global diagnosis. The combination is done while minimizing the computation of the overall diagnosis.

In [43], the authors discuss diagnosis problems in distributed systems composed of several spatially separated sites. At every site, there exists a diagnoser that partially observes the system and is in charge of diagnosing faults associated with the site. Diagnosers are allowed to exchange information. The authors characterize the class of distributed systems where there exists no inter-diagnoser messaging scheme that can replicate the information available to a centralized diagnoser.

[2]It is worth mentioning that the work in [30] has been motivated by the work on failure diagnosis with decentralized information that appeared in [14] and constitutes in fact a major part of this thesis.

1.3 Contribution of the Thesis

The main contributions of the thesis are:

1. The development of a new methodology for failure diagnosis in systems with decentralized information.

2. The solution of a generic optimization problem in sensor selection. Such an optimization problem arises in areas such as failure diagnosis, hypothesis testing, etc.

Below we describe in more detail our contribution in each of the aforementioned areas.

1.3.1 Failure Diagnosis of Coordinated Decentralized Discrete Event Systems

We adopt the DES approach of [40] to failure diagnosis and extend it to account for decentralized information. We restrict attention to a coordinated decentralized architecture with two local sites communicating with a coordinator. This architecture is depicted in Figure 1.1. The top block represents the system model. Each site is composed of two

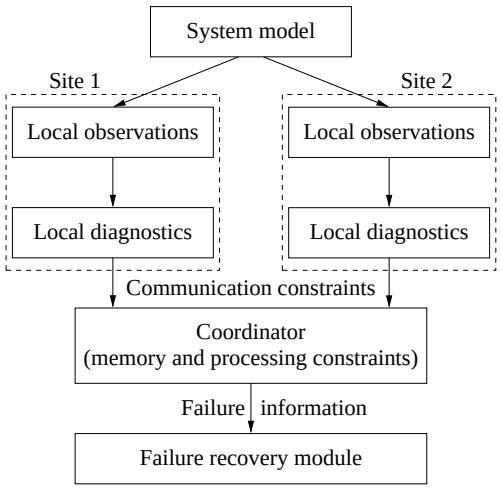

Figure 1.1: Coordinated decentralized architecture

modules: an observation module and a diagnostic module. Site i, $i \in \{1, 2\}$, locally observes the system based on its available sensing capabilities, that is, each site is assigned a set of observable events. The union of these sets is the set of all observable events. Site i locally processes its own observation and generates its diagnostic information. Both sites communicate some form of their diagnostic information to the coordinator. The type of information communicated is determined by the communication rules used by the sites. The task of the coordinator is to process, according to a prescribed decision rule, the

messages received from both sites to infer occurrences of failures. If a failure is detected by the coordinator, it is broadcast to the failure recovery module. We investigate the diagnosability properties of the above architecture under a set of assumptions, the most important of which are:

- Messages communicated between the local sites and the coordinator are received in the order they are locally sent from a site to the coordinator.
- The coordinator does not have a copy of the system model and it has limited memory and limited processing capabilities.

We specify three protocols that realize the proposed architecture; each protocol is defined by the diagnostic information generated at the local sites, the communication rules used by the local sites, and the coordinator's decision rule. We analyze the diagnostic properties of each protocol. We state and prove conditions for a language to be diagnosable under each protocol. These conditions are verifiable off-line. The on-line diagnostic process is carried out using the diagnosers introduced in Sampath et al. or a slight variation of these diagnosers. The key features of the proposed protocols are: (i) they achieve, each under a set of assumptions, the same diagnostic performance as the centralized diagnoser; and (ii) they highlight the "performance vs. complexity" tradeoff that arises in coordinated decentralized architectures.

Some of the advantages of our approach are:

- The approach not only applies to DES but also to systems that could be modeled at a discrete event level for the purpose of failure diagnosis.
- The notion of diagnosability is formally defined and consequently it is possible to analyze diagnostic properties of a given system.
- The on-line diagnosis procedure is achieved using diagnosers which are implemented at local sites.
- The approach is intended for multiple faults, an issue that is not well addressed in many of the other diagnosis frameworks 9centralized and decentralized).

1.3.2 An Optimization Problem in Sensor Selection

Chapter 9 of the thesis addresses a generic optimization problem in sensor selection. We assume that a dynamic system possesses a certain property, call it *Property D*, when a set Γ of sensors is used. There is a cost c_A associated with each set A of sensors that is a subset

of Γ. Given any set of sensors that is a subset of Γ, it is possible to determine, via a test, whether the resulting system-sensor combination possesses Property D. Each test required to check whether or not Property D holds incurs a fixed cost. For each set of sensors A that is a subset of Γ there is an *a priori* probability p_A that the test will be positive, i.e., the system-sensor combination possesses Property D. The objective is to determine a test strategy, i.e., a sequence of tests, to minimize the expected cost, associated with the tests, that is incurred until a least expensive combination of sensors that results in a system-sensor combination possessing Property D is identified. We determine conditions on the sensor costs c_A and the *a priori* probabilities p_A under which the strategy that tests combinations of sensors in increasing order of cost is optimal with respect to the aforementioned objective.

1.4 Organization of the Thesis

This thesis is organized as follows. In Chapter 2, we review the DES approach for failure diagnosis of [40, 41]. We discuss the coordinated decentralized architecture we suggest to account for failure diagnosis of decentralized systems in Chapter 3. In Chapters 4, 5, and 6 we present three protocols that realize the architecture along with an analysis of their diagnostic properties. We discuss some of the issues related to the architecture and it realizations in Chapter 7. In Chapter 8 we discuss the application of our methodology to diagnose a wireless local area network. We formulate and solve a sensor selection problem in Chapter 9. Finally, in Chapter 10 we draw some conclusions, and present some directions for future research.

We note here that the main results presented in this thesis were first presented in [14], [13], and [15]. We also note that most of the computations for the example presented in Chapter 8 were performed using the routines in the UMDES-LIB software library that has been developed by the discrete event systems group at the University of Michigan (cf. `www.eecs.umich.edu/umdes`).

CHAPTER 2

A DISCRETE EVENT SYSTEMS APPROACH TO FAILURE DIAGNOSIS

In this chapter we briefly review the DES approach to failure diagnosis of [40, 41]. The methodology that we suggest to account for failure diagnosis with decentralized information builds upon this approach. Also, we discuss the notion of extended diagnoser that was first introduced in [37], and analyze the relationship between extended diagnosers and diagnosers as introduced in [40, 41]. Extended diagnosers will be used in the design of one of the decentralized diagnostic protocols.

2.1 The System Model

The system to be diagnosed is modeled as an FSM

$$G = (X, \Sigma, \delta, x_0) \tag{2.1}$$

where X is the state space, Σ is the set of events, δ is the partial transition function, and x_0 is the initial state of the system. The model G accounts for the normal and failed behavior of the system. The behavior of the system is described by the prefix-closed language [7] $L(G)$ generated by G. $L(G)$ is a subset of Σ^*, where Σ^* denotes the Kleene closure of the set Σ [22]. In this thesis we will use the language $L(G)$, or simply L, and the system interchangeably.

Some of the events in Σ are observable, i.e., their occurrence can be observed, while the rest are unobservable. Thus, the event set Σ is partitioned as $\Sigma = \Sigma_o \cup \Sigma_{uo}$ where Σ_o represents the set of observable events and Σ_{uo} the set of unobservable events. The observable events in the system may be one of the following: commands issued by the controller, sensor readings occurring after the execution of those above commands, and changes in sensor readings. The unobservable events may be failure events or other events that cause changes in the system state not recorded by sensors (see [38, 41]).

Let $\Sigma_f \subseteq \Sigma$ denote the set of failure events which are to be diagnosed. We assume, without loss of generality, that $\Sigma_f \subseteq \Sigma_{uo}$, since an observable failure event can be trivially diagnosed. Our objective is to identify the occurrence, if any, of the failure events, given that in the traces generated by the system, only the events in Σ_o are observed. In this regard, we partition the set of failure events into disjoint nonempty sets corresponding to different failure types

$$\Sigma_f = \Sigma_{f1} \cup \Sigma_{f2} \cup \ldots \cup \Sigma_{fm}. \tag{2.2}$$

Let Π_f denote this partition. For the motivation of such a partition, the reader is referred to [40, 41]. Hereafter, when we write a failure of type F_i has occurred, we will mean that some event of the set Σ_{fi} has occurred.

2.2 Notation

The empty trace is denoted by ϵ. Let $\overline{s}$ denote the prefix-closure of any trace $s \in \Sigma^*$. We define $||s||$ to be the length of trace s. Whenever we say that there exists a trace s of *arbitrarily long length* having a given property, we mean the following: for all integers n, there exists s, such that $||s|| > n$ and s possesses the given property. We denote by L/s the post-language of L after s, i.e.,

$$L/s = \{t \in \Sigma^* | st \in L\}. \tag{2.3}$$

We define the projection $P : \Sigma^* \rightarrow \Sigma_o^*$ in the usual manner [7, 34]

$$\begin{aligned} P(\epsilon) &= \epsilon \\ P(\sigma) &= \sigma \quad \text{if } \sigma \in \Sigma_o \\ P(\sigma) &= \epsilon \quad \text{if } \sigma \in \Sigma_{uo} \\ P(s\sigma) &= P(s)P(\sigma), \quad s \in \Sigma^*,\ \sigma \in \Sigma. \end{aligned} \tag{2.4}$$

The inverse projection operator P_L^{-1} is defined as

$$P_L^{-1}(y) = \{s \in L : P(s) = y\}. \tag{2.5}$$

Let s_f denote the final event of trace s. We define

$$\Psi(\Sigma_{fi}) = \{s\sigma_f \in L : \sigma_f \in \Sigma_{fi}\}, \tag{2.6}$$

i.e., $\Psi(\Sigma_{fi})$ denotes the set of all traces that end in a failure event belonging to the class Σ_{fi}. Consider $\sigma \in \Sigma$ and $s \in \Sigma^*$. We use the notation $\sigma \in s$ to denote that σ is an event in

the trace s. With slight abuse of notation, we write $\Sigma_{fi} \in s$ to denote the fact that $\sigma_f \in s$ for some $\sigma_f \in \Sigma_{fi}$, or formally, $\overline{s} \cap \Psi(\Sigma_{fi}) \neq \emptyset$. We also define

$$X_o = \{x_0\} \cup \{x \in X : \ x \text{ has an observable event into it}\}. \tag{2.7}$$

Let $L(G, x)$ denote the set of all traces that originate from state x of G. We define

$$L_o(G, x) = \{s \in L(G, x) : s = u\sigma, u \in \Sigma_{uo}^*, \sigma \in \Sigma_o\} \tag{2.8}$$

and

$$L_\sigma(G, x) = \{s \in L_o(G, x) : s_f = \sigma\}. \tag{2.9}$$

$L_o(G, x)$ denotes the set of all traces that originate from state x and end at the first observable event, while $L_\sigma(G, x)$ denotes those traces in $L_o(G, x)$ that end with the particular observable event σ.

The generator G' (see [38, 40]) is the nondeterministic FSM,

$$G' = (X_o, \Sigma_o, \delta_{G'}, x_0), \tag{2.10}$$

where X_o, Σ_o, and x_0 are defined as previously, and the transition relation of G' is given by $\delta_{G'} \subseteq (X_o \times \Sigma \times X_o)$ and is defined as follows:

$$(x, \sigma, x') \in \delta_{G'} \qquad \text{if } \delta(x, s) = x' \text{ for some } s \in L_\sigma(G, x). \tag{2.11}$$

It is easy to verify that $L(G') = P(L)$ where

$$P(L) = \{t : t = P(s) \qquad \text{for some } s \in L\}. \tag{2.12}$$

2.3 Definition of Diagnosability

Loosely speaking, a language is said to be diagnosable with respect to a set of observable events and a failure partition if within a finite delay, the occurrence of any failure can be detected using the history of observable events. More rigorously, diagnosability is defined as follows [38, 40]:

Definition 2.3.1 *A prefix-closed and live language L is said to be diagnosable with respect to the projection P and with respect to the partition Π_f on Σ_f if the following holds*

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)(\| t \| \geq n_i \Rightarrow D)$$

where the diagnosability condition D is

$$(\forall w \in P_L^{-1}(P(st)))\ (\Sigma_{fi} \in w).$$

(A language L is live if for all $s \in L$, there exists $\sigma \in \Sigma$ such that $s\sigma \in L$.) Note here that the above definition is only applicable to centralized systems, since it assumes the availability of all the system information at one (centralized) center or site: there is only one projection P that observes the behavior of the system, in addition to a single inverse projection P_L^{-1}, and both are used to check the diagnosability condition D.

2.4 The Diagnoser

The diagnoser is an FSM built from the system model G. This machine is used to perform diagnosis when it observes on-line the behavior of the system. We first define the set of failure labels $\Delta_f = \{F_1, F_2, \ldots, F_m\}$ where $|\Pi_f| = m$, and the complete set of possible labels

$$\Delta = \{N\} \cup 2^{\Delta_f}. \tag{2.13}$$

Here N is to be interpreted as meaning normal, while F_i, $i \in \{1, 2, \ldots, m\}$ as meaning that a failure of type F_i has occurred. Recall, from Equation 2.7, the definition of X_o and define

$$Q_o = 2^{X_o \times \Delta}. \tag{2.14}$$

The diagnoser for G is the FSM

$$G_d = (Q_d, \Sigma_o, \delta_d, q_0) \tag{2.15}$$

where Q_d, Σ_o, δ_d, and q_0 have the usual interpretation of state space, event set, transition function, and initial state. The initial state of the diagnoser is defined to be $\{(x_0, \{N\})\}$. The transition function δ_d of the diagnoser is constructed in a similar manner to the transition function of an observer of G [7, 22], with an additional aspect that includes attaching failure labels to the states and propagating these labels from state to state (cf. [40]). For more information about the construction of the diagnoser, the reader is referred to [38, 40]. The state space Q_d is the resulting subset of Q_o composed of the states of the diagnoser that are reachable from q_0 under δ_d. Since the state space Q_d of the diagnoser is a subset of Q_o, a state q_d of G_d is of the form $q_d = \{(x_1, l_1), \ldots, (x_n, l_n)\}$, where $x_i \in X_o$ and $l_i \in \Delta$. The following example illustrates the construction of a diagnoser.

Example 2.4.1 *Consider the system shown in Figure 2.1. The set of events is* $\Sigma = \{a, b, c, d, e, \sigma\}$, *and* σ *is the only unobservable event and failure event. The initial state of the system*

is state 1. *We denote by* F_1 *the failure label of the event* σ. *Define* $\Sigma_{o1} = \{a, c, d, e\}$ *to be the set of observable events. Denote by* G_{d1}, *also shown in Figure 2.1, the diagnoser. We illustrate how a state of the diagnoser is generated. Assume that the diagnoser is in state* $(6N,7N)$: *this means that following the observation of events a then c the system is either in state* 6 *following normal behavior (hence the label* N*) or in state* 7 *also following normal behavior. Once the diagnoser observes the event d, the system may have executed event d from state* 6 *and reached state* 8 *following a normal behavior, that is the state reached is* $8N$, *or it may have executed the sequence of events* σd *(note* σ *is unobservable) from state* 7 *and reached state* 10. *In the latter case state* 10 *is associated with label* F_1, *the label of the failure event* σ. *Hence, after observing event d, the diagnoser updates its state to* $(8N, 10F1)$. *The remaining states are constructed following a similar procedure. The diagnoser's initial state is* $1N$ *since, by assumption, the system begins its operation in the normal (non-faulty) state/condition.*

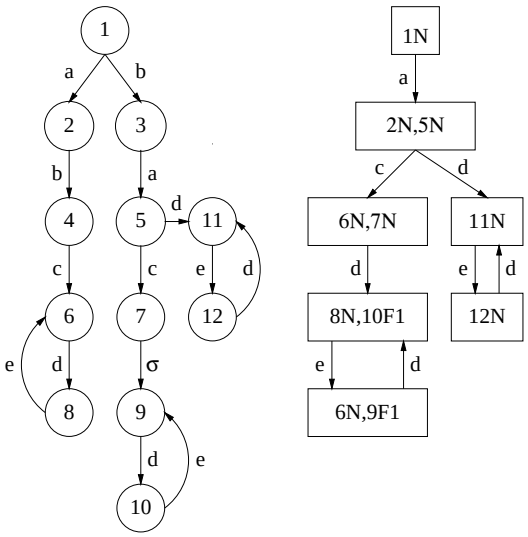

Figure 2.1: The system G, left, and the diagnoser G_{d1} for Example 2.4.1

Next, we provide some definitions that are necessary in order to state the main diagnosability result for centralized systems in Section 2.5. For a detailed discussion and interpretation of this material the reader is referred to [38, 40].

Definition 2.4.1 *(Definition 6-1 in [40]). A state* $q \in Q_d$ *is said to be* F_i*-certain if* $\forall (x, l) \in q$, $F_i \in l$.

Definition 2.4.2 *(Definition 6-3 in [40]). A state* $q \in Q_d$ *is said to be* F_i*-uncertain if* $\exists (x, l), (y, l') \in q$, x *not necessarily distinct from* y, *such that* $F_i \in l$ *and* $F_i \notin l'$.

Definition 2.4.3 *(Definition 7 in [40]). A set of states $x_1, x_2, \ldots, x_n \in X$ is said to form a cycle in G if $\exists s \in L(G, x_1)$ such that $s = \sigma_1\sigma_2\ldots\sigma_n$, and $\delta(x_l, \sigma_l) = x_{(l+1) \bmod n}$, $l = 1, 2, \ldots, n$.*

Definition 2.4.4 *(Definition 8 in [40]). A set of F_i-uncertain states $q_1, q_2, \ldots, q_n \in Q_d$ is said to form an F_i-indeterminate cycle if*

1) $q_1, q_2, \ldots, q_n$ form a cycle in G_d with $\delta_d(q_l, \sigma_l) = q_{l+1}$, $l = 1, 2, \ldots, n-1$, $\delta_d(q_n, \sigma_n) = q_1$, $\sigma_l \in \Sigma_o$, $l = 1, 2, \ldots, n$.

2) $\exists (x_l^k, l_l^k), (y_l^r, \tilde{l}_l^r) \in q_l$, x_l^k not necessarily distinct from y_l^r, $l = 1, 2, \ldots, n$, $k = 1, 2, \ldots, m$, and $r = 1, 2, \ldots, m'$ such that

a) $F_i \in l_l^k$, $F_i \notin \tilde{l}_l^r$ for all l, k, and r.

b) The sequence of states $\{x_l^k\}$, $l = 1, 2, \ldots, n$, $k = 1, 2, \ldots, m$, and $\{y_l^r\}$, $l = 1, 2, \ldots, n$, $r = 1, 2, \ldots, m'$ form cycles in G' with

$$(x_l^k, \sigma_l, x_{l+1}^k) \in \delta_{G'},\ l = 1, 2, \ldots, n-1,\ k = 1, 2, \ldots, m,$$

$$(x_n^k, \sigma_n, x_1^{k+1}) \in \delta_{G'},\ k = 1, 2, \ldots, m-1,\ \text{and}\ (x_n^m, \sigma_n, x_1^1) \in \delta_{G'},$$

and

$$(y_l^r, \sigma_l, y_{l+1}^r) \in \delta_{G'},\ l = 1, 2, \ldots, n-1,\ r = 1, 2, \ldots, m',$$

$$(y_n^r, \sigma_n, y_1^{r+1}) \in \delta_{G'},\ r = 1, 2, \ldots, m'-1,\ \text{and}\ (y_n^{m'}, \sigma_n, y_1^1) \in \delta_{G'}.$$

An F_i-indeterminate cycle in G_d indicates the presence in L of two traces s_1 and s_2 of arbitrarily long length, such that they both have the same observable projection and s_1 contains a failure event from the set Σ_{fi} while s_2 does not.

Finally, the following lemma relates the properties of a diagnoser state to the properties of the traces in the language.

Lemma 2.4.1 *(Lemma 2 in [40]).*

i) Let $\delta_d(q_0, u) = q$. If q is F_i-certain, then $\forall w \in P_L^{-1}(u)$, $\Sigma_{fi} \in w$.

ii) If a state $q \in Q_d$ is F_i-uncertain, then this implies that $\exists s_1, s_2 \in L$ such that $\Sigma_{fi} \in s_1$, $\Sigma_{fi} \notin s_2$, $P(s_1) = P(s_2)$, and $\delta_d[q_0, P(s_1)] = q$.

2.5 Necessary and Sufficient Conditions for Diagnosability

It is intuitive, based on the definition of diagnosability, the properties of the diagnoser, and Definition 2.4.4, that in order for a language to be diagnosable, the diagnoser should not have any F_i-indeterminate cycles for all failure types F_i. This condition is stated formally as follows:

Theorem 2.5.1 *(Theorem 2 in [40]). A language L is diagnosable if and only if its diagnoser G_d satisfies the following condition: there are no F_i-indeterminate cycles in G_d for all failure types F_i.*

2.6 The Extended Diagnoser

The *extended diagnoser* for G was first introduced in [37], and it is the FSM

$$G_d^e = (Q_d^e, \Sigma_o, \delta_d^e, q_0^e) \tag{2.16}$$

where $Q_d^e, \Sigma_o, \delta_d^e$, and q_0^e have the usual interpretation of state space, event set, transition function, and initial state. The initial state of the extended diagnoser is defined to be $\{(x_0, \{N\}), (x_0, \{N\})\}$. A state $q \in Q_d^e$ is of the form

$$q = \{((x_1, l_1), (x_1', l_1')), ((x_2, l_2), (x_2', l_2')), ((x_2, l_2), (x_2'', l_2'')), \ldots, ((x_n, l_n), (x_n', l_n'))\},$$

where each (x, l) pair is in Q_o, i.e., $x \in X_o$ and $l \in \Delta$. A tuple of (x, l) pairs, say $((x_1, l_1), (x_1', l_1'))$, has the following meaning: x_1' is a component of a system state estimate after the occurrence of an observable event and l_1' is its failure label, while x_1 is the immediate predecessor state of x_1' in G' and l_1 is its corresponding failure label. The transition function δ_d^e of the extended diagnoser is constructed in a manner similar to the transition function of the diagnoser G_d, with the additional aspect that every state of G that appears in a state component of G_d is associated with its immediate predecessor state in G' (along the sub-trace of events under consideration) and both states carry their labels; these labels are attained following the same label propagation rules as in [40]. The state space Q_d^e is the resulting subset of $Q_o \times Q_o$ composed of the states of the extended diagnoser that are reachable from q_0^e under δ_d^e. By construction, $L(G_d^e) = L(G_d) = P(L)$. We illustrate the construction of extended diagnosers in the following example.

Example 2.6.1 *Consider the system shown in Figure 2.1 with $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$, $\Sigma_{o1} = \{a, c, d, e\}$. The extended diagnoser G_{d1}^e for this system is shown in Figure 2.2. Consider the state $q = \{(2N, 6N), (5N, 7N)\}$ in G_{d1}^e; q is read as follows: the system is either in state 6 with a normal label, or it is in state 7, also with a normal label; state 6 has been reached (by an observable event, possibly preceded by unobservable events) from state 2, while state 7 has been reached (by an observable event, possibly preceded by unobservable events) from state 5. Now the next observable event is d: if the system is at state 6, then it transitions into state 8, and since there are no failure events along the path from state 6 to state 8 the resulting component of the new state estimate is $(6N, 8N)$; if the system is at state 7, it transitions into state 10 following the occurrence of the sequence σd,*

i.e., a failure of type F_1 has occurred along the path, and the resulting other component of the new state estimate is $(7N, 10F1)$. Therefore the state of G^e_{d1} is $\{(6N, 8N), (7N, 10F1)\}$ after the occurrence of the observable event d. All other extended diagnoser states are constructed by following a similar procedure.

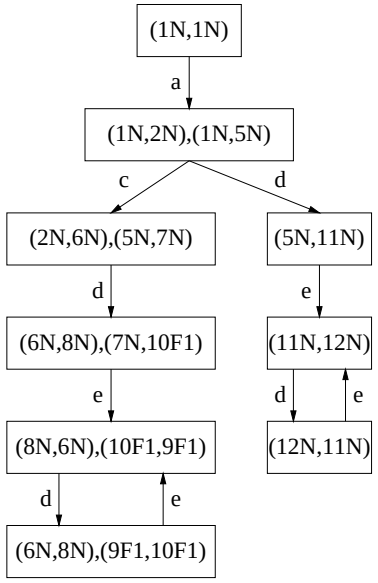

Figure 2.2: The extended diagnoser G^e_{d1} for Example 2.6.1

We define the state projection

$$\begin{aligned} SP : Q_o \times Q_o &\rightarrow Q_o \\ q = \{((x_1, l_1), (x'_1, l'_1)), \ldots, ((x_n, l_n), (x'_n, l'_n))\} &\mapsto SP(q) = \{(x'_1, l'_1), \ldots, (x'_n, l'_n)\}. \end{aligned} \tag{2.17}$$

Then, with a slight abuse of notation, we have that $SP(G^e_d) = G_d$; hence, one diagnoser state may be associated with more than one extended diagnoser states (see Example 2.6.2 below). Therefore, an extended diagnoser state potentially carries more information than a diagnoser state, and this fact will be exploited further in this thesis. In the case of centralized systems, G_d and G^e_d are equivalent from the point of view of diagnosability as defined in Definition 2.3.1; it is for that reason that prior work [38, 39, 40, 41] only considered the simpler G_d.

Example 2.6.2 *Again consider the system shown in Figure 2.1 with $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$. The diagnoser G_{d1} and extended diagnoser G^e_{d1} for this system are shown in Figures 2.1 and 2.2, respectively. We can see that the transition structure of G^e_{d1}*

refines that of G_{d1}. *In particular, state* $q = (8N, 10F1)$ *of* G_{d1} *is associated with states* $q_1 = \{(6N, 8N), (7N, 10F1)\}$ *and* $q_2 = \{(6N, 8N), (9F1, 10F1)\}$ *in* G^e_{d1} *since SP(*q_1*)=SP(*q_2*)=*q.

To provide the necessary and sufficient conditions of diagnosability in terms of G^e_d, we need the following definitions.

Definition 2.6.1 *A state* $q \in Q^e_d$ *is said to be* F_i*-certain if* $\forall (x, l) \in SP(q)$, $F_i \in l$.

Definition 2.6.2 *A state* $q \in Q^e_d$ *is said to be* F_i*-uncertain if* $\exists (x, l)$, $(y, l') \in SP(q)$, x *not necessarily distinct from* y, *such that* $F_i \in l$ *and* $F_i \notin l'$.

Definition 2.6.3 *(Definition 1 in [37]). A set of states* $q_1, q_2, \ldots, q_n \in Q^e_d$ *is said to form a cycle in* G^e_d *if the following is true:*

$$\delta^e_d(q_l, \sigma_l) = q_{(l+1)},\ l = 1, 2, \ldots, n-1,\ \text{and}\ \delta^e_d(q_n, \sigma_n) = q_1,$$

for some observable events σ_i, $i = 1, \ldots, n$.

Definition 2.6.4 *(Definition 2 in [37]). A set of* (x_i, l_i) *pairs, where* $(x_i, l_i) \in Q_o$, $i = 1, 2, \ldots, n$, *is said to form a matched cycle in* G^e_d *if* $\exists\ q_i \in G^e_d$, $i = 1, 2, \ldots, n$, *such that:*

$$((x_i, l_i), (x_{i+1}, l_{i+1})) \in q_{i+1},\ i = 1, 2, \ldots, n-1,\ \text{and}\ ((x_n, l_n), (x_1, l_1)) \in q_1.$$

Note that the existence of such a set of (x_i, l_i) pairs has the following implications (from the construction procedure of G^e_d):

1. q_i, $i = 1, 2, \ldots, n$, form a cycle in G^e_d.
2. x_i, $i = 1, 2, \ldots, n$, form a cycle in G'.

Definition 2.6.5 *(Definition 3 in [37]). A set of states* $q_1, q_2, \ldots, q_n \in Q^e_d$ *forming a cycle of* F_i*-uncertain states in* G^e_d *is said to form an* F_i*-indeterminate cycle in* G^e_d *if the following hold:*

1. $\exists$ *a set of* $(x_j, l_j) \in SP(q_j)$, $j = 1, 2, \ldots, n$, *forming a matched cycle in* G^e_d, *with* $F_i \in l_j$, $j = 1, 2, \ldots, n$,

   and

2. $\exists$ *a set of* $(y_j, l'_j) \in SP(q_j)$, $j = 1, 2, \ldots, n$, *forming a matched cycle in* G^e_d, *with* $F_i \notin l'_j$, $j = 1, 2, \ldots, n$.

Next we state a result that relates the existence of F_i-indeterminate cycles in G_d^e to the existence of F_i-indeterminate cycles in G_d.

Proposition 2.6.1 *Consider a system G, its diagnoser G_d, and its extended diagnoser G_d^e. Then there are F_i-indeterminate cycles in G_d if and only if there are F_i-indeterminate cycles in G_d^e.*

Proof of Proposition 2.6.1 Sufficiency($\Leftarrow$). G_d^e has F_i-indeterminate cycles. Consider a set of states q_k, $k = 1, \ldots, n$, that form an F_i-indeterminate cycle in G_d^e. We claim that the set of states $\{p_1, \ldots, p_m\} = SP(\{q_1, \ldots, q_n\})$, $m \leq n$, forms an F_i-indeterminate cycle in G_d. (Note here that $m \leq n$ since, as discussed earlier, one diagnoser state may be associated with more than one extended diagnoser states.) The claim can be established as follows: by assumption, there exist two sets of states of the form (x_j, l_j), $(y_j, l'_j) \in SP(q_j)$, $j = 1, \ldots, m$, such that $F_i \in l_j$, but $F_i \notin l'_j$ (cf. Definition 2.6.5). Hence the cycle of states $\{p_1, \ldots, p_m\}$ in G_d is an F_i-uncertain cycle. Moreover, by the implications of Definition 2.6.4, the sets $\{x_j\}$ and $\{y_j\}$ form cycles in G'. Therefore the resulting cycle in G_d is F_i-indeterminate by Definition 2.4.4.

Necessity($\Rightarrow$) G_d has F_i-indeterminate cycles. From Definition 2.4.4, there exist two traces s and s' in $L(G)$, such that $P(s) = P(s')$, $F_i \notin s$, $F_i \in s'$ and s, s' are arbitrarily long. Since L(G) is a regular language, then the fact that s, s' are arbitrarily long implies that the system will loop in a cycle, say A (respectively B) if s (respectively s') is executed. Corresponding to A (respectively B) there exists a cycle of pairs (x_j, l_j) (respectively (y_j, l'_j)) in Q_o, $j = 1, \ldots, n$. Moreover, since $P(s) = P(s')$, (x_j, l_j) and (y_j, l'_j), $j = 1, \ldots, n$, belong to the same set of states $\{q_1, \ldots, q_n\}$ in G_d^e; hence they form matched cycles in G_d^e. By the implications of Definition 2.6.4 the states $\{q_1, \ldots, q_n\}$ in G_d^e form a cycle, and the fact that $F_i \notin l_j$ but $F_i \in l'_j$ implies that the cycle is F_i-indeterminate. **Q.E.D.**

Based on Proposition 2.6.1 and Definition 2.3.1 we provide a test to check the diagnosability of a language in terms of the extended diagnoser G_d^e:

Theorem 2.6.1 [1] *A prefix-closed and live language L is diagnosable with respect to the projection P and the failure partition Π_f on Σ_f if and only if its extended diagnoser G_d^e satisfies the following condition: there are no F_i-indeterminate cycles in G_d^e for all failure types F_i.*

Proof of Theorem 2.6.1 The proof is a direct consequence of Definition 2.3.1 and Proposition 2.6.1. **Q.E.D.**

[1] This theorem is presented as an unproved claim in [37].

Remark on terminology We note here that almost all of the notation introduced in this chapter assumes that the set of observable events is Σ_o. In later chapters, we will be using subsets of Σ_o, namely Σ_{o1} and Σ_{o2}; the above notation will still be applicable to the subsets of Σ_o, with the minor change, when necessary, of adding subscripts: a "1" subscript will be used in notation related to Σ_{o1}, while a "2" subscript will be used in notation related to Σ_{o2}. In this case, we define Σ_o to be $\Sigma_{o1} \cup \Sigma_{o2}$.

CHAPTER 3

A COORDINATED DECENTRALIZED ARCHITECTURE

In decentralized systems, the global system information is distributed at several sites. The "agents" or work stations at different sites may communicate and exchange information in real time, or just report some processed version of their information to a center that, in general, possesses limited knowledge about the system. For each distinct information flow we obtain a distinct decentralized architecture. In this thesis, we restrict attention to a coordinated decentralized architecture with two local sites communicating with a coordinator. This architecture is depicted in Figure 1.1. In this chapter, we discuss the architecture. We present protocols that realize the architecture in Chapters 4, 5, and 6.

3.1 The Architecture

In Figure 1.1, which is redrawn in Figure 3.1, the top block represents the system model, or G in the notation of Section 2.1. G models the synchronization of the interaction of all the components that constitute the system (see [38, 41]). Each site is composed of two modules: an observation module and a diagnostic module. The site i, $i \in \{1, 2\}$, locally observes the system based on its available sensing capabilities. Therefore, a projection P_i is associated with site i, where P_i is defined on the set of observable events Σ_{oi} (note here that Σ_{o1} and Σ_{o2} need not be disjoint although sites 1 and 2 may be physically apart). The union of Σ_{o1} and Σ_{o2} is the set of all observable events Σ_o. Site i locally processes its own observation and generates its diagnostic information. Both sites communicate some form of their diagnostic information to the coordinator. The type of information communicated is determined by the communication rules used by the sites. The task of the coordinator is to process, according to a prescribed decision rule, the messages received from both sites to infer occurrences of failures. If a failure is detected by the coordinator, it is broadcast to the failure recovery module. We note that the thesis does not address the issue of recovering from faults, nor the effect of the recovery on the operation of the system.

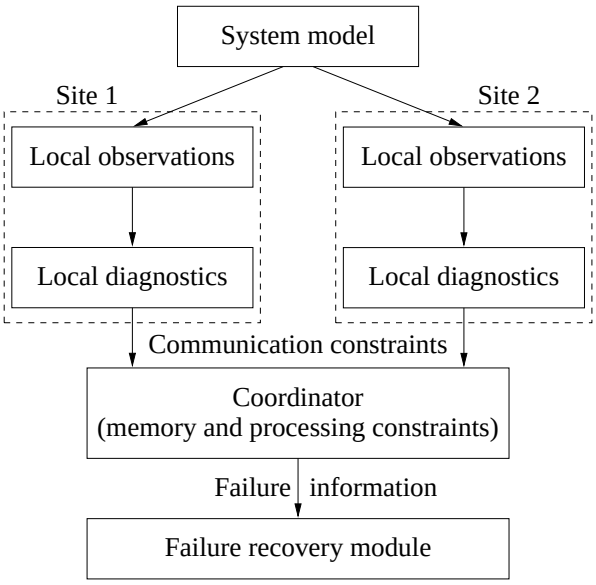

Figure 3.1: Coordinated decentralized architecture

We investigate diagnosability properties of the above architecture under the following assumptions.

A1 $L(G)$ is live.

A2 G has no cycles of unobservable events with respect to either Σ_{o1} or Σ_{o2}.

A3 $L(G)$ is not diagnosable with respect to P_i and Π_f on Σ_f, $i = 1, 2$.

A4 There is reliable communication between the local sites and the coordinator, i.e., all messages sent from a local site are received by the coordinator correctly and in order.

A5 Messages communicated between the local sites and the coordinator are received in the order they are sent (globally).

A6 The sets of observable events at each site are common knowledge [2, 52] to all sites.

A7 The two sites are allowed to report to the coordinator only some processed version of their raw data.

A8 The coordinator does not have a copy of the system model, that is, it does not know the dynamics of the system. It has a simple structure; specifically, it has limited memory and limited processing capabilities.

Assumption **A1** ensures that there are no deadlocks. This assumption can be relaxed easily as discussed in [38, 39]. Assumption **A2** ensures that observations occur with some

regularity with respect to both P_1 and P_2: since detection of failures is based on observable transitions of the system, we require that G does not generate arbitrarily long sequences of unobservable events with respect to either P_1 or P_2. Assumption **A3** eliminates the trivial case where even though the observable events are partitioned, the system is still diagnosable (in a centralized setup) with respect to one of the projections and the failure partition. In such a case the decentralized architecture is necessarily diagnosable: the site that diagnoses the system can assume the responsibilities of the coordinator, and all failure types are consequently diagnosed. Assumption **A5** ensures that the global order of all messages received by the coordinator is preserved. This may be too strong an assumption and approaches to relax it are discussed in Section 3.2. Assumptions **A4**, **A6**, and **A7** are self explanatory. Finally, Assumption **A8** is consistent with features of hierarchical organizations. Assumptions **A1** – **A8** will be used, even if not explicitly stated, in the derivation of all the results in Chapters 4, 5, and 6, unless otherwise specified.

3.2 Addressing Communication Delay Issues in the Coordinated Decentralized Architecture

In coordinated decentralized architectures, like the one in Figure 3.1, the coordinator may receive messages out of order. Ordering of messages from one site to the coordinator may be easily achieved with a transport layer protocol, for instance TCP, or a data link control layer protocol such as Go Back n or Selective Repeat [6]. However, global order, i.e., ordering of messages sent by different local sites to the coordinator, is not necessarily maintained. To achieve global ordering of messages we may use one of the following mechanisms:

(i) We can introduce clocks at the local sites and time-stamp the messages sent by the local sites to the coordinator. In this situation we need to make sure that clocks are synchronized; we can achieve clock synchronization by the use of Global Positioning Systems (GPS) (see [46]). Such a mechanism is not always practical or feasible. For example, synchronizing clocks may be too constraining in low-energy mobile communication networks [51] and telecommunication networks [17, 30]. The amount of information exchanged among the nodes of a communication network to achieve the synchronization of the local clocks is considerable; moreover additional processing and memory storage is required at the local sites.

(ii) We can use untimed discrete event models and design algorithms that order the messages arriving at the coordinator's site. Such an approach is enforced, for instance,

in telecommunication networks where timing information, be it global time or local time, is not available at any node (site) of the telecommunication network [17, 30].

In this thesis we will use the second approach; in fact, we will study the diagnostic properties of two of the protocols presented in this thesis using the approach of ordering messages when Assumption **A5** is violated. Under this approach, to generate a diagnostic decision we store all incoming messages up until the time where all possible orders in which these messages are sent by various local sites can be sorted out. Once these orders are sorted out, the coordinator's decision rule is applied to each one of them. The same procedure is repeated every time incoming messages arrive at the coordinator's site. To highlight our approach to sorting out message orderings at the coordinator's site, we assume that messages sent by local sites are received at the coordinator's site at most "one-step out of order". The same approach can be used when messages are received at most "n-step out of order" where n is finite, $n > 1$; the memory requirements at the coordinator's site increase as n increases. To illustrate the "one-step out of order" assumption, consider the following scenario. Assume the system executes the sequence of events ab. Suppose that event a (resp. b) is observed by local site 1 (resp. local site 2). Denote by x (resp. y) the message generated by the occurrence of event a (resp. b). If the order of reception of messages at the coordinator's site is yx, then x is said to be received "one-step out of order".

3.3 Definition of Diagnosability

As noted in Section 2.3, the definition of diagnosability in [40] (Definition 2.3.1 in this thesis) assumes centralization of the available information; hence it is not directly applicable to coordinated decentralized systems. To define diagnosability for the coordinated decentralized architecture of Figure 3.1 we need some definitions.

Definition 3.3.1 *Within the context of the coordinated decentralized architecture described in Section 3.1 and depicted in Figure 3.1, a protocol is defined by the diagnostic information generated at the local sites, the rules used by the local sites to communicate to the coordinator, and the decision rule used at the coordinator site.*

Let C denote the coordinator's diagnostic information; C is protocol-dependent.

Definition 3.3.2 *The coordinator's diagnostic information C is said to be F_i-certain if based on C, the coordinator is certain that a failure of type F_i has occurred.*

Based on the above definitions we define diagnosability (under a given protocol) as follows:

Definition 3.3.3 *A prefix-closed and live language L is said to be diagnosable under a protocol, a set of projections P_1, P_2 and a failure partition Π_f on Σ_f if the following holds*

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)(\| t \| \geq n_i \Rightarrow C \text{ is } F_i\text{-certain}).$$

Thus diagnosability, as defined above, requires that the detection of any failure should be achieved by the coordinator within a finite delay of the occurrence of that failure.

One could argue that the above definition is too restrictive since it is protocol-dependent, while in fact it should only depend on the architecture itself. In that respect, one could modify the above definition as follows.

Definition 3.3.4 *A prefix-closed and live language L is said to be diagnosable under the coordinated decentralized architecture, a set of projections P_1, P_2 and a failure partition Π_f on Σ_f if there exists a protocol realizing the architecture such that the following holds*

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)(\| t \| \geq n_i \Rightarrow C \text{ is } F_i\text{-certain}).$$

It is obvious that the two definitions are equivalent as far as the property of diagnosability is concerned. The only difference is the fact that Definition 3.3.4 is constructive in the sense that one is supposed to provide a protocol that realizes the architecture *and* verifies the diagnosability property. Having clarified this issue, Definition 3.3.3 will be used throughout this thesis.

3.4 Objective

Any realization of the coordinated decentralized architecture of Section 3.1 cannot outperform the centralized one. Hence, a desirable objective in realizing such an architecture is to aim at diagnosing all failure types that can be diagnosed by the centralized diagnoser. Therefore, the ultimate goal is to determine a protocol that performs as well as the centralized diagnoser would. In case a particular protocol does not achieve this goal, one may want to determine conditions on the system structure under which the protocol performs as well as the centralized diagnostic scheme of [40].

In designing a protocol, the issues of memory storage, communicating messages, and processing power surface. These issues are quite important as far as the implementation of the protocols go in real systems. To address these issues one may need to sacrifice on the diagnostic performance of the protocols and this raises the issue of implementation complexity vs. (diagnostic) performance that exists in coordinated decentralized architectures.

The next three chapters describe three protocols that highlight the tradeoff between diagnostic performance and complexity of implementation as depicted in Figure 3.2. The

points in Figure 3.2 are not quantified, rather they only depict the trend. However the order in which the protocols are pictured is in fact a total order, and this fact will become clear in the next chapters.

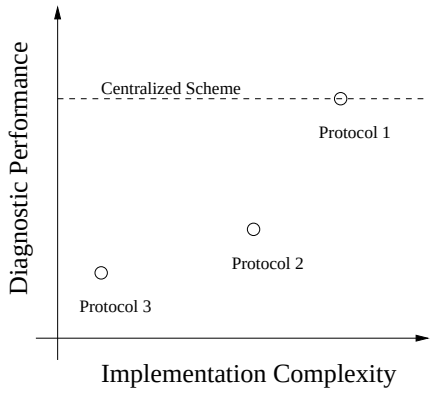

Figure 3.2: The protocols realizing the coordinated decentralized architecture

CHAPTER 4

PROTOCOL 1

We present a protocol for the suggested coordinated decentralized architecture that is capable of diagnosing the same types of failures as the centralized diagnostic scheme of [40]. Initially, the specification of the protocol is done under Assumptions **A1** - **A8** of Section 3.1. Thereafter, we will refer to this protocol as Protocol 1. Afterwards, we present a modification of the protocol that accounts for the relaxation of Assumption **A5** on the preservation of global order.

4.1 Specification of the Protocol

We specify the protocol by defining the diagnostic information generated at the local sites, the communication rules used by the local sites, and the decision rule used at the coordinator site. These are discussed in the following sub-sections.

4.1.1 Diagnostic information at local sites

The diagnostic information at the local site is generated by the extended diagnoser defined in Section 2.6. The state of the extended diagnoser is refined with the *unobservable reach* which is defined as follows.

Definition 4.1.1 *Let* $q = \{((x_1, l_1), (x'_1, l'_1)), \ldots, ((x_n, l_n), (x'_n, l'_n))\}$ *be a state of the extended diagnoser* G^e_{dj}, $j \in \{1, 2, \ldots, m\}$. *Define the set*

$$S_j(q) = \{s \in (\Sigma \backslash \Sigma_{oj})^* : s \in L_\sigma(G, x'_k) \text{ for some } \sigma \in \Sigma_{oi}, i \in \{1, 2, \ldots, m\} \backslash \{j\}, \text{ and some } k \in \{1, \ldots, n\}\}.$$

Then the unobservable reach of q *with respect to* $\Sigma \backslash \Sigma_{oj}$ *is defined as follows:*

$$UR_j(q) = \{q\} \cup \bigcup_{s \in S_j(q)} \{((y_s, l_s), (y'_s, l'_s))\}$$

where (i) y'_s is the successor of some x'_k, $k \in \{1, \ldots, n\}$, after sub-trace $s \in S_j(q)$, (ii) y_s is the immediate predecessor along s of y'_s in G', and (iii) l_s, l'_s are the failure labels corresponding to y_s, y'_s, obtained by propagating the label l'_k of x'_k according to the label propagation function defined in [40].

The unobservable reach appends to the components of each state of the extended diagnoser G^e_{dj} some additional components (along with failure labels and predecessors) that may have been reached following an additional event or a sequence of events that are not observable by the local site j. Note here that in the above definition, y_s may not be equal to x'_k. Also note that while we call $UR_j(q)$ the unobservable reach of q with respect to $\Sigma \backslash \Sigma_{oj}$, its definition stipulates that the sub-traces that are used to generate it end with an observable event (in Σ_o) that is not observed by site j. In the case of two sites (as considered in this thesis), the sub-traces that are used to generate the unobservable reach end with an event in Σ_{oi}, the other set of observable events, as illustrated by the following example.

Example 4.1.1 *Consider the system discussed in Example 2.6.1 and shown in Figure 2.1. The set of events is $\Sigma = \{a, b, c, d, e, \sigma\}$, and σ is the only unobservable and failure event. We denote by F_1 the failure label of the event σ. The extended diagnosers G^e_{d1} and G^e_{d2} associated with the set of observable events $\Sigma_{o1} = \{a, c, d, e\}$ and $\Sigma_{o2} = \{b, d, e\}$ are shown in Figures 2.2 and 4.1, respectively. Consider the state q=$\{(1N, 3N), (1N, 4N)\}$ in G^e_{d2}. To compute the unobservable reach of q with respect to $\Sigma \backslash \Sigma_{o2}$, we first find the set $S_2(q) = \{a, c, ac\}$. The successors of state 3 after traces a and ac are 5 and 7, respectively, while the successor of state 4 after trace c is 6. Therefore, $UR_2(q) = \{(1N, 3N), (1N, 4N), (3N, 5N), (5N, 7N), (4N, 6N)\}$. All state labels are N since there were no failure events along any trace. Note here that although state 7 is a successor of state 3 along the trace ac, the immediate predecessor of 7 in G' (not pictured) is state 5, so the corresponding tuple (after adding the failure labels) is $(5N, 7N)$.*

4.1.2 Communication rules

To define the communication rules, we first note that right after the occurrence of an event that is observable only by one site, say i, the state of the extended diagnoser at site $j \neq i$ does not contain the true system state. Therefore, for the purpose of communicating information from a local site to the coordinator, we need to augment the state of the extended diagnoser with some additional information, the unobservable reach. We define the communication rules **CR** := (**CR1**,**CR2**) as follows:

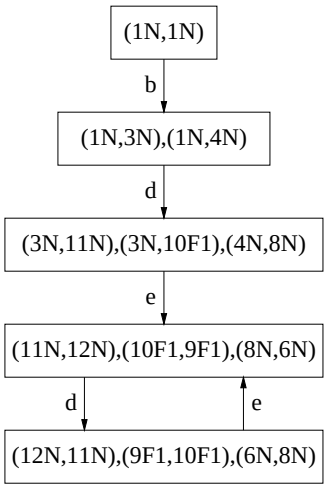

Figure 4.1: The extended diagnoser G^e_{d2} for Example 4.1.1

- **[CRi]**, $i = 1, 2$: After the agent at site i observes an event $\sigma \in \Sigma_{oi}$, it communicates to the coordinator the corresponding state q_i of its extended diagnoser G^e_{di}, its unobservable reach $UR_i(q_i)$ with respect to $\Sigma \backslash \Sigma_{oi}$, and a status bit, $\mathbf{SB_i}$, that takes the values $\mathbf{SB_i} = 1$ when $\sigma \in \Sigma_{oj}$, $j \in \{1, 2\}$, $j \neq i$, or $\mathbf{SB_i} = 0$ when $\sigma \notin \Sigma_{oj}$.

4.1.3 Decision rule

The decision rule of the coordinator consists of two components: (1) a rule according to which its information is updated; and (2) a rule according to which failure occurrences are declared and broadcast to the failure recovery module.

As stated earlier, the coordinator declares that a failure of type F_i has occurred when its diagnostic information C is F_i-certain (cf. Definition 3.3.2). To specify the information update rule we first need to define the following operators (Definitions 4.1.2 and 4.1.3).

Definition 4.1.2 *Let* $q_1 = \{((x_1, l_1), (x'_1, l'_1)), \ldots, ((x_n, l_n), (x'_n, l'_n))\}$ *and* $q_2 = \{((y_1, l_1), (y'_1, l'_1)), \ldots, ((y_m, l_m), (y'_m, l'_m))\}$ *belong to* $Q_o \times Q_o$. *We denote by* $\cap^i_e$, $i \in \{L, R\}$ *the intersection scheme that acts on* q_1 *and* q_2, *and we define it as follows:*

$$q_1 \cap^i_e q_2 \stackrel{\triangle}{=} \{((z, l), (z', l')) \in Q_o \times Q_o\text{: } (z', l') = (x'_k, l'_k) = (y'_j, l'_j) \text{ for some } k, j,\ k \in \{1, 2, \ldots, n\},\ j \in \{1, 2, \ldots, m\}, \text{ and } (z, l) = (x_k, l_k) \text{ if } i{=}L, \text{ otherwise } (z, l) = (y_j, l_j)\}.$$

This intersection scheme is a regular intersection of the components of the two system state estimates along with their failure labels. However, the intersection applies to the components corresponding to the current system state estimates and not to their immediate predecessors. The components of $q_1 \cap^i_e q_2$ corresponding to the immediate predecessors are

determined by operator i. The intersection scheme $\cap_e^i$ introduced by Definition 4.1.2 is illustrated by the following example.

Example 4.1.2 *Consider* $q_1, q_2 \in Q_o \times Q_o$ *such that* $q_1 = \{(6N, 8N), (7N, 10F1)\}$ *and* $q_2 = \{(3N,11N), (3N,10F1), (4N,8N)\}$. *To compute* $q_1 \cap_e^L q_2$ *we find the common components in the two current system state estimates, namely* $8N$ *and* $10F1$, *and we append the predecessors of* $8N$ *and* $10F1$ *in* q_1 *to the states to get* $q_1 \cap_e^L q_2 = \{(6N, 8N), (7N, 10F1)\}$. *Similarly,* $q_1 \cap_e^R q_2 = \{(4N, 8N), (3N, 10F1)\}$.

The second operator we introduce is another intersection scheme, and is defined as follows.

Definition 4.1.3 *Let* $q_1 = \{((x_1, l_1), (x'_1, l'_1)), \ldots, ((x_n, l_n), (x'_n, l'_n))\}$ *and* $q_0 = \{((y_1, l_1), (y'_1, l'_1)), \ldots, ((y_m, l_m), (y'_m, l'_m))\}$ *belong to* $Q_o \times Q_o$. *We denote by* $\cap_c$ *the intersection scheme that acts on* q_1 *and* q_0, *and we define it as follows.*

$$q_1 \cap_c q_0 \triangleq \{((z, l), (z', l')) \in Q_o \times Q_o : (z, l) = (x_i, l_i) = (y'_j, l'_j), \text{ for some } i, j,\ i \in \{1, 2, \ldots, n\},\ j \in \{1, 2, \ldots, m\},\ \text{and } (z', l') = (x'_i, l'_i)\}.$$

The intersection scheme $\cap_c$ is illustrated by the following example.

Example 4.1.3 *Consider* $q_1, q_2 \in Q_o \times Q_o$ *such that* $q_1 = \{(6N, 8N), (7N, 10F1)\}$ *and* $q_0 = \{(4N, 6N)\}$. $q_1 \cap_c q_0 = \{(6N, 8N)\}$ *since the component* $10F1$ *of* q_1 *was reached from the component* $7N$ *which is not present in* q_0.

In addition to the above operators, we need to describe the structure of the coordinator before we precisely specify its information update rule. In addition to the register C where the coordinator stores its current diagnostic information, eight supplementary registers are used for storing messages and previous relevant values necessary for the update of its information. These registers are: *R1*, *R2*, *R3*, *R4*, C_{old}, SB, SB_{1old}, and SB_{2old}. *R1* and *R2* hold the latest states of G_{d1}^e and G_{d2}^e, respectively, *R3* and *R4* hold the latest unobservable reaches of G_{d1}^e and G_{d2}^e, respectively, C_{old} holds the previous coordinator diagnostic information, SB specifies whether the last observed event is observed by both sites (1) or not (0) and SB_{1old}, SB_{2old} provide necessary information to compute the new coordinator diagnostic information.

The information update rule is given in Table 4.1. The rule picks one of the actions **DR1**–**DR6** depending on the available information, i.e., which site observed the last and previous to the last events, and who sent the last message to the coordinator.

Table 4.1: Information update rule at the coordinator site (Protocol 1)

Last report received from G^e_{d1}						
	SB	SB_1	C	New SB	New SB_{1old}	New SB_{2old}
DR1	0	0	$(R_1 \cap^i_e R_4) \cap_c C_{old}$	0	1	0
DR2	0	1	Wait	1	Unmodified	Unmodified
—	1	0	Impossible	—	—	—
DR3	1	1	$(R_1 \cap^i_e R_2) \cap_c C_{old}$	0	1	1
Last report received from G^e_{d2}						
	SB	SB_2	C	New SB	New SB_{1old}	New SB_{2old}
DR4	0	0	$(R_2 \cap^i_e R_3) \cap_c C_{old}$	0	0	1
DR5	0	1	Wait	1	Unmodified	Unmodified
—	1	0	Impossible	—	—	—
DR6	1	1	$(R_1 \cap^i_e R_2) \cap_c C_{old}$	0	1	1
the i superscript in $\cap^i_e$ depends on the current values of the flip-flops SB_{1old} and SB_{2old}, not shown in this table						

The rationale behind the actions **DR1** to **DR6** can be summarized as follows. Once a message from one diagnoser, say G^e_{d1}, reaches the coordinator after the occurrence of an observable event, the state of that diagnoser should contain the true system state. Moreover, if the message says that the event is not observed by the other site (site 2), the current unobservable reach of the diagnoser G^e_{d2} also contains the true system state. Consequently, the logical action is to intersect the state of G^e_{d1} with the unobservable reach of G^e_{d2} using the intersection scheme $\cap^i_e$ (the bits SB_{1old} and SB_{2old} specify the value of i in $\cap^i_e$: if $SB_{1old} = 1$, then $i = L$, that is you append the predecessors from the state of G^e_{d1}; otherwise $i = R$), and then intersect the result with the old coordinator diagnostic information, using the intersection scheme $\cap_c$, to generate the new coordinator diagnostic information. The last intersection is needed to eliminate the possibility of including any illegal behavior in the coordinator diagnostic information.

In case the message received at the coordinator site regards an event that is commonly observed at both sites, the states of diagnosers G^e_{d1} and G^e_{d2} contain the true system state. Therefore, the logical action in this case is to intersect the states of the diagnosers G^e_{d1} and G^e_{d2} by applying the $\cap^i_e$ intersection, and then refine the result by applying the intersection scheme $\cap_c$ as discussed above.

Note that the coordinator is not aware of the rationale described above when it updates its diagnostic information and when it declares that a failure of a certain type has occurred.

The coordinator simply executes the operations $\cap_e^i$, $\cap_c$, updates all of its registers, and declares the occurrence of failures according to the decision rule described above.

We note that before performing any update of the coordinator diagnostic information, the current coordinator diagnostic information is saved into the register C_{old} for later use. Also, the flip-flops are modified once the update of the coordinator diagnostic information is completed. Initially, *R1* and *R2* are initialized with the initial states of G_{d1}^e and G_{d2}^e, respectively, and *R3* and *R4* hold the initial unobservable reaches of G_{d1}^e and G_{d2}^e, respectively.

In summary, the registers of the coordinator are updated according to the information update rule presented in Table 4.1. Once C, the coordinator's diagnostic information is F_i-certain, the coordinator broadcasts to the failure recovery module that a failure of type F_i has occurred.

4.2 Diagnostic Properties of Protocol 1

The diagnostic properties of Protocol 1 are summarized by Theorem 4.2.1, the proof of which is based on the following proposition.

Proposition 4.2.1 *Let q_1, q_2, and q be the states of the extended diagnosers G_{d1}^e, G_{d2}^e, and G_d^e, respectively, after the system executed the trace $s = s_1 a u b$, where $a, b \in \Sigma_o$ $(= \Sigma_{o1} \cup \Sigma_{o2})$, $u \in \Sigma_{uo}^*$. Denote by q_{old} the state of G_d^e after the execution of $s_1 a$. Then the following is true:*

1. *if $b \in \Sigma_{o1} \cap \Sigma_{o2}$ then*

$$(i) \quad q = (q_1 \cap_e^L q_2) \cap_c q_{old}, \ if \ a \in \Sigma_{o1}$$
$$(ii) \quad q = (q_1 \cap_e^R q_2) \cap_c q_{old},$$

2. *otherwise if $b \in \Sigma_{o1} \backslash \Sigma_{o2}$ then*

$$(i) \quad q = (q_1 \cap_e^L UR_2(q_2)) \cap_c q_{old}, \ if \ a \in \Sigma_{o1}$$
$$(ii) \quad q = (q_1 \cap_e^R UR_2(q_2)) \cap_c q_{old},$$

3. *otherwise if $b \in \Sigma_{o2} \backslash \Sigma_{o1}$ then*

$$(i) \quad q = (UR_1(q_1) \cap_e^L q_2) \cap_c q_{old}, \ if \ a \in \Sigma_{o1}$$
$$(ii) \quad q = (UR_1(q_1) \cap_e^R q_2) \cap_c q_{old}.$$

Proof of Proposition 4.2.1 Proof of *1*. We first note that

$$SP(q) \subseteq SP(q_i) \subseteq SP(UR_i(q_i)), i = \{1, 2\}. \tag{4.1}$$

The first inclusion is true since the set of observable events Σ_{oi} is a subset of the original set of observable events Σ_o and $b \in \Sigma_{o1} \cap \Sigma_{o2}$, and the second inclusion is true by definition (cf. Definition 4.1.1). In case (i) we have

$$(q_1 \cap_e^L q_2) \cap_c q_{old} = (q_1 \cap_c q_{old}) \cap_e^L q_2 \tag{4.2}$$

$$(q_1 \cap_c q_{old}) = q. \tag{4.3}$$

(4.2) follows from the definition of the intersection operators $\cap_c$ and $\cap_e^L$. (4.3) is obtained as follows: by definition, $q_1 \cap_c q_{old}$ gives all state estimate tuples in q_1 that are reached by the observable event b. Since b is the next observable event after a in s and $SP(q) \subseteq SP(q_1)$ by (4.1), $q_1 \cap_c q_{old}$ is the state of the diagnoser G_d^e which is q by definition. Combining (4.2) and $SP(q) \subseteq SP(q_2)$ from (4.1) we obtain $q = (q_1 \cap_e^L q_2) \cap_c q_{old}$. To prove case (ii), we note that by Definition 4.1.2 we can write $q_1 \cap_e^L q_2 = q_2 \cap_e^R q_1$. So by exchanging the roles of q_1 and q_2, and using the same arguments as in case (i) we have $q = (q_1 \cap_e^R q_2) \cap_c q_{old}$.

Proof of *2*. We note first that

$$SP(q) \subseteq SP(q_1) \text{ and } SP(q) \subseteq SP(UR_2(q_2)). \tag{4.4}$$

The inclusions are true since the set of observable events Σ_{oi} is a subset of the original set of observable events Σ_o and $b \in \Sigma_{o1} \backslash \Sigma_{o2}$. The proof of case (i) proceeds in the same way as the proof of *1 -(i)* with the minor modification of using $UR_2(q_2)$ instead of q_2. To prove (ii) we have

$$(q_1 \cap_e^R UR_2(q_2)) \cap_c q_{old} = q_1 \cap_e^R (UR_2(q_2) \cap_c q_{old}) \tag{4.5}$$

$$(UR_2(q_2) \cap_c q_{old}) = q. \tag{4.6}$$

(4.5) follows from the definition of the intersection operators $\cap_c$ and $\cap_e^R$. (4.6) is obtained as follows: by definition, $UR_2(q_2) \cap_c q_{old}$ gives all state estimate tuples in $UR_2(q_2)$ that are reached by the observable event b. This is true by Definition 4.1.1: $UR_2(q_2)$ may include state estimate tuples $\{(x, l), (x', l')\}$ whose current state x' may be reached by a sequence of observable events and not only by one observable event, like in the case of the event b; however in such a case the predecessor state x is by definition the immediate predecessor of

x' in G', and this predecessor does not belong to any $SP(x_i)$, where $x_i \in q_{old}$. Since b is the next observable event after a in s and $SP(q) \subseteq SP(UR_2(q_2))$ by (4.4), $UR_2(q_2) \cap_c q_{old}$ is the state of the diagnoser G_d^e which is q by definition. Combining (4.5) and $SP(q) \subseteq SP(q_1)$ from (4.4) we obtain $q = (q_1 \cap_e^R UR_2(q_2)) \cap_c q_{old}$.

Proof of *3*. Exchange the roles of q_1 and $UR_1(q_1)$ with q_2 and $UR_2(q_2)$, respectively, and proceed as in the proof of *2*. **Q.E.D.**

Proposition 4.2.1 can be used to prove the main result concerning the diagnostic properties of Protocol 1.

Theorem 4.2.1 *(i) The coordinator's diagnostic information C under Protocol 1 is the same as the state of the centralized extended diagnoser G_d^e.*
(ii) Protocol 1 achieves the same diagnostic performance as a centralized diagnoser.

Proof of Theorem 4.2.1 (i) We prove part (i) by induction on the number of observable events (in $\Sigma_o = \Sigma_{o1} \cup \Sigma_{o2}$) in the trace s.

Basis of induction: Let $|P(s)| = |b| = 1$. In this case $C_{old} = \{(x_0, N), (x_0, N)\}$ by assumption, where x_o is the initial state of the system. Moreover, by assumption both G_{d1}^e and G_{d2}^e have the same initial state $\{(x_0, N), (x_0, N)\}$. If $b \in \Sigma_{o1} \cap \Sigma_{o2}$ then $q_1 = q_2 = q$ by the construction of the diagnosers. Therefore $q_1 \cap_e^i q_2 = q$ and $q \cap_c C_{old} = q$ by definition. If $b \in \Sigma_{o1} \backslash \Sigma_{o2}$ then $q_1 = q$ by construction and $q \subseteq UR_2(q_2)$ as discussed earlier in the proof of Proposition 4.2.1. Therefore, $q_1 \cap_e^i UR_2(q_2) = q$, and $q \cap_c C_{old} = q$ by definition. The proof of the case when $b \in \Sigma_{o2} \backslash \Sigma_{o1}$ is symmetric to the case where $b \in \Sigma_{o1} \backslash \Sigma_{o2}$.

Induction step: The proof of the induction step is provided by Proposition 4.2.1 since by Assumptions **A4** and **A5** every message is received in the order it was sent.
(ii) From part (i) and the specification of the coordinator's decision rule it follows that Protocol 1 achieves the same diagnostic performance as a centralized diagnoser. **Q.E.D.**

Note that, according to Assumption **A8**, the coordinator has no knowledge of the system model, and has limited memory and limited processing capabilities. Yet, if the coordinator has the memory and processing capabilities required by the decision rule described in Section 4.1.3, it can diagnose the same types of failures as a centralized diagnoser; by receiving the extended diagnoser states (and unobservable reaches) and using the rules $\cap_e^i, i = L, R$ and $\cap_c$ the coordinator, in essence, can keep track of the state of the system in the same way as the centralized diagnoser. Consequently, it has the same diagnostic properties as the centralized diagnoser.

4.3 Necessary and Sufficient Conditions for Diagnosability

In Section 4.2, we showed that the information update rule that is used at the coordinator site is reconstructing the centralized diagnoser state. Consequently the necessary and sufficient conditions for diagnosability with respect to Protocol 1 can be stated with respect to the centralized diagnoser as follows.

Theorem 4.3.1 *A live and prefix-closed language L is diagnosable with respect to Protocol 1, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f if and only if the diagnoser G_d does not have F_i-indeterminate cycles for all failure types F_i.*

Proof of Theorem 4.3.1 Sufficiency($\Leftarrow$). Suppose G_d does not have F_i-indeterminate cycles. Then by Proposition 2.6.1, G_d^e does not have F_i-indeterminate cycles. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, and t is long enough, i.e., $\| t \| > n$, where n can be arbitrarily large. Then, by assumption, st', $t' \in \overline{t}$ does not lead to an F_i-indeterminate cycle in G_d^e. Consequently, an argument similar to the one used in the proof of Theorem 2 in [40] shows that G_d^e will enter an F_i-certain state within a finite number of steps, say n_i'. This implies by Theorem 4.2.1 that the coordinator's diagnostic information C will be F_i-certain within a finite number of steps equal to n_i'. Therefore, $L(G)$ is diagnosable with respect to Protocol 1.
Necessity($\Rightarrow$). We prove the contrapositive. Assume that G_d does enter an F_i-indeterminate cycle. Then by Proposition 2.6.1 G_d^e enters an F_i-indeterminate cycle. This implies that the coordinator's diagnostic information C, which is equal to the centralized diagnoser state, will remain F_i-uncertain for an arbitrarily long number of steps. Hence, L is not diagnosable with respect to Protocol 1. **Q.E.D.**

We conclude this section with an example that illustrates the application of Theorem 4.3.1.

Example 4.3.1 *Consider again the system shown in Figure 2.1 with $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$, $\Sigma_{o1} = \{a, c, d, e\}$, and $\Sigma_{o2} = \{b, d, e\}$. The diagnoser G_d for this system is shown in Figure 4.2. The only cycle whose states carry failure labels is $\{10F1, 9F1\}$; however its states are F_1-certain; hence there are no F_1-indeterminate cycles, and by Theorem 4.3.1 the system is diagnosable by Protocol 1.*

Table 4.2 illustrates the application of Protocol 1 when the system executes the trace abcd. For instance, after the message from site 2 regarding the occurrence of event b reaches the coordinator and all operations are performed we have $SB = 0$, $SB_{1old} = 0$ and $SB_{2old} = 1$ (the bits SB, SB_{1old}, and SB_{2old} are not shown in the table for space limitation). Once

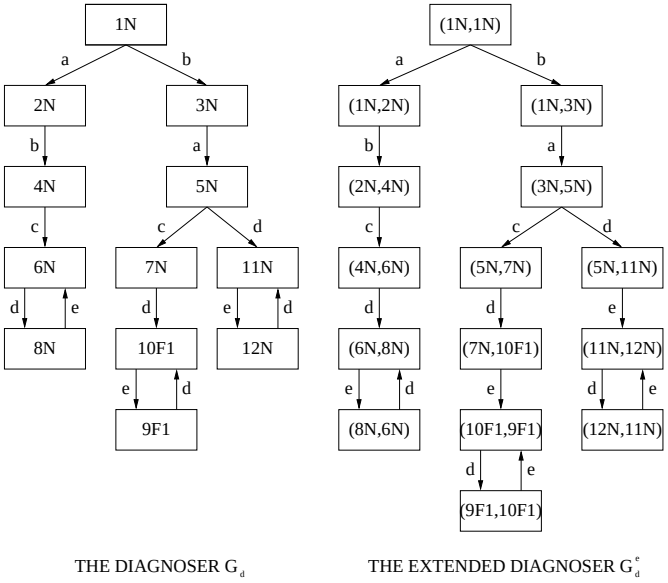

Figure 4.2: The diagnosers for Example 4.3.1

the message regarding the occurrence of the event c reaches the coordinator, the following occurs: since the event is only observed by site 1 and since $SB = 0$*, action* **DR1** *is taken; therefore, the current* $C = (2N, 4N)$ *is saved into* C_{old}*, and since* $SB_{1old} = 0$ *then* $C = (R_1 \cap_e^R R_4) \cap_c C_{old} = (\{(2N, 6N), (5N, 7N)\} \cap_e^R \{(1N, 3N), (1N, 4N), (3N, 5N), (5N, 7N), (4N, 6N)\}) \cap_c (2N, 4N) = \{(4N, 6N), (5N, 7N)\} \cap_c (2N, 4N) = (4N, 6N)$*. The registers* SB*,* SB_{1old} *and* SB_{2old} *are set to 0, 1, 0, respectively. The next observable event is d and it is observed by both sites; assume that the report from site 2 about the occurrence of event d, reaches the coordinator before the one form site 1. Once the report from site 2 reaches the coordinator, action* **DR5** *is executed, i.e.,* SB *is set to 1, because* $SB = 0$*. When the report from site 1 reaches the coordinator, action* **DR3** *is executed, which means that* $C = (R_1 \cap_e^L R_2) \cap_c C_{old} = (\{(6N, 8N), (7N, 10F1)\} \cap_e^L \{(3N, 11N), (3N, 10F1), (4N, 8N)\}) \cap_c (4N, 6N) = \{(6N, 8N), (7N, 10F1)\} \cap_c (4N, 6N) = (6N, 8N)$*. The registers* SB*,* SB_{1old} *and* SB_{2old} *are set to 0, 1, 1, respectively.*

We emphasize the need to incorporate $\cap_c$ *and* C_{old} *in the decision rule to guarantee the performance of Protocol 1. For that matter, consider that the system has just executed the trace abcd. The last row of Table 4.2 describes the content of all the registers at the coordinator site after all messages related to the occurrence of event d have reached the coordinator site. If we compute* C *by only using* $\cap_e^i, i = L, R$*, i.e.,* $C = R_1 \cap_e^L R_2$ *we get* $C = \{(6N, 8N), (7N, 10F1)\}$*, which clearly is not equal to* $(6N, 8N)$*, the state of the centralized diagnoser (cf. Figure 4.2) following the execution of abcd. However, if we incorporate* $\cap_c$ *in the decision rule we indeed reconstruct the state of the centralized diagnoser. The operator*

Table 4.2: Illustration of the application of Protocol 1

Event	$R1$	$R3$	$R2$	$R4$	C_{old}	C
ϵ	(1N,1N)	(1N,1N), (1N,3N)	(1N,1N)	(1N,1N), (1N,2N)	(1N,1N)	(1N,1N)
a	(1N,2N), (1N,5N)	(1N,2N), (1N,5N), (2N,4N)	(1N,1N)	(1N,1N), (1N,2N)	(1N,1N)	(1N,2N)
b	(1N,2N), (1N,5N)	(1N,2N), (1N,5N), (2N,4N)	(1N,3N), (1N,4N)	(1N,3N), (1N,4N), (3N,5N), (5N,7N), (4N,6N)	(1N,2N)	(2N,4N)
c	(2N,6N), (5N,7N)	(2N,6N), (5N,7N)	(1N,3N), (1N,4N)	(1N,3N), (1N,4N), (3N,5N), (5N,7N), (4N,6N)	(2N,4N)	(4N,6N)
d	(6N,8N), (7N,10F1)	(6N,8N), (7N,10F1)	(3N,11N), (3N,10F1), (4N,8N)	(3N,11N), (3N,10F1), (4N,8N)	(4N,46)	(6N,8N)

$\cap_c$ eliminates in an extended diagnoser state all current state estimates whose predecessor state estimates are not current state estimates in the old coordinator state. Since $\cap_c$ is used after the occurrence of every observable event, then by an inductive argument one can visualize that we are "memorizing" the past. In other words, it is not sufficient to retain a one-step memory as provided by the extended diagnoser; rather, an "n-step" memory, where n represents the length of the observed trace, is needed, and that is achieved by incorporating $\cap_c$ in the decision rule.

4.4 Discussion

We first note that the partitioning of observable events does not affect the diagnostic capabilities of Protocol 1: irrespective of the partitioning of the set of observable events Σ_o, as long as the centralized diagnoser is capable of identifying all failure types, so is Protocol 1, and vice-versa. This is a direct consequence of Theorem 4.2.1.

Having demonstrated that Protocol 1 is capable of diagnosing all failure types that are

diagnosed by a centralized diagnoser, one may ask the following question: is it possible to replace the extended diagnosers G^e_{d1} and G^e_{d2} by the diagnosers G_{d1} and G_{d2}, respectively, while maintaining the same fundamental structure, i.e., the same functional form of the communication rules and the coordinator's decision rule (with the obvious modifications dictated by the change from extended diagnosers to diagnosers) so that the resulting protocol achieves the same diagnostic performance as Protocol 1? The following example shows that a modification such as the above does not lead to a protocol with the same diagnostic capabilities as Protocol 1.

Example 4.4.1 *Consider the system discussed in Example 2.4.1 and shown in Figure 2.1 with* $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$, $\Sigma_{o1} = \{a, c, d, e\}$, *and* $\Sigma_{o2} = \{b, d, e\}$. *We showed in Example 4.3.1 that this system is diagnosable with respect to Protocol 1. Consider now the following protocol: the diagnosers* G_{d1}, G_{d2} *(cf. Figure 4.3) replace* G^e_{d1}, G^e_{d2}, *respectively. The fundamental structure, i.e., the functional form of the*

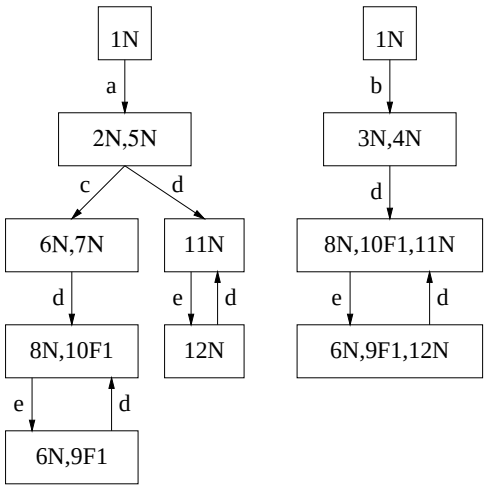

Figure 4.3: The local diagnosers for Example 4.4.1

communication rules and the coordinator's decision rule remain the same as in Protocol 1. The modifications resulting from replacing G^e_{d1}, G^e_{d2} *with* G_{d1}, G_{d2}, *respectively, lead to the following rules: the diagnosers at the local sites communicate their states, their unobservable reaches and a status bit (as in Protocol 1) to the coordinator. The coordinator declares that a failure of type* F_i *has occurred when its diagnostic information* C *is* F_i*-certain (cf. Definition 3.3.2); it broadcasts this decision to the failure recovery module. The coordinator's information update rule results from the information update rule of Protocol 1 by replacing* $\cap^i_e$ *with the regular set intersection and eliminating* $\cap_c$. *We will give a detailed description of this protocol in the next chapter. Assume now that the system is executing the*

trace abcdede...; then G_{d1} and G_{d2} are looping simultaneously in the cycles $\{(8N, 10F1), (6N, 9F1)\}$ and $\{(8N, 10F1, 11N), (6N, 9F1, 12N)\}$, respectively. Moreover these cycles are F_1-indeterminate in their respective diagnosers: traces[1] $s = abc(de)^*$ and $s' = bac\sigma(de)^*$ both lead to the two cycles (since $P_1(s) = P_1(s') = ac(de)^*$ and $P_2(s) = P_2(s') = b(de)^*$) and σ, the failure of type F_1, belongs to s' while it does not belong to s. Under the new protocol, the coordinator is not able to differentiate between the two traces: the state of the coordinator is either $(8N, 10F1)$, after the occurrence of the event d, or $(6N, 9F1)$, after the occurrence of the event e, and this continues indefinitely. We refer to such problem as the "ordering problem" in untimed DES, since using the available diagnostic information, the coordinator is not capable of "ordering" the events in traces s and s', i.e., whether the event that occurred first is a or b. In Protocol 1, this problem was solved by the use of the diagnostic information generated by extended diagnosers at the local sites and the decision rule of the coordinator presented in Section 4.1.3. However, this problem is also of concern for Protocol 1 if Assumption **A5** *that guarantees that messages are received at the coordinator in the order they are sent by different local sites is violated. Such a case is discussed and analyzed in the next section.*

Since the above example shows that the objective of diagnosing all failure types that are diagnosed using the centralized diagnoser cannot be achieved using the abovementioned alteration of Protocol 1, one may seek conditions on the system model G under which this objective can be met. This is discussed in the next chapter.

4.5 Protocol 1D: Relaxation of Assumption A5

In this section, we modify Protocol 1 to account for communication delays. We will refer to the modified protocol as **Protocol 1D** (where 1D stands for the fact that it is a modification of Protocol **1** that allows for communication **D**elays). A key feature of Protocol 1 is the following: under the assumption that all communicated messages are received in the correct order by the coordinator, the coordinator is capable of "tracking" the state of the system as well as a centralized diagnoser, even though it does not have any knowledge of the dynamics of the system. This feature is the key to understanding and analyzing the performance of Protocol 1D. When the messages are received out of order by the coordinator, the coordinator should consider all possible orders according to which the messages may have been sent and for each possible order it should use the information update rule specified by Protocol 1. By doing so the coordinator achieves the following

[1] We commit here, and thereafter in the thesis, a slight abuse of notation by using the Kleene closure * in the expression of a trace.

(see Section 4.5.4): (1) if a specific order of messages corresponds to a legal behavior of the system the coordinator's update is non-empty, and for that order it reconstructs the state of the centralized diagnoser associated with that legal behavior; (2) otherwise the coordinator rejects that order as impossible because it gives rise to an empty update. This implies that: (i) the memory requirements at the coordinator's site may increase considerably, because the coordinator has to store all the above-mentioned diagnoser states; (ii) the coordinator can identify a failure type F_i only when F_i appears in the components of all the centralized diagnoser states reconstructed as described above. Therefore, we expect that, in general, the performance of Protocol 1D will not be the same as that of Protocol 1 (and, consequently, not that of a centralized diagnoser). To achieve the same performance as a centralized diagnoser we will need to impose further structure on the system. We determine conditions under which the protocol is capable of diagnosing the same types of failures as the ones diagnosed using the centralized diagnoser.

As is the case for any protocol that realizes the coordinated decentralized architecture, we need to define the diagnostic information generated at the local sites, the communication rules used by the local sites, and the decision rule used by the coordinator to infer the occurrence of failures. This is done in the following three sub-sections.

4.5.1 Diagnostic information at local sites

The same diagnostic information defined for Protocol 1 is used. Extended diagnosers are implemented at local sites. Consequently, the diagnostic information available at each site is provided by the state of the extended diagnoser. The state information is refined by the unobservable reach as defined in 4.1.1.

4.5.2 Communication rules

The same communication rules defined for Protocol 1 are used as well. After the agent at either site observes an event, it communicates to the coordinator the corresponding state of its diagnoser, its unobservable reach, and the status bit.

4.5.3 Decision rule

The coordinator's decision rule is composed of three steps: (1) store all incoming messages at the coordinator site, and sort out all possible orders in which these messages may be generated by the local sites; (2) apply the information update rule, as defined in Section 4.1.3, to each and every possible sorted out order, and retain all orders that result in a non-empty update (intersection); (3) compare the failure properties of all surviving

updates from step (2), and declare the occurrence of a failure when all these updates are certain of the occurrence of the failure. These steps are discussed in detail in the following three sub-sections. The following analysis assumes that there are no events commonly observed by sites 1 and 2. If there are commonly observed events by both sites the sorting procedure is be extremely simplified since the commonly observed events can be used as a synchronization mechanism.

(1) Sorting out possible orders

All communication messages received at the coordinator's site are stored until the instance where all possible orders in which these messages are generated by the local sites can be figured out. Determining the instance where all possible orders can be sorted out depends on the messages received, namely which site observed an event and subsequently sent the message. In general, one can continue sorting out (uncovering) all possible orders after waiting for the arrival of at most three new messages at the coordinator's site. This is a direct consequence of the "one-step out of order" assumption introduced in Section 3.2 and it is explained below.

Table 4.3 depicts the arrival of three new messages at the coordinator's site. A message with subscript i, $i \in \{1, 2\}$, indicates it has been sent by site i. The left column in Table 4.3

Messages received	Possible orders	Sorted out orders
$x_1 y_2 z_2$	$x_1 y_2 z_2$ or $y_2 x_1 z_2$	$x_1 y_2$ or $y_2 x_1$
$x_2 y_1 z_1$	$x_2 y_1 z_1$ or $y_1 x_2 z_1$	$x_2 y_1$ or $y_1 x_2$
$x_1 y_2 z_1$	$x_1 y_2 z_1$ or $y_2 x_1 z_1$ or $x_1 z_1 y_2$	$x_1 y_2$ or $y_2 x_1$ or $x_1 z_1 y_2$
$x_2 y_1 z_2$	$x_2 y_1 z_2$ or $y_1 x_2 z_2$ or $x_2 z_2 y_1$	$x_2 y_1$ or $y_1 x_2$ or $x_2 z_2 y_1$

Table 4.3: Sorting out possible orders at the coordinator site

describes the order in which messages are received at the coordinator's site (left is earliest), the middle column describes all possible orders that may have resulted in the reception of messages by the coordinator, and the right column describes the orders sorted out. The second column of the first row in Table 4.3 means the following: either the reception order is the same as the one in which the messages are sent, or y_2 could have been sent prior to x_1 (due to the "one-step out of order" communication delay). In both cases z_2 is sent after y_2 due to preservation of local order. The third column of the first row in Table 4.3 means the following: the sorted out (uncovered) orders are $x_1 y_2$ or $y_2 x_1$ and z_2 needs to be stored until new messages are received to figure out the order in which it was generated. The second row is the symmetric to the first and the last two rows are interpreted in

a similar way. Initially, the "order sorting" procedure is to wait for three messages and afterwards uncover all possible orders of the first two messages while keeping the third message for later consideration after new messages arrive. Later, in Section 4.5.3, we will provide a more detailed description of how the sorting procedure is implemented. Also, a brief discussion of the procedure under the assumption of at most "n-step out of order" reception of messages appears in Section 7.2.

We conclude this sub-section with the following two remarks.

Remark 1 Under the "one-step out of order"assumption, some cases require to wait for the arrival of only two messages to continue the sorting out procedure. For example, if two consecutive messages x and y are received by the coordinator from the same site, say site 1, then the coordinator is sure that message x was sent first, by the local order assumption (Assumption **A4**).

Remark 2 In the case of possible orders $x_1z_1y_2$ and $x_2z_2y_1$ in rows three and four, respectively, in Table 4.3, the order in which the three messages are sent is uncovered since by the "one-step out of order" assumption messages y_2 and y_1, received at the coordinator's site prior to messages z_1 and z_2, should have been sent directly after these messages.

(2) Application of the update rule

For every possible order that is sorted out, in addition to the register C where the coordinator stores its current diagnostic information, eight supplementary registers are used for storing messages and previous relevant values necessary for the update of the information. These registers are: *R1*, *R2*, *R3*, *R4*, C_{old}, SB, SB_{1old}, and SB_{2old}, and their functions are the same as in their counterparts defined in the case of Protocol 1. Consequently, the information update rule, depicted in Table 4.1, is used at any instant when all possible orders can be sorted out at the coordinator. For every possible order that survives the update rule (that is, it results in a non-empty intersection), the coordinator keeps the last two updates along with the corresponding states and unobservable reaches and status bits. This is possible by definition of the update rule that only requires information from the last two updates to generate the new one (cf. Table 4.1). If a specific order of messages results in an empty intersection following the application of the update rule, it is rejected by the coordinator. The same procedure as above is applied when new orders are sorted out as a continuation of the already existing ones, and the new updates replace the old ones that are discarded.

Having defined the information update rule, we now provide more details on how the

"order sorting" procedure introduced in Section 4.5.3-(1) and the information update rule are implemented. Initially, the coordinator waits until it receives three messages (as depicted in Table 4.3) before sorting out the possible orders. At that instant orders including only two messages are sorted out, and the information update rule is applied to these orders. The registers for each sorted out order are updated to hold the two newest diagnostic updates along with the corresponding states and unobservable reaches and status bits. Also stored for each order is the third message (cf. Table 4.3). When two new messages are received at the coordinator's site, the coordinator sorts out the new possible orders (out of the third message that was previously stored for each existing sorted out order and the two new messages that have been received) as a continuation of a previously uncovered order. To every new sorted out order, the coordinator applies the information update rule. By successively applying the procedure just described, the coordinator sorts out the possible orders after receiving two new messages, except for "reset" steps where three new messages are needed. A "reset" step is either the first time the coordinator begins sorting out messages (discussed above) or when a previous sorting attempt results in an uncovered order that contains all the received messages (cf. the last uncovered orders in rows three and four in Table 4.3 and the explanation in Remark 2). In general, at any instant of time when the coordinator has received $2n+1$ messages, the orders of at least $2n$ messages are uncovered by the arguments presented above.

(3) Diagnosing failures

If all the information updates that result from applying the procedure described in Sections 4.5.3-(1) and 4.5.3-(2) are certain that a failure of type F has occurred, then the coordinator declares that F has occurred and broadcasts this information to the failure recovery module. Otherwise, a diagnostic decision is postponed.

4.5.4 Diagnostic properties of Protocol 1D

We prove that Protocol 1D performs as well as Protocol 1 under certain conditions on the system structure. Thus, when the coordinator receives messages out of (global) order, we have a degradation in the performance of Protocol 1.

To proceed with the analysis of Protocol 1D, we introduce the following concept.

Definition 4.5.1 *A trace $s \in L(G)$ is said to be ambiguous with respect to the projections P_1 and P_2 and the failure type F_i if there exist a set of traces $S = \{s_1, s_2, \ldots\}$ in $L(G)$ such that $s_1, s_2, \ldots$ are arbitrarily long, and the following are true:*

1. $P_1(s) = P_1(s')$, $s' \in S$,

2. $P_2(s) = P_2(s')$, $s' \in S$,

3. $P(s) \neq P(s')$, $s' \in S$,

4. $|P(s)| = |P(s')|$, $s' \in S$,

5. $F_i \in s$,

6. *and there exists at least* $s' \in S$ *such that* $F_i \notin s'$.

The following example illustrates the concept of ambiguous traces.

Example 4.5.1 *Consider the system of Example 2.4.1 shown in Figure 2.1. The set of events is* $\Sigma = \{a, b, c, d, e, \sigma\}$, *and* σ *is the only unobservable and failure event. Let* F_1 *denote the failure type of the event* σ. *Define* $\Sigma_{o1} = \{a, c, d, e\}$ *and* $\Sigma_{o2} = \{b, d, e\}$. *The trace* $s = bac\sigma(de)^n$, *for a given integer* n, *is ambiguous because there exists a trace* $s' = abc(de)^n$ *such that: (1)* $P_1(s) = P_1(s') = ac(de)^n$ *(2)* $P_2(s) = P_2(s') = b(de)^n$, *(3)* $P(s) = bac(de)^n \neq abc(de)^n = P(s')$, *(4)* $|P(s)| = |bac(de)^n| = |abc(de)^n| = |P(s')|$, *(5)* $\sigma \in F_1$ *belongs to* s, *and (6)* $F_1 \notin s'$.

The main result concerning the performance of Protocol 1D is summarized by the following theorem.

Theorem 4.5.1 *If* $L(G)$ *is diagnosable with respect to Protocol 1, then Protocol 1D eventually performs as well as Protocol 1 if and only if there are no ambiguous traces in* $L(G)$ *with respect to all failure types.*

The proof of Theorem 4.5.1 is based on an important property of Protocol 1D that is summarized by the following proposition.

Proposition 4.5.1 *Consider any possible sorted out ordering of messages, say* w, *produced by the rule of Section 4.5.3-(1) when* $s \in L(G)$ *is executed. If* w *is not the correct order in which messages are sent to the coordinator, then the update rule of Section 4.5.3-(2) applied to* w *results in a non-empty intersection if and only if* w *belongs to* $P(L)$. *The non-empty intersection resulting from the application of the update rule of Section 4.5.3-(2) on* w *gives the state of the centralized extended diagnoser corresponding to* w.

Proof of Proposition 4.5.1 Consider that the system is executing the trace $s_0u_1au_2bu_3c$ where $a \in \Sigma_{o1}$, $b \in \Sigma_{o2}$, $c \in \Sigma_{o1}$, and $u_1, u_2, u_3 \in \Sigma_{uo}^*$. Denote by x, y, and z the messages generated by the occurrence of events a, b, and c, respectively. Without loss of generality

assume that the messages are received in the following order: x, y, then z. This covers the case presented in the third row of Table 4.3 (the fourth row is the symmetric case of the third row while by inspection one realizes that the cases presented in rows 1 and 2 are sub-cases of those presented in rows 3 and 4, respectively). Let q_1 and q_3 be the states of the diagnoser G^e_{d1} after the execution of the the event a and c, respectively, and q_2 be the state of the diagnoser G^e_{d2} after the execution of the event b. Denote by q_{old}, q_{1old}, and q_{2old} the states of the diagnosers G^e_d, G^e_{d1}, and G^e_{d2}, respectively, following the execution of s_0.

We have, from Table 4.3 and the sorting procedure presented in Section 4.5.3-(1), at most three possible continuations of the orders in which messages were sent. These orders are xy, yx, and xzy. One of them is the true one. Assume without loss of generality that it is xy (corresponding to the observation of ab). From Theorem 4.2.1 we know that the application of the update rule of Section 4.5.3-(2) to xy results in the state of the centralized extended diagnoser following the observation of $P(s_0)ab$. Hence, we need to check the result of the proposition for the orders yx (corresponding to the sequence of observable events ba) and xzy (corresponding to the sequence of observable events acb). In the remaining we do the proof for the order yx, and by the same argument we can prove the result for the order xzy. Denote by C_a the coordinator state resulting from applying the update rule to the order yx. We have the following:

$$C_a = \quad (q_1 \cap^R_e UR_2(q_2)) \cap_c C_b \quad := C_2 \cap_c C_b, \tag{4.7}$$

where

$$C_b = \quad (UR_1(q_{old1}) \cap^i_e q_2) \cap_c C_{old} \quad := C_1 \cap_c C_{old}, \tag{4.8}$$

and C_{old} is the state of the coordinator following the order $P(s_0)$.

We prove the proposition by induction on the number of observable events (in Σ_o) in s_0. We first present the proof of the induction step, and the proof of the basis of induction follows from it by minor modifications.

- Induction step. We assume that: (i) for any trace $t_1 \in L(G)$ with $|P(t_1)| \leq n$ all possible orderings (different from the correct one) produced by the rule of Section 4.5.3-(1) when t_1 is executed result in a sequence of non-empty intersections if and only if the orderings belong to $P(L)$; and (ii) the coordinator state C following any of the possible orderings belonging to $P(L)$ is equal to the corresponding state of the extended diagnoser. We want to prove that the same result is true for any trace $t_2 \in L(G)$ with $|P(t_2)| \geq n + 2$[2].

[2]Proving the result for the order yx requires considering a trace having $n + 2$ observable events, while

Let $t_2 = s_0 u_1 a u_2 b u_3 c$, where $|P(s_0)| = n$. We have from the induction hypothesis that C_{old} is equal to the state of the extended diagnoser following the observation $P(s_0)$, that is,

$$C_{old} = \delta_d^e(q_0, P(s_0)) = q_{old}. \tag{4.9}$$

Sufficiency ($\Leftarrow$). Assume $P(s_0)ba \in P(L)$ (we remind the reader that we prove the proposition for the order yx corresponding to the sequence of observable events ba). Since $P(s_0)ba \in P(L)$, applying the update rule for the order $P(s_0)ba$ results in the corresponding state of the centralized extended diagnoser by the results of Theorem 4.2.1.

Necessity ($\Rightarrow$). To prove necessity we exploit (4.7), (4.8), and the fact that the intersections C_a and C_b are non-empty. Assume that the update rule of section 4.5.3-(2) applied to the order $P(s_0)ba$ results in a set of non-empty intersections. We want to prove that $P(s_0)ba \in P(L)$. The update rule applied to $P(s_0)ba$ gives (4.7). By assumption C_a is non-empty; consequently, C_b and C_1 are non-empty. Hence, by the definition of $\cap_e^i$, C_1 is an extended diagnoser state, and it is of the form

$$C_1 = \{((x_{old1}, l_{old1}), (x_{new1}, l_{new1})), \ldots, ((x_{oldr}, l_{oldr}), (x_{newr}, l_{newr}))\}. \tag{4.10}$$

Since C_1 is non-empty there exist traces in $L(G)$ whose projection with respect to P_2 equals $P_2(s_0)b$. Then by (4.8) we obtain

$$C_b = \{((x_{oldn}, l_{oldn}), (x_{newn}, l_{newn})), \ldots, ((x_{oldr'}, l_{oldr'}), (x_{newr'}, l_{newr'}))\}, \tag{4.11}$$

where $n \geq 1$, $r' \leq r$, and

$$x_{oldi} \in SP(q_{old}), i = n, \ldots, r'. \tag{4.12}$$

From (4.8), (4.11), (4.12), and the definition of the state of the extended diagnoser, we conclude that there exist traces in $L(G)$ whose projection with respect to P equals $P(s_0)b$. Hence $P(s_0)b \in P(L)$, and

$$C_b = \delta_d^e(q_{old}, b) = \delta_d^e(q_0, P(s_0)b), \tag{4.13}$$

by the results of Proposition 4.2.1.

Since by assumption C_a in (4.7) is non-empty, C_2 in (4.7) is non-empty. But

$$C_2 = \{((y_{old1}, k_{old1}), (y_{new1}, k_{new1})), \ldots, ((y_{olds}, k_{olds}), (y_{news}, k_{news}))\}. \tag{4.14}$$

proving the result for the order xzy requires considering a trace having $n+3$ observable events. This is due to the fact that the uncovered order in the case of yx includes two additional messages while that of the order xzy includes three additional messages.

Therefore, there exist traces in $L(G)$ whose projection with respect to P_1 equals $P_1(s_0)a$. By (4.7), (4.13), and (4.14)

$$C_a = \{((y_{oldn}, k_{oldn}), (y_{newn}, k_{newn})), \ldots, ((y_{olds'}, k_{olds'}), (y_{news'}, k_{news'}))\}, \quad (4.15)$$

where $n \geq 1$, $s' \leq s$, and

$$y_{oldi} \in SP(C_b), i = n, \ldots, s'. \quad (4.16)$$

From (4.11), (4.13), (4.15), and (4.16) we conclude that there exist traces in $L(G)$ whose projection with respect to P equals $P(s_o)ba$. Hence, $P(s_o)ba \in P(L)$, and consequently

$$C_a = \delta_d^e(C_b, a) = \delta_d^e(q_{old}, ba) = \delta_d^e(q_0, P(s_0)ba), \quad (4.17)$$

by Theorem 4.2.1. This completes the proof of the necessity of the induction step.

- Basis of induction. The proof of the basis of induction is similar to that of the induction step with the minor change of letting $s_0 = \epsilon$. Consequently, $C_{old} = q_{old} = (1N, 1N)$ in (4.9) by definition of the protocol, and the rest of the proof remains as is. **Q.E.D.**

Remark 3 We note that by the sorting procedure described in Section 4.5.3-(1), the sorted out orders $P(s_0ba)$ and $P(s_0acb)$ in the proof of Proposition 4.5.1 result from the same set of messages received by the coordinator when $P(s_0abc)$ occurs. Hence the sorted out order $P(s_0ba)$ (resp. $P(s_0acb)$) leads to the diagnosers state q_1 and q_2 (resp. q_3 and q_2), and the following is true

$$\begin{aligned} P_1(s_0ab) &= P_1(s_0ba), \\ P_2(s_0ab) &= P_2(s_0ba), \\ |P(s_0ab)| &= |P(s_oba)|, \end{aligned} \quad (4.18)$$

for the sorted out order $P(s_0ba)$, and

$$\begin{aligned} P_1(s_0abc) &= P_1(s_0acb), \\ P_2(s_0abc) &= P_2(s_0acb), \\ |P(s_0abc)| &= |P(s_oacb)|, \end{aligned} \quad (4.19)$$

for the sorted out order $P(s_0acb)$. This fact is used in the proof of Theorem 4.5.1 which follows.

Proof of Theorem 4.5.1 Suppose $s \in L(G)$ is executed, where s is arbitrarily long, and $F_i \in s$. According to Proposition 4.5.1, all possible orders of observable events that survive the information update rule at the coordinator site as a result of the execution of s correspond to some $P(s')$, where $s' \in L(G)$. Furthermore s and s' satisfy Properties $1-4$ of Definition 4.5.1 (see Remark 3). Let $S'(s)$ be the set of all such traces $s' \in L(G)$.
Sufficiency ($\Leftarrow$). Suppose $L(G)$ has no ambiguous traces with respect to all failure types. From the definition of ambiguous traces (Definition 4.5.1), we conclude that F_i belongs to all $s' \in S'(s)$. Consequently, since $L(G)$ is diagnosable with respect to Protocol 1, F_i is diagnosed by the coordinator.
Necessity ($\Rightarrow$). Suppose Protocol 1D performs as well as Protocol 1 which by assumption diagnoses all failure types. Since $F_i \in s$ by assumption, then F_i is diagnosed by Protocol 1D. Therefore, F_i belongs to all traces $s' \in S'(s)$. Hence, s is not an ambiguous trace. **Q.E.D.**

Note here that a trace that is not ambiguous may contain prefixes that are ambiguous. Hence, the proof of Theorem 4.5.1 holds true once the system has executed enough events to overcome the ambiguous sub-traces. Therefore, Protocol 1D identifies, under the condition of this theorem, the same failures as the centralized diagnoser but with higher bounded delay.

4.5.5 Remarks

The key features of Protocol 1D are: (1) it achieves the same performance as the centralized diagnoser when there are no ambiguous traces. Hence, the absence of global ordering of the messages received at the coordinator's site prevents the protocol from achieving the same performance as Protocol 1 which achieves the same diagnostic performance as the centralized diagnoser under no restrictions on the system structure. (2) The delay of its diagnostic decision is higher than the delay of the centralized diagnoser or that of Protocol 1. (3) The memory required at the coordinator's site to implement the protocol is larger than that required in Protocol 1.

Proposition 4.5.1 has a significant implication on the implementation of Protocol 1D. Since the update rule only reconstructs states of the centralized (extended) diagnoser, the maximum number, at one time, of surviving updates is bounded by the order of the state space of the extended diagnoser. Although this is a very loose bound, nevertheless it proves that the implementation of the protocol requires finite memory.

Finally, Protocol 1D (as well as Protocol 1) requires continuous communication between the coordinator and the local sites: the update rule at any instant of time requires information from the previously updated coordinator state to generate the new coordinator

state. Therefore, interruption of communication is not feasible under Protocol 1D, and consequently savings on communication cannot be achieved.

CHAPTER 5

PROTOCOL 2

We present a second protocol, called Protocol 2, that realizes the coordinated decentralized architecture and we initially study its diagnostic properties under Assumptions **A1** - **A8**, discussed in Section 3.1, and some additional constraints on the system structure presented in Section 5.1. Afterwards we modify Protocol 2 to account for the relaxation of Assumption **A5** on the preservation of global order and we analyze the diagnostic properties of the modified protocol.

5.1 Objective and Assumptions

In this chapter we present a protocol, called Protocol 2, that has the following features: (i) it employs diagnosers at the local sites (instead of extended diagnosers); (ii) it maintains the same functional form of the communication and decision rules as in Protocol 1; (iii) under certain conditions, identified below, it achieves the same performance as the centralized diagnoser.

To identify conditions under which Protocol 2 achieves the same performance as that of the centralized diagnoser, we introduce the notions of *state-ambiguous* and *failure-ambiguous* traces with respect to the projections P_1 and P_2. State-ambiguous traces are defined as follows.

Definition 5.1.1 *A trace $s \in L(G)$ is said to be state-ambiguous with respect to the projections P_1 and P_2 if there exist two traces, s' and s'' in $L(G)$ such that s' and s'' are arbitrarily long, not necessarily distinct, and the following is true:*

1. *$P_1(s) = P_1(s')$ but $P(s) \neq P(s')$,*
2. *$P_2(s) = P_2(s'')$ but $P(s) \neq P(s'')$,*
3. *s' and s'' share the same failure properties, i.e., a failure of type F_i, $i \in \{1, \ldots, m\}$,*

belongs to s' if and only if a failure of type F_i (not necessarily the same failure event) belongs to s''.

This definition says that the traces s, s' and s, s'' can be distinguished under the projection P; however s and s' are not distinguishable under P_1 while s and s'' are not distinguishable under P_2. Furthermore, s' and s'' have similar failure properties. Thereafter when we refer to a trace as being state-ambiguous, the projections P_1 and P_2 will be understood from the context. The concept of "state-ambiguous traces" is illustrated by the following example.

Example 5.1.1 *Consider the system of Example 2.4.1 shown in Figure 2.1. The set of events is $\Sigma = \{a, b, c, d, e, \sigma\}$, and σ is the only unobservable and failure event. $\Sigma_{o1} = \{a, c, d, e\}$ and $\Sigma_{o2} = \{b, d, e\}$. If we consider the traces $s = bac\sigma(de)^*$ and $s' = s'' = abc(de)^*$, then s is state-ambiguous since (1) $P_1(s) = P_1(s') = ac(de)^*$ but $P(s) = bac(de)^* \neq abc(de)^* = P(s')$, (2) $P_2(s) = P_2(s'') = b(de)^*$ but $P(s) = bac(de)^* \neq abc(de)^* = P(s'')$ and (3) s', s'' being equal share the same failure properties. Example 5.3.1, presented in Section 5.3, provides another example of a state-ambiguous trace where $s' \neq s''$.*

Failure-ambiguous traces are defined as follows.

Definition 5.1.2 *A trace $s \in L(G)$ is said to be failure-ambiguous with respect to the projections P_1 and P_2 and the failure type F_i if there exist two traces, s' and s'' in $L(G)$ such that s' and s'' are arbitrarily long, not necessarily distinct, and the following is true:*

1. *$P_1(s) = P_1(s')$ but $P(s) \neq P(s')$,*
2. *$P_2(s) = P_2(s'')$ but $P(s) \neq P(s'')$,*

3a. *$F_i \in s$ but $F_i \notin s'$.*

3b. *$F_i \in s$ but $F_i \notin s''$.*

4. *s' and s'' share the same failure properties, i.e., a failure of type F_j, $j \in \{1, \ldots, m\}$, $j \neq i$, belongs to s' if and only if a failure of type F_j (not necessarily the same failure event) belongs to s''.*

A failure-ambiguous trace s is also state-ambiguous. However, not every state-ambiguous trace is failure-ambiguous since for that to be true s' and s'' should not share the same failure type F_i with s. Therefore, by definition, we have that the class of failure-ambiguous

traces is a subset of the class of state-ambiguous traces. Thereafter when we refer to a trace as being failure-ambiguous, the projections P_1 and P_2 and the failure type F_i will be understood from the context. The concept of "failure-ambiguous traces" is illustrated by the following example.

Example 5.1.2 *Consider the system of Example 2.4.1 shown in Figure 2.1. The set of events is* $\Sigma = \{a, b, c, d, e, \sigma\}$, *and* σ *is the only unobservable and failure event.* $\Sigma_{o1} = \{a, c, d, e\}$ *and* $\Sigma_{o2} = \{b, d, e\}$. *If we consider the traces* $s = bac\sigma(de)^*$ *and* $s' = s'' = abc(de)^*$, *then* s *is failure-ambiguous since we showed in Example 5.1.1 that* s *is state-ambiguous and moreover* $s' = s''$ *exhibit only normal behavior while* s *has a failure of type* F_1 *(conditions 3a,3b). Example 5.3.1, presented in Section 5.3, provides another example of a failure-ambiguous trace where* $s' \neq s''$.

We will study the performance of Protocol 2 under Assumptions **A1** - **A8** (cf. Section 3.1) and one of the following additional assumptions.

A9 There are no state-ambiguous traces in $L(G)$.

A9′ There are no failure-ambiguous traces (with respect to all failure types) in $L(G)$.

The study of the performance of Protocol 2 under the set of assumptions **A1** – **A9** is performed for the purpose of comparing its performance to that of Protocol 1. The study of the performance of Protocol 2 under the set of assumptions **A1** – **A8** and **A9′** is performed for the purpose of comparing its performance to that of Protocol 3, introduced later in Chapter 6.

5.2 Specification of the Protocol

In this section, we present in detail Protocol 2, a protocol that realizes the coordinated decentralized architecture presented in Section 3.1. The specification of the protocol is done under the Assumptions **A1** - **A8** of Section 3.1.

5.2.1 Diagnostic information at local sites

We begin by specifying the type of diagnostic information generated at local sites. Since diagnosers are implemented at local sites, then the diagnostic information available at each site is provided by the state of the diagnoser. The state information is refined by the *unobservable reach* which is defined as follows.

Definition 5.2.1 *Let* $q = \{(x_1, l_1), \cdots, (x_n, q_n)\}$ *be a diagnoser state. Define the set*

$$S_j(q) \;=\; \{s \in (\Sigma\backslash\Sigma_{oj})^* : s \in L_\sigma(G, x_k) \text{ for some } \sigma \in \Sigma_{oi}, i \in \{1, 2, \ldots, m\}\backslash\{j\}, \text{ and some } k \in \{1, \ldots, n\}\}.$$

Then the unobservable reach of q with respect to $\Sigma\backslash\Sigma_{oj}$ is defined as follows:

$$UR_j(q) = \{q\} \cup \bigcup_{s \in S_j(q)} \{(y_s, l_s)\}$$

where (i) y_s is the successor of some x_k, $k \in \{1, \ldots, n\}$, after sub-trace $s \in S_j(q)$, and (ii) l_s is the failure label corresponding to y_s, obtained by propagating the label l_k of x_k according to the label propagation function defined in [40].

By definition the diagnoser state only represents those states that are reached following an observable event; the unobservable reach appends to the diagnoser state the states that are reached through unobservable events following that observable event up to an event observable by the other sites. The concept of unobservable reach, in the case of two sites, is illustrated by the following example.

Example 5.2.1 *Consider the system discussed in Example 2.4.1 and shown in Figure 2.1. The set of events is $\Sigma = \{a, b, c, d, e, \sigma\}$, and σ is the only unobservable and failure event. We denote by F_1 the failure label of the event σ. Let $\Sigma_{o1} = \{a, c, d, e\}$ and $\Sigma_{o2} = \{b, d, e\}$ be the set of observable events at sites 1 and 2, respectively, and let G_{d1} and G_{d2}, shown in Figure 4.3, be the corresponding diagnosers. We illustrate how the unobservable reach of state $(3N, 4N)$ in G_{d2} is constructed. We first consider sequences of events that the system may execute from state $(3N, 4N)$ and are composed of events that are not observed by site 2 and these sequences end with an event observable by site 1. The considered sequences are: $\{a, ac, c\}$. Note that the sequence $ac\sigma$ is not considered since it ends with an event that is not observed by site 1. The event a leads from state 3 to state 5 following normal behavior. Also from state 3 the sequence of events ac leads to state 7 following normal behavior. The event c leads from state 4 to state 6 following normal behavior. Hence the unobservable reach of state $(3N, 4N)$ is $(3N, 4N, 5N, 6N, 7N)$, where $3N$ and $4N$ belong to the unobservable reach by definition.*

5.2.2 Communication rules

To define the communication rules, we first note that right after the occurrence of an event that is observable only by one site, say i, the state of the diagnoser at site $j \neq i$ does not contain the true system state. Therefore, to efficiently communicate information to the coordinator, each local site must augment the state of its diagnoser with some additional information, the unobservable reach. We define the communication rules **CR** := (**CR1**,**CR2**) as follows:

- **[CRi]**, $i = 1, 2$: After the agent at site i observes an event $\sigma \in \Sigma_{oi}$, it communicates to the coordinator the corresponding state q_i of its diagnoser G_{di}, its unobservable reach $UR_i(q_i)$ with respect to $\Sigma \backslash \Sigma_{oi}$, and a status bit, $\mathbf{SB_i}$, that takes the values $\mathbf{SB_i} = 1$ when $\sigma \in \Sigma_{oj}$, $j \in \{1, 2\}$, $j \neq i$, or $\mathbf{SB_i} = 0$ when $\sigma \notin \Sigma_{oj}$.

5.2.3 Decision rule

The decision rule of the coordinator consists of two components : (1) a rule according to which its information is updated; and (2) a rule according to which failure occurrences are declared and broadcast to the failure recovery module. To specify the information update rule we first describe the structure of the coordinator. The coordinator has five registers, (*R1*, *R2*, *R3*, *R4*, *SB*), besides the register C that holds its diagnostic information. The five registers are used to store incoming messages from the local sites and previous relevant values necessary for the update of its information. R_1 and R_2 hold the latest states of G_{d1} and G_{d2}, respectively, R_3 and R_4 hold the latest unobservable reaches of G_{d1} and G_{d2}, respectively, and SB specifies whether to apply the information update rule to the available information in the registers ($SB = 0$) or wait for the next incoming message ($SB = 1$). At reset, *R1* and *R2* are initialized with the initial states of G_{d1} and G_{d2}, respectively, while *R3* and *R4* hold the unobservable reaches of the initial state of G_{d1} and G_{d2}, respectively. The register SB is initially set to 0. The information update rule is specified in Table 5.1. Based on the available information, the rule picks one of the actions **DR1** - **DR6**. The rationale behind the actions is the following: to compute C we intersect the states of the diagnosers if both sites have observed the last event. In case the true system state is not in the diagnoser state (since the last observable event was not seen by the diagnoser), we use the unobservable reach of the diagnoser instead of its state in the intersection with the other diagnoser's state. We finally note that the coordinator need not be aware of the rationale behind its actions; it simply updates its information and declares the occurrence of failures according to the decision rule described above.

The coordinator declares that a failure has occurred when its diagnostic information C is F_i-certain (cf. Definition 3.3.2).

5.3 Diagnostic Properties of Protocol 2

5.3.1 Diagnostic properties of Protocol 2: no state-ambiguous traces

The diagnostic properties of Protocol 2, if there are no state ambiguous traces in $L(G)$, are summarized by Theorem 5.3.1, whose proof is based on the following proposition.

Table 5.1: Information update rule at the coordinator site (Protocol 2)

Last report received from G_{d1} | **Last report received from G_{d2}**									
Rule	SB	SB_1	C	$New\ SB$	**Rule**	SB	SB_2	C	$New\ SB$
DR1	0	0	$R_1 \cap R_4$	0	**DR4**	0	0	$R_2 \cap R_3$	0
DR2	0	1	Wait	1	**DR5**	0	1	Wait	1
–	1	0	Impossible	-	–	1	0	Impossible	-
DR3	1	1	$R_1 \cap R_2$	0	**DR6**	1	1	$R_1 \cap R_2$	0

Proposition 5.3.1 *Let q_1, q_2, and q be the states of the diagnosers G_{d1}, G_{d2}, and G_d, respectively, after the system executed the trace $s = s_1 a$, where $a \in \Sigma_o$. If there are no state-ambiguous traces in $L(G)$, then we have the following:*

1. $q = q_1 \cap q_2$ *if* $a \in \Sigma_{o1} \cap \Sigma_{o2}$.
2. $q = q_1 \cap UR_2(q_2)$ *if* $a \in \Sigma_{o1} \backslash \Sigma_{o2}$.
3. $q = UR_1(q_1) \cap q_2$ *if* $a \in \Sigma_{o2} \backslash \Sigma_{o1}$.

Proof of Proposition 5.3.1 Proof of *1*: We first note that

$$q \subseteq q_i, i = \{1, 2\}. \tag{5.1}$$

The inclusion is true since the set of observable events Σ_{oi} is a subset of the original set of observable events Σ_o, and $a \in \Sigma_{o1} \cap \Sigma_{o2}$. From (5.1) we have that

$$q \subseteq q_1 \cap q_2. \tag{5.2}$$

Next we show that

$$q_1 \cap q_2 \subseteq q. \tag{5.3}$$

To prove (5.3) we proceed by contradiction. Assume that there exists $(x, l_x) \in Q_o$ such that

$$(x, l_x) \in q_1 \cap q_2 \text{ but } (x, l_x) \notin q. \tag{5.4}$$

By construction we know that $\delta(x_0, s) \neq x$ since if x were the true system state then we should have $(x, l_x) \in q$. The fact that $\delta(x_0, s) \neq x$ implies that there exists a state $y \in X$ such that $\delta(x_0, s) = y$ and (y, l_y) belongs to q, q_1 and q_2, where l_y is the failure label associated with trace s. Therefore, there exist traces s', s'' in $L(G)$, s', s'' not necessarily

distinct, such that

$$\begin{aligned} \delta(x_0, s') &= x, \\ \delta_{d1}(q_{01}, s') &= q_1, \\ P_1(s) &= P_1(s'), \\ \delta(x_0, s'') &= x, \\ \delta_{d2}(q_{02}, s'') &= q_2, \\ P_2(s) &= P_2(s''). \end{aligned} \tag{5.5}$$

Moreover, s', s'' share the same failure properties since the label associated with state x along s' and s'' is the same (and equal to l_x in our notation). In addition, since $(x, l_x) \notin q$,

$$\delta_d(q_0, s') = q' \neq q, \ \delta_d(q_0, s'') = q'' \neq q. \tag{5.6}$$

From (5.6) it follows that

$$P(s) \neq P(s') \text{ and } P(s) \neq P(s''). \tag{5.7}$$

From (5.5), (5.7) and the fact that s', s'' share the same failure properties it follows that s is a state-ambiguous trace. Hence, by contradiction (5.3) is true. From (5.2) and (5.3) it follows that

$$q = q_1 \cap q_2. \tag{5.8}$$

Proof of *2*: We first note that

$$q \subseteq q_1, \text{ and } q \subseteq UR_2(q_2). \tag{5.9}$$

The first inclusion is true as argued in the proof of *1*, and the second follows from the definition of the unobservable reach (cf. Definition 5.2.1) since $a \notin \Sigma_{o2}$. The remaining of the proof proceeds as in *1* with the minor change of replacing q_2 with $UR_2(q_2)$.
Proof of *3*: We first note that

$$q \subseteq q_2, \text{ and } q \subseteq UR_1(q_1). \tag{5.10}$$

The first inclusion is true as argued in the proof of *1*, and the second follows from the definition of the unobservable reach (cf. Definition 5.2.1) since $a \notin \Sigma_{o1}$. The remaining of the proof proceeds as in *1* with the minor change of replacing q_1 with $UR_1(q_1)$. **Q.E.D.**

Proposition 5.3.1 is used to prove the main result concerning the diagnostic properties of Protocol 2 if there are no state-ambiguous traces in $L(G)$.

Theorem 5.3.1 *If there are no state-ambiguous traces in $L(G)$, we have the following:*
(i) The coordinator's diagnostic information C under Protocol 2 is the same as the state of the centralized diagnoser G_d.
(ii) Protocol 2 achieves the same diagnostic performance as the centralized diagnoser.

Proof of Theorem 5.3.1 The proof of (i) is a direct consequence of Proposition 5.3.1 since every message is received in the order it was sent (by Assumptions **A4** and **A5**). From part (i) and the specification of the coordinator's decision rule, it follows that Protocol 2 achieves the same diagnostic performance as the centralized diagnoser. **Q.E.D.**

A consequence of Proposition 5.3.1 and Theorem 5.3.1 is the ability to save on communication (by skipping messages) while maintaining the same diagnostic performance. This is due to the fact that the information update rule (cf. Table 5.1) only uses the current diagnosers' states and unobservable reaches to compute the diagnostic information C. Hence, at any instant of time after, if the latest diagnosers' states and unobservable reaches are made available to the coordinator (following a procedure to be determined), then the results of Theorem 5.3.1 hold. Thus communication is reduced while achieving the same performance. Situations where savings in communication are of paramount importance arise in networks where the nodes (sites) are low energy battery-powered mobile units (cf. [51]). This property of Protocol 2 is also exploited in the modification of the protocol to account for the relaxation of the global order assumption (cf. Section 5.6).

5.3.2 Diagnostic properties of Protocol 2: no failure-ambiguous traces

In this section, we study the performance of Protocol 2 if there are no failure-ambiguous traces in $L(G)$. Based on the "failure-ambiguous traces" concept we have the following proposition.

Proposition 5.3.2 *Let q_1, q_2, and q be the states of the diagnosers G_{d1}, G_{d2}, and G_d, respectively, after the system executed the trace $s = s_1 a$, where $a \in \Sigma_o$. If there are no failure-ambiguous traces (with respect to failure type F_i) in $L(G)$, then we have the following:*

1. *If $a \in \Sigma_{o1} \cap \Sigma_{o2}$ then q is F_i-certain if and only if $q_1 \cap q_2$ is F_i-certain.*
2. *If $a \in \Sigma_{o1} \backslash \Sigma_{o2}$ then q is F_i-certain if and only if $q_1 \cap UR_2(q_2)$ is F_i-certain.*
3. *If $a \in \Sigma_{o2} \backslash \Sigma_{o1}$ then q is F_i-certain if and only if $UR_1(q_1) \cap q_2$ is F_i-certain.*

Proof of Proposition 5.3.2 Proof of *1*: From (5.2) we have that q is F_i-certain if $q_1 \cap q_2$ is F_i-certain. We need to prove that q is F_i-certain only if $q_1 \cap q_2$ is F_i-certain. We proceed

by contradiction. Assume that q is F_i-certain but $q_1 \cap q_2$ is not. Assume that $\delta(x_0, s) = x$. By construction there exists $(x, l_x), (y, l_y) \in Q_o, F_i \in l_x, F_i \notin l_y$ such that

$$(x, l_x), (y, l_y) \in q_1 \cap q_2 \text{ but } (y, l_y) \notin q. \tag{5.11}$$

Therefore, there exist traces s', s'' in $L(G)$, s', s'' not necessarily distinct, such that

$$\begin{aligned} \delta(x_0, s') &= y, \\ \delta_{d1}(q_{01}, s') &= q_1, \end{aligned}$$

$$\begin{aligned} P_1(s) &= P_1(s'), \\ \delta(x_0, s'') &= y, \\ \delta_{d2}(q_{02}, s'') &= q_2, \\ P_2(s) &= P_2(s''). \end{aligned} \tag{5.12}$$

Moreover, s', s'' share the same failure properties since the label associated with state y along s' and s'' is the same (and equal to l_y in our notation). In addition since we know that q is F_i-certain and $F_i \notin l_y$, then s, s' and s, s'' do not share the failure property F_i. Moreover, since $(y, l_y) \notin q$ we have that

$$\delta_d(q_0, s') = q' \neq q, \ \delta_d(q_0, s'') = q'' \neq q. \tag{5.13}$$

From (5.13) it follows that

$$P(s) \neq P(s') \text{ and } P(s) \neq P(s''). \tag{5.14}$$

From (5.12), (5.14) and the facts that s', s'' share the same failure properties and s, s' and s, s'' do not share the failure type F_i it follows that s is a failure-ambiguous trace (with respect to failure type F_i). Hence, by contradiction q is F_i-certain only if $q_1 \cap q_2$ is F_i-certain.

Proof of *2*: From (5.9) we have that q is F_i-certain if $q_1 \cap UR_2(q_2)$ is F_i-certain. We need to prove that q is F_i-certain only if $q_1 \cap UR_2(q_2)$ is F_i-certain. The proof proceeds as in *1* with the minor change of replacing q_2 with $UR_2(q_2)$.

Proof of 3: From (5.10) we have that q is F_i-certain if $UR_1(q_1) \cap q_2$ is F_i-certain. We need to prove that q is F_i-certain only if $UR_1(q_1) \cap q_2$ is F_i-certain. The proof proceeds as in *1* with the minor change of replacing q_1 with $UR_1(q_1)$. **Q.E.D.**

Proposition 5.3.2 is used to prove the main result concerning the diagnostic properties of Protocol 2 if there are no failure-ambiguous traces in $L(G)$.

Theorem 5.3.2 *If there are no failure-ambiguous traces in $L(G)$, Protocol 2 achieves the same diagnostic performance as the centralized diagnoser.*

Proof of Theorem 5.3.2 The proof is a direct consequence of Proposition 5.3.2 (since every message is received in the order it was sent by Assumptions **A4** and **A5**) and the specification of the coordinator's decision rule. **Q.E.D.**

5.3.3 Discussion

Theorem 5.3.1*(i)* states that the coordinator's diagnostic information is equal to the centralized diagnoser state if there are no state-ambiguous traces in $L(G)$. However, for the purpose of failure diagnosis this is not necessary. To diagnose all failure types the coordinator's diagnostic information should be F_i-certain (after the occurrence of a failure of type F_i) if and only if the centralized diagnoser state is F_i-certain, without the need of having the two entities equal. This has been established if there are no failure-ambiguous traces in $L(G)$. Since absence of failure-ambiguous traces is less restrictive than absence of state-ambiguous traces, the result of Theorem 5.3.2 is stronger than that of Theorem 5.3.1.

Although for the purpose of failure diagnosis it is more appropriate to study the performance of Protocol 2 if there are no failure-ambiguous traces in $L(G)$, Theorem 5.3.1 is used to compare the performance of Protocol 2 with that of Protocol 1. Theorem 4.2.1 states that irrespective of the system structure, Protocol 1 reconstructs the state of the centralized diagnoser and hence performs as well as the centralized diagnoser. Therefore, the performance of Protocol 2 is inferior to that of Protocol 1 since reconstructing the state of the centralized diagnoser and performing as well as the centralized diagnoser is conditioned on the fact that there are no state-ambiguous traces.

The following example shows that if the system contains state-ambiguous or failure-ambiguous traces the results of Propositions 5.3.1 and 5.3.2 are not, in general, true.

Example 5.3.1 *Consider the system shown in Figure 5.1. The set of events is $\Sigma = \{a, b, c, d, e, \sigma, \sigma_1\}$, $\Sigma_{uo} = \{\sigma, \sigma_1\}$, $\Sigma_{o1} = \{a, b, d\}$, $\Sigma_{o2} = \{a, c, e\}$ and $\Sigma_{f1} = \{\sigma_1\}$. The trace $s = ab\sigma_1 c(de)^*$ is state/failure-ambiguous since (1) $P_1(s) = P_1(a\sigma b(de)^*)$ but $P(s) \neq P(a\sigma b(de)^*)$, (2) $P_2(s) = P_2(a\sigma c(de)^*)$ but $P(s) \neq P(a\sigma c(de)^*)$, (3) $a\sigma b(de)^*$ and $a\sigma c(de)^*$ share the same failure properties (here both traces do not have failure events), (4) s has a failure of type F_1 while $a\sigma b(de)^*$ and $a\sigma c(de)^*$ exhibit only normal behavior. The diagnosers G_{d1} and G_{d2} are shown in Figure 5.2. If the system executes the trace $ab\sigma_1 c(de)^*$ then after the occurrence of the event d, $R_1 = (5F1, 8N)$ and $R_4 = (4F1, 7N, 5F1, 8N)$. Therefore, by applying action* **DR1** *(since d is observed by site 1 only) the coordinator's diagnostic information C is equal to $R_1 \cap R_4 = q_1 \cap UR_2(q_2) = (5F1, 8N)$. Along the*

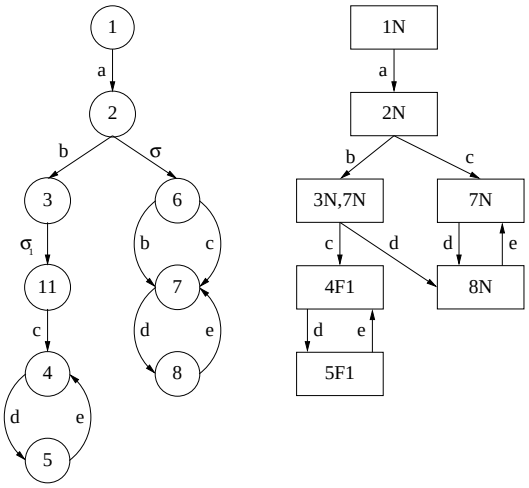

Figure 5.1: The system (left) and centralized diagnoser for Example 5.3.1

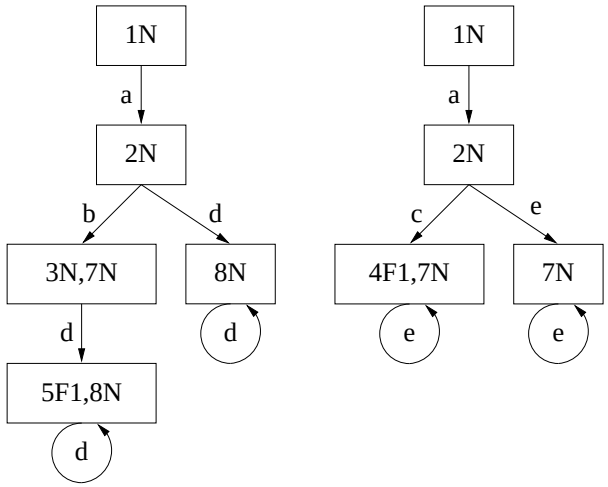

Figure 5.2: The diagnosers G_{d1} (left) and G_{d2} for Example 5.3.1

same trace, after the occurrence of event d the centralized diagnoser state is $q = 5F1$ *which is neither equal to* C *nor has the same diagnostic properties as* C*. Thus the presence of state/failure-ambiguous traces shows that the results of Propositions 5.3.1 and 5.3.2 are not, in general, true.*

The next example shows that the "state-ambiguous trace" ("failure-ambiguous trace") condition is not necessary for Proposition 5.3.1 (respectively 5.3.2) to be true.

Example 5.3.2 *Consider the system shown in Figure 5.3. The set of events is* $\Sigma = \{a, b, c, d, e, \sigma, \sigma_1\}$, $\Sigma_{uo} = \{\sigma, \sigma_1\}$, $\Sigma_{o1} = \{a, b, d\}$, $\Sigma_{o2} = \{a, c, e\}$ *and* $\Sigma_{f1} = \{\sigma_1\}$. *The trace* $s = ab\sigma_1 c(de)^*$ *is state/failure-ambiguous since (1)* $P_1(s) = P_1(a\sigma b(de)^*)$ *but*

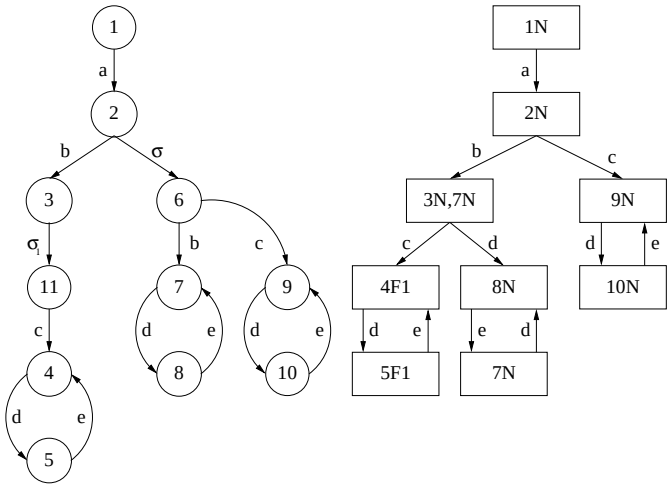

Figure 5.3: The system (left) and centralized diagnoser for Example 5.3.2

$P(s) \neq P(a\sigma b(de)^*)$, (2) $P_2(s) = P_2(a\sigma c(de)^*)$ but $P(s) \neq P(a\sigma c(de)^*)$, (3) $a\sigma b(de)^*$ and $a\sigma c(de)^*$ share the same failure properties (here both traces do not have failure events), (4) s has a failure of type F_1 while $a\sigma b(de)^*$ and$a\sigma c(de)^*$ exhibit only normal behavior. The diagnosers G_{d1} and G_{d2} are shown in Figure 5.4. If the system executes the trace $ab\sigma_1 c(de)^*$

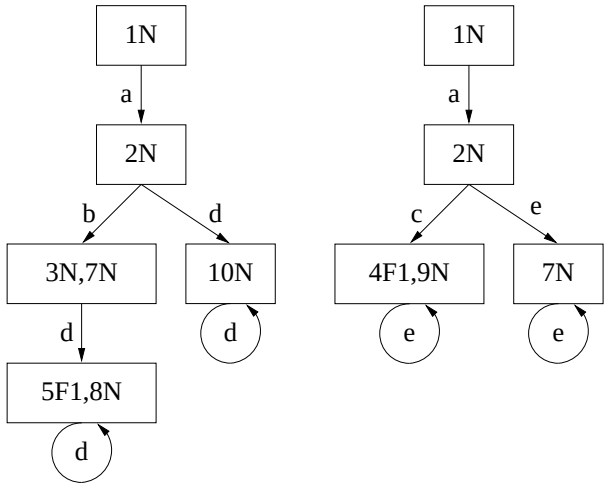

Figure 5.4: The diagnosers G_{d1} (left) and G_{d2} for Example 5.3.2

then after the occurrence of the event d, $R_1 = (5F1, 8N)$ and $R_4 = (4F1, 9N, 5F1, 10N)$. Therefore, by applying action **DR1** *(since d is observed by site 1 only) the coordinator's diagnostic information C is equal to $R_1 \cap R_4 = q_1 \cap UR_2(q_2) = 5F1$. Along the same trace, after the occurrence of event d the centralized diagnoser state is $q = 5F1 = q_1 \cap UR_2(q_2)$ and by definition it has the same diagnostic properties as C. Consequently, the condition on state/failure-ambiguous traces in Propositions 5.3.1 and 5.3.2 is only sufficient for these*

propositions to be true.

5.4 Necessary and Sufficient Conditions for Diagnosability

The results of Section 5.3 imply that if there are no failure-ambiguous traces in $L(G)$, the necessary and sufficient conditions for diagnosability with respect to Protocol 2 can be stated with respect to the centralized diagnoser as follows.

Theorem 5.4.1 *If there are no failure-ambiguous traces in $L(G)$, a live and prefix-closed language L is diagnosable with respect to Protocol 2, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f if and only if the diagnoser G_d does not have F_i-indeterminate cycles for all failure types F_i.*

Proof of Theorem 5.4.1 Sufficiency($\Leftarrow$). Suppose G_d does not have F_i-indeterminate cycles. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, and t is long enough, i.e., $\| t \| > n$, where n can be arbitrarily large. Then, by assumption, st', $t' \in \overline{t}$ does not lead to an F_i-indeterminate cycle in G_d. Consequently, G_d will enter an F_i-certain state within a finite number of steps, say n_i'. This implies by Theorem 5.3.2 that the coordinator's diagnostic information C will be F_i-certain within a finite number of steps equal to n_i'. Therefore, $L(G)$ is diagnosable with respect to Protocol 2.
Necessity($\Rightarrow$). We prove the contrapositive. Assume that G_d does enter an F_i-indeterminate cycle. This implies that the coordinator's diagnostic information C, which carries the same diagnostic properties as the centralized diagnoser state by Theorem 5.3.2, will remain F_i-uncertain for an arbitrarily long number of steps. Hence, L is not diagnosable with respect to Protocol 2. **Q.E.D.**

Having checked that the diagnoser G_d is capable of diagnosing all failure types, the next step could be to find a test to check whether failure-ambiguous traces exist or not. We bypass this and instead provide a direct test to verify whether Protocol 2 performs as well as the centralized diagnoser. If the test fails we conclude that there are failure-ambiguous traces in the language. Otherwise, we know that Protocol 2 diagnoses all failure types that are diagnosed by the centralized diagnoser; however we cannot claim that there are no failure-ambiguous traces in the language. In order to present this test, we introduce G_{test2}. G_{test2} is the FSM

$$G_{test2} = (Q_g, \Sigma_o, \delta_g, g_0) \tag{5.15}$$

where the state space $Q_g \subseteq Q_{d1} \times Q_{d2} \times Q_d \times (Q_{d1} \cap Q_{d2})$, Σ_o is the set of events of G_{test2} and $g_0 = (q_{01}; q_{02}; q_0; q_{01} \cap q_{02}) = (\{(x_0, \{N\})\}; \{(x_0, \{N\})\}; \{(x_0, \{N\})\}; \{(x_0, \{N\})\})$ is

the initial state of G_{test2}. A state p in G_{test2} is denoted by $(g_1; g_2; g; g_c)$ where $g_1 \in Q_{d1}$, $g_2 \in Q_{d2}$, $g \in Q_d$ and $g_c \in Q_{d1} \cap Q_{d2}$. The partial transition function δ_g is defined as follows:

$$\delta_g((g_1; g_2; g; g_c), \sigma) = \begin{cases} (\delta_{d1}(g_1, \sigma); \delta_{d2}(g_2, \sigma); \delta_d(g, \sigma); \delta_{d1}(g_1, \sigma) \cap \delta_{d2}(g_2, \sigma)) \\ \text{if } \sigma \in \Sigma_{o1} \cap \Sigma_{o2}, \\ \\ (\delta_{d1}(g_1, \sigma); g_2; \delta_d(g, \sigma); \delta_{d1}(g_1, \sigma) \cap UR_2(g_2)) \\ \text{if } \sigma \in \Sigma_{o1} \backslash \Sigma_{o2}, \\ \\ (g_1; \delta_{d2}(g_2, \sigma); \delta_d(g, \sigma); UR_1(g_1) \cap \delta_{d2}(g_2, \sigma)) \\ \text{if } \sigma \in \Sigma_{o2} \backslash \Sigma_{o1}. \end{cases}$$

The ideas and objective behind introducing this machine are the following:

1. Synchronize the operation of the diagnosers G_{d1} and G_{d2} to be able to generate C, the coordinator's diagnostic information.

2. Make sure that the synchronized behavior is indeed a legal observed behavior of the system.

It can be verified that $L(G_{test2}) = L(G_d)$. Therefore, G_{test2} observes the system behavior as would G_d after the execution of a given trace s, provides information about the states of the diagnosers G_{d1} and G_{d2} after the execution of s, and computes the coordinator's diagnostic information, C. We are interested in detecting simultaneous occurrences of F_i-indeterminate cycles in both G_{d1} and G_{d2} because by comparing the coordinator's diagnostic information with the centralized diagnoser's information along these cycles we may be able to determine whether Protocol 2 performs as well as a centralized diagnoser. To precisely describe how we do so, we need the following definitions.

Definition 5.4.1 *A state $p = (g_1; g_2; g; g_c)$ in G_{test2} is said to be ambiguous if g is not normal and g, g_c do not have the same failure properties. We say that g, g_c have the same failure properties when the following is true: g is F_i-certain (uncertain) if and only if g_c is F_i-certain (uncertain), for all failure types F_i.*

Definition 5.4.2 *A cycle in G_{test2} is said to be F_i-indeterminate if the corresponding cycles in G_{d1} and G_{d2} are both F_i-indeterminate.*

Definition 5.4.3 *A cycle in G_{test2} is said to be F_i-ambiguous if it is F_i-indeterminate* **and** *all its states are ambiguous.*

The notion of an ambiguous cycle is helpful in providing conditions under which the language is diagnosable as the following theorem indicates.

Theorem 5.4.2 *(i) A live and prefix-closed language L is diagnosable with respect to Protocol 2, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f if for all failure types F_i, the following conditions are true: (a) G_{test2} does not have F_i-ambiguous cycles* **and** *(b) G_d does not have F_i-indeterminate cycles.*
(ii) A live and prefix-closed language L modeled by the FSM G is diagnosable with respect to Protocol 2, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f only if for all failure types F_i the following conditions are true: (a) G_{test2} corresponding to G does not have any F_i-ambiguous cycles **and** *(b) G_d does not have any F_i-indeterminate cycles.*

Proof of Theorem 5.4.2 (i) Suppose G_{test2} does not have any F_i-ambiguous cycles and G_d does not have F_i-indeterminate cycles for all failure types F_i. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$ and t is long enough, i.e., $\| t \| > n$, where n can be arbitrarily large. By the construction of G_{test2} and the assumption that G_{test2} does not have any ambiguous cycles, st cannot lead to F_i-indeterminate cycles in both G_{d1} and G_{d2} along which the coordinator's diagnostic information does not have the same failure properties as the centralized diagnoser state. Therefore C cannot remain F_i-uncertain indefinitely since G_d does not have F_i-indeterminate cycles; hence $L(G)$ is diagnosable with respect to Protocol 2.
(ii) Assume $L(G)$ is diagnosable with respect to Protocol 2 and is modeled by the FSM G. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, t is long enough, i.e., $\| t \| > n$, where n can be arbitrarily large, and s leads to F_i-indeterminate cycles in both G_{d1} and G_{d2}. By assumption we know that the coordinator's diagnostic information C will be F_i-certain in a finite number of steps along st. When C is F_i-certain then the corresponding centralized diagnoser state is F_i-certain because of the rule generating C (cf. Section 5.2.3 and (5.2), (5.9) and (5.10)). Therefore, following st the centralized diagnoser does not enter an F_i-indeterminate cycle and G_{test2} does not enter an F_i-ambiguous cycle (because the coordinator's diagnostic information and the centralized diagnoser will both be F_i-certain along st within a finite number of steps). Since st is arbitrary, then for all failure types F_i, G_{test2} corresponding to G does not have any F_i-ambiguous cycles and G_d does not have any F_i-indeterminate cycles. **Q.E.D.**

We note here that the necessary and sufficient conditions in Theorem 5.4.2 are almost identical, except that the necessary conditions depend on the machine representation of the language. We clarify this subtlety as follows. Protocol 2 is a state-based (as opposed to

language-based) diagnostic scheme (cf. the definition of the decision rule in Section 5.2.3). Hence, we expect the necessary and sufficient conditions to depend on the FSM G that models the language L. In contrast, the necessary and sufficient conditions for diagnosability in the centralized case (cf. Theorem 2.5.1) do not depend on the FSM G, since the arguments used in the proof of Theorem 2 in [40] are all trace-based arguments, i.e., the diagnostic scheme is indeed language-based. In the case of Protocol 2, if G_{test2} does not have F_i-ambiguous cycles, and G_d does not have F_i-indeterminate cycles for all failure types F_i, then Protocol 2 diagnoses all failure types, and we need not worry about G in the statement of the sufficient conditions. However, if G_{test2} has F_i-ambiguous cycles, then we cannot assert that Protocol 2 cannot diagnose a failure of type F_i since there may exist another FSM G' of L, such that Protocol 2 diagnoses all failure types when it is combined with G'. Consequently, the necessary conditions indeed depend on G. The following examples illustrate the above discussion.

Example 5.4.1 *Consider the system discussed in Example 5.3.1 and shown in Figure 5.1. G_{test2} is shown in Figure 5.5. The cycle labeled A in the Figure is an F_1-ambiguous cycle:*

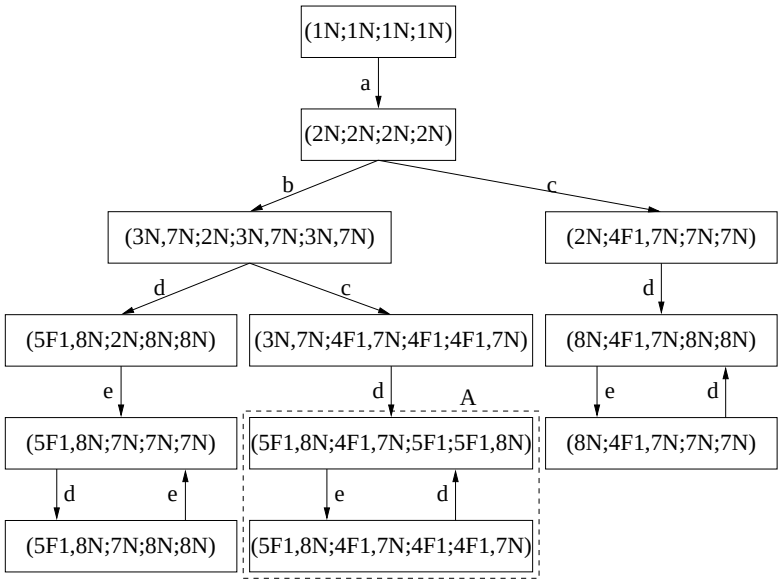

Figure 5.5: G_{test2} for Example 5.4.1

it can be verified that the cycles $(5F1, 8N)$ and $(4F1, 7N)$ are F_1-indeterminate cycles in G_{d1} and G_{d2}, respectively, and both cycles are ambiguous since the state of the centralized diagnoser is F_1-certain ($5F1$ or $4F1$) and the coordinator's diagnostic information is F_1-uncertain ($(5F1, 8N)$ or $(4F1, 7N)$). Therefore the system with the FSM representation shown in Figure 5.1 is not diagnosable under Protocol 2.

Example 5.4.2 *Consider the system discussed in Example 5.3.2 and shown in Figure 5.3. The systems of Examples 5.3.1 and 5.3.2 provide two different FSM representations of the*

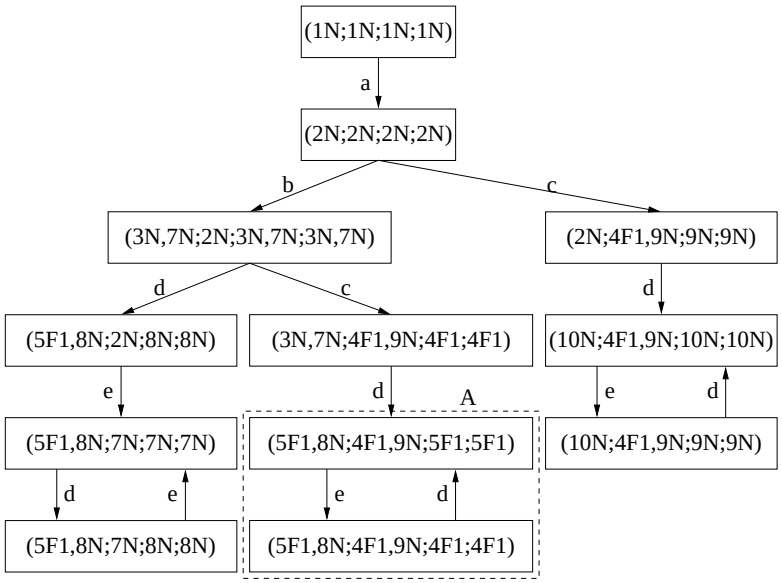

Figure 5.6: G_{test2} for Example 5.4.2

same language. G_{test2} for the system of Example 5.3.2 is shown in Figure 5.6. The cycle labeled A in the figure is an F_1-indeterminate cycle: it can be verified that the cycles $(5F1, 8N)$ and $(4F1, 9N)$ are F_1-indeterminate cycles in G_{d1} and G_{d2}, respectively; however the states $(5F1, 8N; 4F1, 9N; 5F1; 5F1)$ and $(5F1, 8N; 4F1, 9N; 4F1; 4F1)$ share the same failure properties (F_1-certain), therefore the cycle is not F_1-ambiguous. Hence by Theorem 5.4.2 the system with the FSM representation shown in Figure 5.3 is diagnosable under Protocol 2. Note here that although the language exhibits failure-ambiguous traces (cf. Example 5.3.2), the system is diagnosable since we are able to verify using G_{test2} that Protocol 2 performs as well as the centralized diagnoser based on G shown in Figure 5.3. In Example 5.4.1 we showed that the same language represented by another G (cf. Figure 5.1) is not diagnosable under Protocol 2, hence proving that the system model G should be taken into consideration in the necessity proof of Theorem 5.4.2.

5.5 Discussion

We first note that the performance of Protocol 2 is inferior to that of Protocol 1 because only under the restrictions on the system structure discussed in Section 5.3, Protocol 2 performs as well as the centralized diagnoser. However, the communication, memory, and processing requirements for Protocol 2 are less than those of Protocol 1. Indeed, the gen-

eration of diagnostic information at the local sites, in case of Protocol 2, requires less time and memory than the generation of diagnostic information in Protocol 1. Furthermore less information per observed event has to be communicated to the coordinator under Protocol 2. The coordinator's information update and decision rules for Protocol 2 are simpler to implement than the ones used for Protocol 1.

The partitioning of observable events is crucial in deciding whether a language is diagnosable under Protocol 2 or not. In fact, failure-ambiguous traces, which may force Protocol 2 not to perform as well as the centralized diagnoser, are defined with respect to the projections P_1 and P_2 (cf. Definition 5.1.2). Therefore, changing the partitions P_1 and P_2 may eliminate the presence of failure-ambiguous traces, hence allowing Protocol 2 to perform as well as the centralized diagnoser. The following example illustrates this idea.

Example 5.5.1 *Consider the system of Example 2.4.1 shown in Figure 2.1 with* $\Sigma = \{a, b, c, d, e, \sigma\}$, $\Sigma_{uo} = \{\sigma\}$, $\Sigma_{f1} = \{\sigma\}$, $\Sigma_{o1} = \{a, c, d, e\}$, *and* $\Sigma_{o2} = \{b, d, e\}$. *In Example 5.1.2 we showed that the trace* $bac\sigma(de)^*$ *is failure-ambiguous, hence Protocol 2 may not perform as well as the centralized diagnoser with the partitions* P_1 *and* P_2. *Indeed, this can be seen by checking* G_{test2} *for the above system (cf. Figure 5.7): the*

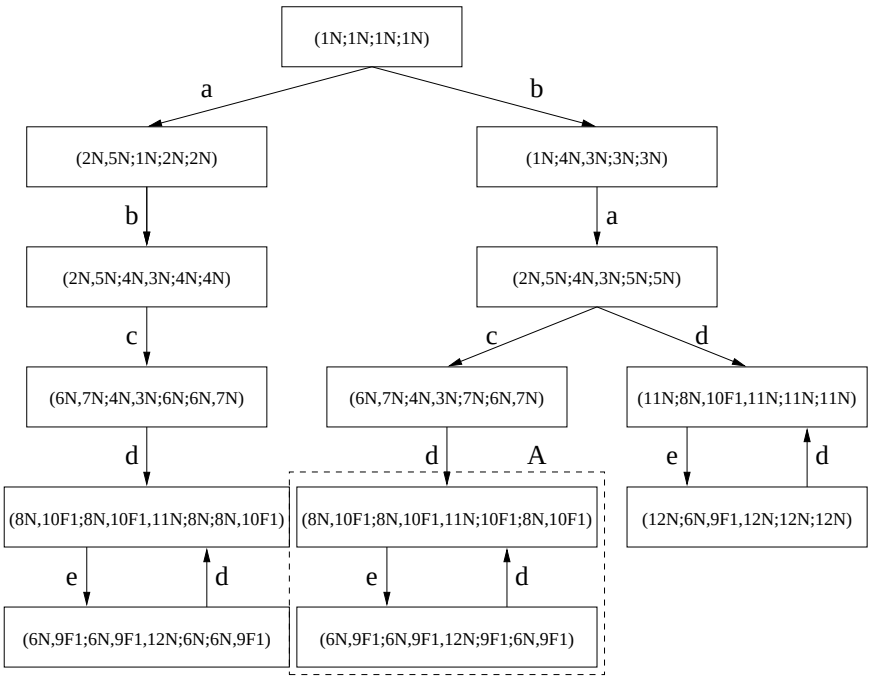

Figure 5.7: G_{test2} for Example 5.5.1

cycle labeled A in the figure is F_1*-ambiguous because the cycles* $\{(8N, 10F1), (6N, 9F1)\}$ *and* $\{(8N, 10F1, 11N), (6N, 9F1, 12N)\}$ *are* F_1*-indeterminate in* G_{d1} *and* G_{d2}, *respectively,*

and one could verify that the corresponding centralized diagnoser state is F_1-certain ($10F1$ and $9F1$) while the coordinator's diagnostic information is F_1-uncertain (($8N, 10F1$) and ($6N, 9F1$)). If we consider now a new partitioning of the observable events where Σ_{o1} is as before and $\Sigma_{o2} = \{a, b, d, e\}$, then Protocol 2 performs as well as the centralized diagnoser: G_{test2} for the system with the new partitioning of observable events (i.e. the new set of projections) is shown in Figure 5.8; clearly there are no F_i-ambiguous cycles in G_{test2}.

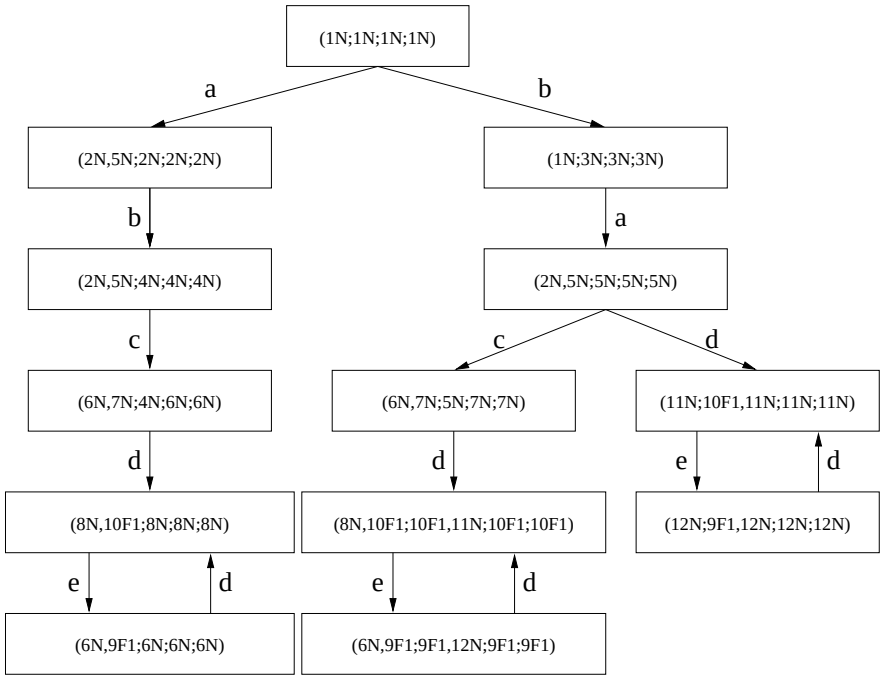

Figure 5.8: G_{test2} with the new set of projections for Example 5.5.1

Example 5.4.2 shows that although the local diagnosers remain F_i-uncertain indefinitely after the occurrence of a failure of type F_i, Protocol 2 detects and isolates the failure. The process of identifying the failures is achieved through the decision rule implemented at the coordinator. The decision rule identifies failures based on the diagnostic information C. The information update rule (cf. Section 5.2.3) processes the messages received from the local sites to update the diagnostic information C. In some practical cases, the coordinator cannot wait to process the data to make decisions, rather it should declare the occurrence of failures based on the raw information it receives from the local sites. The next chapter introduces a protocol that addresses this issue.

It is worth noting that, if there are no state-ambiguous traces in $L(G)$, the information state available at the coordinator's site is sufficient, at any instant of time, to obtain the estimate of the centralized diagnoser. This is true by the same argument we used following

Theorem 5.3.1 to justify the possibility of skipping on communications while maintaining the same diagnostic performance. Therefore, if there are no state-ambiguous traces in $L(G)$, Protocol 2 reconstructs the state of the the centralized diagnoser *even when there is no continuous communication between the local sites and the coordinator.*

The feature of Protocol 2 discussed above is also present in decentralized estimation of linear Gaussian systems, [44] and [54]. However, in [44] and [54] there are no restrictions on the structure of the linear Gaussian system. On the other hand, the estimation problems in [44] and [54] are linear, whereas Protocol 2 deals with a nonlinear estimation problem. Decentralized nonlinear estimation problems for stochastic systems have been investigated in [8]. The coordinated decentralized estimation protocol proposed in [8] achieves the same performance as the optimal centralized estimator under no restrictions on the system structure, but requires continuous communication between the local sites and the coordinator and assumes that the coordinator has knowledge of the system structure. The above comparison shows that, under the assumptions on the system structure discussed in Section 5.1, Protocol 2 has some remarkable features.

As is the case with Protocol 1, the performance of Protocol 2 depends on the assumption that messages are received at the coordinator in the order they are sent locally and globally. We have already argued in Section 3.2 that the local order assumption is justified in real systems, however it may be too strong an assumption to claim that global order is preserved. We discuss the relaxation of Assumption **A5** and the resulting modification in the case of Protocol 2 in the next section.

5.6 Protocol 2D: Relaxation of Assumption A5

In this section, we modify Protocol 2 to account for communication delays. We will refer to the modified protocol as **Protocol 2D** (where 2D stands for the fact that is a modification of Protocol **2** that allows for communication **D**elays). A key feature of Protocol 2 is the following: if the system has no failure ambiguous traces, and if the communicated messages are received in the correct order by the coordinator, then the coordinator can identify exactly the same failure types as the centralized diagnoser even when communication between the local sites and the coordinator is not continuous. The above feature is the key to understanding the performance of Protocol 2D. When communication between the local sites and the coordinator is continuous, but messages are received out of order by the coordinator, the coordinator, as in the case of Protocol 1D, should consider all possible orders according to which the messages may have been sent. For each possible order, it uses the information update rule specified by Protocol 2. A failure is diagnosed only when

according to all possible sorted out orders it is certain that the failure has occurred. We determine conditions under which Protocol 2D performs as well as the centralized diagnoser.

The following sub-sections define Protocol 2D, namely the diagnostic information generated at the local sites, the communication rules used by the local sites, and the decision rule used by the coordinator to infer the occurrence of failures. The memory requirements for Protocol 2D and two modifications of the protocol, in the case of commonly observed events and non-continuous communication between the local sites and the coordinator, are also discussed.

5.6.1 Diagnostic information at local sites

The same diagnostic information defined for Protocol 2 is used. Diagnosers are implemented at local sites. The diagnostic information available at each site is provided by the state of the diagnoser, and is refined by the unobservable reach as specified in Definition 5.2.1.

5.6.2 Communication rules

The communication rules of Protocol 2 are also used for Protocol 2D. After the agent at either site observes an event, it communicates to the coordinator the corresponding state of its diagnoser, its unobservable reach, and the status bit.

5.6.3 Decision rule

The coordinator's decision rule is composed of three steps: (1) store all incoming messages at the coordinator site, and sort out all possible orders in which these messages were generated by the local sites; (2) apply the information update rule, as defined in Section 5.2.3, to each and every possible order, and retain all orders that result in a non-empty update (intersection); (3) compare the failure properties of all surviving updates from step (2), and declare the occurrence of a failure when all these updates are certain of the occurrence of the failure. These steps are discussed in the following three sub-sections. The following analysis assumes that there are no events commonly observed by sites 1 and 2. The case where there are events that are commonly observed at both sites is briefly discussed in Section 5.6.6.

(1) Sorting out possible orders

Messages are stored and sorted out in the same way as Protocol 1D.

(2) Application of the update rule

For every possible order that is sorted out the coordinator has five registers, ($R1$, $R2$, $R3$, $R4$, SB), besides the register C that holds the corresponding diagnostic information, if any. The functions of the five registers are the same as the one used to store incoming messages in the case of Protocol 2. The coordinator applies the information update rule of Table 5.1 to each one of the orders that are sorted out. For every sorted out order that survives the update rule, the coordinator keeps the last update along with the corresponding states and unobservable reaches and status bits. This is possible by definition of the update rule that only requires information from the last update to generate the new one (cf. Table 5.1). If a specific order of messages results in an empty intersection, the coordinator rejects that as an impossible order. The information is stored by the coordinator until the next instant when new sorted out orders are extracted (as a continuation of the already existing ones). Then the same procedure as above is applied to these new orders and the new updates replace the old ones that are discarded. The implementation of the sorting procedure and update rule follows closely those of Protocol 1D.

(3) Diagnosing failures

If all the information updates that result from applying the procedure described in Sections 5.6.3-(1) and 5.6.3-(2) are certain that a failure F has occurred, then the coordinator declares that F has occurred and broadcasts this information to the failure recovery module. Otherwise, a diagnostic decision is postponed.

5.6.4 Diagnostic properties of Protocol 2D

The decision rule discussed in Section 5.6.3 has the following two features: (1) In comparison with Protocol 2, the memory requirements at the coordinator's site may increase considerably because the coordinator has to store all the information updates, discussed in Section 5.6.3-(2), and the corresponding diagnoser states and unobservable reaches. However, the amount of memory required remains finite since the models used are finite-state automata (cf. Section 5.6.5). (2) The coordinator identifies a failure only when that failure is identified by all surviving information updates. Therefore, we expect that, in general, the performance of Protocol 2D will not be the same as that of Protocol 2 where global order is preserved. Interestingly enough, we prove next that the absence of failure-ambiguous traces is a condition sufficient to ensure that Protocol 2D performs as well the centralized diagnoser. The next theorem summarizes the main result concerning the diagnostic performance of Protocol 2D under the "one-step out of order" assumption (cf. Section 3.2).

Theorem 5.6.1 *Under the assumption that messages are received by the coordinator at most "one-step out of order", Protocol 2D eventually identifies all failure types that are detected by the centralized diagnoser if there are no failure-ambiguous traces (with respect to all failure types).*

Proof of Theorem 5.6.1 Consider that the system is executing the trace $s_0u_1au_2bu_3c := su_3c$ where $a \in \Sigma_{o1}$, $b \in \Sigma_{o2}$, $c \in \Sigma_{o1}$, $F_i \in s_0$, and $u_1, u_2, u_3 \in \Sigma_{uo}^*$. Denote by x, y, and z the messages generated by the occurrence of events a, b, and c, respectively. Without loss of generality assume that the messages are received in the following order: x, y, then z. This corresponds to the case presented in the third row of Table 4.3 (the fourth row is the symmetric case of the third row while by inspection one realizes that the cases presented in rows one and two are respectively sub-cases of those presented in rows three and four). Also assume that s is arbitrarily long, i.e., s cannot be a failure-ambiguous trace. Let q_{11} and q_{12} be the states of the diagnoser G_{d1} after the execution of the the events a and c, respectively, and q_2 be the state of the diagnoser G_{d2} after the execution of the event b. The correct order in which these messages are sent is xyz, and from Table 4.3 and the sorting procedure presented in Section 5.6.3-(1) we have that the coordinator considers the following orders: xy, yx, and xzy (see Remark 2 in Section 4.5.3-(1)). Denote by C_1, C_2, and C_3 the coordinator state resulting from applying the update rule to the orders xy, yx, and xzy, respectively.

For the first order xy we have from Table 5.1

$$C_1 = UR_1(q_{11}) \cap q_2. \tag{5.16}$$

Since this is the correct decision then C_1 is not empty. Moreover since s is not failure-ambiguous, C_1 is F_i-certain.

For the second order, namely yx, the coordinator considers results in

$$C_2 = UR_2(q_2) \cap q_{11}. \tag{5.17}$$

Assume that C_2 is not empty. Then, there exist at least two traces s_1' (ending with the event a in Σ_{o1}) and s_2' (ending with an event in Σ_o) sharing the same failure properties such that

$$P_1(s_1') = P_1(s) \tag{5.18}$$

$$P_2(s_2') = P_2(s) \tag{5.19}$$

$$\delta(x_0, s_1') = \delta(x_0, s_2'). \tag{5.20}$$

Since by assumption there are no failure-ambiguous traces, then either $P(s) = P(s'_1)$ (negation of condition 1 in Definition 5.1.2), or $P(s) = P(s'_2)$ (negation of condition 2 in Definition 5.1.2), or s, s'_1, s'_2 share the same failure properties (negation of conditions $3a$, $3b$ in Definition 5.1.2), or combinations of these facts are true. Condition 4 in Definition 5.1.2 cannot be violated because then C_2 is empty. $P(s) = P(s'_1)$ is impossible, as it contradicts the fact that s'_1 ends with an event in Σ_{o1}. If $P(s) = P(s'_2)$ then necessarily s, s'_1, s'_2 share the same failure properties since the language is diagnosable with respect to Σ_o and the failure partition, and hence C_2 is F_i-certain. If s, s'_1, s'_2 share the same failure properties then we have that C_2 is F_i-certain.

For the third order, namely xzy, the coordinator considers results in

$$C_3 = UR_1(q_{12}) \cap q_2. \tag{5.21}$$

Assume that C_3 is not empty. Therefore, there exist at least two traces t'_1 (ending with an event in Σ_o) and t'_2 (ending with the event b in Σ_{o2}) sharing the same failure properties such that

$$P_1(t'_1) = P_1(su_3c), \tag{5.22}$$

$$P_2(t'_2) = P_2(s), \tag{5.23}$$

$$\delta(x_0, t'_1) = \delta(x_0, t'_2). \tag{5.24}$$

But $su_3c \in L(G)$ and $P_2(su_3c) = P_2(s)$; consequently,

$$P_2(t'_2) = P_2(su_3c). \tag{5.25}$$

Since there are no failure-ambiguous traces, then either $P(su_3c) = P(t'_1)$, or $P(su_3c) = P(t'_2)$, or su_3c, t'_1, t'_2 share the same failure properties, or combinations of these facts are true. If $P(su_3c) = P(t'_1)$ then necessarily su_3c, t'_1, t'_2 share the same failure properties since the language is diagnosable with respect to Σ_o and the failure partition, and hence C_3 is F_i-certain. $P(su_3c) = P(t'_2)$ is impossible as it contradicts the fact that t'_2 ends with an event in Σ_{o2}. If su_3c, t'_1, t'_2 share the same failure properties then we have that C_3 is F_i-certain.

From the above analysis we conclude that C_1, C_2, and C_3 are all F_i-certain. Consequently, the failure type F_i is diagnosed by Protocol 2D. Since F_i was arbitrarily chosen, Protocol 2D can diagnose any failure type under the assumption that there are no failure-ambiguous traces. Consequently Protocol 2D performs as well as the centralized diagnoser. **Q.E.D.**

Note here that a trace that is not failure-ambiguous may contain prefixes that are failure-ambiguous. Hence, the above proof holds true once the system has executed enough events

to overcome the failure-ambiguous sub-traces. Therefore, Protocol 2D identifies, under the condition of this theorem, the same failures as the centralized diagnoser but with higher delay.

We conjecture that the result of Theorem 5.6.1 still holds even when messages are received at the coordinator at most "n-steps out of order". We expect that the delays associated with diagnostic decisions and the memory requirements at the coordinator's site will increase with n.

5.6.5 Memory issues in Protocol 2D

In contrast to Protocol 1D where all non-empty updates (intersections) result in a state of the centralized extended diagnoser, possible orders that are considered by the sorting procedure used by step (1) of the decision rule of Protocol 2D may result in a non-empty update (intersection) that is different from any state of the centralized diagnoser. Therefore, one may end up by having a considerably large number of possible updates to be saved. To avoid that problem, we suggest to use some side information at the coordinator's site. All possible states of the coordinator, when global order is preserved, following any possible legal behavior of the system are computed off-line and stored at the coordinator site. This side information helps reducing the number of possible updates to be stored: in case an update results in a state that does not belong to the side information this update is rejected. By doing so we are only tracking legal behavior that is exhibited under Protocol 2, and most importantly we only require finite memory to hold updates since these updates are bounded by the order of the state space of G_{test2}, since G_{test2} holds the states of the coordinator when global order is preserved. As explained in the case of Protocol 1D, the suggested bound is a loose one, nevertheless it proves that the implementation of the protocol requires finite memory.

5.6.6 Procedure using common events

The procedure presented in Section 5.6.3 assumed that all messages communicated to the coordinator regard only events that are observed by either site 1 or site 2, but not both. In fact, if there are events that are commonly observed by both sites then these events can be used as a synchronization mechanism. Since local order is preserved, the coordinator knows how to order the messages that are due to commonly observed events. As a result of the information update rule at the coordinator's site, messages generated by commonly observed events need to contain only the states of the local diagnosers. Therefore, under the assumption that commonly observed events are executed frequently along all traces,

the protocol can be modified as follows: local sites use the diagnosers to generate their diagnostic information; they communicate their diagnosers' states to the coordinator only after the occurrence of commonly observed events; the decision rule of the coordinator is to apply the information update rule as specified in Table 5.1. Denote by **Protocol 2D-C** (where **C** stands for commonly observed events) the above specified protocol. Then, we have the following result.

Theorem 5.6.2 *If there no failure-ambiguous traces (with respect to all failure types), Protocol 2D-C eventually identifies the same failures as the centralized diagnoser.*

Proof of Theorem 5.6.2 Since local order is preserved, messages sent by distinct sites regarding the same commonly observed event can be easily matched. By inspecting the information update rule presented in Table 5.1, one realizes that the update following a commonly observed event only requires information received by the messages and no past information is needed. Then according to Theorem 5.3.2, if there are no failure-ambiguous traces then the update rule eventually results in an F_i-certain coordinator state following the execution by the system of an event that belongs to the failure type F_i. **Q.E.D.**

We note that the word eventually is also used in the statement of the theorem for the same reasons explained after Theorem 5.6.1.

This variation of the protocol saves on communication, processing power, and memory storage (the same memory storage as in the case of Protocol 2 is needed) at the expense of delaying the diagnostic decision in the case where the frequency of occurrence of commonly observed events is low. Note here that the "one-step out of order" assumption is not needed as long as communication delays are bounded.

5.6.7 Polling procedure

Another approach is to advise the coordinator to poll the sites requesting each site to communicate its current state and unobservable reach plus the status bit specifying whether the event that led to the current state is observed by the other site or not. Denote this protocol by **Protocol 2D-P** (where **P** stands for polling). Under the assumption that the system does not execute a new observable event (in Σ_o) between the instant the polling message is generated and the instants it is received by the sites, we have the following result.

Theorem 5.6.3 *Under the assumption that the system does not execute a new observable event ($\in \Sigma_o$) between the instant the polling message is generated and the instants it is received by the sites, Protocol 2D-P eventually identifies the same failures as the centralized diagnoser if there are no failure-ambiguous traces (with respect to all failure types).*

Proof of Theorem 5.6.3 Consider that the system is executing the trace su_1au_2b where where $a, b \in \Sigma_o$, $F_i \in s$, and $u_1, u_2 \in \Sigma_{uo}^*$. Assume the coordinator polls the sites right after the event b was observed. By assumption the polling message is received at the two sites before the system executes a new observable event. Also assume that s is arbitrarily long, i.e., s cannot be a failure-ambiguous trace. If either a, b, or both are common events it is easy to verify that the coordinator can figure out the correct order in which events were executed by the system and consequently applies the correct information update rule. Since s is not failure-ambiguous Theorem 5.3.2 implies that the failure F_i is diagnosed. The only case where the coordinator cannot figure out the correct order in which the events were executed by the system is when $a \in \Sigma_{o1}$ and $b \in \Sigma_{o2}$, or $a \in \Sigma_{o2}$ and $b \in \Sigma_{o1}$. Without loss of generality assume that $a \in \Sigma_{o1}$ and $b \in \Sigma_{o2}$. Denote by x and y the messages generated by the occurrence of events a and b, respectively. Without loss of generality assume that the messages are received in the following order: x then y. Let q_1 and q_2 be the states of the diagnosers G_{d1} and G_{d2} after the execution of the events a and b, respectively. The coordinator considers the following orders: xy, yx. Denote by C_1 and C_2 the coordinator state resulting from applying the update rule to the orders xy and yx, respectively.

For the first order xy we have from Table 5.1

$$C_1 = UR_1(q_1) \cap q_2. \tag{5.26}$$

Since this is the correct decision then C_1 is not empty. Moreover since s is not failure-ambiguous, C_1 is F_i-certain.

For the second order, namely yx, the coordinator considers results in

$$C_2 = UR_2(q_2) \cap q_1. \tag{5.27}$$

Assume that C_2 is not empty. Then, there exist at least two traces s_1' (ending with the event a in Σ_{o1}) and s_2' (ending with an event in Σ_o) sharing the same failure properties such that

$$P_1(s_1') = P_1(s) \tag{5.28}$$

$$P_2(s_2') = P_2(s) \tag{5.29}$$

$$\delta(x_0, s_1') = \delta(x_0, s_2'). \tag{5.30}$$

Since by assumption there are no failure-ambiguous traces then either $P(s) = P(s_1')$ (negation of condition 1 in Definition 5.1.2), or $P(s) = P(s_2')$ (negation of condition 2 in Definition 5.1.2), or s, s_1', s_2' share the same failure properties (negation of conditions $3a$, $3b$ in Definition 5.1.2), or combinations of these facts are true. Condition 4 in Definition 5.1.2

cannot be violated because then C_2 is empty. $P(s) = P(s'_1)$ is impossible, as it contradicts the fact that s'_1 ends with an event in Σ_{o1}. If $P(s) = P(s'_2)$ then necessarily s, s'_1, s'_2 share the same failure properties since the language is diagnosable with respect to Σ_o and the failure partition and hence C_2 is F_i-certain. If s, s'_1, s'_2 share the same failure properties then we have that C_2 is F_i-certain.

From the above analysis we conclude that C_1 and C_2 are both F_i-certain. Consequently, the failure type F_i is diagnosed by Protocol 2D-P. Since F_i was arbitrarily chosen, Protocol 2D-P can diagnose any failure type under the assumption that there are no failure-ambiguous traces. Consequently Protocol 2D-P performs as well as the centralized diagnoser. **Q.E.D.**

We note again that a trace that is not failure-ambiguous may contain prefixes that are failure-ambiguous. Hence, the above-presented proof holds true once the system has executed enough events to overcome the failure-ambiguous sub-traces. Therefore, Protocol 2D-P identifies, under the condition of this theorem, the same failures as the centralized diagnoser but with higher delay.

Protocol 2D-P saves on communication and processing power and memory storage (a comparable memory storage to the case of Protocol 2 is needed). It avoids as well the delaying of diagnostic decisions when the frequency of occurrence of commonly observed events is low. Also, the "one-step out of order" assumption is not needed as long as communication delays are bounded.

5.6.8 Remarks

The key features of Protocol 2D are: (1) it achieves the same performance as the centralized diagnoser when there are no failure-ambiguous traces. That is, the absence of global ordering of the messages received at the coordinator's site does not degrade this aspect of the protocol's performance (compared to Protocol 2). (2) The delay of its diagnostic decision is higher than the delay of the centralized diagnoser. (3) The memory required at the coordinator's site to implement the protocol is larger than that required in Protocol 2. The first feature of Protocol 2D is a bit surprising, whereas its last two features are not unexpected.

As is the case with Protocol 2, communication may be interrupted and savings on communication can be achieved using Protocol 2D. Two modifications of Protocol 2D, namely Protocols 2D-C and 2D-P, result in communication and memory savings while maintaining the same diagnostic performance. However, the diagnostic delays are further increased in Protocol 2D-C, whereas an additional assumption (that of Theorem 5.6.3), that

may be unrealistic in some cases, is required by Protocol 2D-P.

CHAPTER 6

PROTOCOL 3

6.1 Objective and Assumptions

We present a protocol where the coordinator declares the occurrence of failures based on the raw information it receives from the local sites. We call this protocol Protocol 3.

To analyze the performance of Protocol 3 we introduce the notion of fully-ambiguous traces with respect to the projections P_1 and P_2.

Definition 6.1.1 *A trace $s \in L(G)$ is said to be fully-ambiguous with respect to the projections P_1 and P_2 and the failure type F_i if there exist two traces, s' and s'' in $L(G)$ such that s' and s'' are arbitrarily long, not necessarily distinct, and the following are true:*

1. $P_1(s) = P_1(s')$ *but* $P(s) \neq P(s')$.

2. $P_2(s) = P_2(s'')$ *but* $P(s) \neq P(s'')$.

3a. $F_i \in s$ *but* $F_i \notin s'$.

3b. $F_i \in s$ *but* $F_i \notin s''$.

This definition says that the traces s, s' and s, s'' can be distinguished under the projection P; however s and s' are not distinguishable under P_1 while s and s'' are not distinguishable under P_2. Furthermore, there is a difference in failure properties between s, s' and s, s'': if F_i belongs to s then it does not belong to neither s' nor s'' or vice-versa. Note here that a failure-ambiguous trace is a fully-ambiguous trace; however the reverse is not true since there are no restrictions on the failure properties of s' and s'' in the definition of a fully-ambiguous trace. Thereafter when we refer to a trace as being fully-ambiguous, the projections P_1 and P_2 and the failure type F_i will be understood from the context. The concept of "fully-ambiguous traces" is illustrated by the following example.

Example 6.1.1 *Consider the system discussed in Example 5.3.2 and shown in Figure 5.3. The set of events is* $\Sigma = \{a, b, c, d, e, \sigma, \sigma_1\}$, $\Sigma_{uo} = \{\sigma, \sigma_1\}$, $\Sigma_{f1} = \{\sigma_1\}$, $\Sigma_{o1} = \{a, b, d\}$ *and* $\Sigma_{o2} = \{a, c, e\}$. *The trace* $s = ab\sigma_1 c(de)^*$ *is fully-ambiguous (with respect to failure type* F_1*) since (1)* $P_1(s) = P_1(a\sigma b(de)^*) = ab(d)^*$ *but* $P(s) = abc(de)^* \neq ab(de)^* = P(a\sigma b(de)^*)$, *(2)* $P_2(s) = P_2(a\sigma c(de)^*) = ac(e)^*$ *but* $P(s) = abc(de)^* \neq ac(de)^* = P(a\sigma c(de)^*)$ *and (3)* $F_1 \in s$, *but* $F_1 \notin a\sigma b(de)^*$ *and* $F_1 \notin a\sigma c(de)^*$. *Note here that we could have concluded that* s *is fully-ambiguous since we showed in Example 5.3.2 that* s *is failure-ambiguous.*

We will study the performance of Protocol 3 under Assumptions **A1** - **A8** (cf. Section 3.1) and the following additional assumption.

A10 There are no fully-ambiguous traces (with respect to all failure types) in $L(G)$.

6.2 Specification of the Protocol

In this section we specify Protocol 3 in detail. We begin by discussing the diagnostic information at the local sites.

6.2.1 Diagnostic information at local sites

We implement diagnosers at the local sites. Therefore, the state of the diagnoser, after the occurrence of an observable event, is the diagnostic information based on which the site is supposed to infer the occurrence of failures.

6.2.2 Communication rules

Since the coordinator is supposed to declare the occurrence of failures based on the raw information provided by the local sites, the information communicated from the local sites should be as simple and concise as possible. Consequently we define the following communication rules:

- **[CRi]**, $i = 1, 2$: After the agent at site i observes an event $\sigma \in \Sigma_{oi}$ that leads to an F_i-certain state in the diagnoser G_{di}, it communicates the label F_i to the coordinator, meaning that a failure of type F_i has occurred.

6.2.3 Decision rule

The coordinator declares that a failure of type F_i has occurred once its diagnostic information C is F_i-certain. Since local sites communicate the failure labels detected by their corresponding diagnoser, once the coordinator receives a message, either from site 1

or site 2, containing the information F_i, it declares the occurrence of a failure of type F_i and broadcasts the information to the failure recovery module.

The coordinator's diagnostic information C is updated each time a message is received at its site. The rule update is $C = C \cup \{F_i\}$ where F_i is the last incoming message. For the sake of consistency with the definition of diagnosability (cf. Definition 3.3.3), when reading that C is F_i-certain, we understand that F_i belongs to C.

6.3 Diagnostic Properties of Protocol 3

We first present a lemma that relates the existence of F_i-indeterminate cycles in the local diagnoser to fully-ambiguous traces (with respect to failure type F_i).

Lemma 6.3.1 *Consider a trace s in $L(G) \cap \Psi(\Sigma_{fi})$. The trace s is fully-ambiguous with respect to failure type F_i if and only if s leads to F_i-indeterminate cycles in G_{d1} and G_{d2}.*

Proof of Lemma 6.3.1 Sufficiency ($\Leftarrow$). Assume that s leads to F_i-indeterminate cycles in G_{d1} and G_{d2}. By definition (cf. Definition 2.4.4), there exist traces s' and s'' in $L(G)$ arbitrarily long such that the following is true:

$$P_1(s) = P_1(s'), F_i \in s \text{ but } F_i \notin s', \tag{6.1}$$

and

$$P_2(s) = P_2(s''), F_i \in s \text{ but } F_i \notin s''. \tag{6.2}$$

(6.1) and (6.2) imply that s is fully-ambiguous with respect to F_i.
Necessity ($\Rightarrow$). The fact that s is fully ambiguous with respect to F_i implies that there exist two traces s' and s'' such that

$$P_1(s) = P_1(s'), F_i \in s \text{ but } F_i \notin s', \tag{6.3}$$

and

$$P_2(s) = P_2(s''), F_i \in s \text{ but } F_i \notin s''. \tag{6.4}$$

(6.3) and (6.4) imply that there exists an F_i-indeterminate cycle in G_{d1} and G_{d2} simultaneously, i.e., following the trace s. **Q.E.D.**

The above lemma establishes the fact that following the execution of a trace which is not fully-ambiguous, the local diagnosers G_{d1} and G_{d2} cannot loop "simultaneously" in indeterminate cycles. By "simultaneously" we understand following the execution of a trace in the system.

The diagnostic properties of Protocol 3 are summarized by the following theorem.

Theorem 6.3.1 *Protocol 3 performs as well as the centralized diagnoser if and only if there are no fully-ambiguous traces (with respect to all failure types) in the language.*

Proof of Theorem 6.3.1 Sufficiency($\Leftarrow$). Assume that there are no fully-ambiguous traces in the language. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, t is long enough, i.e., $\| t \| > n$, where n can be arbitrarily large and st leads to an F_i-certain state in the centralized diagnoser. By the implication of Lemma 6.3.1 s cannot lead to F_i-indeterminate cycles in both G_{d1} and G_{d2}. Therefore the state of G_{d1} or that of G_{d2} will be F_i-certain in a finite number of steps, which implies that either G_{d1} or G_{d2} will diagnose the failure. Since s is arbitrary, all failures diagnosed by the centralized diagnoser are diagnosed under Protocol 3. Therefore, Protocol 3 performs as well as the centralized diagnoser.
Necessity($\Rightarrow$). We prove the contrapositive. Assume that there are fully-ambiguous traces in the language. Consider a fully-ambiguous trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, t is long enough, i.e., $\| t \| > n$, where n can be arbitrarily large and st leads to a F_i-certain state in the centralized diagnoser. By Lemma 6.3.1, we have that s leads to F_i-indeterminate cycles in both G_{d1} and G_{d2}. Therefore the state of G_{d1} and that of G_{d2} will be F_i-uncertain indefinitely (along st), which implies that $F_i \notin C$. Therefore, L is not diagnosable under Protocol 3, hence Protocol 3 does not perform as well as the centralized diagnoser. **Q.E.D.**

6.4 Necessary and Sufficient Conditions for Diagnosability

The results of Section 6.3 imply that if there are no fully-ambiguous traces in $L(G)$, the necessary and sufficient conditions for diagnosability with respect to Protocol 3 can be stated with respect to the centralized diagnoser as follows:

Theorem 6.4.1 *If there are no fully-ambiguous traces in $L(G)$, a live and prefix-closed language L is diagnosable with respect to Protocol 3, the set of projections P_1, P_2 and the failure partition Π_f on Σ_f if and only if the diagnoser G_d does not have F_i-indeterminate cycles for all failure types F_i.*

Proof of Theorem 6.4.1 The proof is a direct consequence of Theorem 6.3.1. **Q.E.D.**

Having checked that the diagnoser G_d is capable of diagnosing all failure types, the next step could be to find a test to check whether fully-ambiguous traces exist or not. In order to present this test, we introduce G_{test3}. G_{test3} of the system G is defined as follows:

$$G_{test3} = G_{d1} \parallel G_{d2} \parallel G_d$$

where G_{d1}, G_{d2}, and G_d are as defined earlier. The following format for a state p of G_{test3} is adopted: $p = (q_1; q_2; q)$, where q_1, q_2, and q belong to Q_{d1}, Q_{d2}, and Q_d, respectively.

The ideas and objectives behind introducing this machine are the following:

1. Synchronize the operation of the diagnosers G_{d1} and G_{d2}. This necessitates their parallel composition.

2. Make sure that the synchronized behavior is indeed a legal observed behavior of the system. This necessitates the composition of $G_{d1} \parallel G_{d2}$ with the system diagnoser G_d.

Now we have that

$$L(G_{test3}) \stackrel{\triangle}{=} P_1^{-1}[L(G_{d1})] \cap P_2^{-1}[L(G_{d2})] \cap L(G_d), \tag{6.5}$$

where

$$P_i^{-1}(y) = \{s \in (\Sigma_{o1} \cup \Sigma_{o2})^* : P_i(s) = y\}. \tag{6.6}$$

Therefore we have

$$L(G_{test3}) = L(G_d). \tag{6.7}$$

Therefore, G_{test3} observes the system behavior as would G_d after the execution of a given trace s, and also provides information about the states of the diagnosers G_{d1} and G_{d2} after the execution of s. Hence, using G_{test3}, it is possible to identify the states of the diagnosers G_{d1} and G_{d2} after the system has executed a trace in the language. We are interested in detecting simultaneous occurrences of F_i-indeterminate cycles in both G_{d1} and G_{d2}, since testing these cycles may identify whether there are fully-ambiguous traces in the language or not (cf. Lemma 6.3.1). We first need the following technical definition.

Definition 6.4.1 *A cycle in G_{test3} is said to be F_i-indeterminate if the corresponding cycles in G_{d1} and G_{d2} are both F_i-indeterminate.*

Such notion is helpful in providing a test to check whether the system is diagnosable under Protocol 3 or not. The following result introduces the test.

Theorem 6.4.2 *A live and prefix-closed language L is diagnosable with respect to Protocol 3, the set of projections P_1,P_2, and the failure partition Π_f on Σ_f if and only if for all failure types F_i the following is true: G_d does not have F_i-indeterminate cycles* **and** *G_{test3} does not have F_i-indeterminate cycles.*

Proof of Theorem 6.4.2 Sufficiency($\Leftarrow$). G_d and G_{test3} do not have F_i-indeterminate cycles for all failure types F_i. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, and t is long enough, i.e., $\| t \| > n$, where n can be arbitrarily large. By the construction of G_{test3} and the assumptions above, s cannot lead to F_i-indeterminate cycles in both G_{d1} and G_{d2}. Therefore G_{d1} and G_{d2} do not simultaneously loop in F_i-indeterminate cycles, which implies that either G_{d1} or G_{d2} will diagnose the failure. Since s is arbitrary, L is diagnosable under Protocol 3.
Necessity($\Rightarrow$). Assume $L(G)$ is diagnosable under Protocol 3. Consider a trace $st \in L(G)$ such that $s \in \Psi(\Sigma_{fi})$, and t is long enough, i.e., $\| t \| > n$, where n can be arbitrarily large. By definition this implies that C is F_i-certain ($F_i \in C$ more precisely) in a finite number of steps along st. Therefore, the state of one of the diagnosers G_{d1} or G_{d2} is F_i-certain (in a finite number of steps) along st by the specification of Protocol 3. Hence, G_{test3} does not enter an F_i-indeterminate cycle. It is easy to verify that if the state of one of the diagnosers G_{d1} or G_{d2} is F_i-certain then so is the state of G_d. Therefore G_d enters an F_i-certain state in a finite number of steps along st, i.e., G_d does not have F_i-indeterminate cycles. Since s is arbitrary, then for all failure types F_i, G_d and G_{test3} do not have F_i-indeterminate cycles. **Q.E.D.**

6.5 Discussion

We first note that the performance of Protocol 3 is inferior to that of Protocol 2 in the sense that there are more restrictions on the system structure for Protocol 3 to perform as well as the centralized diagnoser than there is for Protocol 2. This can be seen by the following example.

Example 6.5.1 *Consider the system discussed in Example 5.3.2 and shown in Figure 5.3. In Example 5.4.2 we showed that with the FSM representation shown in Figure 5.3, Protocol 2 performs as well as the centralized diagnoser. G_{test3} for the system of Example 5.3.2 is shown in Figure 6.1. The cycle labeled A in the Figure is a F_1-indeterminate cycle: it can be verified that the cycles $(5F1, 8N)$ and $(4F1, 9N)$ are F_1-indeterminate cycles in G_{d1} and G_{d2}, respectively; hence by Theorem 6.4.2 the system is not diagnosable under Protocol 3.*

Protocol 2 performs better than Protocol 3 because failure-ambiguous traces are also fully-ambiguous, however the reverse is not always true. However, the communication, processing, and memory requirements for Protocol 3 are less than those of Protocol 2. Indeed, the diagnostic information generated at the local sites, in case of Protocol 3, is a subset of the coordinator's diagnostic information in the case of Protocol 2. In addition,

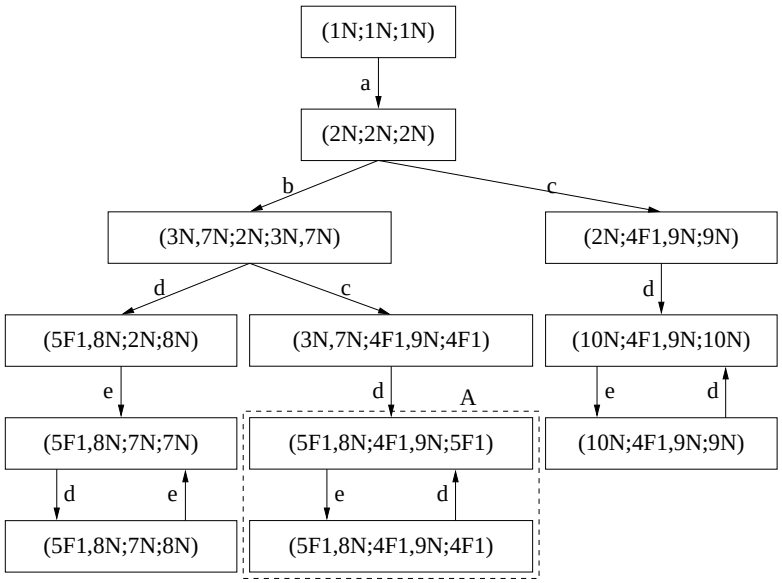

Figure 6.1: G_{test3} for Example 6.5.1

communication is significantly reduced in Protocol 3, and the decision rule does not involve any processing at the coordinator.

As is the case with Protocol 2, the partitioning of observable events is crucial in deciding whether a language is diagnosable under Protocol 3 or not. Changing the partitions P_1 and P_2 may eliminate the presence of fully-ambiguous traces.

CHAPTER 7

REFLECTIONS ON THE PROTOCOLS AND THE ARCHITECTURE

In this chapter we discuss some fundamental issues related to the coordinated decentralized diagnostic protocols presented in this thesis.

7.1 "Performance vs. Complexity" trade-off

As is the case with all coordinated decentralized architectures, the issue of "performance vs. complexity" should be addressed. By "performance vs. complexity" we understand a discussion of the qualitative properties of the protocols in terms of how well they perform and what their memory and processing power requirements are at the local sites and at the coordinator site. The presentation of Protocols 1 - 3 in Chapters 4 - 6 was done in a manner that highlights this trade-off. Protocol 1 performs as well as the centralized diagnoser irrespective of the system structure and the partitioning of observable events. Protocol 2 achieves the same task while constraining the system structure, and Protocol 3 adds additional constraints to those of Protocol 2 to achieve the diagnostic performance of the centralized diagnoser. Note here that the performance of Protocols 2 and 3 depend on the partitioning of observable events. Therefore, the diagnostic performance of the protocols improves from 3 to 2 to 1. However, the memory and processing requirements for implementing the protocols increases from 3 to 2 to 1: from a considerably low amount of processing and communication and a simple decision rule for Protocol 3 to more processing and communication, and a more involved decision rule for Protocol 2, to more processing and communication and an even more complicated decision rule for Protocol 1. Tables 7.1 and 7.2 summarize the above comparison.

In light of the above comparison, given a system with a set of available sensors at its two sites (in other words the projections P_1 and P_2) one should first test whether Protocol 3 is capable of detecting and isolating all possible failures and that the diagnostic delay is

Table 7.1: Comparison of the three protocols

Protocol	Constraints on system structure	Partitioning of observable events
1	None	irrelevant
2	No failure-ambiguous traces	relevant
3	No fully-ambiguous traces	relevant

Table 7.2: Comparison of the three protocols (continued)

Protocol	Diagnostic information	Information communicated	Communication instances	Decision rule
1	Extended diagnoser state, extended unobservable reach	Extended diagnoser state, extended unobservable reach, status bit	upon each observable event occurrence	two intersections
2	Diagnoser state, unobservable reach	Diagnoser state, unobservable reach, status bit	upon the coordinator's request*	one intersection
3	Diagnoser state	Failure label	upon the diagnoser state being failure certain	no intersection

*: Assuming the coordinator polls and not as presented in Section 5.2.2.

acceptable. If that is the case, Protocol 3 should be implemented to diagnose the system on-line. In case Protocol 3 fails to diagnose all possible failures of interest, Protocol 2 should be tested to check whether it achieves the intended diagnostic task within required delay constraints. If the result of the latter test is positive, then Protocol 2 should be implemented; otherwise one is forced to implement Protocol 1. The latter protocol is guaranteed to achieve the same diagnostic performance as a centralized diagnoser would. Figure 7.1 pictorially summarizes the performance of the protocols and is another way of redrawing Figure 3.2.

The suggested approach on which protocol to implement will be illustrated in Chapter 8 where we present the diagnosis of transmitter and receiver faults in a wireless local area network.

We note here that a similar comparison can be made as far as Protocols 1D and 2D are concerned. By noting that ambiguous traces (cf. Definition 4.5.1) are failure-ambiguous traces (cf. Definition 5.1.2) but the reverse is not true, it is easy to conclude that the diagnostic performance of Protocol 1D is superior to that of Protocol 2D. This comes at

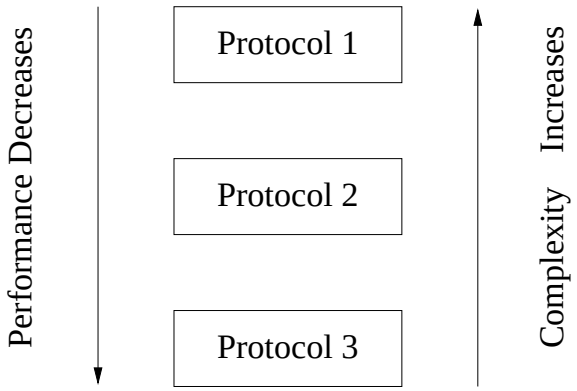

Figure 7.1: Comparison of Protocols

the expense of communicating more information per message and doing more processing at the coordinator site. The latter fact is easy to verify since Protocols 1D and 2D use the same diagnostic information, communication rules, and decision rule (more precisely they implement the same rule more than once for the same diagnostic decision) as Protocols 1 and 2, respectively. For the case where global order of the received messages in not preserved, comparison tables such as Tables 7.1 and 7.2 are not provided; but we note that Protocol 1D will be affected by the partitioning of observable events (cf. Definition 4.5.1 of ambiguous traces).

7.2 Relaxation of Assumption A5

The approach we used to account for communication delays in the case of Protocol 1D and Protocol 2D requires a considerable amount of additional memory and processing at the coordinator site. In contrast, if time-stamps were available, Protocols 1 and 2 would work without any modifications. However, in that case local clocks would need to be synchronized and this requires additional processing and memory storage at the local sites. Therefore, a tradeoff exists between these two mechanisms to handle communication delays, and the type of application usually determines the mechanism to use. For instance, in low energy mobile networks [51] and telecommunication networks [17, 30], it may be infeasible to use time-stamps.

We only considered the "one-step out of order" assumption for reasons of simplicity and compactness. In fact, the same results hold in the case of "n-step out of order" messages. The general sorting rule in such a case is to wait for the arrival of $n+2$ messages and then figure out all possible orders. By doing so the number of these orders increases considerably, but most importantly it remains finite. Afterwards, we apply the update rule for the first

two messages in every possible order and we follow the same procedure as in the case of the "one-step out of order" assumption. Consequently, the diagnostic decision is further delayed and memory requirements increase drastically; nevertheless the results presented in the thesis under the "one-step out of order" assumption hold for the general case of "n-step out of order" messages.

7.3 Extension to m Sites

In our discussion, we have considered the generic case of two sites. The results obtained in this paper can be extended to the case of m sites in a straightforward manner. We will not state and prove these results, but we will give a logical explanation why they should stand.

In Protocol 3, the extension is quite obvious: in the case of m diagnosers, one of the m diagnosers should be able to identify the occurrence (the type) of any failure that occurs. In order to verify such a condition, we extend G_{test3} to include the synchronization of all the local diagnosers, in addition to the centralized one, and check for the existence of cycles where the corresponding cycles of all local diagnosers are indeterminate.

The case of Protocol 2 is more involved. The communication rules are the analogues of **CR1** and **CR2** for all local sites, and the decision rule can be extended in the following way: in case the event is common to all, intersect all the states of the diagnosers, otherwise intersect the states of all diagnosers who saw the event with the unobservable reaches of all diagnosers who did not see the event. A mechanism to identify who saw the last event and who did not should be implemented at the coordinator site. Using such a rule, the test to check diagnosability is to identify the existence of ambiguous cycles in a machine that represents the extension of G_{test2} to m sites. Such a test will provide the correct result since the intersection operator is associative.

In the case of Protocol 1, we adopt the same extension of the communication rules as for Protocol 2. The decision rule can be extended in the same way, the only difference being that $\cap_e^i$ is used instead of the regular intersection. Moreover, after applying the intersection to the states and unobservable reaches, the operator $\cap_c$ is applied to identify the admissible behavior. Since $\cap_e^i$ is associative by definition, the decision rule is reconstructing the centralized diagnoser state at the coordinator site; hence, there exists a test, namely the test on the centralized diagnoser, to check whether Protocol 1 diagnoses the system or not. Note here that in such a case, a mechanism should be implemented at the coordinator site to take care of who saw what and when.

Finally, the extension in the case of Protocols 1D and 2D is done in a similar fashion to

that of Protocols 1 and 2, respectively. This is due to the fact that Protocols 1D and 2D implement the same decision rule as Protocols 1 and 2, respectively (although it may be used more than once for the same diagnostic decision), and the fact that the "one-step out of order" assumption is made independently of the number of available sites.

7.4 General Thoughts on the Approach

Two salient features of the decentralized protocols presented in this paper are: (1) the diagnostic algorithms employed at the local sites are based on centralized diagnosis procedures, that is, diagnostic information at each local site is generated by solving a centralized diagnosis problem at the site as in [37] or [40]; (2) the objective is to determine realizations of the architecture of Section 3.1 that perform as well as the centralized diagnostic scheme.

The use of diagnosers or extended diagnosers at the local sites is guided by the powerful results of [37, 40] on the centralized diagnosis problem. Even though the use of centralized diagnosis procedures provides a reasonable strategy for generating diagnostic information at the local sites, it is far from clear that such procedures always present the best alternative. For example, it may be possible to achieve the same performance as Protocol 1 if at the local sites we use diagnostic algorithms other than extended diagnosers. Such algorithms could take into account the fact that diagnostic information is generated at more than one sites, they could utilize the knowledge that is common [2, 52] to all local sites, and could lead to protocols that perform as well as a centralized diagnoser with less requirements on communication, data processing and memory than Protocol 1. The discovery of such protocols is a very challenging open problem with far reaching implications.

As pointed out in Section 3.4, the performance of the centralized architecture proposed in [40] provides an upper bound on the performance achievable (i.e., the failure events diagnosable) by any realization of the coordinated decentralized architecture of Section 3.1. Achieving the performance of the centralized diagnostic scheme proposed in [40] with a coordinated decentralized architecture requires a certain amount of resources for data storage and data processing both at the local sites and the coordinator, as well as a certain amount of bandwidth for data communication, specified by Protocol 1. When the amount of resources for data storage, processing and communication is limited, the challenge is to determine the best performance achievable under the resource constraint. The presence of such a constraint gives rise to problems that are significantly more difficult than the ones studied in this thesis, because currently there are no tight upper bounds on the diagnostic performance, achievable under a resource constraint, to guide the design of decentralized coordinated protocols. The determination of such tight bounds is an important fundamental

problem with significant practical implications.

CHAPTER 8

EXAMPLE: DIAGNOSING A WIRELESS LOCAL AREA NETWORK

We apply our methodology to a wireless local area network (LAN) used by an automated platoon of vehicles. The objective is to diagnose faults in transmitters and receivers of the radios used by the vehicles composing the platoon.

8.1 Description of the System

The concept of automated platoons of vehicles on intelligent highways has been extensively investigated as of late (cf. `www.path.berkeley.edu`). The objective is to improve the traffic flow capacity of the highways while ensuring safety and reliability. The vehicles on the highways will travel in the form of platoons of multiple vehicles under automated lateral guidance. Each platoon will be composed of a lead vehicle and many followers. In order to implement any control law in such a complex computer controlled system, vehicles in the platoon need to exchange information among each other, and/or communicate to base stations located alongside the highways. For instance, the commanded acceleration of a follower vehicle is a function of variables that cannot be sensed, such as the acceleration and velocity of the lead vehicle [45]. Therefore, platoon operation is supported by a wireless LAN [18] that transmits information from one vehicle to the other at the sampling rate (typically 20ms) of the longitudinal control system. Also available is a wide area network that can be used for occasional messaging between vehicles and/or base stations [43]. We are interested in applying our failure diagnosis methodology to diagnose faults in transmitters and receivers of the radios used by the vehicles composing the platoon.

The physical structure of the platoon LAN could be cast in the suggested coordinated decentralized architecture of Figure 3.1 as follows: each vehicle in the platoon, including the lead vehicle, constitutes a site. Note that we argued in Section 7.3 that our methodology extends in a straightforward manner to m sites. The coordinator could be either one of the

vehicles or the base station with whom the platoon is communicating. Each site has its own radio that can transmit or receive information. The objective is to diagnose transmitters and receivers faults. A centralized diagnostic scheme is difficult to realize and undesirable in a decentralized system that is required to be highly fault tolerant [43]. Hence, a decentralized diagnostic scheme is required. Since the problem is that of diagnosis of a communication network, the wide area network would be used to facilitate communication between the local sites and the coordinator.

We use a simplified model of the LAN operation that captures the features essential for failure diagnosis. For the full model the reader is referred to [16]. The normal operation of the LAN network (the middle cycle in the graph) for a three-vehicle platoon, a leader and two followers, is shown in Figure 8.1, along with two faulty modes of operation: lead vehicle

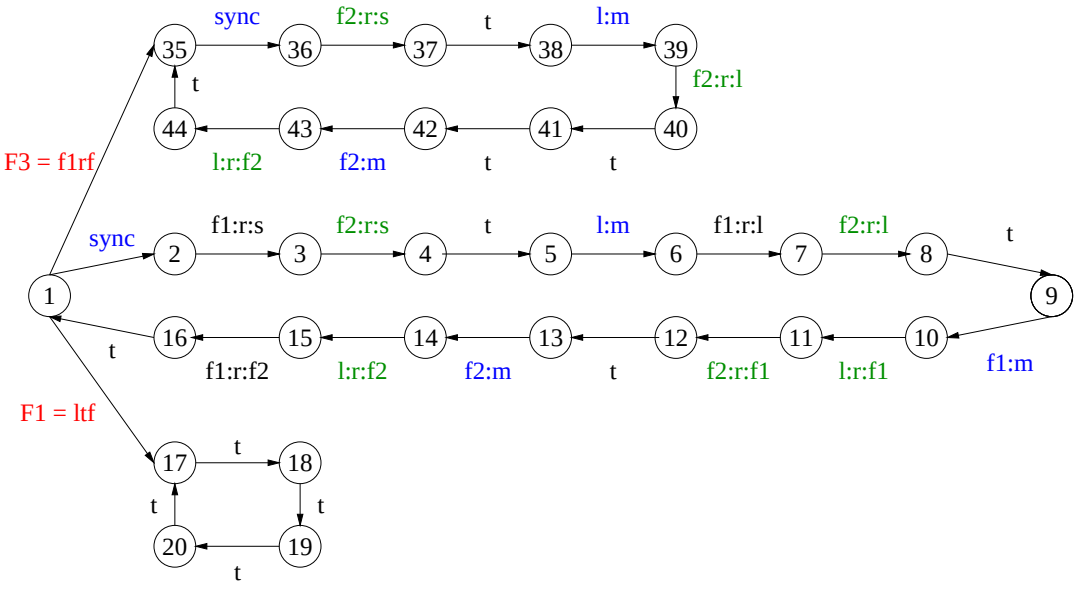

Figure 8.1: Partial LAN model for a three vehicle platoon

transmitter faulty denoted by $F1$ and first follower receiver faulty denoted by $F3$. The complete transition function of the FSM G modeling the LAN is presented in Appendix A.1. The set of events is $\Sigma = \{sync,\ f1:r:s,\ f2:r:s,\ t,\ l:m,\ f1:r:l,\ f2:r:l,\ f1:m,\ l:r:f1,\ f2:r:f1,\ f2:m,\ l:r:f2,\ f1:r:f2,\ lrf,\ ltf,\ f1rf,\ f1tf,\ f2rf,\ f2tf\}$, where the meaning of each event is explained in Table 8.1. The vehicles share a LAN through a TDMA scheme [18]. The longitudinal control sampling interval is divided into slots of fixed duration with one slot being allocated to each vehicle in the platoon. At the beginning of the sampling interval, the lead vehicle transmits a synchronization pulse to which all the platoon vehicles set their clocks. The first slot is then used by the lead vehicle to transmit control information. The second slot is used by the first follower vehicle of the platoon, and

Table 8.1: Definition of events for the LAN model

$sync$	is the transmission of the synchronization pulse by the lead vehicle
$f1:r:s$	is the reception of the synchronization pulse by the first follower vehicle
$f2:r:s$	is the reception of the synchronization pulse by the second follower vehicle
t	is the expiration of a time slot
$l:m$	is the transmission of the lead vehicle control message
$f1:r:l$	is the reception of the lead vehicle control message by the first follower
$f2:r:l$	is the reception of the lead vehicle control message by the second follower
$f1:m$	is the transmission of the control message by the first follower
$l:r:f1$	is the reception of the first follower control message by the lead vehicle
$f2:r:f1$	is the reception of the first follower control message by the second follower vehicle
$f2:m$	is the transmission of the control message by the second follower
$l:r:f2$	is the reception of the second follower control message by the lead vehicle
$f1:r:f2$	is the reception of the second follower control message by the first follower vehicle
lrf	is the lead vehicle receiver fault
ltf	is the lead vehicle transmitter fault
$f1rf$	is the first follower receiver fault
$f1tf$	is the first follower transmitter fault
$f2rf$	is the second follower receiver fault
$f2tf$	is the second follower transmitter fault

so on until the last vehicle transmits. Afterwards, the lead vehicle synchronizes clocks again and the process repeats. It is assumed that each radio can have transmitter or receiver faults (cf. Table 8.1). If a receiver of a vehicle is faulty, then the vehicle does not transmit at all since it does not receive the synchronization pulse. The lead vehicle transmits even if it does not receive anything because it controls the synchronization. Our modeling approach only considers faults that occur at the beginning of the sampling interval. In fact, this avoids the need to consider the possibility of occurrence of faults at any instant of time during one TDMA cycle at the expense of delaying the diagnostic decision: if a fault occurs during a given TDMA cycle, one needs to wait at least until the next cycle before being able to detect its occurrence. Also, single faults are considered due to the fact that multiple faults occur with low probability in this system [43].

Vehicles only observe the messages received by their radios. Therefore, the set of un-

observable events is defined as follows: $\Sigma_{uo} = \{sync,\ l:m,\ f1:m,\ f2:m,\ lrf,\ ltf,\ f1rf,\ f1tf,\ f2rf,\ f2tf\}$. The failure partition is the following: $\Sigma_{f1} = \{ltf\}$, $\Sigma_{f2} = \{lrf\}$, $\Sigma_{f3} = \{f1rf\}$, $\Sigma_{f4} = \{f1tf\}$, $\Sigma_{f5} = \{f2tf\}$, $\Sigma_{f6} = \{f2rf\}$.

Under the assumption that a centralized diagnoser is in charge of failure detection and isolation, it is easy to check that the LAN network is diagnosable in the framework of [40] since the diagnoser G_d (cf. Appendix A.2) does not have any indeterminate cycles for all failure types. However, as stated earlier, it is undesirable to implement a centralized diagnostic scheme for the LAN network.

In [43], the authors diagnose the wireless LAN as follows: they implement a diagnoser at each site (vehicle). Diagnosers may exchange messages regarding the occurrence of their observable events. Each diagnoser is in charge of diagnosing faults associated with its site using its own diagnostic information and messages received from other diagnosers. The authors characterize the class of distributed systems where there exists no inter-diagnoser messaging scheme that can replicate the information available to a centralized diagnoser. They provide necessary and sufficient conditions for diagnosability, however, they do not provide a test to check these conditions.

In contrast to the approach of [43], our failure diagnosis methodology for decentralized systems forces the diagnosers (implemented at the local sites) to communicate some processed version of their diagnostic information to the coordinator. The coordinator is in charge of detecting and isolating failures. We discuss the application of our methodology to diagnose the wireless LAN in the next section.

8.2 Applying the Diagnostic Methodology to the Wireless LAN

Since each vehicle is a site then each site only observes the messages its corresponding vehicle receives in addition to the event t since sites by assumption have local clocks. Denote by sites 1, 2, and 3, the lead, first follower, and second follower vehicles, respectively. Therefore the set of observable events for the three sites are the following: $\Sigma_{o1} = \{t,\ l:r:f1,\ l:r:f2\}$, $\Sigma_{o2} = \{t,\ f1:r:l,\ f1:r:f2\}$, $\Sigma_{o3} = \{t,\ f2:r:l,\ f2:r:f1\}$. The diagnosers to be implemented at the sites are presented in Appendix A, and it can be verified that all these diagnosers do have failure indeterminate cycles, that is, no one diagnoser can diagnose the system independently of the other diagnosers' information.

As suggested in Section 7.1, the first step is to check whether Protocol 3 is capable of diagnosing all failure types. For that purpose we build the FSM G_{test3}, which is presented in Appendix A.6. Note that $G_{test3} = G_{d1} \parallel G_{d2} \parallel G_{d3} \parallel G_d$ for this example, and a state of

G_{test3} following the observation of a trace $s \in L(G)$ is of the form $(p_1; p_2; p_3; p)$ where p_1, p_2, p_3, and p are the states of diagnosers G_{d1}, G_{d2}, G_{d3}, and G_d, respectively. To determine whether Protocol 3 performs as well as the centralized diagnoser, we check for the existence of indeterminate cycles in G_{test3} (cf. Theorem 6.4.2). In fact, such cycles do exist as depicted in Figure 8.2. Following the execution of the events ltf, t, t, t, G_{test3} reaches the first state of

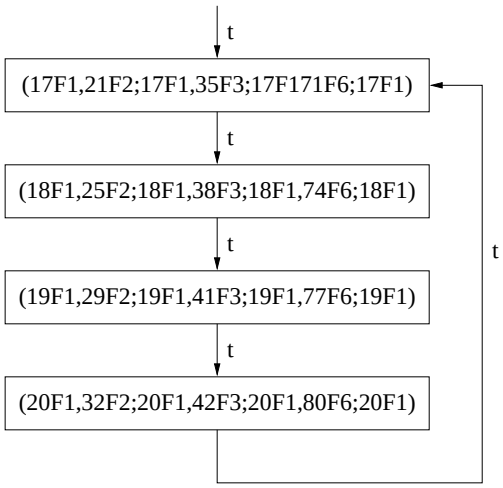

Figure 8.2: An $F1$-indeterminate cycle in G_{test3}

the cycle pictured in the figure, and will indefinitely loop in the cycle following the execution by the system of the trace $(t)^*$. In fact, since the lead transmitter is faulty (ltf) no vehicle will receive the synchronization pulse and nothing will be transmitted and/or received by any vehicle in the platoon; hence the execution of the event t, which indicates the expiration of the time slot, will indefinitely continue. The cycle in the picture is $F1$-indeterminate, since $ltf \in F1$, as none of the diagnosers is capable of detecting the fault that the lead vehicle transmitter is faulty. The lead vehicle does not receive any message from any follower and its diagnoser cannot differentiate whether its transmitter is faulty (represented by state $17F1$) or its receiver (represented by state 21F2) is faulty. The first (second) follower vehicle does not receive the synchronization pulse since the lead transmitter is faulty, and its diagnoser cannot determine whether the lead vehicle transmitter is faulty (state 17F1) or its own receiver is faulty (state 35F1 for the first follower, state 71F6 for the second follower).

The next step is to check whether Protocol 2 is capable of diagnosing all failure types. For that purpose we build the FSM G_{test2}, which is presented in Appendix A.7, to check for the existence of ambiguous cycles. One could verify that such cycles do not exist in G_{test2} and as a consequence of that Protocol 2 performs as well as the centralized diagnoser by Theorem 5.4.2.

As an illustration let us consider how Protocol 2 diagnoses the fault ltf that Protocol 3

cannot (cf. Figure 8.2). The application of Protocol 2 for that scenario is depicted in Table 8.2. From the previous discussion, we know that none of the diagnosers can detect

Table 8.2: Application of Protocol 2: system executes $ltf, t, t, t, t, \ldots$

Event	Lead Diagnoser	Follower 1 Diagnoser	Follower 2 Diagnoser	Coordinator C
ϵ	1N	1N	1N	1N
ltf	–	–	–	–
t	5N,18F1,25F2,38F3,49F4,62F5,74F6	18F1,38F3	18F1,74F6	18F1
t	9N,19F1,29F2,41F3,53F4,66F5,77F6	19F1,41F3	19F1,77F6	19F1
t	20F1,32F2,42F3,54F4	20F1,42F3	20F1,80F6	20F1
t	17F1,21F2	17F1,35F3	17F1,71F6	17F1
t	18F1,25F2	18F1,38F3	18F1,74F6	18F1
t	19F1,29F2	19F1,41F3	19F1,77F6	19F1
t	20F1,32F2	20F1,42F3	20F1,80F6	20F1
t	17F1,21F2	17F1,35F3	17F1,71F6	17F1

that the lead transmitter is faulty. However, if the information update rule of Protocol 2 is applied to the states of the diagnosers as depicted in Table 8.2, the fault is identified by the coordinator. Note here that the vehicles required the use of the wide area network, since if the lead vehicle transmitter is faulty, the lead vehicle cannot broadcast its diagnoser state to the coordinator using its faulty transmitter!

Since Protocol 2 performs as well as the centralized diagnoser, one can implement it to diagnose online the wireless LAN of the automated vehicle platoon. Moreover, since local clocks do exist at each site, that is, each vehicle possesses a local clock by the design of the communication protocol, one need not worry about communication delays and their effect on the performance and implementation of the protocol. Since local clocks do exist and they are periodically synchronized, messages sent to the coordinator by any site can be time stamped, and consequently Protocol 2 performs as well as the centralized diagnoser without the need of an ordering mechanism as the one used in Protocol 2D.

We observe that with the exception of state 49 (cf. Appendix A.7) in G_{test2}, all the states of the coordinator are equal to their corresponding centralized diagnoser states; however this does not rule out the presence of state-ambiguous traces, since the absence of such traces is only a sufficient condition for Protocol 2 to reconstruct the state of the centralized diagnoser.

We finally note that the scenario presented in Table 8.2 is a simple one, where following

the first event after the occurrence of the failure, the coordinator detects and isolates the failure. This is by no means the general case as pointed out during the discussion of the diagnostic properties of the protocols.

CHAPTER 9

AN OPTIMIZATION PROBLEM IN SENSOR SELECTION

We formulate and analyze an optimization problem in sensor selection that arises in failure diagnosis, detection and hypothesis testing, and is motivated by economic or energy-related considerations.

9.1 Introduction

We begin by describing two classes of problems in failure diagnosis and hypothesis testing that highlight the issues we are concerned with.

Assume that a dynamic system is diagnosable when a set Γ of sensors is used. By diagnosable we mean that any failure of interest can be detected and isolated within a finite time after its occurrence by a pre-specified failure diagnosis scheme. One such scheme could be the approach described in Chapter 2 where the set of sensors Γ is associated with a set of observable events. Assume that there is a cost associated with every subset of Γ. We wish (i) to identify a least expensive combination of sensors $A^* \subseteq \Gamma$ under which the system is diagnosable; (ii) to determine the minimum number of tests required to identify such a set A^*, assuming that we use a fixed pre-specified test for diagnosability, for instance the test of Theorem 2.5.1 if the approach for failure diagnosis used is that suggested in Chapter 2.

Consider an M-ary hypothesis testing problem. Assume that when a set Γ of sensors is available

$$P_\Gamma(\text{correct detection}|h_i, y_1, y_2, \ldots, y_T) > \alpha, i = 1, 2, \ldots, M, \tag{9.1}$$

where $P_\Gamma(\text{correct detection}|h_i, y_1, y_2, \ldots, y_T)$ denotes the probability that the i^{th} hypothesis is correctly identified when it is true, given T observations $y_1, y_2, \ldots, y_T$ collected from the set Γ of sensors, and α, $0 < \alpha < 1$, expresses the quality of performance required by the system over a T-horizon problem. Again, a cost is associated with every subset of Γ. We

wish (i) to identify a least expensive combination of sensors, call it A^*, such that

$$P_{A^*}(\text{correct detection}|h_i, y_1, y_2, \ldots, y_T) > \alpha, i = 1, 2, \ldots, M; \tag{9.2}$$

and (ii) to determine the minimum number of tests required to identify such a combination of sensors A^*, assuming that a pre-specified test is used to determine whether any set B of sensors satisfies

$$P_B(\text{correct detection}|h_i, y_1, y_2, \ldots, y_T) > \alpha, i = 1, 2, \ldots, M. \tag{9.3}$$

The costs associated with the sensors could have various interpretations. They could model the actual economic costs of using the given sensors. In this case the problem of finding a least costly set of sensors that diagnoses the system or satisfies (9.2) is of obvious economic interest. This objective is especially justified in situations where a combination of sensors is to be used in a large number of units (e.g. automobiles, appliances, airplanes, unmanned aerial vehicles, etc.) and the cost of that combination is to be taken as the total cost over all units. The cost of a sensor could also model its energy consumption, in which case our sensor selection problem becomes one of minimization of energy consumption in the process of diagnosing the system or satisfying (9.2). Furthermore, in all cases, it is desirable to minimize the expected computational (testing) cost associated with the identification of a least costly combination of sensors, because each test is time and memory consuming and requires significant computational effort.

We formulate and analyze an optimization problem in sensor selection motivated by the above examples. We assume that a system possesses a certain *Property D* (e.g. the system is diagnosable) when a set Γ of sensors is used for information gathering. There are no restrictions on the class of dynamic systems considered. For each set A of sensors that is a subset of Γ, there is an *a priori* probability p_A that the system possesses Property D. There is a cost c_A associated with each set A of sensors that is a subset of Γ. Given a set A of sensors that is a subset of Γ, it is possible to determine, via a pre-specified test, whether the resulting system-sensor combination possesses Property D. The objective is to determine a test strategy, i.e., a sequence of tests, to minimize the expected cost, associated with the tests, that is incurred until a least expensive system-sensor combination that possesses Property D is identified.

Related sensor selection problems have been previously studied in [5, 20, 10] in the context of discrete event systems. In [5] the authors present an algorithm that determines the minimum-cost set of sensors which ensures system testability (a property defined in [5]). In [20] ([10]), the authors present algorithms that minimize the cardinality of the set of observable events and result in observable (normal) languages (see, e.g., [7] for definitions

of these properties). The problems in [5], [20], [10] as well as the problem investigated here have the following similar objective: the goal is to determine an optimal (according to some criterion) set of sensors or events with respect to which the system possesses a certain given property. A major difference, however, between [5, 20, 10] and the present work is that we also determine a sequence of tests that minimizes the computational effort required to identify a least costly set of sensors that results in a system possessing the required property; this issue is not addressed in [5, 20, 10].

9.2 Problem Formulation

In principle, we can formulate the sensor selection problem described in Section 9.1 as a Markovian decision problem with perfect observations (cf. Section 9.3.2). Such a formulation, without any further assumptions, would only allow determination of the optimal sequence of tests by computational methods, and would not provide any further insight into the structure of the optimal policy. To develop insight into the structure of the problem, we need to isolate instances where we can explicitly determine by analytical arguments the optimal sequence of tests. We present such an instance in this chapter, where the analytical solution relies on specific assumptions on the costs of sensors and the *a priori* probabilities.

We formulate and analyze the following sensor selection problem.

Problem P

We are given a dynamic system and a set Γ of sensors. We make the following assumptions:

P-A1 The system possesses Property D when the set Γ of sensors is used. The cardinality of Γ is denoted by K.

P-A2 Any combination of i sensors ($i < K$) that is a subset of Γ is less costly than any combination of $(i+1)$ sensors that is also a subset of Γ. The cost associated with any specific combination of i sensors ($i = 1, 2, \ldots, K-1$) may depend on the combination.

P-A3 The *a priori* probability that the system possesses Property D when a combination of i sensors is used is p_i. Furthermore,

$$0 = p_0 < p_1 \leq p_2 \ldots \leq p_{K-1} < p_K = 1. \tag{9.4}$$

P-A4 If the system with a set A of sensors is tested and found not to possess Property D, then $p_{|B|}$ remains the same for all set of sensors $B \supset A$. If the system with

a set A of sensors is tested and found to possess Property D, then $p_{|C|}$ remains the same for all set of sensors $C \subset A$.

P-A5 A test performed off-line on any combination of sensors reveals whether the system possesses Property D when that combination of sensors is used. The cost of the test is independent of the combination of sensors on which it is performed, and is equal to c.

The objective is to determine a sequence of tests that minimizes the expected cost (associated with these tests) incurred until a least costly combination of sensors with respect to which the system possesses Property D is identified.

We briefly discuss the assumptions made in the above problem formulation. Without Assumption **P-A1** there is no problem to solve. If the cost of any combination of i sensors is equal to the sum of the costs of its individual components and the individual sensor costs are comparable, then Assumption **P-A2** is a reasonable assumption. However, Assumption **P-A2** may not always be true. Moreover, under Assumption **P-A2** ordering sensor combinations according to their cost is the same as ordering them according to their cardinality. In Assumption **P-A3** the probabilities p_i, $i = 1, 2, \ldots, K-1$, are the result of prior knowledge based on experimental data. Assumption **P-A3** is reasonable under the stipulation that all sensors have comparable information gathering ability. Assumption **P-A4** can be justified as follows: the addition (subtraction) of a sensor to (from) a set of sensors increases (decreases) the diagnostic ability of the resulting set of sensors in a highly nonlinear fashion that is very difficult (if not impossible) to quantify. Thus, it is reasonable to assume that: (i) a negative test on a set A of sensors does not provide any additional information about the outcome of the test on any set B of sensors that is a superset of A; and (ii) a positive test on a set A of sensors does not provide any information on any set C of sensors that is a subset of A. Finally, Assumption **P-A5** is true in several instances. For example, it holds in the M-ary hypothesis testing problem discussed in Section 9.1, and in the case when Property D is that of diagnosability introduced in Chapter 2 and the test for diagnosability is the one specified by Theorem 2.5.1.

9.3 Solution

9.3.1 Preliminaries

We can visualize **Problem P** as a decision problem on a digraph (directed graph) as follows. The nodes of the digraph are elements of 2^{Γ}, the power set of Γ. Thus, each node represents a combination of sensors. Two nodes A and B are connected by a link, whose

direction is from node B to node A, if and only if cardinality(B) = cardinality(A) + 1 **and** $A \subset B$. Node $\emptyset$ of the network, called the *empty node*, corresponds to the empty subset of Γ, i.e., there are no sensors. An example of such a digraph in the case where there are four sensors is shown in Figure 9.1. The objective is to minimize the expected number of tests,

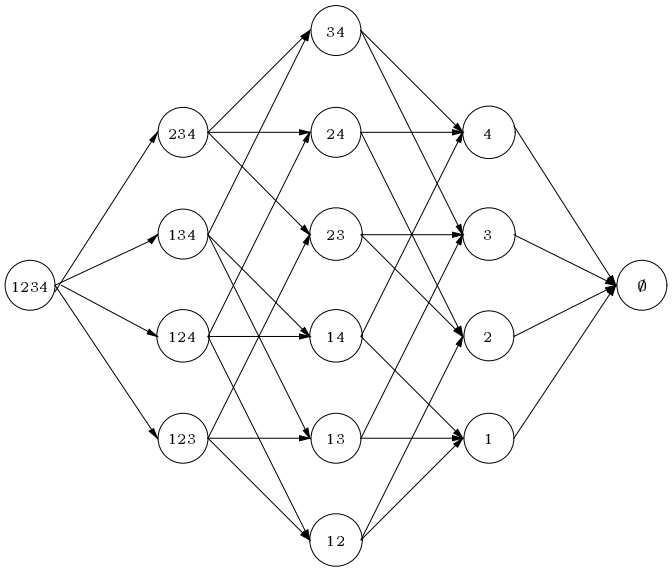

Figure 9.1: The digraph in the case where there are 4 sensors. The notation 12, for instance, is used to denote the set of sensors {1,2}

each one performed on one node of the digraph, required to identify a node that represents a least costly combination of sensors with respect to which the system possesses Property D. Such a visualization facilitates the analysis of **Problem P**.

To analyze **Problem P** we need the following concepts.

Definition 9.3.1 *Node B in the digraph possesses* Property D *if the system possesses* Property D *when the combination of sensors contained in B is used.*

Definition 9.3.2 *Node B in the digraph possesses* Property $\overline{D}$ *if it does not possess Property D.*

The above definitions imply that the nodes of the digraph possess the following characteristics.

C1 The node containing the set Γ, or simply node Γ, possesses Property D.

C2 Node $\emptyset$ possesses Property $\overline{D}$.

C3 If node A possesses Property D, node $B \supset A$ also possesses Property D.

C4 If node A possesses Property $\overline{D}$, node $B \subset A$ also possesses Property $\overline{D}$.

In addition, the following definitions are needed in the subsequent development of the results.

Definition 9.3.3 *Node B in the digraph is said to be a child of node C in the digraph if and only if there exists a directed path in the digraph from C to B.*

Definition 9.3.4 *Node B in the digraph is said to be a parent of node C in the digraph if and only if there exists a directed path in the digraph from B to C.*

Definition 9.3.5 *Let ξ be a set of nodes in the digraph and let $A \in \xi$. A reachable set from ξ, conditioned on the fact that A possesses* Property D*, is a set composed of all nodes in ξ whose cost is strictly less than that of A.*

Definition 9.3.6 *Let ξ be a set of nodes in the digraph and let $A \in \xi$. A reachable set from ξ, conditioned on the fact that A possesses* Property $\overline{D}$*, is a set composed of all nodes in ξ except node A and its children in ξ.*

Definition 9.3.7 *A reachable set is a set that results from applying an arbitrary sequence of n tests $\{t_1, t_2, \ldots, t_n\}$ to the set of nodes $2^{\Gamma} \backslash \{\emptyset, \Gamma\}$, while following the rules in Definitions 9.3.5 and 9.3.6.*

The concepts introduced by Definitions 9.3.1 – 9.3.7 as well as the characteristics **C1** – **C4** are subsequently used in the formulation and analysis of **Problem P** as a Markovian Decision Problem. We note that Definitions 9.3.1 – 9.3.7 are valid independently of whether Assumptions **P-A2** - **P-A4** are true or not. We illustrate the above concepts by the following example.

Example 9.3.1 *Consider the digraph presented in Figure 9.1. Let $\xi = 2^{\Gamma} \setminus \{\emptyset, \Gamma\}$, $A = \{1, 2, 3\}$, and $B = \{1, 2\}$. Assume that the costs of combinations of sensors are ordered as follows*

$$\{1\} < \{2\} < \{3\} < \{1,2\} < \{4\} < \{1,3\} < \{2,3\} < \{1,4\} < \{2,4\} < \{1,2,3\} <$$

$$< \{3,4\} < \{1,2,4\} < \{1,3,4\} < \{2,3,4\}.$$

We have the following:

- *The children of A in ξ are $\{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}$, and $\{2,3\}$.*

- *The parents of B in ξ are* $\{1,2,3\}$ *and* $\{1,2,4\}$.
- *The reachable set from ξ, conditioned on the fact that B possesses* Property D, *is the set* $\{\{1\},\{2\},\{3\}\}$.
- *The reachable set from ξ, conditioned on the fact that A possesses* Property $\overline{D}$, *is the set* $\{\{4\},\{1,4\},\{2,4\},\{3,4\},\{1,2,4\},\{1,3,4\},\{2,3,4\}\}$.

Based on the above preliminaries, we next proceed to analyze **Problem P**.

9.3.2 Analysis

We formulate **Problem P** as a Markovian decision problem with perfect observations (see [33]). We then exploit the characteristics of reachable sets using **C3** and **C4**, and Assumptions **P-A1** – **P-A5** to characterize an optimal strategy for the Markovian decision problem.

The *information state*[1] [23] of the process is the set of nodes of the digraph that have not yet been checked for Property D **and** corresponds to potentially least costly combinations of sensors possessing Property D. In fact, an information state is a reachable set as defined in Section 9.3.1. Let $V(N)$ denote the minimum expected (test) cost incurred when the state is N. Then, $V(N)$ satisfies the optimality equation

$$V(N) = \min_{i \in N} \{c + p_{|i|} * V(N_i^D) + (1 - p_{|i|}) * V(N_i^{\overline{D}})\}, \tag{9.5}$$

where $|i|$ denotes the cardinality of the set i, N_i^D ($N_i^{\overline{D}}$) is the new state of the process, i.e., the reachable set from node i, that results when node i possesses Property D ($\overline{D}$). These states are determined from N by using **C3** and **C4**, Definitions 9.3.5 and 9.3.6, and Assumption **P-A2**. The probabilities $p_{|i|}$, $|i| = 1, 2, \ldots, K-1$ remain unchanged because of Assumption **P-A4**. With a slight abuse of notation, and for simplicity, we thereafter use p_i instead of $p_{|i|}$ in (9.5) to denote the *a priori* probability associated with the set of sensors i. Hence, (9.5) reads as follows

$$V(N) = \min_{i \in N} \{c + p_i * V(N_i^D) + (1 - p_i) * V(N_i^{\overline{D}})\}. \tag{9.6}$$

As stated earlier, Equation (9.6) can be solved by standard computational methods (e.g. value iteration, policy iteration etc ..., see chapter 8 of [23]) under any conditions on the prior probabilities p_i and on the cost of combinations of sensors. A computational solution of Equation (9.6) would not provide any insight into the structure of optimal test strategies. Under the set of Assumptions **P-A1** - **P-A5** we have an instance of the sensor selection

[1] In this chapter we use the terms state and information state interchangeably.

problem where the optimal sequence of tests can be explicitly determined by analytical arguments. The following theorem summarizes the solution to **Problem P** and provides the main result of this chapter.

Theorem 9.3.1 *If* $p_l + p_{l+1} \geq 1$ *for* $l = 1, \ldots, K-2$, *then* **an** *optimal test strategy for* **Problem P** *is to test combinations of sensors in increasing order of (sensor) cost.*

Proof of Theorem 9.3.1 The proof proceeds by induction on the cardinality of the information state N. Let $V_k(N)$ denote the expected cost incurred when we choose to test $k \in N$ when the information state is N and follow the optimal test strategy afterwards, that is

$$V_k(N) = c + p_k * V(N_k^D) + (1 - p_k) * V(N_k^{\overline{D}}) \tag{9.7}$$

and

$$V(N) = \min_{k \in N} V_k(N). \tag{9.8}$$

- *Basis of induction* Let $|N| = 2$, i.e., $N = \{i, j\}$. There are three possibilities for such information states (based on Definitions 9.3.5 and 9.3.6 of reachable sets):

  1. $|i| = |j|$ such that the cost attached to i is less than or equal to that attached to j. In this case

  $$V_i(N) = c + p_i * V(N_i^D) + (1 - p_i) * V(N_i^{\overline{D}}) = c + (1 - p_i) * c, \tag{9.9}$$

  since $N_i^D = \emptyset$, i.e., no more tests are needed, and $N_i^{\overline{D}} = \{j\}$. Also,

  $$V_j(N) = c + p_j * V(N_j^D) + (1 - p_j) * V(N_j^{\overline{D}}) = c + p_j * c + (1 - p_j) * c, \tag{9.10}$$

  since $N_j^D = \{i\}$, and $N_j^{\overline{D}} = \{i\}$. From (9.9) and (9.10) it follows that it is optimal to first test i and then j since $p_i = p_j$, and $p_j * c$ is always strictly positive.

  2. $|i| = |j| - 1$ with i being a child of j. In this case

  $$V_i(N) = c + p_i * V(N_i^D) + (1 - p_i) * V(N_i^{\overline{D}}) = c + (1 - p_i) * c, \tag{9.11}$$

  since $N_i^D = \emptyset$ (by Assumption **P-A2**), i.e., no more tests are needed, and $N_i^{\overline{D}} = \{j\}$. Also,

  $$V_j(N) = c + p_j * V(N_j^D) + (1 - p_j) * V(N_j^{\overline{D}}) = c + p_j * c, \tag{9.12}$$

  since $N_j^D = \{i\}$, and $N_j^{\overline{D}} = \emptyset$. From (9.11) and (9.12) we conclude that it is optimal to first test i and then j because by assumption $p_i + p_j \geq 1$, implying that $1 - p_i \leq p_j$.

3. $|i| = |j| - 1$ such that i is not a child of j. In this case

$$V_i(N) = c + p_i * V(N_i^D) + (1 - p_i) * V(N_i^{\overline{D}}) = c + (1 - p_i) * c, \tag{9.13}$$

since $N_i^D = \emptyset$ (by Assumption **P-A2**), i.e., no more tests are needed, and $N_i^{\overline{D}} = \{j\}$. Also,

$$V_j(N) = c + p_j * V(N_j^D) + (1 - p_j) * V(N_j^{\overline{D}}) = c + p_j * c + (1 - p_j) * c = c + c, \tag{9.14}$$

since $N_j^D = \{i\}$, and $N_j^{\overline{D}} = \{i\}$. From (9.13) and (9.14) it follows that it is optimal to first test i and then j since $1 - p_i$ is always strictly less than 1.

- *Induction step* Assume that the assertion of the theorem is true for any set N' with cardinality k, i.e., under Assumptions **P-A1** - **P-A5** and $p_l + p_{l+1} \geq 1$ for all l, and for any set of cardinality less than or equal to k an optimal test strategy is to test combinations of sensors by increasing order of their cost. We must prove that for any set N of cardinality $k+1$, the assertion of the theorem is still true. Let i, j be elements of N such that the cost attached to i is the smallest cost attached to any combination of sensors in N. Clearly, $|j| \geq |i|$ by Assumption **P-A2**. We need to prove that

$$\begin{aligned} V_i(N) &= c + p_i * V(N_i^D) + (1 - p_i) * V(N_i^{\overline{D}}) \leq c + p_j * V(N_j^D) + (1 - p_j) * V(N_j^{\overline{D}}) \\ &= V_j(N), \end{aligned} \tag{9.15}$$

or

$$p_i * V(N_i^D) + (1 - p_i) * V(N_i^{\overline{D}}) \leq p_j * V(N_j^D) + (1 - p_j) * V(N_j^{\overline{D}}), \tag{9.16}$$

where

$$\begin{aligned} N_i^D &= \emptyset, \\ N_i^{\overline{D}} &= N \backslash \{i\}, \\ N_j^D &= \{x \in N : |x| < |j|, \text{ or } |x| = |j| \text{ and the cost attached to } x \text{ is less than} \\ &\qquad \text{that attached to } j \text{ and } x \neq j\}, \\ N_j^{\overline{D}} &= \{x \in N : x \neq j, \text{ and } x \text{ is not a child of } j\}. \end{aligned} \tag{9.17}$$

We can apply the induction hypothesis to the sets $N_i^{\overline{D}}$, N_j^D, and $N_j^{\overline{D}}$ since their cardinalities are less than or equal to k (cf. (9.17)). By successively applying the

induction hypothesis until all elements in $N_i^{\overline{D}}$, N_j^D, and $N_j^{\overline{D}}$ are exhausted, we get

$$V(N_i^{\overline{D}}) = c * \left[\sum_{n=0}^{v_i-1} (1-p_i)^n + (1-p_i)^{v_i} * \sum_{n=0}^{v_{i+1}-1} (1-p_{i+1})^n + \quad \ldots \right.$$
$$+(1-p_i)^{v_i} * (1-p_{i+1})^{v_{i+1}} * \ldots * (1-p_{j-1})^{v_{j-1}} * \sum_{n=0}^{v_j-1} (1-p_j)^n + \quad \ldots$$
$$\left. +(1-p_i)^{v_i} * (1-p_{i+1})^{v_{i+1}} * \ldots * (1-p_{L-1})^{v_{L-1}} * \sum_{n=0}^{v_L-1} (1-p_L)^n \right], \tag{9.18}$$

$$V(N_j^D) = c * \left[\sum_{n=0}^{r_i-1} (1-p_i)^n + (1-p_i)^{r_i} * \sum_{n=0}^{r_{i+1}-1} (1-p_{i+1})^n + \quad \ldots \right.$$
$$\left. +(1-p_i)^{r_i} * (1-p_{i+1})^{r_{i+1}} * \ldots * (1-p_{j-1})^{r_{j-1}} * \sum_{n=0}^{r_j-1} (1-p_j)^n \right], \tag{9.19}$$

and

$$V(N_j^{\overline{D}}) = c * \left[\sum_{n=0}^{s_i-1} (1-p_i)^n + (1-p_i)^{s_i} * \sum_{n=0}^{s_{i+1}-1} (1-p_{i+1})^n + \quad \ldots \right.$$
$$\left. +(1-p_i)^{s_i} * (1-p_{i+1})^{s_{i+1}} * \ldots * (1-p_{L-1})^{s_{L-1}} * \sum_{n=0}^{s_L-1} (1-p_L)^n \right], \tag{9.20}$$

where v_k denotes the number of combinations of k sensors in $N_i^{\overline{D}}$, r_k denotes the number of combinations of k sensors in N_j^D, s_k denotes the number of combinations of k sensors in $N_j^{\overline{D}}$ and L denotes the highest set cardinality in N with $j \leq L \leq K-1$. In Appendix B.1 we explain in more detail how Equations (9.18)–(9.20) are obtained. Because of (9.17)–(9.20), the inequality we need to prove in (9.16) can be written as follows (after canceling c from both sides)

$$(1-p_i) * \left[\sum_{n=0}^{v_i-1} (1-p_i)^n + (1-p_i)^{v_i} * \sum_{n=0}^{v_{i+1}-1} (1-p_{i+1})^n + \quad \ldots \right.$$
$$+(1-p_i)^{v_i} * (1-p_{i+1})^{v_{i+1}} * \ldots * (1-p_{j-1})^{v_{j-1}} * \sum_{n=0}^{v_j-1} (1-p_j)^n + \quad \ldots$$

$$+(1-p_i)^{v_i} * (1-p_{i+1})^{v_{i+1}} * \ldots * (1-p_{L-1})^{v_{L-1}} * \sum_{n=0}^{v_L-1} (1-p_L)^n \Bigg] \quad \leq$$

$$p_j * \Bigg[\sum_{n=0}^{r_i-1} (1-p_i)^n + (1-p_i)^{r_i} * \sum_{n=0}^{r_{i+1}-1} (1-p_{i+1})^n + \quad \ldots$$

$$+(1-p_i)^{r_i} * (1-p_{i+1})^{r_{i+1}} * \ldots * (1-p_{j-1})^{r_{j-1}} * \sum_{n=0}^{r_j-1} (1-p_j)^n \Bigg] +$$

$$+(1-p_j) * \Bigg[\sum_{n=0}^{s_i-1} (1-p_i)^n + (1-p_i)^{s_i} * \sum_{n=0}^{s_{i+1}-1} (1-p_{i+1})^n + \quad \ldots$$

$$+(1-p_i)^{s_i} * (1-p_{i+1})^{s_{i+1}} * \ldots * (1-p_{L-1})^{s_{L-1}} * \sum_{n=0}^{s_L-1} (1-p_L)^n \Bigg]. \qquad (9.21)$$

We consider two cases: $|j| > |i|$, and $|j| = |i|$ such that the cost attached to j is higher than that attached to i.

Case 1 $|j| > |i|$. In this case, from the definition of reachable sets (Definitions 9.3.5 and 9.3.6), **C3** and **C4** we have the following relations

$$\begin{aligned}
r_j &< v_j \\
r_k &= v_k \qquad k = i+1, \ldots, j-1 \\
r_i &= v_i + 1 \\
s_k &= v_k \qquad k = j+1, \ldots, L \\
s_j &= v_j - 1 \\
s_k &\leq v_k \qquad k = i+1, \ldots, j-1 \\
s_i &\leq v_i + 1. \qquad (9.22)
\end{aligned}$$

The set of equalities and inequalities (9.22) is best explained by referring to (9.17). For instance, $r_j < v_j$ since the elements in $N_i^{\overline{D}}$ having $|j|$ sensors (denoted by v_j) are equal to the elements in N having $|j|$ sensors while the elements in N_j^D having $|j|$ sensors (denoted by r_j) are those elements in $N \backslash \{j\}$ having $|j|$ sensors **and** a cost strictly less than that of element j. Verification of the remaining equalities and inequalities can be found in Appendix B.2. Because of (9.21) and of the relations between the r_k's and v_k's, $k = i, \ldots, j-1$, in (9.22), the inequality we must prove to complete the proof of the induction step in *Case 1*

is

$$(1-p_i)*\left[\sum_{n=0}^{v_i-1}(1-p_i)^n+(1-p_i)^{v_i}*\sum_{n=0}^{v_{i+1}-1}(1-p_{i+1})^n+\quad\ldots\right.$$
$$+(1-p_i)^{v_i}*(1-p_{i+1})^{v_{i+1}}*\ldots*(1-p_{j-1})^{v_{j-1}}*\sum_{n=0}^{v_j-1}(1-p_j)^n+\quad\ldots$$
$$\left.+(1-p_i)^{v_i}*(1-p_{i+1})^{v_{i+1}}*\ldots*(1-p_{L-1})^{v_{L-1}}*\sum_{n=0}^{v_L-1}(1-p_L)^n\right]\quad\leq$$
$$p_j*\left[\sum_{n=0}^{v_i}(1-p_i)^n+(1-p_i)^{v_i+1}*\sum_{n=0}^{v_{i+1}-1}(1-p_{i+1})^n+\quad\ldots\right.$$
$$\left.+(1-p_i)^{v_i+1}*(1-p_{i+1})^{v_{i+1}}*\ldots*(1-p_{j-1})^{v_{j-1}}*\sum_{n=0}^{r_j-1}(1-p_j)^n\right]+$$
$$+(1-p_j)*\left[\sum_{n=0}^{s_i-1}(1-p_i)^n+(1-p_i)^{s_i}*\sum_{n=0}^{s_{i+1}-1}(1-p_{i+1})^n+\quad\ldots\right.$$
$$\left.+(1-p_i)^{s_i}*(1-p_{i+1})^{s_{i+1}}*\ldots*(1-p_{L-1})^{s_{L-1}}*\sum_{n=0}^{s_L-1}(1-p_L)^n\right].\qquad(9.23)$$

Because of Assumption **P-A3**, the condition $p_l+p_{l+1}\geq 1$ for all l of the theorem implies that $1-p_i\leq p_j$. Moreover, the terms multiplied by p_j on the right hand side of (9.23) are term by term greater than or equal to the first $v_i+1+v_{i+1}+\ldots+v_{j-1}+r_j$ terms multiplied by $(1-p_i)$ (in between braces) on the left hand side. Hence, because of (9.23) and the above observation, to complete the proof of the induction step in *Case 1* it is sufficient to prove that

$$(1-p_i)*(1-p_i)^{v_i}*\ldots*(1-p_{j-1})^{v_{j-1}}*(1-p_j)^{r_j+1}*\left[\sum_{n=0}^{v_j-r_j-2}(1-p_j)^n\right.$$
$$+(1-p_j)^{v_j-r_j-1}*\sum_{n=0}^{v_{j+1}-1}(1-p_{j+1})^n$$
$$\left.+(1-p_j)^{v_j-r_j-1}*(1-p_{j+1})^{v_{j+1}}*\ldots*(1-p_{L-1})^{v_{L-1}}*\sum_{n=0}^{v_L-1}(1-p_L)^n\right]$$
$$\leq(1-p_j)*\left[\sum_{n=0}^{s_i-1}(1-p_i)^n+(1-p_i)^{s_i}*\sum_{n=0}^{s_{i+1}-1}(1-p_{i+1})^n+\ldots\right.$$
$$\left.+(1-p_i)^{s_i}*(1-p_{i+1})^{s_{i+1}}*\ldots*(1-p_{L-1})^{s_{L-1}}*\sum_{n=0}^{s_L-1}(1-p_L)^n\right].\qquad(9.24)$$

Since $(1-p_i)*(1-p_i)^{v_i}*(1-p_{i+1})^{v_{i+1}}*\ldots*(1-p_{j-1})^{v_{j-1}}*(1-p_j)^{r_j+1} \leq (1-p_j)$, to establish (9.24) it is sufficient to show that

$$\sum_{n=0}^{v_j-r_j-2}(1-p_j)^n + (1-p_j)^{v_j-r_j-1} * \sum_{n=0}^{v_{j+1}}(1-p_{j+1})^n + \quad \ldots$$

$$+(1-p_j)^{v_j-r_j-1} * (1-p_{j+1})^{v_{j+1}} * \ldots * (1-p_{L-1})^{v_{L-1}} * \sum_{n=0}^{v_L-1}(1-p_L)^n \quad \leq$$

$$\sum_{n=0}^{s_i-1}(1-p_i)^n + (1-p_i)^{s_i} * \sum_{n=0}^{s_{i+1}-1}(1-p_{i+1})^n + \quad \ldots$$

$$+(1-p_i)^{s_i} * (1-p_{i+1})^{s_{i+1}} * \ldots * (1-p_{L-1})^{s_{L-1}} * \sum_{n=0}^{s_L-1}(1-p_L)^n. \quad (9.25)$$

From (9.22) we have that $s_j = v_j - 1 > v_j - r_j - 1$ since $r_j \geq 0$, and $s_k = v_k$, $k = j+1, \ldots, L$. Hence, the following is true

$$(v_j - r_j - 1) + v_{j+1} + \ldots + v_L \leq s_i + s_{i+1} + \ldots + s_L. \quad (9.26)$$

Inequality (9.26) implies that (9.25) is true since the terms in the left hand side of (9.25) are term by term less than or equal to the first $(v_j - r_j - 1) + v_{j+1} + \ldots + v_L$ terms on the right hand side of (9.25) (by noting that $(1-p_j) \leq (1-p_i)$ since $p_i \leq p_j$). The proof of the induction step is complete and the assertion of the theorem is true for *Case 1.*

Case 2 $|j| = |i|$ such that the cost attached to j is higher than that attached to i. In this case Inequality (9.21) becomes

$$(1-p_i) * \left[\sum_{n=0}^{v_i-1}(1-p_i)^n + (1-p_i)^{v_i} * \sum_{n=0}^{v_{i+1}-1}(1-p_{i+1})^n + \quad \ldots\right.$$

$$\left.+(1-p_i)^{v_i} * (1-p_{i+1})^{v_{i+1}} * \ldots * (1-p_{L-1})^{v_{L-1}} * \sum_{n=0}^{v_L-1}(1-p_L)^n\right] \quad \leq$$

$$p_i * \sum_{n=0}^{r_i-1}(1-p_i)^n$$

$$+(1-p_i) * \left[\sum_{n=0}^{s_i-1}(1-p_i)^n + (1-p_i)^{s_i} * \sum_{n=0}^{s_{i+1}-1}(1-p_{i+1})^n + \quad \ldots\right.$$

$$\left.+(1-p_i)^{s_i} * (1-p_{i+1})^{s_{i+1}} * \ldots * (1-p_{L-1})^{s_{L-1}} * \sum_{n=0}^{s_L-1}(1-p_L)^n\right], \quad (9.27)$$

where v_k, r_k, s_k, and L are as defined in *Case 1*, and they are related as follows

$$\begin{aligned} r_i &\leq v_i \\ s_k &= v_k \qquad k = i, \ldots, L. \end{aligned} \tag{9.28}$$

Observe that $s_k = v_k$ for $k = i, \ldots, L$ since the sets $N_i^{\overline{D}}$ and $N_j^{\overline{D}}$ contain the same combinations of $|i|+1$ sensors or more. The inequality $r_i \leq v_i$ is true since the set $N_i^{\overline{D}}$ contains all elements of $|i|$ sensors in N except the element i, while N_j^D only contains elements with $|i|$ sensors that have a cost strictly less than that of the element j.

Hence, to complete the proof of the induction step in Case 2 we must prove that (9.27) is true. For that matter we observe that the terms multiplied by $(1 - p_i)$ on the left hand side of (9.27) are term by term equal to the terms multiplied by $(1 - p_i)$ on the right hand side of (9.27). Hence the inequality in (9.27) is always true.

The proof of the induction step is now complete. Therefore, the assertion of the theorem is true. **Q.E.D.**

Theorem 9.3.1 presents conditions sufficient to guarantee that the strategy that tests combinations of sensors in increasing order of cost is optimal for **Problem P**. This result may be intuitively interpreted as follows: the conditions $p_l + p_{l+1} \geq 1$, $l = 1, 2, \ldots, K-2$, imply that $p_i \geq \frac{1}{2}$ for $i = 2, \ldots, K-1$. Thus, it is assumed a priori that a combination of two or more sensors is equally or more likely to result in a system possessing Property D than a system possessing Property $\overline{D}$. Therefore, given that the cost of testing for Property D is independent of the cardinality of the combination of sensors that is being used, it is plausible that the optimal test will examine combinations of sensors in increasing order of cost.

The proof of Theorem 9.3.1 is achieved by showing that the test strategy claimed to be optimal performs no worse than any other strategy (in an expected sense). The method of proof of the theorem cannot be used to compare *any* two test strategies, none of which is in the one specified by the theorem. Later, in Section 9.3.3, we show that under a slightly modified set of assumptions it is possible to compare test strategies that satisfy certain conditions even when none of these strategies is an optimal one (see Theorem 9.3.2).

Using the result of Theorem 9.3.1, it is possible to determine an analytic expression for the value function $V(N)$ that satisfies Equation (9.6).

Corollary 9.3.1 *Consider* **Problem P** *with the conditions* $p_l + p_{l+1} \geq 1$, $l = 1, 2, \ldots, K-2$.

(i) Let N be a reachable set. Then the minimum expected cost associated with N is

$$V(N) = c * \left[\sum_{n=0}^{n_l-1} (1-p_l)^n + (1-p_l)^{n_l} * \sum_{n=0}^{n_{l+1}-1} (1-p_{l+1})^n + \quad \ldots \right.$$

$$\left. + (1-p_l)^{n_l} * (1-p_{l+1})^{n_{l+1}} * \ldots * (1-p_{h-1})^{n_{h-1}} * \sum_{n=0}^{n_h-1} (1-p_h)^n \right], \qquad (9.29)$$

where l and h denote, respectively, the lowest and highest set cardinalities in N and n_l and n_h denote the number of combinations of sensors in N of cardinality l and h, respectively.

(ii) Let N_1 and N_2 be two reachable sets. If $N_1 \subseteq N_2$ then $V(N_1) \leq V(N_2)$.

Proof of Corollary 9.3.1 The proof of part (i) of this corollary can be found in the proof of Theorem 9.3.1. Namely, in the proof of Theorem 9.3.1 we showed that (9.21) and (9.27) are always true, hence the left hand side of the inequalities (after adding the constant c) represents the cost associated with the optimal policy. That cost can be rewritten as presented in the statement of the corollary. The proof of part (ii) of the corollary is a direct consequence of part (i). If we write the expansions of $V(N_1)$ and $V(N_2)$ according to (9.29), with the terms ordered as given in (9.29), then the $|N_1|$ elements in $V(N_1)$ are term by term less than the first $|N_1|$ elements in $V(N_2)$ since $(1-p_i) > (1-p_{i+1})$ for all i. **Q.E.D.**

The following example illustrates in part the results of our analysis. It shows that when the conditions $p_l + p_{l+1} \geq 1$, $l = 1, 2, \ldots, K-2$, are not satisfied the result of Theorem 9.3.1 is not, in general, true. It also provides a comparison of the test policy g_1, that tests combinations of sensors in increasing order of cost, with two other specific policies, and demonstrates that when the conditions of Theorem 9.3.1 are satisfied, g_1 outperforms these two policies.

Example 9.3.2 *Consider the digraph presented in Figure 9.1 and let the information state $N = 2^\Gamma \setminus \{\Gamma, \emptyset\}$. Assume that the combination of sensors $\{1,2,3,4\}$ achieves Property D, the costs of combinations of sensors are ordered as follows*

$$\{1\} < \{2\} < \{3\} < \{4\} < \{1,2\} < \{1,3\} < \{1,4\} < \{2,3\} < \{2,4\} < \{3,4\} <$$

$$< \{1,2,3\} < \{1,2,4\} < \{1,3,4\} < \{2,3,4\},$$

$p_1 < p_2 < p_3$, and the cost for testing for Property D is equal to unity. Hence Assumptions **P-A1 - P-A3** *and* **P-A5** *are met. We also assume that Assumption* **P-A4** *is satisfied. Denote by g_1 the policy that always tests combinations of sensors in increasing order of cost. We compare g_1 with two other policies g_2 and g_3 defined as follows:*

g_2 *Always test combinations of sensors in decreasing order of cardinality with the tie-breaker being the maximum number of children a combination has;*

g_3 *Always test combinations of sensors whose cardinality equals the median of available cardinalities in the information state with the tie-breaker being the maximum number of children a combination has.*

Figure 9.2 depicts the cost associated with the three policies above as function of p_2 *with fixed* p_1 *and* p_3*. In the top left graph, the conditions of Theorem 9.3.1 are satisfied, so* g_1 *outperforms* g_2 *and* g_3 *as expected. In the top right (bottom left) graph the condition* $p_1 + p_2 \geq 1$ *(all conditions) of Theorem 9.3.1 is (are) violated; yet* g_1 *outperforms* g_2 *and* g_3 *in both instances. In the bottom right graph, the conditions* $p_l + p_{l+1} \geq 1$, $l = 1, 2$, *are all violated and* g_1 *is inferior to* g_2 *and* g_3*.*

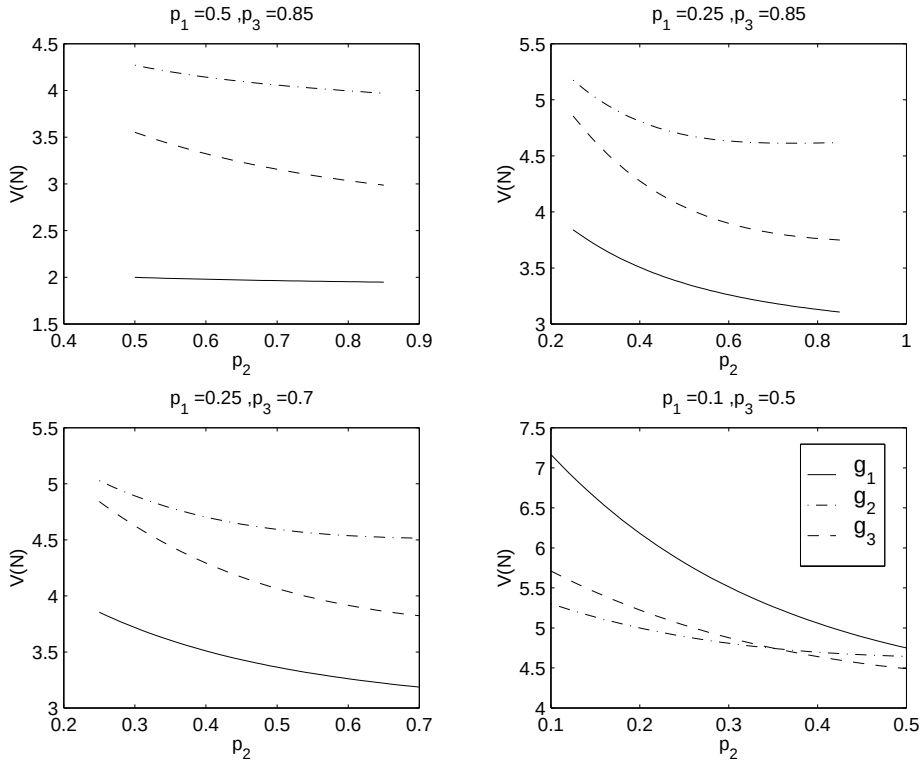

Figure 9.2: Policy comparisons for Example 9.3.2

9.3.3 Analysis of a Special Instance of Problem P

As discussed in Section 9.3.2, under Assumptions **P-A1** - **P-A5** we can show that the test strategy described in Theorem 9.3.1 is optimal but the theorem only includes

a comparison between any strategy and an optimal strategy. By slightly modifying one assumption we can obtain one instance of **Problem P** that has more structural properties than those described in Theorem 9.3.1. Specifically, by replacing Assumption **P-A3** with

P-A3′ The *a priori* probability that the system possesses Property D when the combination of i sensors is used, $1 < i < K-1$, is equal to a constant, p,

we achieve the following:

- Comparison of *any* two test strategies that satisfy certain conditions even when none of these strategies is an optimal one (cf. Theorem 9.3.2).
- Discovery of an additional property of the value function $V(\cdot)$ (cf. Theorem 9.3.3).

Assumption **P-A3′** is not very realistic. It may be justified only in the case where there is no prior experimental data available about the diagnostic properties of any set of sensors; in such a case one might choose $p = p_i = \frac{1}{2}$ for all i. The result of Theorem 9.3.2 that follows is valid for all $p \geq \frac{1}{2}$; Theorem 9.3.3 is true for all p.

Under Assumptions **P-A1**, **P-A2**, **P-A3′**, **P-A4**, and **P-A5**, if $p_l + p_{l+1} \geq 1$ or $p \geq \frac{1}{2}$, Theorem 9.3.1 is still valid since **P-A3′** implies **P-A3**. Furthermore, under **P-A1**, **P-A2**, **P-A3′**, **P-A4**, **P-A5**, and $p \geq \frac{1}{2}$ we can compare the expected costs incurred when at any instant of time we test *any* two distinct combinations of sensors within a reachable set and we follow the optimal test strategy afterwards. The following theorem precisely states the result.

Theorem 9.3.2 *Let N be an information state, and let i and j be two elements in N. Under Assumptions* **P-A1**, **P-A2**, **P-A3′**, **P-A4**, **P-A5**, *and $p \geq \frac{1}{2}$ the following are true:*

$$\textit{(i) If } |N_i^D| + |N_i^{\overline{D}}| \leq |N_j^D| + |N_j^{\overline{D}}| \tag{9.30}$$

$$\textit{and } |N_i^D| \leq |N_j^D| \leq |N_i^{\overline{D}}| \quad , \quad |N_i^D| \leq |N_j^{\overline{D}}| \leq |N_i^{\overline{D}}| \tag{9.31}$$

$$\textit{then } V_i(N) \leq V_j(N) \tag{9.32}$$

where $V_k(N)$ denotes, as previously, the expected (test) cost incurred when we choose to test $k \in N$ when the information state is N and follow the optimal test strategy afterwards.

(ii) An optimal test strategy for the sensor selection problem under consideration is to test combinations of sensors in increasing order of their cost. Moreover, the minimum expected cost associated with N is

$$V(N) = c * \sum_{n=0}^{|N|-1} (1-p)^n. \tag{9.33}$$

Proof of Theorem 9.3.2 Part (ii) is already proved by Theorem 9.3.1 and the fact that Assumption **P-A3′** implies Assumption **P-A3**. Equation (9.33) is obtained by letting all probabilities equal to p in Equation (9.29). We prove part (i) by induction on the cardinality of the information state N.

- *Basis of induction* Let $|N| = 2$, i.e., $N = \{i, j\}$. There are three possibilities for such information states (based on Definitions 9.3.5 and 9.3.6 of reachable sets):

  1,2: $|i| = |j|$ such that the cost attached to i is less than that attached to j, or $|i| = |j| - 1$ where i is not a child of j. In this case

  $$V(N_i^D) = \emptyset, V(N_i^{\overline{D}}) = \{j\}, V(N_j^D) = \{i\}, \text{ and } V(N_j^{\overline{D}}) = \{i\}. \tag{9.34}$$

  Therefore,

  $$|N_i^D| + |N_i^{\overline{D}}| \leq |N_j^D| + |N_j^{\overline{D}}|, \tag{9.35}$$

  and

  $$|N_i^D| \leq |N_j^D| \leq |N_i^{\overline{D}}| \quad , \quad |N_i^D| \leq |N_j^{\overline{D}}| \leq |N_i^{\overline{D}}|. \tag{9.36}$$

  Moreover,

  $$V_i(N) = c + p * V(N_i^D) + (1-p) * V(N_i^{\overline{D}}) = c + (1-p) * c = c * (2-p), \tag{9.37}$$

  and

  $$V_j(N) = c + p * V(N_j^D) + (1-p) * V(N_j^{\overline{D}}) = c + p * c + (1-p) * c = 2 * c. \tag{9.38}$$

  From (9.37) and (9.38) it follows that $V_i(N) \leq V_j(N)$ and it is optimal to test i next.

  3: $|i| = |j| - 1$ with i being a child of j. In this case

  $$V(N_i^D) = \emptyset, V(N_i^{\overline{D}}) = \{j\}, V(N_j^D) = \{i\}, \text{ and } V(N_j^{\overline{D}}) = \emptyset. \tag{9.39}$$

  Consequently,

  $$|N_i^D| + |N_i^{\overline{D}}| \leq |N_j^D| + |N_j^{\overline{D}}|, \tag{9.40}$$

  and

  $$|N_i^D| \leq |N_j^D| \leq |N_i^{\overline{D}}| \quad , \quad |N_i^D| \leq |N_j^{\overline{D}}| \leq |N_i^{\overline{D}}|. \tag{9.41}$$

Furthermore,

$$V_i(N) = c + p * V(N_i^D) + (1-p) * V(N_i^{\overline{D}}) = c + (1-p) * c = c * (2-p), \tag{9.42}$$

and

$$V_j(N) = c + p * V(N_j^D) + (1-p) * V(N_j^{\overline{D}}) = c + p * c = c * (1+p). \tag{9.43}$$

From (9.42), (9.43) and the assumption that $p \geq \frac{1}{2}$ implying that $(2-p) \leq (1+p)$, we conclude that $V_i(N) \leq V_j(N)$.

- *Induction step* Assume that the assertion of the theorem is true for any set N' with cardinality k, i.e., for any set N' of cardinality less than or equal to k, if i and j are elements in N' such that (9.30) and (9.31) are satisfied and $p \geq \frac{1}{2}$ then (9.32) is true and an optimal test strategy is to test combinations of sensors in increasing order of cost. We need to prove that for any set N of cardinality $k+1$, the assertion of the theorem is still true. Let i, j be elements of N such that (9.30) and (9.31) are satisfied. We need to prove that $V_i(N) \leq V_j(N)$ and an optimal test strategy is to test combinations of sensors in increasing order of cost. To prove that $V_i(N) \leq V_j(N)$ we need to show that

$$\begin{aligned} V_i(N) &= c + p * V(N_i^D) + (1-p) * V(N_i^{\overline{D}}) \leq c + p * V(N_j^D) + (1-p) * V(N_j^{\overline{D}}) \\ &= V_j(N), \end{aligned} \tag{9.44}$$

or

$$p * V(N_i^D) + (1-p) * V(N_i^{\overline{D}}) \leq p * V(N_j^D) + (1-p) * V(N_j^{\overline{D}}). \tag{9.45}$$

Let u, v, r, and s denote, respectively, the cardinality of the sets N_i^D, $N_i^{\overline{D}}$, N_j^D, and $N_j^{\overline{D}}$. By assumption we have that

$$u + v \leq r + s, \text{ and } u \leq r \leq v \ , \ u \leq s \leq v. \tag{9.46}$$

Since u, v, r, and s are all less than or equal to k, we obtain, by the induction hypothesis and (ii)

$$V(N_i^D) \quad = c + (1-p) * \{c + (1-p) * [c + \ldots]\} = \quad c * \sum_{n=0}^{u-1} (1-p)^n,$$

and similarly,

$$\begin{aligned} V(N_i^{\overline{D}}) &= c * \sum_{n=0}^{v-1} (1-p)^n, \\ V(N_j^{D}) &= c * \sum_{n=0}^{r-1} (1-p)^n, \\ V(N_i^{\overline{D}}) &= c * \sum_{n=0}^{s-1} (1-p)^n. \end{aligned} \tag{9.47}$$

Proving Inequality (9.44) reduces to showing that

$$\begin{aligned} p * c * \textstyle\sum_{n=0}^{u-1} (1-p)^n + (1-p) * c * \sum_{n=0}^{v-1} (1-p)^n \leq \\ p * c * \textstyle\sum_{n=0}^{r-1} (1-p)^n + (1-p) * c * \sum_{n=0}^{s-1} (1-p)^n \end{aligned} \tag{9.48}$$

subject to the constraints (9.46). By rearranging terms in (9.48) and using the facts that $u \leq r$ and $s \leq v$, we find that proving (9.44) is equivalent to showing

$$(1-p) * c * \sum_{n=s}^{v-1} (1-p)^n \leq p * c * \sum_{n=u}^{r-1} (1-p)^n. \tag{9.49}$$

By the assumption that $p \geq \frac{1}{2}$ we have that $(1-p) \leq p$. Hence, to show that (9.49) is true it is sufficient to prove that the summation on the left hand side is less than the summation on the right hand side. The left hand side summation of (9.49) has fewer terms than the right hand side summation since $v - s \leq r - u$ (cf. (9.46)). Moreover, the terms in the summation on the left hand side are term by term less than the first $v - s$ terms in the summation on the right hand side: $s \geq u$ implies that $(1-p)^s \leq (1-p)^u$, i.e., the first term in the left hand side summation is smaller than the first term in the right hand side summation, and similarly for the remaining $v - s - 1$ terms. Therefore, we conclude that Inequality (9.49) is always true which implies that (9.44) is always true and this concludes the proof of the induction step for part (i) of the theorem. **Q.E.D.**

Remark For this problem the value function $V(N)$ is upper bounded by $\frac{c}{p}$. This can be seen as follows. First, it is clear that $V(N)$ is a nondecreasing function of the cardinality of the information state N. Since the *a priori* probability that the system possesses Property D when any combination of sensors is used is p, as cardinality$(N) \rightarrow \infty$ the average number of tests required to determine the least costly combination of sensors is $\frac{1}{p}$, according to the optimal test strategy of Theorem 9.3.2. Consequently $V(N)$ cannot exceed $\frac{c}{p}$ for any information state N.

The following example illustrates, in part, Theorem 9.3.2.

Example 9.3.3 *Consider the digraph presented in Figure 9.1 and let* $N = 2^{\Gamma} \setminus \{\Gamma, \emptyset, \{1\}, \{2\}\}$ *be the information state. Assume that the costs of combinations of sensors are ordered as follows*

$$\{3\} < \{4\} < \{2,3\} < \{1,3\} < \{1,2\} < \{3,4\} < \{2,4\} < \{1,4\} <$$

$$< \{1,2,3\} < \{2,3,4\} < \{1,3,4\} < \{1,2,4\}$$

and let $p = 0.6$. *Pick* $i = \{1,2\}$ *and* $j = \{1,3,4\}$. *Then*

$$\begin{aligned}
N_i^D &= \{\{3\},\{4\},\{2,3\},\{1,3\}\},\\
N_i^{\overline{D}} &= \{\{3\},\{4\},\{2,3\},\{1,3\},\{3,4\},\{2,4\},\{1,4\},\{1,2,3\},\{2,3,4\},\\
&\quad \{1,3,4\},\{1,2,4\}\},\\
N_j^D &= \{\{3\},\{4\},\{2,3\},\{1,3\},\{1,2\},\{3,4\},\{2,4\},\{1,4\},\{1,2,3\},\{2,3,4\}\},\\
N_j^{\overline{D}} &= \{\{2,3\},\{1,2\},\{2,4\},\{1,2,3\},\{2,3,4\},\{1,2,4\}\},\\
V_i(N) &= c + p * V(N_i^D) + (1-p) * V(N_i^{\overline{D}}),\\
\textit{and } V_j(N) &= c + p * V(N_j^D) + (1-p) * V(N_j^{\overline{D}}).
\end{aligned}$$

Since the above sets satisfy (9.30) and (9.31) of Theorem 9.3.2 part (i), *we must have* $V_i(N) \leq V_j(N)$. *Indeed, by using (9.47), we obtain*

$$V_i(N) = c + p * c * \sum_{n=0}^{3} (1-p)^n + (1-p) * c * \sum_{n=0}^{10} (1-p)^n = 2.6410 * c,$$

and

$$V_j(N) = c + p * c * \sum_{n=0}^{9} (1-p)^n + (1-p) * c * \sum_{n=0}^{5} (1-p)^n = 2.6689 * c > V_i(N).$$

Both $V_i(N)$ *and* $V_j(N)$ *are greater than the optimal cost which is equal to*

$$c * \sum_{n=0}^{11} (1-p)^n = 1.6666 * c \ \textit{(cf. (9.33))}.$$

Finally, we conclude this section by showing that Assumption **P-A3′** together with Assumptions **P-A1**, **P-A2**, **P-A4**, and **P-A5** lead to an additional property of the value function $V(N)$ stated in Theorem 9.3.3 below. This property relates the cost function of two reachable sets, one being a subset of the other. Note that Corollary 9.3.1 presented a similar property under Assumption **P-A3** provided that the conditions $p_l + p_{l+1} \geq 1$ hold for all l. Under Assumptions **P-A1**, **P-A2**, **P-A3′**, **P-A4**, and **P-A5**, Theorem 9.3.3 is valid for

any value of p. Since no restrictions are imposed on p, the strategy that tests combinations of sensors in increasing order of cost is not necessarily an optimal test strategy. Therefore, Theorem 9.3.3 does not follow directly from the results of Theorems 9.3.1 and 9.3.2. Moreover, as presented in the proof, the arguments needed to establish Theorem 9.3.3 are different from those used in the proofs of Theorems 9.3.1 and 9.3.2.

Theorem 9.3.3 *Consider the sensor selection problem under Assumptions* **P-A1**, **P-A2**, **P-A3′**, **P-A4**, *and* **P-A5**. *Let A and B be two reachable sets in the digraph. If $A \subseteq B$ then $V(A) \leq V(B)$.*

Proof of Theorem 9.3.3 We prove this Theorem by induction on the number of elements in A.

- Basis of induction: Let $A = \{a\}$, $|B| = 1 + k$, $k \geq 0$. We have $V(\{a\}) = c$, and $V(\{B\}) \geq \mathrm{c} = V(\{a\}) = V(A)$.
- Induction step: Assume that $V(A) \leq V(B)$ for any two sets A and B such that $A \subseteq B$, $|A| = n$, $|B| = n + k$, for any $k \geq 0$. Pick any A', B' such that $A' \subseteq B'$, $|A'| = n + 1$, $|B'| = n + k + 1$. We need to show that $V(A') \leq V(B')$. Pick an optimal decision at state B', say x. Depending on whether $x \in A'$ or not there are two cases.
  - Case 1: $x \in A'$. The choice x may or may not be an optimal decision for A'. Therefore,

$$V(A') \leq c + p * V(A'^{D}_{x}) + (1 - p) * V(A'^{\overline{D}}_{x}), \tag{9.50}$$

and

$$V(B') = c + p * V(B'^{D}_{x}) + (1 - p) * V(B'^{\overline{D}}_{x}). \tag{9.51}$$

Furthermore,

$$A'^{D}_{x} \subseteq B'^{D}_{x}, |A'^{D}_{x}| \leq n, |B'^{D}_{x}| \leq n + k, \tag{9.52}$$

and

$$A'^{\overline{D}}_{x} \subseteq B'^{\overline{D}}_{x}, |A'^{\overline{D}}_{x}| \leq n, |B'^{\overline{D}}_{x}| \leq n + k, \tag{9.53}$$

by the properties of reachable sets and the fact that $A' \subseteq B'$. Therefore, by the induction hypothesis

$$V(A'^{D}_{x}) \leq V(B'^{D}_{x}), \tag{9.54}$$

and

$$V(A_x'^{\overline{D}}) \leq V(B_x'^{\overline{D}}), \tag{9.55}$$

which, together with (9.50), (9.51), result in

$$V(A') \leq c + p * V(B_x'^{D}) + (1-p) * V(B_x'^{\overline{D}}) = V(B'). \tag{9.56}$$

– Case 2: $x \notin A'$. Here we have two sub-cases:

  * Case 2-i: x or one of its parents was tested (before reaching state A') and proved not to possess Property D, i.e., x possesses property $\overline{D}$. Pick for A' a decision element, say x', such that x' is the lowest cost combination of sensors in A'. Since x' may not be the optimal decision for A',

$$V(A') \leq c + p * V(A_{x'}'^{D}) + (1-p) * V(A_{x'}'^{\overline{D}}); \tag{9.57}$$

furthermore, since x is an optimal decision at B' Equation (9.51) holds. By assumption A' does not contain any child of x, which implies that $A_{x'}'^{\overline{D}}$ does not contain any child of x ($A_{x'}'^{\overline{D}} \subseteq A' \subseteq B'$), but $B_x'^{\overline{D}}$ contains all elements in B' except x and its children. Therefore,

$$A_{x'}'^{\overline{D}} \subseteq B_x'^{\overline{D}}. \tag{9.58}$$

In addition

$$A_{x'}'^{D} = \emptyset \subseteq B_x'^{D}, \tag{9.59}$$

since x' is the lowest cost combination of sensors in A'. By the induction hypothesis, since

$$|A_{x'}'^{\overline{D}}| \leq n, |B_x'^{\overline{D}}| \leq n+k, |A_{x'}'^{D}| \leq n, \text{ and } |B_x'^{D}| \leq n+k, \tag{9.60}$$

we obtain

$$V(A_{x'}'^{\overline{D}}) \leq V(B_x'^{\overline{D}}), \tag{9.61}$$

and

$$V(A_{x'}'^{D}) \leq V(B_x'^{D}). \tag{9.62}$$

From (9.51), (9.57), (9.61), and (9.62) it follows that

$$V(A') \leq c + p * V(B_x'^{D}) + (1-p) * V(B_x'^{\overline{D}}) = V(B'). \tag{9.63}$$

* Case 2-ii: x was tested (before reaching state A') and proved to possess Property D. This implies that A' only contains elements whose number of sensors is strictly less than that of x in addition to those elements with the same number of sensors as x but with a lower cost. This case may be further divided into two sub-cases.

  · Case 2-ii-a: A' contains at least one child of x. Pick the child of x in A' having the highest number of sensors as a decision for A', say x''. Since x'' may or may not be an optimal decision at state A',

$$V(A') \leq c + p * V(A'^{D}_{x''}) + (1-p) * V(A'^{\overline{D}}_{x''}). \tag{9.64}$$

  Furthermore, since x is an optimal decision at state B Equation (9.51) holds. By the choice of x'' it follows that

$$A'^{D}_{x''} \subseteq B'^{D}_{x}, \tag{9.65}$$

  since $A'^{D}_{x''} \subseteq A' \subseteq B'$ and $A'^{D}_{x''}$ has only elements having number of sensors less than or equal that of x'', and consequently strictly less than that of x. By definition,

$$A'^{\overline{D}}_{x''} = A' \backslash \{x'' \text{ and its children}\}, \tag{9.66}$$

  and

$$B'^{\overline{D}}_{x} = B' \backslash \{x \text{ and its children}\}, \tag{9.67}$$

  and since $A' \subseteq B'$, and x'' is the child of x in A' with the maximum number of sensors,

$$A'^{\overline{D}}_{x''} \subseteq B'^{\overline{D}}_{x}. \tag{9.68}$$

  By the induction hypothesis, since

$$|A'^{\overline{D}}_{x''}| \leq n, |B'^{\overline{D}}_{x}| \leq n+k, |A'^{D}_{x''}| \leq n, \text{ and } |B'^{D}_{x}| \leq n+k, \tag{9.69}$$

  we obtain

$$V(A'^{\overline{D}}_{x''}) \leq V(B'^{\overline{D}}_{x}), \tag{9.70}$$

  and

$$V(A'^{D}_{x''}) \leq V(B'^{D}_{x}). \tag{9.71}$$

  Consequently, from (9.51), (9.64), (9.70), and (9.71) we conclude that

$$V(A') \leq c + p * V(B'^{D}_{x}) + (1-p) * V(B'^{\overline{D}}_{x}) = V(B'). \tag{9.72}$$

- Case 2-ii-b: A' contains no children of x. Pick the element in A' that has the highest cost, say x'''. Since x''' may or may not be an optimal decision,

$$V(A') \leq c + p * V(A'^{D}_{x'''}) + (1-p) * V(A'^{\overline{D}}_{x'''}). \tag{9.73}$$

Furthermore, since x is an optimal decision at state B Equation (9.51) holds. The choice of x''' implies that

$$A'^{D}_{x'''} = A' \backslash \{x'''\}, \tag{9.74}$$

i.e., $A'^{D}_{x'''}$ only contains elements in A' that have cost strictly lower than that of x'''. Moreover, by assumption x does not belong to A', and it was tested before reaching A' and proved to possess Property D; hence, all elements in A' and consequently all elements in $A'^{D}_{x'''}$ have strictly lower cost than that of x. Furthermore, B'^{D}_{x} has all of its elements as those elements in B' whose cost is strictly less than that of x. Therefore, since $A' \subseteq B'$, it follows that

$$A'^{D}_{x'''} \subseteq B'^{D}_{x}. \tag{9.75}$$

In addition,

$$A'^{\overline{D}}_{x'''} \subseteq B'^{\overline{D}}_{x}, \tag{9.76}$$

since $A'^{\overline{D}}_{x'''} \subseteq A' \subseteq B'$ and the only elements missing from B' in $B'^{\overline{D}}_{x}$ are x and its children, which are not in $A'^{\overline{D}}_{x'''}$ by the assumption that A' contains no children of x. By the induction hypothesis, since

$$|A'^{\overline{D}}_{x'''}| \leq n, |B'^{\overline{D}}_{x}| \leq n+k, |A'^{D}_{x'''}| \leq n, \text{ and } |B'^{D}_{x}| \leq n+k, \tag{9.77}$$

we obtain

$$V(A'^{\overline{D}}_{x'''}) \leq V(B'^{\overline{D}}_{x}), \tag{9.78}$$

and

$$V(A'^{D}_{x'''}) \leq V(B'^{D}_{x}). \tag{9.79}$$

From (9.51), (9.73), (9.78), and (9.79) we obtain

$$V(A') \leq c + p * V(B'^{D}_{x}) + (1-p) * V(B'^{\overline{D}}_{x}) = V(B'). \tag{9.80}$$

The proof of the induction step is complete, and this completes the proof of Theorem 9.3.3.

Q.E.D.

9.4 Concluding Remarks

We have presented conditions sufficient to guarantee that the strategy that tests combinations of sensors in increasing order of cost is optimal for **Problem P**. This result is intuitive since it is assumed that a combination of two or more sensors is equally or more likely to result in a system possessing Property D than a system possessing Property $\overline{D}$ and the cost of testing for Property D is independent of the cardinality of the combination of sensors that is being used.

Assumptions **P-A2** – **P-A5** as well as $p_l + p_{l+1} \geq 1$, $l = 1, 2, \ldots, M-2$, are crucial in establishing the result of Theorem 9.3.1. A brief critique of these assumptions was presented in Section 9.2. We add a few more remarks that further reveal the importance of these assumptions. We have already noted that Assumption **P-A2** is reasonable but it may not always be true. Relaxation of Assumption **P-A2** may drastically change the structure of the optimization problem because ordering of sensor combinations according to their cost is not the same as the ordering according to their cardinality. Therefore, it may be unlikely that the strategy described in Theorem 9.3.1 will be optimal; even if the same strategy were optimal, a proof of its optimality would require arguments different from those employed in the proof of Theorem 9.3.1. If the sensors' information gathering ability drastically differs from sensor to sensor, then Assumption **P-A3** may not be true. In this case, an optimal test strategy for the sensor selection problem may be different from that of Theorem 9.3.1 and arguments significantly different from those presented in Theorem 9.3.1 may be required to discover the nature of an optimal test strategy. The conditions $p_l + p_{l+1} \geq 1$, $l = 1, 2, \ldots, M-2$, are essential for the validity of the result of Theorem 9.3.1; the plausibility of the main result under these conditions has already been discussed. Finally, Assumption **P-A3′** is stronger than **P-A3**. This is why when Assumption **P-A3′** is used instead of **P-A3** we obtain more structural results about **Problem P**. Specifically, under Assumptions **P-A1**–**P-A5** we have proved the optimality of the strategy that tests combinations of sensors in increasing order of cost, but have been unable to compare any other policies; on the contrary, under Assumptions **P-A1**, **P-A2**, **P-A3′**, **P-A4**, and **P-A5** we can compare policies that satisfy the conditions of Theorem 9.3.2.

We conclude by noting that although the methodology presented in this paper was motivated by the failure diagnosis and hypothesis testing examples, it can be used in other situations as long as the property to be tested exhibits the same characteristics as Property D, namely **C3** and **C4**. For example, in the field of discrete event systems, the properties of controllability, observability and normality possess **C3** and **C4** (see [7]).

CHAPTER 10

CONCLUSION

We summarize the main contributions of this thesis; we also compare the work presented in this thesis to other approaches for failure diagnosis with decentralized information that were mentioned in Chapter 1; and we present some directions for future research.

10.1 Summary of Contributions

In this thesis we have addressed two classes of problems: (i) failure diagnosis of DES with decentralized information; and (ii) a sensor selection problem.

Our main contributions on failure diagnosis with decentralized information are

- the definition of diagnosability for decentralized systems within the context of a coordinated decentralized architecture,
- the design of a set of diagnostic protocols that highlight the "performance vs. complexity" tradeoff,
- a formal analysis of the diagnostic properties of the suggested protocols,
- the solution of a non-linear decentralized estimation problem (although for the purpose of failure diagnosis, one need not necessarily solve the estimation problem to diagnose the system),
- a systematic procedure to account for communication delays that may be encountered during the implementation of the diagnostic protocols,
- the application of the diagnostic methodology to a wireless local area network.

We have formulated an optimization problem in sensor selection as a Markovian decision problem and identified one instance where the optimal solution can be analytically determined.

10.2 Related Work

We provide a brief comparison of our work on failure diagnosis with decentralized information with some related work in the literature, namely [3], [12], [30], [43][1]. Although similarities exist between our approach and the approaches presented in the aforementioned references, the reader needs to be aware that the modeling and informational assumptions of our work and theirs' are clearly different. Nonetheless, we will attempt to do a fair comparison on the following issues: the models used, the definition of diagnosability, the analysis of diagnostic properties, and the effect of communication delays.

- Local models vs. global models

  In our framework, global models were used to generate diagnosers at the local sites. Global models are needed in order for the communication of state information to be meaningful; otherwise, local sites would not be able to understand state information that is not based on their model of the system. This fact is also recognized in [43]. In contrast, [3], [12], and [30] use local models in their framework. However, additional information is needed to overcome the abovementioned problem. For instance, local sites in the work of [3], [12], and [30] are aware which events (that may be received from other local sites) affect their diagnostic information. Moreover, a global history of the occurrence of events is kept to generate the diagnostic decision. Therefore, although local models are used by design, additional information is needed to eliminate illegal behavior, that is, to reconstruct what is consistent with the global behavior.

- Diagnosability

  With the exception of [43], no other approach could be found which has formally defined diagnosability. This is a very critical point since one needs to understand what is a diagnosable system before attempting to diagnose it! We have provided a definition of diagnosability within the context of a coordinated decentralized architecture. Our definition is architecture-dependent. It may be a little bit restrictive for a definition to rely on a specific architecture, however, to the best of our knowledge, there are no general definitions of diagnosability that are architecture-independent. Therefore, we believe that our approach represents a step in the right direction. The same remark applies to the work of [43].

- Diagnostic properties

[1] We note that the philosophy of the approach we are suggesting is totally different from that of [21] and [28], hence they will not be referred to in the comparison.

We have designed diagnostic protocols and analyzed their diagnostic properties. We have also provided tests that run in finite time to check these properties. This is desirable for real applications: before implementing a diagnostic tool, one is capable of checking whether each and every failure of interest can be detected or not. This aspect is lacking in the other approaches. For instance, in [30], if a diagnostic decision cannot be made within a given period, the procedure is to inform the operator that is overseeing the operation of the system, that a failure may have occurred. The work of [43] does provide necessary and sufficient conditions for diagnosability, however, the authors do not provide a test to check these conditions.

- Communication delays

  Dealing with communication delays is important since global order cannot be guaranteed in decentralized systems. We have dealt with the issue of communication delays by suggesting an ordering algorithm (of messages) without the use of timing information. The ordering algorithm assumes some bounds on communication delays. Mechanisms similar to ours have been used in [3] and [30]. The work of [12] does not address delays, rather it assumes that global order is preserved.

Related sensor selection problems have been previously studied in [5, 20, 10] in the context of discrete event systems. The problems in [5], [20], [10] as well as the problem investigated in this thesis have a similar objective of determining an optimal (according to some criterion) set of sensors or events with respect to which the system possesses a certain given property. A major difference, however, between [5, 20, 10] and our work is that we also determine a sequence of tests that minimizes the computational effort required to identify a least costly set of sensors that results in a system possessing the required property; this issue is not addressed in [5, 20, 10].

10.3 Future Research Directions

Researchers began addressing the problem of failure diagnosis with decentralized information recently. Therefore, many interesting and challenging problems began to surface as of late. We will discuss a few directions in which the work in this thesis may be extended.

In Section 3.3, we mentioned that the definition of a protocol could include the partitioning of observable events as a design parameter. By doing so, one could eliminate traces in the language that are ambiguous, failure-ambiguous, or fully-ambiguous. In fact, the conditions under which Protocols 1D, 2 (as well as all its modification presented in Section 5.6), and 3 perform as well as the centralized diagnoser depend on the partitioning of

observable events at the local sites. One way to handle this problem is to attempt to modify the partitioning of observable events in a way that these protocols will perform as well as the centralized diagnoser, or in other words to get rid off ambiguous, failure-ambiguous, and fully-ambiguous traces. This problem is quite interesting and challenging. The issue is to develop algorithms that find partitioning of observable events under which the aforementioned traces do not exist. In addition to that, one could set up an optimization problem to find the "optimal partitioning" among all other possible partitioning that do not result in ambiguous, failure-ambiguous, or fully-ambiguous traces. Optimality could be defined with respect to the total cost associated with the observable events or the cardinality of the intersection of the sets of observable events or some other physical constraints related to the distribution of the sensors (in charge of observing the events).

Two possible extensions of the thesis have already been discussed in Section 7.4 but it is worth mentioning them again. The first one is that of designing protocols that do not mimic a centralized diagnostic approach at the local sites when realizing the coordinated decentralized architecture. The second one is that of finding upper bounds on the performance of the architecture under some imposed constraints on memory, processing power and communication. These are indeed interesting open problems that have not been addressed to date to the best of our knowledge.

Other issues of interest relate to the architecture itself. The architecture depicted in Figure 3.1 reflects a hierarchical organization architecture, where communication is initiated from top to bottom, that is local sites only communicate to the coordinator. One could consider another architecture where there is no coordinator and the local sites exchange information with each other. In such a case, the objective would be to design protocols to support communication among the sites, and diagnostic protocols, so that the local sites jointly achieve the same performance as a centralized diagnoser would. The issue of minimization of communicated messages should be an important criterion, if not the most important criterion to consider when designing the protocols.

As discussed in Section 10.2, we have so far used global models to generate diagnostic information at local sites. In most large scale decentralized systems, local sites may have partial knowledge about the global system, or even when they have complete knowledge of the global system, it may be infeasible to use it. In such a case local sites are forced to generate their diagnostic information based on local models. Local models may represent the operation of subsystems, the synchronization of which constitute the operation of the global system. The key difficulty here, as discussed previously in Section 10.2, is that local sites cannot exchange/communicate state information, since they have partial knowledge about the overall system. The objective would be to design communication and/or diagnostic

protocols to achieve the same performance as a centralized diagnoser, bearing in mind the aforementioned difficulty. If that is not feasible, one would like to identify conditions on the system structure for theses protocols to perform as well as a centralized diagnoser.

Another issue worth of investigation is that of transient faults. This remark also applies to the methodology of [40]. Transient failures are failures that may occur and either disappear later on or are automatically corrected. The work we have presented as well as the work of [40], assumes that failures are permanent. Permanent failures result in a change in the mode of operation of the system, which remains in the faulty mode up until a failure recovery is initiated. However, many systems exhibit transient failures, for instance, telecommunication networks [36]. A modification of our approach would be needed in order to account for transient failures. The objective would not only be to infer the occurrences of failures using the record of observations, but also to detect when these failures disappear or are not present anymore.

Finally, we conclude this chapter with a discussion on two of issues related to the problem of sensor selection that we addressed in Chapter 9. The assumptions under which we investigated the sensor selection problem (cf. Section 9.2) as well as the sufficient conditions presented in Theorem 9.3.1 are indeed crucial in establishing our results. Relaxing the assumption on sensor cost may drastically change the structure of the optimization problem since ordering of sensor combinations according to their cost is not the same as ordering them according to their cardinality. Moreover, the techniques used in proving Theorem 9.3.1 will not hold if the assumption that a priori probabilities are an increasing function in the cardinality of the combination of sensors used is violated. In both cases, it is unlikely that the strategy described by Theorem 9.3.1 will be optimal. It may be the case that an index policy could be an optimal strategy; the index could be a function of both the cost of the combination of sensors used and the a priori probability that the system is diagnosable with respect to these sensors.

We should emphasize that the a priori probabilities were fixed during all stages of the optimization problem (cf. Assumption **P-A4**). In fact this may not always be true: a test on a combination of sensors, whether it results in a diagnosable or non-diagnosable system, could affect the a priori probabilities of other combinations of sensors with respect to which the system has not been tested for diagnosability. In such a case the probabilities are dynamically changing and this in turn changes the structure of the optimization problem. We do not have any clear idea about the nature of an optimal policy in such a case.

The two research problems presented above are challenging open problems and worth of further investigation.

APPENDIX A

A.1 Transition Table for the LAN Model of Chapter 8

State	Event	δ(State,Event)	State	Event	δ(State,Event)
1	$sync$	2	25	$l:m$	26
	ltf	17	26	$f1:r:l$	27
	lrf	21	27	$f2:r:l$	28
	$f1rf$	35	28	t	29
	$f1tf$	45	29	$f1:m$	30
	$f2tf$	58	30	$f2:r:f1$	31
	$f2rf$	71	31	t	32
2	$f1:r:s$	3	32	$f2:m$	33
3	$f2:r:s$	4	33	$f1:r:f2$	34
4	t	5	34	t	21
5	$l:m$	6	35	$sync$	36
6	$f1:r:l$	7	36	$f2:r:s$	37
7	$f2:r:l$	8	37	t	38
8	t	9	38	$l:m$	39
9	$f1:m$	10	39	$f2:r:l$	40
10	$l:r:f1$	11	40	t	41
11	$f2:r:f1$	12	41	t	42
12	t	13	42	$f2:m$	43
13	$f2:m$	14	43	$l:r:f2$	44
14	$l:r:f2$	15	44	t	35
15	$f1:r:f2$	16	45	$sync$	46
16	t	1	46	$f1:r:s$	47
17	t	18	47	$f2:r:s$	48
18	t	19	48	t	49
19	t	20	49	$l:m$	50
20	t	17	50	$f1:r:l$	51
21	$sync$	22	51	$f2:r:l$	52
22	$f1:r:s$	23	52	t	53
23	$f2:r:s$	24	53	t	54
24	t	25	54	$f2:m$	55

State	Event	δ(State,Event)	State	Event	δ(State,Event)
55	$l:r:f2$	56	68	$f2:r:f1$	69
56	$f1:r:f2$	57	69	t	70
57	t	45	70	t	58
58	$sync$	59	71	$sync$	72
59	$f1:r:s$	60	72	$f1:r:s$	73
60	$f2:r:s$	61	73	t	74
61	t	62	74	$l:m$	75
62	$l:m$	63	75	$f1:r:l$	76
63	$f1:r:l$	64	76	t	77
64	$f2:r:l$	65	77	$f1:m$	78
65	t	66	78	$l:r:f1$	79
66	$f1:m$	67	79	t	80
67	$l:r:f1$	68	80	t	71

A.2 Transition Table for the Diagnoser G_d of Chapter 8

State	State Components	Event	δ_d(State,Event)
1	1N	$f1:r:s$	2
		$f2:r:s$	3
		t	4
2	3N,23F2,47F4,65F5,73F6	$f2:r:s$	5
		t	6
3	37F3	t	7
4	18F1	t	8
5	4N,24F2,48F4,61F5	t	9
6	74F6	$f1:r:l$	10
7	38F3	$f2:r:l$	11
8	19F1	t	12
9	5N,25F2,49F4,62F5	$f1:r:l$	13
10	76F6	t	14
11	40F3	t	15
12	20F1	t	16
13	7N,27F2,51F4,64F5	$f2:r:l$	17
14	77F6	$l:r:f1$	18
15	41F3	t	19
16	17F1	t	4
17	8N,28F2,52F4,65F5	t	20
18	79F6	t	21
19	42F3	$l:r:f2$	22
20	9N,29F2,53F4,66F5	t	23
		$l:r:f1$	24
		$f2:r:f1$	25
21	80F6	t	26

State	State Components	Event	δ_d(State,Event)
22	44F3	t	27
23	54F4	$l:r:f2$	28
24	11N,68F5	$f2:r:f1$	29
25	31F2	t	30
26	71F6	$f1:r:s$	31
27	35F3	$f2:r:s$	3
28	56F4	$f1:r:f2$	32
29	12N,69F5	t	33
30	32F2	$f1:r:f2$	34
31	73F6	t	6
32	57F4	t	35
33	13N,70F5	t	36
		$l:r:f2$	37
34	34F2	t	38
35	45F4	$f1:r:s$	39
36	58F5	$f1:r:s$	40
37	15N	$f1:r:f2$	41
38	21F2	$f1:r:s$	42
39	47F4	$f2:r:s$	43
40	60F5	$f2:r:s$	44
41	16N	t	1
42	23F2	$f2:r:s$	45
43	48F4	t	46
44	61F5	t	47
45	24F2	t	48
46	49F4	$f1:r:l$	49
47	62F5	$f1:r:l$	50
48	25F2	$f1:r:l$	51
49	51F4	$f2:r:l$	52
50	64F5	$f2:r:l$	53
51	27F2	$f2:r:l$	54
52	52F4	t	55
53	65F5	t	56
54	28F2	t	57
55	53F4	t	23
56	66F5	$l:r:f1$	58
57	29F2	$f2:r:f1$	25
58	68F5	$f2:r:f1$	59
59	69F5	t	60
60	70F5	t	36

A.3 Transition Table for the Diagnoser G_{d1} of Chapter 8

State	State Components	Event	δ_{d1}(State,Event)
1	1N	t	2
2	5N,18F1,25F2,38F3,49F4,62F5,74F6	t	3
3	9N,19F1,29F2,41F3,53F4,66F5,77F6	t	4
		$l:r:f1$	5
4	20F1,32F2,42F3,54F4	t	6
		$l:r:f2$	7
5	11N,68F5,79F6	t	8
6	17F1,21F2	t	9
7	44F3,56F4	t	10
8	13N,70F5,80F6	t	11
		$l:r:f2$	12
9	18F1,25F2	t	13
10	35F3,45F4	t	14
11	58F5,71F6	t	15
12	15N	t	1
13	19F1,29F2	t	16
14	38F3,49F4	t	17
15	62F5,74F6	t	18
16	20F1,32F2	t	6
17	41F3,53F4	t	19
18	66F5,77F6	$l:r:f1$	20
19	42F3,54F4	$l:r:f2$	7
20	68F5,79F6	t	21
21	70F5,80F6	t	11

A.4 Transition Table for the Diagnoser G_{d2} of Chapter 8

State	State Components	Event	δ_{d2}(State,Event)
1	1N	$f1:r:s$	2
		t	3
2	3N,23F2,47F4,60F5,73F6	t	4
3	18F1,38F3	t	5
4	5N,25F2,49F4,62F5,74F6	$f1:r:l$	6
5	19F1,41F3	t	7
6	7N,27F2,51F4,64F5,76F6	t	8
7	20F1,42F3	t	9
8	9N,29F2,53F4,66F5,77F6	t	10
9	17F1,35F3	t	3
10	13N,32F2,54F4,70F5,80F6	t	11
		$f1:r:f2$	12

State	State Components	Event	δ_{d2}(State,Event)
11	58F5,71F6	$f1:r:s$	13
12	16N,34F2,57F4	t	14
13	60F5,73F6	t	15
14	1N,21F2,45F4	$f1:r:s$	2
		t	3
15	62F5,74F6	$f1:r:l$	16
16	64F5,76F6	t	17
17	66F5,77F6	t	18
18	70F5,80F6	t	11

A.5 Transition Table for the Diagnoser G_{d3} of Chapter 8

State	State Components	Event	δ_{d3}(State,Event)
1	1N	$f2:r:s$	2
		t	3
2	4N,24F2,37F3,48F4 ,61F5	t	4
3	18F1,74F6	t	5
4	5N,25F2,38F3,49F4,62F5	$f2:r:l$	6
5	19F1,77F6	t	7
6	8N,28F2,40F3,52F4,65F5	t	8
7	20F1,80F6	t	9
8	9N,29F2,41F3,53F4,66F5	t	10
		$f2:r:s$	11
9	17F1,71F6	t	3
10	42F3,54F4	t	12
11	12N,31F2,69F5	t	13
12	35F3,45F4	$f2:r:s$	14
13	13N,32F2,70F5	t	15
14	37F3,48F4	t	16
15	1N,21F2,58F5	$f2:r:s$	2
		t	3
16	38F3,49F4	$f2:r:l$	17
17	40F3,52F4	t	18
18	41F3,53F4	t	10

A.6 Transition Table for G_{test3} of Chapter 8

$G_{test3} = G_{d1} \parallel G_{d2} \parallel G_{d3} \parallel G_d$, and a state of G_{test3} following the observation of a trace $s \in L(G)$ is of the form $(p_1; p_2; p_3; p)$ where p_1, p_2, p_3, and p are the states of diagnosers G_{d1} (presented in A.3), G_{d2} (presented in A.4), G_{d3} (presented in A.5), and G_d (presented in A.2) , respectively, following the observation of $P_1(s)$, $P_2(s)$, $P_3(s)$, and $P(s)$, respectively. Within the "State Component" column, the state numbers of the diagnosers will be used

instead of the diagnosers' state components for brevity and clarity.

State	State Components	Event	$\delta_{G_{d1}\|\|G_{d2}\|\|G_{d3}\|\|G_d}$(State,Event)
1	1;1;1;1	t	2
		$f1:r:s$	3
		$f2:r:s$	4
2	2;3;3;4	t	5
3	1;2;1;2	t	6
		$f2:r:s$	7
4	1;1;2;3	t	8
5	3;5;5;8	t	9
6	2;4;3;6	$f1:r:l$	10
7	1;2;2;5	t	11
8	2;3;4;7	$f2:r:l$	12
9	4;7;7;12	t	13
10	3;8;5;14	t	14
11	2;4;4;9	$f1:r:l$	15
12	2;3;6;11	t	16
13	6;9;9;16	t	17
14	3;8;5;14	$l:r:f1$	18
15	2;6;4;13	$f2:r:l$	19
16	3;5;8;15	t	20
17	9;3;3;4	t	21
18	5;8;5;18	t	22
19	2;6;6;17	t	23
20	4;7;10;19	$l:r:f2$	24
21	13;5;5;8	t	25
22	8;10;7;21	t	26
23	3;8;8;20	t	27
		$l:r:f1$	28
		$f2:r:f1$	29
24	7;7;10;22	t	30
25	16;7;7;12	t	13
26	11;11;9;26	$f1:r:s$	31
27	4;10;10;23	$l:r:f2$	32
28	5;8;8;24	$f2:r:f1$	33
29	3;8;11;25	t	34
30	10;9;12;27	$f2:r:s$	35
31	11;13;9;31	t	36
32	7;10;10;28	$f1:r:f2$	37
33	5;8;11;29	t	38
34	4;10;13;30	$f1:r:f2$	39
35	10;9;14;3	t	40
36	15;15;3;6	$f1:r:l$	41

State	State Components	Event	$\delta_{G_{d1}\|\|G_{d2}\|\|G_{d3}\|\|G_d}$(State,Event)
37	7;12;10;32	t	42
38	8;10;13;33	t	43
		$l:r:f2$	44
39	4;12;13;34	t	45
40	14;3;16;7	$f2:r:l$	46
41	15;16;3;10	t	47
42	10;14;12;35	$f1:r:s$	48
43	11;11;15;36	$f1:r:s$	49
44	12;10;13;37	$f1:r:f2$	50
45	6;14;15;38	$f1:r:s$	51
46	14;3;17;11	t	52
47	18;17;5;14	$l:r:f1$	53
48	10;2;12;39	$f2:r:s$	54
49	11;13;15;40	$f2:r:s$	55
50	12;12;13;41	t	56
51	6;2;15;42	$f2:r:s$	57
52	17;5;18;15	t	58
53	20;17;5;18	t	59
54	10;2;14;43	t	60
55	11;13;2;44	t	61
56	1;14;15;1	t	2
		$f1:r:s$	62
		$f2:r:s$	63
57	6;2;2;45	t	64
58	19;7;10;19	$l:r:f2$	24
59	21;18;7;21	t	26
60	14;4;16;46	$f1:r:l$	65
61	15;15;4;47	$f1:r:l$	66
62	1;2;15;2	t	6
		$f2:r:s$	7
63	1;14;2;3	t	8
64	9;4;4;48	$f1:r:l$	67
65	14;6;16;49	$f2:r:l$	68
66	15;16;4;50	$f2:r:l$	69
67	9;6;4;51	$f2:r:l$	70
68	14;6;17;52	t	71
69	15;16;6;53	t	72
70	9;6;6;54	t	73
71	17;8;18;55	t	74
72	18;17;8;56	$l:r:f1$	75
73	13;8;8;57	$f2:r:f1$	76
74	19;10;10;23	$l:r:f2$	32
75	20;17;8;58	$f2:r:f1$	77

State	State Components	Event	$\delta_{G_{d1}\|\|G_{d2}\|\|G_{d3}\|\|G_d}$(State,Event)
76	13;8;11;25	t	78
77	20;17;11;59	t	79
78	16;10;13;30	$f1:r:f2$	80
79	21;18;13;60	t	43
80	16;12;13;34	t	45

A.7 Transition Table for G_{test2} of Chapter 8

A state of G_{test2} following the observation of a trace $s \in L(G)$ is of the form $(p_1; p_2; p_3; p; p_c)$ where p_1, p_2, p_3, and p are the states of diagnosers G_{d1} (presented in A.3), G_{d2} (presented in A.4), G_{d3} (presented in A.5), and G_d (presented in A.2) , respectively, following the observation of $P_1(s)$, $P_2(s)$, $P_3(s)$, and $P(s)$, respectively, and the component p_c is the corresponding state of the coordinator after applying the information update rule to the states (and unobservable reaches) of the diagnosers. Within the "State Component" column, the components of the states of the diagnosers G_{d1}, G_{d2}, and G_{d3} will be replaced by the diagnosers' state numbers for brevity and clarity, and only the components of the state of the diagnoser G_d and the corresponding coordinator state p_c will be shown.

State	State Components	Event	δ_g(State,Event)
1	1:1;1;1N;1N	t	2
		$f1:r:s$	3
		$f2:r:s$	4
2	2;3;3;18F1;18F1	t	5
3	1;2;1;3N,23F2,47F4,60F5,73F6; 3N,23F2,47F4,60F5,73F6	t	6
		$f2:r:s$	7
4	1;1;2;37F3;37F3	t	8
5	3;5;5;19F1;19F1	t	9
6	2;4;3;74F6;74F6	$f1:r:l$	10
7	1;2;2;4N,24F2,48F4,61F5;4N,24F2,48F4,61F5	t	11
8	2;3;4;38F3;38F3	$f2:r:l$	12
9	4;7;7;20F1;20F1	t	13
10	2;6;3;76F6;76F6	t	14
11	2;4;4;5N,25F2,49F4,62F5;5N,25F2,49F4,62F5	$f1:r:l$	15
12	2;3;6;40F3;40F3	t	16
13	6;9;9;17F1;17F1	t	17
14	3;8;5;77F6;77F6	$l:r:f1$	18
15	2;6;4;7N,27F2,51F4,64F5;7N,27F2,51F4,64F5	$f2:r:l$	19
16	3;5;8;41F3;41F3	t	20
17	9;3;3;18F1;18F1	t	21

State	State Components	Event	δ_g(State,Event)
18	5;8;5;79F6;79F6	t	22
19	2;6;6;8N,28F2,52F4,65F5;8N,28F2,52F4,65F5	t	23
20	4;7;10;42F3;42F3	$l:r:f2$	24
21	13;5;5;19F1;19F1	t	25
22	8;10;7;80F6;80F6	t	26
23	3;8;8;9N,29F2,53F4,66F5;9N,29F2,53F4,66F5	t	27
		$l:r:f1$	28
		$f2:r:f1$	29
24	7;7;10;44F3;44F3	t	30
25	16;7;7;20F1;20F1	t	13
26	11;11;9;71F6;71F6	$f1:r:s$	31
27	4;10;10;54F4;54F4	$l:r:f2$	32
28	5;8;8;11N,68F5;11N,68F5	$f2:r:f1$	33
29	3;8;11;31F2;31F2	t	34
30	10;9;12;35F3;35F3	$f2:r:s$	35
31	11;13;9;73F6;73F6	t	36
32	7;10;10;56F4;56F4	$f1:r:f2$	37
33	5;8;11;12N,69F5;12N,69F5	t	38
34	4;10;13;32F2;32F2	$f1:r:f2$	39
35	10;9;14;37F3;37F3	t	40
36	15;15;3;74F6;74F6	$f1:r:l$	41
37	7;12;10;57F4;57F4	t	42
38	8;10;13;13N,70F5;13N,70F5	t	43
		$l:r:f2$	44
39	4;12;13;34F2;34F2	t	45
40	14;3;16;38F3;38F3	$f2:r:l$	46
41	15;16;3;76F6;76F6	t	47
42	10;14;12;45F4;45F4	$f1:r:s$	48
43	11;11;15;58F5;58F5	$f1:r:s$	49
44	12;10;13;15N;15N	$f1:r:f2$	50
45	6;14;15;21F2;21F2	$f1:r:s$	51
46	14;3;17;40F3;40F3	t	52
47	18;17;5;77F6;77F6	$l:r:f1$	53
48	10;2;12;47F4;47F4	$f2:r:s$	54
49	11;13;15;60F5;60F5,73F6*	$f2:r:s$	55
50	12;12;13;16N;16N	t	56
51	6;2;15;23F2;23F2	$f2:r:s$	57
52	17;5;18;41F3;41F3	t	58
53	20;17;5;79F6;79F6	t	59
54	10;2;14;48F4;48F4	t	60
55	11;13;2;61F5;61F5	t	61

* The only ambiguous state in G_{test2}, that is, a state where p and p_c do not share the same failure properties.

State	State Components	Event	δ_g(State,Event)
56	1;14;15;1N;1N	t	2
		$f1:r:s$	62
		$f2:r:s$	63
57	6;2;2;24F2;24F2	t	64
58	19;7;10;42F3;42F3	$l:r:f2$	24
59	21;18;7;80F6;80F6	t	26
60	14;4;16;49F4;49F4	$f1:r:l$	65
61	15;15;4;62F5;62F5	$f1:r:l$	66
62	1;2;15;3N,23F2,47F4,60F5,73F6; 3N,23F2,47F4,60F5,73F6;	t	6
		$f2:r:s$	7
63	1;14;2;37F3;37F3	t	8
64	9;4;4;25F2;25F2	$f1:r:l$	67
65	14;6;16;51F4;51F4	$f2:r:l$	68
66	15;16;4;64F5;64F5	$f2:r:l$	69
67	9;6;4;27F2;27F2	$f2:r:l$	70
68	14;6;17;52F4;52F4	t	71
69	15;16;6;65F5;65F5	t	72
70	9;6;6;28F2;28F2	t	73
71	17;8;18;53F4;53F4	t	74
72	18;17;8;66F5;66F5	$l:r:f1$	75
73	13;8;8;29F2;29F2	$f2:r:f1$	76
74	19;10;10;54F4;54F4	$l:r:f2$	32
75	20;17;8;68F5;68F5	$f2:r:f1$	77
76	13;8;11;31F2;31F2	t	78
77	20;17;11;69F5;69F5	t	79
78	16;10;13;32F2	$f1:r:f2$	80
79	21;18;13;70F5;70F5	t	43
80	16;12;13;34F2	t	45

APPENDIX B

B.1 Computation of Equations (9.18), (9.19), and (9.20)

Assume that the three lowest-cost combinations of sensors in the set $N_i^{\overline{D}}$ (cf. (9.17)) are a, b, and d in that order, and a and b are combinations of i sensors while d is a combination of $i+1$ sensors. Since the cardinality of $N_i^{\overline{D}}$ is less than or equal to k we can apply the induction hypothesis to the set $N_i^{\overline{D}}$, i.e., we test next the element with the smallest cost. Therefore, we have

$$V(N_i^{\overline{D}}) \;=\; c + (1-p_i) * V(N_i^{\overline{D}} \backslash \{a\}). \tag{B.1}$$

By applying the induction hypothesis to $N_i^{\overline{D}} \backslash \{a\}$ in (B.1), we have

$$V(N_i^{\overline{D}}) \;=\; c + (1-p_i) * c + (1-p_i)^2 * V(N_i^{\overline{D}} \backslash \{a,b\}). \tag{B.2}$$

By applying the induction hypothesis to $N_i^{\overline{D}} \backslash \{a,b\}$ in (B.2), we have

$$\begin{aligned} V(N_i^{\overline{D}}) \;=\; & c + (1-p_i) * c + (1-p_i)^2 * c \\ & + (1-p_i)^2 * (1-p_j) * V(N_i^{\overline{D}} \backslash \{a,b,d\}). \end{aligned} \tag{B.3}$$

Finally, by successively applying the induction hypothesis until exhausting all elements in $N_i^{\overline{D}}$ we get (9.18). Following a similar procedure, one can verify (9.19) and (9.20).

B.2 Verification of (9.22): properties of reachable sets

The reachable sets from (9.17) are $N_i^{\overline{D}}$, N_j^D, and $N_j^{\overline{D}}$. The properties of these reachable sets are listed below (reproducing (9.22))

$$r_j < v_j \tag{B.4}$$

$$r_k = v_k \qquad k = i+1, \ldots, j-1 \tag{B.5}$$

$$r_i = v_i + 1 \tag{B.6}$$

$$s_k = v_k \qquad k = j+1, \ldots, L \tag{B.7}$$

$$s_j = v_j - 1 \tag{B.8}$$

$$s_k \leq v_k \qquad k = i+1, \ldots, j-1 \tag{B.9}$$

$$s_i \leq v_i + 1, \tag{B.10}$$

where v_k denotes the number of combinations of k sensors in $N_i^{\overline{D}}$, r_k denotes the number of combinations of k sensors in N_j^D, s_k denotes the number of combinations of k sensors in $N_j^{\overline{D}}$, and L denotes the highest set cardinality in N with $j \leq L \leq K-1$. We need to verify (B.5)–(B.10) since we verified (B.4) in Section 9.3.2.

From (9.17), the set $N_i^{\overline{D}}$ is equal to the set N without the element i. The set N_j^D includes all element in N except the elements with more than $|j|$ sensors and those with $|j|$ sensors but with strictly higher cost than j. The set $N_j^{\overline{D}}$ includes all elements in N except j and its children. The elements of $N_i^{\overline{D}}$ and N_j^D with more than $|i|$ sensors and less than $|j|$ sensors are the same, hence (B.5) is true. N_j^D has all elements in N with $|i|$ sensors while $N_i^{\overline{D}}$ includes the same elements with the exception of the element i, and that verifies (B.6). The sets $N_i^{\overline{D}}$ and $N_j^{\overline{D}}$ share the same elements with more than $|j|+1$ sensors, and that verifies (B.7). The only element of $|j|$ sensors missing from $N_j^{\overline{D}}$ and belonging to $N_i^{\overline{D}}$ is the element j, hence (B.8) is true. Inequality (B.9) is true since $N_i^{\overline{D}}$ contains all elements in N with more than $|i|$ sensors and less than $|j|$ sensors, while $N_j^{\overline{D}}$ contains the same elements with the exception of those that are children of the element j. The same argument applies to the elements with $|i|$ sensors with the possible addition of the element i, which does not belong to $N_i^{\overline{D}}$, to the set $N_j^{\overline{D}}$, and that verifies (B.10).

BIBLIOGRAPHY

[1] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard, "Fault detection and diagnosis in distributed systems: An approach by partially stochastic petri nets," *Journal of Discrete Event Dynamical Systems: Theory and Applications*, pp. 203–231, August 1998.

[2] R. J. Aumann, "Agreeing to disagree," *The Annals of Statistics*, vol. 4, no. 6, pp. 1236–1239, 1976.

[3] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella, "Diagnosis of large active systems," *Artificial Intelligence*, vol. 110, pp. 135–183, 1999.

[4] M. Basseville and I. Nikiforov, *Detection of Abrupt Changes – Theory and Application*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[5] S. Bavishi and E. Chong, "Automated fault diagnosis using a discrete event systems framework," in *Proc. 9th IEEE International Symposium on Intelligent Control*, pp. 213–218, 1994.

[6] D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[7] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Boston, MA, 1999.

[8] D. A. Castanon and D. Teneketzis, "Distributed estimation algorithms for nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 30, no. 5, pp. 418–425, May 1985.

[9] Y.-L. Chen and G. Provan, "Modeling and diagnosis of timed discrete event systems – a factory automation example," in *Proc. 1997 American Control Conference*, 1997.

[10] H. Darabi and M. Jafari, "Supervisory control and optimal partial observation problem,", Preprint, Rutgers University, 1998.

[11] R. Davis and W. Hamscher, "Model based reasoning: Troubleshooting," in *Readings in Model Based Diagnosis*, W. Hamscher, L. Console, and J. Kleer, editors, pp. 3–24, Morgan Kaufmann, 1992.

[12] S. Deb, A. Mathur, P. Willett, and K. Pattipati, "De-centralized real-time monitoring and diagnosis," in *Proc. IEEE Conf. on Systems, Man and Cybernetics*, pp. 2998–3003, October 1998.

[13] R. Debouk, S. Lafortune, and D. Teneketzis, "On an optimization problem in sensor-selection," Technical Report CGR99-04, Department of Electrical Engineering and Computer Science, University of Michigan, March 1999. Submitted to the Journal of Discrete Evet Dynamical Systems: Theory and Applications.

[14] R. Debouk, S. Lafortune, and D. Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete-event systems," *Journal of Discrete Event Dynamical Systems: Theory and Applications*, vol. 10, pp. 33–86, 2000.

[15] R. Debouk, S. Lafortune, and D. Teneketzis, "On the effect of communication delays in failure diagnosis of decentralized discrete-event systems," Technical Report CGR00-04, Department of Electrical Engineering and Computer Science, University of Michigan, May 2000.

[16] F. Eskafi, "A diagnostic system design for the intra-platoon communication system,", Preprint, California PATH, University of California, Berkeley, December 1997.

[17] E. Fabre, A. Benveniste, C. Jard, S. L. Ricker, and M. Smith, "Distributed state reconstruction for discrete event systems," in *Proc. 39th IEEE Conf. on Decision and Control*, Sydney, Australia, December 2000.

[18] B. Foreman, "A survey od wireless communication technologies for automated vehicle control," in *Proc. SAE Future Transportation Technology Conference*, August 1995.

[19] P. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge based redundancy," *Automatica*, vol. 26, pp. 459–474, 1990.

[20] A. Haji-Valizadeh and K. Loparo, "Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems," *IEEE Trans. Automat. Contr.*, vol. 41, no. 11, pp. 1579–1593, November 1996.

[21] L. Holloway and S. Chand, "Time templates for discrete event fault monitoring in manufacturing systems," in *Proc. 1994 American Control Conference*, pp. 701–706, 1994.

[22] J. Hopcroft and J. Ullman, *Introduction to automata theory, languages, and computation*, Addison Wesley, Reading, MA, USA, 1979.

[23] P. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adpative Control*, Prentice Hall, Englewood Cliffs, NJ, 1986.

[24] L. Telksnys, Editor, *Detection of Changes in Random Processes*, Optimization Software Inc. Publications Division, New York, NY, 1986.

[25] S. Lapp and G. Powers, "Computer-aided synthesis of fault trees," *IEEE Trans. Reliability Engineering*, vol. 26, no. 1, pp. 2–13, April 1977.

[26] F. Lees, "Process computer alarm and disturbance analysis: Review of the state of the art," *Computers and Chemical Engineering*, vol. 7, no. 6, pp. 669–694, 1983.

[27] F. Lin, "Diagnosability of discrete event systems and its application," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 4, no. 2, pp. 197–212, May 1994.

[28] S. Mohindra and P. Clark, "A distributed fault diagnosis method based on digraph models: Steady-state analysis," *Computers and Chemical Engineering*, vol. 17, no. 2, pp. 193–209, 1993.

[29] O. O. Oyeleye, F. Finch, and M. Kramer, "Qualitative modeling and fault diagnosis of dynamic processes by midas," in *AIChE Spring Annual Meeting*, 1989.

[30] Y. Pencolé, "Decentralized diagnoser approach: Application to telecommunication networks," in *Proc. of DX'2000, Eleventh International Workshop on Principles of Diagnosis*, pp. 185–192, June 2000.

[31] A. Pouliezos and G. Stavrakakis, *Real time fault monitoring of industrial processes*, Kluwer Academic Publishers, Boston, MA, 1994.

[32] G. Provan and Y.-L. Chen, "Model-based diagnosis and control reconfiguration for discrete event systems: An integrated approach," in *Proc. 38th IEEE Conf. on Decision and Control*, pp. 1762–1768, December 1999.

[33] M. L. Puterman, *Markov Decision Processes : Discrete Stochastic Dynamic Programming*, John Wiley & Sons, New York, NY, 1994.

[34] P. Ramadge and W. Wonham, "The control of discrete-event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[35] S. L. Ricker and E. Fabre, "On the construction of modular observers and diagnosers for discrete-event systems," in *Proc. 39th IEEE Conf. on Decision and Control*, Sydney, Australia, December 2000.

[36] L. Rosé and M.-O. Cordier, "Diagnosing discrete-event systems: An experiment in telecommunication networks," in *Proc. of WODES 1998, International Workshop on Discrete Event Systems*, pp. 130–137. Published by IEE, London, England, August 1998.

[37] M. Sampath. *The extended diagnoser*, November 1993. Unpublished memorandum.

[38] M. Sampath, *A Discrete Event Systems Approach to Failure Diagnosis*, PhD thesis, The University of Michigan, 1995. Available at `http://www.eecs.umich.edu/umdes`.

[39] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 43, no. 7, pp. 908–929, July 1998.

[40] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 40, no. 9, pp. 1555–1575, September 1995.

[41] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE Trans. Contr. Syst. Tech.*, vol. 4, no. 2, pp. 105–124, March 1996.

[42] W. Scherer and C. White, "A survey of expert systems for equipment maintenance and diagnostics," in *Fault Detection and Reliability: Knowledge Based & Other Approaches*, M. Singh, K. Hindi, G. Schmidt, and S. Tzafestas, editors, Pergamon Press, 1987.

[43] R. Sengupta, "Diagnosis and communication in distributed systems," in *Proc. of WODES 1998, International Workshop on Discrete Event Systems*, pp. 144–151. Published by IEE, London, England, August 1998.

[44] J. L. Speyer, "Computation and transmission requirements for a decentralized linear-quadratic-gaussian control problem," *IEEE Trans. Automat. Contr.*, vol. 24, pp. 266–269, April 1979.

[45] D. Swaroop, *String Stability of Interconnected Systems*, PhD thesis, The University of California, Berkeley, 1994.

[46] U. Schmid, Guest Editor, "Special issue on global time in large scale distributed real time systems," *Real Time Systems*, vol. 12, no. I, II, and III, pp. 1–351, Jan, Mar, and May 1997.

[47] N. Ulerich and G. Powers, "On-line hazard aversion and fault diagnosis in chemical processes: The digraph + fault-tree method," *IEEE Trans. Reliability Engineering*, vol. 37, no. 2, pp. 171–177, June 1988.

[48] N. Viswanadham, "Control systems: Reliability," in *Systems and Control Encyclopedia: Theory, Technology, Applications*, M. Singh, editor, Pergamon Press, 1997.

[49] N. Viswanadham and T. Johnson, "Fault detection and diagnosis of automated manufacturing systems," in *Proc. 27th IEEE Conf. on Decision and Control*, pp. 2301–2306, 1988.

[50] R. Vries, "An automated methodology for generating a fault tree," *IEEE Trans. Reliability Engineering*, 1990.

[51] W. Stark, Project Director, "Low energy electronics design for mobile platforms," Project Report of ARO-MURI for the period 9/96 to 7/99, Electrical Engineering and Computer Science Department, The University of Michigan, 1999.

[52] R. B. Washburn and D. Teneketzis, "Asymptotic agreement among communicating decision makers," *Stochastics*, vol. 13, no. 1–2, pp. 103–129, 1984.

[53] A. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, pp. 601–611, 1976.

[54] A. S. Willsky, M. Bello, D. A. Castanon, B. C. Levy, and G. Verghese, "Combining and updating of local estimates and regional maps along sets of one-dimensional tracks," *IEEE Trans. Automat. Contr.*, vol. 27, no. 4, pp. 799–813, August 1982.

[55] S. H. Zad, R. Kwong, and W. Wonham, "Fault diagnosis in discrete-event systems: Framework and model reduction," in *Proc. 37th IEEE Conf. on Decision and Control*, pp. 3769–3774, December 1998.

[56] S. H. Zad, R. Kwong, and W. Wonham, "Fault diagnosis in timed discrete-event systems," in *Proc. 38th IEEE Conf. on Decision and Control*, pp. 1756–1761, December 1999.

ABSTRACT

FAILURE DIAGNOSIS OF DECENTRALIZED DISCRETE EVENT SYSTEMS

by

Rami Ismail Debouk

Co-Chairs: Stéphane Lafortune and Demosthenis Teneketzis

We address the problem of failure diagnosis in discrete event systems with decentralized information. We propose a coordinated decentralized architecture consisting of local sites communicating with a coordinator that is responsible for diagnosing the failures occurring in the system. We extend the notion of diagnosability, originally introduced in Sampath et al. (IEEE Trans. on Automatic Control, Sep. 1995) for centralized systems, to the proposed coordinated decentralized architecture. We specify three protocols that realize the proposed architecture; each protocol is defined by the diagnostic information generated at the local sites, the communication rules used by the local sites, and the coordinator's decision rule. We analyze the diagnostic properties of each protocol. We state and prove conditions for a language to be diagnosable under each protocol. These conditions are verifiable off-line. The on-line diagnostic process is carried out using the diagnosers introduced in Sampath et al. or a slight variation of these diagnosers. The key features of the proposed protocols are: (i) they achieve, each under a set of assumptions, the same diagnostic performance as the centralized diagnoser; and (ii) they highlight the "performance vs. complexity" tradeoff that arises in coordinated decentralized architectures.

The correctness of two of the protocols relies on some stringent global ordering assumption on message reception at the coordinator's site, the relaxation of which is analyzed. We present an algorithm that attempts the ordering of messages at the coordinator site without the use of timing information. We prove that the algorithm may degrade the diagnostic performance of a protocol. Moreover, the implementation of the algorithm requires considerable additional memory and processing power at the coordinator site.

We also study an optimization problem in sensor selection that could be applied to the area of failure diagnosis. We formulate the sensor selection problem as a Markovian decision problem and identify one instance where the optimal solution can be analytically determined.